Martin O'Hanlon • David Whale

Nouvelle matière
du programme
du collège et du lycée

Apprendre à coder en PYTHON avec MINECRAFT

Dès 10 ans



Copyright © 2016 Eyrolles

EYROLLES

Apprendre à coder en PYTHON avec MINECRAFT

Et si tu programmais en t'amusant !

Cet ouvrage s'adresse à tous les jeunes qui aiment jouer à Minecraft et qui souhaitent s'initier à la programmation pour aller plus loin. Et quitte à découvrir le code, autant s'initier à Python, un langage puissant, facile à assimiler et amusant.

En programmant dans Minecraft, tu pourras rendre tes aventures encore plus passionnantes, originales et personnelles. Tu détourneras en outre des éléments du jeu pour les faire agir de façon totalement inédite, voire en inventer de nouveaux auxquels même les créateurs du jeu n'avaient pas songé.

Au fil de ta lecture, tu verras entre autres comment :

- écrire des programmes en Python sur ton Mac, PC ou Raspberry Pi ;
- créer des maisons, des structures et fabriquer une machine à dupliquer des éléments du jeu ;
- interagir avec le jeu à l'aide de circuits électroniques très simples ;
- créer des objets intelligents et coder un programme d'invasion alien ;
- concevoir d'impressionnantes structures 2D et 3D comme des sphères et des pyramides ;
- imaginer et développer ton propre mini-jeu interactif dans Minecraft.

Tu as peut-être déjà atteint un niveau expert dans le jeu, mais tu te sens limité par le temps que tu passes à bâtir de nouvelles structures. Ou peut-être souhaites-tu trouver un moyen d'augmenter encore les capacités du jeu en y ajoutant des fonctionnalités intelligentes et d'automatisation. Quelles que soient tes raisons, ce livre t'accompagnera tout au long de tes aventures de programmation dans Minecraft.

À propos des auteurs

Martin O'Hanlon a consacré sa carrière à la conception et à la programmation de systèmes informatiques. Sa passion pour la programmation et l'enseignement l'ont amené à créer le blog Stuffaboutcode.com où il partage ses expériences, ses connaissances et ses idées. David Whale a créé quant à lui un atelier de programmation Minecraft et Raspberry Pi. Il anime le site : blog.whaleygeek.co.uk.

À qui s'adresse cet ouvrage ?

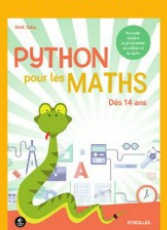
Aux collégiens, lycéens, parents, enseignants et associations.

Sur www.editions-eyrolles.com/go/pythonminecraft

Télécharge les kits de démarrage PC/Mac et le code source des exemples du livre.



Dans la même collection







Copyright © 2016 Eyrolles.

www.editions-eyrolles.com

Apprendre à coder en PYTHON avec MINECRAFT

Copyright © 2016 Eyrolles.

Copyright © 2016 Eyrolles.

Martin O'Hanlon

David Whale

Apprendre à coder en PYTHON avec MINECRAFT

Copyright © 2016 Eyrolles.

EYROLLES

ÉDITIONS EYROLLES
61, bd Saint-Germain
75240 Paris Cedex 05
www.editions-eyrolles.com

© 2014 John Wiley and Sons, Ltd.

All Rights Reserved. Authorised translation from the English language edition published by John Wiley & Sons Limited. Responsibility for the accuracy of the translation rests solely with Groupe Eyrolles and is not the responsibility of John Wiley & Sons Limited. No part of this book may be reproduced in any form without the written permission of the original copyright holder John Wiley & Sons Limited.

Traduction autorisée de l'ouvrage en langue anglaise intitulé *Adventures in Minecraft* par David Whale et Martin O'Hanlon (ISBN 978-1-118-94691-6), publié par John Wiley & Sons.

Traduit et adapté de l'anglais par Lara El Keilany.

Mis en pages par Sandrine Escobar.

En application de la loi du 11 mars 1957, il est interdit de reproduire intégralement ou partiellement le présent ouvrage, sur quelque support que ce soit, sans autorisation de l'éditeur ou du Centre français d'exploitation du droit de copie, 20, rue des Grands-Augustins, 75006 Paris.

© Groupe Eyrolles, 2016, ISBN : 978-2-212-14292-1

Copyright © 2016 Eyrolles.

À ma femme Léonie. Sans toi, tout cela n'existerait pas.

Martin

À ma femme Gail, pour sa patience
lorsque je passais mes journées à jouer à Minecraft.

David

À propos des auteurs

Martin O'Hanlon a consacré sa carrière à la conception et à la programmation de systèmes informatiques. Sa passion pour la programmation et l'enseignement l'ont amené à créer le blog <Stuff about="code" /> (`www.stuffaboutcode.com`) où il partage ses expériences, ses connaissances et ses idées. Martin anime régulièrement des présentations et des ateliers sur la programmation dans Minecraft. Il s'adresse à un public de codeurs, d'enseignants et de jeunes, et les invite à découvrir des nouveautés tout en faisant de la programmation une activité ludique.

David Whale s'amuse à détourner des objets du quotidien en rédigeant des programmes informatiques, pour des résultats des plus inattendus ! Mordu de programmation depuis l'âge de 11 ans, il continue aujourd'hui d'inventer des logiciels et aime enseigner la programmation. Il dirige une entreprise de conseil en informatique dans le comté d'Essex, au Royaume-Uni. Il travaille également en tant que bénévole pour l'Institute of Engineering and Technology (l'IET), que ce soit en cours, lors des clubs informatiques du week-end ou pour animer des ateliers de programmation destinés aux jeunes dans tout le pays. Vous pouvez suivre ses aventures sur son blog :

`http://blog.whaleygeek.co.uk`.

Copyright © 2016 Eyrolles.

Remerciements

Nombreux sont ceux qui participent à la publication d'un livre. Bien qu'ils soient trop nombreux ici pour les citer tous, nous tenions tout de même à remercier les personnes suivantes.

- Tout le personnel de Mojang, pour avoir conçu ce formidable jeu, pour leur génie et leur savoir-faire qui ont permis de rendre ce jeu programmable. Sans leurs connaissances, ce livre n'aurait simplement pas vu le jour.
- La Fondation Raspberry Pi et toute la communauté open source, sans qui Raspberry Pi n'existerait pas, ni même le serveur Bukkit. Or, ces deux plates-formes sont indispensables, car elles ont permis d'écrire ce livre pour le grand public.
- Nos testeurs et jeunes experts de Minecraft, Zachary Igielman, Lauren Trussler, Sam Whale, Ben Foden et Ria Parish, qui ont testé nos programmes et ont fourni des retours vraiment utiles. Nous n'en aurions probablement jamais bénéficié si nous avions uniquement présenté ce livre à des jeunes du public visé.
- M. Sukkin Pang, pour ses conseils avisés, pour son aide précieuse dans le choix des dispositifs électroniques adéquats pour nos projets et pour nous avoir aidés à contrôler des circuits de PC et de Mac le plus simplement du monde sans trop nous ruiner.
- Cliff O'Reilly, pour s'être assuré que tout fonctionnait correctement et pour avoir tout testé trois fois à notre place (un test à chaque fois sur chacun des systèmes d'exploitation).
- Sarah Wright, pour ses illustrations géniales. Ce sont des images magnifiques qui saisissent parfaitement le sens des concepts abordés dans chaque chapitre.
- Ben Ramachandra, le jeune garçon qui a participé à l'événement Fire Tech Camp de l'Imperial College London à Noël, en 2013. Ton enthousiasme était tel pendant les cours de Python pour Minecraft que c'est grâce à toi que nous avons eu l'idée de ce livre !
- Roma Agrawal, ingénieure en structures qui a participé à la conception du gratte-ciel The Shard, au Royaume-Uni, pour ses suggestions et ses remarquables exemples de grands bâtiments dans le chapitre 4 et les bonus. Nous espérons qu'ils inspireront les plus belles créations aux joueurs !
- Enfin, nous tenons à remercier Carrie-Anne Philbin d'avoir eu l'audace et la détermination d'écrire son premier livre *Adventures in Raspberry Pi*, sans lequel la suite d'ouvrages du même nom n'aurait pas vu le jour. Regarde un peu de quoi tu es responsable, Carrie-Anne !

Copyright © 2016 Eyrolles.

Table des matières

Avant-propos 1

Introduction 3

Minecraft, qu'est-ce que c'est ? 3
Le monde virtuel 4
Comment l'idée de Minecraft est-elle née ? 4
Programmer dans Minecraft, qu'est-ce que ça veut dire ? 5
À qui ce livre s'adresse-t-il ? 6
Que vas-tu apprendre dans ce livre ? 6
Nous supposons que tu sais déjà... 7
Ce dont tu auras besoin pour les projets 8
Remarque à l'attention des parents et des enseignants 9
Structure du livre 9
Le site Internet associé 11
Autres ressources 11
Conventions de lecture 12
Pour aller plus loin 14

Chapitre 1

Bonjour le monde de Minecraft 15

Configurer ton Raspberry Pi pour programmer dans Minecraft 17
Installer Minecraft sur Raspberry Pi 18
Démarrer Minecraft sur Raspberry Pi 19
Configurer ton PC ou ton Mac pour programmer dans Minecraft 21
Installer le kit de démarrage et Python sur ton PC Windows 22
Installer le kit de démarrage et Python sur ton Mac 24
Démarrer Minecraft sur ton PC Windows ou Mac 26
Arrêter Bukkit 29
Créer un programme 30
Exécuter un programme 33
Arrêter un programme 35

Copyright © 2016 Eyrolles.

Chapitre 2

Détecter les déplacements de ton joueur 37

Localiser ton joueur 38
Te préparer pour l'aventure 40
Afficher la position de ton joueur 40
Améliorer l'affichage de ta position 43
Utiliser postToChat pour afficher ta position ailleurs 45
Introduire une boucle de jeu 45
Bâtir le jeu Bienvenue à la maison 48
Utiliser des instructions if pour créer un paillasson magique 48
Connaître la position précise de ton joueur 49
Construire un paillasson magique 51
Écrire le jeu Bienvenue à la maison 51
Utiliser le géorepérage pour prélever un impôt 55
Enregistrer les coordonnées des angles de ton enclos 57
Écrire le programme de géorepérage 59
Déplacer ton joueur 61
Explorer d'autres pistes de localisation 64

Chapitre 3

Automatiser tes constructions 67

Créer des blocs 68
Construire plus d'un bloc à la fois 70
Utiliser des boucles for 71
Créer plusieurs blocs à l'aide d'une boucle for 72
Construire une immense tour à l'aide d'une boucle for 73
Faire de la place 75
Utiliser setBlocks pour construire encore plus vite 75
Saisir des données depuis le clavier 77
Construire une maison 79
Construire plusieurs maisons 84
Utiliser les fonctions Python 84
Construire une rue bordée de maisons à l'aide d'une boucle for 88
Poser des moquettes de couleurs différentes de manière aléatoire 90
Produire des nombres aléatoires 91
Poser les moquettes 92
Explorer d'autres pistes de construction 95

Copyright © 2016 Eyrolles.

Chapitre 4

Interagir avec les blocs 97

Identifier les blocs sur lesquels tu te déplaces 98
Comment savoir si tes pieds touchent la terre ferme 99
Construire des ponts magiques 101
Utiliser les listes Python en guise de mémoire magique 104
Essayer les listes Python 104
Construire des ponts éphémères à l'aide d'une liste Python 107
Détecter les blocs que ton joueur frappe 112
Créer un jeu de chasse au trésor 115
Rédiger les fonctions et la boucle de jeu principale 116
Semer des trésors dans le ciel 117
Récupérer un trésor une fois qu'il a été touché 118
Ajouter un signal lumineux 119
Ajouter ton constructeur de ponts 120
Explorer d'autres pistes d'interaction 122

Chapitre 5

Interagir avec des circuits électroniques . . 125

Ce dont tu as besoin dans ce chapitre 126
Créer un prototype à l'aide d'une plaque d'essais 129
Élaborer un circuit permettant d'allumer une LED 131
Connecter des composants électroniques à ton ordinateur 133
Paramétrer un PC ou un Mac pour contrôler des circuits électroniques 134
Configurer les pilotes 134
Obtenir le numéro du port série 135
Contrôler une LED 137
Allumer une LED avec un ordinateur 138
Faire clignoter la LED 141
Exécuter un programme GPIO 143
Rédiger le programme de paillasson magique avec une LED 146
Utiliser un afficheur 7 segments 147
Qu'est-ce qu'un afficheur 7 segments ? 147
Brancher l'afficheur 7 segments 149
Programmer en Python pour contrôler l'afficheur 7 segments 151
Utiliser un module Python pour contrôler l'afficheur 153
Créer un détonateur 155
Réaliser le câblage d'un bouton 155
Rédiger le programme du détonateur 158
Explorer d'autres pistes de programmation impliquant des circuits électroniques . . 162

Copyright © 2016 Eyrolles.

Chapitre 6

Utiliser des fichiers de données 165

Lire une donnée dans un fichier 166
L'intérêt des fichiers de données 166
Créer un donneur d'indices.....167
Construire des labyrinthes à l'aide d'un fichier de données 170
Comprendre les fichiers CSV 171
Construire un labyrinthe.....173
Construire une imprimante de blocs en 3D 179
Fabriquer un petit objet de test pour imprimer en 3D.....179
Rédiger le code de l'imprimante 3D.....182
Construire un scanneur de blocs en 3D 185
Construire une machine à dupliquer 189
Rédiger la structure du programme de la machine à dupliquer.....190
Afficher le menu 194
Bâtir la salle de reproduction 195
Démolir la salle de reproduction 197
Scanner dans la salle de reproduction.....198
Nettoyer la salle de reproduction.....199
Imprimer dans la salle de reproduction 199
Créer un répertoire de fichiers 200
Explorer d'autres pistes avec les fichiers de données 203

Chapitre 7

Construire des structures en 2D et en 3D 205

Le module minecraftstuff.....206
Créer des segments, des cercles et des sphères.....206
Tracer des segments.....208
Dessiner des cercles 210
Dessiner des sphères 211
Créer une horloge dans Minecraft 212
Dessiner des polygones 218
Pyramides.....221
Explorer d'autres pistes avec les objets en 2D et en 3D.....225

Copyright © 2016 Eyrolles.

Chapitre 8
Créer des blocs intelligents 227
Utiliser les nombres aléatoires pour rendre ton ami plus intéressant 234
Personnaliser un objet plus gros 237
Invasion d'extraterrestres 240
Explorer d'autres pistes dans le domaine de la simulation 247

Chapitre 9
La grande aventure 249
Un jeu dans le jeu 249
Première partie : construire l'arène 251
Deuxième partie : placer les obstacles 255
Le mur 255
Programmer plusieurs obstacles en même temps 258
Creuser le lit de la rivière 261
Creuser les trous 264
Troisième partie : la jouabilité 268
Démarrer le jeu 268
Récupérer des diamants 270
Plus une seconde à perdre ! 273
Localiser le joueur 274
Indiquer la fin d'un niveau de jeu et calculer des points 275
Afficher le message « Game over » 277
Quatrième partie : ajouter un bouton-poussoir et un afficheur 278
Ce dont tu as besoin 278
Préparer ton matériel 279
Faire le décompte des diamants 280
Indiquer le temps restant 281
Continuer l'aventure dans Minecraft et explorer les pistes qui s'offrent à toi ... 282

Annexe A
Pour aller plus loin 285
Sites Internet 285
Minecraft 285
Python 287
Bukkit 287
Autres techniques et astuces d'automatisation 288

Copyright © 2016 Eyrolles.

Projets et tutoriels 289
Vidéos 290
Livres 291

Index 293

Chapitre 10 bonus *(en ligne)*
L'ascenseur Minecraft 299

Ce dont tu as besoin 300
Construire l'ascenseur Minecraft 300
Compléter la liste des composants électroniques pour l'ascenseur 300
Écrire la structure du programme 303
Écrire la boucle de jeu principale 306
Mettre l'ascenseur en service 308
Créer l'ascenseur 308
Détecter le déclenchement des boutons d'appel électroniques 309
Utiliser un mode de service pour activer l'ascenseur manuellement 311
Afficher le numéro des étages 312
Automatiser l'ascenseur 314
Les fonctions marche et arrêt 314
Achever le panneau de contrôle de l'ascenseur 315
Faire fonctionner l'ascenseur lorsque le joueur se trouve à l'extérieur 316
Détecter lorsque le joueur se trouve dans l'ascenseur 317
Détecter lorsque ton joueur frappe un bloc pour appeler l'ascenseur 318
Explorer d'autres pistes avec l'ascenseur Minecraft 320

Annexe B *(en ligne)*
Tableau de références 321

Glossaire *(en ligne)* 331

Index *(en ligne)* 337

Copyright © 2016 Eyrolles.

Avant-propos

J'ai toujours aimé consacrer ma pause déjeuner aux élèves pour leur permettre de terminer leurs devoirs, faire des recherches sur Internet, envoyer des e-mails à leurs professeurs ou encore imprimer leurs exercices. Lorsque, comme à mon habitude, je parcoure les allées d'ordinateurs pendant l'heure du déjeuner au club informatique, je peux voir à quel point les élèves mettent du cœur à l'ouvrage : propulsion d'oiseaux multicolores, combats contre des extraterrestres ou tentative de marche arrière dans un parking… Reconnaissons au passage l'intérêt pédagogique de ce dernier exercice !

Plus sérieusement, je pense que plus les enfants passent du temps sur les ordinateurs, plus leur utilisation devient naturelle. Un simple jeu permet à lui seul d'améliorer la coordination des yeux et des mains et de familiariser l'enfant avec le clavier. Il suffit de voir comme, en un clin d'œil, ils parviennent à se connecter, ouvrir le navigateur et saisir des séries de mots-clés les plus courtes possibles dans un moteur de recherche. J'ai en général à peine le temps d'ouvrir la porte de la salle informatique que les élèves se sont déjà installés et ont démarré un jeu, les yeux rivés sur leur écran.

Un jour, alors que je faisais ma ronde habituelle dans les allées dans l'espoir de trouver l'exception qui se consacrerait à une activité productive, je suis tombé sur un élève occupé à empiler assidûment ce qui ressemblait à des cubes en 3D dans un jeu. Il paraissait complètement absorbé. Je me suis alors arrêté, curieux de comprendre pourquoi cette tâche, pourtant si simple, retenait toute son attention. C'est alors que j'ai vu apparaître une maison qu'il avait créée en empilant tous ces blocs.

C'est comme ça que j'ai connu Minecraft, grâce à ce petit architecte en herbe. Cet élève avait la capacité de s'évader, imperturbable, dans le chaos d'une salle d'informatique à la pause déjeuner et de disparaître derrière un monde qu'il avait lui-même créé ; une maison virtuelle qu'il avait conçue, construite et parcourue en long et en large. Les jeux auxquels s'adonnent les enfants sont des choses extrêmement contagieuses. Une fois qu'ils ont touché un public assez large, une fois qu'assez d'enfants ont mordu à l'hameçon, ils s'y mettent tous sans exception. Et plus le jeu prend de l'impulsion, plus vite les enfants vont s'y atteler pour être en avance sur les autres. C'est ainsi qu'en seulement un été, Minecraft les avait déjà tous conquis.

Cependant, contrairement aux autres modes passagères, je me suis vite aperçu que ce jeu sortait du lot. J'ai découvert que les élèves pouvaient automatiser leurs constructions, concevoir les systèmes électriques de leurs maisons, les éclairer à leur guise et y installer des ascenseurs. Ils pouvaient même échafauder des escaliers dissimulés entre les murs. En bref, j'ai compris qu'ils pouvaient créer presque tout ce qu'ils voulaient. Tandis que le cours d'histoire se prolonge à l'heure du déjeuner devant les ordinateurs, où certains construisent les châteaux qu'ils viennent d'étudier, d'autres se disputent la création du plus beau squelette vu en cours de biologie, jusqu'à ce que le cours de géographie fasse irruption et que le volcan qu'ils ont conçu déverse sa lave et détruise tout sur son passage, laissant partir tout leur travail en fumée.

Lorsque l'école a fait l'acquisition d'un compte MinecraftEdu, mes élèves se sont mis à collaborer. Ensemble, ils ont reproduit le logo de l'école en grand, construit un

Copyright © 2016 Eyrolles.

labyrinthe et organisé un concours pour en sortir. Ils ont pris les mesures des salles de cours et ont bâti une réplique de l'école dans Minecraft.

Plus important encore, cela m'a ouvert les portes d'une classe virtuelle. Un lieu fantastique où je survole les avatars de mes élèves et leur enseigne tous les secrets de la logique. J'ai commencé par construire une porte OU exclusif en utilisant des blocs. Et j'ai ensuite demandé aux élèves de former des groupes pour élaborer une calculatrice en reliant plusieurs circuits.

À ma grande surprise, j'ai découvert que Minecraft existait aussi sur Raspberry Pi. Mais bien mieux que cela encore, j'ai appris qu'il était possible d'utiliser le langage de programmation Python pour y jouer. Mes élèves s'y sont aussitôt mis. À l'aide de boucles `for` et en disposant des blocs, nous avons créé des plates-formes et des trampolines, des boules de feu et des dragsters ; des jeux dans le jeu dont les capacités dépassent même l'imagination de leurs concepteurs. Les élèves se sont totalement approprié ce monde virtuel.

L'un d'eux est même parvenu à inventer par ses propres moyens un script qui permet de construire des maisons autour du joueur et un autre a élaboré un jeu de démineur. Ils se sont tous mis à vouloir apprendre la programmation, mus par le désir de créer et d'automatiser des tâches encore plus complexes au sein de Minecraft et de rivaliser ainsi avec les inventions de leurs camarades.

Aujourd'hui, notre école est équipée d'un réseau Minecraft personnalisé, doté de ses propres serveurs, construit par les élèves et pour les élèves. En s'appuyant sur leur réseau, ils peuvent recevoir des notes sur le travail qu'ils ont soumis, exactement comme s'il s'agissait d'évaluer leurs devoirs à la maison. Ils peuvent se lancer des défis entre eux et coder des pans entiers d'un monde, puis partir à leur découverte ensemble.

Ce livre est un premier pas dans l'exploration de ce monde incroyable. Il enseigne le code, et il le fait en utilisant l'un des jeux les plus populaires de la planète ! Petit à petit, il invite les enfants à sortir du monde de Minecraft pour entrer dans le monde de la programmation informatique. L'objectif à atteindre est concret et cela les motive à apprendre le plus vite possible.

Ce qui rend ce jeu si puissant, c'est qu'il offre aux joueurs la possibilité de visualiser instantanément le résultat de leur code. La programmation est une discipline qui parfois s'avère fastidieuse et inintéressante pour les enfants, qui ne parviennent pas toujours à se figurer le résultat de leur code lorsqu'il apparaît à l'écran comme un simple texte. Mais grâce à Minecraft, ils visualisent ces résultats dans un contexte qu'ils comprennent et qui les divertit.

Je suis d'ailleurs convaincu que Minecraft a la capacité d'influencer toute une génération et de montrer à quel point la programmation est une discipline amusante et enrichissante. Alors, ne perdez pas une minute de plus et faites le plein d'aventures dans Minecraft en suivant les conseils avisés de Martin et de David tout au long de votre périple. Laissez parler votre créativité, soyez enthousiaste, construisez des structures fascinantes et surtout, apprenez à coder en chemin !

Ben Smith BSc (Hons), Directeur informatique,
ArnoldKEQMS, Lytham St. Annes.

Copyright © 2016 Eyrolles.

Introduction

ES-TU UN aventurier ? Aimes-tu essayer et apprendre de nouvelles choses ? Es-tu un fan invétéré de Minecraft ? Souhaites-tu repousser les limites de ta connaissance de ce jeu en apprenant à programmer ? Veux-tu épater tes amis par ta créativité et tes tours de magie ? Si ta réponse est un « oui » sonnant et trébuchant, alors ce livre est fait pour toi.



Minecraft, qu'est-ce que c'est ?

Minecraft est un jeu indépendant de type bac à sable dans lequel tu peux bâtir des structures, recueillir des éléments, partir à la recherche de minerais et combattre des monstres pour survivre. Il t'apparaît sous la forme d'un monde virtuel en 3D fait de différents types de blocs et dans lequel chaque bloc a sa propre place au sein du maillage 3D. La figure I-1 montre à quoi ressemble le monde de Minecraft.



FIGURE I-1 Le monde de Minecraft

Copyright © 2016 Eyrolles.

Le monde virtuel

Dans les jeux de type bac à sable, tu évolues dans un monde virtuel. C'est comme si tu étais dans un bac à sable avec des bords très éloignés ; comme un immense parc rempli de sable. Au lieu de passer des niveaux dans un ordre préétabli, tu erres dans le monde virtuel et choisis les objectifs que tu souhaites atteindre et la manière dont tu veux t'y attaquer. Comme tu fais tes propres choix dès le début, ce type de jeux renferme d'infinies possibilités. Tu crées des histoires bien à toi et tu te déplaces dans le monde 3D en acquérant de nouvelles compétences et de nouvelles fonctionnalités que tu découvres par hasard au fil de tes expériences.

Dans Minecraft, ton joueur ou avatar s'appelle Steve. Tu le guides à travers le monde virtuel pour poursuivre les missions que tu t'es fixées. Si tu parviens à survivre la première nuit à l'assaut des monstres, tu peux continuer à poursuivre tes missions passionnantes pour interagir avec d'autres joueurs et bâtir d'immenses structures qui n'ont pour limites que celles de ton imagination.

Ainsi donc, dans un jeu de type bac à sable, tu es toi-même maître de tes décisions et tu n'as pas à suivre un chemin défini par les concepteurs du jeu. Pour en savoir plus sur ce type de jeux, rends-toi sur la page suivante : `https://fr.wikipedia.org/wiki/Monde_ouvert`. Quant au prénom du joueur, nous ne savons pas très bien pourquoi les concepteurs ont choisi Steve, mais tu peux tenter d'élucider ce mystère en suivant ce lien : `http://minecraft-fr.gamepedia.com/Joueur`.

Comment l'idée de Minecraft est-elle née ?

Les jeux indépendants sont des jeux vidéo créés par des individus ou par de petites équipes de programmeurs. Ils sont souvent développés sans aide financière ni soutien de la part d'éditeurs de jeux. Mais, par leur nature indépendante, ils s'avèrent souvent plus innovants que les autres et plaisent davantage au grand public. D'après Wikipédia, Minecraft a été créé par le programmeur suédois Markus Persson, plus connu sous le pseudonyme de « Notch », qui en a tout d'abord publié une version bêta en 2009. La première publication officielle du jeu a eu lieu en 2011. Notch a fondé une entreprise suédoise baptisée Mojang AB qui continue de développer son jeu sur plusieurs plates-formes informatiques, notamment sur PC, Mac, Raspberry Pi, Linux, iOS, Android, Xbox 360, PlayStation et Wii.

Pour en savoir plus sur l'histoire fantastique de Minecraft, tu peux voir le documentaire qui lui est dédié (en anglais) : *Minecraft: The Story of Mojang* (`http://en.wikipedia.org/wiki/Minecraft:_The_Story_of_Mojang`).

Copyright © 2016 Eyrolles.

Programmer dans Minecraft, qu'est-ce que ça veut dire ?

Ce livre traite de la programmation et utilise Minecraft à des fins pédagogiques. Si tu cherches des conseils utiles sur la construction de structures et sur les techniques de combat, il existe d'autres ouvrages intéressants en vente sur le marché. Nous les avons listés dans l'annexe A ; ils pourront t'aider.

En programmant dans Minecraft, tu peux rendre tes aventures encore plus passionnantes, originales et personnelles. Si tu optes pour le jeu normal, tu suivras les règles fondamentales de Minecraft telles qu'elles ont été établies par les concepteurs du jeu. Mais si tu rédiges des programmes qui peuvent interagir avec le monde de Minecraft, tu seras en mesure d'exécuter automatiquement des tâches complexes et répétitives, comme la construction d'avenues entièrement bordées de maisons et de structures gigantesques. Tu pourras en outre détourner des éléments du jeu pour les faire agir de façon totalement inédite, voire en inventer de nouveaux auxquels même les créateurs du jeu n'avaient pas songé. Et surtout, tu apprendras à maîtriser une discipline importante, à savoir la programmation en langage Python. Tu seras alors capable de mettre tes connaissances en pratique dans bien des domaines, en dehors de Minecraft. La figure I-2 représente une immense avenue bordée de maisons qui ont été construites automatiquement grâce à un court programme en Python.



FIGURE I-2 Une immense avenue bordée de maisons qui ont été construites grâce à un programme de 20 lignes en Python

Dans une vidéo récente sur l'importance de l'apprentissage de la programmation chez les enfants (`www.youtube.com/watch?v=nKIu9yen5nc`, en anglais), Will.i.am déclare que « les grands programmeurs sont les stars d'aujourd'hui ». En suivant les

Copyright © 2016 Eyrolles.

chapitres de ce livre, tu intégreras de nouvelles connaissances qui te feront découvrir un jeu davantage personnel, original et ambitieux. Grâce à tes tours de magie, tu impressionneras à coup sûr tes amis et tes compagnons de jeu, qui s'empresseront alors de t'interroger sur les secrets de tes prouesses. La réponse, évidemment, c'est la programmation !

À qui ce livre s'adresse-t-il ?

Apprendre à coder en Python avec Minecraft s'adresse à tous les jeunes qui aiment jouer à Minecraft et qui souhaiteraient apprendre la programmation pour aller plus loin. Si les premiers chapitres de cet ouvrage ont été soumis à des lecteurs de 8 ans, les lecteurs âgés de 11 à 15 ans seront toutefois un peu plus à l'aise. Les chapitres les plus difficiles qui se trouvent à la fin de l'ouvrage peuvent également convenir à un public plus âgé.

Tu as peut-être déjà atteint un niveau expert dans le jeu, mais tu te sens limité par le temps que tu passes à bâtir de nouvelles structures. Ou peut-être souhaites-tu trouver un moyen d'augmenter encore les capacités du jeu en y ajoutant des fonctionnalités intelligentes et d'automatisation. Quelles que soient tes raisons, ce livre t'accompagnera tout au long de tes aventures de programmation dans Minecraft. Et comme tout bon aventurier, les livres sont les outils les plus indispensables à emporter dans ta besace. Ce trek de longue haleine commencera par des tâches relativement simples, telles que la rédaction de messages dans le chat du jeu. Tu feras ensuite une escale pour apprendre les bases de la programmation dans Minecraft en langage Python. Puis tu finiras en totale autonomie, en utilisant tes nouvelles compétences de programmation pour créer ton propre jeu au sein du monde de Minecraft. Et lorsque ton périple d'apprentissage touchera à sa fin, tu auras tous les outils en poche pour devenir un pionnier de la programmation dans Minecraft !

Que vas-tu apprendre dans ce livre ?

Ce livre t'invite à découvrir divers aspects du jeu Minecraft et à interagir avec ses fonctionnalités en utilisant le langage de programmation Python. Tu apprendras notamment à repérer des blocs dans un monde en 3D en utilisant leurs coordonnées, à localiser ton joueur, à créer et à briser des blocs dans le monde virtuel et à détecter le fait qu'un bloc a été brisé par ton joueur.

Si tu utilises un système Raspberry Pi, tu apprendras à installer et à utiliser l'interface de programmation de Minecraft, livrée avec l'édition Minecraft Pi. Si tu utilises un PC ou un Mac, ce livre t'indiquera comment configurer et faire fonctionner ton serveur Minecraft en utilisant le serveur CraftBukkit que la communauté a développé. Tu apprendras également à le programmer en utilisant la même interface de programmation Minecraft que celle du Raspberry Pi en utilisant le module Raspberry Pi Juice.

Copyright © 2016 Eyrolles.

Tu apprendras à écrire des programmes en langage Python. Tu commenceras par des programmes aussi simples que le « Hello Minecraft World » (« Bonjour le monde de Minecraft »). Et tu finiras par créer et interagir avec de gigantesques objets en 3D, que tu pourras personnaliser à ta guise grâce à tes nouvelles connaissances.

En utilisant le module gratuit MinecraftStuff d'aide à la programmation en Python, tu auras en plus la possibilité d'améliorer tes créations 2D et 3D avec des blocs, des lignes, des polygones et du texte.

Mais tes aventures ne se limiteront pas seulement au monde virtuel de Minecraft ! Nous t'enseignerons en plus des techniques pour connecter Minecraft à des composants électroniques. Ainsi, tu seras capable de détecter et de contrôler des objets du monde réel depuis Minecraft. Nous te donnerons la clé d'un trésor bien gardé, à savoir : briser les frontières entre le monde virtuel et le monde réel !



Il existe deux modes de jeu : le mode Survie et le mode Créatif. Tu utiliseras le mode Créatif tout au long du livre. Nous n'aborderons pas le mode Survie, pour la simple et bonne raison qu'il est extrêmement frustrant de voir un creeper (créature qui explose, au bout d'une seconde et demie, quand le joueur se trouve à proximité) te tuer alors même que tu viens d'exécuter l'un de tes programmes. Il existe déjà beaucoup de livres intéressants en vente qui expliquent comment survivre une fois la nuit tombée dans Minecraft. Nous te fournissons d'ailleurs des liens vers ces ouvrages et vers d'autres ressources utiles dans l'annexe A, qui se trouve à la fin du livre. Toutefois, tous les programmes que tu créeras dans le mode Créatif te serviront également dans le mode Survie.

Nous supposons que tu sais déjà...

Comme cet ouvrage traite de programmation dans Minecraft et que nous souhaitons nous concentrer sur cet aspect-là, nous supposons que tu possèdes déjà certaines connaissances et le matériel indiqué ci-dessous.

1. Tu disposes d'un ordinateur déjà configuré (un Raspberry Pi fonctionnant sous Raspbian, un PC fonctionnant sous Windows ou un Mac fonctionnant sous OS X), qui répond aux exigences minimales pour jouer à Minecraft et qui est en bon état de marche.
2. Tu sais utiliser ton ordinateur, un clavier et une souris. Tu sais lancer des programmes depuis le menu du système et tu es capable de manipuler les différents éléments du menu d'une application, tels que `Fichier>Nouveau>Enregistrer`.
3. Tu es équipé d'une connexion Internet qui fonctionne correctement et tu sais utiliser un navigateur web pour télécharger des fichiers.

Copyright © 2016 Eyrolles.

4. Si tu utilises un PC ou un Mac, tu disposes déjà d'un identifiant Minecraft et tu as installé le jeu sur ton ordinateur.

5. Tu sais jouer à Minecraft, et notamment, tu sais le démarrer, te déplacer dans le jeu, choisir des éléments dans l'inventaire et créer ou briser des blocs dans le monde.

Comme ce livre traite de la programmation dans Minecraft, nous ne supposons pas que tu détiens des connaissances en programmation. Chapitre après chapitre, nous t'indiquerons la marche à suivre pour apprendre à programmer.

Ce dont tu auras besoin pour les projets

Ce livre est écrit pour une utilisation de Minecraft sur les trois systèmes d'exploitation suivants : Raspbian pour Raspberry Pi, Windows pour PC et OS X pour Mac. Minecraft est également compatible avec d'autres systèmes, dont plusieurs distributions de Linux pour PC, mais il n'en sera pas question dans ce livre.

Pour simplifier la configuration du jeu sur les différents systèmes, nous avons élaboré trois kits de démarrage, un pour chaque système. Tu peux télécharger le kit qui te convient directement sur le site Internet des éditions Eyrolles (`www.editions-eyrolles.com/dl/14292`). Le premier chapitre t'indiquera pas à pas comment le télécharger, l'installer et le lancer. Ces kits de démarrage contiennent tout ce dont tu auras besoin pour être prêt en un rien de temps, à l'exception du jeu Minecraft lui-même !

Afin de pouvoir télécharger ton kit, assure-toi de disposer d'une connexion Internet sur ton ordinateur. Tu y trouveras presque tout ce dont tu auras besoin pour progresser dans le livre. Certains chapitres exigent des conditions particulières, que nous indiquons dès le début du chapitre correspondant afin que tu puisses t'y préparer avant de commencer.

Aux chapitres 5 et 9, nous t'indiquerons comment connecter de petits circuits électroniques entre eux pour relier le monde virtuel de Minecraft au monde réel. Pour cela, tu auras besoin de te procurer une série de composants électroniques, disponibles dans la plupart des magasins d'informatique (une liste de liens t'est fournie en annexe).

Le Raspberry Pi est doté de broches d'entrée et de sortie numérique (GPIO) auxquelles tu peux directement connecter tes composants électroniques. Dans la mesure où les PC et les Mac ne disposent pas de broches GPIO, nous avons sélectionné pour toi un dispositif qui peut se connecter aux ports USB de ton ordinateur pour ces projets. Là aussi, tu trouveras une liste de liens des principaux points de vente dans l'annexe A.

Toutefois, tes meilleurs alliés lors de ton voyage seront ta passion et ton enthousiasme pour le jeu, ainsi qu'un brin de curiosité et d'envie de mettre tes propres idées à l'épreuve afin de repousser sans cesse les limites de tes acquis !

Copyright © 2016 Eyrolles.

Remarque à l'attention des parents et des enseignants

Ce livre est divisé en plusieurs chapitres indépendants qu'on peut consulter individuellement. Chacun aborde un aspect spécifique de la programmation dans Minecraft. Les notions de langage Python y sont présentées progressivement et expliquées pas à pas. Les premiers chapitres s'adressent principalement aux débutants et à mesure qu'on avance dans le livre, le niveau de difficulté augmente, sollicitant davantage l'attention du lecteur sur les notions de Python.

Chaque chapitre contient un exercice pratique avec des instructions détaillées, que le lecteur pourra cocher s'il le souhaite dès lors qu'il l'aura terminé. Les instructions sont rédigées de manière pédagogique, aussi intelligiblement que s'il s'agissait d'un listing de programmation et avec des commentaires en marge. Les encadrés intitulés « Percer les secrets du code » comportent des explications plus fouillées. Les élèves pourront y revenir plus tard, sans interrompre le fil de leur lecture et de leurs essais de programmation.

Il faudra certainement plus d'une leçon pour venir à bout d'un chapitre, mais chacun se divise en plusieurs parties. Chaque partie est elle-même divisée en sous-parties qui sont articulées de telle sorte qu'on peut organiser les leçons par objectif ou étaler une activité sur plusieurs cours.

Sensible à la casse, le langage Python est indenté à gauche pour mieux faire apparaître la structure du code. L'aide d'un adulte pourra parfois s'avérer nécessaire, notamment pour les lecteurs les plus jeunes, afin de s'assurer qu'ils respectent bien la casse et l'indentation de leur code et pour éviter qu'ils y introduisent des erreurs. On peut toutefois télécharger tous les programmes sur le site Internet du livre (`www.editions-eyrolles.com/dl/14292`) pour comparer son indentation à celle de nos codes en cas d'erreur.

Plusieurs écoles se sont déjà procuré la version 3 de Python, mais à notre connaissance, Mojang n'a pas encore développé d'interface de programmation Minecraft compatible avec cette version. Il faudra donc se contenter d'utiliser la version 2, comme l'indique le premier chapitre.

Structure du livre

Chaque chapitre du livre constitue une aventure unique qui t'enseigne de nouvelles connaissances et introduit de nouveaux concepts à mesure que tu progresses dans les exercices de programmation. Dans ce livre, les chapitres sont indépendants et tu peux les lire individuellement. Cependant, il te sera peut-être plus facile de les parcourir dans l'ordre, car nous y présentons les concepts de programmation par ordre croissant de difficulté.

Il est en outre absolument essentiel que tu lises le chapitre 1 avant tous les autres, car il t'indique comment télécharger et installer tout ce dont tu as besoin pour commencer

Copyright © 2016 Eyrolles.

et te permet de vérifier que tout fonctionne correctement. Et tu y découvriras des connaissances de base qui te seront utiles tout au long du livre, même si nous introduisons des rappels dans les premiers chapitres en guise d'amorce.

Les trois premiers chapitres, notamment, s'adressent aux débutants qui ont peu ou n'ont pas de connaissances en programmation. Ils éclaircissent certains termes de jargon et expliquent les concepts à mesure que tu les découvres. Les chapitres 2, 3 et 4 abordent des éléments clés de Minecraft que tout bon joueur qui se respecte se doit de maîtriser. Tu y apprendras surtout à *détecter* les événements qui se produisent dans le monde de Minecraft, en effectuant de simples *calculs* mathématiques. Et tu découvriras comment faire *réagir* tes programmes de différentes manières, par exemple pour rédiger un message dans le chat ou créer des blocs automatiquement. Ces trois concepts essentiels – *détecter*, *calculer* et *réagir* – alimenteront tes lectures tout au long du chapitre. Tu apprendras à les manipuler avec habileté pour rédiger des programmes encore plus complexes et amusants !

Dans les chapitres 5 et 6, tu t'appuieras sur tes nouveaux acquis pour découvrir comment connecter Minecraft au monde réel. Ce sera l'occasion de t'initier à la formidable discipline de l'informatique physique en confectionnant de petits circuits informatiques interagissant avec Minecraft. Tu es encore loin de t'imaginer tous les jeux qu'il est possible de créer grâce à cela ! En abordant le chapitre 6, tu verras comment tirer parti de dossiers contenant un grand nombre de données pour enregistrer et dupliquer de larges structures à l'aide d'une « machine à dupliquer ».

Les chapitres 7 et 8, quant à eux, révèlent l'utilité du module gratuit MinecraftStuff, qui permet de bâtir des lignes, des cercles et d'autres figures en 2D à partir de blocs, voire de magnifiques sphères et pyramides en 3D. Autant de figures qui pourront ensuite servir de socle à d'immenses structures, trop difficiles à bâtir de ses propres mains. Dans le chapitre 8, tu apprendras à doter les objets d'une intelligence et d'une personnalité. Grâce à ces techniques, tu pourras épater tes amis en les invitant à jouer à tes propres « jeux dans le jeu ».

Enfin, le chapitre 9 reprend tous les concepts et acquis des chapitres précédents et t'invite à les mettre en pratique une dernière fois pour créer un superbe jeu, dans lequel tu devras redoubler d'habileté et de ruse pour éviter des objets ou pour les emporter avec toi, tout en gagnant le plus de points possible ! Tu auras également la possibilité de mettre tes connaissances d'électronique en pratique en actionnant d'un seul doigt des dispositifs qui auront une incidence dans le monde virtuel de Minecraft.

L'annexe A (intitulée « Pour aller plus loin ») fournit toute une panoplie d'informations que tu pourras utiliser à ta guise pour parfaire tes connaissances et compléter ton apprentissage du langage Python. Tu disposeras ainsi d'un bagage supplémentaire pour rédiger des programmes encore plus impressionnants que ceux dont il est question dans ce livre.

L'annexe B (intitulée « Tableau des références »), que tu pourras télécharger sur le site des éditions Eyrolles (`www.editions-eyrolles.com/dl/14292`), regroupe l'ensemble des fonctionnalités de programmation utilisées dans le livre, les instructions

Copyright © 2016 Eyrolles.

spécifiques à Minecraft ainsi qu'un tableau des différents types de blocs avec lesquels tu peux bâtir des structures. Tu y trouveras une mine d'informations indispensables pour tes projets. À consulter sans modération, donc !

Enfin, le « Glossaire », également disponible sur le site des éditions Eyrolles (www.editions-eyrolles.com/dl/14292), reprend toutes les définitions que contiennent les chapitres de ce livre. Il est là pour t'aider à assimiler le lexique et les termes que nous employons.

Le site Internet associé

Au fil de ta lecture, tu verras que nous faisons plusieurs fois référence au site Internet qui est associé à ce livre : www.editions-eyrolles.com/dl/14292. Tu y trouveras les kits de démarrage dont tu auras besoin pour commencer à programmer dans Minecraft ainsi qu'une série de vidéos didactiques (en anglais) que nous avons créées pour t'aider en cas de blocage. Tu pourras également y télécharger des fichiers de code pour venir à bout des plus gros exercices.

Enfin, pour que l'aventure continue, tu trouveras en prime, toujours sur le site Internet du livre (www.editions-eyrolles.com/dl/14292), un chapitre entier dédié à la construction d'un ascenseur fonctionnel qui te permettra de passer d'un étage à un autre en un clin d'œil. Ce chapitre bonus est assez difficile et sollicite toutes les connaissances que tu auras acquises et mises en pratique dans ce livre. Tu devras même contrôler ton ascenseur à l'aide de circuits électroniques.

Notre maison d'édition (Eyrolles) a gentiment accepté de te fournir un document PDF à télécharger sur le site Internet (www.editions-eyrolles.com/dl/14292) associé en guise d'annexe. Gardes-le sous la main, car il te sera très utile lors de tes aventures et même plus tard, lorsque tu participeras à de nouveaux projets de programmation. Le site Internet des éditions Eyrolles héberge aussi un glossaire. Même si les définitions sont incluses dans les chapitres, tu pourras t'y rendre à tout moment pour les consulter.

Autres ressources

Les ordinateurs sont des appareils complexes et les systèmes d'exploitation et les logiciels évoluent sans cesse. Nous avons fait notre possible pour t'épargner les soucis liés à ces changements en mettant un kit de démarrage à ta disposition sur le site, que tu pourras télécharger dès le premier chapitre. Il contient presque tout ce dont tu auras besoin. Cependant, si tu rencontres des problèmes ou si tu as besoin d'une aide supplémentaire, voici une liste de quelques sites à consulter :

- Créer un identifiant, télécharger et installer Minecraft : http://minecraft.net
- Jouer à Minecraft : http://minecraft.gamepedia.com/Minecraft_Wiki
- Raspberry Pi : www.raspberrypi.org

Copyright © 2016 Eyrolles.

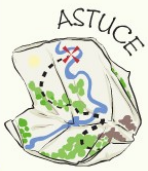
- Microsoft Windows : `http://support.microsoft.com`
- Mac et OS X : `www.apple.com/support`
- Le langage Python : `www.python.org`
- L'environnement de développement de programmation IDLE (en anglais) : `https://docs.python.org/2/library/idle.html`
- La version Pi de Minecraft : `http://pi.minecraft.net`
- Le serveur CraftBukkit : `http://wiki.bukkit.org`
- Le module Bukkit RaspberryJuice : `http://dev.bukkit.org/bukkit-plugins/raspberryjuice`

Conventions de lecture

Pour te guider dans ta lecture, tu trouveras les encadrés suivants dans l'ouvrage.



Ces encadrés éclaircissent les concepts et les termes qui peuvent te sembler obscurs.



Ces encadrés te donnent des astuces qui rendront tes tâches de programmation beaucoup plus simples.



Ces encadrés contiennent des avertissements importants qui te permettront de réaliser certaines tâches en toute sécurité.



Ces encadrés contiennent des exercices rapides qui te permettront d'évaluer tes connaissances et d'approfondir certains sujets.

Copyright © 2016 Eyrolles.

Ces encadrés nous permettent de développer certains points et de te fournir des informations supplémentaires qui pourront t'être utiles.



Ces encadrés t'invitent à te rendre sur le site Internet associé au livre (www.editions-eyrolles.com/dl/14292) pour visualiser des vidéos didactiques (en anglais) qui t'expliquent comment effectuer certaines tâches pas à pas.



Tu trouveras également deux autres types d'encadrés dans le livre. Les encadrés « Défi » te proposent des exercices supplémentaires à réaliser si tu souhaites pousser ton apprentissage un peu plus loin, en effectuant par exemple des modifications ou en ajoutant de nouvelles fonctionnalités aux projets réalisés. Et les encadrés « Percer les secrets du code » approfondissent certains concepts ou caractéristiques du code pour que tu puisses davantage te familiariser avec le langage de programmation Python. En somme, tu peux tout d'abord te concentrer sur le fonctionnement de tes programmes et t'attarder ensuite sur ces encadrés afin d'en savoir plus sur le fonctionnement de tes codes et décupler leurs capacités.

En suivant nos instructions, veille à bien rédiger le code exactement comme il figure dans le livre. Le langage Python tient compte de l'indentation, c'est-à-dire que les espaces au début de chaque ligne de code ont une importance pour le sens du programme. Veille donc à bien les respecter. Pour te faciliter la tâche, nous avons mis les lignes de code en évidence sur un fond coloré pour en faire ressortir l'indentation. Mais ne t'en fais pas trop. Nous t'expliquerons plus en détail comment fonctionne l'indentation en temps voulu, dans les premiers chapitres.

Parfois, tu auras besoin de rédiger de très longues lignes de code, trop longues pour tenir sur une seule ligne dans ce livre. Donc, si tu vois le signe ↩ à la fin d'une ligne de code, cela signifie que la ligne en question et la suivante font partie d'une seule et même ligne. Tu devras donc la retranscrire dans ton programme sur une seule ligne et non sur deux lignes séparées, comme le montre l'exemple suivant :

```
print("Bienvenue dans Apprendre à coder grâce à Minecraft de ↩
Martin O'Hanlon et David Whale")
```

Copyright © 2016 Eyrolles.



Si tu lis ce livre sur une liseuse, configure la police sur la plus petite taille pour t'assurer que tes programmes s'afficheront correctement lorsque tu les rédigeras. Cette mesure de précaution permettra d'éviter les sauts de ligne inutiles à cause des marges trop petites, pouvant conduire à des erreurs dans tes programmes.

La plupart des chapitres s'achèvent sur un petit « Tableau de références » qui reprend les principaux concepts et instructions évoqués. Tu pourras le consulter lorsque tu auras besoin de te rafraîchir la mémoire. Tu trouveras également un guide de référence complet dans l'annexe B du livre qui regroupe les instructions de programmation les plus importantes de Minecraft et de Python. Nous espérons qu'il t'apportera une aide précieuse tout au long de tes aventures.

Chaque chapitre que tu achèveras te donnera le droit d'arborer un badge, que tu pourras récupérer sur le site Internet du livre (`www.editions-eyrolles.com/dl/14292`).

Pour aller plus loin

Dans l'annexe A, tu trouveras tous les outils nécessaires pour conduire tes connaissances en programmation vers un niveau supérieur, notamment grâce à une liste complète de sites Internet, d'organisations, de vidéos et d'autres ressources. La plupart de ces ressources te conduiront vers des forums où tu pourras poser des questions librement ou rencontrer d'autres programmeurs de Minecraft.

Tu peux aussi nous contacter directement en nous envoyant un message sur nos sites Internet respectifs :

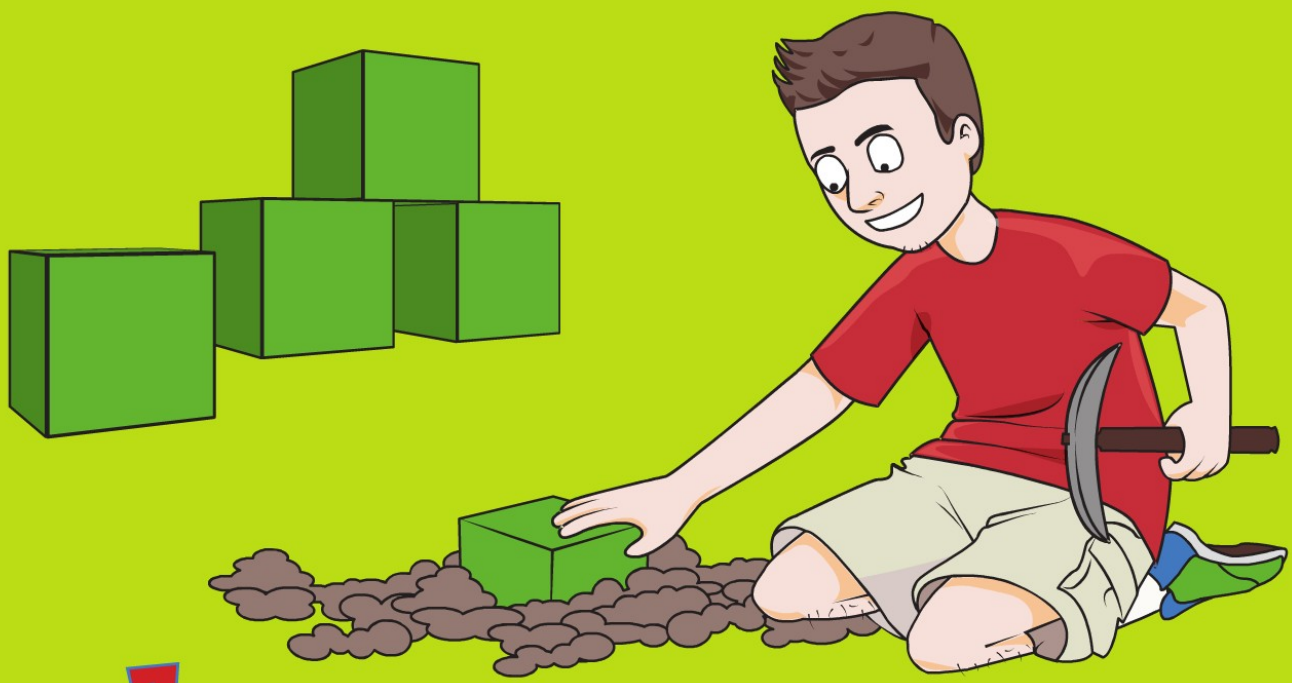
Martin : `www.stuffaboutcode.com`

David : `http://blog.whaleygeek.co.uk`

En route pour l'aventure !



Copyright © 2016 Eyrolles.



Chapitre 1

Bonjour le monde de Minecraft

DANS CE LIVRE, tu vas apprendre à rédiger des programmes qui interagiront avec ton monde Minecraft et te permettront de réaliser des choses passionnantes. Pour cela, tu utiliseras un langage de programmation appelé **Python**. La possibilité de diriger le monde de Minecraft à l'aide d'un programme Python était tout d'abord réservée à la version Pi de Minecraft pour les ordinateurs Raspberry Pi. Si tu n'as pas de Raspberry Pi, mais que tu joues à Minecraft sur un ordinateur Windows ou Mac, alors pas de panique ! Tu auras juste à réaliser quelques paramétrages avant de commencer, ce que nous t'expliquerons en temps voulu.

Python est le langage de programmation que nous utilisons dans cet ouvrage.



Cet ouvrage contient un bon nombre d'aventures qui t'enseigneront la programmation appliquée au jeu Minecraft. Tu y trouveras toute une panoplie de tours de passe-passe que tu peux réaliser dans Minecraft pour épater tes amis et rendre le jeu encore plus

Copyright © 2016 Eyrolles.

amusant. Tu découvriras notamment des techniques fabuleuses pour te déplacer dans le jeu et tu apprendras en un rien de temps à bâtir des villes entières et des créations encore jamais vues dans le jeu.

Le langage de programmation Python est fourni avec un éditeur de texte appelé IDLE, qui te servira pour rédiger, modifier et exécuter les programmes enseignés dans ce livre.



Dans ces encadrés, je t'explique des choses ou te donne des informations supplémentaires qui te seront utiles.

Lorsque les informaticiens apprennent un nouveau langage de programmation ou une nouvelle manière de réaliser quelque chose, ils commencent toujours par écrire un programme « Hello world » (« Bonjour le monde », en français). C'est un programme très simple qui permet d'afficher l'expression « Bonjour le monde » sur l'écran de l'ordinateur, afin de s'assurer que tout est installé et fonctionne correctement.

Dans ce premier chapitre, tu configureras ton ordinateur de façon à ce qu'il te permette d'afficher le texte « Bonjour le monde de Minecraft » sur le chat de Minecraft (comme dans la figure 1-1).



FIGURE 1-1 Bonjour le monde de Minecraft.

Pour apprendre à programmer dans Minecraft avec ce livre, tu as besoin d'utiliser l'un des ordinateurs suivants : un PC fonctionnant sous Windows, un Mac fonctionnant sous OS X ou un Raspberry Pi fonctionnant sous Raspbian. La configuration de ton

Copyright © 2016 Eyrolles.

ordinateur variera en fonction du type de dispositif que tu possèdes, mais une fois que tu l'auras terminée, tu programmeras Minecraft exactement de la même manière sur toutes les machines. Pour cette étape, tu peux télécharger un kit de démarrage sur le site Internet associé à ce livre (`www.editions-eyrolles.com/dl/14292`). Les kits de démarrage ont été testés pour garantir que tous les exercices du livre fonctionnent correctement. Tu verras que ton kit contient un fichier README, qu'il te faudra consulter. Il décrit le contenu du kit et la façon dont il a été créé. En théorie, ces informations suffisent pour configurer ton ordinateur sans aide supplémentaire, mais nous te recommandons plutôt de suivre les instructions de ce livre, car tu en tireras davantage de bénéfices.

En fonction de l'ordinateur que tu utilises, rends-toi directement à la section qui te concerne, à savoir : « Configurer ton Raspberry Pi pour programmer dans Minecraft » ou « Configurer ton PC ou ton Mac pour programmer dans Minecraft ».

Il est essentiel que tu réussisses l'étape de configuration de ton ordinateur, car sinon tu pourrais vite t'emmêler les pinceaux. Veille donc à suivre les instructions avec beaucoup d'attention.



Configurer ton Raspberry Pi pour programmer dans Minecraft

Si tu utilises un Raspberry Pi, tu dois passer par deux étapes pour configurer ton ordinateur avant de pouvoir créer ton premier programme Minecraft.

1. Télécharger et installer Minecraft Pi Edition, c'est une version de Minecraft spécialement conçue pour les Raspberry Pi.
2. Télécharger et décompresser le kit de démarrage pour Raspberry Pi. Il contient tout ce dont tu as besoin pour partir à l'aventure avec Minecraft. Tout est contenu dans un dossier nommé `MyAdventures`, où tu enregistreras également tous tes programmes Minecraft.

Tu peux aussi visualiser une vidéo en anglais t'expliquant comment configurer ton Raspberry Pi. Pour cela, rends-toi sur le site Internet du livre à l'adresse `www.editions-eyrolles.com/dl/14292`.



Copyright © 2016 Eyrolles.

L'interface graphique (GUI, *Graphical User Interface* en anglais) de Raspberry Pi, également connue sous le nom de X Window, est celle que nous utilisons tout au long du livre. Cette GUI est déjà installée sur Raspbian, mais selon la configuration de ton Raspberry Pi, il se peut qu'elle ne charge pas au démarrage. Tu devras dans ce cas démarrer ton dispositif à l'aide d'un identifiant et exécuter l'interpréteur de commandes.

Si ton Raspberry Pi est configuré pour démarrer à l'aide de l'interpréteur de commandes, tu devras t'identifier puis saisir `startx` et appuyer sur `Entrée` afin de charger la GUI lorsque l'interpréteur s'affichera.

Dans le doute, installe à nouveau ton système Raspbian sur ton Raspberry Pi au moment où tu commences ce livre pour t'assurer de son bon fonctionnement. Pour en savoir plus sur la configuration de ton Raspberry Pi et l'installation de Raspbian, rends-toi sur `www.raspberrypi.org/help`.

Installer Minecraft sur Raspberry Pi

Après avoir démarré ton Raspberry Pi avec l'interface graphique, tu peux passer à l'installation de Minecraft Pi Edition. Télécharge le fichier d'installation de cette version sur le site `http://pi.minecraft.net`. Tu y trouveras également des instructions sur la manière d'extraire le jeu du fichier et de le démarrer.

Les instructions suivantes t'indiquent comment faire pour télécharger et extraire Minecraft Pi Edition.

1. Double-clique sur l'icône LXTerminal qui se trouve sur le bureau (elle ressemble à un écran d'ordinateur noir). La fenêtre de LXTerminal va alors s'ouvrir, t'invitant à saisir des instructions.
2. Afin de télécharger le fichier d'installation de Minecraft Pi Edition, saisis les instructions suivantes une par une dans la fenêtre LXTerminal, en appuyant sur `Entrée` après chacune d'elles :

```
cd ~
wget https://s3.amazonaws.com/assets.minecraft.net/pi↩
/minecraft-pi-0.1.1.tar.gz
```

   Une barre de progression apparaîtra, affichant le pourcentage de téléchargement.

3. Pour extraire Minecraft Pi Edition, saisis :

```
tar -zxvf minecraft-pi-0.1.1.tar.gz
```

   Minecraft Pi Edition sera extrait dans un dossier nommé `mcpi`.

Ensuite, tu devras télécharger le kit de démarrage pour Raspberry Pi et extraire le contenu du dossier `MyAdventures` dans l'ordre suivant.

Copyright © 2016 Eyrolles.

1. Ouvre ton navigateur web (par exemple Midori), rends-toi sur le site Internet du livre (`www.editions-eyrolles.com/dl/14292`) et télécharge le kit de démarrage pour Raspberry Pi.
2. Une fenêtre va s'ouvrir, te demandant si tu souhaites ouvrir ou télécharger le dossier. Clique sur `Ouvrir` afin de démarrer Xarchiver, un programme qui décompresse les fichiers.
3. Clique sur `Action>Extract` dans le menu `Xarchiver`.
4. Saisis `/home/pi` dans la zone de texte `Extract to:` (comme le montre la figure 1-2).
5. Clique ensuite sur `Extract`. Tes fichiers seront décompressés et enregistrés dans le dossier **`/home/pi`**, que tu pourras consulter à tout moment.

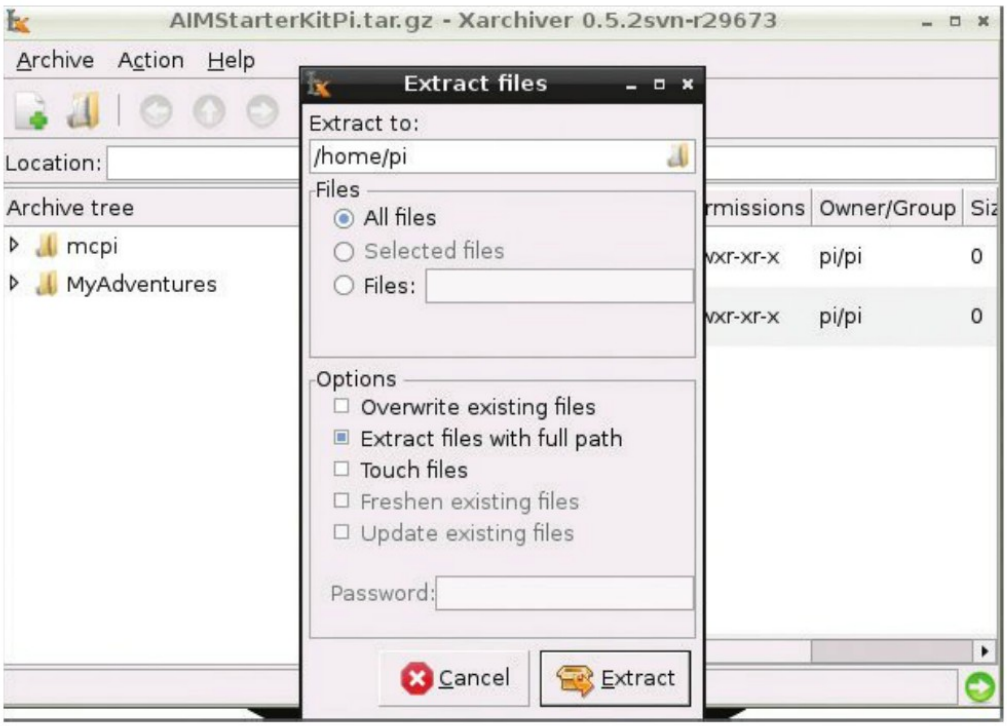


FIGURE 1-2 Choisis un emplacement où décompresser ton kit de démarrage pour Raspberry Pi.

Démarrer Minecraft sur Raspberry Pi

Maintenant que tu as téléchargé et installé Minecraft Pi Edition, démarre le jeu et fais quelques tours d'essai avant de te lancer dans l'écriture de ton premier programme.

Copyright © 2016 Eyrolles.



Dans les prochains chapitres, les instructions te suggéreront directement de démarrer Minecraft sans te rappeler ces étapes. Si tu as oublié comment démarrer Minecraft sur Raspberry Pi et que tu as besoin d'un rappel, il te suffira de relire cette section.

Pour démarrer Minecraft, suis les étapes ci-dessous.

1. Double-clique sur l'icône LXTerminal qui se trouve sur ton bureau. La fenêtre de LXTerminal va alors s'ouvrir, t'invitant à saisir des instructions.
2. Pour démarrer Minecraft, saisis les instructions suivantes (comme l'indique la figure 1-3) :

```
cd ~
cd mcpi
./minecraft-pi
```

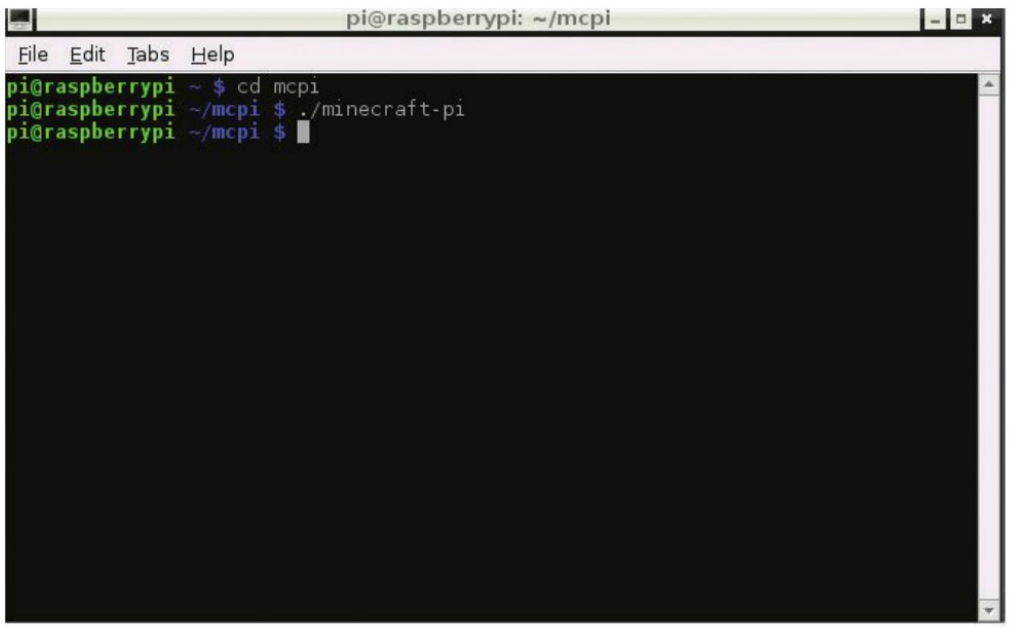


FIGURE 1-3 Pour démarrer Minecraft, les instructions sont simples.

3. Et voilà ! Tu peux désormais commencer à jouer à Minecraft. Clique sur `Démarrer le jeu>Créer un nouveau monde`.

Le menu principal dispose de deux options : `Démarrer le jeu pour bâtir un nouveau monde ou entrer dans un monde existant` et `Rejoindre le jeu pour rejoindre le monde d'un autre joueur de Minecraft Pi Edition`.

Copyright © 2016 Eyrolles.

Ton Raspberry Pi est maintenant configuré et tu es parvenu à lancer le jeu. Tu peux désormais ignorer la partie qui va suivre (à moins que tu souhaites également installer Minecraft sur un PC ou un Mac en plus de ton Raspberry Pi). Passe directement à la section intitulée « Créer un programme » de ce chapitre.

Configurer ton PC ou ton Mac pour programmer dans Minecraft

Que tu utilises un PC sous Windows ou un Mac, tu dois tout d'abord t'assurer que Minecraft est installé sur ton ordinateur et qu'il fonctionne correctement. Si tu n'as pas encore acquis ce jeu et si tu ne disposes pas d'un identifiant pour y jouer, rends-toi sur `www.minecraft.net` pour l'acheter. En cas de problème pendant l'installation, le démarrage ou le jeu, tu disposes d'une aide (en anglais) à portée de main sur le site `https://help.mojang.com`.

Pour programmer la version complète de Minecraft sur PC et Mac, tu auras besoin d'utiliser **Bukkit** (`http://bukkit.org`), le serveur open source de Minecraft, et le plug-in RaspberryJuice (`http://dev.bukkit.org/bukkit-plugins/raspberryjuice`), comme pour Raspberry Pi.

Bukkit est un serveur Minecraft que tu peux modifier en créant des plug-ins qui interagissent avec le jeu de différentes manières. Le plug-in RaspberryJuice pour Bukkit te permet d'écrire des programmes et de modifier Minecraft, comme si tu utilisais un ordinateur Raspberry Pi, à cela près que, pour créer tes programmes Minecraft, tu utiliseras Bukkit, le **plug-in** RaspberryJuice et le langage de programmation Python.

Bukkit est un serveur Minecraft qui permet aux utilisateurs de modifier le jeu à l'aide de plug-ins.



Un **plug-in** est un programme qui fonctionne à travers le serveur Bukkit et te permet de modifier Minecraft.



Enfin, tu devras télécharger le langage de programmation Python et l'installer sur ton ordinateur. Tout au long de l'ouvrage, tu utiliseras la version 2 de Python, même si la version 3 est déjà sortie, car le répertoire Python fourni par Mojang – celui que tu utiliseras pour

Copyright © 2016 Eyrolles.

programmer dans Minecraft – ne fonctionne qu'avec la version 2. Tous les programmes enseignés ici ont été testés avec la version 2.7.6. Celle-ci n'est pas obligatoire, mais elle est recommandée. Dans tous les cas, tu devras au moins utiliser une version 2 et quelques.



Si tu souhaites en savoir plus sur Python, rends-toi sur le site `python.org`. Tu peux le télécharger à l'adresse `www.python.org/download`. Et la page Wiki dédiée à ce langage, `wiki.python.org`, contient une montagne d'informations, de vidéos didactiques et de liens vers des sites Internet de la communauté Python.

Pour configurer ton PC ou ton Mac et rédiger ton premier programme Minecraft, tu vas devoir suivre ces trois étapes.

1. Télécharger le kit de démarrage pour PC ou Mac et le décompresser. Tu verras qu'il contient un serveur Bukkit pré-configuré avec le plug-in RaspberryJuice ainsi qu'un dossier intitulé `MyAdventures` où tu pourras enregistrer tes programmes Minecraft.
2. Télécharger et installer le langage de programmation Python.
3. Configurer Minecraft pour utiliser la même version que celle de Bukkit et connecter le jeu au serveur Bukkit.



Tu peux aussi visualiser une vidéo t'expliquant comment configurer ton PC ou ton Mac. Pour cela, rends-toi sur le site Internet du livre, à l'adresse `www.editions-eyrolles.com/dl/14292`.

Installer le kit de démarrage et Python sur ton PC Windows

Si tu utilises un PC sous Windows, la configuration s'effectuera en deux étapes.

1. Télécharge et enregistre sur ton bureau le kit de démarrage pour Windows.
2. Puis télécharge et installe le langage de programmation Python.

Télécharger et décompresser le kit de démarrage

Pour télécharger le kit de démarrage Windows et enregistrer son contenu sur ton bureau (en l'enregistrant sur ton bureau, tu pourras y accéder plus facilement), suis les instructions ci-dessous.

Copyright © 2016 Eyrolles.

1. Ouvre le navigateur web de ton PC (par exemple, Internet Explorer ou Chrome), saisis l'adresse du site Internet associé au livre (`www.editions-eyrolles.com/dl/14292`) et télécharge le kit de démarrage pour Windows.
2. Une fois le téléchargement du dossier compressé terminé, ouvre-le. Pour cela, clique sur `Ouvrir` ou ouvre le dossier de téléchargement, puis double-clique sur le fichier `AIMStarterKitPC.zip`.
3. Le dossier compressé ne contient qu'un seul dossier, nommé `AdventuresInMinecraft`. Copie-le sur ton bureau : place ton curseur sur le dossier, puis appuie sur le bouton gauche de la souris et, tout en le maintenant appuyé, fais glisser le dossier sur ton bureau (voir la figure 1-4).

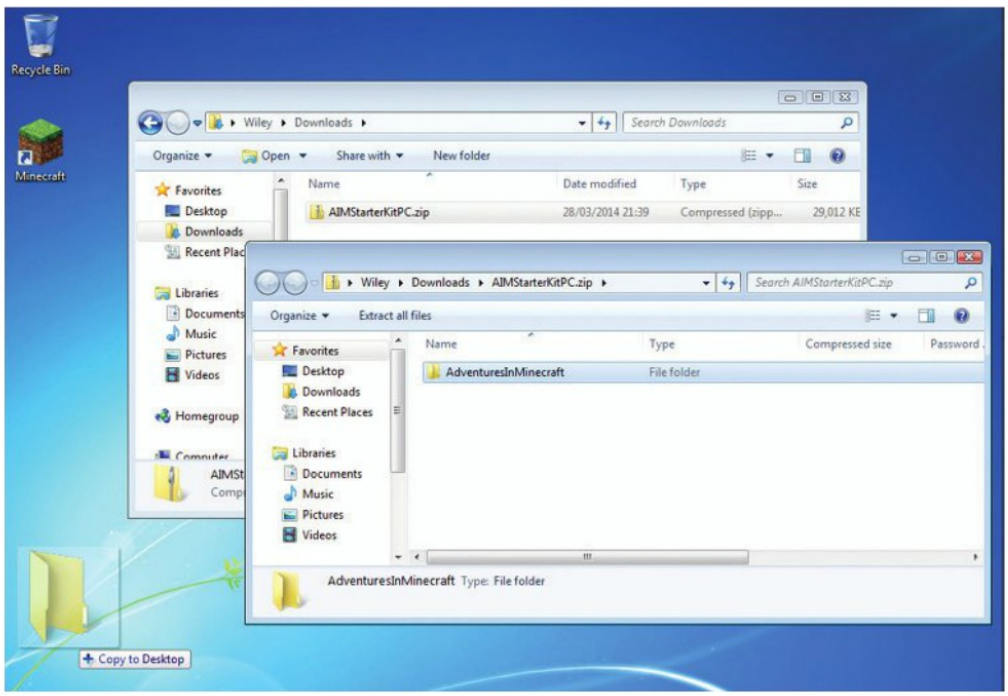


FIGURE 1-4 Copie le dossier AdventuresInMinecraft sur ton bureau pour pouvoir y accéder plus facilement.

Télécharger et installer Python

Comme tu rédigeras tes codes en Python, installe le langage de programmation ainsi que l'éditeur de texte IDLE de la manière suivante.

Si tu ne possèdes pas de compte administrateur sur ton ordinateur, tu auras besoin qu'un administrateur se connecte à l'aide de son identifiant pour pouvoir installer Python.



Copyright © 2016 Eyrolles.

1. Ouvre le navigateur web de ton PC (Internet Explorer, Chrome, etc.). Rends-toi sur `www.python.org/download/releases/2.7.6` et clique sur le lien du fichier Windows x86 MSI Installer (2.7.6) pour commencer l'installation.
2. Une fois le fichier `python-2.7.6.msi` téléchargé, exécute-le en cliquant sur `Ouvrir>Exécuter` ou en ouvrant le dossier téléchargé et en double-cliquant sur le fichier.
3. Il se peut qu'une boîte de dialogue s'affiche, te demandant si tu souhaites exécuter ce fichier. Clique alors sur `Exécuter`.
4. Clique ensuite sur `Next` pour lancer l'installation.
5. Tu devras alors choisir un emplacement pour terminer l'installation. Nous te conseillons d'utiliser l'emplacement par défaut. Clique à nouveau sur `Next` et note l'emplacement quelque part pour que tu puisses facilement retrouver tes fichiers Python lorsque tu en auras besoin.
6. Ne modifie aucune des options de la fenêtre `Customize Python`, clique simplement une nouvelle fois sur `Next`.
7. Le `Contrôle de compte d'utilisateur` s'affichera alors peut-être pour te demander si tu souhaites autoriser le programme à installer des logiciels sur ton ordinateur. Le cas échéant, clique sur `Oui`.
8. Enfin, attends que le programme de configuration finisse de s'installer, puis clique sur `Finish`.

Installer le kit de démarrage et Python sur ton Mac

Si tu utilises un Mac, la configuration s'effectuera en deux étapes.

1. Télécharge et enregistre sur ton bureau le kit de démarrage pour Mac.
2. Télécharge et installe le langage de programmation Python sur ton Mac.

Télécharger et décompresser le kit de démarrage

Suis ces quelques étapes pour télécharger le kit de démarrage pour Mac et enregistrer son contenu sur ton bureau (en l'enregistrant sur ton bureau, tu pourras y accéder plus facilement).

1. Ouvre le navigateur web de ton Mac (par exemple Safari ou Chrome), saisis l'adresse du site Internet du livre (`www.editions-eyrolles.com/dl/14292`) et télécharge le kit de démarrage pour Mac.
2. Une fois que le téléchargement du dossier compressé est terminé, ouvre-le. Pour cela, clique sur le fichier ou ouvre le dossier de téléchargement et double-clique sur le fichier `AIMStarterKitPC.zip`.

Copyright © 2016 Eyrolles.

3. Le dossier compressé ne contient qu'un seul dossier, nommé `AdventuresInMinecraft`. Copie-le sur ton bureau : place ton curseur sur le dossier, puis appuie sur le bouton gauche de la souris et, tout en le maintenant appuyé, fait glisser le dossier sur ton bureau (voir la figure 1-5).

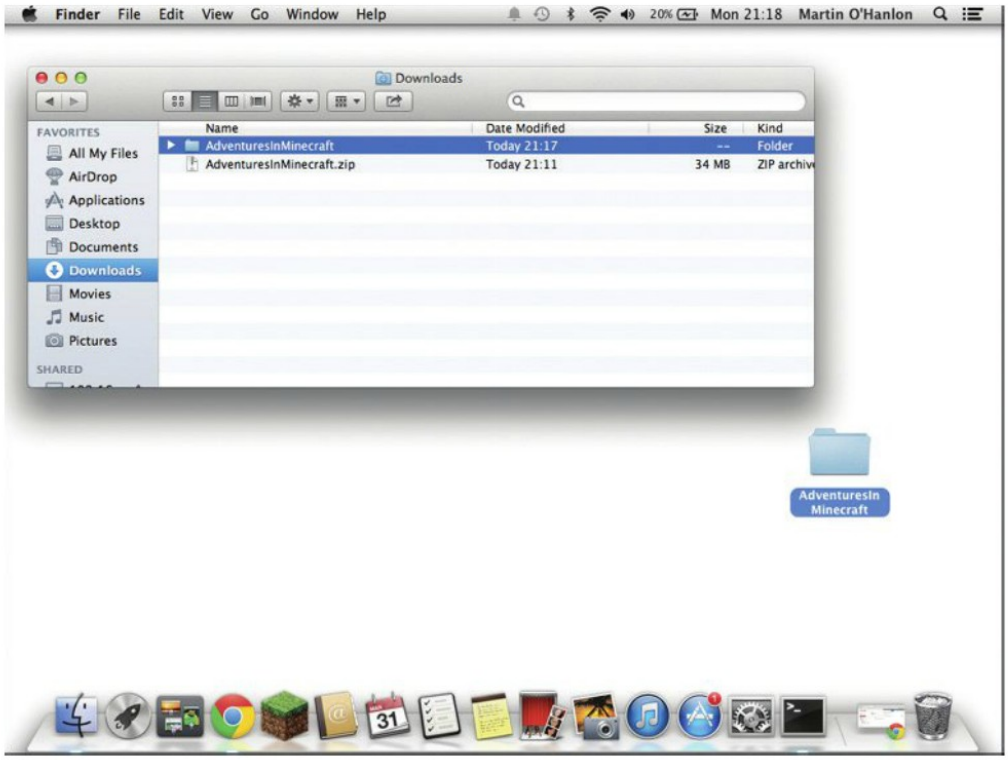


FIGURE 1-5 Copie le dossier AdventuresInMinecraft sur ton bureau.

Télécharger et installer Python

Comme tu rédigeras tes codes en Python, installe le langage de programmation ainsi que l'éditeur de texte IDLE, de la manière suivante.

> En fonction de la configuration de ton ordinateur, tu auras peut-être besoin de saisir ton mot de passe Apple ou qu'un administrateur se connecte avec son identifiant pour installer Python.
>
> 
>

1. Ouvre le navigateur web de ton Mac. Rends-toi sur `www.python.org/download/releases/2.7.6` et clique sur le lien de téléchargement du Mac OS X 64-bit/32-bit x86-64/i386 Installer (2.7.6) pour les versions OS X 10.6 et antérieures de Mac.

Copyright © 2016 Eyrolles.

2. Une fois le fichier `python-2.7.6-macosx10.6.dmg` téléchargé, ouvre-le en cliquant dessus ou ouvre le dossier de téléchargement dans le Finder et double-clique sur le fichier.

3. Recherche le fichier intitulé `Python.mpkg`. Il s'agit du fichier d'installation du programme. Fais un clic droit sur ce fichier, puis clique sur `Ouvrir avec>Installer` pour lancer l'installation de Python.



Sur les versions OS X 10.8 et antérieures, il est possible qu'un message s'affiche indiquant que Python ne peut pas être installé, car il provient d'un « éditeur inconnu ». Pas de panique ! Cela signifie simplement que le programme d'installation de Python n'est pas compatible avec le pare-feu de ton système OS X 10.8. Mais tu peux tout de même l'installer sans aucun problème, car il ne constitue pas une menace pour ton ordinateur. Clique simplement sur `Ouvrir`.

4. L'écran d'introduction affichera des informations au sujet du programme d'installation de Python. Clique sur `Continue`.

5. L'écran `Read Me` affiche des informations importantes au sujet de Python. Clique à nouveau sur `Continue`.

6. L'écran `License` affiche le contrat de licence du logiciel. Clique encore une fois sur `Continue` puis sur `Agree`.

7. L'écran de `Type d'installation` affiche l'espace qui sera utilisé sur le disque. Clique sur `Installer`. Tu auras également la possibilité de modifier l'emplacement du fichier d'installation si nécessaire.

8. Saisis ton mot de passe Apple et clique sur `Installer le logiciel` pour lancer l'installation et voir la progression. Attends que Python finisse de s'installer.

9. Clique sur `Fermer` lorsque le programme d'installation indique que le logiciel a été installé avec succès.

Démarrer Minecraft sur ton PC Windows ou Mac



Dans les prochains chapitres, les instructions te suggéreront directement de démarrer Minecraft sans te rappeler toutes ces étapes. Il te suffira de relire cette partie si jamais tu as oublié comment démarrer Bukkit et de connecter Minecraft au serveur sur ton PC ou ton Mac.

Copyright © 2016 Eyrolles.

Tu as certes installé toutes les applications dont tu as besoin sur ton PC Windows ou Mac, mais tu n'es pas encore tout à fait prêt. Avant de pouvoir commencer l'aventure, tu dois d'abord redémarrer le serveur Bukkit et y connecter ton jeu Minecraft en suivant ces étapes.

1. Ouvre tout d'abord le dossier `AdventuresInMinecraft` que tu as enregistré sur ton bureau en double-cliquant dessus.
2. Lance le programme StartBukkit en double-cliquant dessus. L'interpréteur de commandes Bukkit s'ouvrira alors.

> Si tu utilises un Mac, il se peut que tu reçoives un message indiquant que « StartBukkit.command » ne peut s'ouvrir, car il provient d'un éditeur inconnu, selon la configuration de ton ordinateur. Si tel est le cas, effectue un clic droit sur le programme StartBukkit et clique sur `Ouvrir` avec le terminal (au lieu de faire un double-clic).
>
> 
>

3. Appuie sur une touche de ton clavier pour lancer Bukkit. Pendant le démarrage, des messages apparaîtront à l'écran pour t'indiquer la progression.

> Au premier démarrage, il se peut que ton ordinateur affiche un message te demandant la permission de lancer le programme ou d'autoriser Bukkit à accéder au serveur. Si tel est le cas, accepte.
>
> 
>

4. Une fois que Bukkit a fini de charger, le message « Done » apparaît (voir la figure 1-6).

```
21:48:43 [INFO] Loading properties
21:48:43 [INFO] Default game type: CREATIVE
21:48:43 [INFO] Generating keypair
21:48:44 [INFO] Starting Minecraft server on *:25565
21:48:44 [INFO] This server is running CraftBukkit version git-Bukkit-1.6.4-R2.0
-b2918jnks (MC: 1.6.4) (Implementing API version 1.6.4-R2.0)
21:48:45 [INFO] [RaspberryJuice] Loading RaspberryJuice v1.3
21:48:45 [WARNING] **** SERVER IS RUNNING IN OFFLINE/INSECURE MODE!
21:48:45 [WARNING] The server will make no attempt to authenticate usernames. Be
ware.
21:48:45 [WARNING] While this makes the game possible to play without internet a
ccess, it also opens up the ability for hackers to connect with any username the
y choose.
21:48:45 [WARNING] To change this, set "online-mode" to "true" in the server.pro
perties file.
21:48:45 [INFO] Preparing level "world"
21:48:45 [INFO] Preparing start region for level 0 (Seed: -3789787156735749996)
21:48:46 [INFO] Preparing spawn area: 21%
21:48:47 [INFO] Preparing start region for level 1 (Seed: -3789787156735749996)
21:48:48 [INFO] Preparing spawn area: 50%
21:48:49 [INFO] Preparing start region for level 2 (Seed: -3789787156735749996)
21:48:50 [INFO] [RaspberryJuice] Enabling RaspberryJuice v1.3
21:48:50 [INFO] Server permissions file permissions.yml is empty, ignoring it
21:48:50 [INFO] Done (5.294s)! For help, type "help" or "?"
>
```

FIGURE 1-6 Une fois chargé et prêt à être utilisé, l'interpréteur de commandes Bukkit affiche « Done ».

Copyright © 2016 Eyrolles.

Dans tous les chapitres de ce livre, nous utilisons la version 1.6.4 de Minecraft et de Bukkit. Cela signifie que tu dois modifier le Launcher Minecraft pour qu'il lance cette version du jeu. En utilisant celle-ci plutôt que la dernière, tu pourras être sûr que les codes et programmes enseignés dans ce livre fonctionneront correctement. Par ailleurs, tu ne devras effectuer cette opération qu'une seule fois, car Minecraft se souviendra de la version que tu utilises.



Si tu préfères malgré tout utiliser une version de Minecraft plus récente, le fichier `README` contenu dans le kit de démarrage explique comment créer ton propre kit de démarrage. Seuls les utilisateurs experts qui savent configurer leur ordinateur, modifier des fichiers de configuration et exécuter des programmes Java sont invités à effectuer cette manipulation.

Pour configurer le Launcher Minecraft afin d'utiliser la version 1.6.4, suis ces étapes.

1. Démarre Minecraft.
2. Lorsque la fenêtre du Launcher de Minecraft s'affiche, clique sur `Edit Profile` en bas à gauche de ton écran, en dessous du nom d'utilisateur.
3. Dans la liste déroulante `Use version`, sélectionne la version 1.6.4, comme indiqué dans la figure 1-7.
4. Clique enfin sur `Save Profile`.

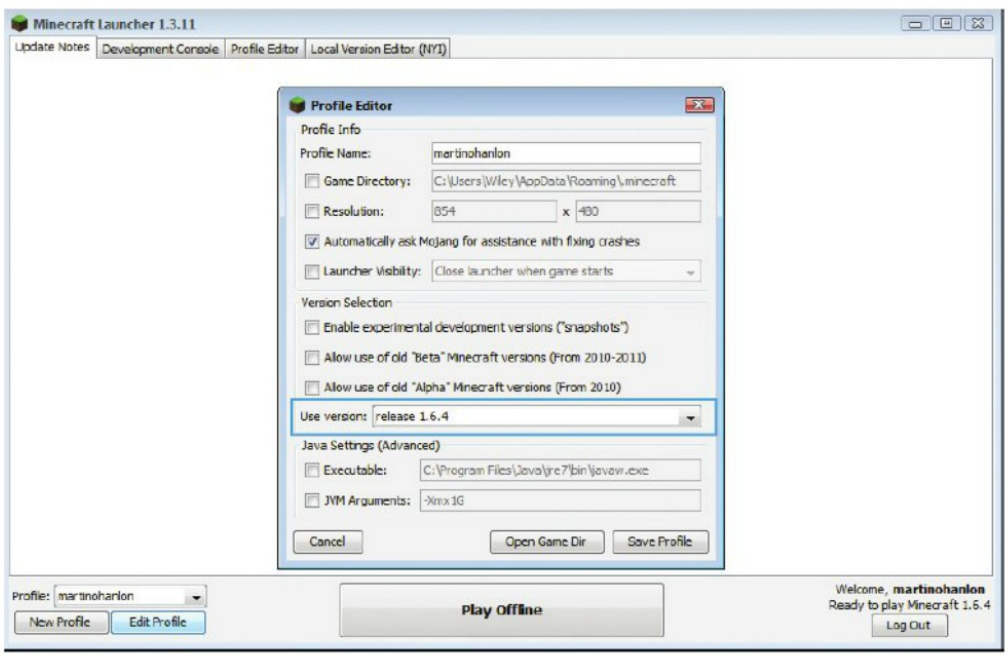


FIGURE 1-7 Choisis la version de Minecraft que tu souhaites utiliser.

Copyright © 2016 Eyrolles.

Maintenant, connecte Minecraft au serveur Bukkit.

1. Clique sur `Play` dans le Launcher de Minecraft pour ouvrir le jeu.
2. Dans le menu `Minecraft` qui s'affiche, clique sur `Multijoueur`.

> Si tu utilises un PC sous Windows, tu verras apparaître une boîte de dialogue d'avertissement te demandant d'autoriser Minecraft à communiquer avec le réseau. Clique sur `Autoriser l'accès`.
>
> 
>

3. Dans le menu `Multijoueur`, clique sur `Connexion rapide`.
4. Saisis `localhost` dans le champ `Adresse du serveur` et clique sur `Rejoindre` (voir la figure 1-8). C'est fait ! Tu vas désormais pénétrer dans le monde Minecraft que tu as créé sur le serveur Bukkit. T'y voilà enfin.



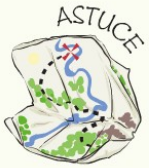
FIGURE 1-8 Connecte Minecraft au serveur Bukkit.

Arrêter Bukkit

Lorsque tu auras terminé ton tour de jeu pour la journée, tu devras te déconnecter du serveur Bukkit. Pour cela, appuie sur la touche `Échap` lorsque tu es dans Minecraft pour revenir sur la page du menu, puis clique sur `Quitter le jeu`. Tu peux désormais éteindre Bukkit en toute sécurité en écrivant `stop` dans l'interpréteur de commandes de Bukkit et en appuyant sur la touche `Entrée`.

Copyright © 2016 Eyrolles.

Tu verras alors une série de mises à jour Bukkit s'afficher à l'écran, t'indiquant tout ce que le serveur exécute avant de s'éteindre. Lorsque l'interpréteur indiquera « Closing listening thread », il se peut que tu voies le message « GRAVE » s'afficher. Surtout, pas de panique ! Cela est tout à fait normal et n'aura aucun impact sur ton ordinateur ou tes programmes Minecraft.



Tu peux saisir du texte dans l'interpréteur de commandes Bukkit pour contrôler plusieurs aspects du jeu Minecraft, tels que l'heure de la journée ou la météo. Par exemple, si le soleil se couche dans ton monde Minecraft, mais que tu n'es pas encore prêt pour affronter la nuit, saisis `set time 1` dans l'interpréteur de commandes pour changer l'horaire et revenir au matin. S'il commence à pleuvoir, fais disparaître les nuages en écrivant `weather clear`. Tu pourras trouver une liste complète des commandes Bukkit en écrivant `help`.

Créer un programme

Félicitations ! Tu viens de configurer ton ordinateur avec succès et Minecraft fonctionne correctement. Ces étapes de configuration t'auront peut-être semblé fastidieuses, mais elles sont bel et bien terminées et tu ne devras les renouveler que si tu souhaites utiliser un ordinateur différent. Passons maintenant aux choses sérieuses. Tu vas créer ton tout premier programme « Bonjour le monde de Minecraft ».



Dans les prochains chapitres, les instructions te suggéreront directement de démarrer IDLE sans te rappeler toutes ces étapes. Si tu as oublié comment le faire, il te suffira de relire cette section.

Tout d'abord, tu dois lancer Python et ouvrir IDLE en effectuant ceci :

- sur Raspberry Pi : double-clique sur l'icône `IDLE` (et non IDLE3) qui se trouve sur ton bureau ;
- sur PC : clique sur le menu `Démarrer>Tous les programmes>Python 2.7>IDLE (Python GUI)` ;
- sur PC (Windows 8) : clique sur le menu `Démarrer`, puis sur l'onglet `IDLE (GUI Python)` de l'écran d'accueil ou utilise la fonction `Rechercher` pour trouver IDLE ;
- sur Mac : clique sur `Aller` dans la barre de menus `Finder>Applications>Python 2.7`. Double-clique sur `IDLE`.

Copyright © 2016 Eyrolles.

L'interpréteur de commandes Python va alors apparaître.

Si tu utilises un Raspberry Pi, l'interpréteur de commandes Python prendra éventuellement quelques secondes avant de s'afficher.



Au premier démarrage d'IDLE, il se peut que ton ordinateur affiche un message te demandant la permission de lancer le programme ou d'autoriser IDLE à accéder au serveur. C'est normal. Tu dois simplement accepter.



Une fois que l'interpréteur de commandes Python s'est affiché, tu peux commencer à créer de nouveaux programmes dans IDLE. Celui que tu es sur le point de créer n'a rien de bien extraordinaire. Il s'agit, comme tout bon programmeur informatique, de créer un programme de test « Bonjour le monde de Minecraft » pour vérifier que tout est bien installé sur ton ordinateur. Voilà comment faire :

Dans les prochains chapitres, il te sera directement indiqué de créer un nouveau programme et de l'enregistrer dans `MyAdventures`. Si tu as oublié comment le faire, il te suffira de relire ces quelques instructions.



1. Crée un nouveau fichier en cliquant sur `File>New File` dans le menu `IDLE`. Remarque : sur Raspberry Pi, `New File` correspond à `New Window`.
2. Enregistre le fichier dans le dossier `MyAdventures` en cliquant sur `File>Save` dans le menu de l'éditeur IDLE.
3. Sélectionne le dossier `MyAdventures` :

   - Sur Raspberry Pi : sélectionne `/home/pi` dans la liste déroulante du `Répertoire` et double-clique sur `MyAdventures` dans l'explorateur de dossiers.
   - Sur PC : clique sur `Bureau` dans le volet de navigation à gauche, puis double-clique sur `AdventuresInMinecraft` dans l'explorateur de dossiers et double-clique sur `MyAdventures`.

Copyright © 2016 Eyrolles.

- Sur Mac : clique sur `Bureau` dans l'explorateur de dossiers, puis clique sur `AdventuresInMinecraft` et sur `MyAdventures`.

4. Tu dois maintenant nommer ton fichier. Saisis le nom `BonjourMondeDeMinecraft.py` et clique sur `Save`. L'extension `.py` indique à ton ordinateur que le fichier est un programme Python.



Il est important que tu enregistres le programme dans le dossier `MyAdventures`, car c'est l'emplacement où tu enregistreras tous les programmes de ce livre. Il contient en outre tout ce dont tu as besoin pour faire fonctionner tes programmes.

5. Et maintenant, à tes programmes ! Commence à rédiger ton programme « Bonjour le monde de Minecraft » en saisissant le code qui suit dans l'interpréteur de commandes IDLE. Respecte bien les majuscules et les minuscules, car le langage Python est **sensible à la casse** :

```
import mcpi.minecraft as minecraft
mc = minecraft.Minecraft.create()
mc.postToChat("Bonjour le monde de Minecraft")
```



Python est un langage de programmation **sensible à la casse**, c'est-à-dire que tu dois saisir le code en respectant les minuscules et les majuscules. Par exemple, Python ne traitera pas le terme *Minecraft* écrit avec une majuscule de la même manière que *minecraft* écrit en minuscules. De plus, si tu les épèles mal, le code donnera des résultats incorrects ; tu devras alors revenir en arrière pour savoir où l'erreur s'est glissée.



Tu apprendras ce que signifie ce code et ce à quoi il sert dans le prochain chapitre. Pour l'instant, contente-toi de faire apparaître les termes « Bonjour le monde de Minecraft » à l'écran pour vérifier que tout fonctionne correctement.

6. Enregistre ton programme en sélectionnant `File>Save` dans le menu de l'éditeur IDLE.

Copyright © 2016 Eyrolles.

Exécuter un programme

Dans les prochains chapitres, les instructions te suggéreront directement d'exécuter un programme Python sans te rappeler toutes ces étapes. Si jamais tu as oublié comment le faire, il te suffira de relire cette section.



Bravo ! Tu viens de créer un programme. Il est temps de le tester. Pour cela, tu dois indiquer à IDLE de l'exécuter, de la façon suivante.

1. Avant d'exécuter le programme « Bonjour le monde de Minecraft », tu dois avoir allumé Minecraft et démarré un jeu. Si ce n'est pas le cas, fais-le en t'aidant des instructions qui t'ont été données plus haut.
2. Redéfinis la taille de ta fenêtre Minecraft pour pouvoir voir les fenêtres du jeu et de l'éditeur IDLE en même temps. Si tu utilises un PC ou un Mac et que Minecraft s'affiche en plein écran, appuie sur la touche `F11` pour sortir du mode plein écran et afficher Minecraft dans une fenêtre. Tu devrais voir les fenêtres suivantes :

    - sur Pi : l'interpréteur de commandes Python, l'éditeur de texte IDLE affichant ton programme `BonjourMondeDeMinecraft.py` et la fenêtre de Minecraft (voir la figure 1-9) ;

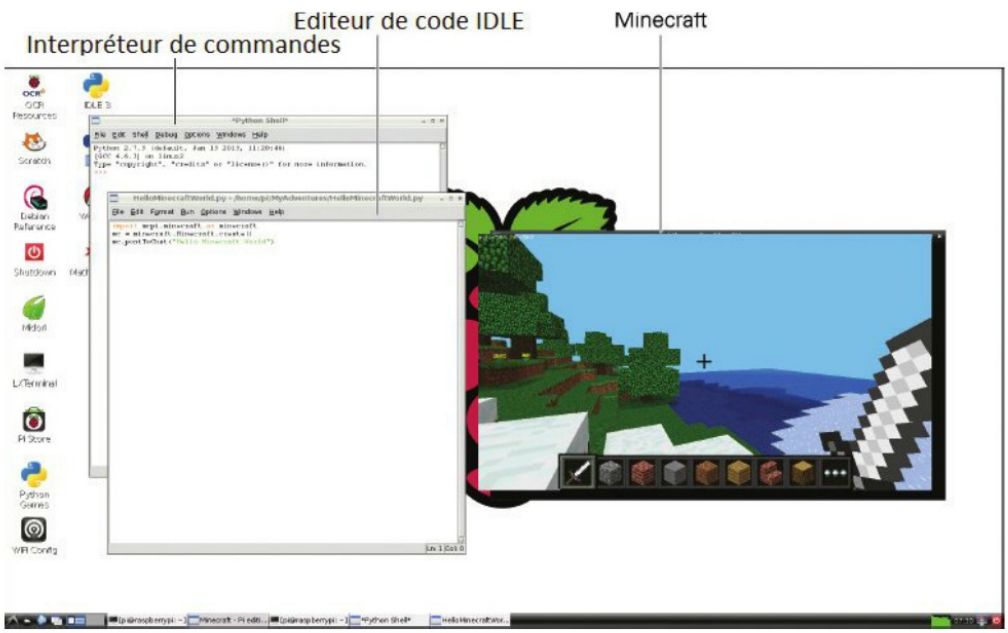


FIGURE 1-9 Raspberry Pi est prêt à exécuter ton programme.

Copyright © 2016 Eyrolles.

- sur PC : l'interpréteur de commandes Python, l'éditeur de texte IDLE affichant ton programme `BonjourMondeDeMinecraft.py`, l'interpréteur Bukkit et la fenêtre de Minecraft (voir la figure 1-10) ;

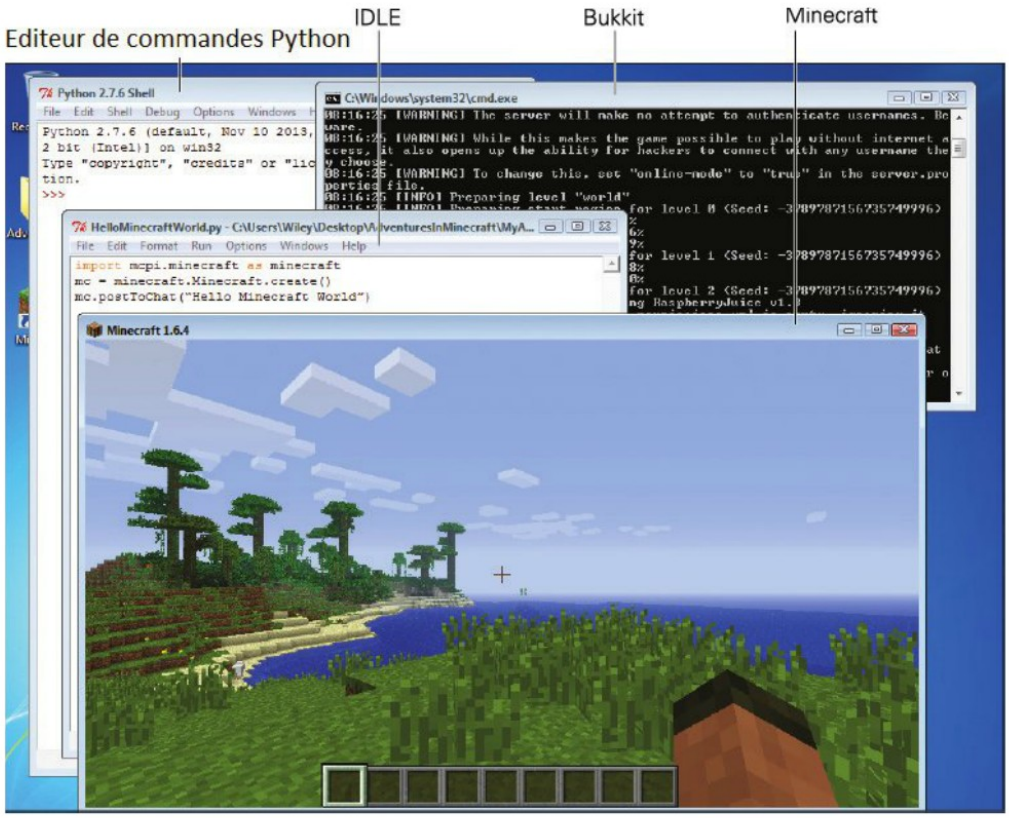


FIGURE 1-10 Ton PC est prêt à exécuter ton programme.

- sur Mac : l'interpréteur de commandes Python, l'éditeur de texte IDLE affichant ton programme `BonjourMondeDeMinecraft.py`, l'interpréteur Bukkit et la fenêtre de Minecraft (voir la figure 1-11).

3. À présent, appuie sur la touche `Échap` pour ouvrir le menu de Minecraft et sélectionne l'éditeur de texte IDLE avec ta souris.

4. Exécute le programme `BonjourMondeDeMinecraft.py` soit en cliquant sur `Run>Run Module` dans le menu `IDLE`, soit en appuyant sur la touche `F5`.

5. IDLE passera alors automatiquement à l'interpréteur de commandes Python et exécutera le programme. Si le code contient des erreurs, tu les verras s'afficher en rouge. Dans ce cas, prends soin de bien vérifier le code que tu as saisi dans l'éditeur de texte IDLE et de le comparer au code fourni dans le livre pour voir où se situe l'erreur.

6. Mets la fenêtre du jeu Minecraft au premier plan. Y remarques-tu quelque chose ? Tes efforts ont porté leurs fruits et tu peux désormais voir « Bonjour le monde de Minecraft » s'afficher dans le chat. Cela devrait ressembler à la figure 1-1.

Copyright © 2016 Eyrolles.

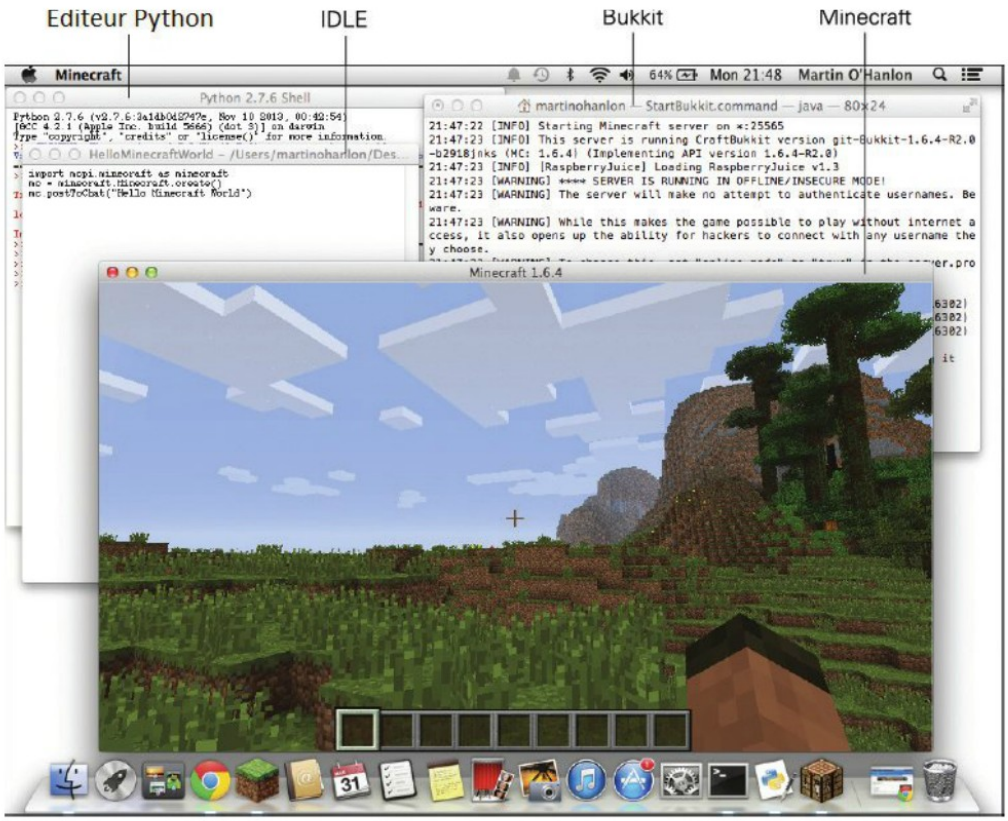


FIGURE 1-11 Ton Mac est prêt à exécuter ton programme.

Si tu as tout configuré correctement, le message « Bonjour le monde de Minecraft » s'affichera dans le chat, comme sur la figure 1-1. Si ce n'est pas le cas, tes erreurs s'afficheront dans l'interpréteur de commandes Python. Pour les corriger, n'hésite pas à relire les instructions de ce chapitre. Il est très important que tu configures ton ordinateur correctement dès le début. Si tu ne le fais pas, les programmes que tu rédigeras plus tard ne fonctionneront pas.



Arrêter un programme

Dans les prochains chapites, les instructions te suggéreront directement d'arrêter le programme sans te rappeler ces étapes. Si jamais tu as oublié comment faire, il te suffira de relire cette section.



Copyright © 2016 Eyrolles.

Tu viens d'exécuter ton programme « Bonjour le monde de Minecraft ». Celui-ci affiche le message à l'écran et s'arrête aussitôt. Cependant, il est important que tu saches comment arrêter un programme de force, car dans les prochains chapitres, ceux que tu exécuteras ne s'arrêteront pas sans recevoir d'instruction de ta part ! Pour arrêter un programme, sélectionne la fenêtre de l'interpréteur de commandes Python et clique sur `Shell>Restart Shell` dans le menu ou maintiens les touches `Ctrl+F6` appuyées.

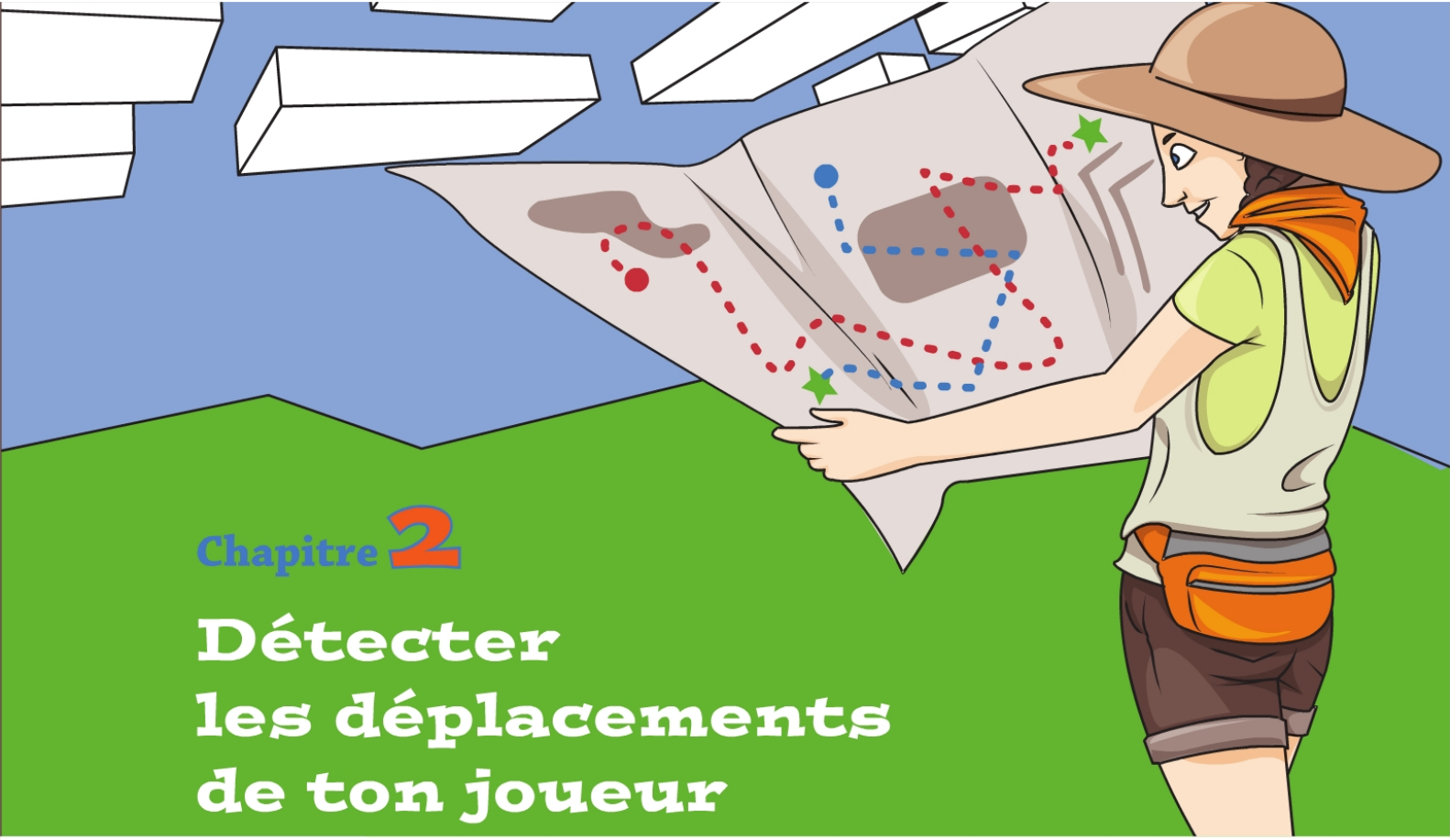


Niveau terminé : Félicitations ! Tu viens de réaliser le plus difficile. Tu as installé et lancé Minecraft et tu as rédigé ton tout premier programme pour saluer tout le monde.

Dans le prochain chapitre...

Dans le chapitre 2, tu apprendras quelques notions de programmation fondamentales dans Minecraft et Python, qui te permettront notamment de localiser ton joueur. Tu aborderas également le thème du « jeu dans le jeu » tout en créant des jeux simples qui réagissent différemment selon l'emplacement de ton joueur dans le monde de Minecraft.

Copyright © 2016 Eyrolles.



Chapitre 2

Détecter les déplacements de ton joueur

LORSQUE TU JOUES à Minecraft, tu découvres un monde que d'autres personnes ont conçu pour toi. Ce monde est amusant, mais il le serait bien plus encore si tu pouvais l'influencer à ta manière. Minecraft est fait pour ça. Le jeu t'offre l'environnement de programmation idéal pour laisser libre cours à ton imagination débordante et inventer des choses incroyables grâce au langage Python. Ce chapitre va te montrer comment profiter de cet environnement pour concevoir des choses amusantes en utilisant Python.

L'une d'elles consistera à créer un jeu dans le jeu, Minecraft étant l'univers que tu influenceras grâce à ton **programme**. En progressant, tu verras que la plupart des programmes Minecraft se composent de trois éléments principaux qui les rendent particulièrement amusants et captivants, à savoir : détecter des éléments du monde, par exemple la position de ton joueur, calculer des opérations, comme les résultats d'un jeu, et animer des choses, telles que les déplacements de ton joueur.

Dans ce chapitre, tu verras comment localiser ton joueur dans le monde et provoquer des actions de jeu en fonction de tes mouvements. Dans le jeu « Bienvenue à la maison », tu apprendras à créer un paillasson magique qui te souhaitera la bienvenue lorsque tu marcheras dessus. Nous t'initierons aussi à la technique du géorepérage en utilisant les blocs barrières de Minecraft. Et le jeu que tu vas créer te mettra au défi de récupérer des objets plus rapidement que tes amis. Enfin, tu apprendras à propulser ton joueur dans les airs si celui-ci reste bloqué dans un enclos trop longtemps !

Copyright © 2016 Eyrolles.



Un **programme** est une série d'instructions que tu rédiges dans un langage informatique spécifique et que l'ordinateur respecte à la lettre. Les instructions varient selon le langage de programmation et tu dois t'assurer de les écrire correctement. Dans ce livre, nous utilisons le langage Python.



Une **instruction** est un terme informatique général qui signifie que tu commandes à ton ordinateur d'effectuer une action complète, en rédigeant par exemple une ligne de code comme `print("Bonjour Steve")`. Certains préfèrent employer le terme « commande », mais il est davantage utilisé pour désigner les ordres que tu saisis dans une invite de commandes pour que ton ordinateur exécute une action.

Le langage Python a sa propre définition du terme « instruction », mais pour les besoins de ce livre nous l'emploierons pour désigner un ordre ou une simple ligne de programmation écrite dans ce langage. Si tu souhaites en savoir plus sur le sujet, tu peux consulter la documentation en ligne en te rendant sur : `https://docs.python.org/2/`

Localiser ton joueur

Avant de commencer à localiser ton joueur, tu dois d'abord savoir comment le monde de Minecraft est conçu. Ce monde est uniquement composé de blocs, chacun étant un cube d'un mètre carré. Lorsque ton joueur, Steve, se déplace, le jeu connaît la position de ton joueur grâce à une série de **coordonnées**. Le monde est composé de blocs remplis de matériaux dont le volume correspond généralement à un mètre cube. Il existe quelques exceptions, comme les tapis et les liquides tels que l'eau ou la lave, qui occupent un volume plus petit. Mais ils ne peuvent remplir un volume inférieur à un mètre cube lorsqu'on les mélange à d'autres matériaux.



Une **coordonnée** est un ensemble de nombres qui indiquent une position. Dans Minecraft, les coordonnées 3D sont utilisées pour indiquer la position exacte de ton joueur dans le jeu en trois dimensions, chaque coordonnée contient donc trois nombres. Pour en savoir plus sur les données de localisation, tu peux consulter l'article `http://en.wikipedia.org/wiki/Coordinate_system`. Et si tu souhaites obtenir davantage d'informations sur la manière dont Minecraft traite ces données, rends-toi sur la page `http://minecraft.gamepedia.com/Coordinates`.

Copyright © 2016 Eyrolles.

Connaître la position de ton joueur dans le monde de Minecraft te permet de provoquer des réactions intelligentes lorsque tu te déplaces. Par exemple, tu peux afficher des messages dans le chat à des endroits précis, bâtir des structures automatiquement lorsque tu es en mouvement et changer la position de ton joueur lorsque tu te rends dans certains endroits. Ce livre t'apprendra tous les secrets de ces formidables techniques de jeu, dont la plupart se trouvent précisément dans ce chapitre !

Nous appellerons ces coordonnées x, y et z. Tu dois faire très attention lorsque tu parles de coordonnées de position dans Minecraft, notamment si tu utilises des termes comme « gauche » et « droite », car la position des objets dépend de la direction vers laquelle tu te tournes. Il est préférable de penser ta position comme si tu disposais d'une boussole. La figure 2-1 montre comment les coordonnées x, y et z évoluent en fonction de tes déplacements dans le jeu.



Tu comprendras plus facilement comment tes coordonnées de position évoluent si tu les étudies directement dans Minecraft. Sur Raspberry Pi, tu peux voir tes coordonnées x, y et z s'afficher en haut à gauche de ton écran lorsque tu te déplaces. Sur PC ou Mac, il te suffit d'appuyer sur la touche `F3` pour les afficher. Pour en savoir plus sur les contrôles avancés du jeu, rends-toi sur la page `http://minecraft.gamepedia.com/Controls`.

- x augmente à mesure que tu te déplaces vers l'est et diminue lorsque tu vas vers l'ouest.
- y augmente lorsque tu te rapproches du ciel et diminue lorsque tu descends.
- z augmente à mesure que tu te déplaces vers le sud et diminue lorsque tu vas vers le nord.

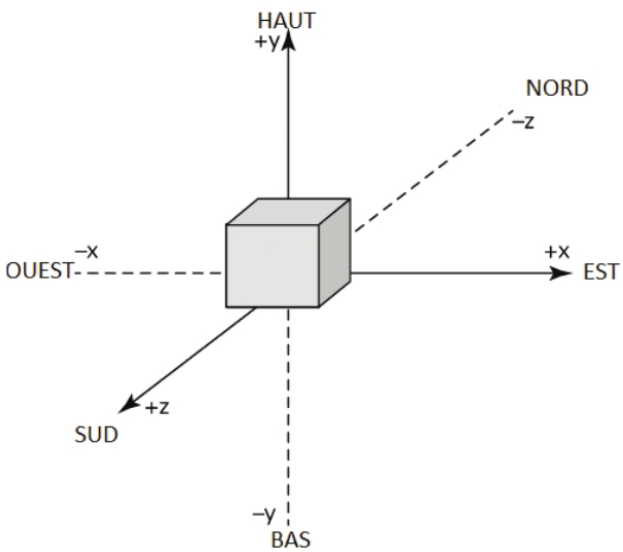


FIGURE 2-1 Les coordonnées x, y et z dans Minecraft correspondent aux pôles d'une boussole comme celle-ci.

Copyright © 2016 Eyrolles.

Te préparer pour l'aventure

Maintenant que tu es parvenu à exécuter le programme « Bonjour le monde de Minecraft » dans le premier chapitre, il est temps que tu rédiges ton propre programme et que tu te mettes à jouer !

Dans le chapitre 1, nous t'avons expliqué comment faire fonctionner tous tes outils. À présent, tu vas perfectionner tout cela dans chaque nouveau chapitre. Toutes les étapes que tu dois connaître pour démarrer tes applications et commencer à rédiger des programmes sont décrites dans le chapitre 1, mais pour te rafraîchir la mémoire au début, nous effectuerons quelques rappels.



Si tu souhaites te rappeler comment faire fonctionner Minecraft sur ta machine, rends-toi sur le site Internet du livre : `www.editions-eyrolles.com/dl/14292` et regarde la vidéo intitulée « Adventure 1 » (en anglais).

1. Lance Minecraft, IDLE ainsi que le serveur si tu es sur un PC ou un Mac. Tu t'es déjà certainement un peu familiarisé avec le processus, mais n'hésite pas à relire le chapitre 1 si tu as besoin de te rafraîchir la mémoire.
2. Démarre IDLE, l'environnement de développement intégré (EDI) du langage Python, comme tu l'as fait dans le chapitre 1. C'est là que tu rédigeras et exécuteras tous tes programmes ; c'est en somme ta fenêtre sur l'univers de Python.

Afficher la position de ton joueur

Pour bien comprendre comment fonctionnent les coordonnées de localisation, le mieux est encore de rédiger un petit programme pratique. Celui que nous allons écrire dès maintenant permet d'afficher les coordonnées 3D de ton joueur à l'écran, même lorsqu'il est en mouvement.

1. Dans l'interpréteur de commandes Python, crée un nouveau programme via `File>New File`. Une nouvelle fenêtre va alors s'ouvrir dans laquelle tu vas rédiger le code de ton nouveau programme.
2. Avant de commencer à écrire un programme, nous te conseillons toujours d'enregistrer le fichier. Ainsi, si tu as besoin de quitter ton poste de travail, tu n'auras pas à lui trouver un nom à la hâte. Choisis donc de l'enregistrer, c'est ce que font tous les professionnels pour s'assurer de ne pas perdre leur travail. Pour cela, choisis `File>Save As` dans le menu de l'interpréteur de commandes Python, puis saisis le nom de ton programme : `ouSuisJe.py`. Assure-toi de bien enregistrer ton fichier dans le dossier `MyAdventures`.
3. Ton programme communiquera directement avec le jeu par le biais de l'interface de programmation (**API**) de Minecraft. Pour accéder à cette API, tu devras

Copyright © 2016 Eyrolles.

importer le module *minecraft* qui permet à ton programme Python d'accéder à toutes les données du jeu. Utilise le code suivant :

```
import mcpi.minecraft as minecraft
```

4. Pour communiquer avec un jeu en cours d'exécution, tu dois t'y connecter. Pour cela, saisis la ligne suivante :

```
mc = minecraft.Minecraft.create()
```

> Le langage Python est sensible à la casse. Prends donc garde à bien écrire les majuscules et les minuscules correctement ; c'est très important. Dans l'instruction que tu viens de saisir, le deuxième terme « Minecraft » doit commencer par une majuscule. Sans cela, l'instruction ne fonctionnera pas.
>
> 
>

5. Ensuite, demande à Minecraft de te fournir les données de position de ton joueur en utilisant `getTilePos()`, comme ceci :

```
pos = mc.player.getTilePos()
```

6. Enfin, demande-lui d'afficher les coordonnées de ton joueur. `print()` les affichera dans l'interpréteur de commandes Python lorsque tu exécuteras le programme :

```
print(pos.x)
print(pos.y)
print(pos.z)
```

7. Enregistre ton fichier en faisant `File>Save` dans le menu de l'éditeur de texte.
8. Puis exécute ton programme en cliquant sur `Run>Run Module`.

Tu devrais maintenant voir s'afficher les coordonnées de ton joueur dans l'interpréteur de commandes Python (voir la figure 2-2). Plus loin dans ce chapitre, tu utiliseras ces coordonnées pour connaître la position de ton joueur dans le monde et construire un paillasson magique qui te souhaitera la bienvenue lorsque tu marcheras dessus !

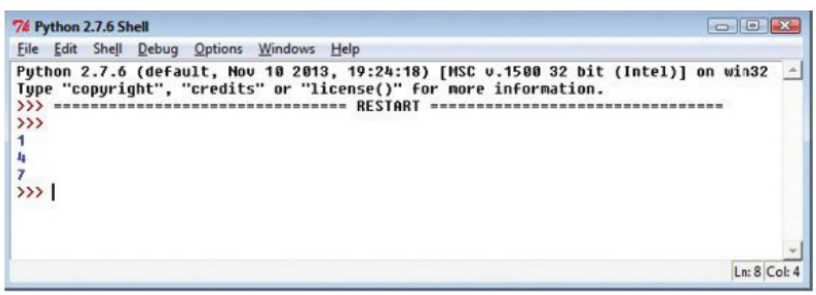


FIGURE 2-2 Utilise getTilePos pour afficher la position de ton joueur dans l'interpréteur de commandes Python.

Copyright © 2016 Eyrolles.



L'acronyme **API** *(Application Programming Interface)* signifie « interface de programmation ». Une API te permet d'accéder en toute sécurité à certaines parties du programme d'une application depuis tes propres programmes. En l'occurrence, c'est en utilisant l'API de Minecraft que tu peux accéder au jeu dans tes programmes Python. Mais avant de connecter tes programmes Python au jeu via cette API, tu dois tout d'abord démarrer Minecraft. Le terme `mpci` dans la ligne `import mcpi.minecraft as minecraft` signifie *Minecraft Pi*, car la toute première version de l'API ne pouvait s'exécuter que sur Raspberry Pi.

Une **interface** est un ensemble de règles qui indiquent comment, en tant que programmeur, tu peux accéder à certaines parties du système d'exploitation d'un ordinateur. L'API est un ensemble de règles qui t'indiquent comment communiquer avec un programme en cours d'exécution – ici, le jeu Minecraft. Tous tes programmes Minecraft auront accès au jeu Minecraft en cours d'exécution via l'API de Minecraft.



Tous les programmes Python que tu vas créer devront accéder à l'API de Minecraft. Pour cela, tes programmes doivent être en mesure d'accéder au module *mcpi.minecraft* stocké dans le dossier `mcpi` de ton dossier `MyAdventures`. Pour garantir cet accès, tu dois toujours enregistrer tes programmes Python dans le dossier `MyAdventures`.

PERCER LES SECRETS DU CODE

Félicitations, tu viens d'utiliser une variable dans ton programme ! `pos.x` est une variable, au même titre que `pos.y` et `pos.z`.

Une **variable** est tout simplement un nom que tu donnes à un fragment de mémoire de ton ordinateur. Lorsque tu places un nom à gauche d'un signe égal, cela crée une variable, à laquelle tu affectes une valeur, comme dans :

```
a = 10
```

Si tu utilises une variable à l'intérieur d'une **instruction** `print`, le programme affichera la valeur de cette variable dans l'interpréteur de commandes Python. Donc, si *a = 10*, alors l'instruction `print(a)` affichera le nombre 10 – la valeur affectée à la variable `a` – lorsque tu y feras appel.

La variable `pos` que tu utilises dans `pos = mc.player.getTilePos()` est un type spécial de variable. Imagine qu'il s'agit simplement d'une boîte composée de trois compartiments nommés x, y et z : `print(pos.x)`

Copyright © 2016 Eyrolles.

Déplace ton joueur à un autre endroit et exécute le programme à nouveau. Regarde maintenant les coordonnées qui s'affichent dans l'interpréteur de commandes Python ; celles-ci devraient avoir changé. Tu pourras utiliser ce petit programme quand tu le souhaiteras, notamment lorsque tu auras besoin de confirmer ta position dans le monde de Minecraft.



Une **variable** est une donnée que tu stockes dans la mémoire de ton ordinateur. Tu peux attribuer de nouvelles valeurs à tes variables n'importe où dans tes programmes. Tu peux également les rappeler et les afficher ou les utiliser dans des calculs. Pour te donner une meilleure idée de ce qu'est une variable, imagine qu'il s'agit de la touche mémoire d'une calculatrice.

Améliorer l'affichage de ta position

Avant de continuer, tu as la possibilité d'améliorer l'affichage de la position de ton joueur pour qu'elle soit plus facile à comprendre. En effet, placer les résultats sur trois lignes distinctes prend de la place. Il serait préférable de pouvoir les lire sur une seule et même ligne, comme ceci :

```
x=10 y=2 z=20
```

Pour cela, suis ces indications ci-dessous.

1. Modifie ton programme `ouSuisJe.py` en effaçant les trois instructions `print()` et en les remplaçant par cette ligne :

```
print("x="+str(pos.x) + "y="+str(pos.y) + "z="+str(pos.z))
```

2. Sauvegarde ton programme via `File>Save` dans le menu de l'éditeur, puis exécute-le en cliquant sur `Run>Run Module`. Regarde comment le résultat s'affiche. Il apparaît désormais sur une seule ligne. N'est-ce pas plus clair ?

PERCER LES SECRETS DU CODE

Dans la nouvelle version de ton programme `ouSuisje.py`, tu as réalisé un petit tour de magie que nous devons éclaircir ici.

En Python, l'instruction `print()` affiche tout ce que tu places entre les parenthèses et le résultat apparaît dans l'interpréteur de commandes Python. Tu as déjà affiché plusieurs informations à l'aide de cette instruction. Prends le temps de les examiner un peu, en commençant par cet exemple :

```
print("bonjour")
```

Copyright © 2016 Eyrolles.

Comme tu l'as vu, `print()` affiche le terme « bonjour ». Les guillemets droits autour du mot indiquent à Python que tu souhaites afficher « bonjour » exactement comme tu l'as écrit. Si tu saisis des espaces entre les guillemets, elles apparaîtront aussi. Le texte qui se trouve à l'intérieur des guillemets s'appelle une **chaîne de caractères**.

Voici un autre exemple d'utilisation de `print()` :

```
print(pos.x)
```

Dans cette instruction, `print()` affiche la valeur affectée à la variable `pos.x`. Dans ce cas, il s'agit d'un nombre, mais celui-ci peut varier à chaque fois que tu utilises `print()`, en fonction de la valeur que tu auras affectée à la variable affichée.

Enfin, tu dois comprendre le rôle de `str()` dans ton code. L'instruction `print()` te permet de combiner des nombres et des chaînes de caractères de plusieurs manières. Lorsque tu rédiges des combinaisons de nombres et de chaînes dans Python, comme dans le code ci-dessous, tu dois indiquer à Python de convertir les nombres en chaînes pour pouvoir les unir aux autres caractères de la chaîne. `str()` est construite à l'intérieur de Python et convertit toutes les variables ou valeurs comprises entre ses parenthèses en chaîne de caractères. Le symbole plus (`+`) indique à Python d'unir « `x=.` » à toute valeur affectée à la variable `pos.x` pour créer une chaîne complète, qui s'affiche ensuite dans la fenêtre de l'interpréteur de commandes Python.

```
print("x= " + str(pos.x))
```

Maintenant, essaye de saisir cette même ligne sans `str()` et observe ce qui se passe :

```
print("x= " + pos.x)
```



Une **chaîne de caractères** est une variable qui permet de stocker toute une suite de lettres, de nombres et de symboles. En clair, c'est comme si tu enfilais des perles sur un collier ou un bracelet, chaque perle étant ornée d'un nombre, d'une lettre ou d'un symbole différent, le tout dans ordre bien défini et toujours identique.

Les chaînes de caractères sont utiles dans bien des cas, notamment si tu souhaites enregistrer ton nom, une adresse ou un message à afficher sur le chat de Minecraft.

Copyright © 2016 Eyrolles.

Utiliser postToChat pour afficher ta position ailleurs

Avec le programme `ouSuisJe.py`, tu joues à Minecraft dans la fenêtre du jeu, mais ta position apparaît dans l'interpréteur de commandes Python, c'est assez gênant. Tu peux facilement y remédier en utilisant une technique que tu as déjà apprise dans le chapitre 1 : `postToChat` ! Voilà comment faire :

1. Remplace l'instruction `print()` par `mc.postToChat()`, comme ceci :

```
print("x=" + str(pos.x) + " y=" + str(pos.y) + " z=" +
      str(pos.z))
mc.postToChat("x=" + str(pos.x) + " y=" + str(pos.y) + " z="
      + str(pos.z))
```

2. Clique sur `File>Save` pour sauvegarder ton programme, puis exécute-le en cliquant sur `Run>Run Module` dans le menu de l'éditeur.

Félicitations ! La position de ton joueur s'affiche désormais dans le chat de Minecraft, ce qui doit te sembler bien plus pratique pour jouer.

Introduire une boucle de jeu

Tu en conviendras, exécuter ton programme `ouSuisJe.py` à chaque fois que tu souhaites savoir où tu es n'est pas très pratique. Heureusement, tu peux remédier à cela en modifiant ton programme et en y ajoutant une boucle de jeu. Presque tous les autres programmes que tu apprendras à rédiger dans ce livre incluent une boucle de jeu, c'est-à-dire un procédé qui leur permet d'interagir en continu avec Minecraft. De même, la plupart des jeux auxquels tu joues d'habitude continuent de tourner jusqu'à ce que tu les arrêtes. C'est une technique utile qu'il est bon d'acquérir. En informatique, on appelle cela une **boucle infinie**, car elle tourne sans jamais s'arrêter.



Une **boucle infinie** est une boucle qui ne s'arrête jamais. Elle tourne à l'infini et l'unique façon de l'arrêter est de mettre fin au programme Python lui-même. Pour cela, rends-toi dans l'interpréteur de commandes Python et clique sur `Shell>Restart Shell` dans le menu ou, lorsque tu te trouves dans l'interpréteur, maintiens les touches `Ctrl+F6` appuyées.

Pour ajouter la boucle, tu dois modifier le programme existant. Le nouveau programme sera très semblable, donc tu gagneras du temps. N'oublie pas de sauvegarder ton nouveau fichier sous un nom différent pour ne pas écraser le premier.

1. Commence tout d'abord par enregistrer ton nouveau fichier en sélectionnant `File>Save As` dans le menu de l'éditeur et appelle-le `ouSuisJe2.py`. Vérifie

Copyright © 2016 Eyrolles.

qu'il est bien enregistré dans le dossier `MyAdventures`, sinon il ne fonctionnera pas.

2. Tu dois maintenant importer un nouveau module qui te permettra de ralentir le temps. En effet, si tu ne ralentis pas la boucle, ton chat sera inondé de messages qui t'empêcheront de voir ce que tu fais. Ajoute la ligne en **gras** ci-dessous au programme existant :

```
import mcpi.minecraft as minecraft
import time
```

3. Modifie ensuite la partie principale de ton programme en ajoutant les lignes en gras suivantes, sans oublier d'indenter les lignes situées en dessous de `while True` pour indiquer à Python les instructions qu'il doit répéter. Réfère-toi à l'encadré « Percer les secrets du code » correspondant pour connaître la raison de cette indentation :

```
while True:
      time.sleep(1)
      pos = mc.player.getTilePos()
    mc.postToChat("x=" + str(pos.x) + " y=" + str(pos.y) ↩
+ " z=" + str(pos.z))
```

4. Va dans `File>Save` dans le menu de l'éditeur pour sauvegarder ton fichier.

5. Tu peux maintenant exécuter ton programme en sélectionnant `Run>Run Module` dans le menu de l'éditeur. Fais un tour dans le monde de Minecraft et tu verras tes coordonnées s'afficher dans le chat toutes les secondes.

PERCER LES SECRETS DU CODE

L'**indentation** est très importante dans tous les langages informatiques, car elle permet de mettre en évidence la structure des programmes et la manière dont s'imbriquent les blocs d'instructions. Mais elle est encore bien plus importante en Python, car le langage l'utilise pour comprendre la signification des programmes, contrairement aux autres langages, comme le C, qui utilise des crochets « {} » pour regrouper des instructions. Il faut notamment y faire attention lorsque tu utilises des boucles `while`, entre autres instructions, car elle indique quels blocs d'instructions appartiennent à la boucle et les répétera sans arrêt le cas échéant. Dans la boucle de jeu que tu as créée, toutes les instructions se trouvant en dessous de `while True:` sont indentées, car elles appartiennent à la boucle et seront répétées à chaque tour.

Dans l'exemple ci-dessous, tu peux remarquer que « bonjour » ne s'affiche qu'une seule fois, tandis que « tic » s'affiche chaque seconde. Quant aux instructions `time.sleep()` et `print()`, elles appartiennent à la boucle, car elles sont indentées :

Copyright © 2016 Eyrolles.

```
print("bonjour")
  while True: # ceci est le début de la boucle
      time.sleep(1)
      print("tic")
```

Si en Python l'indentation importe davantage que dans les autres langages de programmation, c'est parce qu'elle permet d'indiquer, à l'aide d'alinéas, à quelles boucles les instructions appartiennent. Si tu te trompes d'un niveau par exemple, c'est tout le sens de ton programme que tu modifies.

L'**indentation** est l'espace qui se situe à gauche, en marge de tes lignes de programmation. Les espaces d'indentation permettent de structurer le programme et de regrouper plusieurs instructions en blocs distincts, sous des boucles ou d'autres instructions. En Python, les niveaux d'indentation sont importants, car ils permettent aussi de modifier le sens des programmes.



Il est important que tu comprennes le rôle de l'indentation. Lorsque tu indentes ton code, tu peux à la fois utiliser des espaces et des alinéas, mais lorsque tu combines ces deux modes d'indentation dans un seul et même programme, cela peut vite semer la pagaille. Il est donc préférable de ne suivre qu'une seule méthode d'indentation. La plupart des gens emploient la touche `Tab` pour indenter leur texte, car cela va beaucoup plus vite que d'utiliser des espaces. Si tu suis ce choix, alors tu devras t'y tenir le reste du temps et ne pas utiliser d'espaces.



Maintenant que tu es parvenu à exécuter une boucle de jeu infinie, ton programme ne s'arrêtera plus ! Avant d'exécuter ton prochain programme, arrête celui que tu viens d'exécuter via `Shell>Restart Shell` dans le menu de l'interpréteur de commandes ou maintiens les touches `Ctrl+F6` appuyées.



Copyright © 2016 Eyrolles.

Bâtir le jeu Bienvenue à la maison

Il est temps de mettre en pratique cette technique de la boucle en développant une application simple. Le jeu « Bienvenue à la maison » détectera la position de ton joueur pour suivre ses déplacements dans le monde de Minecraft. Tu apprendras également à effectuer des repérages plus complexes à l'aide de l'instruction `if` en Python. Enfin, tu construiras un paillasson magique qui fera apparaître le message « Bienvenue à la maison » dans le chat dès lors que tu passeras dessus.



Tu peux regarder une vidéo didactique sur la création du jeu « Bienvenue à la maison » en te rendant sur le site Internet du livre `www.editions-eyrolles.com/dl/14292`. Choisis la vidéo « Adventure 2 » (en anglais).

Utiliser des instructions if pour créer un paillasson magique

Pour savoir si ton joueur se trouve sur le paillasson, tu utiliseras une instruction `if` qui te permettra de comparer la position de ton joueur avec celle du paillasson. Lorsque les deux positions seront identiques, cela voudra dire que ton joueur est arrivé à la maison.

Tout d'abord, pour observer comment l'instruction `if` fonctionne, fais l'essai par toi-même dans l'interpréteur de commandes Python.

1. Clique sur l'interpréteur de commandes Python. Veille à bien cliquer à droite des signes `>>>` pour t'assurer d'écrire ton code au bon endroit.
2. Saisis la ligne suivante dans l'interpréteur de commandes Python et celui-ci l'exécutera dès que tu appuieras sur la touche `Entrée`. Cela n'affichera rien, mais aura pour effet d'affecter le nombre `20` à la variable nommée `a` :

```
a = 20
```

3. Vérifie que la variable `a` contient la bonne valeur en saisissant l'instruction suivante dans l'interpréteur de commandes Python :

```
print(a)
```

   Le nombre `20` devrait apparaître à l'écran.

4. Essaye d'exécuter une instruction `if` pour voir si la valeur affectée à la variable `a` est supérieure à 10. Note que tu dois placer le deux-points à la fin de l'instruction `if`. Lorsque tu appuies sur `Entrée` à la fin de la première ligne, Python indente automatiquement la prochaine ligne pour toi :

Copyright © 2016 Eyrolles.

```
if a>10:
    print("grand")
```

5. Maintenant, appuie sur `Entrée` à nouveau pour indiquer à l'interpréteur de commandes que tu as terminé de saisir l'instruction `if`. Tu devrais voir le terme « grand » s'afficher dans l'interpréteur de commandes. La figure 2-3 te montre ce à quoi ton écran doit ressembler.

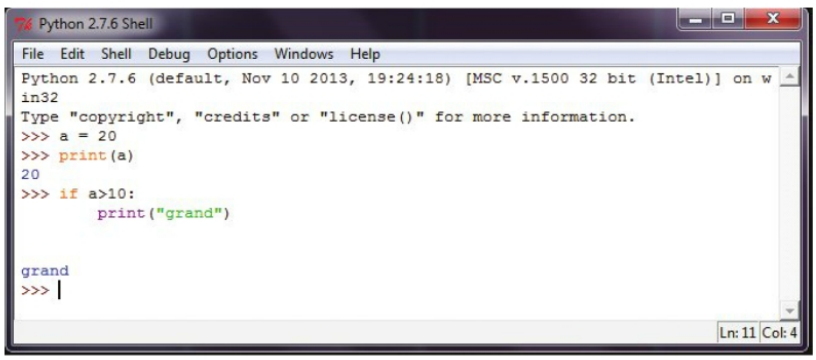


FIGURE 2-3 Utiliser une instruction if de façon interactive dans l'interpréteur de commandes Python

L'instruction `if` peut générer deux résultats. Par exemple, dans le code précédent, `if a>10` est soit `True` – c'est-à-dire « vrai » – (`a` est supérieur à `10`), soit `False` – « faux » – (`a` n'est pas supérieur à `10`). Les instructions `if` ne peuvent avoir que deux résultats possibles. Nous parlerons à nouveau des valeurs `True` et `False` un peu plus tard dans ce chapitre.



Connaître la position précise de ton joueur

Pour que ton programme puisse calculer si ton joueur franchit le paillasson magique, il devra vérifier au moins deux éléments de tes coordonnées. Si tu décèles que les coordonnées x et z du paillasson sont identiques à celles de ton joueur, tu pourras donc raisonnablement en conclure que ton joueur se trouve sur le paillasson. Fais-en l'expérience en utilisant le mot-clé `and` dans l'instruction `if` pour calculer les deux conditions en même temps. La coordonnée y importe moins ici. Nous n'allons donc pas la vérifier. Ainsi, même si ton joueur survole le paillasson, tu verras tout de même le message de bienvenue s'afficher.

Au préalable, effectue ces quelques étapes dans l'interpréteur de commandes Python pour t'assurer que tout fonctionne correctement.

Copyright © 2016 Eyrolles.

1. Après les signes `>>>`, saisis les informations suivantes pour définir une variable :

```
a = 8
```

2. Saisis ensuite une instruction `if` contenant le terme `and`. Cette instruction sert à déterminer si la variable `a` se situe entre deux nombres. La première fois que tu saisiras ce code, tu verras le terme « entre » s'afficher dans l'interpréteur de commandes Python, car 8 est supérieur à 5 et inférieur à 10. Souviens-toi d'appuyer sur `Entrée` une deuxième fois pour mettre fin à l'indentation après le `if` :

```
if a>5 and a<10:
    print("entre")
```

3. Maintenant, modifie la valeur de `a` pour qu'elle soit inférieure à 5 et regarde ce qui se passe (voir la figure 2-4) :

```
a = 3
if a>5 and a<10:
    print("entre")
```

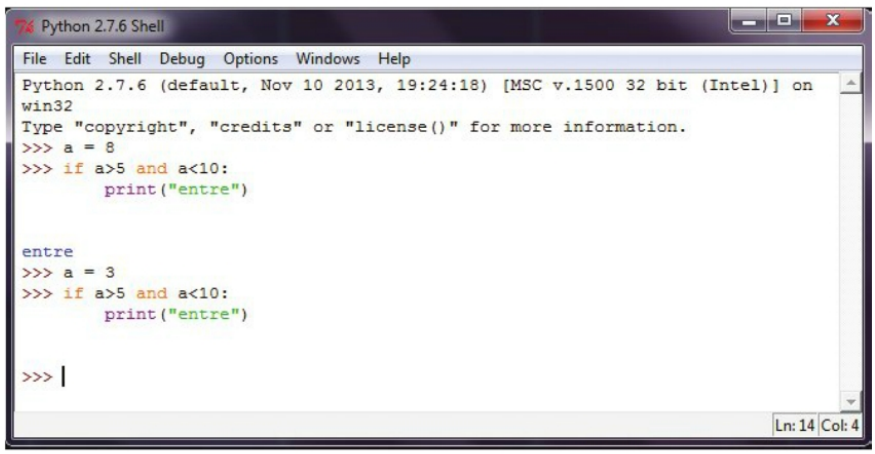


FIGURE 2-4 Tu utilises une instruction if avec and pour vérifier deux conditions.

DÉFI



Refais l'exercice précédent en employant des nombres différents, surtout des nombres situés entre 5 et 10. Avec quels nombres parviens-tu à afficher le terme « entre » ?

Copyright © 2016 Eyrolles.

Lorsque tu testes des programmes, il est toujours très utile d'employer des nombres différents. Pour t'éviter un nombre incalculable de tests, contente-toi de vérifier ton instruction `if` uniquement avec des nombres avoisinant les conditions. Dans l'exemple précédent, il faudrait effectuer le test avec les nombres 4, 5, 6 et 9, 10, 11. Si les tests s'avèrent concluants, nous pourrons raisonnablement penser que tous les autres nombres fonctionnent aussi.



Construire un paillasson magique

Avant de rédiger ton jeu, tu dois tout d'abord bâtir une structure dans le monde de Minecraft pour donner à ton programme un contenu avec lequel il va interagir. Tout ce dont tu as besoin, en somme, c'est d'un paillasson. Mais avant tout, démarre Minecraft et charge le monde que tu as déjà visité.

1. Pour placer un paillasson sur le sol, choisis un objet de l'inventaire et effectue un clic droit pour le placer sur le sol devant toi. Un paillasson en bois fera parfaitement l'affaire.
2. Pour connaître les coordonnées de ton paillasson dans le jeu, exécute le programme `ouSuisJe.py` à nouveau et place-toi sur le paillasson. Note les valeurs des coordonnées x, y et z sur un papier, car tu en auras besoin au moment de rédiger ton prochain programme pour définir la position de ton paillasson dans le monde de Minecraft.

Tu dois probablement avoir hâte de commencer à construire d'immenses structures ! Tu apprendras à le faire automatiquement dans le chapitre 3, mais pour l'heure construis une maison simple pour te concentrer sur le programme. Crée par exemple une charmante maison en t'assurant de placer le paillasson à l'entrée, sur le pas de la porte.

Écrire le jeu Bienvenue à la maison

Maintenant que tu as compris comment l'instruction `if` fonctionne, tu peux te lancer dans la rédaction du jeu « Bienvenue à la maison » en suivant les indications ci-après.

1. Va dans `File>New File` dans le menu de l'interpréteur de commandes Python pour créer un nouveau programme.
2. Sélectionne `File>Save As` dans le menu de l'éditeur pour sauvegarder ton programme et nomme-le `bienvenueALaMaison.py`. N'oublie pas d'enregistrer ton programme dans le dossier `MyAdventures` pour qu'il fonctionne.
3. Importe les modules dont tu as besoin pour ce programme en insérant les lignes suivantes :

```
import mcpi.minecraft as minecraft
import time
```

Copyright © 2016 Eyrolles.

4. Connecte-toi au jeu Minecraft, en respectant bien la casse du mot `Minecraft` :

```
mc = minecraft.Minecraft.create()
```

5. Introduis la boucle principale du jeu qui permet de détecter la position de ton joueur, en ajoutant du retard pour qu'elle ne s'exécute pas trop rapidement :

```
while True:
    time.sleep(1)
    pos = mc.player.getTilePos()
```

6. Ajoute l'instruction `if` qui permet de vérifier si ton joueur se trouve sur le paillasson, en utilisant `and` pour vérifier deux conditions. Les coordonnées x et z de ton paillasson doivent correspondre à la position de ton joueur pour que le programme puisse indiquer que celui-ci se trouve sur le paillasson. Tu as bien retenu les coordonnées de ton paillasson ? Écris-les dans ton code de cette manière pour que le programme puisse savoir où il se situe exactement :

```
    if pos.x == 10 and pos.z == 12:
        mc.postToChat("bienvenue à la maison")
```

Il est temps de voir si ton programme fonctionne ! Va dans `Run>Run Module` dans l'éditeur tout en te déplaçant dans Minecraft. Lorsque ton joueur franchit le pas de la porte, tu devrais voir apparaître « bienvenue à la maison », comme sur la figure 2-5. Super !

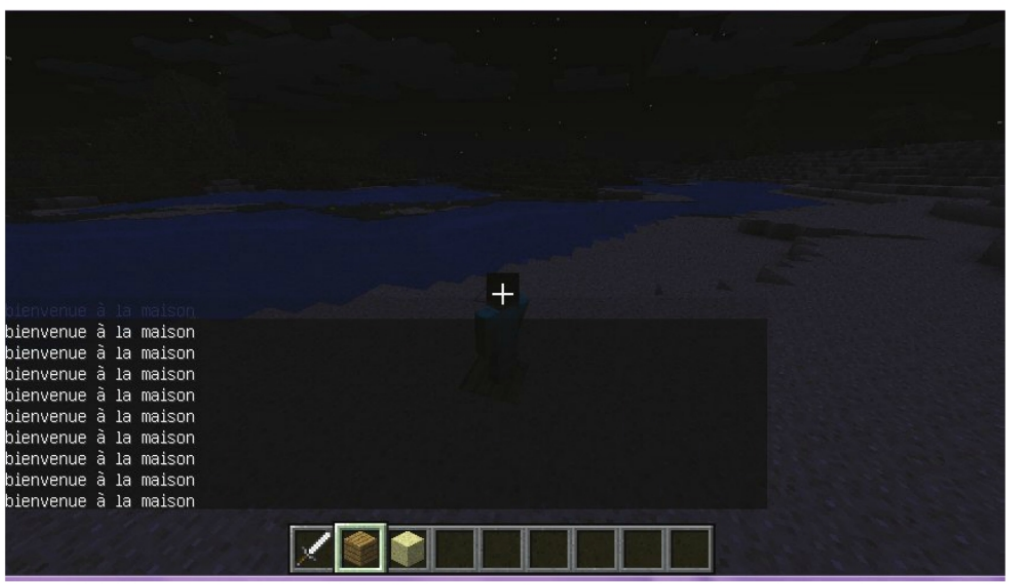


FIGURE 2-5 Lorsque tu franchis le pas de la porte, le message « bienvenue à la maison » s'affiche.

Copyright © 2016 Eyrolles.

Dans l'instruction `if`, as-tu remarqué le double symbole `==` ? Une erreur fréquente consiste à n'utiliser qu'un seul symbole `=`. Essaye de reproduire l'erreur et vois ce qu'il se passe en exécutant ton programme. Python utilise le signe égal (`=`) une fois pour dire : « affecte la valeur située à droite du signe égal à la variable située à sa gauche ». Par exemple : `a = 3`. En revanche, Python utilise le signe égal deux fois (`==`) pour dire : « compare la valeur de droite avec la valeur de gauche ». Par exemple : `a == 3`.



Fais bien attention à l'indentation lorsque tu codes ton jeu « Bienvenue à la maison ». Les instructions incluses dans la boucle `while True` sont indentées une fois. L'instruction `mc.postToChat()` appartenant à l'instruction `if` est indentée deux fois. Si tu te trompes dans l'indentation, ton programme sera erroné.



Que tu le croies ou non, tu viens d'acquérir les connaissances fondamentales de programmation que tu utiliseras plus loin dans ce chapitre. Chaque programme Minecraft comprend les étapes suivantes : importer des modules, se connecter au jeu, former une boucle, détecter un événement et réagir en fonction de celui-ci. À partir de ces simples fondements, tu seras capable d'accomplir de grandes choses !



DÉFI

Penses-tu qu'en vérifiant la coordonnée y (celle de la hauteur), tu parviendras à mieux détecter la position de ton joueur par rapport au paillasson ? Fais-en l'essai. Modifie ton programme pour vérifier chacune des trois coordonnées du paillasson et regarde si cela fonctionne mieux.



POUR ALLER PLUS LOIN

À un moment donné, il t'arrivera probablement d'introduire des erreurs dans ton code sans le vouloir. Tu verras alors un message d'erreur s'afficher dans l'interpréteur de commandes Python. Nous te conseillons donc d'analyser d'un peu plus près le type d'erreurs que tu pourrais commettre, pour ne pas t'inquiéter lorsque tu te trouveras dans cette situation.

Copyright © 2016 Eyrolles.

Il se peut par exemple que tu voies s'afficher une erreur de **syntaxe** si tu saisis mal un symbole ou si tu rédiges ton code de manière désordonnée. Tu obtiendras notamment un message d'erreur si tu omets un symbole ou si tu ajoutes une variable au mauvais endroit.

Par exemple, si tu saisis le signe égal seul dans ton interpréteur de commandes Python, tu verras s'afficher une erreur de syntaxe comme dans la figure 2-6, car ce signe est généralement utilisé pour des variables ou autres informations :

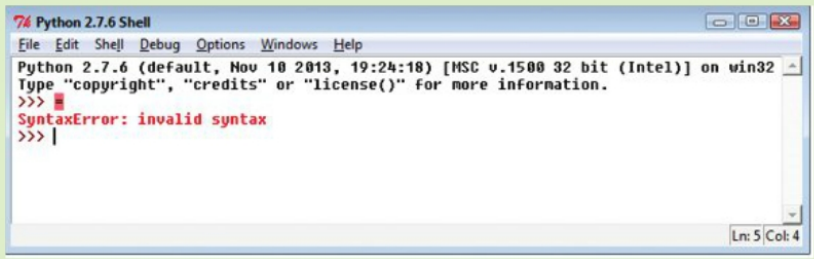


FIGURE 2-6 Python t'indique que ton code contient une erreur.

Maintenant, saisis la ligne suivante dans ton interpréteur de commandes Python :

```
print(zz)
```

Tu verras que Python te signale une erreur de nom, comme tu peux le voir sur la figure 2-7, car tu n'as pas encore affecté de valeur à la variable `zz`, donc Python n'est pas en mesure de reconnaître cette variable.

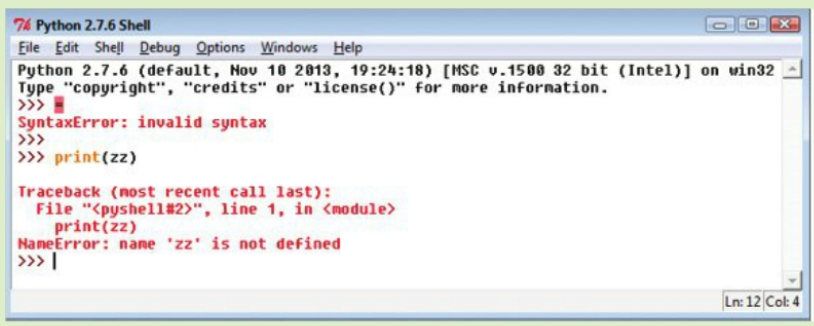


FIGURE 2-7 Python te signale une erreur de nom.

Essaye de saisir une boucle `while` en omettant volontairement d'insérer le deux-points (:) à la fin de la ligne :

```
while True
```

Là encore, tu obtiendras un message d'erreur de syntaxe, car tu n'as pas utilisé le deux-points alors que Python les exige.

Copyright © 2016 Eyrolles.

Ne t'inquiète pas lorsque tu vois un message d'erreur dans ton programme. Analyse attentivement la ligne concernée et tâche de corriger l'erreur que tu as commise. Parfois, tu obtiendras un message d'erreur de syntaxe pour une erreur commise plus haut dans le code. Veille donc à bien analyser l'ensemble des lignes concernées par le message pour savoir où se situe le problème. Si tu n'y parviens pas, demande à un ami de le faire pour toi. Il est souvent bien plus facile de repérer les erreurs des autres ! Lorsque des erreurs sont signalées dans l'interpréteur de commandes Python, le numéro de la ligne où elles se trouvent est également indiqué. Pour le connaître, regarde en bas à droite de la fenêtre `IDLE` pour repérer la ligne où se situe ton curseur, ou rends-toi dans `Edit> GoTo` Line dans le menu `IDLE`.

La **syntaxe** fait référence aux règles d'un langage (en l'occurrence, le langage Python) et à l'ordre dans lequel tu saisis les informations.



Utiliser le géorepérage pour prélever un impôt

Tu vas maintenant rédiger un nouveau jeu qui s'appelle `impot.py` et qui te permet de construire un pré cerné d'une barrière. Ton programme aura la possibilité de détecter lorsque ton joueur entrera dans cet enclos et lui prélèvera une somme d'argent tout le temps qu'il passera à l'intérieur. Le défi que tu vas devoir relever consistera à retirer des objets de l'enclos le plus vite possible pour éviter de recevoir une facture trop salée.

Dans ton programme `bienvenueALaMaison.py`, tu as utilisé une instruction `if` pour vérifier si les coordonnées de ton joueur étaient identiques à celles du paillasson. Pour le savoir, tu dois placer ton joueur précisément au même emplacement que le paillasson. Or, il est possible de réaliser cette tâche encore plus rapidement grâce à la technique dite de **géorepérage**. C'est ce que tu vas faire ici dans ton nouveau jeu. Détecter la position de ton joueur sera d'autant plus simple que ce programme scannera une région du monde de Minecraft au lieu de se focaliser sur des coordonnées précises. Ainsi, tu pourras localiser ton joueur en scannant des régions entières et n'auras plus à te soucier de retrouver des points précis dans l'espace.

Copyright © 2016 Eyrolles.



Plus généralement, le **géorepérage** est une technique qui consiste à encercler des coordonnées sur une carte. Lorsqu'un objet pénètre dans ce périmètre virtuel, quelque chose se produit en conséquence. L'objet en question peut être n'importe quoi. Il peut s'agir d'un joueur du monde de Minecraft ou, dans le monde réel, d'une personne, d'une machine comme une tondeuse à gazon ou encore d'un animal.

Dans bien des cas, la technique du géorepérage est utilisée pour avertir qu'un objet – un troupeau ou un véhicule par exemple – sort d'un périmètre défini. Pour en savoir plus sur l'utilisation du géorepérage en général, rends-toi sur la page `https://fr.wikipedia.org/wiki/Géorepérage`. Tu y découvriras comment le géorepérage est utilisé dans le monde réel, notamment pour suivre le trajet des éléphants dans la nature : `http://www.cbsnews.com/news/kenya-uses-text-messages-to-track-elephant/`

Pour que cette technique fonctionne, tu as besoin de deux choses : un objet ou une région à sonder et les coordonnées du contour de cet objet. Mais avant d'entrer dans le vif du sujet, construis un enclos à l'aide de barrières Minecraft et note les coordonnées de ses angles.



Tu n'es pas obligé de construire une barrière pour exécuter le programme, mais cela rendra le jeu un peu plus intéressant si tu le fais. En effet, une clôture t'empêche d'avancer, tu es obligé de sauter par-dessus ou de la longer pour trouver l'entrée de l'enclos. Or, ce sont les difficultés et les obstacles qui font tout l'attrait des jeux. Cette barrière Minecraft constituera les limites du périmètre de ton géorepérage virtuel et te permettra de mieux illustrer la question.

Ton enclos devrait ressembler à celui de la figure 2-8.

Copyright © 2016 Eyrolles.



FIGURE 2-8 Un enclos de 10 blocs de largeur et 10 de longueur.

Enregistrer les coordonnées des angles de ton enclos

Pour pouvoir géorepérer ton enclos à l'aide de ton programme, tu as besoin d'enregistrer les coordonnées de ses angles en toute fiabilité afin de stocker les valeurs dans ton programme Python.

Le plus simple est encore d'exécuter ton programme `ouSuisJe.py` puis de te diriger vers les angles de ton enclos pour récupérer les coordonnées x, y et z de chacun des angles. Prends un papier et dessine un schéma de ton enclos en indiquant ces coordonnées, car tu en auras besoin lorsque tu rédigeras ton programme. Souviens-toi que la coordonnée z augmente à mesure que tu te déplaces vers le sud et diminue lorsque tu te diriges vers le nord, comme tu as pu le voir dans la figure 2-1.

Dans la figure 2-9, tu peux voir les coordonnées de l'enclos que nous avons bâti lorsque nous avons rédigé ce livre. Nous avons indiqué les coordonnées des quatre angles afin de pouvoir isoler les nombres les plus grands et les plus petits des directions x et z. Tu remarqueras que les coordonnées les plus petites se trouvent en haut à gauche.

Ton programme de géorepérage a besoin de quatre nombres. Tout d'abord, tu dois repérer les coordonnées x les plus grandes et les plus petites. Note-les ensuite sur ton schéma. Dans l'exemple de la figure 2-9, la coordonnée x la plus petite est de 10 et la plus grande est de 20. Nous avons choisi de nommer la coordonnée x la plus petite `X1` et la plus grande `X2`.

Copyright © 2016 Eyrolles.

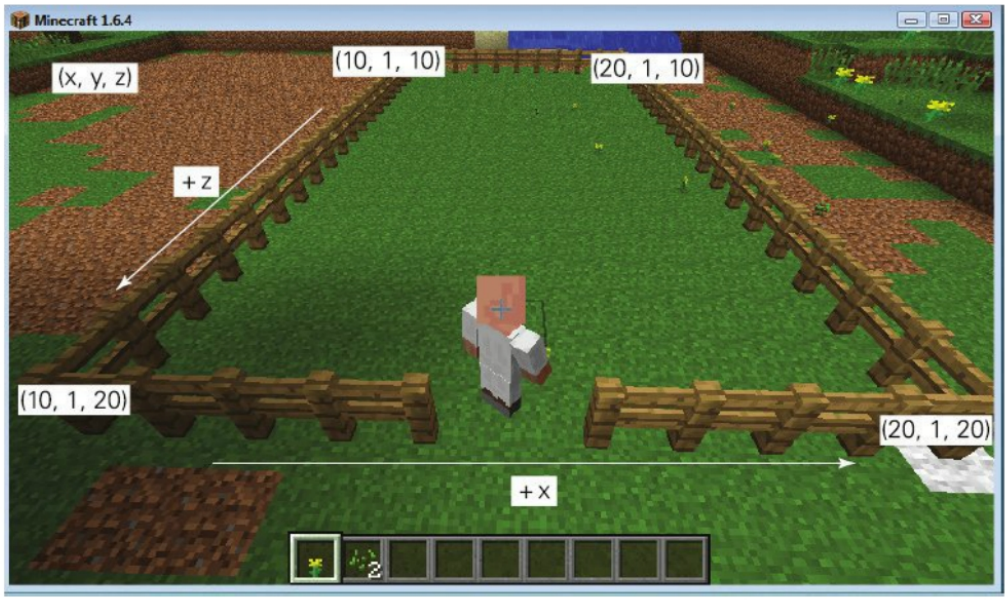


FIGURE 2-9 Maintenant que tu as noté les coordonnées des angles de ton enclos, tu es prêt pour la technique du géorepérage.



Selon la position de ton joueur dans le monde de Minecraft, il se peut que certaines coordonnées soient négatives (par exemple : `x=-5`, `y=0` et `z=-2`), ce qui peut compliquer un peu les calculs. Pour te faciliter la tâche, déplace-toi dans le monde de Minecraft afin d'obtenir des coordonnées positives (par exemple : `x=5`, `y=2`, `z=4`). Si tu disposes d'un Raspberry Pi, les coordonnées de ton joueur s'affichent en haut à gauche de ton écran de jeu. Si tu disposes d'un PC ou d'un Mac, appuie sur la touche `F3` pour les afficher.

Ensuite, tu devras calculer les plus petites et les plus grandes coordonnées `z`. Dans l'exemple de la figure 2-9, la coordonnée `z` la plus petite est de 10 et la plus grande est de 20. J'ai choisi de nommer la coordonnée `z` la plus petite `Z1` et la plus grande `Z2` :

```
X1 = 10
Z1 = 10
X2 = 20
Z2 = 20
```

Tâche d'attribuer le même nombre à tes coordonnées les plus grandes et les plus petites, car tu en auras besoin pour rédiger ton programme par la suite.

Copyright © 2016 Eyrolles.

Écrire le programme de géorepérage

La dernière étape consiste à rédiger le programme qui va mettre la technique du géorepérage en pratique et prélever une somme d'argent dès lors que ton joueur entrera dans l'enclos. Ce programme est structuré de la même manière que le programme `ouSuisJe.py`.

Tu vas également utiliser ce qu'on appelle des **constantes** dans ce nouveau jeu. Tu pourras ainsi déplacer ton enclos plus facilement si besoin et n'auras pas à fouiller dans tes programmes à la recherche des valeurs à modifier pour réaliser cette action.



Comme pour une variable, une **constante** est le nom qu'on donne à un fragment de mémoire dans lequel tu peux stocker des valeurs qui restent identiques à chaque fois que le programme s'exécute.

Contrairement aux autres langages de programmation, Python ne dispose pas vraiment de manière de gérer les constantes. En revanche, pour pallier cela, les programmeurs Python utilisent généralement des conventions de programmation (c'est-à-dire une méthode). Ils écrivent notamment les constantes en lettres capitales pour pouvoir les reconnaître plus facilement et ainsi ne pas en modifier la valeur initiale. Les autres langages de programmation ont leur propre méthode pour reconnaître les constantes et dès lors qu'on essaye d'en changer la valeur, ils affichent un message d'erreur. Le langage Python ne dispose pas de cette particularité.

Passons maintenant à la rédaction du programme.

1. Ouvre tout d'abord une nouvelle fenêtre dans IDLE en sélectionnant `File>New File` dans le menu de l'interpréteur.
2. Choisis `File>Save As` dans le menu et nomme ton fichier `impot.py`.
3. Importe les modules dont tu as besoin pour ce programme :

```
import mcpi.minecraft as minecraft
import time
```

4. Connecte-toi au jeu Minecraft :

```
mc = minecraft.Minecraft.create()
```

5. Tu dois maintenant définir les valeurs des constantes de tes quatre points de géorepérage. Utilise les coordonnées que tu as calculées tout à l'heure. Pour ce qui nous concerne, nous avons repris les coordonnées de notre programme, mais il se peut que les vôtres soient différentes :

```
X1 = 10
Z1 = 10
```

Copyright © 2016 Eyrolles.

```
X2 = 20
Z2 = 20
```

6. Crée ensuite une variable qui conservera un registre des impôts prélevés au joueur. Comme ton joueur n'aura pas encore subi de prélèvement d'impôt au début du jeu, tu dois créer une nouvelle variable que tu nommeras `impot` et que tu mettras à zéro :

```
impot = 0
```

7. Crée la boucle principale de ton jeu avec une instruction `while` pour faire tourner ton programme. Comme il va tourner sans arrêt, utilise l'instruction `sleep` pour le retarder d'une seconde entre chaque tour, cela te donnera en outre une idée du temps que ton joueur passe dans l'enclos. Plus tard, tu ajouteras `1` à la variable `impot` après chaque tour de boucle. Comme tu ralentis la boucle d'une seconde, cela veut dire que l'impôt augmentera de 1 toutes les secondes. Respecte bien l'indentation des lignes à partir de cette étape :

```
while True:
    time.sleep(1)
```

8. Pour savoir si ton joueur se trouve dans le périmètre de géorepérage, tu dois connaître sa position. Comme dans tes précédents programmes, utilise `player.getTilePos()` pour le savoir :

```
    pos = mc.player.getTilePos()
```

9. Tu es maintenant arrivé à l'étape la plus importante du programme. C'est ici que tu vas indiquer à ton programme d'utiliser les quatre coordonnées que tu as calculées plus tôt pour savoir si ton joueur se situe dans le périmètre et, si c'est le cas, pour lui prélever un impôt tout en le lui signalant dans le chat. Pour cela, saisis :

```
    if pos.x>X1 and pos.x<X2 and pos.z>Z1 and pos.z<Z2:
        impot = impot+1
        mc.postToChat("Tu dois la somme de :" + str(impot))
```



À ce stade, tu dois faire très attention à la manière dont tu indentes ton code. Python utilise l'indentation pour reconnaître les blocs d'instructions. Comme tu peux le remarquer, les instructions qui se trouvent en dessous de `while True:` sont indentées d'un cadratin. Cela permet à Python de savoir que ces instructions appartiennent à la boucle `while`. Mais, regarde l'instruction `if` de plus près : les deux lignes qui la suivent du programme sont indentées d'un cadratin supplémentaire. Cela veut dire qu'elles font partie de l'instruction `if` et qu'elles s'exécuteront uniquement si ton joueur se trouve dans l'enclos.

Copyright © 2016 Eyrolles.

Sauvegarde ton programme à nouveau et exécute-le en sélectionnant `Run>Run Module` dans le menu de l'éditeur.

Amuse-toi à entrer et sortir de l'enclos avec ton joueur. Lorsque tu es dans l'enclos, tu dois voir apparaître toutes les secondes dans le chat un message t'indiquant le montant de l'impôt que tu dois payer. Lorsque tu en sors, le calcul de ton impôt s'arrête. Enfin, il ne t'aura certainement pas échappé qu'à chaque fois que tu retournes dans l'enclos, ton programme se souvient du montant que tu dois payer.

DÉFI

Pour que le jeu soit plus amusant, tu dois donner une mission à ton joueur. Crée quelques blocs à l'intérieur de l'enclos, puis fais fonctionner ton programme à nouveau. Le but est de récupérer tous les blocs en payant le moins d'impôts possible. Il te suffit de briser les blocs pour les récupérer. Même si ton programme ne reconnaît pas ces actions, cela rendra le jeu bien plus intéressant. Tu peux encore t'amuser à parsemer toutes sortes d'objets inventés dans ton enclos et inviter tes amis à relever le défi de les glaner en payant le moins d'impôts possible. Alors, à toi de jouer !



Déplacer ton joueur

Il existe un moyen de rendre ton jeu encore plus captivant en faisant notamment appel à une autre fonctionnalité de l'API de Minecraft que tu trouveras bien utile par la suite. Il s'agit de transporter ton joueur à un autre endroit dans Minecraft ! Pour cela, tu devras modifier ton programme `impot.py` et lui ajouter la possibilité d'éjecter ton joueur de l'enclos s'il y reste plus de trois secondes. Tu devras alors revenir dans l'enclos pour récupérer tes blocs.

Tu devras tout d'abord localiser un point en dehors de l'enclos où tu souhaites que ton joueur atterrisse. Le moyen le plus simple d'y parvenir est d'ajouter 2 points aux coordonnées X et Z les plus grandes de ton enclos pour obtenir d'autres coordonnées. Pour rajouter un effet spectaculaire à ton action, choisis une coordonnée Y à une altitude élevée. Le nombre que tu choisiras dépendra de l'altitude à laquelle tu as construit ton enclos, mais si ce dernier se trouve à une hauteur de `y = 0`, tu pourrais choisir par exemple une hauteur de 10. Ton joueur sera alors propulsé dans les airs et la gravité le fera retomber sur terre.

Tu vas donc modifier ton programme `impot.py` pour éjecter ton joueur hors de l'enclos s'il y reste plus de trois secondes, en suivant ces instructions.

1. Définis trois nouvelles constantes au début de ton programme pour indiquer la position d'un abri. Cet abri se trouvera juste en dehors de l'enclos et lorsque ton

Copyright © 2016 Eyrolles.

joueur sera éjecté, il atterrira à cet endroit précis. Les nouvelles lignes à ajouter sont en gras :

```
X1 = 10
Z1 = 10
X2 = 20
Z2 = 20

ABRI_X = X2 + 2
ABRI_Y = 10
ABRI_Z = Z2 + 2
```

2. Tu vas maintenant préciser le temps que ton joueur peut passer dans l'enclos. Pour cela, tu dois créer une autre variable que tu peux appeler `dansEnclos`. Affecte-lui le nombre de secondes que ton joueur peut passer dans l'enclos. Elle va alors l'éjecter lorsqu'il aura passé trop de temps à l'intérieur. Là encore, n'ajoute que les lignes marquées en gras à ton programme :

```
impot = 0
dansEnclos = 0
```

3. Saisis ensuite une ligne de code pour incrémenter la variable `dansEnclos` de 1 lorsque ton joueur se trouve dans l'enclos. Pour cela, ajoute le code en gras :

```
if pos.x>X1 and pos.x<X2 and pos.z>Z1 and pos.z<Z2:
    impot = impot+1
    mc.postToChat("Tu dois la somme de: " + str(impot))
    dansEnclos = dansEnclos+1
```

4. Tu vas maintenant procéder à l'ajout d'une instruction `else` pour que le programme remette le compteur `dansEnclos` à zéro après que ton joueur a été éjecté en dehors l'enclos. Fais bien attention à indenter l'instruction `else` correctement pour que Python interprète bien ton code. Nous t'expliquerons un peu plus tard à quoi servent l'instruction `else` et le symbole `#` que tu vas utiliser. Pour l'instant contente-toi de saisir le code en gras :

```
    mc.postToChat("Tu dois la somme de:" + str(impot))
    dansEnclos = dansEnclos+1
else: # en dehors de l'enclos
    dansEnclos = 0
```

5. Enfin, ajoute quelques lignes de code supplémentaires à la fin de ton programme pour propulser le joueur dans les airs s'il reste dans l'enclos plus de trois secondes. La gravité le fera retomber sur terre et il pourra alors se ruer vers l'enclos à nouveau. Ce code doit se trouver tout à la fin de ton programme. L'instruction `if` doit être indentée d'un cadratin, car elle fait partie de la boucle `while True` et le bloc d'instructions qui suit `if` doit être indenté de deux cadratins, car il fait partie du `if`. Saisis le code suivant :

Copyright © 2016 Eyrolles.

```
if dansEnclos>3:
    mc.postToChat("Tu es trop lent !")
    mc.player.setPos(ABRI_X, ABRI_Y, ABRI_Z)
```

6. Sauvegarde ton programme et exécute-le en cliquant sur `Run>Run Module` dans le menu de l'éditeur.

Tu vas bien t'amuser maintenant, car tu vas devoir redoubler d'ingéniosité pour entrer et sortir de l'enclos sans te faire éjecter ! Comme tout à l'heure, sélectionne une série d'objets à parsemer dans ton enclos et lance un défi à tes amis : qui de vous tous remportera cette folle course contre la montre en glanant le plus de blocs possible sans y laisser trop de plumes (voir la figure 2-10) ?

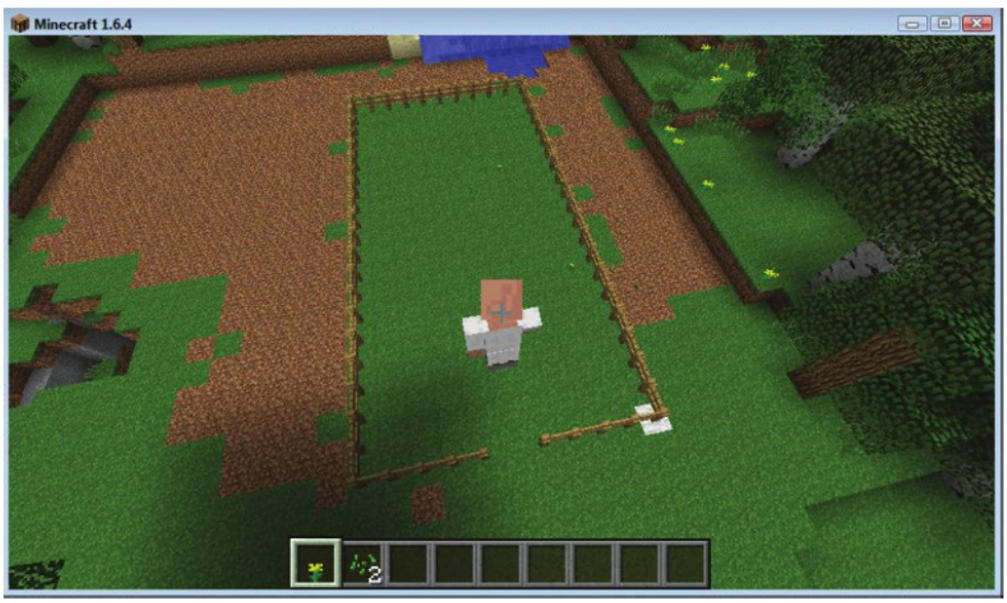


FIGURE 2-10 Ton joueur est propulsé dans les airs lorsqu'il reste trop longtemps dans l'enclos.

PERCER LES SECRETS DU CODE

Tu viens d'utiliser deux nouveaux éléments du langage Python. Il est temps de les approfondir un peu.

Tout d'abord, tu as utilisé l'instruction `else`, un mot-clé qui vient compléter l'instruction `if`. Il doit toujours se trouver au même niveau d'indentation que le `if` auquel il est associé et les instructions que tu souhaites rédiger sous `else` doivent aussi être indentées. En voici un exemple :

```
if a>3:
    print("grand")
```

Copyright © 2016 Eyrolles.

```
else:
    print("petit")
```

Si la variable `a` renferme une valeur supérieure à 3, Python affichera « grand » dans l'interpréteur de commandes, sinon (`else`), c'est le terme « petit » qui s'affichera. Autrement dit, si `a>3` est vrai, alors le mot « grand » s'affichera, mais si `a>3` est faux, alors c'est « petit » qui s'affichera.

Une instruction `if` ne s'accompagne pas forcément d'une instruction `else` ; celle-ci est optionnelle. Mais tu auras parfois besoin de traiter la condition vraie différemment de la condition fausse.

Le deuxième élément que tu as utilisé est un commentaire. Tu verras qu'il est parfois bien utile de laisser un commentaire en marge de ton code pour te souvenir de certaines choses ou pour donner des indications à un autre lecteur.

Les commentaires commencent toujours par le symbole `#` :

```
# ceci est un commentaire
```

Tu peux laisser un commentaire n'importe où sur tes lignes de code. Dès lors que tu utiliseras le symbole dièse, tout ce qui suivra jusqu'à la fin de la ligne sera ignoré par Python.

Explorer d'autres pistes de localisation

Dans ce chapitre, tu as appris à mettre en pratique les fondements d'une boucle de jeu et à utiliser les techniques de localisation et de géorepérage pour créer un jeu amusant. Lorsque tu crées des jeux, nous te recommandons vivement d'inviter tes amis et ta famille à y jouer et de leur demander leur avis. C'est le meilleur moyen d'obtenir des conseils qui t'aideront à améliorer tes jeux.

- Le jeu « Bienvenue à la maison » est amusant, mais il a un problème : si ton joueur franchit le pas de la porte, il ne cesse d'afficher « Bienvenue à la maison » et cela t'empêche de bien voir. Peux-tu trouver un moyen de modifier le programme pour qu'il ne te salue qu'une seule fois lorsque tu entres chez toi ?
- Une bonne technique pour trouver de nouvelles idées est de découvrir d'autres jeux et d'observer comment ils tiennent les joueurs en haleine. Sois curieux, joue à d'autres jeux et fais une liste de toutes les caractéristiques que pourrait contenir un bon jeu, notamment en ce qui concerne le déplacement du joueur.
- Effectue des recherches sur Internet au sujet de la technique du géorepérage et vois comment l'industrie en fait usage. Si certaines idées t'intéressent, lance-toi dans l'écriture d'un programme Minecraft qui en reprend les principaux aspects, en utilisant les connaissances que tu as acquises dans cette aventure.

Copyright © 2016 Eyrolles.

Tableau de références	
Importer l'API de Minecraft	**Se connecter à Minecraft**
`import mcpi.minecraft as minecraft`	`mc = minecraft.Minecraft.create()`
Obtenir la position du joueur	**Afficher un message dans le chat de Minecraft**

`pos = mc.player.getTilePos()`
`x = pos.x`
`y = pos.y`
`z = pos.z`
	`mc.postToChat("Bonjour Minecraft")`

Définir la position du joueur	

`x = 5`
`y = 3`
`z = 7`
`mc.player.setTilePos(x, y, z)`	



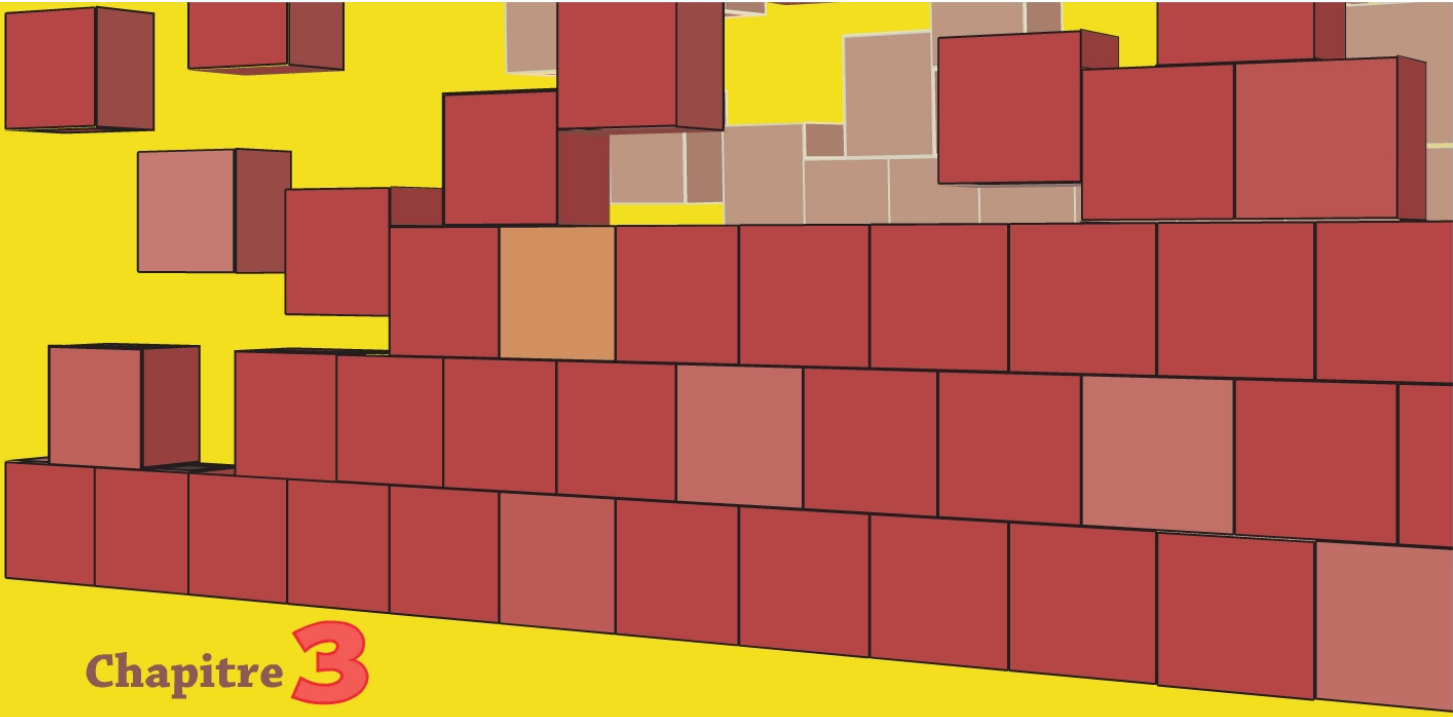
Niveau terminé : Tu as créé un super jeu qui réagit de plusieurs manières en fonction des déplacements de ton joueur dans Minecraft.

Dans le prochain chapitre...

Dans le chapitre 3, tu vas apprendre à construire de grandes structures, telles que des maisons, de façon automatique en utilisant des blocs et des boucles Python. Avec un seul programme Python, tu as la possibilité de bâtir d'immenses structures bien plus vite que si tu les construisais à la main. Tu pourras ainsi concevoir une ville entière en un rien de temps.

Copyright © 2016 Eyrolles.

Copyright © 2016 Eyrolles.



Chapitre 3

Automatiser tes constructions

IL N'Y A RIEN de plus amusant que de construire des structures dans Minecraft. Tu peux construire presque tout ce dont tu rêves. Tes seules limites sont celles du jeu Minecraft lui-même et de ta propre imagination. Tu peux ajouter des maisons, des châteaux, des rivières souterraines, des hôtels à plusieurs étages avec piscines et même des villes entières ! Mais construire des objets complexes avec plusieurs types de blocs peut prendre un temps fou et te demander beaucoup de travail, notamment si cela implique des répétitions. Alors, pourquoi ne pas essayer d'automatiser certaines tâches ? Rien de tel pour épater une fois de plus tes amis.

Le langage Python est idéal pour automatiser des tâches complexes. Dans ce chapitre, tu apprendras quelques tours de passe-passe qui te permettront de construire automatiquement un grand nombre de blocs dans Minecraft. Puis tu répéteras l'opération en introduisant des boucles dans ton code, afin de bâtir d'immenses structures comme des rues entières bordées de maisons (voir la figure 3-1). Tes amis seront d'autant plus impressionnés de parcourir tes allées de maisons qu'ils voudront à tout prix connaître ton secret ! Tu apprendras également à exécuter des commandes directement depuis ton clavier, ce qui te permettra de réutiliser des programmes autant de fois que tu le voudras sans toucher au code !

Copyright © 2016 Eyrolles.



FIGURE 3-1 Une maison construite à partir de blocs Minecraft

Créer des blocs

Dans Minecraft, chaque type de bloc dispose de sa propre valeur de données. Si tu as déjà joué à Minecraft, tu dois probablement en connaître quelques-uns. Mais l'interface de programmation de Minecraft te permet de désigner les blocs par leur nom plutôt que par leur valeur de données, ce qui te facilite grandement la tâche. En fait, ces noms sont des constantes – notion que nous avons abordée dans le chapitre 2 – auxquelles on a attribué les valeurs de données des types de blocs. Pour connaître la liste des types de blocs les plus courants, leurs noms et leurs valeurs de données, consulte l'annexe B dans la partie réservée aux références.

Le monde de Minecraft est fait de cubes invisibles qui sont tous dotés de coordonnées uniques. Chacun de ces cubes contient un type de bloc précis dont ton ordinateur connaît l'identité, même s'il s'agit d'un bloc de type `AIR`. En tant que programmeur Minecraft, tu peux demander à l'interface de programmation du jeu de t'indiquer le type de bloc correspondant à une coordonnée précise. Tu peux aussi modifier n'importe quel type de bloc, quelles que soient ses coordonnées. Par exemple, tu peux construire des ponts en remplaçant des blocs de type `AIR` par des blocs de type `ROCHE`, ce que tu apprendras à faire dans le chapitre 4.

Tu vas maintenant rédiger un simple programme pour illustrer tout cela. Ce programme consistera à placer automatiquement un simple bloc en face de ton joueur. Une fois cette opération effectuée, tu seras capable de bâtir tout ce que tu voudras automatiquement !

1. Lance Minecraft, IDLE ainsi que le serveur si tu es sur un PC ou un Mac. Tu t'es déjà certainement un peu familiarisé avec le processus, mais n'hésite pas à relire le chapitre 1 si tu as besoin de te rafraîchir la mémoire.

Copyright © 2016 Eyrolles.

2. Ouvre IDLE, l'environnement de développement intégré (EDI) du langage Python. Tu vas utiliser cette fenêtre pour rédiger et exécuter tous tes programmes en Python.

3. Clique sur `File>New File` dans le menu de l'interpréteur. Va dans `File>Save As` et nomme ton fichier `bloc.py`. N'oublie pas d'enregistrer tes programmes dans le dossier `MyAdventures`, sinon ils ne fonctionneront pas.

4. Maintenant, importe les modules dont tu as besoin pour ce chapitre. Tu utiliseras un module complémentaire nommé `block`, qui contient toutes les valeurs de données des types de blocs que renferme Minecraft. Pour cela, saisis :

```
import mcpi.minecraft as minecraft
import mcpi.block as block
```

5. Connecte-toi au jeu Minecraft avec l'instruction suivante :

```
mc = minecraft.Minecraft.create()
```

6. Ensuite, obtiens la position de ton joueur en utilisant la variable `pos`. Tu utiliseras ces coordonnées pour calculer l'espace disponible en face de ton joueur et y créer ton nouveau bloc :

```
pos = mc.player.getTilePos()
```

7. Crée maintenant un bloc en face de ton joueur, sur la base des coordonnées de ta position. Tu en sauras plus sur les **coordonnées relatives** dans les paragraphes suivants. En utilisant `pos+3` dans ton programme, tu t'assures de ne pas écraser ton joueur avec le bloc que tu vas créer ! Pour cela, saisis :

```
mc.setBlock(pos.x+3, pos.y, pos.z, block.STONE.id)
```

8. Clique sur `File>Save` pour sauvegarder ton programme, puis exécute-le en cliquant sur `Run>Run Module` dans le menu de l'éditeur.

Tu devrais maintenant voir apparaître un bloc de pierre tout près de toi ! Tu viens d'indiquer au monde de Minecraft de créer un bloc en face de ton joueur. À partir de ces simples instructions, tu peux désormais te lancer dans la création automatique de structures très intéressantes en les programmant directement dans Minecraft.

DÉFI

La pierre (`STONE`) est une matière un peu ennuyeuse. Jette un œil à l'annexe B (à télécharger) et amuse-toi à remplacer l'élément `STONE` de ton programme par d'autres types de blocs. Certains sont disponibles sur des systèmes d'exploitation, mais pas sur d'autres. Nous les avons catalogués pour toi à l'annexe B.



Par exemple, le type de bloc obsidienne brillante (`GLOWING_OBSIDIAN`) existe sur Raspberry Pi, mais pas sur PC ni Mac. Certains blocs sont soumis à la force de la gravité. Utilise les blocs eau (`WATER`) et sable (`SAND`) à la place de la roche et regarde ce qui se passe !

Copyright © 2016 Eyrolles.



Les **coordonnées relatives** sont une série de coordonnées définies par rapport à un autre point, par exemple par rapport à la position de ton joueur dans Minecraft. Les coordonnées `pos.x`, `pos.y+10`, `pos.z+3` indiquent un point dans Minecraft situé à 10 blocs au-dessus et 3 blocs au sud de ton joueur. En d'autres termes, ces coordonnées sont relatives à la position de ton joueur. Dès lors que ton joueur se déplacera dans le monde de Minecraft, ces coordonnées relatives changeront également.

Les **coordonnées absolues** sont une série de coordonnées qui utilisent des valeurs fixes pour définir la position d'un point donné, comme celle d'un bloc Minecraft par exemple. Les coordonnées `x=10`, `y=10`, `z=15` sont des coordonnées absolues. Chaque fois que tu les utilises dans Minecraft, tu fais référence à un bloc qui est situé aux points 10, 10 et 15 et dont la position ne change jamais.

DÉFI



Si tu as déjà fait tous les exercices du chapitre 2, ouvre et exécute ton programme `ouSuisJe.py` puis déplace-toi dans le monde de Minecraft pour te diriger vers un espace dégagé. Note les coordonnées x, y et z de ta position. Modifie ton programme `bloc.py` et remplace les coordonnées `setBlock()` par les **coordonnées absolues** de ta position. Exécute le programme et vois ce qu'il se passe. D'après toi, à quoi peuvent servir les coordonnées relatives et absolues dans tes programmes ?

Construire plus d'un bloc à la fois

Maintenant que tu connais les bases de la construction de blocs, tu peux ajouter tout ce que tu veux dans ton monde Minecraft ! Tu peux notamment t'amuser à compléter ton programme pour construire plus d'un bloc à la fois. Lorsque tu construiras des bâtiments, tu devras faire appel à des notions de mathématiques très simples pour calculer les coordonnées de chacun des blocs de tes bâtiments.

C'est ce que tu vas faire dès maintenant en complétant ton programme `construire.py` de manière à créer cinq blocs supplémentaires en face de ton joueur, en attribuant à chacun des coordonnées différentes à l'aide de `setBlock()`. Ce programme va te permettre de bâtir une structure qui ressemble à un dé de jeu. Pour cela, suis ces quelques étapes :

1. Pour commencer, vas dans `File>Save As` dans le menu de l'éditeur et enregistre ton programme en le nommant `de.py`.

Copyright © 2016 Eyrolles.

2. Ajoute maintenant ces lignes à la fin du programme :

```
mc.setBlock(pos.x+3, pos.y+2, pos.z, block.STONE.id)
mc.setBlock(pos.x+3, pos.y+4, pos.z, block.STONE.id)
mc.setBlock(pos.x+3, pos.y,   pos.z+4, block.STONE.id)
mc.setBlock(pos.x+3, pos.y+2, pos.z+4, block.STONE.id)
mc.setBlock(pos.x+3, pos.y+4, pos.z+4, block.STONE.id)
```

3. Une fois ton programme enregistré, exécute-le. Tu devrais voir apparaître une structure représentant les six points d'un dé, juste en face de toi (voir la figure 3-2) !



FIGURE 3-2 La structure Minecraft qui se trouve en face du joueur représente les six points d'un dé.

DÉFI

Modifie le programme `de.py` pour remplir les espaces autour des points en roche avec des blocs de couleur blanche, afin que ta structure ressemble davantage à un dé de jeu. Joue ensuite avec les blocs `ROCHE` et `AIR` pour changer les numéros de ton dé. Souviens-toi qu'il existe une liste détaillée de blocs à l'annexe B.



Utiliser des boucles for

Tu peux construire presque tout ce que tu veux en ajoutant des blocs un par un, mais c'est un peu comme si tu dessinais une image complexe pixel après pixel sur ton écran d'ordinateur. Tu dois trouver une manière de répéter tes tâches de construction pour bâtir des structures plus grandes sans pour autant allonger ton programme.

Copyright © 2016 Eyrolles.

Créer plusieurs blocs à l'aide d'une boucle for

Heureusement, comme dans tout langage informatique, Python possède ce qu'on appelle une boucle. Tu as déjà rencontré cette notion dans le chapitre 2 avec la boucle de jeu `while True:`. Une boucle te permet tout simplement de répéter certaines opérations plusieurs fois. Celle que tu utiliseras ici s'appelle `for`. On l'appelle parfois une **variable de comptage**, car elle s'exécute un nombre défini de fois.

Comme tu l'as fait avec la boucle `while` dans le chapitre 2, tu dois indenter les instructions de programmation associées à la boucle `for` pour que Python comprenne que seules ces instructions doivent se répéter lorsque tu exécutes la boucle.

Pour comprendre comment cette boucle fonctionne, utilise l'interpréteur de commandes Python et utilise-la comme ceci.



La boucle `for` est appelée **variable de comptage** ou variable de comptage contrôlé, car elle s'exécute un nombre défini de fois. Dans l'instruction `range()`, tu indiques le nombre de fois que tu souhaites que la boucle s'exécute. La boucle `for` dans Python est assez spéciale, car elle peut compter d'autres éléments que des nombres, comme tu pourras le voir dans le chapitre 6.

1. Dans la fenêtre de l'interpréteur de commandes Python, clique à droite de la dernière invite de commandes `>>>`.
2. Saisis le code suivant et appuie sur la touche `Entrée` :

```
for a in range(10):
```

   Python n'interprétera rien, car il attend de recevoir les instructions de la boucle, autrement dit ce que tu saisiras dans le « corps » de la boucle.

3. L'interpréteur de commandes Python va automatiquement indenter la ligne suivante d'un cadratin. Ainsi, il saura que l'instruction `print()` appartient bien à la boucle `for`. À présent, saisis-la :

```
print(a)
```

4. Appuie sur la touche `Entrée` deux fois : la première fois pour indiquer la fin de l'instruction `print()`, la seconde pour dire à Python que la boucle est terminée.

Tu devrais maintenant voir s'afficher la suite de chiffres 0 à 9 dans l'interpréteur de commandes Python.

Copyright © 2016 Eyrolles.

PERCER LES SECRETS DU CODE

En l'analysant de plus près, la boucle `for` que tu viens d'utiliser se compose de plusieurs paramètres intéressants :

```
for a in range(10):
    print(a)
```

Ici, la lettre `a` renvoie à ce qu'on appelle une « variable de contrôle de boucle ». Cette variable contient la valeur de la boucle à chaque tour. Au premier tour de boucle, la valeur équivaut à 0, au deuxième tour, elle équivaut à 1, et ainsi de suite.

Le paramètre `range(10)` indique à Python que tu souhaites produire une séquence de chiffres allant de 0 à 9, que ta boucle `for` passera en revue.

Le deux-points (`:`) à la fin remplissent la même fonction que ceux que tu as utilisés dans les chapitres précédents avec la boucle `while` et les instructions `if`. Ils permettent d'indiquer le début de ton bloc d'instructions. Dans ce cas précis, le bloc d'instructions ou « corps » de ton code contient les informations que tu vas répéter 10 fois.

Toutes les instructions se trouvant à l'intérieur de la boucle (c'est-à-dire toutes les instructions indentées d'un cadratin à partir de l'instruction `for`), pourront accéder à la variable `a` dont la valeur changera à chaque tour de boucle.

Tu peux en outre utiliser la variable de contrôle de boucle `a` à tout moment dans le bloc d'instructions de ta boucle sans pour autant faire appel à `a` chaque fois que tu en as besoin. Enfin, tu peux lui donner n'importe quel nom ; il est d'ailleurs recommandé de nommer tes variables de façon plus descriptive, car les personnes qui liront ton programme seront ainsi plus à même de le comprendre. En l'occurrence, il serait préférable de remplacer `a` par `decompte` ou `nombre_de_fois`.

Construire une immense tour à l'aide d'une boucle for

Comment ferais-tu pour construire une immense tour de blocs dans Minecraft ? Maintenant que tu en sais un peu plus sur les boucles `for`, tu es tout à fait capable de réaliser cet exploit. Suis simplement ces indications.

1. Commence par créer un nouveau programme via `File>New File` dans le menu. Choisis `File>Save As` dans le menu et nomme-le `tour.py`.
2. Comme d'habitude, importe les modules dont tu as besoin :

```
import mcpi.minecraft as minecraft
import mcpi.block as block
```

Copyright © 2016 Eyrolles.

3. Puis connecte-toi au jeu Minecraft :

```
mc = minecraft.Minecraft.create()
```

4. Si tu bâtis ta tour à l'endroit où ton joueur se trouve, il sera plus facile de la retrouver par la suite. La première étape consistera alors à identifier la position de ton joueur en saisissant le code suivant :

```
pos = mc.player.getTilePos()
```

5. Tu construiras une tour d'une hauteur de 50 blocs. Pour ce faire, tu dois la bâtir à l'aide d'une boucle `for` qui s'exécutera 50 fois. N'oublie pas de saisir le deux-points (`:`) à la fin de la ligne :

```
for a in range(50):
```

6. La ligne suivante est indentée, car elle appartient au corps de la boucle `for`. La coordonnée y représente la hauteur dans le monde de Minecraft. Affecte la position de ton joueur à la variable de contrôle de boucle `a`. Cela te permettra d'indiquer au programme de construire une structure en hauteur dans Minecraft à chaque tour de boucle.

```
mc.setBlock(pos.x+3, pos.y+a, pos.z, block.STONE.id)
```

7. Enregistre ton programme et exécute-le. A-t-il fonctionné ? Ta boucle `for` devrait avoir créé une tour gigantesque en face de ton joueur, comme celle que tu peux voir à la figure 3-3. Fais le calcul : la tour mesure-t-elle bien 50 blocs de hauteur ?

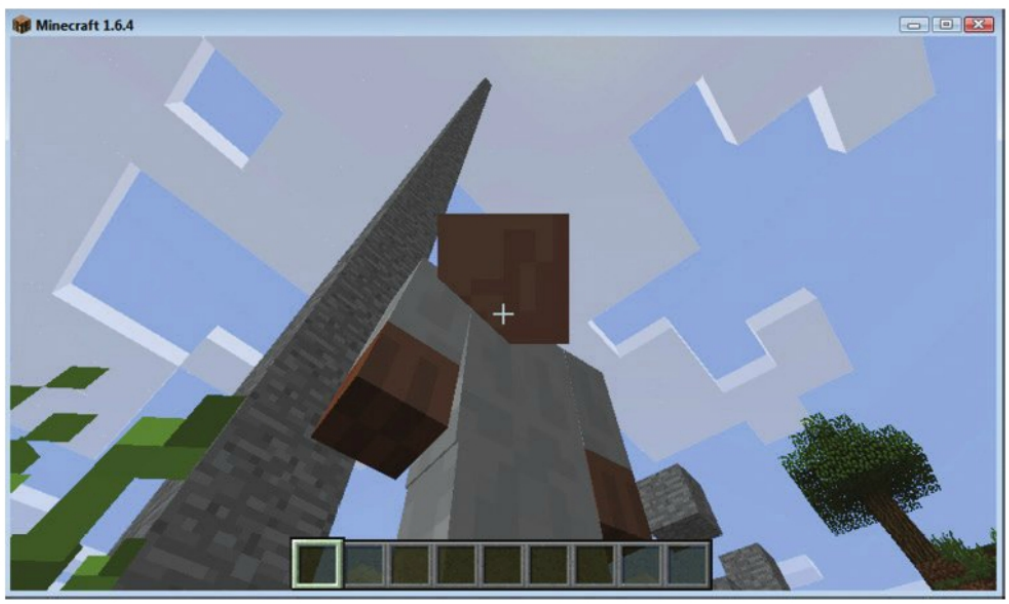


FIGURE 3-3 Cette immense tour a été bâtie dans Minecraft à l'aide d'une boucle for.

Copyright © 2016 Eyrolles.

Dans le code précédent, la boucle `for` tourne 50 fois et, à chaque tour, elle exécute les lignes indentées en dessous, de sorte que la valeur de la variable `a` augmente d'un cube à chaque fois. Elle additionne la valeur de `a` (la variable de contrôle de boucle) à celle de `pos.y`, qui correspond à la hauteur à laquelle se situe ton joueur sur le sol. C'est ainsi que ta tour de 50 blocs est créée !



DÉFI

Modifie la valeur de `range()` dans la boucle `for` pour créer une tour encore plus grande. Jusqu'où pourras-tu aller sans dépasser le cadre de ton écran ?



Faire de la place

Trouver l'espace nécessaire pour construire une structure dans Minecraft peut parfois s'avérer frustrant. Bien souvent, ton joueur se trouve dans des espaces montagneux ou au beau milieu d'une forêt qui l'empêchent de bâtir de grandes structures. Grâce à Python, tu pourras rapidement remédier à ce problème en rédigeant un script qui te permettra de faire de la place autour de toi.

Utiliser setBlocks pour construire encore plus vite

Tous les blocs dans Minecraft disposent d'un numéro d'identité (`ID`), y compris les espaces inoccupés ou remplis d'air, qui s'appellent `block.AIR.id`. Si tu identifies plusieurs blocs avec `block.AIR.id` sur une grande surface, cela te permettra de faire de la place pour construire des objets. Une liste des numéros d'identité des blocs, à savoir des numéros de types de blocs, t'est fournie à l'annexe B, pour les plus communs d'entre eux. Par exemple, le numéro de `block.AIR.id` est 0 et celui de `block.STONE.id` est 1. Souviens-toi que tous les espaces vides que tu vois dans Minecraft sont en réalité des blocs d'un mètre cube remplis d'air, donc dont l'identité est `block.AIR.id` !

Copyright © 2016 Eyrolles.



Les ordinateurs sont certes très rapides, mais plus tu insères des instructions dans une boucle `for`, plus le programme mettra de temps à s'exécuter. Minecraft met du temps à interpréter ta demande de définir un bloc dans le jeu, d'en modifier le type et de rafraîchir la vue. Tu pourrais toujours rédiger une grande boucle définissant des centaines de blocs avec `block.AIR.id` pour faire de la place devant toi. Cela fonctionnerait, mais prendrait beaucoup de temps. Heureusement, il existe un moyen de définir plusieurs blocs à la fois : `setBlocks()`. Note le `s` à la fin du terme. `setBlock()` permet de définir un bloc tandis que `setBlocks()` définit tout un volume de blocs en une seule et même opération !

L'interface de programmation Minecraft possède une instruction `setBlocks()` que tu peux utiliser pour attribuer la même identité à plusieurs blocs au sein d'un espace rectangulaire tridimensionnel (3D) donné. Comme il s'agit d'une seule requête, Minecraft peut l'exécuter plus efficacement, donc plus rapidement que si tu utilisais une instruction `setBlock()` au sein d'une boucle `for` comme tu l'as fait plus tôt pour la tour.

L'instruction `setBlocks()` s'exécutant sur un espace en 3D, elle a besoin de deux séries de coordonnées : celles de l'un des angles du parallélépipède et celles de son opposé. Chaque coordonnée 3D contient trois nombres. Tu dois donc indiquer six nombres pour définir un espace 3D dans Minecraft.

Ces notions étant maintenant claires pour toi, tu vas rédiger un petit programme bien pratique que tu pourras utiliser quand tu le voudras, afin de libérer de l'espace devant toi avant de passer à la construction d'un bâtiment :

1. Commence par créer un nouveau programme en choisissant `File>New File` dans le menu.
2. Enregistre ton programme en cliquant sur `File>Save As` et nomme-le `deblayage.py`.
3. Importe ensuite les modules dont tu as besoin :

```
import mcpi.minecraft as minecraft
import mcpi.block as block
```

4. Puis connecte-toi au jeu :

```
mc = minecraft.Minecraft.create()
```

5. Tu souhaites maintenant faire de la place en face de ton joueur. Pour cela, tu utiliseras des coordonnées relatives. Tu dois donc commencer par récupérer les données de position de ton joueur :

```
pos = mc.player.getTilePos()
```

Copyright © 2016 Eyrolles.

6. Puis fais de la place en libérant 50 mètres cubes d'espace. Tu te serviras de la position de ton joueur pour définir l'angle inférieur gauche de l'espace que tu souhaites nettoyer. L'angle supérieur droit de l'espace se trouvera 50 blocs plus loin, ce que tu indiqueras dans les trois dimensions x, y et z.

```
mc.setBlocks(pos.x, pos.y, pos.z, pos.x+50, pos.y+50,
             pos.z+50, block.AIR.id)
```

7. À présent, sauvegarde ton programme.

Ce qui est génial, dans le programme que tu viens de créer, c'est qu'il te permet de libérer 50 mètres cubes d'espace partout où tu vas. Il te suffit de l'exécuter à tout moment en cliquant sur `Run>Run Module` dans le menu de Python et tout ce qui se trouve devant toi disparaîtra comme par magie, montagnes et arbres compris !

Fais quelques pas et exécute le programme à nouveau pour libérer encore plus d'espace.

Saisir des données depuis le clavier

Pour rendre ton programme `deblayage.py` encore plus pratique, tu peux simplifier la modification des données spatiales à déblayer. Ainsi, pour construire une petite structure, tu pourras déblayer un petit terrain, et pour construire des superstructures, tu pourras libérer plusieurs kilomètres d'espace.

Tu utiliseras pour cela une instruction Python qui s'appelle `raw_input()` et qui te permettra de saisir les données directement sur ton clavier. Ainsi, tu pourras te rendre dans n'importe quel endroit de Minecraft, exécuter ton script, saisir un nombre correspondant aux dimensions de l'espace à libérer, et le programme réalisera ton souhait. Tu n'auras donc plus à modifier ton programme à chaque fois que tu souhaiteras déblayer des espaces de tailles différentes dans Minecraft.



Il existe deux versions de Python : Python 2 et Python 3. L'interface de programmation de Minecraft est développée pour fonctionner avec Python 2, et c'est Python 2 que tu utilises dans tous les chapitres de ce livre. Il est possible de rédiger des programmes Python pour Minecraft avec la version 3, mais tu dois utiliser pour cela un module mcpi mis à jour. Si tu consultes des sites Internet utilisant la version 3, les programmes se serviront de l'instruction `input()` au lieu de `raw_input()` pour pouvoir saisir des données depuis le clavier. Lorsque nous avons écrit ce livre, les toutes dernières versions dont nous disposions étaient la 2.7.6 et la 3.3.3. Il existe beaucoup de différences entre Python 2 et Python 3. C'est pourquoi il est important de se limiter à la version 2 pour les exercices de ce livre.

Maintenant que tu sais comment `raw_input()` fonctionne, il est temps de l'ajouter à ton programme. Pour cela, il te suffit d'effectuer deux petites modifications dans ton fichier `deblayage.py`, comme suit.

Copyright © 2016 Eyrolles.

1. Tout d'abord, utilise `raw_input()` pour demander à l'utilisateur de saisir un nombre correspondant à l'espace à défricher et affecte ce nombre à une variable que tu appelleras `dimensions`. Comme dans le chapitre 2, où tu as utilisé `str()` pour faire une conversion en chaîne, tu vas utiliser ici `int()` pour convertir la chaîne de caractères de `raw_input()` en un nombre avec lequel tu pourras ensuite effectuer des calculs. Essaie de saisir ce code sans `int()` et observe ce qui se passe ! Ne saisis que les lignes nouvelles indiquées en gras :

```
pos = mc.player.getTilePos()
dimensions = int(raw_input("dimensions de l'espace
                           à déblayer ?"))
```

2. Modifie ensuite la ligne `setBlocks()` pour remplacer le nombre 50 par ta nouvelle variable `dimensions` :

```
mc.setBlocks(pos.x, pos.y, pos.z, pos.x+dimensions,
             pos.y+dimensions, pos.z+dimensions, block.AIR.id)
```

3. Sauvegarde ton programme et exécute-le avec `Run>Run Module`.

Maintenant, déplace-toi dans Minecraft et trouve un endroit rempli d'arbres et de montagnes. Saisis un nombre dans la fenêtre de l'interpréteur de commandes Python lorsque celle-ci s'affiche – choisis par exemple le nombre 100 – et appuie sur la touche `Entrée` de ton clavier. Tous les arbres et toutes les montagnes de tes alentours devraient disparaître comme par magie pour laisser place, comme dans la figure 3-4, à un espace gigantesque où tu pourras construire des structures !

Mets ce programme de côté pour l'instant. Il te sera bien utile plus tard, lorsque tu auras besoin de faire de l'espace dans Minecraft.



FIGURE 3-4 Exécute ton programme deblayage.py et note comme les arbres ont été sciés à la base !

Copyright © 2016 Eyrolles.

DÉFI



Le programme `deblayage.py` permet de dégager de l'espace en face de ton joueur, qui se trouve à l'angle inférieur gauche du volume déblayé. Cela dit, il se peut que tu doives tourner ton joueur pour voir l'espace déblayé, car tout dépend de la direction dans laquelle il était tourné. En fonction de tes objectifs, il s'avérerait peut-être plus pratique pour toi de libérer de l'espace tout autour de ton joueur. Le cas échéant, tu peux modifier les coordonnées relatives de l'instruction `setBlocks()` pour modifier les coordonnées des angles de ton parallélépipède, de façon à ce que ton joueur se trouve au centre. Tu auras peut-être également besoin d'indiquer à `deblayage.py` de « creuser » dans le sol en diminuant la valeur de la coordonnée y, pour poser les fondations de ton futur bâtiment, mettre du gazon ou placer d'autres blocs sur le sol. Essaie de le faire pour savoir si tu trouveras la solution. Tu verras que ton programme sera d'autant plus utile ainsi !

Construire une maison

Lorsque tu joues à Minecraft en mode survie, l'une des premières choses que tu dois faire est de te confectionner un abri pour te protéger contre le danger qui te guette pendant la nuit. Et si tu te construisais directement une maison en appuyant sur un seul bouton ? Heureusement, en programmant dans Minecraft, tu peux le faire facilement et tu verras que réaliser des tâches très complexes ne tient qu'à un bouton !

Dans le programme `deblayage.py`, tu as appris à attribuer la même valeur de données à plusieurs blocs en même temps, dans une seule et même instruction. Maintenant, tu vas apprendre à utiliser les mêmes techniques pour construire une maison en un clin d'œil.

Tu pourrais très bien construire ta maison en utilisant une instruction `setBlocks()` pour chaque mur. Pour cela, tu devrais utiliser quatre instructions `setBlocks()` séparément et faire fonctionner tes méninges pour calculer toutes les coordonnées des angles de tes murs. Heureusement, il existe un moyen plus simple de construire des structures rectangulaires creuses. Tout ce que tu as à faire, c'est de construire un grand cube et d'utiliser `setBlocks()` en modifiant légèrement ses coordonnées pour remplir l'intérieur avec des blocs de type `AIR`.

Mais avant d'aller plus loin, tu dois prendre le temps d'analyser les calculs mathématiques que tu vas effectuer pour concevoir ta maison, car tu auras besoin de les saisir correctement dans ton programme. La figure 3-5 représente une esquisse de la maison, où sont indiquées toutes les coordonnées importantes que tu devras prendre en compte lors de la construction. Lorsque tu construis une structure complexe, il est primordial de la concevoir d'abord sur le papier et de calculer toutes les coordonnées importantes. Cette maison est très spéciale, car c'est le programme qui calcule la taille

Copyright © 2016 Eyrolles.

et l'emplacement de la porte et des fenêtres. Tu comprendras plus tard pourquoi cela est si important.

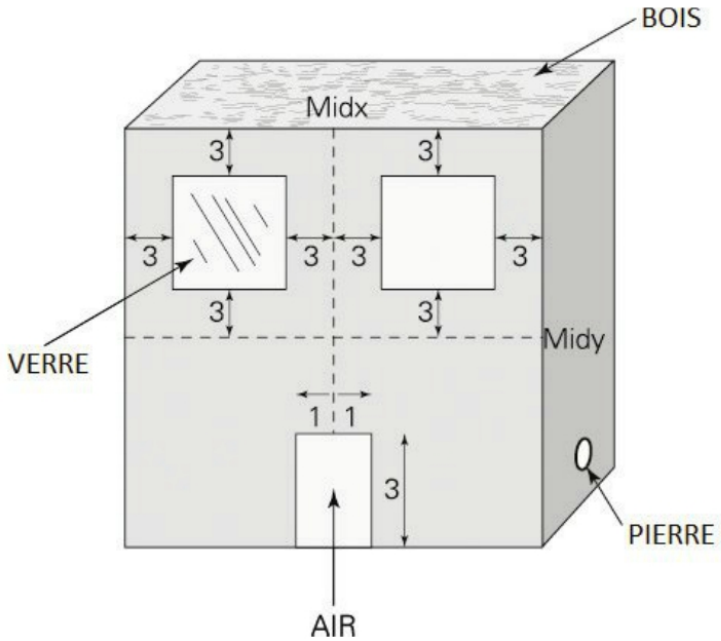


FIGURE 3-5 L'esquisse de ta maison, avec le calcul de toutes les coordonnées importantes



Tu peux regarder une vidéo didactique sur la manière de créer ta maison en te rendant sur le site Internet du livre `www.editions-eyrolles.com/dl/14292`. Choisis la vidéo « Adventure 3 » (en anglais).

Maintenant que tu as une idée claire de la façon dont tu souhaites construire ta maison, lance-toi dans les travaux en suivant ces quelques étapes.

1. Commence par créer un nouveau programme en choisissant `File>New File` dans le menu.
2. Enregistre ton programme en cliquant sur `File>Save As` dans le menu et nomme-le `constructionMaison.py`.
3. 3. Importe ensuite les modules nécessaires :

```
import mcpi.minecraft as minecraft
import mcpi.block as block
```

4. Puis connecte-toi au jeu :

```
mc = minecraft.Minecraft.create()
```

Copyright © 2016 Eyrolles.

5. Utilise maintenant une constante pour définir les dimensions de ta maison. Tu utiliseras cette constante `DIMENSIONS` assez souvent dans ton script. En utilisant une constante au lieu du nombre `20`, il te sera plus facile de modifier la taille de ta maison par la suite :

```
DIMENSIONS = 20
```

Rappelle-toi la convention de programmation et saisis bien les constantes en majuscules. Même si ce n'est pas obligatoire en langage Python, cela te permet de savoir qu'il s'agit d'une variable et que tu souhaites la conserver telle quelle lorsque ton programme s'exécute.



6. Vérifie la position de ton joueur pour construire ta maison juste à côté :

```
pos = mc.player.getTilePos()
```

7. Affecte les coordonnées x, y et z du joueur aux nouvelles variables. Cela te permettra de saisir les prochaines instructions plus facilement et t'aidera lorsque tu seras amené à réaliser d'autres travaux sur ta maison. Tu veilleras à augmenter la variable x de ta position de deux blocs, pour ne pas construire la maison sur ton joueur :

```
x = pos.x+2
y = pos.y
z = pos.z
```

8. Calcule les deux variables `midx` et `midy`, qui sont les points du milieu de la façade de ta maison dans les directions x et y. Cela te permettra plus tard de calculer facilement les coordonnées des fenêtres et de la porte. Ainsi, si tu modifies les dimensions de ta maison, les positions des fenêtres et de la porte s'ajusteront automatiquement.

```
midx = x + DIMENSIONS/2
midy = y + DIMENSIONS/2
```

9. Construis les murs extérieurs de ta maison sous forme de grands parallélépipèdes. Tu peux choisir n'importe quel type de bloc ici, mais les pierres sont idéales pour commencer, car elles te permettront de créer une maison de style ancien.

```
mc.setBlocks(x, y, z, x+DIMENSIONS, y+DIMENSIONS,
             z+DIMENSIONS, block.COBBLESTONE.id)
```

10. À présent, creuse l'intérieur de la maison en la remplissant de blocs d'air. Tu remarqueras que tu utilises de simples notions de mathématiques pour calculer

Copyright © 2016 Eyrolles.

les coordonnées des blocs d'air situés de l'intérieur, avec des coordonnées relatives aux angles extérieurs des façades en pierres :

```
mc.setBlocks(x+1, y, z+1, x+DIMENSIONS-2, y+DIMENSIONS-1,
             z+DIMENSIONS-2, block.AIR.id)
```

11. Creuse une enclave pour la porte, en utilisant une fois de plus des blocs de type `AIR`. Tu n'utiliseras pas de porte Minecraft dans ce cas-là, car ta maison est trop grande pour cela. Tu vas en revanche créer une grande embrasure de trois blocs de haut et deux de large. Ta porte doit se situer au milieu de la façade avant de ta maison. Et grâce à `midx`, tu connais l'emplacement de la coordonnée x de son milieu :

```
mc.setBlocks(midx-1, y, z, midx+1, y+3, z, block.AIR.id)
```

12. Creuse deux fenêtres en utilisant des blocs de type `GLASS`. Comme il s'agit d'une grande maison, tu placeras tes fenêtres trois blocs en dessous du toit et trois blocs à partir du centre de la façade de la maison. Si tu modifies ta constante `DIMENSIONS`, exécute le script à nouveau et toutes tes données s'ajusteront automatiquement : l'embrasure de la porte et les fenêtres se trouveront à la bonne place et ta structure ressemblera à une maison normale.

```
mc.setBlocks(x+3, y+DIMENSIONS-3, z, midx-3, midy+3, z,
             block.GLASS.id)
mc.setBlocks(midx+3, y+DIMENSIONS-3, z, x+DIMENSIONS-3,
             midy+3, z, block.GLASS.id)
mc.setBlocks(x, y+DIMENSIONS-1, z, x+DIMENSIONS,
             y+DIMENSIONS-1, z+DIMENSIONS, block.WOOD.id)
mc.setBlocks(x+1, y-1, z+1, x+DIMENSIONS-2, y-1,
             z+DIMENSIONS-2, block.WOOL.id, 14)
```

Tu remarqueras un numéro supplémentaire à la fin de l'instruction `setBlocks()`. Ce numéro sert à définir la couleur de la moquette. Ici, le numéro 14 désigne la couleur rouge. Cette question est abordée un peu plus loin dans l'encadré « Percer les secrets du code ».

Sauvegarde ton programme, rends-toi dans un endroit où tu trouveras assez d'espace et exécute le script avec `Run>Run Module`. Tu devrais voir ta maison apparaître comme par magie sous tes yeux ! Entres-y et fais une petite visite. Lève les yeux vers le toit et regarde par les fenêtres. Tu peux prendre le temps d'apprécier ton merveilleux œuvre, que tu as construit en si peu de temps (voir la figure 3-6) !

N'oublie pas que ton script `construireMaison.py` s'exécute toujours là où ton joueur se situe. Fais un petit tour dans le monde de Minecraft et exécute ton programme à nouveau pour construire une autre maison. Tu as maintenant la possibilité de faire pousser les maisons comme des champignons, n'est-ce pas formidable ?

Copyright © 2016 Eyrolles.



FIGURE 3-6 Cette maison est le résultat d'un programme Python.

PERCER LES SECRETS DU CODE

Lorsque tu as posé ta moquette, tu as ajouté un numéro à l'instruction `setBlocks()` :

```
mc.setBlocks(x1, y1, z1, x2, y2, z2, block.WOOL.id, 14)
```

Voyons d'un peu plus près ce que cela signifie.

La laine (`WOOL`) est un type de bloc très intéressant, car il dispose de ce qu'on appelle une « donnée supplémentaire ». Cela veut dire que tu peux non seulement transformer le bloc en laine, mais aussi que tu peux utiliser cette donnée supplémentaire pour en changer l'apparence. D'autres types de blocs, dans le monde de Minecraft, disposent de données supplémentaires qu'ils utilisent de plusieurs manières. Pour la laine, le numéro de la « donnée supplémentaire » te permet de choisir la couleur du bloc. Cette fonctionnalité pratique fait du type `WOOL` un objet modulable, car tu peux choisir parmi toute une palette de couleurs, ce qui te permet de personnaliser tes conceptions et de les rendre plus réalistes.

À la fin de ce chapitre, tu trouveras un tableau regroupant toutes les couleurs du bloc `WOOL` et leurs numéros correspondants. L'annexe B fournit également une liste des autres types de blocs qui disposent de données supplémentaires modifiables.

Copyright © 2016 Eyrolles.



En t'aventurant sur Internet, tu es probablement déjà tombé sur d'autres programmes Minecraft. Si c'est le cas, tu dois te demander pourquoi, dans ce livre, nous signalons les blocs de cette manière : `block.WOOL.id`. Pourquoi placer `.id` à la fin du bloc ? Et pourquoi les autres scripts sur Internet ne le mentionnent-ils pas ? C'est parce que `setBlock()` et `setBlocks()` ne fonctionnent pas toujours lorsque tu mentionnes une donnée supplémentaire, sauf si tu utilises `.id` à la fin du bloc. Donc, pour faire simple et faciliter ta lecture, nous avons choisi de toujours mentionner le suffixe `.id` des blocs. Même si tu n'es pas toujours obligé de le conserver, il est préférable de le mentionner par défaut. De cette manière, tu ne te préoccuperas plus de savoir si tu dois le mettre ou non.



Essaie de remplacer la constante `DIMENSIONS` de ton script par un nombre plus élevé, comme 50, et exécute ton programme à nouveau. Observe ce qui se passe. Maintenant, abaisse sa valeur à 10. Que se passe-t-il ? D'après toi, pourquoi ce phénomène se produit-il lorsque tu affectes un nombre très petit à cette constante ?

Construire plusieurs maisons

Formidable ! Tu viens de construire ta propre maison. Mais il serait bien dommage de s'arrêter en si bon chemin. Souviens-toi du script `tour.py` que tu as créé un peu plus tôt dans ce chapitre. Il est très facile de concevoir une boucle `for` qui répète les instructions de ton programme plusieurs fois de manière définie. Alors, ne penses-tu pas qu'il serait tout aussi facile de construire une rue bordée de maisons, voire une ville entière, en utilisant une boucle dans ton programme `construireMaison.py` ?

Avant de te mettre au travail, tu vas devoir effectuer de petites modifications sur ton programme de construction de maison pour l'alléger un peu.

Utiliser les fonctions Python

Tu auras peut-être envie, plus tard, de construire une ville entière avec des maisons différentes. Une telle tâche pourrait vite s'avérer fastidieuse et complexe. Heureusement, le langage de programmation Python te facilite la vie, car il renferme une fonctionnalité qui te permet d'aborder la tâche par petits blocs de code réutilisables que l'on appelle des **fonctions**.

Copyright © 2016 Eyrolles.

Au sein d'une **fonction** Python, tu peux rassembler plusieurs instructions et leur donner un nom spécifique. Lorsque tu auras besoin d'exécuter ces instructions, tu n'auras plus qu'à écrire le nom de la fonction, suivi de parenthèses.



Tu ne t'en es peut-être pas encore aperçu, mais en suivant les conseils de ce livre, tu as déjà utilisé des fonctions, par exemple lorsque tu as posté un message sur le chat de Minecraft à l'aide de `mc.postToChat("bonjour")`, dans le tout premier chapitre. En fait, `postToChat()` est une fonction, c'est-à-dire un simple bloc d'instructions qui s'exécutent dès que tu utilises le terme `postToChat()` dans tes lignes de code.



Avant de modifier ton programme `construireMaison.py` et d'utiliser une fonction pour la maison, entraîne-toi à utiliser quelques fonctions dans ta console Python pour bien comprendre de quoi il retourne.

1. Clique sur la fenêtre de l'interpréteur de commandes Python pour l'afficher au premier plan.
2. Définis une nouvelle fonction que tu appelleras `monnom` en saisissant le code suivant dans ton interpréteur de commandes :

```
def monnom():
```

   Le mot-clé `def` veut simplement dire « définir une nouvelle fonction, dont voici le nom ». Comme pour les mots-clés `while`, `if` et `for`, tu dois ajouter un deux-points `:` à la fin de la ligne pour indiquer que tu souhaites écrire d'autres instructions dans le corps de cette fonction.

3. L'interpréteur de commandes Python indentera automatiquement la ligne suivante pour toi et l'interprétera comme une instruction appartenant à la fonction. Enfin, saisis quelques lignes avec la fonction `print` pour afficher ton nom et tes informations. Tu constateras alors que rien ne s'affiche lorsque tu saisis ces lignes. Ne sois pas surpris, c'est normal. Patience, tout s'éclaircira dans quelques instants !

```
print("Je m'appelle David")
print("Je suis programmeur")
print("J'adore programmer dans Minecraft")
```

4. Enfin, appuie sur `Entrée` deux fois pour que Python reconnaisse que tu as fini de saisir les lignes de code dans les instructions indentées.

Copyright © 2016 Eyrolles.

5. Demande ensuite à ton interpréteur d'exécuter la fonction en saisissant son nom suivi de parenthèses, comme ceci :

```
monnom()
```

6. Essaie de saisir `monnom()` plusieurs fois et observe ce qui se passe.

À première vue, il t'a peut-être semblé étrange de ne rien voir apparaître après avoir saisi des instructions dans l'interpréteur. Normalement, lorsqu'on écrit dans l'interpréteur Python, il réagit dès qu'on appuie sur `Entrée`, alors pourquoi cela n'a pas fonctionné cette fois-ci ? En réalité, c'est parce que tu viens de faire quelque chose de différent : tu as « défini » une nouvelle fonction que tu as appelée `monnom` et tu as demandé à Python de se souvenir des trois appels à `print` à l'intérieur de la fonction. Python a enregistré ces appels dans la mémoire de l'ordinateur, au lieu de les exécuter sur-le-champ. Maintenant, dès que tu écriras `monnom()`, l'interpréteur exécutera les instructions enregistrées et tu verras apparaître les trois lignes de texte à l'écran.

Les fonctions sont très puissantes, elles te permettent d'associer plusieurs instructions de programmation Python à un seul et même nom, comme s'il s'agissait d'un mini-programme en somme. Dès que tu souhaiteras exécuter ces instructions, il te suffira d'écrire le nom de la fonction.

Mettons ces quelques connaissances en pratique et définissons une fonction pour dessiner une maison. Ainsi, lorsque tu voudras créer une maison en roche, tu n'auras plus qu'à taper `maison()` et elle se construira toute seule !

1. Afin de ne pas écraser ton programme `construireMaison.py`, va dans `File>Save As` dans le menu `Édition` et nomme ce nouveau fichier `construireMaison2.py`.

2. En haut du programme, après l'instruction `import`, définis une nouvelle fonction que tu nommeras maison :

```
def maison():
```

3. Déplace les lignes `midx`, `midy` et toutes les instructions `setBlocks()` pour les indenter en dessous de la ligne `def maison():`. Voilà ce à quoi devrait désormais ressembler ton programme. Surtout, veille bien à indenter tes lignes de code correctement :

```
import mcpi.minecraft as minecraft
import mcpi.block as block

mc = minecraft.Minecraft.create()

DIMENSIONS = 20

def maison():
    midx = x + DIMENSIONS/2
```

Copyright © 2016 Eyrolles.

```
    midy = y + DIMENSIONS/2
    mc.setBlocks(x, y, z, x+DIMENSIONS, y+DIMENSIONS,
                 z+DIMENSIONS, block.COBBLESTONE.id)
    mc.setBlocks(x+1, y, z+1, x+DIMENSIONS-2,
                 y+DIMENSIONS-1, z+DIMENSIONS-2, block.AIR.id)
    mc.setBlocks(midx-1, y, z, midx+1, y+3, z, block.AIR.id)
    mc.setBlocks(x+3, y+DIMENSIONS-3, z, midx-3, midy+3, z,
                 block.GLASS.id)
    mc.setBlocks(midx+3, y+DIMENSIONS-3, z, x+DIMENSIONS-3,
                 midy+3, z, block.GLASS.id)
    mc.setBlocks(x, y+DIMENSIONS-1, z, x+DIMENSIONS,
                 y+DIMENSIONS-1, z+DIMENSIONS, block.WOOD.id)
    mc.setBlocks(x+1, y-1, z+1, x+DIMENSIONS-2, y-1,
                 z+DIMENSIONS-2, block.WOOL.id, 14)

pos = mc.player.getTilePos()
x = pos.x
y = pos.y
z = pos.z

maison()
```

4. Comme tu peux le voir, dans la dernière instruction de ton programme, tu as simplement ajouté le nom `maison()`. Cette ligne exécutera le code stocké dans la mémoire de l'ordinateur que tu avais préalablement défini grâce à l'instruction `def maison():`.

5. Enregistre ton programme, déplace-toi dans Minecraft et exécute-le. Tu devrais voir une maison apparaître en face de toi.



À présent, tu dois certainement penser : « C'est bien tout ça, mais après ? ». Tu as déplacé quelques instructions dans le programme et cela donne exactement le même résultat que tout à l'heure ! En programmation, il est parfois nécessaire d'améliorer la structure de ton code avant de le développer. C'est exactement ce que tu viens de faire ici. Tu as légèrement restructuré ton code, ce qui te permettra par la suite de construire des maisons plus facilement.

DÉFI



La fonction `maison()` construira toujours la maison en utilisant les valeurs que tu as définies dans `x`, `y` et `z`. Amuse-toi à ajouter des instructions à la fin de ton programme pour changer les variables `x`, `y` et `z` puis insère une autre `maison()` et constate le résultat. Combien de maisons penses-tu pouvoir construire de cette manière ?

Copyright © 2016 Eyrolles.

PERCER LES SECRETS DU CODE

Maintenant que tu as placé toutes tes instructions `setBlocks()` dans la fonction `maison()`, le programme te réserve encore bien des surprises.

Les variables `x`, `y`, `z` ainsi que la constante `DIMENSIONS` appartiennent à la catégorie des **variables globales**. On les appelle ainsi, car on a défini leur valeur dans le corps principal du programme, c'est-à-dire la partie non indentée du code (hors de la fonction `maison`). Le terme « global » signifie qu'on peut les utiliser partout dans le programme, y compris à l'intérieur de la fonction `maison()`. Ainsi, si tu as relevé le défi de dessiner plusieurs maisons en modifiant les valeurs des variables `x`, `y` et `z` dans le corps principal du programme, tu auras remarqué que cela fonctionne parfaitement. En effet, ces variables étant globales, elles peuvent être utilisées partout dans le programme.

Dans le chapitre 6, tu apprendras que les variables globales peuvent devenir un vrai casse-tête lorsqu'elles sont utilisées dans un programme plus important qui contient des défauts. Tu verras alors qu'il existe un meilleur moyen de partager les informations à l'aide des fonctions. Mais pour l'instant, l'utilisation des variables globales est bien pratique et ton programme est trop petit pour que cela devienne problématique.



Une **variable globale** est une variable qu'on peut utiliser partout dans le programme. Toute variable (ou constante) n'étant pas indentée à gauche est une variable globale à laquelle on peut accéder n'importe où dans le programme. Ainsi, si tu définis la variable `a = 1` sans l'indenter à gauche, tu pourras l'utiliser partout dans ton code.

De même, toute variable étant indentée à gauche n'est pas une variable globale. On l'appellera alors plus souvent « variable locale ». Si tu places l'instruction `a = 1` dans une instruction `def`, donc à l'intérieur d'une fonction, tu ne pourras accéder à cette variable qu'à l'intérieur de cette même fonction. Il existe bien d'autres questions à creuser concernant les variables globales, mais à ce stade, tu n'as pas besoin d'en savoir davantage.

Construire une rue bordée de maisons à l'aide d'une boucle for

Tu es désormais capable d'utiliser toutes les connaissances que tu as acquises dans ce chapitre pour construire une rue entièrement bordée de maisons. Si tu devais construire toutes ces maisons manuellement en sélectionnant les éléments dans l'inventaire, tu y passerais des heures et obtiendrait un résultat probablement approximatif, contenant des erreurs ou des maisons à dimensions inégales. En automatisant la construction

Copyright © 2016 Eyrolles.

de maison ou d'autres structures à l'aide de la programmation, tu peux aller plus vite tout en étant beaucoup plus précis et ainsi construire des bâtiments parfaitement proportionnés !

Pour construire plusieurs maisons à la fois, tu dois ajouter une boucle `for` à ton script, comme ceci.

1. Pour ne pas abîmer ton joli programme `construireMaison2.py`, va dans `File>Save As` dans le menu et enregistre-le dans un nouveau fichier que tu appelleras `construireRue.py`.
2. À la fin du script, ajoute une boucle `for` au-dessus de la dernière instruction `maison()` ainsi qu'une ligne de code pour modifier la coordonnée `x` utilisée pour indiquer l'emplacement de la maison. Toutes les maisons sont construites à plusieurs blocs d'intervalle, qui correspondent à la valeur affectée à ta variable `DIMENSIONS`. Les lignes nouvelles sont indiquées en gras :

```
for h in range(5):
    maison()
    x = x + DIMENSIONS
```

Sauvegarde ton programme, déplace-toi dans Minecraft pour te rendre dans un endroit avec suffisamment d'espace et exécute le script. Comme sur la figure 3-7, tu devrais obtenir une grande rue bordée de cinq maisons, qui t'obstruerait presque entièrement la vue ! Fais un tour dans chacune des maisons et vérifie qu'elles ne contiennent aucun défaut.



FIGURE 3-7 Une rue de cinq maisons identiques construites automatiquement à l'aide d'un programme Python

Copyright © 2016 Eyrolles.

DÉFI



Réfléchis à la manière dont tu pourrais modifier ta boucle pour qu'elle empile tes maisons de façon à créer une immense tour au lieu de les construire les unes à côté des autres. Tente le coup : transforme ton script `tour.py` en fonction, que tu nommeras `tour()`, puis utilise une boucle pour construire plusieurs tours dans le monde de Minecraft.



Selon l'endroit où tu te trouves dans Minecraft au moment où tu construis tes maisons, et selon la nature du terrain qui t'entoure – que tu sois par exemple en forêt ou dans la montagne –, tu obtiendras des résultats intéressants et parfois même surprenants. Il se peut, en effet, que l'une de tes maisons se soit nichée au creux d'un arbre ou d'une montagne, ou encore qu'elle s'affaisse légèrement si par mégarde tu l'as construite sur l'eau ! Pour éviter cet inconvénient, tu peux toujours modifier ta fonction `maison()` et y introduire des fondations en `bedrock`, afin qu'elle résiste aux variations de terrain.

Poser des moquettes de couleurs différentes de manière aléatoire

Tu vois désormais fleurir une multitude de maisons dans ton monde. C'est maintenant que les choses commencent à devenir intéressantes. Grâce à quelques lignes de code, tu es parvenu à construire des ouvrages déjà très complexes !

Cependant, tu auras peut-être envie d'égayer toutes ces allées de maisons identiques en les personnalisant un peu. Alors, comment faire pour ajouter une touche personnelle à tout cela ?

L'un des moyens d'y parvenir est de rédiger plusieurs fonctions différentes de `maison()`, telles que `maisonnette()`, `maisonMitoyenne()` et même `pavillon()`, puis de modifier ton script pour placer ces fonctions à plusieurs endroits différents afin de diversifier ton paysage urbain.

Une autre manière de rendre tes structures plus intéressantes consisterait à en modifier certains éléments, tels que la moquette, de sorte qu'ils changent à chaque fois que tu exécutes le programme. Cela deviendrait d'autant plus intéressant pour toi en tant que programmeur, car tu te réserverais quelques surprises à découvrir au fil de tes visites.

Copyright © 2016 Eyrolles.

Produire des nombres aléatoires

Les ordinateurs sont des machines très précises. Au quotidien, nous leur faisons tous confiance et nous nous attendons à ce qu'ils exécutent les programmes de manière prévisible et avec précision. Lorsque nous versons 10 euros sur notre compte en banque, nous nous attendons en général à ce que l'ordinateur qui effectue l'opération bancaire crédite cette somme exacte, pas plus, pas moins, et ce à chaque fois que nous faisons appel à ce service électronique. La notion d'aléa peut donc sembler un peu étrange dans ce contexte.

Pourtant, s'il est un domaine où cette notion s'avère essentielle, c'est bien dans la conception de jeux vidéo. Si les jeux répétaient les mêmes actions à chaque fois, ils deviendraient bien trop prévisibles et ennuyeux. Presque tous les jeux auxquels tu joues contiennent une part d'imprévu qui pimente l'action et permet de te tenir en haleine.

Heureusement pour toi, le langage Python dispose d'un module intégré qui produit des **nombres aléatoires** automatiquement. Tu n'auras donc pas à rédiger de code pour cela, mais devras simplement utiliser ce générateur de nombres intégré.

Pour savoir à quoi il ressemble, utilise-le dans ton interpréteur de commandes Python, comme ceci.

1. Clique sur la fenêtre de l'interpréteur de commandes Python pour l'afficher au premier plan.
2. Importe le module aléatoire (*random*, en anglais) pour pouvoir utiliser la fonction aléatoire intégrée, en saisissant :

```
import random
```

3. Demande au programme d'afficher un nombre au hasard entre 1 et 100 :

```
print(random.randint(1,100))
```

   Tu devrais voir apparaître un nombre entre 1 et 100 à l'écran. Saisis l'instruction `print` à nouveau. Quel nombre obtiens-tu cette fois-ci ?

4. Maintenant, utilise une boucle `for` pour afficher plusieurs nombres aléatoires. Pour cela, assure-toi de bien indenter la deuxième ligne afin que Python comprenne que l'instruction `print` fait partie de la boucle `for` :

```
for n in range(50):
    print(random.randint(1,100))
```

Les deux nombres entre parenthèses dans la fonction `randint()` indiquent à cette dernière les limites de la séquence de nombres qu'elle doit produire, avec 1 comme minimum et 100 comme maximum.

Copyright © 2016 Eyrolles.



Un **nombre aléatoire** provient généralement d'une séquence de nombres aléatoires, à savoir d'une liste de nombres n'étant fondée sur aucun modèle prédéfini ni sur aucune séquence répétitive.

Les ordinateurs sont des machines très précises et, bien souvent, ils ne génèrent pas de nombres stricto sensu aléatoires. En revanche, ils génèrent ce qu'on appelle des nombres pseudo-aléatoires. Ces nombres semblent provenir d'une séquence aléatoire, mais ils répondent en fait à un modèle. Pour en savoir plus sur les nombres aléatoires, rends-toi sur la page `https://fr.wikipedia.org/wiki/Générateur_de_nombres_aléatoires`. Et pour te renseigner sur les vrais nombres aléatoires, visite le site `www.random.org` (en anglais).

Poser les moquettes

Plus tôt dans ce chapitre, tu as utilisé un nombre supplémentaire dans la fonction `setBlocks()` pour colorer ta moquette en rouge, grâce à la donnée supplémentaire que contient le type de bloc `WOOL`. Les données supplémentaires du type `WOOL` sont comprises entre 0 et 15. Autrement dit, tu peux choisir parmi 16 couleurs existantes. Suis ces quelques étapes pour introduire une fonction `random` dans ton programme de construction, de manière à ce qu'il pose une moquette de couleur différente dans chaque maison :

1. Enregistre ton script `construireRue.py` en cliquant sur `File>Save As` dans le menu et renomme-le `construireRue2.py`.
2. Ajoute cette instruction d'importation au début de ton script pour importer le générateur de nombres aléatoires :

```
import random
```

3. Modifie ta fonction `maison()` pour qu'elle produise une couleur de moquette aléatoire à chaque utilisation. Comme le type de bloc `WOOL` ne comprend que 16 couleurs, la séquence de nombres aléatoires que tu dois indiquer est comprise entre 0 et 15. Affecte le nombre aléatoire à une variable que tu nommeras `c` pour limiter la longueur de ta ligne de code sur la moquette et ainsi en faciliter la lecture. Assure-toi également de bien indenter les lignes en question, car elles font toutes deux partie de la fonction `maison()` :

```
    c = random.randint(0, 15)
    mc.setBlocks(x+1, y-1, z+1, x+DIMENSIONS-2, y-1,
                 z+DIMENSIONS-2, block.WOOL.id, c)
```

4. Sauvegarde ton programme.

Tu dois maintenant trouver un endroit bien dégagé dans Minecraft où tu pourras construire encore plus de maisons ! Trouve donc ton lieu de prédilection et exécute ton nouveau programme. Il devrait produire la même rue bordée de maisons, à ceci près

Copyright © 2016 Eyrolles.

qu'elles disposent toutes d'une moquette de couleur différente, choisie de manière aléatoire. Regarde la figure 3-8 pour voir à quoi ressemblent ces nouvelles maisons. Si tu veux t'assurer que les moquettes sont bien toutes différentes, tu devras te rendre dans chacune de tes maisons avec ton joueur pour en avoir le cœur net.



FIGURE 3-8 Toutes les maisons sont tapissées d'une moquette de couleur différente.

DÉFI

Définis une fonction `maison2()` pour concevoir une maison différente. Introduis une instruction aléatoire dans la principale boucle `for` de ton script afin de créer des maisons à l'aide d'un nombre aléatoire. Toutes tes maisons seront ainsi bâties de manière aléatoire. Développe ton programme pour créer trois types de maisons. Plus tu construiras de maisons différentes et plus ta rue sera intéressante. Tu peux même t'amuser à utiliser des coordonnées négatives, comme `y=-5`, pour ajouter un sous-sol ou une piscine intérieure !



Copyright © 2016 Eyrolles.



Les grandes structures que tu peux créer dans Minecraft n'ont pour limites que celles de ton imagination. Bien sûr, ce monde est lui-même techniquement limité par un plafond que tes plus hautes structures ne pourront pas percer. De plus, certains bâtiments dans la vie réelle ont des structures complexes, parfois inclinées ou incurvées. En témoigne la forme du plus haut gratte-ciel de l'Union européenne, The Shard, que tu pourras voir en te rendant sur `http://the-shard.com`. Tu pourrais tout à fait reproduire ce formidable ouvrage d'art dans Minecraft, mais sa complexité architecturale te donnerait bien du fil à retordre.

Ne perds pas patience pour autant. Contente-toi pour l'heure de construire des structures simples et rectangulaires et, dans le chapitre 7, Martin t'apprendra à utiliser des fonctions spéciales qui te permettront de concevoir des bâtiments inclinés et incurvés. Une fois que tu auras assimilé ces connaissances, tu pourras créer absolument tout ce que tu voudras !

Les seules limites qui s'imposeront à toi lors de tes travaux de construction seront la mémoire de ton ordinateur et la ligne de l'horizon dans le jeu. En effet, plus une structure est grande, plus elle sollicite de mémoire informatique. Les dimensions du monde de Minecraft sont, à ce titre, idéales, car la mémoire qu'il occupe est avantageuse. De même, ton joueur ne peut voir que jusqu'à une certaine distance, notamment sur Raspberry Pi, où l'horizon est particulièrement restreint.

Tableau de références

Importer et utiliser les constantes de nom de bloc	**Définir/modifier l'identité d'un bloc à un endroit donné**
`import mcpi.block as block` `b = block.DIRT.id`	`mc.setBlock(5, 3, 2, block.DIRT.id)`
Définir/modifier l'identité de plusieurs blocs à la fois	**Types de blocs pratiques pour bâtir une maison**

`mc.setBlocks(0,0,0,5,5,5,block.DIRT.id)`	`block.AIR.id`
	`block.STONE.id`
	`block.COBBLESTONE.id`
	`block.GLASS.id`
	`block.WOOD.id`
	`block.WOOL.id`
	`block.SLATE.id`

Toutes les nouvelles instructions Python et Minecraft que tu as appris à utiliser dans ce chapitre sont répertoriées dans l'annexe B de ce livre.

Copyright © 2016 Eyrolles.

Tu trouveras une liste complète des numéros d'identité des différents blocs de Minecraft sur la page wiki du jeu : `http://minecraft-fr.gamepedia.com/Blocs`. Quant aux valeurs supplémentaires de Minecraft, elles sont référencées ici : `http://minecraft-fr.gamepedia.com/Valeurs`.

Comme tu as pu le voir au moment de poser tes moquettes aléatoires dans tes maisons, la `laine` est un type de bloc très pratique pour la construction. Ci-après, tu trouveras une liste des différentes couleurs que tu peux attribuer au type de bloc `block.WOOL.id`. Rouvre ton programme `construireRue2.py` pour voir comment inclure ces valeurs supplémentaires dans l'interface de programmation de Minecraft.

0 blanc	1 orange	2 magenta	3 bleu ciel
4 jaune	5 vert clair	6 rose	7 gris
8 gris clair	9 cyan	10 violet	11 bleu
12 marron	13 vert	14 rouge	15 noir

Explorer d'autres pistes de construction

Dans ce chapitre, tu as appris à construire des blocs un par un et des ensembles de blocs dans Minecraft à l'aide d'une simple ligne de code. Cela t'a permis de réaliser des structures impressionnantes. Tu sais également qu'avec les fonctions tu peux alléger tes scripts et les découper en plusieurs unités logiques. Et tu sais que les boucles `for` te permettent de répéter des instructions à l'infini. Grâce à cela, tu disposes des bagages nécessaires pour bâtir absolument tout ce que tu imagines !

- En faisant appel aux techniques acquises dans ce chapitre, crée un dé en utilisant des fonctions pour concevoir chacune des faces. Rédige une boucle qui s'arrêtera de tourner au hasard, en tombant à chaque fois sur une face différente. Utilise `random.randint()` pour que ton dé s'arrête sur un chiffre au hasard et amuse-toi, avec tes amis, à deviner le chiffre sur lequel ton dé Minecraft s'arrêtera.
- Fais une liste des éléments que tu pourrais définir au hasard dans ta rue. Tu peux même t'inspirer de ta propre rue et observer ce qui différentie chaque maison. Crée plus de fonctions, une pour chaque maison, et essaie de concevoir des rues plus complexes avec des maisons de styles différents.
- Réunis tes amis et rassemblez tous vos scripts de construction de maisons en un seul et même grand programme, que vous utiliserez pour construire une ville constituée de maisons aux styles divers et variés. Vous pourrez alors prendre le temps de partir à la découverte de votre magnifique ouvrage et de vous émerveiller à la vue de toute cette richesse architecturale !

Copyright © 2016 Eyrolles.



Niveau terminé : tu as conçu de superbes bâtiments et construit de magnifiques structures dans Minecraft !

Dans le prochain chapitre...

Dans le chapitre 4, tu vas apprendre à détecter les blocs et à interagir avec eux. Tu vas notamment apprendre à détecter les types de blocs sur lesquels tu marches et ce que tu dois faire lorsque ton joueur les brise. Tu utiliseras toutes ces nouvelles connaissances pour élaborer un jeu captivant de chasse au trésor !

Copyright © 2016 Eyrolles.



Chapitre 4

Interagir avec les blocs

TU PEUX RENDRE le jeu Minecraft bien plus intéressant en le faisant réagir à l'environnement dans lequel ton joueur se déplace. En effet, au fil de tes quêtes, les choix auxquels tu es confronté dépendent des actions que tu as entreprises auparavant, ce qui modifie le cours du jeu à chaque fois que tu t'y connectes. L'API de Minecraft te permet d'interagir avec les blocs en identifiant le type de ceux sur lesquels tu marches ou que tu frappes avec ton épée.

Dans ce chapitre, tu vas créer un pont magique pour apprendre les fondements de l'interaction avec les blocs. Ce pont a quelque chose de spécial, car lorsque tu te mets à marcher sur l'eau ou dans le vide, il apparaît comme par magie sous tes pieds pour assurer ta sécurité. Bientôt, tu verras l'horizon se remplir de ponts ! Une deuxième version de ton constructeur de ponts contiendra une liste Python permettant de mémoriser les endroits où tu en auras construit pour les faire disparaître sous tes yeux dès lors que tes pieds toucheront à nouveau la terre ferme.

Enfin, tu vas apprendre à détecter lorsqu'un bloc est frappé. Ensuite, tu vas concevoir un fantastique jeu de chasse au trésor en utilisant le pont magique pour dégoter des trésors disséminés un peu partout dans le ciel et les récupérer. Tu compléteras alors ton monde Minecraft en y ajoutant des signaux lumineux ainsi qu'un système de points.

Copyright © 2016 Eyrolles.

Dans ce chapitre, tu vas te mettre dans la peau d'un véritable ingénieur informatique et concevoir un long programme fonction après fonction. Une fois tes fonctions prêtes, tu les regrouperas toutes au sein d'un seul programme. Alors, attache ta ceinture, ouvre grand les yeux et prépare-toi à décoller pour vivre une aventure extraordinaire en plein ciel !

Identifier les blocs sur lesquels tu te déplaces

Tu as appris, dans le chapitre 2, qu'il est possible de localiser ton joueur et d'obtenir ses coordonnées grâce à l'instruction `getTilePos()`. Ces coordonnées sont les valeurs x, y et z de l'emplacement de ton joueur au sein du monde de Minecraft à un moment donné. Tu as récupéré ces coordonnées pour savoir si Steve se trouvait sur un paillasson magique ou, en faisant appel à la technique du géorepérage, s'il se trouvait à l'intérieur d'un champ.

Mais à moins que tes programmes ne tiennent à jour une carte détaillée indiquant la position de chaque bloc du monde de Minecraft, utiliser la technique de localisation par position à chaque fois que tu en as besoin n'est pas la solution la plus flexible qui existe, sans compter que tes programmes vont devenir plus complexes. Bien sûr, tu peux toujours dessiner ta propre carte et la conserver quelque part. Mais il serait bien dommage que tu te donnes tout ce mal sachant que ton ordinateur stocke déjà ces informations en mémoire pour pouvoir afficher le monde de Minecraft à l'écran.

Heureusement, l'API de Minecraft possède elle aussi une fonction `getBlock()`. Celle-ci t'ouvre les portes de la mémoire interne de Minecraft et te permet d'accéder à la carte du monde. En utilisant les coordonnées, tu peux manipuler la carte pour connaître l'identité de chacun des blocs du monde, et pas seulement celle de celui sur lequel Steve se trouve.

Tu sais également, depuis le chapitre 3, comment modifier n'importe quel bloc du jeu pour lui donner une autre identité. Heureusement, les fonctions `setBlock()` et `getBlock()` sont liées. Donc, si tu utilises `setBlock()` pour définir l'identité d'un bloc et `getBlock()` immédiatement après, tu récupéreras exactement la même identité.

Tu vas maintenant te lancer dans la construction d'un fantastique jeu, mais cela va avoir pour conséquence d'allonger ton programme. Or, la meilleure façon de bâtir un long programme, c'est de rédiger d'abord de petits morceaux de scripts et de les assembler ensuite, après s'être assuré de leur bon fonctionnement. Commençons donc par rédiger un simple script pour savoir si ton joueur se trouve sur un terrain stable.

Copyright © 2016 Eyrolles.

Comment savoir si tes pieds touchent la terre ferme

Lance Minecraft, IDLE ainsi que le serveur si tu es sur un PC ou un Mac. Tu t'es déjà certainement un peu familiarisé avec ce processus, mais n'hésite pas à relire le chapitre 1 si tu as besoin de te rafraîchir la mémoire. Tu vas maintenant construire un programme qui te fournira des informations exactes sur l'emplacement de ton joueur. En somme, tu dois savoir si ton joueur a les pieds sur terre avant de pouvoir te lancer dans la construction d'un pont magique.

1. Ouvre une nouvelle fenêtre en cliquant sur `File>New File`. Enregistre ton programme en le nommant `piedsATerre.py`. N'oublie pas d'enregistrer tes programmes dans le dossier `MyAdventures`, sinon ils ne fonctionneront pas.

2. Ensuite, importe les modules dont tu as besoin. Tu dois utiliser le module `Minecraft` habituel ainsi que le module `block` pour interagir avec des blocs :

```
import mcpi.minecraft as minecraft
import mcpi.block as block
import time
```

3. Puis connecte-toi au jeu Minecraft :

```
mc = minecraft.Minecraft.create()
```

4. Étant donné que plus tard tu vas inclure certaines parties de ce script dans un programme plus important, tu dois pour l'instant rédiger l'essentiel de ton code dans une fonction que tu baptiseras `piedsATerre()`. Tu pourras ainsi réutiliser ton code plus facilement par la suite. En premier lieu, cette fonction va se contenter de récupérer les coordonnées géographiques de Steve :

```
def piedsATerre():
    pos = mc.player.getTilePos()
```

5. `getBlock()` capture l'identité d'un bloc en fonction des coordonnées que tu lui fournis. Comme `pos.x`, `pos.y` et `pos.z` sont les coordonnées de ton joueur, tu dois saisir `pos.y-1` pour cibler le bloc qui se trouve sous les pieds de Steve :

```
    b = mc.getBlock(pos.x, pos.y-1, pos.z)
       # remarque : pos.y-1 est très important !
```

6. Pour savoir si ton joueur est en sécurité, tu dois simplement faire preuve de logique. S'il marche sur l'eau ou dans le vide, c'est qu'il est en danger. Sinon, il est en sécurité. Comme les blocs « stables » existent par centaines, il est plus simple d'identifier les types de blocs « dangereux ». L'instruction suivante est assez longue, alors assure-toi de bien la saisir sur une seule ligne :

```
    if b == block.AIR.id or b == block.WATER_STATIONARY.id
       or b == block.WATER_FLOWING.id:
        mc.postToChat("en danger")
```

Copyright © 2016 Eyrolles.

7. Si le bloc sur lequel Steve se trouve fait partie de la catégorie des blocs dangereux, tu publies un message sur le chat de Minecraft indiquant que ton joueur est en danger. Si, en revanche, il marche sur les blocs de l'autre catégorie, tu peux considérer qu'il est en sécurité.



Assure-toi d'indenter correctement tes lignes de code, sinon ton programme ne fonctionnera pas. Pour rappel, le code contenu dans une fonction doit être indenté d'un cadratin et toutes les lignes situées à l'intérieur des instructions `if` et `else` doivent être indentées d'un cadratin supplémentaire. Les instructions `if` et `else` doivent se succéder, car elles sont liées. Elles se trouvent logiquement au même niveau.

```
        mc.postToChat("en danger")
    else:
        mc.postToChat("en securite")
```

8. La fonction `piedsATerre()` est maintenant terminée. Laisse une ligne vide juste après pour indiquer que la fonction s'arrête là. Ensuite, reviens à la ligne sans indentation, pour commencer à saisir l'instruction `while` de ta boucle de jeu. Comme dans tes scripts précédents, introduis un court laps de temps puis utilise ta nouvelle fonction `piedsATerre()` qui se chargera de faire tout le travail.

```
while True:
    time.sleep(0.5)
    piedsATerre()
```

Maintenant, clique sur `File>Save` pour enregistrer ton programme et exécute-le en cliquant sur `Run>Run Module` dans le menu de l'éditeur.

Tu ne remarques rien ? Si tu te déplaces, les mots « en sécurité » ou « en danger » s'affichent dans le chat, comme le montre la figure 4-1. Essaie de t'envoler dans le ciel ou de faire quelques longueurs dans la mer pour voir ce qui se passe.



Dans ton programme `piedsATerre.py`, comme dans les programmes précédents, tu as introduit un court laps de temps pour ralentir ta boucle de jeu principale. Cela n'est pas toujours nécessaire. Plus tard, par exemple, les programmes sur lesquels tu travailleras risqueront de ne pas fonctionner correctement si tu leur imposes des délais. Mais il est parfois pratique de ralentir un peu la cadence, notamment lorsque tu utilises le chat pour publier des messages, car celui-ci peut vite s'encombrer. Pour te donner une idée de l'utilité de ce délai, raccourcis-le ou retire-le complètement de ton programme.

Copyright © 2016 Eyrolles.



FIGURE 4-1 Le programme piedsATerre te signale que tu es en danger lorsque tu te trouves dans l'eau.

Construire des ponts magiques

Dans ton précédent programme, tu as écrit un script permettant de détecter les blocs que tu foules du pied et de publier un message dans le chat. En faisant cela, tu as posé les bases d'un programme plus grand que tu développeras plus tard. Mais voyons pour l'heure si tu peux mettre ce script en contexte en utilisant quelques techniques que tu as apprises dans le chapitre 3 à l'aide de la fonction `setBlock()`.

En modifiant légèrement ton script `piedsATerre.py`, tu peux en faire un programme de construction qui fait apparaître un pont de verre sous tes pieds à chaque pas que tu réalises, t'évitant ainsi de tomber à l'eau ou dans le vide ! Comme tu vas réutiliser cette nouvelle fonction plus tard dans ce chapitre, veille bien à la nommer correctement.

1. Clique sur `File>Save As` et remplace le nom de ton programme `piedsATerre.py` par `pontMagique.py`.
2. Modifie également le nom de la fonction `piedsATerre()` pour l'appeler `construirePont()`, et modifie l'instruction `if/else` comme indiqué en gras, en remplaçant `mc.postToChat()` par `mc.setBlock()`. Ainsi, dès que ton joueur mettra un pied dans l'eau ou dans le vide, ce script construira un pont en verre pour le porter. La ligne de ton instruction `if` est longue, fais donc attention à la saisir correctement :

```
def construirePont():
    pos = mc.player.getTilePos()
    b = mc.getBlock(pos.x, pos.y-1, pos.z)
    # remarque : pos.y-1 est très important !
    if b == block.AIR.id or b == block.WATER_STATIONARY.id ↩
```

Copyright © 2016 Eyrolles.

```
    or b == block.WATER_FLOWING.id:
        mc.setBlock(pos.x, pos.y-1, pos.z, block.GLASS.id)
```

3. Comme tu peux le voir dans la deuxième étape, l'instruction `else` et la fonction `mc.postToChat()` ont été enlevées du code, car tu n'en as plus besoin.

4. Si ton pont met trop de temps à se construire, ton joueur tombera dans le vide. Tu dois donc supprimer le délai que tu as introduit tout à l'heure dans la boucle de jeu. Enfin, n'oublie pas d'utiliser ta nouvelle fonction `construirePont()` :

```
while True:
    construirePont()
```

5. Enregistre ton programme en cliquant sur `File>Save` dans le menu de l'éditeur.

Exécute ton programme et amuse-toi à sauter dans le vide ou à marcher sur l'eau. Dès lors que ton joueur se trouvera en danger, un pont en verre apparaîtra comme par magie sous ses pieds pour le protéger, comme tu peux le voir sur la figure 4-2. Il peut maintenant marcher sur l'eau. Quel miracle !



Ce programme fonctionne à merveille sur Raspberry Pi. Mais sur PC et sur Mac, le temps de réponse n'est pas aussi rapide, ce qui peut exposer ton joueur à des chutes si les blocs n'apparaissent pas à temps. Prends ton temps et ne te laisse pas aller à courir trop vite dans le ciel en pensant que le pont garantira ta sécurité. Pour améliorer ton jeu, tu peux allonger ou raccourcir les délais, ou encore ramper en enfonçant la touche `Maj` tout en avançant.



FIGURE 4-2 Le programme pontMagique.py construit un pont en verre permettant à ton joueur de marcher au-dessus de l'eau.

Copyright © 2016 Eyrolles.

Dans le programme `pontMagique.py`, tu as supprimé le délai qui permettait de ralentir la boucle principale. Que se passe-t-il si tu réintroduis un délai ? Fais l'essai en introduisant des délais de longueurs différentes, entre 0,1 et 2 secondes, et note comme ils altèrent la création de ton pont magique. Quelle durée te convient le mieux ?



PERCER LES SECRETS DU CODE

Arrêtons-nous un instant sur les fonctions, car nous n'avons pas encore tout à fait élucidé le mystère à leur sujet.

Depuis le début de ce livre, tu utilises des fonctions. Par exemple, `mc.postToChat()` et `mc.getTilePos()` sont des fonctions qui font toutes deux partie de l'API de Minecraft. Tu as également créé tes propres fonctions en utilisant le mot-clé `def`, comme dans le chapitre 2 où tu as conçu la fonction `maison()`. Mais que cachent réellement ces fonctions lorsque tu les utilises dans ton code ?

Jusqu'à présent, j'ai un peu triché en te disant : « utilise telle fonction par-ci » ou « introduis telle fonction par-là », mais j'aurais dû en fait employer un terme plus approprié. En effet, lorsque tu utilises `maison()` ou `construirePont()` dans ton code, on dit que tu **appelles** ces fonctions. Qu'est-ce que cela veut dire ?

À chaque fois que ton programme Python s'exécute, l'ordinateur se souvient toujours des instructions sur lesquelles il travaille, comme si un doigt invisible pointait la ligne de code en cours d'exécution. Normalement, ce doigt invisible se déplace sur la page de haut en bas. De même, lorsque tu utilises une boucle, le doigt invisible suit le même procédé puis revient sur la première ligne et ainsi de suite, jusqu'à ce que la boucle se termine.

Mais lorsque tu fais appel à une fonction, quelque chose de fascinant se produit dans les coulisses de ton code. Python se souvient des lignes sur lesquelles le doigt invisible s'est arrêté – imagine qu'il y colle comme un Post-It – puis il se plonge dans les lignes de code de ta fonction, par exemple la fonction `construirePont()`. Lorsqu'il arrive à la fin de ta fonction `construirePont()`, il revient sur le Post-It qu'il a laissé en marge et reprends sa course à partir de là.

Les fonctions sont utiles à bien des égards, et principalement pour ces deux raisons : elles te permettent de diviser un long programme en petites parties de code, et tu peux réutiliser leur code à plusieurs reprises au sein du même programme.

C'est notamment grâce à cela que tu pourras alléger tes programmes et en faciliter la modification.

Copyright © 2016 Eyrolles.



Lorsque tu **appelles** une fonction, Python se souvient de l'endroit où il s'arrête dans ton script et plonge temporairement au sein de la fonction, à partir de `def`, c'est-à-dire au début de ta fonction. Lorsqu'il arrive à la fin de la fonction, il fait un saut en arrière, là où il s'y est introduit.

Utiliser les listes Python en guise de mémoire magique

Dans tous les programmes que tu as rédigés jusqu'à présent, tu as utilisé certaines formes de variables pour enregistrer des informations qui changent à mesure que le programme s'exécute. Chacune d'entre elles est dotée d'un nom et d'une valeur et si tu souhaites que tes programmes se souviennent de plus d'un élément à la fois, tu dois utiliser plusieurs variables.

Cependant, ces variables ont une taille fixe et ne peuvent contenir qu'un seul élément à la fois : soit un nombre, soit une chaîne de caractères. Dans la plupart des programmes que tu vas développer en tant que programmeur Minecraft, tu vas avoir besoin d'enregistrer un nombre indéfini d'objets. Or tu ne connais jamais à l'avance le nombre d'objets que tu vas insérer dans un programme.

Heureusement, comme tout langage de programmation moderne, Python dispose d'une fonctionnalité intitulée **liste** qui peut contenir plusieurs données à la fois.



Une **liste** est un type de variable qui permet d'entreposer plusieurs données dans un seul et même endroit. Parmi toutes les possibilités qu'elle t'offre, tu peux notamment y ajouter de nouveaux objets, mesurer sa longueur, accéder aux objets à des endroits précis et les supprimer, quel que soit leur emplacement dans la liste. Les listes sont très pratiques pour regrouper des séries d'objets du même ordre. Tu peux par exemple créer une liste de nombres à classer, une liste des membres d'un groupe d'utilisateurs, voire une liste de blocs dans un jeu Minecraft.

Essayer les listes Python

La meilleure manière de comprendre de quoi il retourne, c'est encore de les utiliser dans l'interpréteur de commandes Python.

Copyright © 2016 Eyrolles.

Fais bien attention aux parenthèses `()` et aux crochets `[]` dans cette partie. Pour l'instant, tu ne vois peut-être pas la différence, mais tout s'éclaircira à mesure que tu progresseras dans la lecture de cette partie.



1. Affiche l'interpréteur de commandes Python au premier plan en cliquant dessus. Clique à droite de la dernière invite de commandes `>>>`, là où tu commenceras la rédaction de tes instructions. Avant cela, n'oublie pas d'arrêter ton programme précédent en cliquant sur `Shell>Restart Shell` dans le menu de ton interpréteur Python, ou en appuyant simultanément sur `Ctrl+F6`. Sans cela, les instructions qui suivront ne fonctionneront pas.

2. Crée une liste vide et indique ce qu'elle contient :

```
a = [] # Création d'une liste vide
print(a)
```

3. Ajoute un nouvel objet à ta liste et fais un nouvel appel à `print` :

```
a.append("bonjour")
print(a)
```

4. Ajoute un deuxième objet à ta liste et fais encore un appel à `print` :

```
a.append("minecraft")
print(a)
```

5. Vérifie la longueur de ta liste :

```
print(len(a))
```

6. Accède à certains objets en fonction de leur emplacement dans la liste. Ces emplacements sont appelés des **index** :

```
print(a[0]) # [0] est l'index du premier objet
print(a[1]) # [1] est l'index du second objet
```

7. Retire un mot à la fin de la liste, puis affiche ce mot ainsi que les objets restants de la liste :

```
mot = a.pop()
print(mot)
print(a)
```

8. Vérifie la longueur de ta liste, puis celle de la chaîne de caractères de ton mot :

```
print(len(a))
print(len(mot))
```

Copyright © 2016 Eyrolles.

9. Retire le dernier objet de la liste, affiche l'objet et la liste en question, puis affiche la longueur de la liste :

```
mot = a.pop()
print(mot)
print(a)
print(len(a))
```

La figure 4-3 montre l'interpréteur de commandes Python affichant les résultats de cet essai.



Un **index** est un nombre qui permet d'accéder à un objet en particulier dans une liste donnée. Pour signaler un index, tu dois utiliser des crochets, comme cela. `a[0]` représente le premier objet de la liste et `a[1]` le second. Les index en Python commencent toujours par le chiffre 0. 0 désigne donc toujours le premier objet d'une liste. Dans les instructions que tu viens de rédiger, tu as indexé des objets dans une liste.

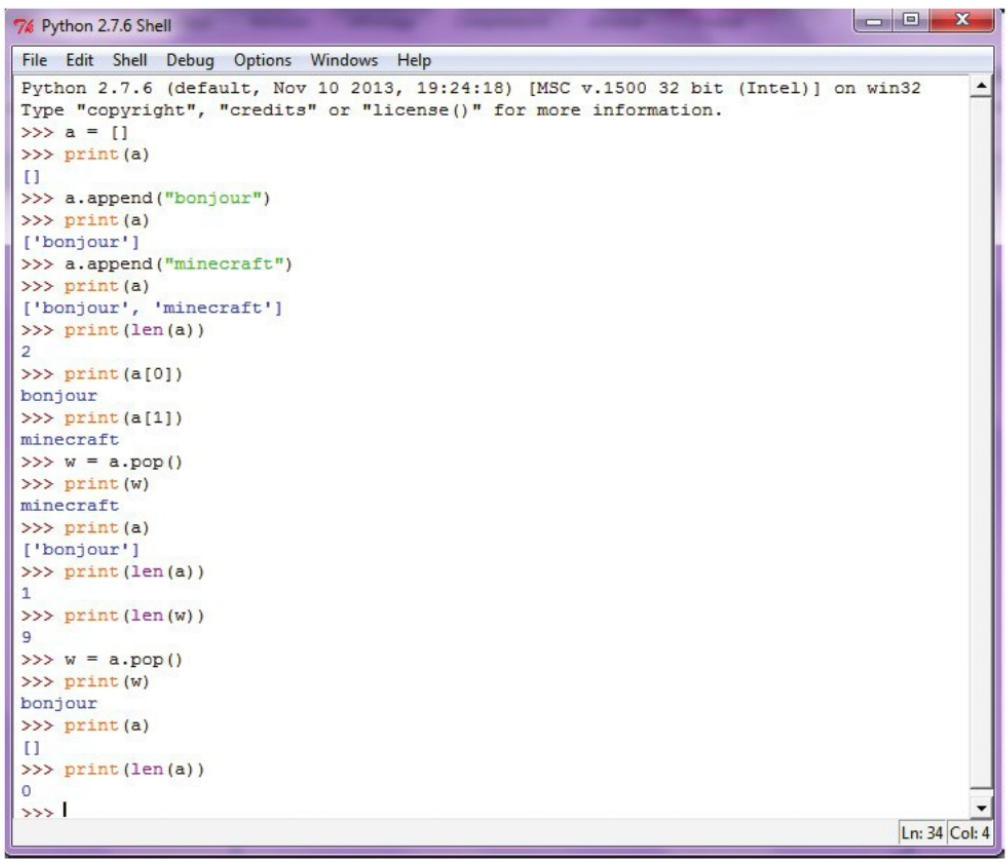


FIGURE 4-3 Exemple de liste dans l'interpréteur de commandes Python

Copyright © 2016 Eyrolles.

La longueur d'une liste dépend du nombre d'objets que tu y insères ou que tu en retires. Que se passe-t-il si tu essaies d'accéder à un objet inexistant au sein d'une liste ? Essaie de saisir les données suivantes dans ton interpréteur de commandes Python et observe ce qui se passe :




```
b = []

print(b[26])
```

D'après toi, comment peux-tu éviter ce problème dans tes programmes ? Remarque : il existe plusieurs bonnes réponses à cette question. Pour les connaître, fais quelques recherches sur Internet pour apprendre à contourner ce problème.

Une liste Python est un type de variable polyvalent qu'on peut utiliser pour stocker plusieurs types d'objets. Mais ce n'est pas l'unique type de variable Python capable de cela. À ce stade, ce que tu dois surtout retenir des listes, c'est que tu n'as pas besoin de connaître leur longueur à l'avance pour exécuter un programme. Une liste grandira et rétrécira automatiquement à mesure que tu utiliseras les méthodes `append()` et **pop()** pour ajouter des objets à la fin ou pour en supprimer.



Lorsque tu utilises la méthode **pop** dans une liste, elle a pour effet de supprimer le dernier objet de cette liste.



Construire des ponts éphémères à l'aide d'une liste Python

Tu vas maintenant utiliser ces nouvelles connaissances pour rédiger un programme de construction de ponts qui les fera disparaître une fois que ton joueur aura touché la terre ferme. Ce programme ressemble à `pontMagique.py`, alors crée un nouveau fichier à partir de celui-là et renomme-le. Nous avons cependant montré le programme dans son ensemble ici pour procéder étape par étape. Copie et colle ton programme `pontMagique.py` pour t'éviter de tout saisir à nouveau et pour gagner un peu de temps.

Copyright © 2016 Eyrolles.



Tu peux regarder une vidéo didactique sur la création du jeu des ponts magiques en te rendant sur le site Internet du livre www.editions-eyrolles.com/dl/14292. Choisis la vidéo « Adventure 4 » (en anglais).

1. Crée un nouveau fichier en cliquant sur File>New File, nomme-le pontEvanescent.py et clique sur File>Save As dans le menu de l'éditeur.

2. Importe ensuite les modules dont tu as besoin :

```
import mcpi.minecraft as minecraft
import mcpi.block as block
import time
```

3. Puis connecte-toi au jeu Minecraft :

```
mc = minecraft.Minecraft.create()
```

4. À présent, tu vas créer une liste de ponts vide. Pour le moment, ta liste ne contient aucun bloc ; elle n'affiche donc aucun contenu :

```
pont = []
```

5. Définis une fonction construirePont() qui construira un pont pour toi. Tu utiliseras cette fonction dans le tout dernier programme de ce chapitre, alors assure-toi de la nommer correctement. La plupart du code qui se trouve au début de cette fonction est identique à celui du programme pontMagique.py. Tu peux donc le copier-coller dans ta fonction pour gagner du temps. Veille à indenter les lignes de code correctement :

```
def construirePont():
    pos = mc.player.getTilePos()
    b = mc.getBlock(pos.x, pos.y-1, pos.z)
    # remarque : pos.y-1 est très important !
    if b == block.AIR.id or b == block.WATER_STATIONARY.id ↩
    or b == block.WATER_FLOWING.id:
        mc.setBlock(pos.x, pos.y-1, pos.z, block.GLASS.id)
```

6. Lorsque tu commences à construire ton pont, tu dois te souvenir de l'endroit où tu as placé tes blocs afin de pouvoir les supprimer par la suite lorsque ton joueur touche la terre ferme. Utilise pour cela une autre liste pour stocker les trois coordonnées au même endroit et les ajouter à la liste des blocs de ton pont. Pour obtenir de plus amples explications à ce sujet, réfère-toi à l'encadré « Percer les secrets du code » p. 120.

```
        coordonnee = [pos.x, pos.y-1, pos.z]
        pont.append(coordonnee)
```

Copyright © 2016 Eyrolles.

7. Pour faire disparaître ton pont lorsque ton joueur touche la terre ferme, tu dois utiliser une instruction `else` et vérifier si ton joueur se trouve sur un bloc en verre ou non. Si ce n'est pas le cas, ton programme va commencer à supprimer les blocs du pont. Ton programme doit également vérifier que ton pont n'a pas complètement disparu, sinon, Python affichera une erreur lorsque tu tenteras de supprimer des éléments de la liste vide. Le mot-clé `elif` est la contraction de « `else if` » ; les symboles `!=` signifient « différent de ». Attention à l'indentation ici : l'instruction `elif` est indentée d'un cadratin, car elle fait partie de la fonction `construirePont()`. En revanche, l'instruction `if` qui suit est indentée de deux cadratins, car elle fait partie de l'instruction `elif` :

```
    elif b != block.GLASS.id:
        if len(pont) > 0:
```

8. Les prochaines lignes sont indentées de trois cadratins, car elles font partie de l'instruction `if`, qui appartient à la fonction `elif`, qui est elle-même indentée sous la fonction `construirePont()` ! Pfiou !

Un peu plus haut dans tes lignes de code, tu as ajouté une liste de trois coordonnées à la liste de blocs de ton pont. Tu t'en souviens ? Maintenant, tu vas y indexer tes coordonnées de la manière suivante : `coordonnée[0]` pour x, `coordonnée[1]` pour y et `coordonnée[2]` pour z. Ajoute aussi la fonction `time.sleep()` pour faire disparaître ton pont progressivement et te laisser le temps de voir la magie opérer :

```
            coordonnee = pont.pop()
            mc.setBlock(coordonnee[0], coordonnee[1], ↩
            coordonnee[2], block.AIR.id)
            time.sleep(0.25)
```

9. Enfin rédige la boucle de jeu principale. Comme dans les exercices précédents, tu auras peut-être envie d'utiliser des temps de pause différents dans ta boucle de jeu pour améliorer ton constructeur de pont. N'oublie pas que cette boucle de jeu fait partie du corps principal de ton programme. En effet, l'instruction `while True:` n'est pas du tout indentée. Fais donc bien attention à ce détail :

```
while True:
    time.sleep(0.25)
    construirePont()
```

Sauvegarde ton programme à l'aide de `File>Save` puis exécute-le en cliquant sur `Run>Run Module`. Déplace ton joueur dans le monde de Minecraft : aventure-toi dans le vide ou sur un lac puis retourne sur la terre ferme. Que vois-tu ? La figure 4-4 montre ton pont en train de disparaître alors que Steve a regagné la terre ferme.

Copyright © 2016 Eyrolles.

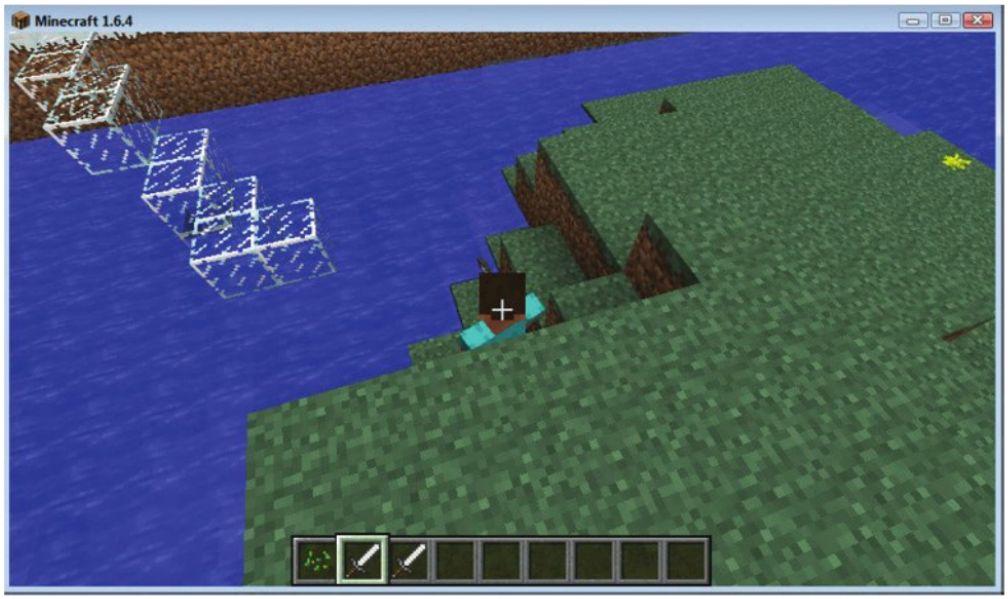


FIGURE 4-4 Le pont disparaît comme par magie une fois que ton joueur regagne la terre ferme.

PERCER LES SECRETS DU CODE

Plus tôt, lorsque tu as fait l'essai de ta première liste dans l'interpréteur de commandes Python, tu lui as ajouté des chaînes de caractères. Mais dans le programme `pontEvanescent.py`, un autre phénomène se produit et mérite qu'on s'y intéresse de plus près.

Les listes Python peuvent contenir presque n'importe quels types de données, tels que des chaînes de caractères, des nombres et même d'autres listes. Tu ne t'en es peut-être pas encore aperçu, mais tu as bel et bien réalisé ce tour de passe-passe dans ton programme.

La ligne suivante crée une liste de trois objets. Il s'agit des coordonnées x, y et z d'un bloc que tu viens de placer :

```
coordonnee = [pos.x, pos.y-1, pos.z]
```

En saisissant des coordonnées entre crochets – `[]` –, tu crées une liste contenant déjà des valeurs au lieu de créer une liste vide. Tu aurais pu procéder de cette façon :

```
coordonnee = []
coordonnee.append(pos.x)
coordonnee.append(pos.y-1)
coordonnee.append(pos.z)
```

Copyright © 2016 Eyrolles.

Plus tard, lorsque ton programme récupère les coordonnées d'un bloc pour le supprimer, il extrait une liste de coordonnées de la liste de ton pont :

```
coordonnee = pont.pop()
```

La liste qui est stockée dans la variable `coordonnee` contient trois objets : `coordonnee[0]` correspond à la position x de ton bloc, `coordonnee[1]` correspond à sa position y et `coordonnee[2]` à sa position z.

La figure 4-5 montre à quoi ressemble cette « liste de listes ».

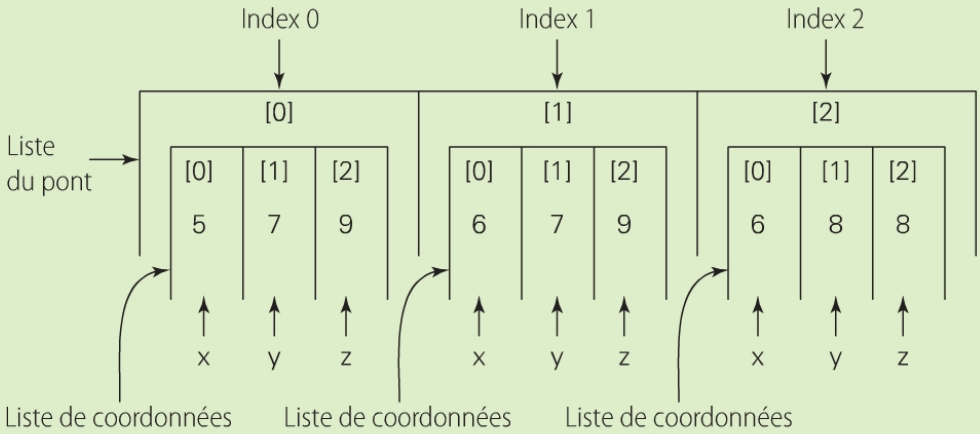


FIGURE 4-5 Un pont est une liste de coordonnées et chaque coordonnée contient une liste.



En fait, en plus des listes que tu viens d'utiliser, Python dispose de plusieurs types de variables capables de stocker des objets. Un programmeur professionnel en langage Python aurait probablement utilisé une autre fonctionnalité dénommée **tuple** pour stocker des coordonnées dans ce programme. Afin d'éviter toute confusion, nous avons préféré limiter la présentation de nouveaux concepts. Mais tu es libre d'effectuer des recherches en ligne sur les **tuples** et d'étudier ce qui les différencie des listes ainsi que les avantages qu'ils apportent à un programme Python.

Copyright © 2016 Eyrolles.

Détecter les blocs que ton joueur frappe

Ta boîte à outils est presque complète. Il ne te reste plus qu'une chose à y ajouter avant de te lancer dans l'aventure, à savoir : détecter les blocs que ton joueur frappe. En acquérant cette compétence, tu auras la possibilité de créer des jeux et des programmes vraiment intéressants, car elle permettra à ton joueur d'interagir directement avec chaque bloc du jeu Minecraft.

Comme il s'agit d'un exercice indépendant des autres, commence par créer un nouveau programme. Tu l'intégreras plus tard au dernier programme de ce chapitre.

1. Ouvre un nouveau fichier à l'aide de `File>New File` dans la fenêtre de l'éditeur. Clique sur `File>Save As` et nomme ton fichier `frappe.py`.
2. Importe les modules dont tu as besoin :

```
import mcpi.minecraft as minecraft
import mcpi.block as block
import time
```

3. Puis connecte-toi au jeu Minecraft :

```
mc = minecraft.Minecraft.create()
```

4. Calcule ensuite la position de ton joueur et choisis un emplacement légèrement à côté pour y faire apparaître un diamant. Celui-ci constituera ton trésor :

```
mc = minecraft.Minecraft.create()
pos_diamant = mc.player.getTilePos()
pos_diamant.x = pos_diamant.x + 1
mc.setBlock(pos_diamant.x, pos_diamant.y, pos_diamant.z, ↩
 block.DIAMOND_BLOCK.id)
```

5. Maintenant, définis une fonction que tu nommeras `verifFrappe()`. Celle-ci te servira dans ton programme final, donc veille à la nommer correctement :

```
def verifFrappe():
```

6. Demande à l'API de Minecraft de te fournir une liste d'événements qui sont survenus dans le jeu. Il s'agit simplement d'utiliser une liste Python, comme tu l'as fait plus tôt dans ton programme `pontEvanescent.py` :

```
    evenements = mc.events.pollBlockHits()
```

7. Provoque chacun des événements tour à tour, en utilisant une boucle `for`. Pour en savoir plus sur cette nouvelle façon d'utiliser la boucle `for`, consulte l'encadré « Percer les secrets du code ».

```
    for e in evenements:
```

Copyright © 2016 Eyrolles.

```
        pos = pos.e
```

8. Demande à ton programme de voir si la position du bloc que ton joueur vient de frapper avec son épée est la même que celle du diamant. Si c'est le cas, demande-lui de poster un message sur le chat de Minecraft :

```
        if pos.x == diamant_pos.x and pos.y == diamant_pos.y
        and pos.z == diamant_pos.z:
            mc.postToChat("TOUCHÉ")
```

9. Enfin rédige ta boucle de jeu. Pour l'instant, tu vas utiliser une pause d'une seconde pour limiter la rapidité avec laquelle tes messages apparaissent sur le chat Minecraft, mais n'hésite pas à utiliser des durées différentes pour obtenir un résultat qui te conviendra mieux. Fais attention à l'indentation lors de la saisie :

```
while True:
    time.sleep(1)
    verifFrappe()
```

Enregistre ton programme à l'aide de `File>Save As` et exécute-le en cliquant sur `Run>Run Module` dans le menu de l'éditeur.

Déplace légèrement ton joueur pour voir le diamant, puis frappe-le sur chaque face avec ton épée. Que vois-tu ? Comme le montre la figure 4-6, lorsque tu le frappes, le message « TOUCHÉ » (« HIT » en anglais) apparaît dans le chat de Minecraft.



FIGURE 4-6 Tu viens de frapper un bloc.

Copyright © 2016 Eyrolles.



Si tu te trompes de bouton en cliquant et que tu supprimes malencontreusement ton trésor, tu ne pourras plus le frapper ! Tu devras alors exécuter ton programme à nouveau pour créer un autre trésor ou placer manuellement un bloc de diamant au même endroit avant de retenter l'expérience. Si cela arrive, c'est parce que le bloc de type `AIR` ne réagit pas aux coups.

DÉFI



Modifie ton programme `frappe.py` afin qu'il lise la variable `e.face` de l'événement passé dans la boucle `for`. Cette variable `e.face` affiche un numéro différent en fonction de la face qui a été touchée. Commence par afficher `e.face` dans le chat Minecraft puis frappe chaque face du bloc de diamant pour connaître leurs numéros. Enfin, modifie ton programme pour faire apparaître un message différent chaque fois que tu touches une face différente du diamant.

PERCER LES SECRETS DU CODE

Dans le programme de détection des coups portés à un bloc, tu as utilisé une boucle `for` d'une manière différente des premières fois. Voyons ce qui se cache derrière ce nouvel usage, car cette fonctionnalité puissante mérite bien que l'on s'y attarde un peu.

La boucle `for` normale que tu as utilisée dans les chapitres précédents ressemblait à cela :

```
for i in range(6):
    print(i)
```

`range(6)` est en fait une fonction qui génère une liste de nombres `[0,1,2,3,4,5]`. Tu peux le vérifier en saisissant les lignes suivantes dans ton interpréteur de commandes :

```
print(range(6))
```

Tu verras alors la suite de chiffres suivante apparaître :

```
[0, 1, 2, 3, 4, 5]
```

Cela ne te rappelle-t-il rien ? Ce qui se passe en réalité, c'est que la fonction `range()` génère une nouvelle liste de chiffres.

L'instruction Python `for/in` va alors faire tourner tous les objets d'une liste. Pour cela, elle va d'abord enregistrer le premier objet dans la variable de

Copyright © 2016 Eyrolles.

contrôle de boucle (ci-dessus appelée `i`). Elle va ensuite exécuter le corps de la boucle, puis elle va enregistrer le second objet dans la variable de contrôle de boucle et ainsi de suite jusqu'à arriver à la fin de la liste.

Tu peux réaliser cela avec toutes les listes, quel que soit leur contenu. Fais maintenant l'essai suivant dans l'interpréteur de commandes Python :

```
for nom in ["Alex", "Sophie", "Lucie", "Pierre"]:
    print(«bonjour « + nom)
```

Tu peux aussi faire fonctionner une boucle dans une chaîne de caractères, comme cela :

```
nom = "Alex"
for caractere in nom:
    print(caractere)
```

Créer un jeu de chasse au trésor

Jusqu'à présent dans ce chapitre, tu t'es contenté d'accumuler des connaissances et de créer des bribes de code pour tester des fonctionnalités te permettant de détecter des objets dans Minecraft. Il est temps, maintenant, d'assembler tous les morceaux du puzzle pour créer un jeu complet. Le jeu que tu vas créer s'appelle « Chasse dans le ciel ». C'est une chasse au trésor durant laquelle tu devras collecter le plus de blocs de diamants disséminés au hasard dans le ciel, en t'aidant de signaux lumineux. Le gagnant sera le joueur qui aura obtenu le plus de points.

Nous allons compliquer la tâche pour pimenter un peu le jeu : chaque fois que tu avanceras, tu sèmeras derrière toi une traînée d'or qui te coûtera des points. Ainsi, lorsque tu partiras à la recherche de tes trésors, tu devras choisir le chemin le plus court pour ne pas perdre trop de points. Dans le cas contraire, tu verras ton score chuter à une vitesse impressionnante ! Tu devras donc utiliser tes connaissances de navigation dans Minecraft pour aller droit au but et récupérer tes diamants.

À chaque trésor débusqué, tu cumuleras des points et la traînée d'or disparaîtra comme par magie. Mais attention ! En disparaissant, il se peut qu'elle laisse quelques cavités derrière elle. Prends garde à ne pas trébucher sur ce terrain accidenté si tu souhaites économiser des points !

Ce programme se compose, pour l'essentiel, des quelques portions de code réutilisables que tu as déjà rédigées au début du chapitre. Afin de gagner du temps, tu pourras couper et coller ces lignes et les modifier à ta guise. Dans cette section, nous avons toutefois tenu à reproduire le programme dans son intégralité pour nous assurer que tu en saisisses bien tous les tenants et les aboutissants.

Copyright © 2016 Eyrolles.

Les créateurs de logiciels professionnels commencent souvent par bâtir un plan simple de leur programme à l'aide d'instructions `print` qu'ils testent pour s'assurer de sa solidité. Ils l'enrichissent ensuite en y ajoutant progressivement de nouvelles fonctionnalités. Dans ce chapitre, tu vas te glisser dans la peau d'un vrai programmeur professionnel, car tu vas, toi aussi, rédiger ton programme en le testant petit à petit. Commençons donc par poser la structure de la boucle de jeu. Nous verrons ensuite comment l'étoffer avec des fonctions très simples.

Rédiger les fonctions et la boucle de jeu principale

1. Commence par créer un nouveau fichier en choisissant `File>New File` dans le menu. Enregistre-le en cliquant sur `File>Save As` dans le menu et nomme-le `chasseAuTresorDansLeCiel.py`.

2. Importe les modules dont tu as besoin :

```
import mcpi.minecraft as minecraft
import mcpi.block as block
import time
import random
```

3. Puis connecte-toi au jeu Minecraft :

```
mc = minecraft.Minecraft.create()
```

4. Définis ensuite une variable `score` qui réalisera un suivi de tes résultats à mesure que tu avanceras dans le jeu. Tu utiliseras également une constante `ECART` pour compliquer le jeu. Celle-ci te permettra de définir la distance à laquelle tu souhaites semer tes trésors par rapport à la position de ton joueur. Pour commencer, affecte-lui un petit nombre que tu augmenteras par la suite une fois que tu auras testé ton code et que tu compléteras le programme :

```
score = 0
ECART = 5
```

5. Comme tu vas développer ce programme pas à pas, commence par créer des fonctions d'apparat pour chacune des fonctionnalités de ton programme. Les fonctions Python ont besoin de contenir au minimum une instruction pour être valides. Utilise donc l'instruction `print` pour rédiger un message. Cela te servira de balise pour plus tard, car tu remplaceras le texte par des lignes de code :

```
tresor_x = None # la coordonnée x du trésor
def poserTresor():
    print("poserTresor")
def verifFrappe():
    print("verifFrappe")
def annonceLumineuse():
    print("annonceLumineuse")
```

Copyright © 2016 Eyrolles.

```
pont = []
def construirePont():
    print("constuirePont")
```

6. Maintenant, passe à la rédaction de ta boucle de jeu. Dans le jeu final, ta boucle s'exécutera à une vitesse plutôt soutenue (10 tours par seconde) pour que la traînée d'or que tu laisseras derrière toi soit d'une longueur suffisante. Pour les besoins de ton test, réduis-la à un tour par seconde. Cela sera amplement suffisant. Dans le corps de la boucle, tu vas introduire tes fonctions d'apparat que tu remplaceras plus tard par du code.

```
while True:
    time.sleep(1)
    if tresor_x == None and len(pont) == 0:
        poserTresor()
    construirePont()
    annonceLumineuse()
    verifFrappe()
```

7. Sauvegarde ton programme à l'aide de `File>Save` puis exécute-le en cliquant sur `Run>Run Module`. Aucune erreur ne devrait s'afficher. En revanche, tu devrais voir des messages apparaître toutes les secondes dans l'interpréteur de commandes Python. Maintenant que la structure de ton programme est définie, tu peux commencer à ajouter des éléments de code.

Semer des trésors dans le ciel

La première fonction que tu vas rédiger consiste à semer des trésors au hasard dans le ciel. Pour cela, tu vas utiliser trois variables globales pour déterminer les coordonnées du trésor, dont la valeur initiale sera `None`. La valeur `None` est une valeur spéciale en langage Python. Elle indique que la variable est bien enregistrée dans la mémoire, mais qu'elle ne contient rien. Tu l'utiliseras dans ta boucle de jeu pour savoir si tu dois semer un nouveau trésor ou non.

1. Crée les variables globales pour déterminer la position de ton trésor (n'ajoute que les lignes en gras) :

```
tresor_x = None
tresor_y = None
tresor_z = None
```

2. Puis complète la fonction `poserTresor()` en remplaçant la fonction `print` saisie tout à l'heure par ces lignes de code :

```
def poserTresor():
    global tresor_x, tresor_y, tresor_z
    pos = mc.player.getTilePos()
```

Copyright © 2016 Eyrolles.

3. Ensuite, utilise la fonction `random` pour placer le trésor à quelques blocs de ton joueur, dont les valeurs se trouvent dans la constante `ECART`, mais assure-toi de définir la coordonnée `y` pour que ton objet se situe juste au-dessus de ton joueur, quelque part dans le ciel :

```
    tresor_x = random.randint(pos.x, pos.x+ECART)
    tresor_y = random.randint(pos.y+2, pos.y+ECART)
    tresor_z = random.randint(pos.z, pos.z+ECART)
    mc.setBlock(tresor_x, tresor_y, tresor_z,
                block.DIAMOND_BLOCK.id)
```

Exécute ton programme : un trésor devrait apparaître dans le ciel à proximité de ton joueur.

Récupérer un trésor une fois qu'il a été touché

Tu vas maintenant reprendre le code du programme `frappe.py` et le modifier légèrement pour savoir quand un trésor a été touché par l'épée de ton joueur.

1. Retire l'instruction `print` de la fonction `verifFrappe()` et remplace la par le code ci-après. Les variables `score` et `tresor_x` doivent être cataloguées comme des variables globales, car la fonction `verifFrappe()` va modifier leur valeur. En langage Python, il est nécessaire de regrouper au sein d'une même fonction toutes les variables globales dont Python va modifier la valeur. Si tu ne te plies pas à cette règle, ton programme ne fonctionnera pas :

```
def verifFrappe():
    global score
    global tresor_x
```

2. Passe tous les événements de frappe à la loupe et vois si leur position correspond à celle de ton trésor :

```
    evenements = mc.events.pollBlockHits()
    for e in evenements:
        pos = e.pos
        if pos.x == tresor_x and pos.y == tresor_y ↩
and pos.z == tresor_z:
            mc.postToChat("TOUCHÉ !")
```

3. Tu vas maintenant devoir indiquer à ton programme d'ajouter des points au score dès que le joueur touche un trésor, puis de faire disparaître ce dernier. N'oublie pas de définir `tresor_x` sur `None` pour que la fonction `poserTresor()` puisse en créer un nouveau ensuite. Attention aussi à l'indentation : ce code fait partie du corps de l'instruction `if` :

```
            score = score + 10
```

Copyright © 2016 Eyrolles.

```
            mc.setBlock(tresor_x, tresor_y, tresor_z,
                        block.AIR.id)
            tresor_x = None
```

Enregistre et exécute ton programme, puis vérifie que le trésor disparaît bien lorsque ton joueur le frappe avec son épée. Tu devrais également en voir apparaître un nouveau non loin de ton joueur.

Ajouter un signal lumineux

Le signal lumineux servira à afficher ton score ainsi que la distance approximative du nouveau trésor dans le chat de Minecraft, à une seconde d'intervalle. Voici comment l'ajouter.

1. Crée une variable `chronometre`. Comme ta boucle de jeu tournera 10 fois toutes les secondes, tu devras introduire 10 boucles à chaque seconde. C'est ce que ton `chronometre` va t'aider à faire. Si tu modifies la vitesse de la boucle, tu vas également devoir ajuster la valeur `PAUSE`. Ajoute ce code juste au-dessus de la fonction `annonceLumineuse()` (remarque : il n'est pas nécessaire d'indenter le code ici) :

```
PAUSE = 10
chronometre = PAUSE
```

2. Supprime l'instruction `print()` de la fonction `annonceLumineuse()` et ajoute le `chronometre` en tant que variable globale, pour que la fonction puisse en modifier la valeur :

```
def annonceLumineuse():
    global chronometre
```

3. Un trésor apparaîtra dans le ciel si la variable `tresor_x` contient une valeur. Tu dois vérifier que le trésor a bien été créé, car dans le cas contraire le chat de Minecraft t'enverra une annonce lumineuse :

```
    if tresor_x != None:
```

4. Cette fonction sera appelée 10 fois par seconde dans la boucle de jeu, donc tu ne dois mettre à jour l'annonce lumineuse que tous les 10 tours :

```
        chronometre = chronometre - 1
        if chronometre == 0:
            chronometre = PAUSE
```

5. Une fois que le chronomètre s'arrête (après que la fonction a été appelée 10 fois ou toutes les secondes), calcule la distance qui te sépare du trésor. La fonction `abs()` permet de trouver la valeur absolue (une valeur positive) de la différence entre deux positions. En assemblant toutes les différences positives, le résultat augmente à mesure que tu t'éloignes du trésor et diminue lorsque tu t'en

Copyright © 2016 Eyrolles.

approches. Attention à l'indentation dans les lignes suivantes, car ce code appartient au corps de l'instruction `if` précédente :

```
            pos = mc.player.getTilePos()
            diffx = abs(pos.x - tresor_x)
            diffy = abs(pos.y - tresor_y)
            diffz = abs(pos.z - tresor_z)
            diff = diffx + diffy + diffz
            mc.postToChat("score :" + str(score) + "tresor ↩
:" + str(diff))
```

Enregistre et exécute ton programme, puis assure-toi que l'annonce lumineuse et le score s'affichent correctement sur le chat de Minecraft. Comme tu en es encore à l'étape de test et de développement, ta boucle de jeu est définie pour s'exécuter 10 fois plus lentement que prévu. Donc, pour le moment tu devrais voir les messages s'afficher toutes les 10 secondes sur le chat. Pour t'en assurer, compte jusqu'à 10 dans ta tête. Dans ton interpréteur de commandes Python, tu verras encore quelques fonctions d'apparat s'afficher toutes les secondes, car ton programme n'est pas tout à fait terminé.

Ajouter ton constructeur de ponts

Tu vas maintenant ajouter à ton programme l'application de construction de pont que tu as appelée `pontEvanescent.py`. Pour cela, tu vas devoir la modifier légèrement afin qu'elle puisse déterminer si ton joueur se trouve sur de l'or et, dans le cas contraire, créer une traînée d'or.

1. Assure-toi que ta fonction `construirePont()` soit identique aux lignes suivantes. Les lignes importantes qui ont changé par rapport à ton programme `pontEvanescent.py` sont indiquées en gras :

```
pont = []
def construirePont():
    global score
    pos = mc.player.getTilePos()
    b = mc.getBlock(pos.x, pos.y-1, pos.z)
    if tresor_x == None:
        if len(pont) > 0:
            coordonnee = pont.pop()
            mc.setBlock(coordonnee[0], coordonnee[1], ↩
 coordonnee[2], block.AIR.id)
            mc.postToChat("pont:" + str(len(pont)))
            time.sleep(0.25)
    elif b != block.GOLD_BLOCK.id:
        mc.setBlock(pos.x, pos.y-1, pos.z, ↩
    block.GOLD_BLOCK.id)
        coordonnee = [pos.x, pos.y-1, pos.z]
        pont.append(coordonnee)
        score = score - 1
```

Copyright © 2016 Eyrolles.

2. Félicitations ! Ton programme est maintenant terminé. Comme ton jeu est prêt à être lancé, modifie l'instruction `time.sleep(1)` dans la boucle de jeu et réduis sa vitesse à 0,1 seconde pour que ta boucle tourne 10 fois par seconde. Le chronomètre de la fonction `annonceLumineuse` comptera 10 tours de boucle et affichera un message à chaque seconde écoulée dans le chat de Minecraft.

Enregistre ton programme et exécute-le. Vérifie que la traînée d'or disparaît bien lorsque tu récupères un trésor et que ton score diminue à chaque bloc d'or que tu laisses derrière toi.

Laisse maintenant les calculs de côté et amuse-toi ! Tu verras qu'il n'est pas si facile de conserver un score élevé, mais le jeu en vaut la chandelle.

Sur la figure 4-7, tu peux voir le score et l'annonce lumineuse s'afficher dans le chat Minecraft.



FIGURE 4-7 L'annonce lumineuse t'indique tes données de vie pendant le jeu de chasse au trésor.

Copyright © 2016 Eyrolles.

Tableau de références	
Obtenir l'identité d'un bloc à un endroit précis	**Connaître les blocs qui ont été touchés**

`b = mc.getBlock(10, 5, 2)`	`frappes = mc.events.pollBlockHits()`
	`for frappe in frappes:`
	`pos = frappe.pos`
	`print(pos.x)`

Créer des listes	**Ajouter un objet à la fin d'une liste**
`a = [] # On crée une liste vide` `a = [1,2,3] # On crée le début d'une liste #`	`a.append("bonjour")`
Afficher le contenu d'une liste	**Connaître la taille d'une liste**
`print(a)`	`print(len(a))`
Accéder aux objets d'une liste grâce à leur index	**Accéder au dernier objet d'une liste**
`print(a[0]) # 0=premier, 1=deuxième`	`print(a[-1])`
Supprimer le dernier objet d'une liste	**Passer tous les objets d'une liste en revue**
`mot = a.pop() # supprime le dernier objet #` `print(mot) # l'objet vient d'être supprimé #`	`for objet in a:` `print(objet)`

Explorer d'autres pistes d'interaction

Dans ce chapitre, tu as appris à utiliser la fonction `getBlock()` pour détecter l'identité du bloc sur lequel ton joueur se trouve. Tu as aussi appris à utiliser `events.pollBlockHits()` pour indiquer un événement dans le chat. Sans compter que tu as bâti un jeu fantastique dans Minecraft en le dotant, en prime, d'un système de points !

- Paramètre ta constante `ECART` de manière à augmenter la distance entre le prochain trésor et ton joueur. Cela te rendra la tâche un peu plus complexe et tu pourras même ajouter une fonctionnalité supplémentaire pour te guider dans ta quête. Tu pourras par exemple poster « chaud » ou « froid » dans le chat si tu te rapproches ou t'éloignes du but.
- Conçois un meilleur système de points pour augmenter tes résultats et obtenir un meilleur score. N'hésite pas à réaliser plusieurs essais pour aboutir à un système équitable, c'est-à-dire à un système qui te récompense suffisamment lorsque tu

Copyright © 2016 Eyrolles.

trouves un trésor et qui ne te pénalise pas trop lorsque tu pars à la recherche des trésors.

- Réalise quelques recherches sur Internet et renseigne-toi sur le théorème de Pythagore. Peux-tu estimer une meilleure distance pour affiner ta fonction `annonceLumineuse()` ? (Psst ! J'ai un secret à te confier : Martin aborde la question dans le chapitre 7. Fais-y un tour, peut-être que tu pourras récupérer quelques informations utiles pour t'aider.)



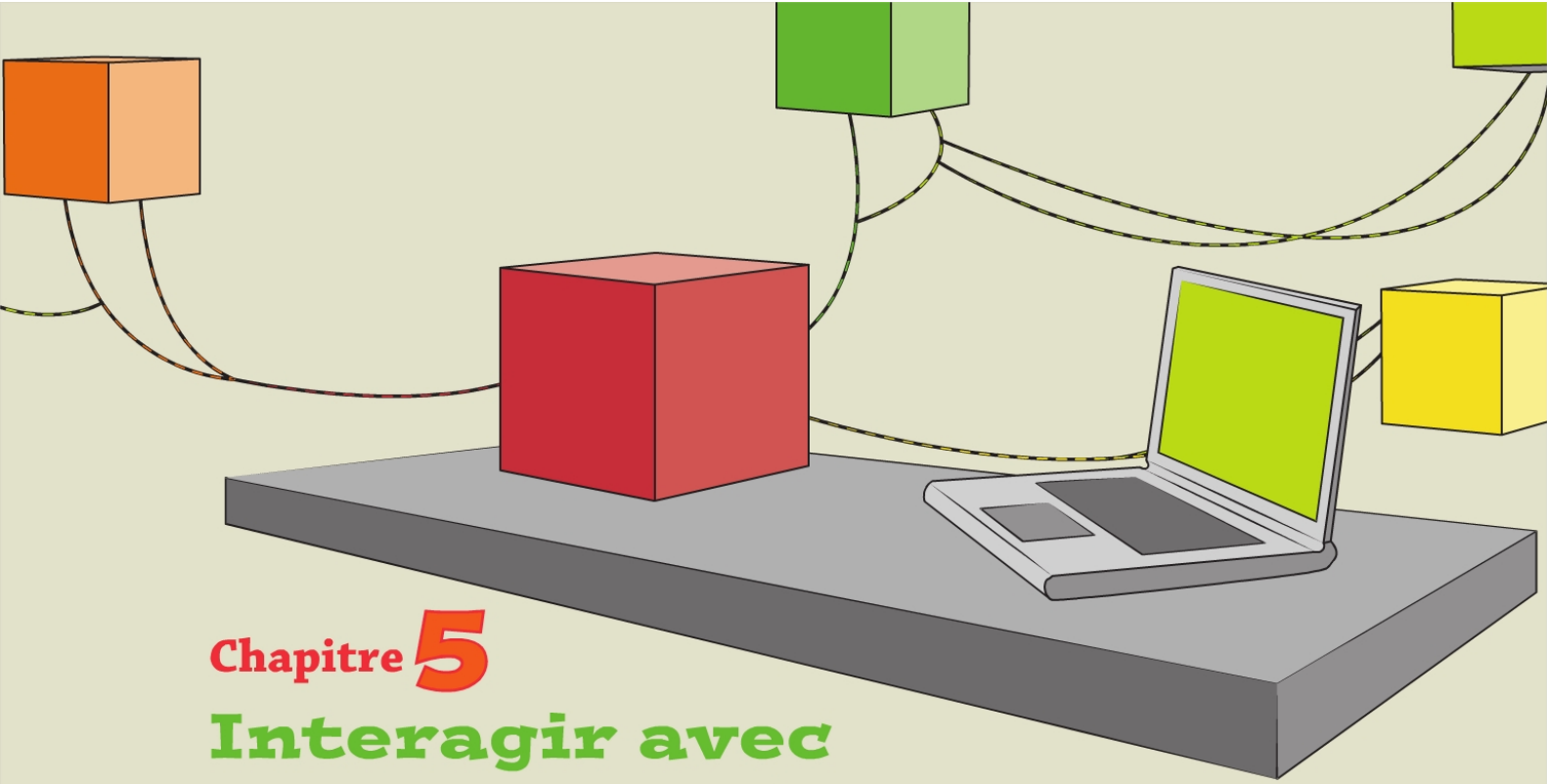
Niveau terminé : dans cet épisode, tu as accompli deux miracles pour le prix d'un. Tu as défié les lois de la gravité et marché sur l'eau !

Dans le prochain chapitre...

Dans le chapitre 5, tu vas apprendre à t'évader du monde virtuel de Minecraft, au-delà même des frontières de ton ordinateur, en reliant le jeu à des objets du monde réel. Tu vas utiliser des composants électroniques pour construire ton propre contrôleur de jeu et tu vas bâtir un nouveau jeu dans le jeu !

Copyright © 2016 Eyrolles.

Copyright © 2016 Eyrolles.



Chapitre 5
Interagir avec des circuits électroniques

LORSQUE TU JOUES à Minecraft, même via l'interface de programmation, tu évolues dans un monde virtuel. L'unique moyen que tu as d'interagir avec le jeu est d'utiliser ton clavier et ta souris pour manipuler les commandes définies par les programmeurs qui ont conçu le jeu.

Mais il existe une autre manière d'interagir avec Minecraft. Pour cela, il faut briser les barrières du jeu bac à sable et le relier au monde physique. Tu découvriras alors que la frontière qui sépare le monde virtuel du monde réel peut vite s'effacer, rendant ton jeu bien plus fertile et captivant.

Dans ce chapitre, tu vas apprendre à connecter Minecraft à de petits circuits électroniques en te servant de l'API. Tu vas commencer par faire clignoter une diode chaque fois que tu franchiras la porte de ta maison. Puis tu vas ajouter à ton dispositif ce qu'on appelle communément un afficheur 7 segments, qui te permettra d'activer un compte à rebours et bien d'autres informations sur ton jeu. Tu ajouteras ensuite un bouton-poussoir pour provoquer toutes sortes d'actions utiles dans Minecraft. Pour finir, tu assembleras le tout afin de créer un détonateur rouge entièrement fonctionnel doté d'un compte à rebours, comme celui que tu peux voir sur la figure 5-1. Grâce à cela, tu n'auras plus aucun mal à libérer de l'espace dans Minecraft et tes amis seront si ébahis qu'ils s'empresseront de te demander de leur en confectionner un !

Copyright © 2016 Eyrolles.

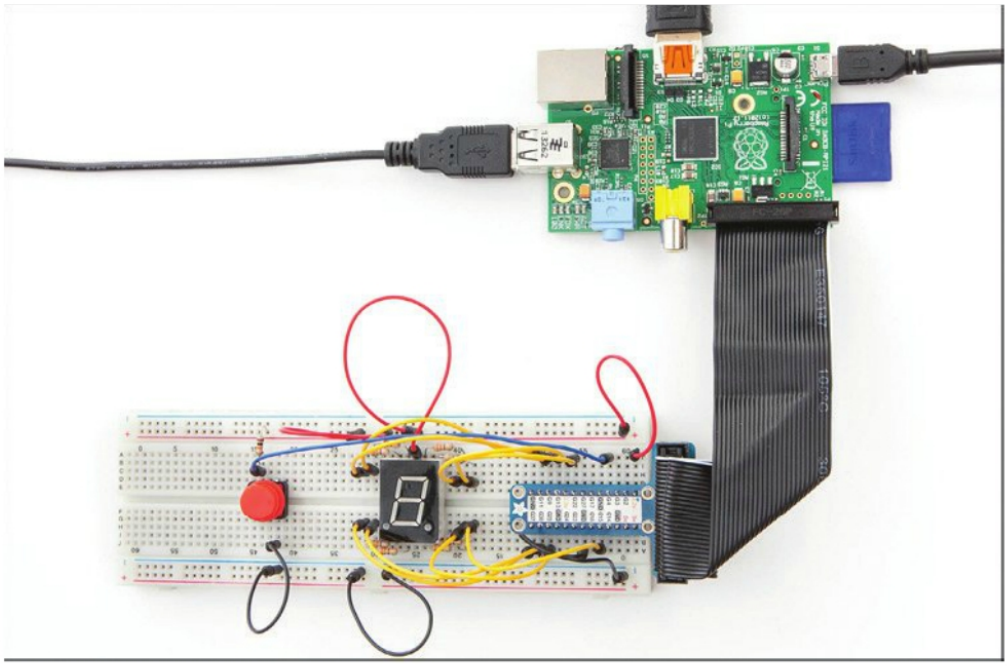


FIGURE 5-1 Un détonateur à bouton-poussoir connecté à un Raspberry Pi

Ce dont tu as besoin dans ce chapitre

Voici une liste complète de tous les éléments qui te serviront à construire tes circuits dans ce chapitre. Tu auras notamment besoin de composants électroniques, mais aussi d'autres outils te permettant de les connecter à ton ordinateur. Tous les composants requis sont exposés dans les figures 5-2 et 5-3 :

- 1 plaque d'essais ;
- au moins 4 boutons-poussoirs ;
- au moins 1 LED de la couleur de ton choix (mais plus tu auras de couleurs, plus l'exercice sera amusant !) ;
- 1 afficheur 7 segments ;
- au moins 10 résistances de 330 ohms à utiliser avec tes LED et ton afficheur ;
- au moins 4 résistances de 10 Kohms à utiliser avec tes boutons ;
- au moins 20 fils-jarretières avec connecteurs, ou des petits fils à âme rigide et une pince à dénuder ;
- 1 petit coupleur 2 piles AA ou AAA.

Copyright © 2016 Eyrolles.

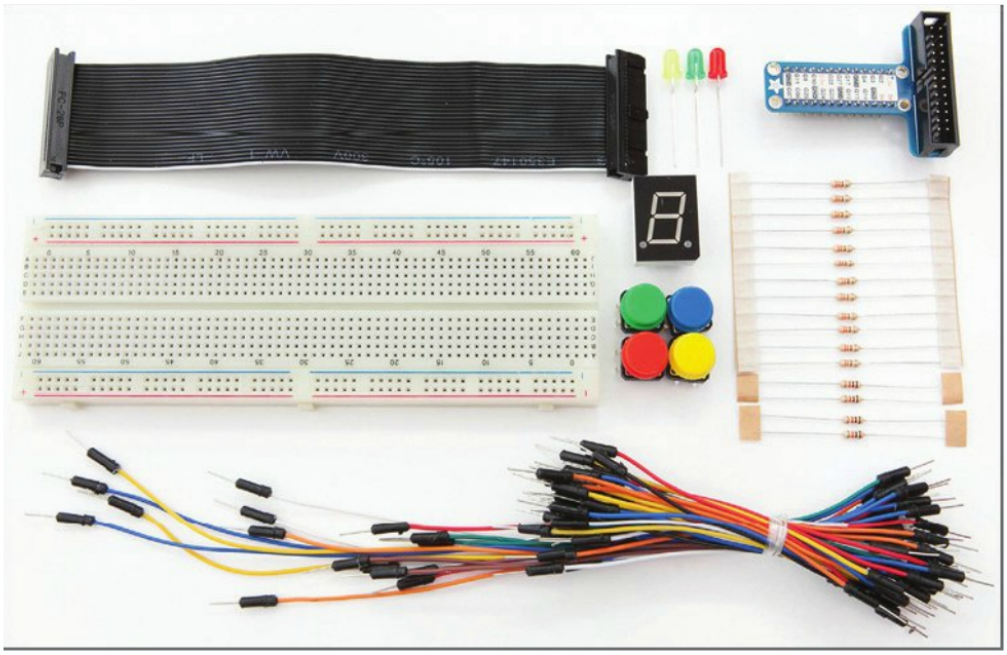


FIGURE 5-2 Les composants requis pour un Raspberry Pi

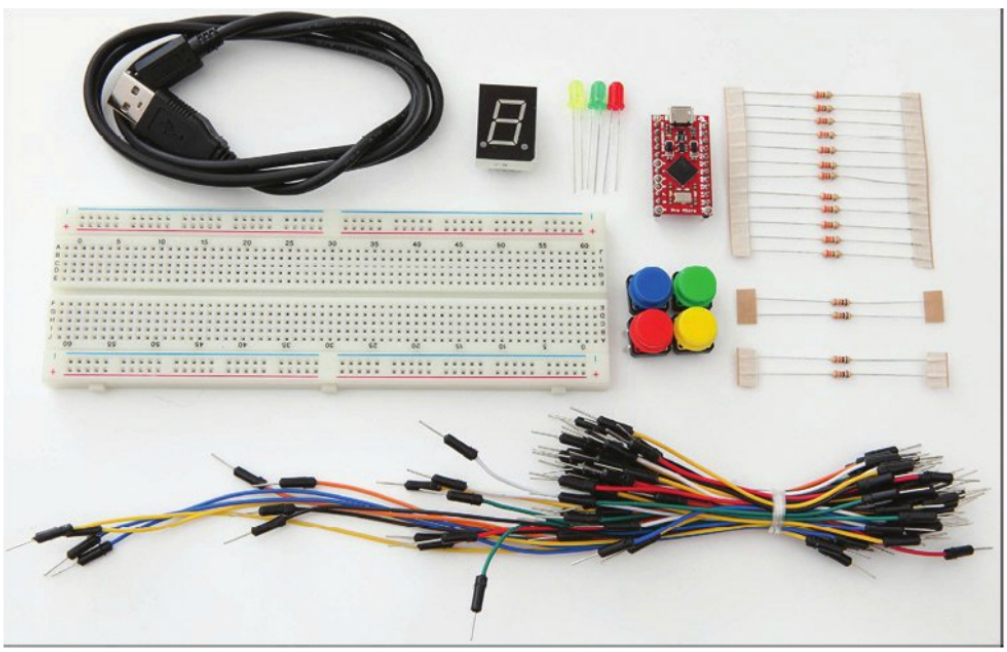


FIGURE 5-3 Les composants requis pour un PC ou un Mac

Sur Internet, les fournisseurs de composants électroniques ne manquent pas. Tu peux acheter les composants sur plusieurs sites, tels que : `www.selectronic.fr`, `www.conrad.fr`, `www.electronique-diffusion.fr`, ainsi que dans la plupart des magasins d'électronique comme Maplin Electronics (`www.maplin.co.uk`). Certains magasins vendent les

Copyright © 2016 Eyrolles.

composants par sachets, surtout en ce qui concerne les LED et les résistances, mais tu pourras aussi trouver des sacs de fils-jarretières avec connecteurs. Si toutefois tu n'en trouves pas, tu peux toujours acheter du fil à âme rigide au mètre et une pince à dénuder pour créer tes propres fils-jarretières. Dans la suite de ce chapitre, nous parlerons de fils-jarretières, mais les instructions sont les mêmes si tu utilises des fils à âme rigide.

Sur Raspberry Pi

Utilise les connecteurs des fils-jarretières pour les rattacher directement aux broches de ton Raspberry Pi puis branche-les à ta plaque d'essais. Cette opération requiert beaucoup de minutie et compter toutes les broches peut s'avérer fastidieux. C'est pourquoi, dans ce chapitre, j'ai préféré utiliser un Pi-T-Cobbler d'Adafruit. Il se compose d'un câble plat qui part du Raspberry Pi et se branche directement à la plaque d'essais par des broches numérotées. Si tu sais souder, tu peux choisir un kit Pi-T-Cobbler et le souder toi-même, ce qui n'est pas plus mal, car cela te permettra d'économiser un peu d'argent. La figure 5-2 t'indique tous les composants dont tu as besoin pour réaliser les exercices de ce chapitre avec un Raspberry Pi.

Pour te procurer le Pi-T-Cobbler d'Adafruit, rends-toi directement sur les sites `www.adafruit.com` ou `www.lextronic.fr` et choisis le Pi-T-Cobbler pré-assemblé, sauf si tu souhaites t'amuser à souder les composants toi-même.



Au moment où nous avons rédigé ce livre, Raspberry Pi publiait son modèle B+. Tous les chapitres de ce livre sont donc écrits pour le Raspberry Pi B+, mais nous te conseillons tout de même d'acheter un fil supplémentaire avec ton Pi-T-Cobbler pour que tu puisses le connecter correctement au Raspberry Pi B+ à 40 broches. Le câble dont tu as besoin se termine par un embout à 40 broches qui se branche au Raspberry Pi B+. De l'autre côté, il possède un embout à 26 broches, que tu connectes au Pi-T-Cobbler. Pour te le procurer, tu peux te rendre sur `www.lextronic.fr` et l'acheter en même temps que le Pi-T-Cobbler.

Sur PC et Mac

Contrairement aux Raspberry Pi, les PC et les Mac ne sont pas dotés de broches intégrées. C'est pourquoi j'ai créé une petite plate-forme élaborée à partir d'un circuit Arduino que tu pourras utiliser sur tes machines. Pour effectuer les exercices de ce chapitre, tu peux utiliser n'importe quel circuit Arduino, à condition d'installer mon logiciel open source spécial qui permet de faire fonctionner la plate-forme comme s'il s'agissait d'un Raspberry Pi avec ses broches d'entrée et de sortie. Pour en savoir plus sur Arduino, rends-toi sur le site : `http://arduino.cc`. Dans ce chapitre, tu utiliseras un circuit Arduino Pro Micro. Tu auras également besoin d'un câble USB pour connecter le dispositif à ton ordinateur.

Copyright © 2016 Eyrolles.

Pour acheter le circuit Arduino Pro Micro, connecte-toi à : https://www.sparkfun.com/products/12587 ou sur une autre boutique en ligne spécialisée. Le numéro de référence du produit est : DEV-12587. Mais pour qu'il soit complètement opérationnel, tu devras effectuer quelques soudures et le programmer à l'aide du logiciel conçu pour le faire fonctionner comme une plate-forme d'entrée et de sortie. Pour gagner du temps et t'éviter le travail de soudure, tu peux directement acheter la plate-forme soudée, testée et configurée avec le logiciel en te rendant sur la page (en anglais) : http://skpang.co.uk/catalog/pro-micro-33v8mhz-with-headers-and-anyio-firmware-p-1327.html. Le numéro de référence du produit est : AR-PROMICRO-ANYIO. Tu n'auras plus qu'à brancher le circuit à ta plaque d'essais et à le connecter à ton PC ou à ton Mac à l'aide d'un câble USB, pour le faire fonctionner comme une plate-forme d'entrée et de sortie. La figure 5-3 t'indique tous les composants dont tu as besoin pour réaliser les exercices de ce chapitre sur un PC ou un Mac.

Si tu as un Mac, il doit au moins disposer de la version 10.6 d'OS X pour être compatible avec le circuit Arduino.



Pour réaliser les exercices de ce chapitre, tu disposes d'un dossier de fichiers sources appartenant à un logiciel. Tous ces fichiers se trouvent dans le kit de démarrage que tu as téléchargé sur le site Internet du livre. Un peu plus loin, nous fournissons également une liste de sites Internet sur lesquels toi et tes amis pourrez télécharger la dernière version des fichiers sources du logiciel, si jamais tu souhaites les utiliser pour d'autres projets et faire quelques expérimentations en dehors de Minecraft.



Une fois que tu te seras procuré tous les éléments dont tu as besoin, il sera temps de passer aux choses sérieuses et de commencer à construire ton tout premier circuit électronique !

Créer un prototype à l'aide d'une plaque d'essais

Lorsqu'un ingénieur électronique conçoit un circuit pour un nouveau produit, il crée tout d'abord ce qu'on appelle un **prototype** du circuit, qu'il consolide ensuite pour le reproduire par milliers.

Copyright © 2016 Eyrolles.



Un **prototype** est le modèle réduit d'un objet conçu pour réaliser des tests de fonctionnement. Il est bien plus simple, en effet, de résoudre les problèmes sur un seul objet plutôt que sur mille ou un million !



Sur Internet, tu trouveras de superbes photos des tout premiers prototypes du Raspberry Pi qu'Eben Upton a conçus en 2006, marquant la création de la Fondation Raspberry Pi. La plupart des circuits électroniques naissent sous la forme d'un prototype similaire et subissent de nombreuses modifications avant de ressembler au circuit que tu as sous les yeux. Pour voir les photos, rends-toi sur : `http://www.raspberrypi.org/raspberry-pi-2006-edition/`.

Sur la plupart des circuits électroniques, on soude les composants ensemble pour renforcer la connexion électrique et garantir un fonctionnement sur le long terme. Mais si les circuits soudés présentent l'avantage d'être plus solides, ils ne s'avèrent pas très pratiques lorsque tu souhaites te lancer dans des expérimentations impliquant la manipulation de plusieurs composants et la conception de plusieurs montages électroniques à tester, car tu passerais bien trop de temps à dessouder et ressouder tes composants. Pour éviter ce genre de problèmes, les ingénieurs utilisent généralement ce qu'on appelle une **plaque d'essais** sans soudure.



Une **plaque d'essais** est un outil réutilisable qui te permet de monter des circuits sans avoir recours à la soudure. Ces plaques contiennent plusieurs rangées de trous dans lesquels tu peux brancher des fils, des câbles ou des composants afin de créer des circuits. Les deux bandes de trous supérieure et inférieure de la plaque émettent un courant électrique. Normalement, elles sont encadrées d'une ligne rouge, désignant le côté positif de 3,3 volts (ce côté contient parfois l'indication « + » ou « VCC »), et d'une ligne noire ou bleue, désignant le côté négatif de 0 volt (parfois marqué d'un « - » ou de l'abréviation « GND »).

Sur la figure 5-4, tu peux voir deux exemples de plaques d'essais sans soudure, à gauche avec des composants branchés pour créer un circuit et à droite, l'intérieur de la plaque. Si tu regardes de plus près, tu peux voir les bandes de métal qui servent à réaliser les connexions. Il est très important de savoir comment une plaque d'essais fonctionne, car lorsque tu brancheras des composants dans les trous, tu sauras ainsi précisément comment les connexions interagiront dans tes circuits électriques.

Copyright © 2016 Eyrolles.

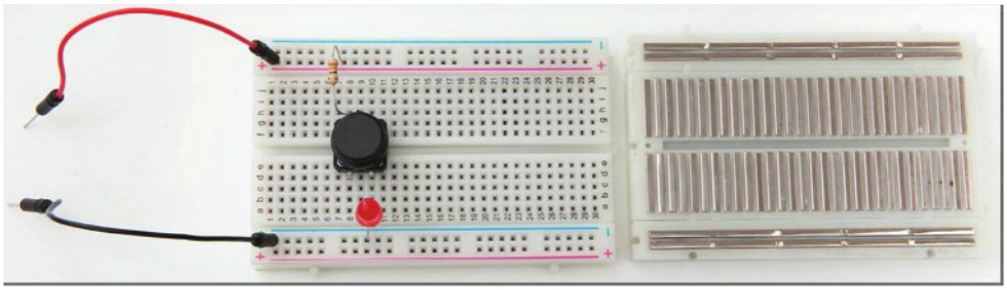


FIGURE 5-4 À gauche, une plaque d'essais ; à droite, l'intérieur d'une plaque d'essais avec ses connexions internes

Élaborer un circuit permettant d'allumer une LED

La figure 5-4 présente un circuit simple qui permet de tester des composants facilement. Pour faire fonctionner n'importe quel circuit, on doit le placer sous **tension**. Pour cela, tu dois connecter une pile 3 V aux bandes supérieure et inférieure de la plaque d'essais, appelées également *power-rails* (en anglais). Ce sont des barrettes longues qui assurent une connexion sur toute la longueur de la plaque et qui sont bien pratiques, car elles permettent d'apporter du courant à plusieurs composants à la fois. L'élément rouge que tu peux voir sur l'image est une **diode électroluminescente** (**LED**) ; une petite lumière que tu apprendras à connaître d'ici peu. Le composant aux rayures de couleur est une **résistance** et le dispositif noir situé en dessous est un bouton-poussoir. Si tu suis les câbles du circuit depuis la batterie jusqu'à la fin, tu retomberas sur la batterie. Cela veut dire qu'il s'agit d'un circuit électrique complet. Et lorsque tu appuies sur le bouton-poussoir, cela libère du **courant** électrique qui active la LED.



La **tension**, ou voltage, est la différence d'énergie électrique qui existe entre deux points d'un même circuit. Elle est équivalente à la pression qu'exerce l'eau dans les tuyaux de canalisation ; c'est cette même pression qui pousse le courant à s'écouler dans un circuit. Elle se mesure en volts (notés V).



Une **diode électroluminescente** (**LED**) s'allume lorsqu'un courant électrique la traverse. Une LED n'autorise le courant à passer que dans un seul sens et ne s'allume que si le courant passe dans le bon sens. Il existe des LED de toutes les couleurs. Elles sont dotées d'une patte courte (la cathode) et d'une patte longue (l'anode), qui t'aident à placer la diode dans le bon sens sur un circuit afin que le courant puisse la traverser.

Copyright © 2016 Eyrolles.



Une **résistance** est un composant électrique qui permet de résister au courant dans un circuit. Par exemple, si le courant est trop puissant, cela peut endommager ta LED. Pour éviter cela, tu vas donc placer une résistance d'une valeur suffisamment élevée sur ton circuit pour réguler le courant qui passera à travers ta diode. La valeur d'une résistance se calcule en ohms, qu'on signale à l'aide du symbole oméga, Ω. Pour protéger ta diode, tu dois choisir la bonne valeur de résistance. Celle-ci est indiquée sur ton composant par des rayures de couleur répondant à un code précis, qui se lit de gauche à droite. Pour en savoir plus sur le code couleur des résistances, tu peux consulter la page : `https://fr.wikipedia.org/wiki/CEI_60757`.

PERCER LES SECRETS DU CODE

Sur le circuit de la figure 5-4, une résistance a été utilisée. Une LED est un composant délicat qui ne peut recevoir qu'un courant de faible intensité, ne dépassant généralement pas les 20 mA (soit 0,02 A, ce qui est plutôt faible). Si tu connectes une LED directement à une pile, le courant qui circulera dans le circuit sera bien trop fort et endommagera ta LED. Pour éviter cela, on ajoute une résistance entre la pile et la LED pour limiter le courant qui passe dans le circuit. La valeur de la résistance est signalée par les rayures colorées qui sont peintes sur son corps.

Si tu utilises des piles plus puissantes, tu dois alors placer une résistance d'une valeur plus élevée sur ton circuit pour réduire l'intensité du courant. Pour ce projet, j'ai calculé les valeurs de résistance à ta place, mais il existe une multitude de sites Internet qui t'aident à calculer la valeur des résistances en fonction de la puissance des piles, par exemple : `www.ohmslawcalculator.com/led_resistor_calculator.php`.



Le **courant** est le taux d'énergie électrique qui circule au-delà d'un point dans un circuit. C'est l'équivalent du débit d'eau dans un tuyau. Il se mesure en ampères (notés A). Pour les courants plus faibles, on utilisera le milliampère (mA) comme unité de mesure. Il existe plusieurs théories intéressantes au sujet du sens de circulation du courant dans un circuit. Pour les connaître, rends-toi sur : `www.allaboutcircuits.com/vol_1/chpt_1/7.html` (en anglais).

Copyright © 2016 Eyrolles.

DÉFI

Essaye de construire le circuit de la figure 5-4 avec tes composants électroniques puis appuie sur le bouton pour allumer la diode. Tu auras besoin d'un petit coupleur de piles AA ou AAA. Fais attention à bien placer la LED dans le bon sens sur le circuit. Si elle ne s'allume pas, vérifie tes branchements et repositionne-la.



Connecter des composants électroniques à ton ordinateur

Dans ce chapitre, tu vas connecter une LED à ton ordinateur et l'activer à l'aide d'un programme Python. Or, pour prendre le contrôle de composants électroniques depuis ton ordinateur, tu dois d'abord trouver un moyen de connecter ton ordinateur aux circuits qui te permettront de les activer. Pour ce faire, tu as besoin de **broches GPIO** (**General Purpose Input/Output**).

Les **broches GPIO** permettent de recevoir ou d'émettre des signaux vers ou depuis le processeur central d'un système informatique. Elles ont été conçues pour répondre à un usage général et les concepteurs de circuits peuvent leur assigner une fonction précise, telle que le contrôle de circuits externes.



Certains ordinateurs, comme les Raspberry Pi, disposent d'un port GPIO intégré. Une broche GPIO peut être utilisée en mode entrée pour détecter la tension d'un dispositif externe ou en mode sortie pour réguler sa tension. La tension de ton Raspberry Pi ou de ton circuit Arduino s'élève à 3,3 volts, ce qui est plutôt bas, mais suffisant pour allumer une LED ou détecter un bouton-poussoir. À titre de comparaison, l'électricité domestique en France est limitée à 240 volts, ce qui est très élevé (et dangereux). Quant aux pylônes électriques disséminés un peu partout dans le pays, ils transportent un courant de pas moins de 230 000 volts ! Rien à voir, donc, avec le courant qui circule dans un circuit informatique.

Un ordinateur est une machine numérique, c'est-à-dire qu'il repose sur un système de numéros. Aujourd'hui, tous nos ordinateurs sont numériques et utilisent les chiffres 0 et 1. Lorsque tu utilises des broches GPIO sur un ordinateur, le 0 indique généralement une tension nulle, soit 0 volt, tandis que le 1 indique la présence de tension. Dans notre cas, cette dernière s'élève à 3,3 volts. Lorsque tu souhaites détecter la tension d'un dispositif externe à l'aide d'un port GPIO, l'ordinateur ne peut indiquer cette tension qu'avec les chiffres 0 ou 1. Une tension proche de 0 volt sera donc signalée par un 0,

Copyright © 2016 Eyrolles.

tandis qu'une tension proche de 3,3 volts sera signalée par un 1. Toute tension qui se situe entre ces deux valeurs sera signalée par un 0 ou par un 1 dans ton ordinateur, bien qu'elle se situe dans une zone grise.

Paramétrer un PC ou un Mac pour contrôler des circuits électroniques

Si tu as un Raspberry Pi, tu peux ignorer cette section et te rendre directement à la suivante, intitulée « Contrôler une LED ». En effet, un Raspberry Pi est déjà doté de broches GPIO intégrées, ce qui en fait un excellent outil pour contrôler des circuits électroniques.

Les PC et les Mac, quant à eux, ne possèdent pas de broches GPIO intégrées. Tu dois donc les ajouter toi-même à l'aide d'un circuit externe. Il existe une multitude de manières de le faire, mais pour ce qui nous concerne, nous avons choisi d'utiliser un circuit Arduino préconfiguré. Tu n'apprendras rien sur Arduino dans ce livre, mais si tu souhaites en savoir plus sur le petit ordinateur, tu peux consulter l'annexe A, qui te renverra vers quelques sites d'intérêt.



Dans les instructions qui vont suivre, je considère que tu as acquis un circuit présoudé et préconfiguré, comme indiqué au début du chapitre. Si tu as choisi d'utiliser une plaque vierge pour la souder et la configurer toi-même, tu devras lire les instructions de ce guide pratique en anglais : `https://www.sparkfun.com/products/12587`. Puis tu devras installer le logiciel GPIO open source que j'ai spécialement conçu pour cela. Ce petit logiciel, couplé au module Python *anyio* que j'ai inclus dans le kit de démarrage, te permettra de faire fonctionner tes broches GPIO exactement comme sur un ordinateur Raspberry Pi.

Pour exécuter les programmes de ce chapitre, tu devras alors penser à les modifier légèrement afin qu'ils prennent en compte ton cas de figure. Si tu dois le programmer toi-même, le logiciel à installer dans le circuit Arduino se trouve dans le dossier `anyio/arduino/firmware` de ton kit de démarrage. Par ailleurs, si tu souhaites te procurer la dernière version du package anyio pour pouvoir l'utiliser avec tes amis dans le cadre d'autres projets, tu peux toujours te rendre sur ma page GitHub : `https://github.com/whaleygeek/anyio`.

Configurer les pilotes

Lorsque tu connectes un périphérique à ton ordinateur, celui-ci doit disposer d'un logiciel spécial qui lui permet de communiquer avec le périphérique en question. Ce logiciel est en fait un pilote informatique. Il t'est peut-être déjà arrivé d'utiliser un pilote

Copyright © 2016 Eyrolles.

informatique sur ton ordinateur, par exemple si tu as installé une imprimante ou un autre périphérique. C'est exactement la même opération que tu vas réaliser ici.

1. Branche le câble USB sur le circuit Arduino et l'autre extrémité sur ton ordinateur.
2. Ensuite, suis les instructions qui s'affichent à l'écran (comme elles varient d'une version à une autre du système d'exploitation, je ne détaillerai malheureusement pas les étapes ici) et procède à l'installation des pilotes pour utiliser le Pro Micro sur ton ordinateur.

Sur Mac

Tu vas voir s'afficher un message t'indiquant qu'un nouveau clavier a été ajouté. Clique simplement sur la croix rouge pour le faire disparaître. Et voilà ! L'installation de ton circuit est terminée.

Sur PC

Windows va te demander de télécharger un pilote. Tu n'as pas besoin de le faire, car Windows dispose déjà de tous les pilotes nécessaires. En revanche, comme ton système d'exploitation ne reconnaît pas toujours les périphériques inconnus, j'ai fourni un fichier dans le kit de démarrage pour aider Windows à détecter le périphérique. Ce fichier s'appelle `ProMicro.inf` et il se trouve dans le dossier `anyio/arduino/firmware` du kit de démarrage, à l'intérieur du dossier `MyAdventures`. Recherche le fichier `ProMicro.inf` en question et clique sur `OK`. Windows devrait maintenant être en mesure de reconnaître le périphérique en tant que port USB.



Si malgré tout ton ordinateur ne reconnaît pas le périphérique, consulte le guide pratique du site anglais SparkFun. Plutôt bien détaillé, il est mis à jour à chaque fois qu'une nouvelle version du système d'exploitation est déployée : `https://www.sparkfun.com/products/12587`.

Obtenir le numéro du port série

Le circuit Arduino que tu viens de connecter à ton ordinateur est reconnu comme un port série. Sans t'attarder sur ce que cela signifie, tu dois t'assurer que le kit anyio choisisse le bon port avant de l'utiliser comme extension GPIO.

Pour t'aider, j'ai rédigé un petit programme d'aide que tu trouveras également dans le kit anyio. La figure 5-5 te montre à quoi il ressemble.

Suis donc ces indications pour choisir le bon port.

Copyright © 2016 Eyrolles.

1. Ouvre IDLE et clique sur `File>Open` pour rechercher un programme Python.
2. Dans le dossier `MyAdventures`, ouvre le fichier `findPort.py`, qui s'affichera alors dans l'éditeur Python.
3. Exécute ce programme en sélectionnant `Run>Run Module` dans le menu. Tu verras alors des instructions s'afficher à l'écran.
4. Débranche le câble USB du circuit Arduino et appuie sur `Entrée` pour commencer l'analyse. Cette étape consiste à retirer le port USB de ton ordinateur pour que le programme puisse analyser tous les ports disponibles sur ta machine et en établir une liste. Le port que tu vas ajouter n'y figure pas.
5. Tu vas ensuite devoir brancher à nouveau le périphérique pour que le programme puisse recommencer l'analyse et détecter tous les nouveaux périphériques qui se sont ajoutés à la liste. Branche à nouveau ton câble, attends quelques secondes et appuie sur `Entrée`.
6. Si l'opération réussit, tu devrais obtenir un message indiquant le nom ou le numéro du port qui a été trouvé et le programme devrait te demander si tu souhaites le mémoriser. Saisis la lettre *y* (pour « yes », c'est-à-dire « oui » en français) et appuie sur `Entrée`. Comme tu peux le voir sur la figure 5-5, le programme `findPort.py` va créer un fichier nommé `findport.cache` qui mémorise le numéro de port de ton circuit Arduino. Plus tard, lorsque tu utiliseras les fonctions GPIO, ce fichier te permettra de retrouver le port du circuit Arduino sans difficulté.

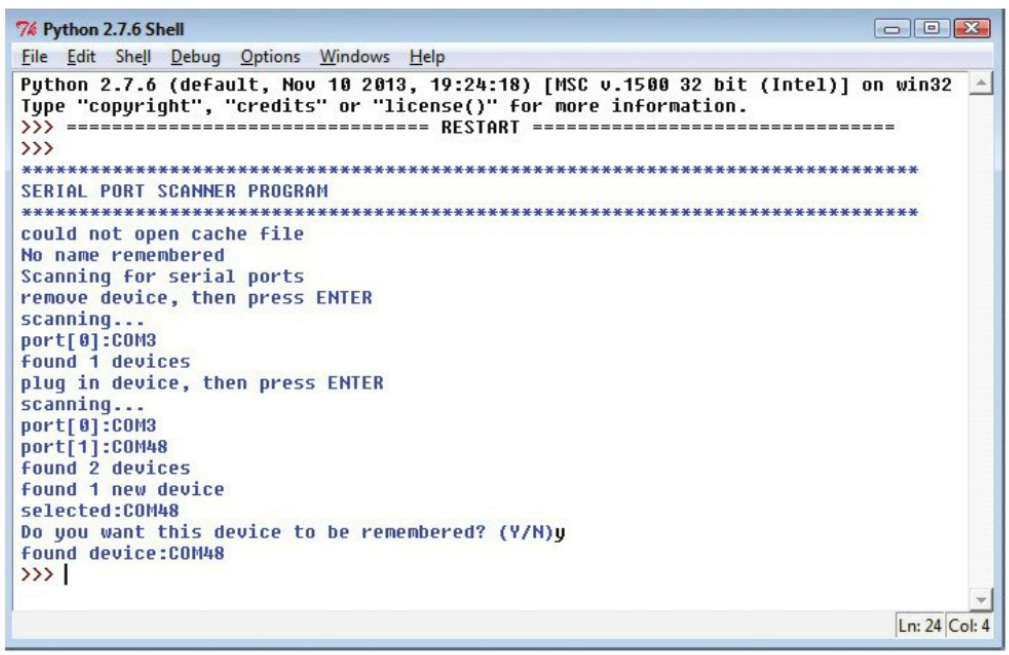


FIGURE 5-5 Le résultat obtenu suite à l'exécution du programme findPort.py

Copyright © 2016 Eyrolles.

Si tu branches d'autres périphériques USB sur ton ordinateur et que tu redémarres ta machine ou rebranches ton circuit Arduino, il se peut que le numéro de port change et fasse échouer ton programme. Le cas échéant, il te suffira d'exécuter à nouveau le programme `findPort.py` ou de supprimer le fichier `findport.cache` et d'exécuter le programme pour effectuer une nouvelle analyse.



Il est assez difficile de rédiger un kit compatible avec toutes les versions d'un système d'exploitation, d'autant plus que les éditeurs les renouvellent sans cesse. Si tu rencontres des problèmes lors de l'installation de ton périphérique, fais un tour sur le site Internet du livre (`www.editions-eyrolles.com/dl/14292`) pour voir s'il existe une mise à jour du kit. Sinon, tu trouveras toujours les dernières actualités et mises à jour sur ma page GitHub à cette adresse : `https://github.com/whaleygeek/anyio`.



Contrôler une LED

Dans le chapitre 1, tu as rédigé un programme intitulé `bonjourMondeDeMinecraft.py` qui permettait d'afficher un message dans le chat de Minecraft. Avec des circuits électroniques, tu peux tout à fait rédiger un programme équivalent au « Bonjour le monde de minecraft » qui, au lieu d'afficher un message à l'écran, fait clignoter une diode. Ainsi, une fois que tu auras appris à vérifier que tout va bien en paramétrant ton ordinateur pour qu'il fasse clignoter une LED, tu seras en mesure de progresser et de créer des projets encore plus amusants.

La LED à laquelle nous avons affaire ici est un peu spéciale, car tu vas la relier au monde de Minecraft. Dans cette section, tu vas développer une idée que tu as déjà étudiée dans le chapitre 2, lorsque tu as rédigé le programme du paillasson magique. Cette fois-ci, ton paillasson sera connecté au circuit et la LED se mettra à clignoter dès que ton joueur marchera dessus. Mais si tu peux connecter ce circuit à une LED, alors il est également possible de le connecter à n'importe quel autre type d'objet. Imagine-toi donc tout ce que tu pourrais programmer et provoquer dans le monde réel en plaçant ton joueur sur le paillasson. Tu pourrais tout à fait allumer la lumière de ta chambre ou ouvrir les rideaux de ta fenêtre, ou que sais-je encore !

Si tu réalises ce projet sur un PC ou un Mac, tu vas utiliser le kit anyio. Tu le trouveras dans le kit de démarrage, mais tu peux aussi le télécharger en te rendant sur les sites mentionnés au début de ce chapitre.



Copyright © 2016 Eyrolles.

Allumer une LED avec un ordinateur

Tu vas tout d'abord préparer ton circuit sur la plaque d'essais. Les figures 5-6 et 5-7 montrent comment la LED est raccordée à ton ordinateur.

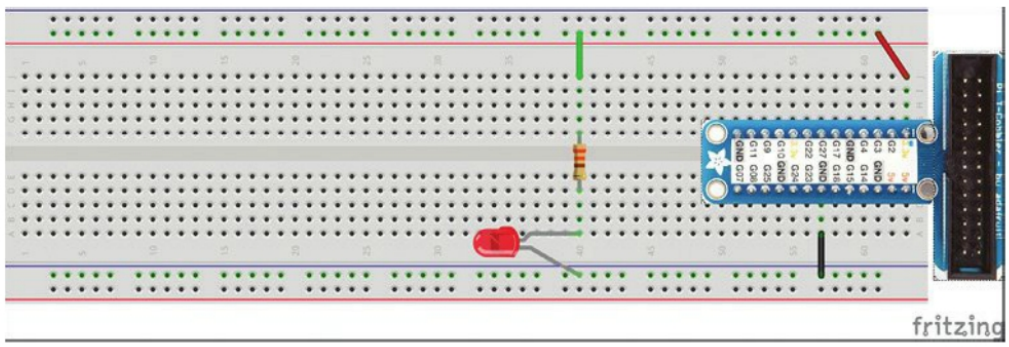


FIGURE 5-6 Une LED et une résistance raccordées à un Raspberry Pi

1. Prends la patte la plus courte de ta LED. Il s'agit de son pôle négatif, appelé cathode. Connecte-la à la barrette à tension nulle (0 V), située en bas de ta plaque d'essais.
2. L'autre patte de la LED, la plus longue, correspond au pôle positif, l'anode. Place l'une des pattes de ta résistance sur la même barrette que l'anode de la LED (dans la partie moyenne inférieure de la plaque). Ensuite, introduis l'autre patte de la résistance dans un trou de la bande moyenne supérieure de ta plaque d'essais. Pour rappel, la résistance est nécessaire, car elle régule le courant qui traverse la LED.
3. Pour tester la LED, place un fil électrique sur la patte de la résistance connectée à la partie moyenne supérieure de ta plaque d'essais et connecte l'autre extrémité du fil à la bande supérieure de la plaque d'essais. Tu le déplaceras plus tard.

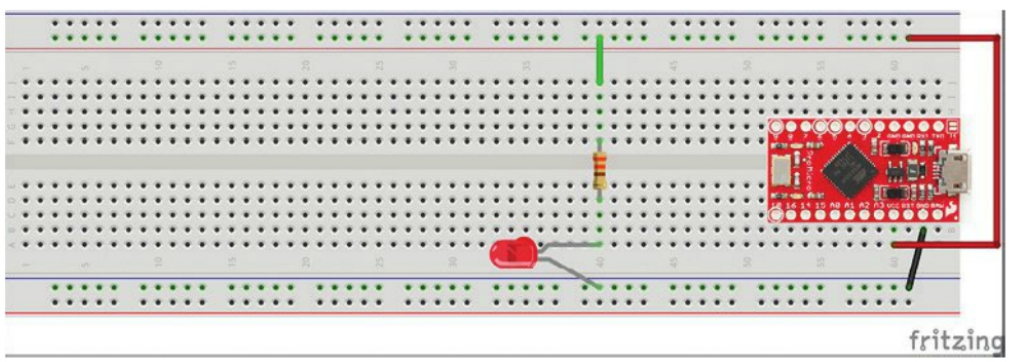


FIGURE 5-7 Une LED et une résistance raccordées à un circuit Arduino

Les instructions qui suivent sont différentes si tu utilises un Raspberry Pi ou un circuit Arduino sur PC ou Mac. Passe donc directement à la section qui te concerne.

Copyright © 2016 Eyrolles.

Sur Raspberry Pi

1. Si tu disposes d'une étiquette autocollante pour le Pi-T-Cobbler, colle-la comme sur la figure 5-8. Cela t'aidera à reconnaître les broches GPIO plus facilement. Connecte le Pi-T-Cobbler à la plaque d'essais de façon à ce que la prise noire se retrouve sur la droite, à l'extérieur de la plaque. Introduis les broches du Pi-T-Cobbler dans les trous de la partie droite de la plaque, jusqu'au bord. La moitié des broches doit être branchée sur la partie moyenne supérieure de la plaque et l'autre moitié sur la partie moyenne inférieure. Appuie assez fort pour t'assurer que les broches soient bien enfoncées dans les trous.
2. Connecte le câble plat au Pi-T-Cobbler et au Raspberry Pi. Le connecteur est marqué d'une encoche et la prise du Pi-T-Cobbler contient une fente, ce qui signifie que tu ne peux brancher le câble que dans un sens. En branchant le câble plat au Raspberry Pi, fais attention de placer la cosse en plastique vers le haut par rapport au circuit, sinon les broches seront à l'envers et cela ne fonctionnera pas ! Tu peux voir tous les branchements sur la figure 5-8.

La dernière chose qu'il te reste à faire est de vérifier que ton ordinateur est capable d'allumer la LED. En effet, avant de la faire clignoter à l'aide du programme Python, tu dois d'abord t'assurer que la LED peut fonctionner avec le courant de ton ordinateur.



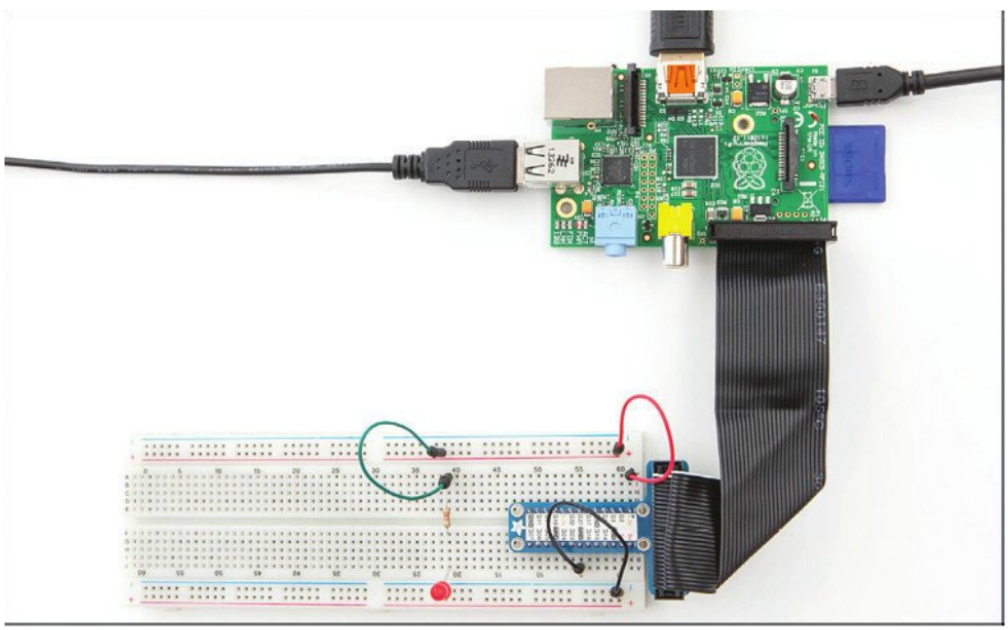


FIGURE 5-8 Le Pi-T-Cobbler est connecté au Raspberry Pi.

Copyright © 2016 Eyrolles.

3. Branche un fil électrique sur la bande positive de la plaque (dans la partie supérieure) et relie-le à la broche 3,3 V du Pi-T-Cobbler.
4. Branche ensuite un fil sur la bande négative de la plaque (dans la partie inférieure) et relie-le à la broche 0 V du Pi-T-Cobbler.

Si ton Raspberry Pi est allumé, la LED devrait s'allumer, car elle est désormais connectée à l'alimentation de ton ordinateur.

Sur Arduino

1. Aligne ton circuit Arduino sur la plaque d'essais de sorte que la prise USB se retrouve sur le côté droit de la plaque, à l'extérieur. Il se peut que tu doives appuyer assez fort pour enfoncer toutes les broches dans la plaque d'essais. Une simple pression sur la totalité de la surface du circuit suffira et t'évitera d'endommager les broches. Tu peux voir tous les branchements sur la figure 5-9.
2. Connecte ensuite le circuit Arduino à ton ordinateur via le port USB.



La dernière chose qu'il te reste à faire est de vérifier que ton ordinateur est capable d'allumer la LED. En effet, avant de la faire clignoter à l'aide du programme Python, tu dois d'abord t'assurer que la LED peut fonctionner avec le courant de ton ordinateur.

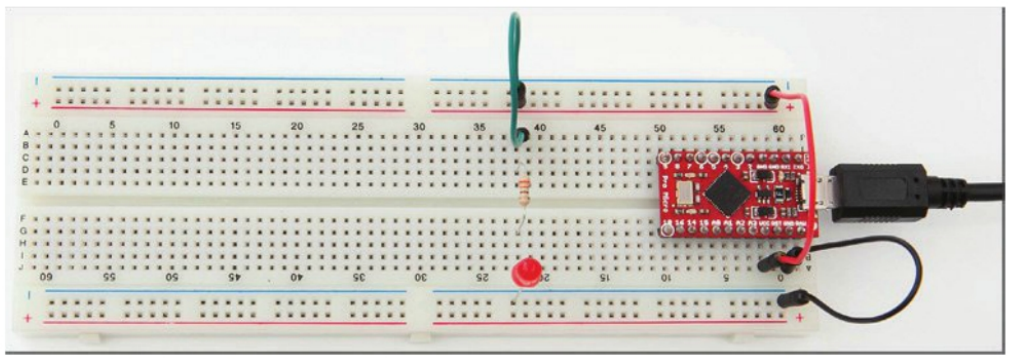


FIGURE 5-9 Le circuit Arduino est connecté à la plaque d'essais.

3. Branche un fil électrique sur la bande positive de la plaque (dans la partie supérieure) puis relie-le à la broche VCC du circuit Arduino (VCC est un terme technique qui désigne le pôle positif ; il équivaut à 3,3 volts sur ton circuit Arduino).
4. Branche ensuite un fil sur la bande négative de la plaque (dans la partie inférieure) puis relie-le à la broche GND du circuit Arduino (GND est un terme technique qui désigne le pôle négatif d'un circuit, soit un courant de 0 volt).

Copyright © 2016 Eyrolles.

Si ton circuit Arduino est connecté, la LED devrait s'allumer, car elle est désormais connectée à l'alimentation de ton ordinateur.



Si ta LED ne s'allume pas, vérifie que tu l'as placée dans le bon sens. Inverse son branchement pour voir si elle fonctionne. Tu peux également retourner ta plaque d'essais pour voir comment elle se compose et retracer les lignes de ton circuit depuis la broche à 3,3 volts jusqu'à celle de 0 volt, en t'arrêtant à chaque composant pour voir s'ils sont bien branchés. Généralement, si une LED ne s'allume pas, c'est qu'elle est placée dans le mauvais sens ou que ses pattes ont été enfoncées dans les mauvais trous sur la plaque d'essais, ne formant pas un circuit complet.

Faire clignoter la LED

Maintenant que ta LED est bien raccordée et qu'elle s'allume comme prévu, il ne te reste plus qu'à rédiger un programme Python pour la faire clignoter.

1. Démarre IDLE et crée un nouveau programme en cliquant sur `File>New File`, puis nomme-le `testLED.py`.
2. Importe le module de temps que tu utiliseras pour introduire un délai entre chaque clignotement.

```
import time
```



Dans tous les scripts de ce chapitre, j'utilise les instructions `import RPi.GPIO as GPIO` ou `import anyio.GPIO as GPIO`, exactement comme s'il s'agissait de l'instruction `import mcpi.minecraft as minecraft`, que tu connais déjà. Le mot-clé `as` indique à Python de renommer le module et de lui attribuer le nom qui se situe à sa droite. Cela s'avère particulièrement utile lorsque les noms des modules sont très longs. Mais cela signifie aussi que tu n'auras qu'à modifier une ou deux lignes des instructions et programmes de ce chapitre pour communiquer avec ton circuit Arduino ou ton Raspberry Pi, ce qui est plutôt cool. Merci Python !

3. Importe le module qui te permet de prendre le contrôle du port GPIO et attribue une constante au numéro du port connecté à ta LED, comme ceci :

Sur Raspberry Pi

```
import RPi.GPIO as GPIO
LED = 15
```

Copyright © 2016 Eyrolles.

Sur Arduino

```
import anyio.GPIO as GPIO
LED = 15
```

4. Détermine le mode de numérotation de la broche GPIO. Dans le code qui va suivre, tu vas voir que nous utilisons les lettres BCM. C'est l'abréviation du mot anglais « Broadcom », qui est le nom de la puce du processeur de ton Raspberry Pi. Ces lettres signifient que tu indiques à Raspberry Pi d'utiliser les numéros GPIO au lieu des numéros de broche qui figurent sur ton circuit. Définis maintenant ta broche GPIO en mode sortie pour que ton programme puisse modifier le courant d'alimentation de ta LED :

```
GPIO.setmode(GPIO.BCM)
GPIO.setup(LED, GPIO.OUT)
```

5. Maintenant, passe à la rédaction de la fonction qui te permettra de provoquer un clignotement. On passe un paramètre `t` à la fonction pour pouvoir modifier la vitesse de clignotement de la LED. Quant à la fonction `output()`, elle servira à modifier la tension du GPIO. Lorsque tu utiliseras `True` la tension s'élèvera à 3,3 volts et lorsque tu utiliseras `False` elle tombera à 0 volt. Cela aura pour effet d'allumer et d'éteindre la LED :

```
def clignotement(t)
    GPIO.output(LED, True)
    time.sleep(t)
    GPIO.output(LED, False)
             time.sleep(t)
```

6. Rédige une boucle pour faire clignoter la LED, puis appelle la fonction `GPIO.cleanup()` à la fin du programme, afin d'effectuer un nettoyage et de remettre les broches GPIO dans leur état initial. Enfin, pour en savoir plus sur le bloc `try/finally`, je te conseille de lire l'encadré « Percer les secrets du code » plus loin.

```
try:
    while True:
        clignotement(0.5)
finally :
    GPIO.cleanup()
```

7. Pour que la LED puisse clignoter, elle doit être connectée à la broche GPIO sur ton ordinateur. Suis donc ces quelques étapes.

   Sur Raspberry Pi

   Déplace le fil qui relie la résistance à la barrette de 3,3 volts. Débranche l'extrémité connectée à la barrette et branche-la à la broche 15 du port GPIO (sur l'étiquette Pi-T-Cobbler, elle est nommée G15).

Copyright © 2016 Eyrolles.

Sur Arduino

Déplace le fil qui relie la résistance à la barrette de 3,3 volts. Débranche l'extrémité connectée à la barrette et branche-la à la broche 15 du port GPIO sur le circuit Arduino.

Le circuit Arduino est lui-même doté de LED, la jaune permet de savoir lorsque le circuit reçoit des commandes par USB. Tu la verras clignoter deux fois plus vite que celle de ta plaque d'essais, car à chaque fois que cette dernière reçoit l'instruction de s'allumer ou de s'éteindre, la LED jaune de ton circuit Arduino se met à clignoter. Ne sois pas surpris ; cela veut dire que tout fonctionne correctement !



Exécuter un programme GPIO

Il est enfin temps que tu exécutes ton programme et que tu vérifies si la LED fonctionne !

Sur PC ou Mac

Exécute le programme comme d'habitude depuis IDLE. Le circuit Arduino est branché via un port USB et ton PC ou ton Mac peut désormais s'y connecter en continu.

Sur Raspberry Pi

Exécuter le programme sur un Raspberry Pi est un peu plus compliqué. En effet, bien que tu continues d'utiliser IDLE pour ouvrir et modifier tes programmes, ceux qui fonctionnent avec le GPIO devront s'exécuter différemment. Pour en savoir plus sur les raisons de ces manœuvres supplémentaires, consulte l'encadré « Percer les secrets du code ».

1. Ouvre le programme LXTerminal depuis le bureau ou le menu. Cela va ouvrir une fenêtre d'invite de commandes, que tu utiliseras pour saisir des instructions sur ton Raspberry Pi.
2. Accède au dossier `MyAdventures` en saisissant la commande `cd`.

```
cd MyAdventures
```

3. Exécute ton programme Python en utilisant la commande `sudo` (pour connaître la signification de ce terme, lis l'encadré « Percer les secrets du code ») :

```
sudo python testLED.py
```

Ta LED a-t-elle clignoté ? Cela peut te sembler anodin, mais tu viens d'effectuer quelque chose d'incroyable. Tu as non seulement réussi à prendre le contrôle d'un circuit électronique, mais ce faisant tu as également dépassé les limites de ton ordinateur !

Copyright © 2016 Eyrolles.



Il se peut que le message d'erreur suivant s'affiche : « Channel already in use » (ce qui signifie littéralement « Canal déjà occupé »). Pas de panique, c'est normal. Tu vas bientôt apprendre à utiliser la fonction `GPIO.cleanup()` pour éviter ce message d'erreur.

La prochaine étape consistera à relier ce programme au monde de Minecraft.



Souviens-toi que lorsque tu fais clignoter une LED grâce à un programme Python, tu utilises un programme semblable à celui de « Bonjour le monde de Minecraft », que tu connais déjà. Lorsque tu raccordes un circuit électronique à un ordinateur, il est important de vérifier qu'il fonctionne correctement avant de t'aventurer plus loin. Si ta LED ne clignote pas, vérifie bien tous tes branchements et assure-toi que le fil qui relie ta LED à la broche GPIO est inséré dans le bon trou sur la plaque d'essais. Plus tôt, tu as vérifié que ta LED fonctionnait correctement en la raccordant à l'alimentation de ton ordinateur et en lui envoyant un courant de 3,3 volts. Si elle ne fonctionne pas ici, tu peux en déduire que ton programme Python contient une erreur ou que le fil qui relie la LED au port GPIO est mal branché.

PERCER LES SECRETS DU CODE

Dans les paragraphes précédents, tu as employé deux tours de magie qu'il serait bon d'éclaircir un peu ici.

Tout d'abord, à quoi sert l'instruction `try/finally` avec `GPIO.cleanup()` ?

Afin d'éviter les messages d'erreur à chaque fois que tu exécutes ton programme, n'oublie pas d'utiliser la fonction `GPIO.cleanup()` à la fin de ton script. Elle permet de redéfinir tes broches GPIO en mode entrée et de désactiver ton circuit. En d'autres termes, cela signifie que tes broches ne pourront plus prendre le contrôle du circuit. Si, par inadvertance, tu te trompes de broche lors des branchements, tu peux provoquer un court-circuit en allumant ton dispositif, ce qui pourrait endommager ton circuit. Pour éviter cela, nous te conseillons d'appeler la fonction `GPIO.cleanup()` à la fin de ton programme. Ainsi, les éventuels courts-circuits n'endommageront pas ton périphérique lorsque tu effectueras des modifications.

Mais voilà qu'un problème se pose : comme tu as utilisé la boucle `while True:` dans ton script, tu as créé une boucle de jeu infinie. Or, la seule façon d'arrêter un programme est d'appuyer sur `Ctrl+F6` ou d'aller dans le menu de

Copyright © 2016 Eyrolles.

l'interpréteur de commandes IDLE et de cliquer sur `Restart Shell`, car quand tu utilises cette fonctionnalité, le programme s'arrête immédiatement.

Pas d'inquiétude, il existe une solution. C'est ici qu'entre en jeu le bloc `try/finally`, qui permet de lever ce que l'on appelle des exceptions. Je ne m'attarderai pas sur les exceptions dans ce livre, mais sache que tu ne peux pas t'en passer lorsque tu utilises un port GPIO. En gros, lorsque tu appuies sur `Ctrl+F6` ou que tu arrêtes le programme depuis le menu IDLE, Python exécute l'exception `KeyboardInterrupt`, qui provoque l'arrêt du programme. Pour permettre à Python d'anticiper un arrêt, les programmeurs ont recours à une petite astuce qui consiste à utiliser le bloc `try/finally`. Après le mot-clé `finally`, tu peux saisir plusieurs instructions que tu souhaites exécuter avant l'arrêt du programme. Ainsi, lorsque tu appuies sur `Ctrl+F6` ou que tu cliques sur `Restart Shell` dans le menu de l'interpréteur, le programme passe directement aux instructions qui se trouvent dans le bloc `finally:`, les exécute, puis s'arrête. Cela te permettra ainsi de toujours appeler la fonction `GPIO.cleanup()` pour remettre les broches GPIO en position de sécurité, c'est-à-dire dans leur état initial, à la fin de ton programme.

Je n'aborderai pas le sujet des exceptions plus en détail dans ce livre, mais si tu souhaites en savoir plus, tu peux toujours consulter la page : `https://wiki.python.org/moin/HandlingExceptions`.

Deuxièmement, à quoi sert le mot-clé `sudo` sur Raspberry Pi et pourquoi faut-il exécuter les programmes GPIO depuis LXTerminal ?

Tous les composants des ordinateurs Raspberry Pi sont protégés. Un utilisateur `pi` normal ne peut donc pas y accéder en tant que tel ; il doit avoir des privilèges de super-utilisateur. Le terme `sudo` signifie « substitute user and do » (en français, « se substituer à l'utilisateur et prendre le contrôle ») et permet de passer du statut d'utilisateur `pi` à celui d'utilisateur `root` (le super-utilisateur), puis d'exécuter le programme `testLED.py`.

Ce dernier peut ainsi accéder au périphérique GPIO, même s'il est protégé. Si tu n'exécutes pas tes programmes GPIO de cette manière sur Raspberry Pi, tu recevras un message d'erreur.

Lorsque j'ai écrit ce livre, j'ai essayé de trouver une explication claire au terme `sudo`. Mais étonnamment, à chaque fois que j'effectuais une recherche sur Google à ce sujet, je tombais sur une page de mon blog, qui est apparemment très appréciée ! Tu pourras donc lire tout ce qu'il faut savoir sur ce mot-clé et son utilisation ici : `http://blog.whaleygeek.co.uk/sudo-on-raspberry-pi-why-and-why-you-need-to-ask-kids-too`.

Copyright © 2016 Eyrolles.

Rédiger le programme de paillasson magique avec une LED

La dernière étape qu'il te reste à franchir est de relier ta LED au jeu Minecraft. Heureusement, cette étape est facile à réaliser, car elle s'appuie sur des connaissances que tu maîtrises déjà.

1. Enregistre ton programme `testLED.py` sous un nom différent en cliquant sur `File>Save As` dans le menu de l'éditeur et nomme-le `bienvenueLED.py`.

2. Indique les modules Minecraft que tu dois importer au début du fichier et connecte-toi au jeu :

```
import mcpi.minecraft as minecraft
import mcpi.block as block
mc = minecraft.Minecraft.create()
```

3. En dessous, pose des constantes indiquant les coordonnées de ton paillasson de bienvenue. Tu dois donc décider où tu souhaites poser ton paillasson dans le monde et choisir les coordonnées en conséquence. Sur Raspberry Pi, celles-ci s'affichent en haut à gauche de l'écran lorsque tu joues à Minecraft. Sur PC et Mac, tu dois appuyer sur `F3` pour les faire apparaître :

```
MAISON_X = 0
MAISON_Y = 0
MAISON_Z = 0
```

4. Juste après ces constantes, crée un bloc de laine au niveau de ton paillasson. Voici donc ton paillasson, exactement comme celui que tu as créé dans le chapitre 2 :

```
mc.setBlock(MAISON_X, MAISON_Y, MAISON_Z, block.WOOL.id, 15)
```

5. Modifie ta boucle en copiant les instructions ci-après. Fais particulièrement attention à l'indentation. Ce code permet de détecter en continu la position de ton joueur et utilise la technique du géorepérage pour savoir s'il se trouve sur le paillasson. Si c'est le cas, il active la LED qui clignote pour t'indiquer que tu es bien arrivé à la maison :

```
try:
    while True:
        pos = mc.player.getTilePos()
        if pos.x == MAISON_X and pos.z == MAISON_Z:
            clignotement(0.5)
finally :
    GPIO.cleanup()
```

Exécute ce programme en suivant les instructions indiquées dans la section « Exécuter un programme GPIO » et vérifie que la LED s'allume lorsque tu franchis le pas de ta porte.

Copyright © 2016 Eyrolles.

Fantastique ! Tu es arrivé chez toi et tu as maintenant tous les outils en main pour repousser les limites du monde virtuel de Minecraft et le relier à des objets du monde réel. Si tu es capable d'allumer une LED à l'aide d'un programme, tu seras également capable de contrôler d'autres dispositifs électroniques. Laisse-toi guider par ton imagination !

DÉFI

Procure-toi des LED de couleurs différentes et branche-les sur d'autres broches de ton port GPIO. Choisis-en une bleue et ajoute-la à ton programme Python en appelant la fonction `getBlock()` pour la faire clignoter lorsque ton joueur touche l'eau. Ajoutes-en ensuite une verte et fais-la clignoter lorsque tu touches l'herbe. En faisant preuve de créativité, combien d'événements peux-tu encore signaler avec des LED dans Minecraft ?



Utiliser un afficheur 7 segments

Dans le prochain chapitre, tu auras recours à un type d'afficheur numérique dont l'utilisation est très répandue. Il te servira notamment à afficher des chiffres et d'autres symboles dès qu'un événement surviendra dans Minecraft. Tu pourras ainsi l'utiliser dans le dernier jeu que nous t'apprendrons à créer dans ce livre. Mais avant de te lancer dans l'aventure et de l'utiliser avec Minecraft, tu dois d'abord effectuer quelques manipulations pour vérifier qu'il fonctionne correctement. Une fois que tu auras passé cette étape de vérifications, tu pourras relier le dispositif au jeu, comme tu viens de le faire avec la LED.

Si tu joues sur un PC ou un Mac, tu dois utiliser le kit anyio pour réaliser ce projet. Ce kit est disponible dans le kit de démarrage, mais tu peux aussi le télécharger depuis les sites indiqués au début de ce chapitre.



Qu'est-ce qu'un afficheur 7 segments ?

Si tu observes autour de toi ou que tu te rends dans le magasin le plus proche, tu verras qu'une multitude d'objets divers et variés sont dotés d'afficheurs 7 segments. Montres numériques, micro-ondes à affichage digital, lecteurs CD et même boîtiers de contrôle de chauffage ; tous ces dispositifs utilisent un modèle à sept segments. Sur la figure 5-10, tu peux voir le schéma d'un afficheur 7 segments et son circuit interne.

Copyright © 2016 Eyrolles.

Tous les afficheurs 7 segments sont dotés de plusieurs LED. L'affichage le plus commun, celui que tu utiliseras dans ce projet, est en fait doté de huit LED, l'une d'elles servant à afficher le séparateur décimal en bas à droite. Seules sept LED correspondent au modèle en question et forment un huit, comme tu peux le voir sur la figure 5-10.

Dans le cadre de ce projet, tu dois retenir deux informations importantes au sujet des afficheurs 7 segments :

- si tu sais rédiger un programme capable d'allumer et d'éteindre une LED, alors tu sauras aussi rédiger un programme capable de contrôler ce dispositif, car il contient huit LED séparées, soit huit connexions GPIO séparées ;
- lorsque tu utilises ce modèle d'afficheur 7 segments, tu peux afficher toute une succession de symboles en composant avec les différentes LED. Grâce à ces simples segments, tu as la possibilité d'afficher n'importe quel chiffre de 0 à 9.

Dans la figure 5-10, outre le circuit interne de l'afficheur 7 segments, tu disposes également d'un schéma indiquant comment les broches sont reliées aux segments. Tu remarqueras que chaque LED dispose d'une anode connectée à la même broche. C'est cette broche que tu brancheras sur la barrette positive de ta plaque d'essais. Quant à la flèche du symbole de la diode, elle t'indique le sens du courant, qui part du côté positif vers le côté négatif.

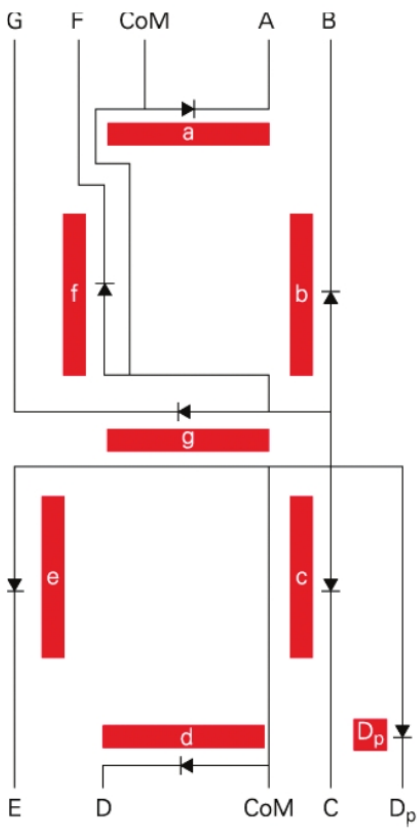


FIGURE 5-10 La disposition des segments et le circuit interne d'un afficheur 7 segments à anode commune

Copyright © 2016 Eyrolles.

Dans l'industrie, on utilise généralement les lettres A B C D E F G pour désigner les segments de l'afficheur. Si tu observes celui-ci de plus près, tu remarqueras que les segments sont légèrement inclinés vers la droite. Cela n'a qu'une fonction esthétique, que les fabricants ont adoptée pour améliorer l'apparence de ces composants.



Il existe tout un tas d'afficheurs 7 segments en vente sur le marché, et bon nombre d'entre eux ont des circuits différents. Certains sont à cathode commune, c'est-à-dire que toutes les cathodes des LED sont reliées entre elles. D'autres sont à anode commune, ce qui veut dire que toutes les anodes des LED sont reliées entre elles. Tu trouveras également des circuits agencés différemment. Pour ce projet, j'utilise un afficheur 7 segments à anode commune un peu spécial que je me suis procuré sur le site `www.selectronic.fr`, mais si tu utilises un afficheur différent, vérifie bien la configuration du circuit avant de le connecter. Si c'est ton cas et que tu utilises, par exemple, un afficheur à cathode commune, j'ai ajouté quelques commentaires à ton attention dans les scripts de ce chapitre, à côté de la constante `ON`, pour que tu saches comment adapter tes instructions.



Brancher l'afficheur 7 segments

Toutes les conditions sont maintenant réunies pour que tu puisses brancher ton afficheur 7 segments à la plaque d'essais et vérifier qu'il fonctionne correctement. Suis ces quelques étapes :

1. L'afficheur 7 segments étant doté de cinq broches en haut et cinq broches en bas, insère-le au centre de ta plaque d'essais, de sorte que les dix broches se trouvent à distance égale.
2. Connecte un fil à la moitié supérieure de l'afficheur, en partant de la broche du milieu (l'anode commune) et relie-le à la barrette de 3,3 volts située dans la partie supérieure de la plaque d'essais. (Si tu utilises un afficheur différent du mien, par exemple un afficheur à cathode commune, tu devras connecter le fil à la moitié inférieure de l'afficheur, en partant de la broche du milieu, et le relier à la barrette de 0 volt située dans la partie inférieure de la plaque d'essais.)
3. Comme cet afficheur contient des LED, tu devras utiliser des résistances, à l'exemple de l'exercice précédent. Et comme il contient précisément huit LED, tu devras utiliser huit résistances. Prends chacune des résistances de 330 ohms (code couleur : orange, orange, marron), plie-leur les pattes et place-les de part et d'autre de l'afficheur, comme sur la figure 5-12. Afin d'éviter tout court-circuit, réfère-toi à la figure 5-4 pour te rappeler la configuration du circuit de la plaque d'essais.

Copyright © 2016 Eyrolles.

4. Tu peux maintenant tester ton afficheur. Pour cela, tu vas utiliser un fil que tu brancheras temporairement au circuit. Connecte une extrémité du fil à la bande négative de la plaque (dans la partie inférieure) et utilise l'autre extrémité pour la connecter tour à tour aux pattes libres des résistances, c'est-à-dire aux pattes qui ne sont pas branchées à l'afficheur. Chaque fois que tu toucheras une patte, un segment de l'afficheur s'allumera. Compare les segments au schéma de la figure 5-10, qui correspond à l'afficheur que j'ai utilisé dans ce projet. Puis vérifie si le circuit interne de ton afficheur correspond au schéma de ce livre en t'aidant des lettres indiquées sur chaque segment. Si ton circuit est différent, note les lettres des segments qui correspondent aux résistances, car tu en auras besoin par la suite. (Voir la figure 5-11.)

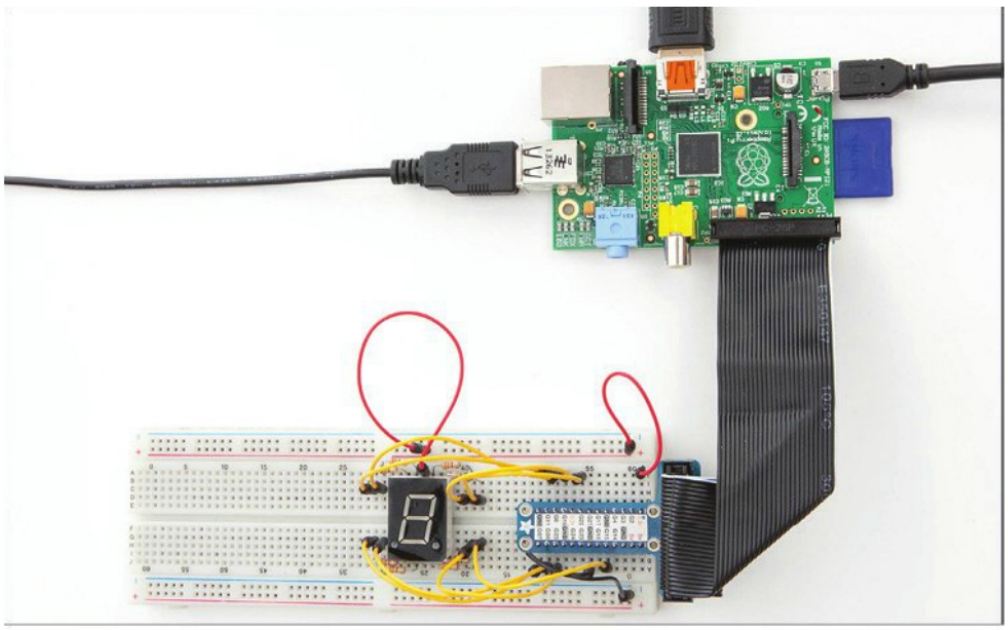


FIGURE 5-11 La LED à 7 segments est branchée à la plaque, prête à être testée sur Raspberry Pi.

Une fois que tu auras terminé les essais et que tu seras parvenu à faire fonctionner l'afficheur, la prochaine étape consistera à connecter ton afficheur au port GPIO de ton ordinateur. Tu pourras ainsi contrôler les LED du périphérique directement depuis ton interpréteur de commandes Python. Retire le fil de test de la plaque et munis-toi d'autres fils. Avec ces derniers, connecte chaque patte libre des résistances aux broches du port GPIO, en suivant le schéma de la figure 5-12 (pour un Raspberry Pi) ou de la figure 5-13 (pour un circuit Arduino). Si, lors du test précédent, tu as découvert que ton circuit était différent du mien, aide-toi des notes que tu as prises pour déterminer les lettres correspondant aux résistances et utilise ces informations pour connecter chaque segment aux broches GPIO qui conviennent. Si tu négliges cette étape, tu risques de voir des motifs bien étranges s'afficher lorsque tu exécuteras ton programme Python !

Copyright © 2016 Eyrolles.

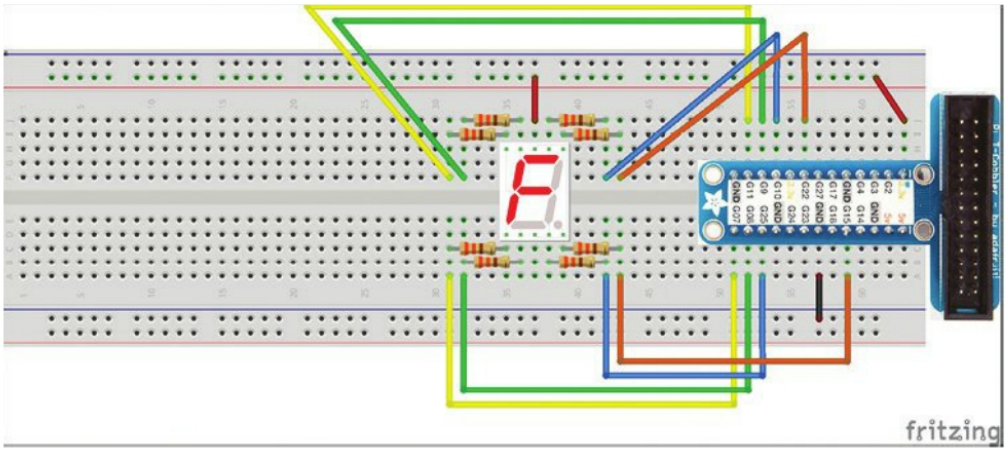


FIGURE 5-12 Schéma du circuit de l'afficheur 7 segments sur Raspberry Pi

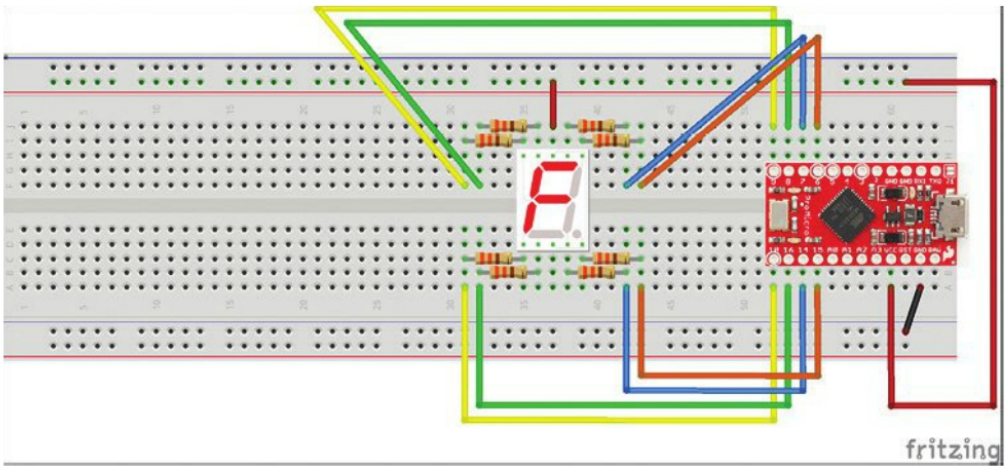


FIGURE 5-13 Schéma du circuit de l'afficheur 7 segments sur Arduino

Programmer en Python pour contrôler l'afficheur 7 segments

Maintenant que tu as testé ton afficheur 7 segments et que tu l'as correctement connecté aux huit broches GPIO de ton ordinateur, tu peux commencer à écrire des programmes Python pour l'activer.

1. Crée un nouveau programme en sélectionnant `File>New File`, puis nomme-le `testAfficheur.py`.
2. Importe tous les modules nécessaires et définis des constantes en leur affectant les numéros des huit broches GPIO :

Copyright © 2016 Eyrolles.

Sur Raspberry Pi

```
import RPi.GPIO as GPIO
BROCHES_LED = [10,22,25,8,7,9,11,15] # ordre important
```

Sur Arduino

```
import anyio.GPIO as GPIO
BROCHES_LED = [7,6,14,16,10,8,9,15] # ordre important
```

3. Détermine le mode de numérotation du port GPIO :

```
GPIO.setmode(GPIO.BCM)
```

4. Choisis ton type d'afficheur (pour ce qui me concerne, j'ai utilisé un afficheur à anode commune dans mes circuits) :

```
ON = False # False = anode commune, True = cathode commune
```

5. Rédige une boucle configurant les huit broches en mode sortie :

```
for g in BROCHES_LED:
    GPIO.setup(g, GPIO.OUT)
```

6. Comme tu le sais déjà, les LED de ton afficheur seront soit allumées, soit éteintes. Il existe un super moyen de définir un beau modèle de fonctionnement pour ton afficheur : les listes. En effet, tu peux construire un superbe modèle en enregistrant huit valeurs `True` ou `False` dans une liste, chacun des index de la liste correspondant à une LED de l'afficheur. Te souviens-tu que tu as appris à créer des listes avec des index dans le chapitre 4 ? Eh bien, c'est exactement ce que tu vas faire ici. Dans la liste ci-dessous, tous les segments de l'afficheur (A, B, C, D, E, F et G) seront définis sur ON (à l'aide de la valeur `True` en guise d'index), à l'exception du séparateur décimal (DP) qui sera désactivé, car tu placeras la valeur `False` en fin de liste. L'index `[0]` dans la liste correspond au segment A, l'index `[1]` au segment B, et ainsi de suite.

```
modele = [True, True, True, True, True, True, True, False]
          # ABCDEFG (sauf DP)
```

7. Rédige une boucle qui passera en revue toutes les broches de ton modèle :

```
for g in range(8):
    if modele[g]:
        GPIO.output(BROCHES_LED[g], ON)
    else:
        GPIO.output(BROCHES_LED[g], not ON)
```

8. La fin du programme ne tient désormais plus qu'à une touche. Tu dois demander à ton programme d'attendre ton signal avant de s'arrêter, sinon toutes les LED

Copyright © 2016 Eyrolles.

s'éteindront et il ne se passera rien. Pour cela, utilise la fonction `raw_input()`. Elle exige en effet que tu appuies sur la touche `Entrée` pour que ton programme passe à la ligne d'après :

```
raw_input("fini ?")
```

9. Enfin, demande au programme de nettoyer le port GPIO avant de s'arrêter :

```
GPIO.cleanup()
```

Exécute le programme et vois ce qu'il se passe. Si tu utilises un Raspberry Pi, n'oublie pas d'exécuter ce programme depuis LXTerminal à l'aide de la commande `sudo python testAfficheur.py`.

DÉFI

Essaye de calculer les valeurs à insérer dans ton modèle pour afficher les chiffres de 0 à 9. Rappelle-toi que les valeurs `True` et `False` dans la liste modèle sont agencées dans l'ordre suivant : A, B, C, D, E, F, G, DP. Teste-les sur ton afficheur. Combien de lettres de l'alphabet peux-tu faire apparaître sur ton afficheur 7 segments ? Pour t'aider, tu trouveras des exemples pratiques de modèles sur cette page Wikipédia : `https://fr.wikipedia.org/wiki/Afficheur_7_segments`.



Utiliser un module Python pour contrôler l'afficheur

Tu t'es probablement aperçu qu'il existe une multitude de modèles pratiques à utiliser sur ton afficheur 7 segments. Pour te faciliter la tâche, je t'ai confectionné un petit module Python que j'ai appelé `seg7.py` et que j'ai inclus dans le dossier `anyio` de ton kit de démarrage. Tu peux utiliser et modifier ce module autant que tu le souhaites et l'ajouter à tes propres jeux et programmes. En suivant ces quelques étapes, tu verras qu'il est très facile de l'insérer dans un programme existant.

1. Commence un nouveau programme en sélectionnant `File>New File`, puis enregistre-le sous le nom `testAfficheur2.py`.
2. Importe mon module d'afficheur ainsi que le module de temps pour pouvoir ajouter un délai.

```
import anyio.seg7 as display
import time
```

3. Copie toutes les lignes de ton script `testAfficheur.py` depuis le début jusqu'à la commande `ON=False`.

Copyright © 2016 Eyrolles.

4. Ensuite, configure le module d'afficheur en lui donnant accès aux broches de ton périphérique GPIO et en lui fournissant une liste des numéros de broches à utiliser pour chaque segment :

```
display.setup(GPIO, BROCHES_LED, ON)
```

5. Rédige une boucle qui compte de 0 à 9 indéfiniment. Si j'ai choisi d'utiliser une boucle ici, c'est pour que tu puisses la laisser tourner indéfiniment pendant que tu modifies le câblage de ton afficheur, au cas où certains segments ne fonctionneraient pas. Ainsi, tu ne seras pas obligé d'exécuter ton programme à chaque fois.

```
try:
    while True:
        for d in range(10):
            display.write(str(d))
            time.sleep(0.5)
finally :
    GPIO.cleanup()
```

Enregistre le programme, puis exécute-le et observe ce qu'il se passe. L'afficheur devrait compter de 0 à 9 sans discontinuer. N'oublie pas que sur Raspberry Pi tu dois utiliser `sudo python testAfficheur2.py` dans une fenêtre LXTerminal.



Au lieu d'utiliser `time.sleep(0.5)` dans le script précédent, tu peux employer un délai plus long, de deux secondes par exemple. Tu peux même utiliser une fonction `raw_input()` pour que le modèle affiché reste dans la même position, pendant que tu manipules le câblage de ton circuit et que tu t'assures que tout est correctement en place.

DÉFI



Démarre le programme `seg7.py` qui se trouve dans le dossier `MyAdventures/anyio` et suis les instructions du fichier qui t'indiquent comment ajouter de nouveaux symboles. Parmi les lettres que tu as créées dans le défi précédent, identifie celles que tu peux coder dans ce module. Quels mots peux-tu épeler à l'aide de l'afficheur ? Peux-tu, par exemple, écrire ton prénom ? Sinon, quelles lettres te manque-t-il ?

Copyright © 2016 Eyrolles.

Créer un détonateur

Dans le projet final de ce chapitre, tu vas construire un énorme détonateur rouge. Lorsque tu appuieras sur le bouton-poussoir, il déclenchera un compte à rebours sur l'afficheur 7 segments. Ton joueur aura donc assez de temps pour courir et se mettre à l'abri avant l'explosion. Puis, un énorme cratère apparaîtra à l'endroit où ton joueur se trouvait lorsque tu as actionné le détonateur. Ce nouveau programme constitue un accessoire bien pratique à ajouter à ta boîte à outils, car il te permet de faire de l'espace en un rien de temps dans Minecraft et de construire des structures. En outre, ce projet va te faire découvrir un autre mode de GPIO : le mode d'entrée. Il sert notamment à détecter lorsqu'un bouton est actionné, donc tu pourras mettre tes connaissances sur l'afficheur 7 segments en pratique !

Tu peux regarder une vidéo didactique sur la création d'un détonateur en te rendant sur le site Internet du livre : `www.editions-eyrolles.com/dl/14292`. Choisis la vidéo « Adventure 5 » (en anglais).



Pour les besoins de ce projet et si tu es sur Mac ou PC, tu devras utiliser le kit anyio. Il t'est fourni dans le kit de démarrage.



Réaliser le câblage d'un bouton

La première étape de ce projet consiste à réaliser le câblage du bouton-poussoir de ton détonateur. Choisis un bouton rouge pour qu'il ait l'air encore plus foudroyant !

Le type d'interrupteur que tu vas utiliser dans cet exercice est un bouton-poussoir tactile. Cela veut dire que le bouton ne laisse passer le courant que lorsque tu appuies dessus, refermant ainsi le circuit. Cependant, le bouton ne restera pas en position fermée indéfiniment ; dès que tu relèveras le doigt, cela coupera le courant.

Sur la figure 5-14, tu peux voir à quoi ressemble le circuit sur Raspberry Pi ; sur la figure 5-15, tu as le schéma du circuit sur Arduino. Pour réaliser ton câblage, tu peux soit te concentrer sur ces schémas, soit suivre les instructions énumérées ci-dessous, à toi de voir !

Copyright © 2016 Eyrolles.

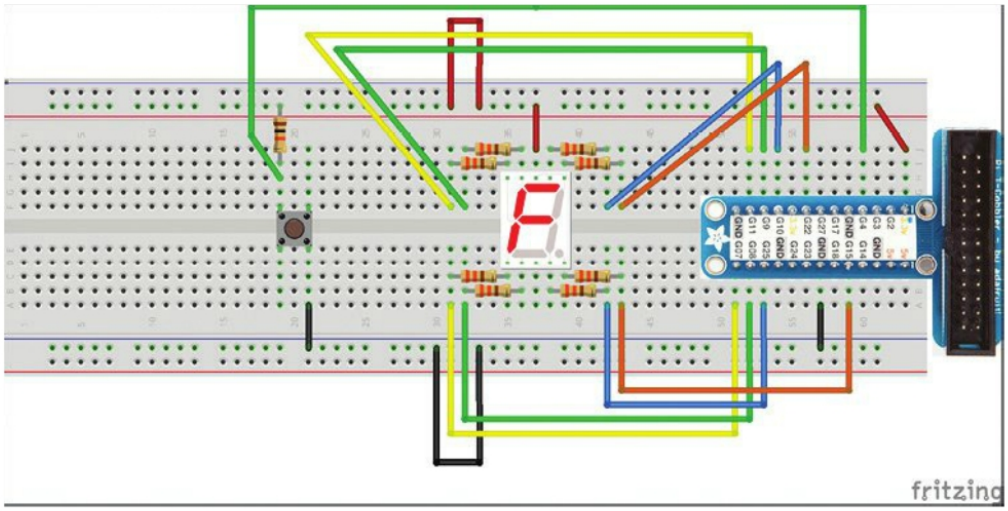


FIGURE 5-14 Schéma du circuit d'un bouton raccordé à un Raspberry Pi

1. Place ton bouton sur une partie encore libre de ta plaque d'essais. Le bouton que j'ai utilisé (voir la figure 5-16) est doté de quatre broches qui s'insèrent parfaitement au centre de la plaque. En appuyant sur le bouton, tu vas connecter les broches de gauche à celles de droite, à condition de les avoir placées sur les parties supérieure et inférieure de la plaque, comme tu peux le voir sur les figures 5-14 et 5-15.

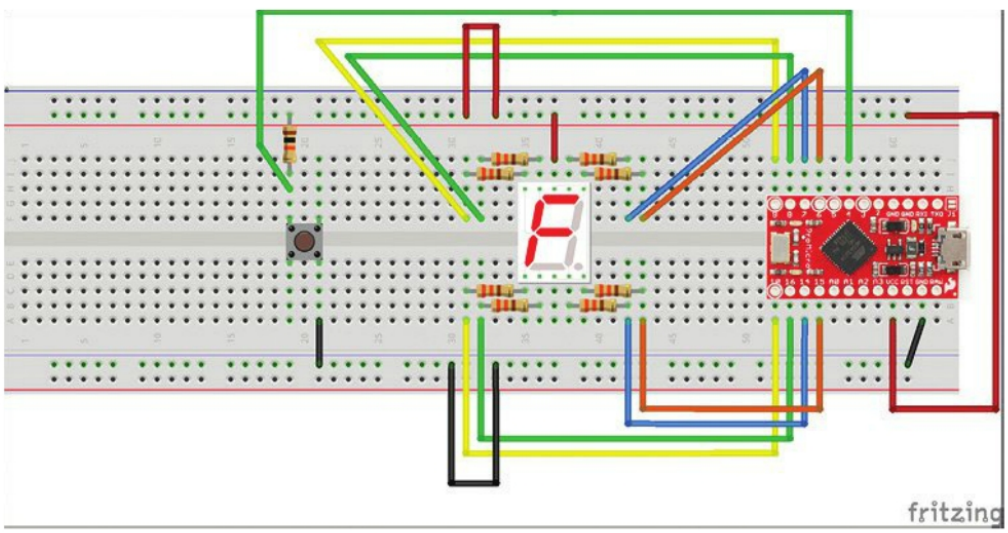


FIGURE 5-15 Schéma du circuit d'un bouton raccordé à un circuit Arduino

2. Tu vas avoir besoin d'une résistance pour tirer le courant d'entrée du bouton à 3,3 volts. Si tu n'utilises pas de résistance, ton bouton provoquera de fausses pressions et ton détonateur n'arrêtera pas de s'éteindre ! (Pour en savoir plus sur la nécessité d'utiliser une résistance dans ce cas précis, tu peux lire l'encadré « Percer les secrets du circuit ».) Place ta résistance de tirage 10 K (code couleur :

Copyright © 2016 Eyrolles.

marron, noir, orange) en connectant une patte à la broche du côté gauche du bouton et l'autre patte à la bande de 3,3 volts située en haut de la plaque.

3. Pose un fil électrique entre la résistance et le bouton et branche l'autre extrémité à la broche 4 du GPIO.

4. Connecte ensuite un autre fil électrique à la broche du côté inférieur droit du bouton et relie-le à la bande de 0 volt située en bas de ta plaque. Pour savoir comment fonctionne ce raccordement, consulte l'encadré « Percer les secrets du circuit », mais de manière générale, tous les boutons sont raccordés aux ordinateurs de cette façon. La figure 5-16 te montre notamment comment raccorder un bouton à une broche GPIO.



Il existe des boutons-poussoirs en tous genres. Si tu utilises le même type de bouton que moi pour ce projet, tu remarqueras que les quatre broches sont légèrement inclinées pour mieux s'accorder à la plaque et faciliter la soudure. Fais très attention lorsque tu insères le bouton dans la plaque et veille à ne pas casser les broches. Pour cela, il te suffit d'appuyer de façon uniforme et assez fermement sur le bouton. Si tu possèdes une petite pince universelle, tu peux également l'utiliser pour aplatir les broches et faciliter ainsi leur insertion dans la plaque d'essais.

PERCER LES SECRETS DU CIRCUIT

Après avoir inséré ton bouton sur la plaque, tu as utilisé une résistance qui est appelée « résistance de tirage », car elle tire le courant de la broche jusqu'à 3,3 volts.

Lorsque le bouton n'est pas actionné et que le circuit est ouvert, le fil reliant la broche GPIO au bouton fournit un courant de 3,3 volts, que la broche du GPIO traduira numériquement comme le chiffre 1. En revanche, lorsque tu appuies sur le bouton, son mécanisme interne va refermer le circuit en reliant ses deux broches gauches à ses deux broches droites. Le fil du port GPIO va alors être tiré fortement vers le bas, vers un courant de 0 volt, à savoir celui de la bande inférieure de la plaque. À ce moment-là, la broche GPIO va traduire ce courant comme un 0. La résistance est puissante (10 Kohms équivalent à 10 000 ohms), mais cette puissance est nécessaire, car sans elle, le bouton provoquerait un court-circuit en connectant la bande de 3,3 volts à celle de 0 volt, ce qui ferait redémarrer ton ordinateur ! Cela est fortement déconseillé, car dans certains cas, un tel redémarrage pourrait endommager ton ordinateur. Quant à la dénomination 10 K, il s'agit d'une abréviation que les ingénieurs utilisent pour désigner une valeur de 10 000 ohms.

Copyright © 2016 Eyrolles.

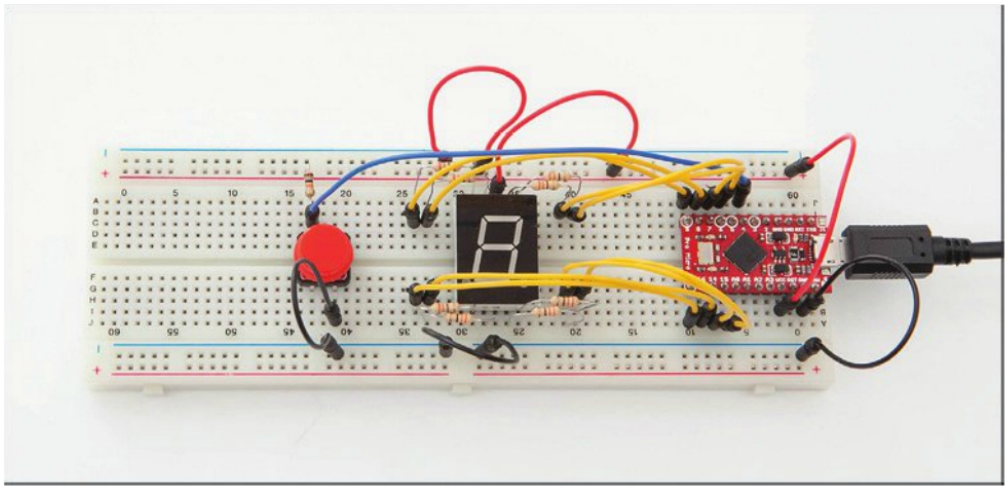


FIGURE 5-16 Sur cette plaque, le bouton est raccordé à un port GPIO via un circuit Arduino et dispose d'une résistance de tirage.

Rédiger le programme du détonateur

Maintenant que tu as raccordé ton bouton, il ne te reste plus qu'à rédiger un programme Python qui te permettra de surveiller l'état de ton bouton à l'aide d'une boucle. Lorsqu'il détectera le signal numérique 0 sur la broche GPIO (autrement dit, lorsque tu appuieras sur le bouton), il lancera un compte à rebours de 5 à 0 sur l'afficheur 7 segments. Il provoquera alors une énorme explosion dans le monde de Minecraft en formant un gigantesque cratère.

1. Crée un nouveau programme en sélectionnant `File>New File`, puis nomme-le `detonateur.py`.
2. Importe les modules dont tu as besoin :

```
import mcpi.minecraft as minecraft
import mcpi.block as block
import time
import anyio.seg7 as display
```

3. Configure les paramètres GPIO de ton ordinateur :

Sur Raspberry Pi

```
import RPi.GPIO as GPIO
BOUTON = 4
BROCHES_LED = [10,22,25,8,7,9,11,15] # ordre important
```

Sur Arduino

```
import anyio.GPIO as GPIO
```

Copyright © 2016 Eyrolles.

```
BOUTON = 4
BROCHES_LED = [7,6,14,16,10,8,9,15] # ordre important
```

4. Configure la broche GPIO du bouton en mode entrée et configure aussi les broches GPIO de l'afficheur.

```
GPIO.setmode(GPIO.BCM)
GPIO.setup(BOUTON, GPIO.IN)
ON = False # False = anode commune, True = cathode commune
display.setup(GPIO, BROCHES_LED, ON)
```

5. Connecte-toi au jeu Minecraft :

```
mc = minecraft.Minecraft.create()
```

6. Rédige une fonction pour provoquer les événements suivants : lâcher un bloc de TNT à un endroit précis, déclencher un compte à rebours et former un gros cratère là où le bloc de TNT a été déposé. Les lignes de code suivantes vont poser un bloc de TNT légèrement à côté de ton joueur afin qu'il ne lui tombe pas sur la tête ! Le cratère que tu vas provoquer mesure 20 blocs de longueur (10 à gauche et 10 à droite de Steve). Ses dimensions sont calculées dans la fonction `setBlocks()`, comme ceci :

```
def bombe(x, y, z):
    mc.setBlock(x+1, y, z+1, block.TNT.id)
    for t in range(6):
        display.write(str(5-t))
        time.sleep(1)

mc.postToChat("BOUM !")
mc.setBlocks(x-10, y-5, z-10, x+10, y+10, z+10, block.AIR.id)
```

7. Rédige la boucle de jeu principale de façon à ce qu'elle attende l'activation du détonateur puis qu'elle jette une bombe. Souviens-toi que ton bouton se connecte à la bande de 0 volt lorsque tu appuies dessus et que tu dois donc comparer le port GPIO avec `False` pour qu'il détecte le déclenchement du bouton.

```
try:
    while True:
        time.sleep(0.1)
        if GPIO.input(BOUTON) == False:
            pos = mc.player.getTilePos()
            bombe(pos.x, pos.y, pos.z)
finally:
    GPIO.cleanup()
```

Sauvegarde ton programme et exécute-le. Si tu as un Raspberry Pi, n'oublie pas de saisir l'instruction `sudo python detonateur.py` dans la fenêtre LXTerminal.

Copyright © 2016 Eyrolles.

Trouve un endroit dans Minecraft et appuie sur le gros bouton rouge de ton détonateur, puis cours t'abriter quelque part pour sauver ta peau ! La figure 5-17 te montre à quoi ressemble l'énorme cratère après l'explosion.



FIGURE 5-17 Un gigantesque cratère formé par une explosion dans Minecraft



La taille du cratère est déterminée par les nombres contenus dans la fonction `setBlocks()`. Mais penses-tu que c'est le meilleur moyen de la définir ? Comment t'y prendrais-tu si tu souhaitais doubler sa taille et combien de lignes de ton script modifierais-tu pour cela ? Tu me prends peut-être pour un fou, mais je suis persuadé que tu peux trouver un moyen de modifier très simplement la taille du cratère.

DÉFI



Crée une explosion du tonnerre en ajoutant des lignes de géorepérage à ton programme, comme dans le chapitre 2, afin de détecter si ton joueur se trouve dans le périmètre de l'explosion.

Utilise une fonction `mc.player.setTilePos()` pour propulser ton joueur dans le ciel au moment de l'explosion, si celui-ci se trouve dans le rayon de la détonation.

Copyright © 2016 Eyrolles.

DÉFI



Ajoute trois nouveaux boutons à ton circuit, comme tu l'as fait pour ton premier bouton. Rédige des codes supplémentaires pour provoquer des événements différents à l'aide de ces boutons. Par exemple, tu peux publier des messages aléatoires dans le chat Minecraft, téléporter ton joueur dans des cachettes secrètes, poser des blocs de types différents à ton emplacement, ou encore bâtir une maison en face de Steve. Pour cela, tu peux utiliser les numéros de broches GPIO suivants et les attribuer aux trois autres boutons.

BOUTON	RASPBERRY PI	ARDUINO
BOUTON2	14	5
BOUTON3	23	2
BOUTON4	24	3



Laisse ta plaque de montage telle quelle une fois que tu auras terminé le chapitre. Martin utilisera cette même configuration dans le jeu du chapitre 9, et tu l'utiliseras aussi dans le chapitre bonus, que tu peux télécharger sur le site Internet associé (www.editions-eyrolles.com/dl/14292).

Copyright © 2016 Eyrolles.

Tableau de références	
Configurer les broches GPIO	**Interpréter et contrôler des broches GPIO**

`import RPi.GPIO as GPIO # RaspberryPi`
`import anyio.GPIO as GPIO # Arduino`
`GPIO.setmode(GPIO.BCM)`
`GPIO.setup(5, GPIO.OUT) # sortie`
`GPIO.setup(5, GPIO.IN) # entrée`
	`GPIO.output(5, True) # ouvert (3.3V)`
	`GPIO.output(5, False) # fermé (0V)`
	`if GPIO.input(6) == False:`
	`    print("enfoncé")`

Remettre en état les broches GPIO après utilisation	**Utiliser le module de l'afficheur 7 segments**

`try:`
`    fais_quelquechose() # rédige le`
`                       # code ici`
`finally :`
`    GPIO.cleanup()`
	`import anyio.seg7 as display`
	`BROCHES_LED = [10,22,25,8,7,9,11,15]`
	`ON = False # anode commune`
	`ON = True # cathode commune`
	`display.setup(GPIO, BROCHES_LED, ON)`

Afficher des caractères sur l'afficheur 7 segments	**Autres fonctions de l'afficheur 7 segments**

`display.write("3")`
`display.write("A")`
`display.write("up")`
	`display.setdp(True) # séparateur`
	`                     # décimal activé`
	`display.setdp(False) # séparateur`
	`                     # décimal`
	`                     # désactivé`
	`display.clear()`
	`display.pattern([1,1,1,1,1,1,1,1])`

Explorer d'autres pistes de programmation impliquant des circuits électroniques

Dans ce chapitre, tu as relié le monde de Minecraft au monde réel et tu as élargi ton propre horizon en t'initiant à la discipline fascinante de l'électronique. En effet, tu as appris à détecter et à prendre le contrôle d'objets du monde réel. Tu as employé tes nouvelles connaissances pour faire clignoter des LED, faire apparaître des formes sur des afficheurs 7 segments et détecter lorsque des boutons sont actionnés. Mieux encore, tu as réussi à t'échapper des tréfonds du monde de Minecraft. Avec ces nouveaux acquis, tu peux désormais créer tes propres manettes de jeu et périphériques d'affichage !

- Il n'y a pas assez de place dans ce livre pour réaliser d'autres jeux avec des circuits électroniques. Mais dès que j'ai commencé à penser à des idées de jeu, elles fusaient tellement dans mon esprit que je n'ai pas su m'arrêter ! Impossible, par exemple, de te laisser terminer ce livre sans t'enseigner le jeu de l'Ascenseur Minecraft. C'est

Copyright © 2016 Eyrolles.

pourquoi je l'ai inclus dans le chapitre bonus, que tu peux télécharger sur le site Internet du livre (www.editions-eyrolles.com/dl/14292). Il faut absolument que tu l'essayes ! Quant à moi, je ne compte plus les heures que j'ai passées à faire défiler l'ascenseur à toute vitesse, des bas-fonds du monde de Minecraft jusqu'aux frontières de l'espace. Je me suis même amusé à construire de gigantesques tours pour l'abriter ! Un jour, je construirai la réplique parfaite du gratte-ciel Shard (http://www.the-shard.com) autour de l'ascenseur, mais qui sait, peut-être que tu y parviendras avant moi !

- Un jour, l'un des élèves de notre club Raspberry Pi a eu une formidable idée : pourquoi ne pas concevoir une carte au trésor sur une plaque et l'orner de petites diodes ? Il suffirait de reproduire la carte du monde de Minecraft en dur et de l'orner de LED pour indiquer des défis à relever. Dès que le joueur aurait terminé un défi, la LED concernée s'allumerait pour indiquer sa victoire. J'aimerais tellement voir ce jeu se concrétiser ! J'espère qu'un jour je trouverai une vidéo YouTube le présentant. Seras-tu le premier à relever cet incroyable défi ?

- Un joueur a eu l'idée géniale de connecter un accéléromètre électronique à son ordinateur afin qu'il puisse détecter les mouvements de son bras et les reproduire à l'aide d'un bras mécanique. Regarde ce que ça donne sur cette vidéo : http://www.youtube.com/watch?v=_KGc9vlOrNk. Essaye toi aussi de raccorder un accéléromètre à ton ordinateur et de trouver un moyen d'interagir avec Minecraft en bougeant les bras ! Si tu y parviens, tu pourras présenter ton projet dans un salon d'exposition local et inspirer d'autres personnes à s'initier au monde fascinant de la robotique dans Minecraft !



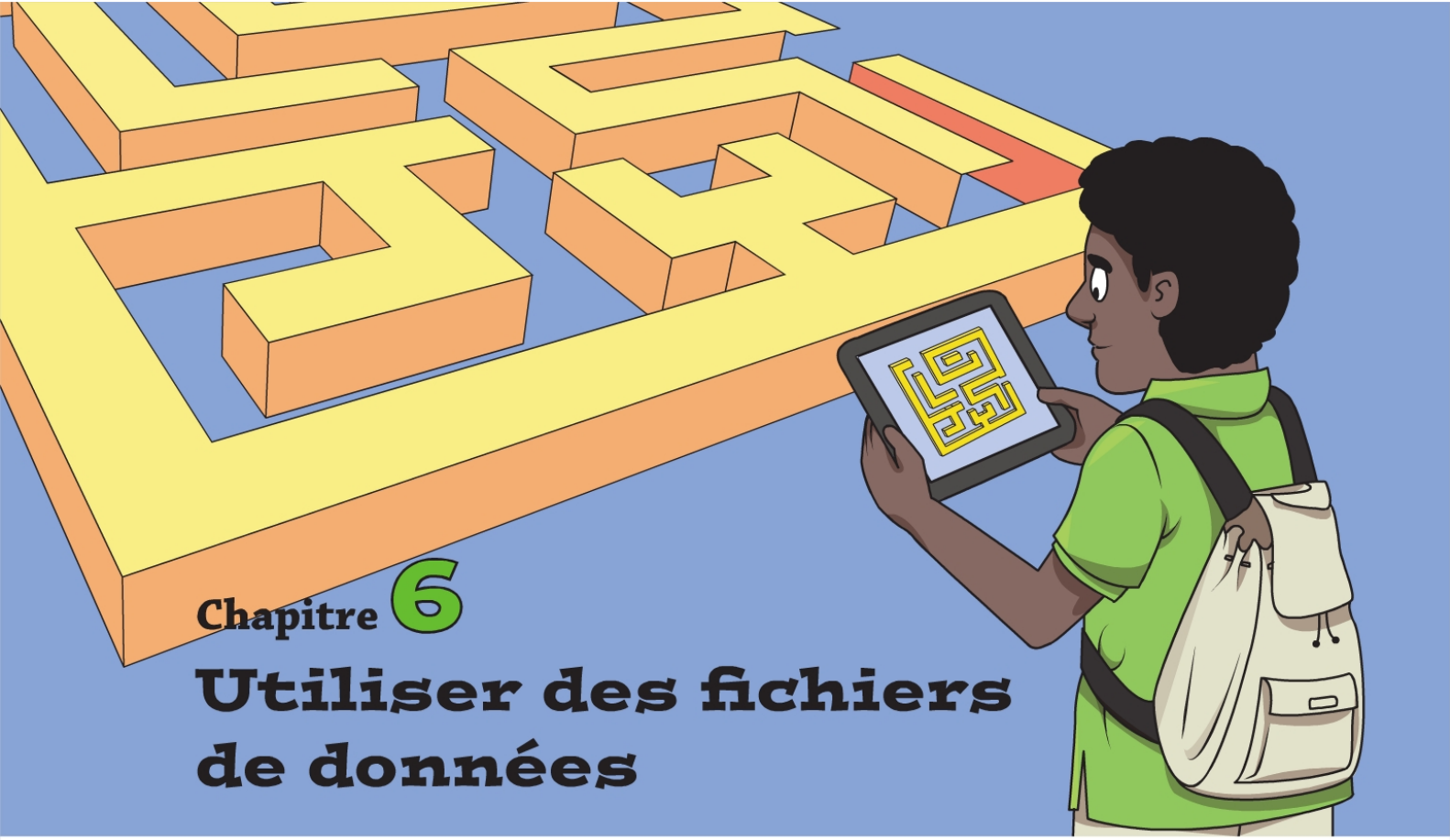
Niveau terminé : ça y est, tu as brisé les barrières qui séparaient le monde virtuel de Minecraft du monde réel ! Tu es maintenant l'un des pionniers de cette discipline de pointe que représente l'électronique dans Minecraft. Ton expérience de jeu n'a désormais plus aucune limite !

Copyright © 2016 Eyrolles.

Dans le prochain chapitre...

Dans le chapitre 6, tu vas apprendre à lire des données dans des fichiers pour construire de gigantesques structures automatiquement, telles que des labyrinthes. Tu utiliseras tes nouvelles compétences pour développer une immense machine capable de dupliquer de larges objets en 3D, n'importe où dans Minecraft. Même les arbres ne seront plus enracinés dans le sol comme avant !

Copyright © 2016 Eyrolles.



Chapitre 6

Utiliser des fichiers de données

C'EST LORSQUE TU commences à manipuler d'importantes quantités de données que la programmation trouve tout son intérêt. Ton programme devient alors une série de règles qui déterminent la manière dont ces données vont être lues et traitées et la forme qu'elles vont prendre. C'est à partir de là que les données vont commencer à jouer un rôle important.

Dans ce chapitre, tu vas d'abord apprendre à créer des fichiers de texte qui vont te servir à concevoir des labyrinthes. Cela te permettra de construire automatiquement des labyrinthes dans Minecraft pour que tes amis et toi puissiez vous y perdre. Tu verras qu'avec un petit programme Python, il est possible de construire des structures aussi grandes que des labyrinthes. Tu n'en croiras pas tes yeux !

Tu vas ensuite t'inspirer de ton programme de création de labyrinthe pour en développer un autre beaucoup plus complet, qui te permettra de bâtir une gigantesque structure de scannage et d'impression en 3D, également appelée salle de reproduction. Tout ce que tu confectionneras à l'intérieur pourra être enregistré dans un fichier et réutilisé plus tard. Tu pourras notamment téléporter tes objets à des endroits différents dans Minecraft et même les charger sur d'autres ordinateurs pour les faire apparaître chez tes amis ! Tu te constitueras ton propre répertoire d'objets et pourras les reproduire si rapidement dans Minecraft que tous tes amis voudront eux aussi posséder une machine à dupliquer !

Copyright © 2016 Eyrolles.

Lire une donnée dans un fichier

Un programme informatique n'est pas aussi intelligent qu'il en a l'air : il se contente bêtement de suivre les instructions qu'on lui donne. Si on ne change aucune donnée du programme, il s'exécutera toujours de la même manière. Les programmes Minecraft que tu as réalisés jusqu'à présent sont assez interactifs, car ils changent de comportement en fonction des événements qui surviennent dans le jeu. Ces événements constituent des données pour le programme.

L'intérêt des fichiers de données

Il existe une autre manière de changer le comportement d'un programme informatique chaque fois que tu l'exécutes : tu peux enregistrer des données dans un fichier texte et demander à ton ordinateur de le lire au démarrage. Tu peux alors créer tout un dossier de fichiers de texte et même rédiger un menu pour choisir le fichier à ouvrir en fonction de ce que tu souhaites réaliser avec le programme. On dit de ce type de programme qu'il est « orienté donnée », car ce sont principalement les données stockées dans le fichier de texte externe qui vont l'influencer.

En y réfléchissant, la plupart des programmes que tu utilises sur ton ordinateur font déjà appel à des fichiers de données. Le programme de traitement de texte que tu utilises pour faire tes devoirs enregistre ton travail dans un fichier de données ; lorsque tu prends une photo avec un appareil numérique, elle est stockée dans un fichier de données et ton programme d'édition de photo accède à ton image depuis ce fichier. Même Minecraft utilise des fichiers de données en coulisses, pour enregistrer ou charger ton monde par exemple, mais aussi pour les kits de textures qui servent à personnaliser tes blocs. Utiliser des fichiers de données dans un programme te permet de travailler en toute flexibilité, car tu peux ainsi lui demander d'exécuter tout un tas de tâches différentes, sans pour autant avoir besoin de le modifier à chaque fois pour y parvenir. L'avantage est que tu peux aussi partager ces fichiers de données avec tes amis, si toutefois ils possèdent le même programme que toi.



Le programme que j'ai publié sur mon blog il y a quelque temps est un bel exemple d'utilisation de fichiers de données dans Minecraft. Je l'ai baptisé le « Constructeur BMP de Minecraft » (`http://blog.whaleygeek.co.uk/minecraft-pi-with-python/`), car il permet de lire une image depuis un fichier bitmap (BMP) et de la reproduire bloc par bloc dans Minecraft. Grâce à ce programme, j'ai pu reproduire le logo de Raspberry Pi à l'identique dans le jeu. Il est tellement énorme que lorsqu'on se trouve à la base et qu'on lève la tête pour le regarder, on a du mal à croire qu'il s'agit du logo ! Ce programme est capable de lire n'importe quelle image BMP et de la reproduire à l'identique dans le jeu, sans qu'on ait besoin d'intervenir.

Lorsqu'on écrit un programme orienté données, il faut tout d'abord savoir ouvrir et lire des fichiers texte depuis son ordinateur à l'aide de Python.

Copyright © 2016 Eyrolles.

Créer un donneur d'indices

Pour apprendre à ouvrir et lire des fichiers texte, tu vas devoir écrire un programme relativement simple de donneur d'indices. Ce programme va te permettre de lire un fichier texte contenant des trucs et astuces sur le jeu Minecraft et d'afficher ces conseils un par un sur le chat, de manière aléatoire. Pour cela, tu dois d'abord créer un fichier et y rédiger des indices. Tu peux le faire dans l'éditeur Python, comme pour les programmes.

Lance Minecraft, IDLE ainsi que le serveur si tu es sur un PC ou un Mac. Tu t'es déjà certainement un peu familiarisé avec le processus, mais n'hésite pas à relire le chapitre 1 si tu as besoin de te rafraîchir la mémoire.

1. Dans IDLE, crée un nouveau fichier de texte en cliquant sur `File>New File` dans le menu. Enregistre ton programme en le nommant `astuces.txt`.

2. Puis énumère quatre ou cinq astuces en plaçant chacune sur une ligne de texte. Ci-après, j'ai rédigé quelques exemples d'astuces Minecraft pour t'aider, mais cela sera beaucoup plus amusant si tu inventes tes propres astuces. N'oublie pas d'appuyer sur `Entrée` à la fin de chaque ligne, pour créer une **fin de ligne** à l'intérieur du fichier lorsque tu changes d'astuce. Tu en sauras davantage sur les fins de ligne un peu plus tard dans ce chapitre.

```
Construis-toi un abri avant que les monstres n'apparaissent ↩
et ne viennent te dévorer.
Utilise des blocs d'eau pour alimenter des canaux souterrains.
Construis une structure en sable et détruis la base.
Utilise les deux modes de vue à la première et à la ↩
troisième personne.
Appuie deux fois sur la barre d'espace pour voler dans le ciel.
```

3. Enregistre ce fichier. Ne l'exécute pas puisqu'il ne s'agit pas d'un programme Python mais d'un fichier de texte que tu vas demander à Python d'utiliser.



Une **fin de ligne** est un caractère spécial invisible que l'ordinateur utilise pour marquer la fin d'une ligne de texte.

On l'appelle aussi parfois un « retour chariot », qui date de l'époque où on utilisait encore les machines à écrire. Lorsqu'on voulait aller à la ligne, on utilisait le retour chariot, qui faisait revenir le chariot à gauche de la page, une ligne plus bas.



N'utilise pas de virgule dans ton fichier `astuces.txt`, ni dans aucun des messages que tu veux faire apparaître avec `mc.postToChat()`. L'API de Minecraft comporte un petit bug qui coupe les phrases à l'endroit où on a placé une virgule.

Copyright © 2016 Eyrolles.

Maintenant que tu as créé ton fichier d'astuces, c'est-à-dire de données, il est temps de passer à l'écriture du programme qui va les traiter. Ce programme choisira une astuce au hasard dans le fichier et l'affichera au bout d'un certain temps dans le chat de Minecraft. Ainsi, pendant que tu joueras, tu verras s'afficher des conseils utiles.

1. Ouvre un nouveau fichier en cliquant sur `File>New File` et enregistre-le en le nommant `astuceChat.py`.

2. Importe les modules dont tu as besoin :

```
import mcpi.minecraft as minecraft
import time
import random
```

3. Puis connecte-toi au jeu Minecraft :

```
mc = minecraft.Minecraft.create()
```

4. Définis ensuite une constante avec le nom du fichier pour pouvoir l'échanger facilement par la suite avec un autre fichier :

```
NOMDEFICHIER = "astuces.txt"
```

5. Ouvre le fichier en mode de lecture (réfère-toi à l'encadré « Percer les secrets du code » pour obtenir plus de précisions sur cette étape) :

```
f = open(NOMDEFICHIER, "r")
```

6. Et lis toutes les lignes du fichier dans une liste que tu nommeras `astuces`. Souviens-toi que tu as déjà utilisé des listes dans les chapitres 4 (constructeur de pont magique) et 5 (électronique) :

```
astuces = f.readlines()
```

7. Puis ferme le fichier lorsque tu auras terminé de le lire. Dès lors que tu as fini d'utiliser un fichier, il faut toujours penser à le refermer, car s'il reste ouvert il ne pourra pas être utilisé par d'autres applications, telles que l'éditeur dans lequel tu as rédigé ton premier texte :

```
f.close()
```

8. Maintenant rédige une boucle de jeu avec des temps d'attente aléatoires, oscillant entre trois et sept secondes :

```
while True:
    time.sleep(random.randint(3,7))
```

9. Ensuite, dis au programme de choisir un message aléatoire dans la liste `astuces` et de l'afficher sur le chat. Pour cela, tu dois utiliser la fonction `strip()` (qui veut dire déshabiller ou dépouiller), pour te débarrasser de tous les caractères de fin de ligne

Copyright © 2016 Eyrolles.

superflus. Tu en sauras plus sur les caractères de fin de ligne un peu plus tard, donc ne t'inquiète pas si tu ne comprends pas bien de quoi il retourne pour l'instant.

```
msg = random.choice(astuces)
mc.postToChat(msg.strip())
```

Enregistre ton programme et exécute-le. Que se passe-t-il ? À mesure que tu avances dans le monde de Minecraft et que tu joues, tu devrais voir une astuce s'afficher sur le chat au petit bonheur, comme sur la figure 6-1. Observe la manière dont les messages apparaissent. Tu verras qu'ils disparaîtront petit à petit.



FIGURE 6-1 Des astuces apparaissent de façon aléatoire sur le chat au fil du jeu.

DÉFI

Ajoute des astuces à ton fichier `astuces.txt` en t'aidant de ton expérience du jeu Minecraft. Donne ton fichier `astuces.txt` et ton programme `astuceChat.py` à un ami qui débute dans Minecraft afin qu'il puisse l'utiliser et réaliser des choses extraordinaires dans le jeu. Tu peux même écrire toute une série de fichiers `astuces.txt` sur des thèmes différents et contenant plus ou moins de détails selon le niveau des joueurs. Publie-les ensuite sur un petit site Internet et laisse-les en libre accès pour que d'autres joueurs puissent les utiliser, sans oublier de partager aussi le programme `astuceChat.py`.



Copyright © 2016 Eyrolles.

PERCER LES SECRETS DU CODE

Dans le programme `astuceChat.py`, la ligne suivante renferme un petit mystère :

```
f = open(NOMDEFICHIER, "r")
```

La fonction `open()` permet d'ouvrir le fichier affecté à la constante `NOMDEFICHIER`. Mais que signifie le « `r` » à la fin ? Lorsque tu ouvres un fichier, tu dois préciser à la fonction `open()` le niveau d'accès que tu souhaites lui attribuer. Ici, le « `r` » (pour « read », qui signifie « lire » en anglais) te permet d'ouvrir le fichier en mode lecture uniquement. Si jamais tu essayes de modifier le texte du fichier, tu recevras un message d'erreur. Cette fonctionnalité te permet de protéger tes fichiers contre tout dysfonctionnement. Si tu veux ouvrir un fichier et le modifier, tu peux utiliser la lettre « `w` » (pour « write », qui signifie « écrire » en anglais) ; et si tu veux faire les deux à la fois, utilise « `rw` » (pour lire et écrire dans le fichier).

Enfin, que signifie le `f` ? Le `f` est en fait une autre variable, qui stocke ce qu'on appelle un descripteur de fichier. Un descripteur de fichier te permet de « saisir » un fichier. C'est une sorte de référence virtuelle du fichier qui se trouve dans le système de ton ordinateur. Si tu veux réaliser quelque chose sur un fichier ouvert, utilise la lettre `f` pour pouvoir t'y référer. Cette variable est très pratique, car comme toutes les autres variables que tu as utilisées auparavant, tu peux ouvrir plusieurs fichiers à la fois avec plusieurs variables de fichier, comme cela :

```
f1 = open("config.txt", "r")
f2 = open("niveaux.txt", "r")
f3 = open("score.txt", "rw")
```

Tu peux utiliser les variables `f1`, `f2` et `f3` dans la suite de ton script pour identifier correctement les fichiers que tu souhaites lire ou modifier.

Construire des labyrinthes à l'aide d'un fichier de données

Maintenant que tu sais comment demander à tes programmes de lire des fichiers de données, tu es prêt pour l'aventure. Tu sais déjà à quel point il est simple de placer des blocs dans le monde de Minecraft, grâce aux chapitres précédents. Mais que dirais-tu si tu pouvais construire des blocs en utilisant des listes de blocs répertoriées dans un fichier de données ? C'est exactement ce que tu vas faire ici : tu vas construire un grand labyrinthe en 3D dans Minecraft, en enregistrant les données de ton labyrinthe dans un fichier externe ! Mais avant tout, tu dois définir la manière dont tu vas organiser tes données dans le fichier.

Copyright © 2016 Eyrolles.

Tu peux regarder une vidéo didactique sur la création d'un labyrinthe en te rendant sur le site Internet du livre www.editions-eyrolles.com/dl/14292. Choisis la vidéo « Adventure 6 » (en anglais).



Les labyrinthes que tu vas concevoir ici sont en 3D, car tu vas les construire dans un monde en 3D. Mais en réalité ce sont des labyrinthes en 2D construits à partir de blocs en 3D, car ils ne comportent qu'une seule couche. En clair, tu ne devras enregistrer que les données x et z de chaque bloc du labyrinthe dans ton fichier, pour que celui-ci soit rectangulaire.

Tu devras également décider de l'agencement de tes blocs dans le fichier de données. Enfin, comme tu n'utiliseras que des murs et de l'air, le chiffre 1 représentera les murs et le 0 représentera l'air. Tu pourras toujours modifier plus tard les types de blocs servant à la construction d'un mur à l'intérieur de ton programme Python.

Comprendre les fichiers CSV

Pour ce programme, tu vas utiliser un fichier texte un peu spécial, qui s'appelle **CSV**. Il te permettra de définir tes labyrinthes sous la forme de fichiers au sein du système de fichiers de ton ordinateur.

Un fichier **CSV** est un fichier qui contient des valeurs séparées par des virgules. On y écrit des valeurs sous forme de texte et on les sépare par des virgules. Dans un fichier CSV, il est possible de créer un tableau ou une base de données simple. Chaque ligne du fichier représente une rangée du tableau ou une donnée de la base de données ; et le texte des lignes entre les virgules représente une colonne du tableau ou un champ. Certains fichiers CSV utilisent la première ligne du fichier pour enregistrer les titres des colonnes du tableau ou les noms des champs. Toutes les applications de tableur et de base de données les plus connues sont compatibles avec les fichiers CSV.



Le format CSV est très simple et son utilisation est très répandue. Voici un exemple de texte CSV issu d'un tableau de base de données rassemblant des informations sur des joueurs de Minecraft. Dans ce fichier CSV, la première rangée (ou ligne) est réservée aux noms des champs et toutes les autres lignes comportent des données séparées par des virgules :

```
Nom,Pseudo,Spécialité
David,w_geek,Le code en Python
Roma,physics_gurl,La conception de grands bâtiments
Ryan,mr_teck,Les robots Minecraft
Craig,rrrrrrrr,Expert TNT
```

Copyright © 2016 Eyrolles.

Ce fichier CSV contient donc une rangée avec les titres `Nom`, `Pseudo` et `Spécialité`. Il contient également quatre rangées de données, comportant chacune trois champs.



Lorsqu'on conçoit des structures complexes en informatique, il est parfois utile d'avoir recours à d'autres outils. En tant qu'ingénieur informatique, j'utilise régulièrement les tableurs pour concevoir et représenter des données. Ils constituent un formidable moyen de créer des tableaux de données, car on peut les exporter au format CSV et les charger ensuite dans d'autres applications. Pour t'entraîner, essaye de concevoir un labyrinthe dans ton application de tableur favorite. Crée des colonnes étroites et utilise les formules pour remplir tes cellules en blanc lorsque tu saisis le chiffre `0` et en jaune lorsque tu saisis le chiffre `1`. Une fois que tu as visualisé ton labyrinthe de cette manière, exporte-le au format CSV et nomme-le `labyrinthe.csv`, puis exécute ton programme et amuse-toi à te perdre avec tes amis !

Dans le fichier de données de ton labyrinthe, tu n'as pas besoin d'une rangée de titres, car chaque colonne du tableau comporte le même type de données, à savoir le chiffre 0 pour représenter le vide et le chiffre 1 pour représenter un morceau de mur. Tu peux télécharger le fichier du labyrinthe depuis le site Internet du livre, mais voici son contenu si tu préfères le créer toi-même.

1. Crée un nouveau fichier en cliquant sur `File>New File`. Clique sur `File>Save As` et nomme ton fichier `labyrinthe1.csv`.
2. Saisis les lignes suivantes en les recopiant méticuleusement et fais attention à conserver le même nombre de colonnes sur chaque ligne. Si tu observes ces données de près, tu devrais déjà pouvoir te faire une idée de la structure du labyrinthe ! Au total, il contient 16 lignes, avec 16 chiffres par ligne, alors tu peux t'aider d'un crayon pour vérifier que tu prends bien toutes les données en compte :

```
1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1
1,1,1,1,1,1,1,1,1,0,1,0,1,1,0,1
1,0,0,1,0,0,0,0,1,0,1,0,1,0,0,1
1,1,0,1,0,1,1,0,0,0,0,0,1,0,1,1
1,1,0,1,0,1,1,1,1,1,1,1,1,0,1,1
1,1,0,0,0,1,1,1,1,1,1,0,0,0,1,1
1,1,1,1,1,1,1,0,0,0,0,1,1,1,1,1
1,0,0,0,0,1,0,0,0,0,0,0,0,0,0,1
1,0,1,1,1,1,0,0,0,0,0,1,1,1,1,1
1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1
1,0,1,1,1,1,1,1,1,1,0,1,1,1,1,1
1,0,1,0,0,0,0,0,0,1,0,0,0,0,0,1
1,0,1,0,1,1,1,1,1,0,1,1,1,1,1,0,1
1,0,0,0,0,0,0,0,0,1,0,0,0,0,0,1
1,1,1,1,1,1,1,1,1,1,1,0,1,1,1,1,1
```

Copyright © 2016 Eyrolles.

3. Enregistre ce fichier. Tu l'utiliseras lorsque tu auras écrit le programme Python te permettant de lire et de traiter ces données.

Construire un labyrinthe

Maintenant que tu disposes d'un fichier de données détaillant la structure de ton labyrinthe, il ne te reste plus qu'à rédiger le programme Python qui va lire ces données et construire ton labyrinthe dans Minecraft avec des blocs.

1. Ouvre un nouveau fichier en cliquant sur `File>New File`, puis enregistre-le en cliquant sur `File>Save As` et nomme-le `constructionCSV.py`.

2. Importe les modules dont tu as besoin :

```
import mcpi.minecraft as minecraft
import mcpi.block as block
```

3. Et connecte-toi au jeu Minecraft :

```
mc = minecraft.Minecraft.create()
```

4. Définis quelques constantes. La constante `ESPACE` est le type de bloc que tu vas utiliser entre les murs. Normalement, tu y mettras de l'air, mais tu peux aussi essayer d'y inclure d'autres types de blocs pour rendre ton labyrinthe encore plus intéressant. La constante `MUR` est le type de blocs que tu vas utiliser pour créer les murs de ton labyrinthe. Tu dois donc faire attention à ce que tu choisis, car certains blocs ne permettent pas de bâtir des murs solides. Quant à la constante `SOL`, elle sert de base à ton labyrinthe :

```
ESPACE = block.AIR.id
MUR = block.GOLD_BLOCK.id
SOL = block.GRASS.id
```

N'utilise pas le `SABLE` pour le sol de ton labyrinthe. Si jamais tu poses du sable comme fondation, il se peut que le sol se dérobe sous les pieds de ton joueur en fonction du terrain sur lequel il se trouve !



5. Ouvre le fichier contenant les données de ton labyrinthe. Ici, tu vas utiliser la constante `NOMDEFICHIER`, pour te faciliter la tâche lorsque tu modifieras plus tard le nom du fichier afin de lire des données de labyrinthe différentes :

```
NOMDEFICHIER = "labyrinthe1.csv"
f = open(NOMDEFICHIER, "r")
```

6. Localise ton joueur et calcule les coordonnées de l'angle inférieur de ton labyrinthe. Ajoute 1 aux coordonnées x et z de ton joueur afin d'éviter de construire le

Copyright © 2016 Eyrolles.

labyrinthe sur sa tête. Tu peux également t'amuser à construire le labyrinthe dans le ciel en modifiant la coordonnée y :

```
pos = mc.player.getTilePos()
ORIGINE_X = pos.x+1
ORIGINE_Y = pos.y
ORIGINE_Z = pos.z+1
```

7. Comme la coordonnée z variera à la fin de chaque ligne de données, commence par celle du point de départ de ton labyrinthe. Elle changera légèrement plus tard dans le programme.

```
z = ORIGINE_Z
```

8. Tu vas maintenant faire tourner une boucle sur toutes les lignes du fichier. Pour cela, lis attentivement l'encadré « Percer les secrets du code », qui t'explique toutes les étapes de la fonction `readlines()`. La boucle `for` te permet de lire chaque ligne du fichier une à une. À chaque tour de boucle, la variable `line` contient la donnée de la ligne qui vient d'être lue :

```
for line in f.readlines():
```

9. Tu vas maintenant diviser la ligne en plusieurs parties, là où sont placées les virgules. Pour en savoir plus sur la fonction `split()`, consulte l'encadré intitulé « Percer les secrets du code ». Souviens-toi que les lignes qui font partie du corps de la boucle `for` doivent être indentées d'un cadratin :

```
    data = line.split(",")
```

10. Fais bien attention à cette étape, car tu vas introduire une autre boucle `for`. C'est ce qu'on appelle une **boucle imbriquée**, car une boucle `for` est imbriquée dans une autre `for`. Dans ton programme, tu vas devoir réinitialiser la coordonnée x au début de chaque nouvelle rangée du labyrinthe. C'est pourquoi, tu dois le faire en dehors de la boucle `for` qui dessine une rangée entière :

```
    x = ORIGINE_X
    for cell in data:
```

11. Tous les chiffres que le programme va lire dans la ligne seront considérés comme du texte (une chaîne de caractères), donc tu devras placer des guillemets droits " autour de chacun pour que ton programme puisse fonctionner. L'instruction `if/else` te permet de construire soit un espace soit un mur en fonction du chiffre que le programme aura lu dans le fichier CSV. Pour rappel, le 0 correspond à un espace et toute autre donnée conduit à la construction d'un mur. Fais bien attention à l'indentation dans ce passage : l'instruction `if` est indentée deux fois, car elle fait partie de la boucle `for cell in data`, qui fait elle-même partie de la boucle `for line in f.readlines()`. La variable `b` est quant à elle très utile, car elle permet de raccourcir et de simplifier le programme. (Essaye de réécrire ce passage en omettant volontairement la variable `b` et tu verras ce qu'il se passe !)

Copyright © 2016 Eyrolles.

```
            if cell == "0":
                b = ESPACE
            else:
                b = MUR
            mc.setBlock(x, ORIGINE_Y, z, b)
            mc.setBlock(x,ORIGINE_Y+1, z, b)
            mc.setBlock(x, ORIGINE_Y-1, z, SOL)
```

12. Mets la coordonnée x à jour à la fin de la boucle `for cell`. Cette ligne doit être indentée, car elle s'aligne sur l'instruction précédente `mc.setBlock()` et elle fait partie du corps de la boucle `for cell in data`.

```
            x = x + 1
```

13. Mets la coordonnée z à jour à la fin de la boucle qui traite chaque rangée de données. Ne l'indente que d'un niveau, car elle fait partie de la boucle `for line in f.readlines()`.

```
        z = z + 1
```

Enregistre ton programme et vérifie bien tous tes niveaux d'indentation. Si jamais tu te trompes, le programme ne fonctionnera pas comme prévu et tu obtiendras un labyrinthe vraiment très étrange !

Exécute ton programme et tiens-toi prêt à voir apparaître un superbe labyrinthe sous tes yeux. Parcoure ses allées et essaye d'en trouver la sortie sans briser les murs et sans t'envoler dans les airs. Sur la figure 6-2, tu peux voir à quoi il ressemble de l'intérieur.



FIGURE 6-2 Vue de l'intérieur du labyrinthe dans Minecraft

Copyright © 2016 Eyrolles.

Si tu es égaré et que tu ne trouves pas la solution, tu auras toujours la possibilité de tricher un peu et de t'envoler pour voir ton labyrinthe d'en haut, comme sur la figure 6-3.



FIGURE 6-3 Vue aérienne du labyrinthe dans Minecraft



Une **boucle imbriquée** est une boucle à l'intérieur d'une autre boucle. On appellera la première une « boucle externe », et la deuxième une « boucle interne ». En Python, on imbrique une boucle dans une autre en l'indentant d'un niveau supplémentaire. On peut imbriquer plusieurs boucles les unes dans les autres autant de fois qu'on le souhaite.



Si tu laisses des lignes vides à la fin de ton fichier de données, elle seront lues et interprétées par ton programme Python. Heureusement, cela n'aura aucune incidence sur ton programme, mais il se peut que cela provoque la création de blocs en trop à la fin de ton labyrinthe.

PERCER LES SECRETS DU CODE

Dans les chapitres précédents, tu as appris à utiliser les listes Python et tu sais désormais qu'elles peuvent regrouper une série d'objets à des emplacements numérotés. Plusieurs fonctions inhérentes au langage Python fournissent

Copyright © 2016 Eyrolles.

des données sous forme de liste, comme tu viens de le voir dans le constructeur de labyrinthe.

Cette ligne de code en est l'exemple :

```
for line in f.readlines():
```

Souviens-toi que *f* est un descripteur de fichier, c'est-à-dire quelque chose dont tu peux t'emparer pour accéder à un fichier ouvert. Le type `file` dans Python possède une fonction intégrée qui s'appelle `readlines()`. Elle permet simplement de lire chaque ligne du fichier et de les accoler une à une à la fin de la liste. Tu dois également te souvenir, grâce aux chapitres précédents, que lorsqu'on inclut une liste dans une boucle `for`, celle-ci passe en revue tous ses objets.

Tout ce que la boucle `for` fait, c'est qu'elle lit toutes les lignes d'un fichier qui se trouvent au sein d'une liste et qu'elle les passe en revue l'une après l'autre. À chaque tour de boucle, la variable `line` (c'est-à-dire la variable de contrôle de boucle) s'empare de la ligne de données suivante.

Par exemple, si ton fichier contient trois lignes, comme ceci :

```
ligne un
ligne deux
ligne trois
```

la fonction `f.readlines()` renverra une liste ressemblant à ceci :

```
['ligne un', 'ligne deux', 'ligne trois']
```

Par ailleurs, tu as déjà utilisé une liste dans une autre partie de ton script, mais peut-être que tu ne t'en es pas aperçu. C'était ici :

```
data = line.split(",")
```

La variable `line` contient une chaîne de caractères qui correspond au texte d'une ligne complète du fichier CSV. Toutes les variables contenant une chaîne de caractères comportent une fonction intégrée qui s'appelle `split()`. Cette fonction divise la chaîne de caractères pour en faire une liste. Les caractères `","` entre parenthèses indiquent à la fonction `split` l'endroit où la séparation doit s'opérer.

Imagine que tu disposes d'une variable `line` comportant trois mots séparés par une virgule, comme ceci :

```
line = "un,deux,trois"
data = line.split(",")
```

Ta variable `data` comportera alors une liste de trois objets, ce qui donnera le résultat suivant :

```
['un','deux','trois']
```

Copyright © 2016 Eyrolles.

DÉFI



Conçois ton propre labyrinthe dans un autre fichier CSV en dessinant beaucoup d'allées sinueuses, d'impasses et de spirales pour rendre la tâche encore plus ardue. Mets tes amis au défi de retrouver la sortie. Tu peux t'amuser à cacher des trésors çà et là (en utilisant par exemple un `DIAMOND_BLOCK`, c'est-à-dire un bloc de diamant) ainsi qu'à la sortie du labyrinthe pour que tes amis puissent prouver qu'ils en ont exploré toutes les allées. Tu peux même t'amuser à construire un labyrinthe gigantesque qui recouvre tout le monde de Minecraft ! Trouveras-tu le moyen d'enregistrer la position des trésors dans le fichier du labyrinthe ?



Lorsque j'ai conçu les programmes de ce chapitre, j'ai dû apporter de légères modifications au programme du labyrinthe, car il était parfois trop difficile à résoudre. Par exemple, que se passe-t-il si tu retires la ligne de code qui pose les fondations du labyrinthe ? Essaye de construire les murs en utilisant d'autres types de blocs, tels que le `CACTUS` ou l'`EAU` (`WATER`) et observe ce qu'il se passe. J'ai dû tester le programme à plusieurs reprises avant de trouver les types de blocs qui me convenaient pour la construction des murs et du sol !

DÉFI



Une erreur s'est glissée dans le programme et c'est à toi de la retrouver. Même si une ligne se termine par un 0 dans le programme du labyrinthe, il construira un mur en or à la place. D'après toi, quelle en est la raison ? Essaye de corriger ce défaut par toi-même. Indice : tu as déjà résolu un problème similaire dans le programme `astuceChat.py`.



Dans ce chapitre, tu as utilisé les fonctions `readlines()` et `split()` pour traiter des fichiers CSV. Python est un langage de programmation très vaste et déjà très reconnu, qui possède de nombreuses fonctions intégrées. Tu peux t'intéresser à son module intégré d'importation de CSV en cliquant sur `import csv`, car c'est une méthode encore plus puissante que celle que nous avons utilisée ici. En ce qui me concerne, je préfère développer mes programmes pas à pas, parce que cela m'aide à comprendre comment ils fonctionnent. C'est pourquoi j'ai utilisé les fonctions `readlines()` et `split()` dans ce livre.

Copyright © 2016 Eyrolles.

Construire une imprimante de blocs en 3D

On s'est bien amusé à créer des labyrinthes, mais maintenant que tu sais utiliser des fichiers de données, bien d'autres possibilités vont s'ouvrir à toi ! Nous avons construit une structure avec deux types de blocs sur une seule couche, mais il serait dommage de se limiter à cela ! Pour créer ton imprimante et scanneur 3D, tu vas t'appuyer sur ton programme de construction de labyrinthe, le but étant de pouvoir créer des répliques d'arbres, de maisons et de bien d'autres objets encore, en seulement quelques clics ! En réalité, il s'agit d'un constructeur de blocs, mais j'aime bien l'idée de l'imprimante 3D. Car lorsque le programme se met en route, tu peux voir les structures se construire rangée par rangée, comme le papier qui sort d'une imprimante ou les objets qu'une imprimante 3D construit couche après couche.

Tu commences à te débrouiller plutôt bien en programmation et chaque nouveau script que tu écris est toujours plus compliqué que les précédents. Comme dans le chapitre précédent, où tu as rédigé ton programme à partir de fonctions, tu vas développer ce programme pas à pas.



La technique qui consiste à transformer les différentes fonctionnalités d'un programme en fonctions et à les rassembler ensuite dans un script plus grand est très courante chez les développeurs professionnels. Elle part du principe qu'un grand programme n'est qu'une suite de petits programmes. Si la structure de ton grand programme est correcte dès le départ, tu pourras approfondir tes fonctions pas à pas tout en développant et testant ton programme. En général, j'aime bien me créer des petites démos simples et rapides pour pouvoir montrer aux curieux ce que mon programme est capable de réaliser.

Fabriquer un petit objet de test pour imprimer en 3D

Dans ton programme de labyrinthe, tu as enregistré des données dans un fichier CSV dans lequel chaque ligne, ou rangée, correspond à une ligne de blocs dans Minecraft. Chaque rangée comporte elle-même des colonnes (séparées par des virgules) qui correspondent à un bloc de la ligne. Il s'agit là d'une structure en 2D, car elle pose un bloc uniquement sur les coordonnées x et z du rectangle où tu as choisi de construire ton labyrinthe. Même s'il apparaît dans le monde de Minecraft et qu'il est fait de cubes, le labyrinthe que tu as créé est en réalité en 2D, car il ne comporte qu'une seule couche de blocs.

Pour construire des formes en 3D, tu vas devoir ajouter la dimension y à ton programme. Si tu conçois qu'un objet en 3D est une succession de couches, le programme que tu vas développer n'est pas tellement plus compliqué que celui que tu as écrit pour bâtir ton labyrinthe. Pour créer un cube dont les faces mesurent cinq blocs de longueur et cinq de largeur, il te suffit d'enregistrer cinq couches d'informations.

Copyright © 2016 Eyrolles.

Seulement voilà, ce n'est pas aussi simple que cela. Ton programme Python ne connaît pas à l'avance le nombre de lignes que le fichier contient. Il pourrait éventuellement en faire une estimation approximative, mais il serait préférable de concevoir un format de fichier CSV assez flexible pour que le programme puisse appréhender des objets de tailles différentes.

Pour résoudre ce problème, tu vas ajouter ce que l'on appelle des **métadonnées** à la première ligne du fichier, qui détaillent les dimensions de l'objet que tu vas construire.



Les **métadonnées** sont des données sur des données. Si ton fichier de données contient des types de blocs servant à construire des objets, tu peux par exemple apporter des précisions sur les dimensions des objets, leur nom ou le nom de leur créateur en guise de métadonnées, c'est-à-dire les informations qui concernent ces données. De même que les métadonnées d'une photo enregistrée sur ton ordinateur définissent la date et l'heure à laquelle elle a été prise ou même le nom de l'appareil photo, les métadonnées de ton imprimante 3D te permettront d'obtenir tout un tas d'informations utiles sur tes objets en 3D.

Tu vas maintenant bâtir une petite grotte en 3D, que tu peux voir à la figure 6-4, pour t'exercer à tester les capacités d'impression en 3D de ton programme. Tu peux télécharger ce petit objet depuis le site Internet des éditions Eyrolles (`www.editions-eyrolles.com/dl/14292`) ou saisir les instructions qui vont suivre, comme tu l'as fait plus tôt avec le labyrinthe.



FIGURE 6-4 Tu as bâti une petite grotte à l'aide d'un fichier CSV.

Copyright © 2016 Eyrolles.

Fais bien attention à laisser des lignes vides entre les différentes couches de ton objet. Elles te permettent de voir où commence chacune d'elles. De plus, le programme Python que tu écris s'attend à tomber sur des lignes vides, alors si tu les omets il ne fonctionnera pas correctement.



N'hésite pas à utiliser les fonctions copier-coller dans les étapes suivantes pour éviter de saisir trop de code à la main et ainsi limiter les risques d'erreur.



1. Ouvre un nouveau fichier en cliquant sur `File>New File` et enregistre-le en le nommant `objet1.csv`.

2. Saisis la première ligne de métadonnées relative aux dimensions de l'objet. Ton objet de test aura les dimensions d'un cube de 5 × 5 blocs, le x correspondant à la largeur, le y à la hauteur et le z à la profondeur :

```
5,5,5
```

3. Saisis ensuite la première couche d'épaisseur de ton objet test, en t'assurant de laisser une ligne vide avant la première ligne de données. Ta première couche de blocs correspond aux fondations de l'objet ; tu dois donc remplir toute la surface en saisissant à chaque fois le chiffre 1 qui correspond à l'emplacement de chacun des blocs :

```
1,1,1,1,1
1,1,1,1,1
1,1,1,1,1
1,1,1,1,1
1,1,1,1,1
```

4. Le cube que tu construis est vide à l'intérieur et dispose d'une ouverture, tu dois donc dessiner trois couches avec ce format-là. N'oublie pas de laisser une ligne vide avant chacune des trois séries de chiffres :

```
1,1,1,1,1
0,0,0,0,1
0,0,0,0,1
0,0,0,0,1
1,1,1,1,1
```

Copyright © 2016 Eyrolles.

5. Copie cette série de chiffres et colle-la deux fois à la suite en t'assurant de laisser une ligne vide avant chacune.

6. Puis termine en recouvrant ton objet d'une couche solide. Pour cela, copie les rangées de données de l'étape 3, en laissant une ligne vide devant sa première rangée de chiffres.

Enregistre ton fichier sans l'exécuter. De toute façon, comme il ne s'agit pas d'un fichier Python, tu ne peux pas l'exécuter. En revanche, pour l'utiliser tu devras créer un programme Python et c'est ce que nous allons faire dans la prochaine étape de ce chapitre.

Rédiger le code de l'imprimante 3D

Maintenant que tu as fini d'écrire tes données de test, il est temps que tu te mettes à l'écriture du programme de l'imprimante 3D qui te permettra de construire la grotte dans le monde de Minecraft. Ce script ressemble beaucoup au programme du labyrinthe, à cela près qu'il contient trois boucles imbriquées, c'est-à-dire une boucle pour chacune des coordonnées x, y et z.



Ce programme est sans doute le plus élaboré que tu as écrit depuis le début de ce livre. Prends le temps de l'écrire étape par étape pour le réussir du premier coup et tu seras d'autant plus satisfait du résultat ! Si tu sens que tu bloques à un certain point, commence par vérifier avec attention toutes tes indentations. Et si tu es toujours bloqué après cela, n'oublie pas que tu as la possibilité de télécharger tous les programmes de ce livre sur le site du livre à l'adresse suivante : `www.editions-eyrolles.com/dl/14292`.

1. Ouvre un nouveau fichier en cliquant sur `File>New File`, puis enregistre-le en cliquant sur `File>Save As` dans le menu et nomme-le `impression3D.py`.

2. Importe les modules dont tu as besoin :

```
import mcpi.minecraft as minecraft
import mcpi.block as block
```

3. Connecte-toi au jeu Minecraft :

```
mc = minecraft.Minecraft.create()
```

4. Crée une constante en lui attribuant le nom de ton fichier de données. Cela te facilitera la tâche pour plus tard lorsque tu liras plusieurs fichiers contenant plusieurs objets :

```
NOMDEFICHIER = "objet1.csv"
```

Copyright © 2016 Eyrolles.

5. Définis une fonction `impression3D()`. Cette fonction te servira plus tard pour la réalisation de ton projet final, donc veille à la nommer correctement :

```
def impression3D(nomdefichier, originex, originey, originez):
```

6. Comme tu l'as fait pour ton programme de construction de labyrinthe, ouvre le fichier de données en mode lecture et lis toutes les lignes dans une liste Python :

```
    f = open(nomdefichier, "r")
    lignes = f.readlines()
```

7. Le premier élément de ta liste se trouve à l'index 0. Il s'agit de la première ligne de ton fichier, celle qui regroupe les métadonnées. Divise cette ligne en plusieurs parties en utilisant la fonction `split()`. Enregistre ensuite ces trois nombres dans les variables `dimensionsx`, `dimensionsy` et `dimensionsz`. Dans cette étape, tu dois utiliser la fonction `int()`, car lorsque tu lis des lignes dans un fichier, elles sont interprétées comme des chaînes de caractères. Or, tu as besoin de les lire comme des nombres, car, plus tard, tu les utiliseras pour effectuer des calculs.

```
    coords = lignes[0].split(",")
    dimensionsx = int(coords[0])
    dimensionsy = int(coords[1])
    dimensionsz = int(coords[2])
```

8. Définis une variable `idxligne` pour enregistrer l'index de la ligne dans la liste `lignes[]` que tu utilises. Comme ton programme va parcourir la liste `lignes` en lisant les données correspondant aux différentes couches de ton objet 3D, tu ne peux pas utiliser cette variable comme variable de contrôle de boucle :

```
    idxligne = 1
```

9. La première boucle `for` parcourt chacune des couches verticales des données lues dans le fichier. Les premières lignes du fichier correspondent à la couche qui recouvre la base de l'objet, c'est-à-dire au niveau le plus bas des coordonnées y. Pour que les étapes du processus d'impression soient visibles dans Minecraft, tu peux ajouter une fonction `postToChat()` à cet endroit :

```
    for y in range(dimensionsy)
        mc.postToChat(str(y))
```

10. Ignore les lignes vides entre chaque couche dans ton fichier en ajoutant le chiffre 1 à la variable `idxligne`. Comme tu le sais, ces lignes vides servent uniquement à rendre la lecture du fichier plus agréable, ton programme doit les ignorer.

```
        idxligne = idxligne + 1
```

11. Commence une boucle imbriquée `for` qui lira la prochaine ligne et la divisera en plusieurs parties après chaque virgule. Fais bien attention à l'indentation ici ; la boucle `for` doit être indentée de deux cadratins (un pour la fonction et un pour

Copyright © 2016 Eyrolles.

la boucle `for y`) et le code à l'intérieur de la boucle `for x` doit être indenté de trois cadratins.

```
        for in range(dimensionsx):
            ligne = lignes[idxligne]
            idxligne = idxligne + 1
            donnee = ligne.split(",")
```

12. Maintenant, commence la rédaction de ta troisième boucle `for` qui va parcourir chacun des blocs de la ligne qui vient d'être traitée et les construira dans Minecraft. La boucle `for z` est indentée de trois cadratins et le corps de la boucle doit être indenté quatre fois, donc sois bien vigilant en rédigeant ces lignes, sinon tes objets auront une forme bien étrange !

```
        for z in range(dimensionsz):
            blockid = int(donnee[z])
            mc.setBlock(originex+x, originey+y, originez+z, ↩
blockid)
```

13. Enfin, saisis les lignes du programme principal, qui ne doivent comporter aucune indentation. Elles vont lire la position de ton joueur et vont ensuite commander à la fonction `impression3D()` d'imprimer ton objet en 3D juste à côté de lui :

```
pos = mc.player.getTilePos()
impression3D(NOMDEFICHIER, pos.x+1, pos.y, pos.z+1)
```

Sauvegarde ton programme et exécute-le. Que vois-tu ? Promène-toi un peu dans le monde de Minecraft et exécute le programme. À chaque fois, tu devrais voir apparaître une grotte en pierre juste à côté de ton joueur. Sur la figure 6-5, tu peux voir comme il est simple de construire plusieurs grottes en pierre en un claquement de doigts.

DÉFI



Crée d'autres fichiers CSV contenant des objets 3D, modifie la constante `NOMDEFICHIER` dans le script `impression3D.py` et exécute-le pour disséminer des objets un peu partout dans le monde de Minecraft. Comme tes objets sont de plus en plus sophistiqués, tu peux t'amuser à imaginer plusieurs techniques de conception avant d'enregistrer les chiffres dans le fichier. Tu peux par exemple les dessiner dans un tableur ou encore te procurer une grande boîte de cubes en plastique pour les construire en dur puis les reprendre couche après couche pour indiquer les numéros de bloc dans un fichier CSV.

Copyright © 2016 Eyrolles.



FIGURE 6-5 Impression 3D de plusieurs grottes en pierre dans Minecraft

Construire un scanneur de blocs en 3D

Ton imprimante 3D est déjà très puissante. Tu peux créer une grande bibliothèque de fichiers CSV pour construire tout un tas d'objets différents. Et dès que tu as besoin d'un objet en particulier, il te suffit d'exécuter ton programme `impression3D.py` avec la constante `NOMDEFICHIER` qui correspond à l'objet de ton choix pour le faire apparaître dans le monde de Minecraft.

En revanche, si tu souhaitais construire des objets plus complexes et plus grands, il te faudrait saisir tous les chiffres dans le fichier CSV à la main et cette tâche s'avérerait bien fastidieuse. Heureusement, le jeu Minecraft est idéal pour construire des objets complexes. Alors, pourquoi ne pas lui demander de créer lui-même ces fichiers CSV pour toi ? Dans ce chapitre, tu vas apprendre à créer une copie identique d'un arbre de ton choix pour la reproduire un peu partout dans le monde de Minecraft. Présentée ainsi, l'opération semble difficile à réaliser, mais il n'en est rien ! Car scanner en 3D, c'est un peu comme imprimer en 3D à l'envers.

1. Ouvre un nouveau fichier en cliquant sur `File>New File`, puis enregistre-le en cliquant sur `File>Save As` et nomme-le `scannage3D.py`.
2. Importe les modules dont tu as besoin :

```
import mcpi.minecraft as minecraft
import mcpi.block as block
```

Copyright © 2016 Eyrolles.

3. Connecte-toi au jeu Minecraft :

```
mc = minecraft.Minecraft.create()
```

4. Crée des constantes avec les noms des fichiers CSV et la taille de la zone que tu souhaites scanner :

```
NOMDEFICHIER = "arbre.csv"
DIMENSIONSX = 5
DIMENSIONSY = 5
DIMENSIONSZ = 5
```

5. Définis une fonction `scannage3D()`. Tu l'utiliseras plus tard dans un programme, donc assure-toi de la nommer correctement :

```
def scannage3D(nomdefichier, originex, originey, originez)
```

6. Dans cette fonction, ouvre le fichier de données que tu vas créer, mais cette fois-ci en activant le mode de modification, c'est-à-dire en ajoutant `"w"` à l'intérieur de la fonction `open()` :

```
    f = open(nomdefichier, "w")
```

7. La première ligne du fichier doit contenir les métadonnées, tu dois donc y indiquer les valeurs des coordonnées x, y et z. Si tu veux connaître la signification et l'utilité du suffixe `\n` à la fin de la ligne de code, rends-toi dans l'encadré « Percer les secrets du code ».

```
    f.write(str(DIMENSIONSX) + "," + str(DIMENSIONSY) + "," ↩
+ str(DIMENSIONSZ) + "\n")
```

8. Le programme est l'exact opposé du programme d'impression et possède lui aussi trois boucles imbriquées. Je l'ai affiché dans son ensemble afin que l'indentation ressorte mieux et que tu puisses la retranscrire correctement. Pour en savoir plus sur la manière dont les lignes séparées par des virgules sont créées, lis l'encadré « Percer les secrets du code » :

```
    for y in range(DIMENSIONSY):
        f.write("\n")
        for x in range(DIMENSIONSX):
            line = ""
            for z in range(DIMENSIONSZ):
                blockid = mc.getBlock(originex+x, ↩
originey+y, originez+z)
                if line != "":
                    line = line + ","
                line = line + str(blockid)
            f.write(line + "\n")
        f.close()
```

Copyright © 2016 Eyrolles.

9. Pour finir, le programme principal ne contient aucune indentation. Ces dernières lignes détectent ta position, calculent un espace cubique autour de ton joueur et demandent ensuite à la fonction `scannage3D()` de scanner tout cet espace pour l'inclure dans ton fichier CSV.

```
pos = mc.player.getTilePos()
scannage3D(NOMDEFICHIER, pos.x-(DIMENSIONSX/2), pos.y, ↩
pos.z-(DIMENSIONSZ/2))
```

Enregistre ton programme. Vérifie à nouveau que tes lignes de code sont bien indentées avant de l'exécuter.

Maintenant, cours vers un arbre et appuie-toi contre son tronc, puis lance ton programme `scannage3D.py`. Que se passe-t-il ?

Ne vois-tu rien se produire ? C'est tout à fait normal, souviens-toi que ce programme scanne un objet à l'intérieur d'un fichier CSV, donc tu dois regarder dans ton fichier `arbre.csv` pour voir quels chiffres il contient désormais. Ouvre le fichier `arbre.csv` en cliquant sur `File>Open` dans l'éditeur. La figure 6-6 montre une section du fichier `arbre.csv` que j'ai capturée après avoir embrassé un arbre ! Comme l'espace de scannage que j'ai défini est assez petit, il se peut que tu ne scannes qu'une petite partie de l'arbre que tu as choisi. Tu peux remédier à cela en modifiant les constantes `DIMENSIONS` de ton script pour scanner une zone plus grande.

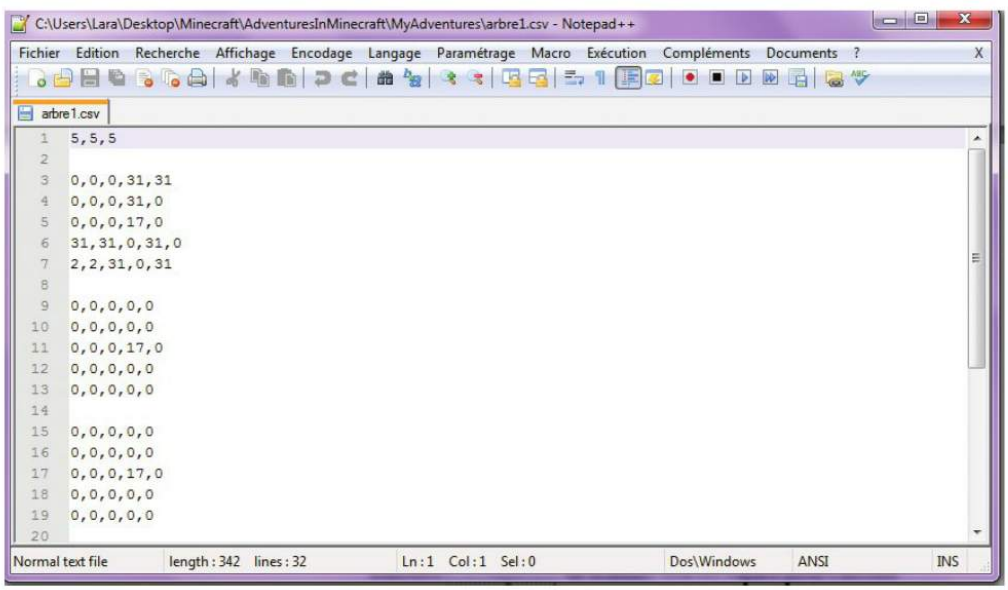


FIGURE 6-6 Contenu d'un fichier CSV après scannage d'un arbre

Copyright © 2016 Eyrolles.



N'oublie pas que la direction dans laquelle ton joueur se tourne ne correspond pas forcément à la position qui a été scannée, donc regarde bien tout autour de toi. Le scanneur 3D numérise une zone comprise entre la position de ton joueur et des valeurs de coordonnées plus élevées. Donc, si ton joueur se trouve à l'emplacement 0,0,0, le scanneur 3D numérisera une zone comprise entre ces coordonnées et des coordonnées plus élevées, par exemple 4,4,4. Réfère-toi au schéma du chapitre 2, qui explique le principe des coordonnées et la direction que chacune d'entre elles représente.



Normalement, IDLE n'ouvre que les fichiers dotés de l'extension `.py` dans ses fenêtres `Open` et `Save`, mais tu peux aussi visualiser tes fichiers `.csv` en cliquant sur `Files of Type>All Files`.

DÉFI



Maintenant que tu disposes d'un scanneur et d'une imprimante 3D, modifie ton programme d'impression 3D pour utiliser une constante `NOMDEFICHIER` du fichier `arbre.csv`, puis promène-toi un peu partout dans Minecraft tout en exécutant le script `impression3D.py`. En faisant cela, tu devrais pouvoir parsemer des arbres sur ton chemin. Essaye d'imprimer des arbres dans le ciel et sous l'eau pour voir ce qui se passe. En combien de temps vas-tu parvenir à faire pousser toute une forêt ?

PERCER LES SECRETS DU CODE

Dans le programme `scannage3D.py`, tu as utilisé un caractère spécial au sein de la fonction `write()`. Cela mérite une petite explication :

```
f.write("\n")
```

Le caractère `\n` (appelé « backslash n ») est un caractère spécial non imprimable que Python te permet d'utiliser entre guillemets. Il signifie « passe à la prochaine ligne ». La lettre `n` est l'initiale de « newline » (en français, « nouvelle ligne »). Quant au backslash, la barre oblique inversée, il permet de distinguer cette lettre `n` d'une lettre n normale.

Copyright © 2016 Eyrolles.

Dans le script `scannage3D.py`, tu as conçu ton fichier de sorte à laisser une ligne vide entre chacune des couches de données de ton objet pour faciliter la lecture et la modification du fichier CSV par d'autres personnes. L'instruction `f.write("\n")` réalise exactement la même chose : elle insère une ligne vide dans ton fichier.

Plus loin dans le programme `scannage3D.py`, tu peux t'amuser à rédiger du code autour de la variable `ligne`. En voici les instructions les plus importantes :

```
for x in range(DIMENSIONSX):
    ligne = ""
    for z in range(DIMENSIONSZ):
        blockid = mc.getBlock(originex+x, originey+y, ↩
  originez+z)
        if ligne != "": # la ligne n'est pas vide
            ligne = ligne + ","
        ligne = ligne + str(blockid)
```

Le code souligné correspond à un modèle de codage très répandu, généralement utilisé pour créer des valeurs séparées par des virgules. Au début de la boucle `x`, on affecte une chaîne de caractères vide à la variable. Chaque fois que la boucle `z` produit une autre valeur, elle vérifie d'abord si la première ligne est vide et, dans le cas contraire, elle ajoute une virgule avant d'ajouter les données d'identité du bloc. L'instruction `if` est nécessaire, car elle permet d'éviter qu'une virgule apparaisse en début de ligne, avant le premier nombre, ce qui pourrait perturber la lecture du programme `impression3D.py` par la suite.

Construire une machine à dupliquer

Tu disposes maintenant de tous les scripts nécessaires pour développer ta propre machine à dupliquer en 3D, qui fera fureur chez tous tes amis ! Grâce à elle, tu pourras confectionner une salle à dupliquer dans le monde de Minecraft et créer tous les objets que tu veux. Tu pourras alors les conserver dans des fichiers, les réintroduire dans la salle pour les modifier ou les reproduire comme par magie un peu partout dans le monde. Enfin, tu pourras aussi t'évader de ce monde et faire disparaître la salle à dupliquer sans laisser aucune trace derrière toi.

À l'instar des chapitres précédents, tu vas utiliser les programmes que tu as créés et les assembler pour obtenir un programme plus grand. De même, comme ce programme possède plusieurs fonctionnalités, tu vas rédiger un menu pour qu'il soit plus facile à manipuler.

Copyright © 2016 Eyrolles.

Rédiger la structure du programme de la machine à dupliquer

La première chose à faire est de concevoir la structure du programme pour qu'il tienne en un seul morceau. Pour cela, tu vas commencer par rédiger des fonctions factices qui ne feront qu'afficher leur nom lorsque tu les appelleras. Tu les compléteras par la suite avec le contenu des fonctions de tes autres programmes. C'est exactement la même technique que tu as employée pour développer ton jeu de chasse au trésor dans le chapitre 4, t'en souviens-tu ?

1. Ouvre un nouveau fichier en cliquant sur `File>New File` et enregistre-le en le nommant `duplicateur.py`.

2. Importe les modules dont tu as besoin :

```
import mcpi.minecraft as minecraft
import mcpi.block as block
import glob
import time
import random
```

3. Connecte-toi au jeu Minecraft :

```
mc = minecraft.Minecraft.create()
```

4. Définis quelques constantes pour déterminer les dimensions de ta machine à dupliquer. Ne choisis pas des dimensions trop élevées, sinon ta machine à dupliquer tardera à scanner et à imprimer les objets. Pense également à définir la position par défaut de ta salle de reproduction :

```
DIMENSIONSX = 10
DIMENSIONSY = 10
DIMENSIONSZ = 10
sallex = 1
salley = 1
sallez = 1
```

5. Rédige quelques fonctions d'apparat correspondant aux fonctionnalités du programme. Tu les compléteras plus tard :

```
def constructionSalle(x, y, z):
    print("constructionSalle")

def demolitionSalle():
    print("démolitionSalle")

def nettoyageSalle():
    print("nettoyageSalle")

def repertoireFichiers():
```

Copyright © 2016 Eyrolles.

```
    print("repertoireFichiers")

def scannage3D(nomdefichier, originex, originey, originez)
    print("scannage3D")

def impression3D(nomdefichier, originex, originey, originez):
    print("impression3D")
```

6. La fonction factice du menu est assez spéciale, car elle renvoie une valeur à la fin de son exécution. Pour l'instant, tu peux faire apparaître une option aléatoire et utiliser le mot-clé **return** pour tester les versions précédentes de ton programme. Tu reviendras dessus plus tard pour le compléter. Si tu rédiges un menu factice, c'est pour pouvoir tester la structure de ton programme. Tu le modifieras plus tard pour en faire un vrai menu :

```
def menu():
    print("menu")
    time.sleep(1)
    return random.randint(1,7)
```

7. Rédige la boucle de jeu principale qui va te permettre d'afficher le menu et d'utiliser la fonction correspondant à cette fonctionnalité. Pour en savoir plus sur l'utilisation de la variable `anotherGo`, lis l'encadré « Percer les secrets du code » :

```
anotherGo = True
while anotherGo:
    choix = menu()
    if choix == 1:
        pos = mc.player.getTilePos()
        constructionSalle(pos.x, pos.y, pos.z)
    elif choix == 2:
        repertoireFichiers()
    elif choix == 3:
        nomdefichier = raw_input("nom du fichier ?")
        scannage3D (nomdefichier, sallex+1, salley+1, sallez+1)
    elif choix == 4:
        nomdefichier = raw_input("nom du fichier ?")
        impression3D(nomdefichier, sallex+1, salley+1, sallez+1)
    elif choix == 5:
        scannage3D("scantemp", sallex+1, salley+1, sallez+1)
        pos = mc.player.getTilePos()
        impression3D("scantemp", pos.x+1, pos.y, pos.z+1)
    elif choix == 6:
        nettoyageSalle()
    elif choix == 7:
        demolitionSalle()
    elif choix == 8:
        anotherGo = False
```

Copyright © 2016 Eyrolles.

Sauvegarde ton programme de test puis exécute-le. Que se passe-t-il ? Sur la figure 6-7, tu peux voir ce qui s'est affiché lorsque j'ai exécuté le programme sur mon ordinateur : il t'indique en effet les opérations qu'il effectue, c'est-à-dire qu'il choisit une entrée du menu de manière aléatoire, puis il exécute la fonction que cette entrée renferme. Comme les fonctions n'affichent que leur nom, c'est tout ce que tu peux voir pour l'instant. La suite de mots qui s'affichera sur ton écran sera différente de celle de la figure 6-7, car la fonction `menu()` choisit une entrée au hasard à chaque fois que ton programme y fait appel.

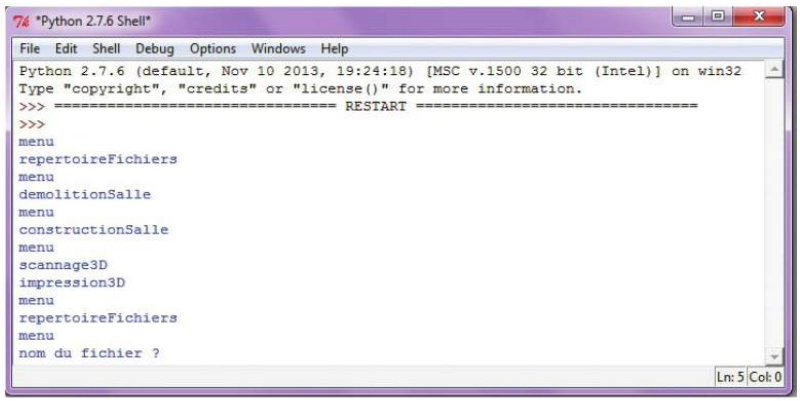


FIGURE 6-7 Ce qui s'affiche après l'exécution du programme de test



L'instruction **return** permet de renvoyer la valeur d'une fonction lorsqu'elle retourne dans le programme qui a appelé la fonction pour la première fois.

Par exemple, lorsque tu souhaites obtenir un nombre au hasard, tu appelles la fonction `random.randint()`. Cette fonction renvoie une valeur que tu peux alors stocker dans une variable, comme ceci : `a = random.randint(1,100)`. Le mot-clé `return` dans le langage Python te permet d'utiliser exactement la même technique de renvoi de valeurs au sein de tes propres fonctions.



La fonction `raw_input()` permet de lire une ligne de texte saisie à l'aide du clavier. Si tu saisis une chaîne de caractères entre les parenthèses, cette chaîne de caractères sera utilisée comme invite de commandes, donc l'instruction `nom = raw_input("Comment t'appelles -tu ?")` pose une question dont elle recevra également la réponse.

Lorsque tu utilises `raw_input()` pour lire du texte saisi au clavier, la fonction te renvoie toujours une chaîne de caractères. Si tu souhaites saisir un nombre (par exemple, dans ton menu), tu devras utiliser la fonction `int()` pour convertir la chaîne de caractères lue en nombre. Certains programmeurs Python préfèrent rédiger cette commande sur une seule et même ligne, comme ceci : `age = int(raw_input("Quel âge as-tu ?"))`.

Copyright © 2016 Eyrolles.

Tout au long de ce livre, tu as utilisé la version 2 du langage Python. Si tu changes d'ordinateur, il se peut que la version 3 y soit installée. Ces deux versions du langage diffèrent sur plusieurs points. Par exemple, dans la version 3, tu dois utiliser la fonction `input()` au lieu de la fonction `raw_input()` propre à la version 2. Dans ce livre, j'ai choisi d'utiliser la version 2 du langage Python, car l'API de Minecraft est conçue pour fonctionner uniquement avec celle-ci. Si tu souhaitais utiliser la version 3, tu serais obligé d'effectuer des modifications.



PERCER LES SECRETS DU CODE

Même si tu as déjà utilisé des variables en **booléen** dans les chapitres précédents, je te recommande vivement de t'arrêter quelques secondes pour comprendre la manière dont la variable `anotherGo` a été utilisée dans le programme `duplicateur.py`. En voici les lignes le plus importantes :

```
anotherGo = True
while anotherGo:
    choix = menu()
    if choix == 8:
        anotherGo = False
```

En programmation, la structure de cette boucle est assez commune : elle se répète au moins une fois, te demande si tu veux engager un autre tour et, en cas de réponse négative, met fin à la boucle. Il existe plusieurs manières d'arriver au même résultat en Python, notamment en utilisant une variable booléenne, qui constitue une option idéale, car elle permet de voir clairement comment le programme fonctionne.

Python dispose d'une instruction nommée `break`, qui permet de briser une boucle. Même si tu ne vas pas l'utiliser dans ce livre, tu peux toujours effectuer une recherche sur Internet et t'amuser à modifier ta boucle en remplaçant la variable booléenne par cette instruction `break`.

Une **variable booléenne** est une variable qui ne peut prendre qu'une valeur sur deux, soit `True` ou `False`. Elles tiennent leur nom du mathématicien George Boole, qui a beaucoup travaillé sur la logique formelle dans les années 1980, une discipline qui a permis de poser les fondements de l'informatique moderne. Tu peux en lire davantage sur Boole en te rendant sur cette page : `http://fr.wikipedia.org/wiki/George_Boole`.



Copyright © 2016 Eyrolles.

Afficher le menu

Ajouter un menu à ton programme peut s'avérer très utile s'il propose beaucoup d'options. Ce menu affiche toutes les options disponibles et effectue un tour de boucle, en attendant que tu saisisses le numéro de l'une d'elles. Si tu saisis un mauvais numéro, le menu s'affichera à nouveau en te demandant de réessayer.

1. Modifie la fonction du menu et remplace-la par les lignes de code suivantes (attention à l'indentation) :

```
def menu():
    while True:
        print("MENU DU DUPLICATEUR")
        print(" 1. CONSTRUIRE la salle de reproduction")
        print(" 2. AFFICHER les fichiers existants")
        print(" 3. SCANNER le contenu de la salle de ↩
reproduction dans un fichier")
        print(" 4. CHARGER le contenu d'un fichier dans ↩
la salle de reproduction")
        print(" 5. IMPRIMER le contenu de la salle de ↩
reproduction en face de Steve")
        print(" 6. NETTOYER la salle de reproduction")
        print(" 7. DEMOLIR la salle de reproduction")
        print(" 8. FERMER")
        choix = int(raw_input(«choix : «))
        if choix < 1 or choix > 8:
            print("Tu dois choisir un chiffre entre 1 et 8")
        else:
            return choix
```

Enregistre ton programme et exécute-le. Qu'est-ce qui change par rapport au menu factice ? Sur la figure 6-8, tu peux voir ce à quoi ressemble le vrai menu.

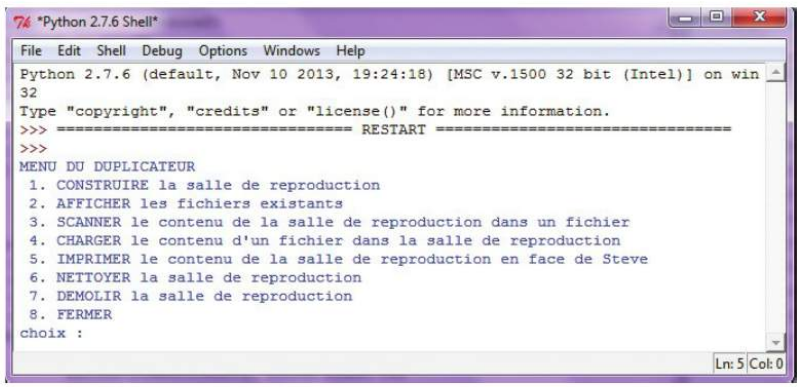


FIGURE 6-8 Le menu

Copyright © 2016 Eyrolles.

En posant la structure de ton programme, avec son menu et toutes ses fonctions factices, puis en le complétant en détail fonction par fonction et en le testant, tu travailles exactement comme le ferait un ingénieur professionnel qui développe ses propres applications informatiques. La meilleure méthode pour concevoir un programme consiste en effet à l'écrire pas à pas, en testant tes changements au fur et à mesure, jusqu'à ce que tu en viennes à bout. De cette manière, tu pourras faire une démonstration à quiconque te le demandera en un rien de temps !



Bâtir la salle de reproduction

La salle de reproduction sera en verre et sera ouverte d'un côté pour que ton joueur puisse y entrer et facilement créer ou supprimer des blocs à l'intérieur.

Pour cela, tu dois remplacer la fonction `constructionSalle()` par les lignes de code suivantes. Fais attention aux lignes longues qui se trouvent dans la fonction `setBlocks()`. Tu remarqueras cependant que je n'ai pas utilisé la flèche ↲ dans les lignes suivantes, car ici Python me permet de les diviser en plusieurs lignes (pour en savoir plus sur cette exception, tu peux lire l'encadré « Percer les secrets du code ») :

```
def constructionSalle(x, y, z):
    global sallex, salley, sallez
    sallex = x
    salley = y
    sallez = z
    mc.setBlocks(sallex, salley, sallez,
                 sallex+DIMENSIONSX+2, salley+DIMENSIONSY+2,
                 sallez+DIMENSIONSZ+2, block.GLASS.id)
    mc.setBlocks(sallex+1, salley+1, sallez,
                 sallex+DIMENSIONSX+1, salley+DIMENSIONSY+1,
                 sallez+DIMENSIONSZ+1, block.AIR.id)
```

Sauvegarde ton programme et exécute-le à nouveau. Essaye l'option 1 du menu pour t'assurer que ton programme te permet bien de construire la salle de reproduction. Déplace-toi dans Minecraft et choisis l'option 1 à nouveau, puis regarde ce qu'il se passe ! La figure 6-9 te montre la salle de reproduction telle que tu devrais la voir.

Copyright © 2016 Eyrolles.

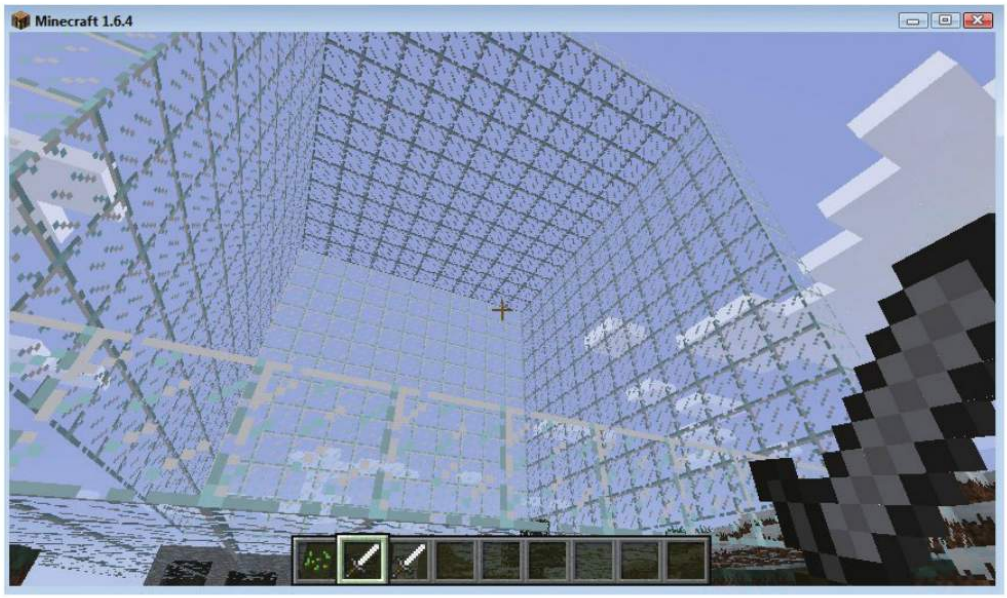


FIGURE 6-9 La salle de reproduction construite sous tes yeux

PERCER LES SECRETS DU CODE

Python utilise l'indentation (espace ou tabulation) au début des lignes de code pour distinguer les divers groupes d'instructions. Lorsque tu as utilisé des boucles, les mots-clés `if/else` ou même des fonctions, tu as indenté tes blocs de code en fonction. En programmation, cette pratique est peu commune et distingue le langage Python des autres langages de programmation comme le C ou le C++, qui utilisent des caractères spéciaux, tels que les accolades { }, pour regrouper des instructions. Tu dois donc faire particulièrement attention à l'indentation dans Python, sinon ton programme ne fonctionnera pas correctement !

La plupart du temps, Python ne te permet pas de séparer des lignes en plusieurs morceaux, ce qui explique pourquoi tu vois souvent la flèche ↲ dans ce livre, qui t'indique que la ligne ne doit pas être coupée.

Cependant, il existe deux autres méthodes pour traiter les lignes de code trop longues dans Python.

La première méthode consiste à utiliser un caractère de continuation de ligne : si le dernier caractère de ta ligne est une barre oblique inversée (\), cela veut dire que tu peux passer à une autre ligne.

```
a = 1
if a == 1 or \
a == 2:
    print("oui")
```

Copyright © 2016 Eyrolles.

Quant à la deuxième méthode, elle consiste à couper une longue ligne de code là où il est évident qu'elle ne peut pas finir. Par exemple, lorsque tu définis des valeurs initiales dans une liste ou quand tu utilises une fonction, tu peux tout à fait couper tes lignes de code aux endroits où Python saura pertinemment qu'une suite est attendue. Ci-dessous, tu peux voir deux exemples probants qui t'aideront à comprendre comment Python te permet de telles coupes. Les crochets et parenthèses restés ouverts indiquent à Python que la ligne n'est pas terminée et qu'elle finira dès qu'ils se refermeront :

```
noms = ["David",
"Steve",
"Joan",
"Joanne"
]
mc.setBlocks(x1, y1, z1,
x2, y2, z2, block.AIR.id)
```

Démolir la salle de reproduction

Après avoir exécuté ton programme de duplicateur à plusieurs reprises, tu finiras certainement avec de nombreuses salles de reproduction éparpillées un peu partout dans Minecraft. Or à chaque fois, c'est la dernière que tu crées qui fonctionne réellement. Si tu continues ainsi, tu te trouveras bientôt avec tellement de salles que tu ne sauras plus où donner de la tête ! Pour remédier à ce problème, tu vas devoir ajouter une fonctionnalité à ton programme qui t'invitera à démolir ta salle de reproduction après t'en être servi afin d'éviter que ton monde ne croule sous les vieux duplicateurs.

Ce programme de démolition ressemble en tous points au programme `deblayage.py` du chapitre 3. Tout ce que tu as besoin de connaître, ce sont les coordonnées extérieures de ta salle. Et comme ta salle est plus grande d'un bloc par rapport à ton espace de reproduction, qui lui est défini par `DIMENSIONSX`, `DIMENSIONSY` et `DIMENSIONSZ`, il te suffira d'effectuer un simple calcul mathématique.

Modifie ta fonction `demolitionSalle()` pour qu'elle ressemble à ceci :

```
def demolitionSalle():
    mc.setBlocks(sallex, salley, sallez,
                 sallex+DIMENSIONSX+2, salley+DIMENSIONSY+2,
                 sallez+DIMENSIONSZ+2, block.AIR.id)
```

Enregistre ton programme et exécute-le à nouveau. Démolir ta salle de reproduction ne devrait plus te poser de problème. Choisis simplement l'option 1 de ton menu pour la construire, puis l'option 7 pour la démolir.

Copyright © 2016 Eyrolles.

Scanner dans la salle de reproduction

Tu as déjà écrit cette partie du programme dans ton script `scannage3D.py`, donc tu peux t'en servir en y apportant de légères modifications.

1. Remplace la fonction `scannage3D()` par le code suivant. Ce code est presque identique à celui du programme `scannage3D.py`. Seules les lignes en gras y ont été ajoutées pour afficher la progression des opérations de scannage dans le chat de Minecraft. En effet, scanner une grande salle peut prendre beaucoup de temps, alors il est parfois bien pratique de connaître l'avancée de ton programme pendant son exécution. Pour gagner un peu de temps, copie et colle les lignes de code de ton programme précédent :

```
def scannage3D(nomdefichier, originex, originey, originez):
    f = open(nomdefichier, "w")
    f.write(str(DIMENSIONSX) + "," + str(DIMENSIONSY) + "," ↩
 + str(DIMENSIONSZ) + "\n")

    for y in range(DIMENSIONSY):
        mc.postToChat("scannage en cours :" + str(y))
        f.write("\n")
        for x in range(DIMENSIONSX):
            ligne = ""
            for z in range(DIMENSIONSZ):
                blockid = mc.getBlock(originex+x, ↩
 originey+y, originez+z)
                if ligne != "":
                    ligne = ligne + ","
                ligne = ligne + str(blockid)
            f.write(ligne + "\n")
    f.close()
```

2. Enregistre ton programme et teste-le à nouveau. Pour cela, rends-toi dans la salle de reproduction et construis quelque chose, puis choisis l'option 3 dans ton menu. Ouvre le fichier qui a été créé pour vérifier que l'objet a été scanné correctement. Sur la figure 6-10, tu peux voir une partie d'un fichier scanné. Tu remarqueras qu'il contient beaucoup de zéros. C'est parce que ta salle a également scanné tous les blocs `AIR`.

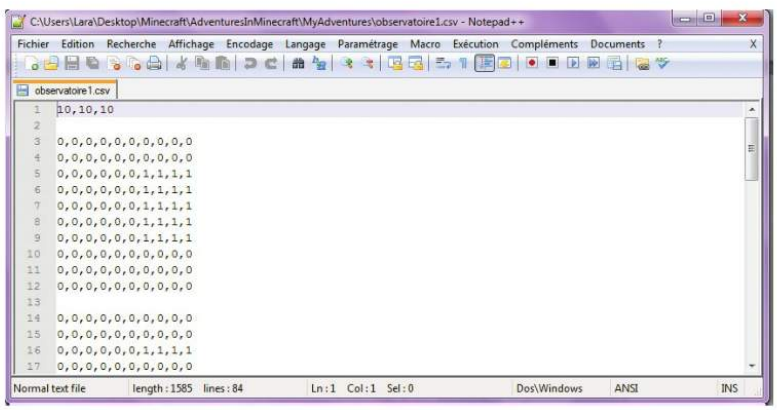


FIGURE 6-10 Le contenu d'un fichier scanné

Copyright © 2016 Eyrolles.

Nettoyer la salle de reproduction

Besoin d'un nettoyage de printemps ? Il peut parfois s'avérer bien pratique de recommencer de zéro et de débarrasser la salle de tout ce qu'elle contient. Tu pourrais tout aussi bien parvenir au même résultat en la détruisant et en la reconstruisant, mais il est tellement simple d'ajouter une fonctionnalité de nettoyage qu'il serait dommage de s'en passer. Tout ce qui change par rapport à la fonction `demolitionSalle()`, ce sont les coordonnées.

1. Modifie la fonction `nettoyageSalle()` pour qu'elle ressemble au code suivant. Tu noteras que les coordonnées de départ sont toutes supérieures aux coordonnées des parois de la salle et que les coordonnées de fin ne sont pas plus élevées que celles de la fonction `demolitionSalle()`. De cette manière, tu vas nettoyer l'espace intérieur de la salle sans en détruire les murs :

```
def nettoyageSalle():
    mc.setBlocks(sallex+1, salley+1, sallez+1,
                 sallex+DIMENSIONSX+1, salley+DIMENSIONSY+1,
                 sallez+DIMENSIONSZ+1, block.AIR.id)
```

2. Sauvegarde ton programme et exécute-le à nouveau. Crée un objet dans la salle puis choisis l'option 6 pour savoir si tu es capable de nettoyer la salle en un claquement de doigts.

Imprimer dans la salle de reproduction

Imprimer (ou reproduire) un objet qui se trouve dans ta salle est une opération on ne peut plus simple à réaliser, car tu as déjà rédigé le programme qui te permet de le faire : il s'agit de ton programme `impression3D.py`. Le voici mis en contexte :

```
def impression3D(nomdefichier, originex, originey, originez):
    f = open(nomdefichier, "r")
    lignes = f.readlines()
    coords = lignes[0].split(",")
    dimensionsx = int(coords[0])
    dimensionsy = int(coords[1])
    dimensionsz = int(coords[2])
    idxligne = 1
    for y in range(dimensionsy):
        mc.postToChat("impression en cours:" + str(y))
        idxligne = idxligne + 1
        for x in range(dimensionsx):
            ligne = lignes[idxligne]
            idxligne = idxligne + 1
            donnee = ligne.split(",")
            for z in range(dimensionsz):
                blockid = int(donnee[z])
                mc.setBlock(originex+x, originey+y,
                            originez+z, blockid)
```

Copyright © 2016 Eyrolles.

Enregistre ton programme et exécute-le à nouveau. Construis un objet dans la salle puis déplace-toi dans le jeu et choisis l'option 5 du menu. Tout ce qui se trouve dans ta salle sera scanné puis reproduit juste à côté de ton joueur. Tu peux répéter l'opération autant de fois que tu le souhaites et reproduire des objets un peu partout dans Minecraft. Essaye d'en reproduire dans le ciel et sous l'eau pour voir ce qu'il se passe ! Sur la figure 6-11, tu peux voir le résultat de l'opération après l'exécution du programme.



FIGURE 6-11 Un rocher reproduit à l'identique à côté du joueur

Créer un répertoire de fichiers

Ta dernière mission consiste à écrire une petite fonction bien pratique qui va te permettre de répertorier tous les fichiers dans un système de gestion de fichiers CSV. Tu pourrais aussi bien utiliser ton propre gestionnaire de fichiers pour cela, mais il est quand même plus intéressant d'ajouter cette fonctionnalité à ton programme, car tu auras ainsi tout ce dont tu as besoin dans un seul et même endroit :

1. Modifie la fonction `repertoireFichiers()` pour pouvoir utiliser la fonction `glob.glob()` qui lira tous les fichiers d'un répertoire et les imprimera. Pour en savoir plus sur cette fonction, réfère-toi à l'encadré « Percer les secrets du code ».

```
def repertoireFichiers():
    print("\nFICHIERS:")
    fichiers = glob.glob("*.csv")
    for nomdefichier in fichiers:
        print(nomdefichier)
    print("\n")
```

Copyright © 2016 Eyrolles.

2. Enregistre ton programme et exécute-le à nouveau. Scanne quelques objets dans des fichiers CSV, choisis l'option 2 dans le menu et assure-toi qu'aucun objet ne manque. Sur la figure 6-12, tu peux voir les fichiers que j'ai créés lorsque j'ai exécuté ce programme sur mon ordinateur.

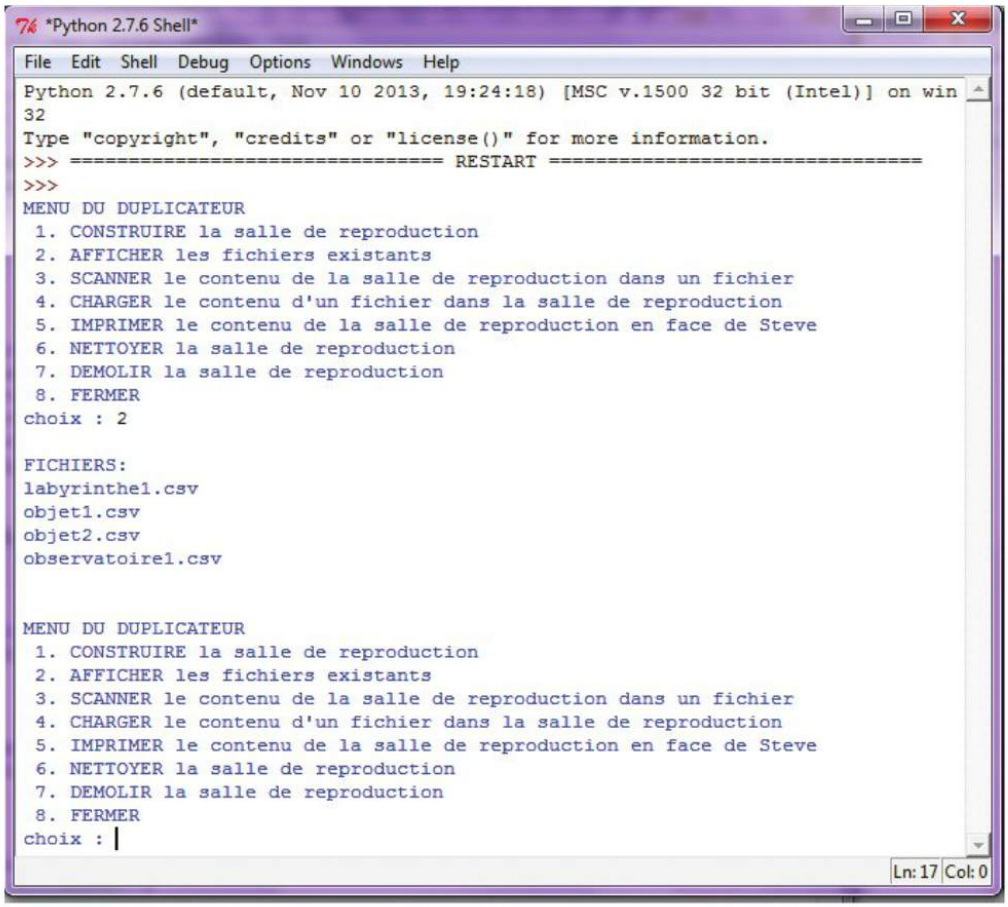


FIGURE 6-12 Le répertoire des fichiers CSV que tu as créés

PERCER LES SECRETS DU CODE

`glob.glob()` : quel curieux nom pour une fonction ! Si drôle qu'on le répète même deux fois !

Si le nom `glob` est utilisé deux fois, c'est parce que la première occurrence est le nom du module (le module `glob`) depuis lequel il a été importé au début du programme. La deuxième est le nom de la fonction `glob` qui se trouve à l'intérieur du module du même nom.

Mais à quoi sert la fonction `glob` et pourquoi porte-t-elle ce nom ?

Copyright © 2016 Eyrolles.

Pour ce qui concerne le nom, il s'agit simplement de l'abréviation de l'expression « global command » (commande de base, en français), qui remonte aux tous débuts du système d'exploitation d'Unix. Pour en savoir plus sur l'histoire qui se cache derrière l'abréviation `glob`, tu peux lire l'article de l'encyclopédie Wikipédia qui y est consacré (en anglais) : `http://en.wikipedia.org/wiki/Glob_(programming)`.

Quant à son utilité, elle permet d'obtenir une liste de fichiers répondant à une recherche précise ou à un **métacaractère**. Par exemple, lorsque tu utilises `glob.glob("*.csv")`, Python analyse le répertoire dans le système de ton ordinateur et affiche une liste de tous les fichiers se terminant par `.csv`.

Donc, si tu as placé les fichiers `un.csv`, `deux.csv` et `trois.csv` dans ton dossier `MyAdventures` et que tu utilises `glob.glob("*.csv")`, tu obtiendras une liste Python qui ressemble à cela :

```
['un.csv', 'deux.csv', 'trois.csv']
```

Or, souviens-toi que la boucle *for* parcourt tous les éléments d'une liste. Donc, retiens bien les lignes de code suivantes, car elles sont très utiles :

```
for nom in glob.glob("*.csv") :
      print(nom)
```



Un **métacaractère** est un caractère spécial qui permet de sélectionner plusieurs noms ou mots identiques. Il fonctionne un peu comme un « joker » dans un jeu de cartes, car il peut représenter tout et n'importe quoi.

En Python, le métacaractère est souvent utilisé pour sélectionner plusieurs fichiers du même format dans un répertoire, par exemple tous les fichiers de type « CSV ». Pour cela, on utilise `glob.glob("*.csv")`. Dans cette commande, l'astérisque (`*`) indique la position du métacaractère et la fonction `glob.glob()` trouvera tous les fichiers `.csv` que contient l'ordinateur.

Copyright © 2016 Eyrolles.

DÉFI



Si tu saisis un nom de fichier qui n'existe pas dans ton programme `duplicateur.py`, ce dernier va planter en t'indiquant une erreur et va laisser ta salle de reproduction telle quelle dans le monde de Minecraft, sans la détruire. Essaye d'améliorer ton programme en recherchant sur Internet un moyen de détecter les fichiers non existants pour éviter que ton programme ne plante.

Lorsque Martin a testé mon programme `duplicateur.py`, il y a trouvé un bug : la salle de reproduction est en effet plus grande qu'elle devrait l'être. Cela veut dire que si tu construis un objet de la taille de ta salle, une portion de blocs ne sera pas scannée dans le fichier. Donc ton objet sera reproduit sans cette portion de blocs. Cela pourrait devenir gênant, surtout si tu construis une maison ! Essaye de détecter la cause de ce bug et de voir si tu peux le réparer.

Tableau de références

Lire des lignes dans un fichier	Écrire des lignes dans un fichier

`f = open("données.txt", "r")`
`astuces = f.readlines()`
`f.close()`
`for t in astuces:`
`    print(t)`
	`f = open("scores.txt", "w")`
	`f.write("Victoria:26000\n")`
	`f.write("Amanda:10000\n")`
	`f.write("Ria:32768\n")`
	`f.close()`

Obtenir une liste de noms de fichiers analogues	**Enlever des espaces blancs en trop dans des chaînes de caractères**

`import glob`
`noms = glob.glob("*.csv")`
`for n in noms:`
`    print(n)`
	`a = "\n\n salut \n\n"`
	`a = a.strip()`
	`print(a)`

Explorer d'autres pistes avec les fichiers de données

Dans ce chapitre, tu as appris à lire et à écrire dans des fichiers de données. Cela t'ouvre d'infinies possibilités d'enregistrer et de restaurer des parties du monde de Minecraft et même d'introduire d'immenses volumes de données que tu peux trouver ailleurs, sur des sites Internet par exemple. Tu as aussi construit ton propre labyrinthe en 3D en le dotant d'innombrables couloirs sinueux et d'impasses. Puis tu as terminé en bâtissant une imprimante 3D en parfait état de fonctionnement, avec un menu très complet, qui va d'ailleurs te servir dans beaucoup de programmes !

- Il existe toute une panoplie de sites Internet recensant des « données en temps réel ». Lors de mes recherches pour la préparation de ce livre, je suis tombé sur un

Copyright © 2016 Eyrolles.

site qui recense les données de stationnement de la ville de Nottingham : `http://data.nottinghamtravelwise.org.uk/parking.xml`. Ces données sont mises à jour toutes les cinq minutes et fournissent un suivi de toutes les entrées et sorties des aires de stationnement de la ville. J'ai écrit à l'occasion un programme Python qui traite ces données et reproduit des informations utiles, comme tu peux le voir sur ma page GitHub : `https://github.com/whaleygeek/pyparsers`. Essaye d'utiliser ce programme pour créer un jeu dans Minecraft avec des aires de stationnement et des voitures à l'intérieur. Tu pourrais par exemple en faire un jeu de chasse à la place de stationnement en temps limité !

- Tout ce que tu as toujours voulu savoir sur les labyrinthes en 3D est expliqué en détail sur ce site web : `http://www.astrolog.org/labyrnth/algrithm.htm`. Cette page inclut un tas d'exemples de labyrinthes en 3D multijoueurs, qui sont enregistrés dans des fichiers de traitement de texte ordinaires. Parcours un peu le site et vois si tu peux trouver un fichier de labyrinthe multijoueurs. Modifie ensuite ton programme de labyrinthe pour te servir des techniques apprises lors de la création de l'imprimante 3D et crée un immense labyrinthe composé de plusieurs couches. Si le cœur t'en dit, tu pourras même t'inspirer des quelques conseils du site pour écrire un programme capable de produire des labyrinthes automatiquement !

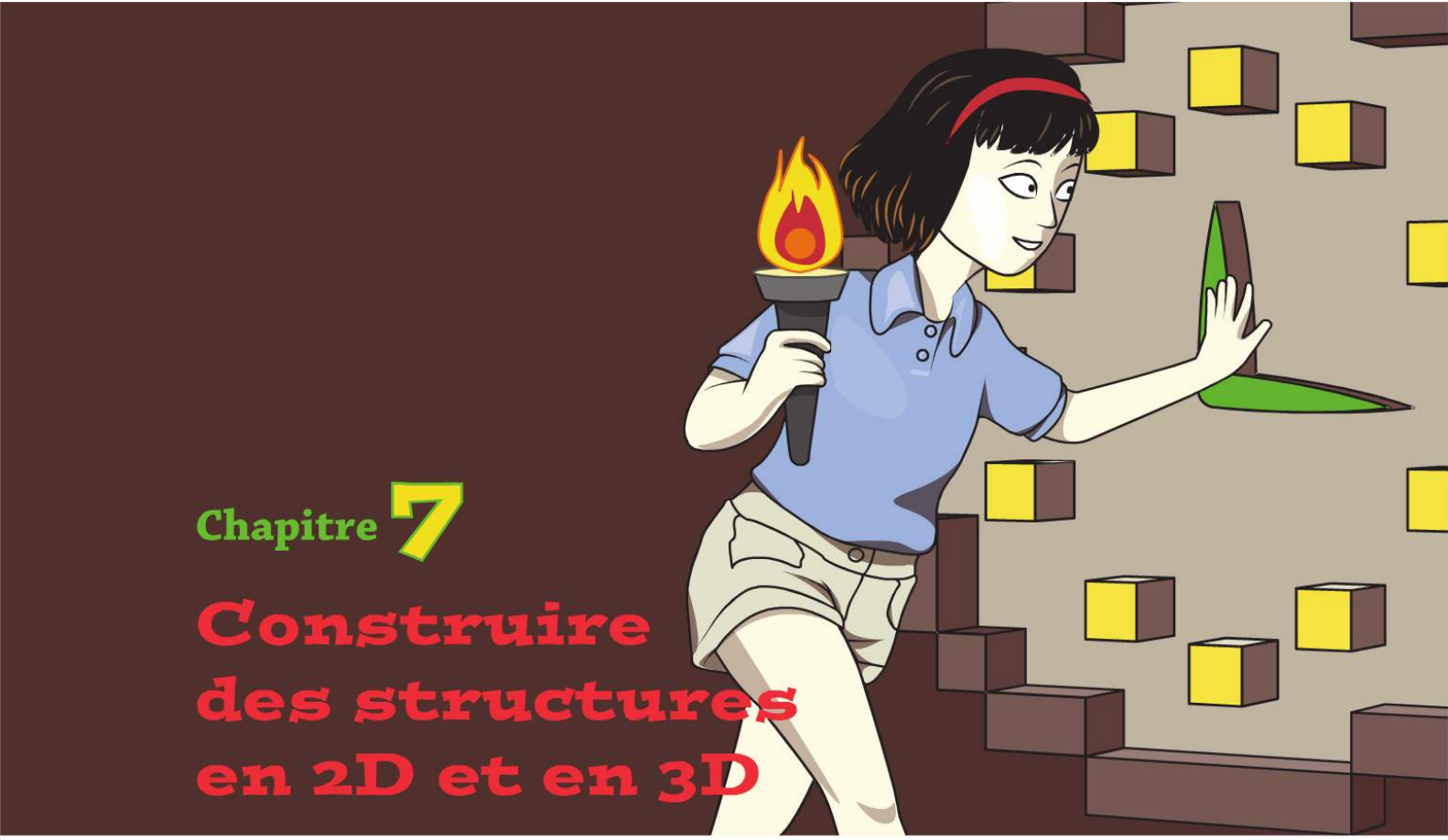


Niveau terminé : bravo ! Tu as réussi à briser les lois de la physique en faisant magiquement apparaître et disparaître d'immenses objets dans Minecraft, tout ça en un claquement de doigts !

Dans le prochain chapitre...

Dans le chapitre 7, tu vas apprendre à construire des objets complexes en 2D et en 3D à l'aide de Python. Tu apprendras notamment à contrôler la course folle du temps en construisant une horloge géante, à bâtir des polygones et autres objets complexes avec quelques lignes de code et à mener une expédition hors du commun sur les sentiers des pyramides !

Copyright © 2016 Eyrolles.



Chapitre 7

Construire des structures en 2D et en 3D

L'UN DES PLUS grands avantages de la programmation dans Minecraft, c'est qu'en plus de pouvoir visualiser tes créations sur un écran 2D, tu peux aussi leur donner vie en les construisant dans un monde en 3D. Tu peux t'amuser à courir autour, à les voir de l'intérieur, à les agrandir et même à les faire exploser ! En récupérant les idées que tu as utilisées pour afficher des objets en 3D sur un écran en 2D, tu peux concevoir des objets en 3D dans Minecraft et faire des miracles !

Dans ce chapitre, tu apprendras à utiliser le module `minecraftstuff` pour tracer des lignes et des formes en 2D. Avec un peu de mathématiques, ces dernières te permettront de programmer une horloge si grande que tu pourras poser ton joueur sur ses aiguilles en mouvement (voir la figure 7-1). Une fois que tu seras passé maître dans l'art de créer des formes en 2D, tu apprendras à les assembler pour bâtir d'immenses structures en 3D en seulement quelques secondes.

Copyright © 2016 Eyrolles.



FIGURE 7-1 Une gigantesque horloge Minecraft

Le module minecraftstuff

`minecraftstuff` est un module d'extension de l'API de Minecraft qui a été écrit spécialement pour les besoins de ce livre. Il contient toutes les lignes de codes nécessaires pour dessiner des formes et prendre le contrôle d'objets en 3D. Il est doté d'un lot de fonctions appelé `MinecraftDrawing`, qui te permet de créer des segments, des cercles, des sphères et d'autres formes. Comme ce code se trouve dans un **module** à part, tu pourras programmer en toute simplicité et comprendre plus facilement ce que tu fais. Les modules sont un moyen de scinder un programme en plusieurs sections. Lorsqu'un programme devient trop gros, il devient aussi trop compliqué à comprendre et repérer des problèmes peut prendre beaucoup de temps.

Tu trouveras le module `minecraftstuff` dans le kit de démarrage à télécharger sur le site Internet du livre `www.editions-eyrolles.com/dl/14292`. Tu pourras l'importer dans n'importe quel programme Minecraft exactement comme les modules `minecraft` et `block`.

Créer des segments, des cercles et des sphères

En assemblant plusieurs petits objets, tu peux créer des structures aussi complexes et larges que tu le souhaites. Dans ce chapitre, tu vas utiliser des formes aussi diverses et variées que des segments, des cercles et des sphères en vue de construire une immense structure Minecraft. Dans cette section, tu vas d'abord créer un nouveau programme, importer les modules dont tu as besoin et assembler les modules `minecraft` et

Copyright © 2016 Eyrolles.

minecraftdrawing. Plus tard, tu utiliseras les fonctions de ces modules pour tracer des segments, des cercles et des sphères.

Pour commencer, lance Minecraft et IDLE ainsi que le serveur Bukkit si tu es sur un PC ou un Mac. Tu t'es déjà certainement un peu familiarisé avec le processus, mais n'hésite pas à relire le chapitre 1 si tu as besoin de te rafraîchir la mémoire.

1. Ouvre IDLE et crée un nouveau fichier. Nomme ton programme segments CerclesEtSpheres.py et enregistre-le dans le dossier MyAdventures.
2. Importe ensuite les modules minecraft, block, minecraftstuff et time en saisissant les lignes suivantes :

```
import mcpi.minecraft as minecraft
import mcpi.block as block
import mcpi.minecraftstuff as minecraftstuff
import time
```

3. Établis la connexion avec Minecraft :

```
mc = minecraft.Minecraft.create()
```

4. Utilise le module minecraftstuff pour créer un objet MinecraftDrawing en saisissant la ligne suivante :

```
mcdrawing = minecraftstuff.MinecraftDrawing(mc)
```

PERCER LES SECRETS DU CODE

MinecraftDrawing est une classe du module minecraftstuff. Les classes font partie de la programmation dite « orientée objet », qui consiste à assembler des fonctions et des données similaires – ici, les fonctions et les données des dessins Minecraft – pour en faciliter la compréhension et l'utilisation. Quand tu utilises une classe et lui donnes un nom (par exemple, dessinmc), cela te donne la possibilité de créer un objet appartenant à cette classe. C'est ce qu'on appelle « l'instanciation ». Un objet ressemble à une variable, mais en plus de détenir des valeurs ou des données, il peut aussi contenir des fonctions !

La programmation orientée objet (POO) est une discipline complexe qui fait couler beaucoup d'encre. De nombreux livres ont été écrits sur le sujet pour expliquer ce que c'est et comment la mettre à profit. Tu trouveras cependant une introduction, en anglais, bien pratique aux classes dans Python à l'adresse suivante : en.wikibooks.org/wiki/A_Beginner's_Python_Tutorial/Classes.

Pour voir d'un peu plus près le code que renferme le module minecraftstuff, tu peux ouvrir le fichier minecraftstuff.py qui se trouve dans le dossier MyAdventures/mcpi avec IDLE.

Copyright © 2016 Eyrolles.

Tracer des segments

L'objet `MinecraftDrawing` possède une fonction `drawLine()` qui nécessite, lorsqu'on fait appel à elle, qu'on lui fournisse en paramètres deux coordonnées (x, y, z) et un type de bloc. La fonction trace alors une ligne de blocs entre les deux positions en question. Elle fonctionne exactement comme la fonction `setBlocks()` que tu as appris à utiliser dans le chapitre 3.



Pour s'exécuter, une fonction a besoin d'informations, comme `setBlock()` qui requiert des coordonnées x, y, z ainsi qu'un type de bloc. Ces informations sont ce qu'on appelle des **paramètres**. Lorsqu'un programme utilise cette fonction, ont dit qu'il l'**appelle** et qu'il lui **passe** des valeurs en paramètre.

La fonction suivante trace une ligne de blocs comme celle que tu peux voir à la figure 7-2 :

```
drawLine(x1, y1, z1, x2, y2, z2, blockType, blockData)
```



FIGURE 7-2 La fonction drawLine() trace une ligne de blocs entre deux séries de coordonnées x,y et z.

Tu vas maintenant modifier ton programme pour utiliser la fonction `drawLine()` et créer trois lignes dans Minecraft : une verticale, une horizontale et une diagonale, toutes trois partant de la position de ton joueur. Pour cela, tu dois ajouter le code suivant à la fin du programme `segmentsCerclesEtSpheres.py` :

Copyright © 2016 Eyrolles.

1. Tu dois tout d'abord trouver la position de ton joueur en saisissant le code suivant :

```
pos = mc.player.getTilePos()
```

2. Pour tracer un segment vertical de 20 blocs en partant de la position de ton joueur, saisis les lignes suivantes :

```
mcdrawing.drawLine(pos.x, pos.y, pos.z,
                   pos.x, pos.y + 20, pos.z,
                   block.WOOL.id, 1)
```

3. Maintenant, trace un segment horizontal de 20 blocs juste en face en saisissant les lignes suivantes :

```
mcdrawing.drawLine(pos.x, pos.y, pos.z,
                   pos.x + 20, pos.y, pos.z,
                   block.WOOL.id, 2)
```

4. Puis trace une diagonale qui se terminera 20 blocs plus loin et plus haut :

```
mcdrawing.drawLine(pos.x, pos.y, pos.z,
                   pos.x + 20, pos.y + 20, pos.z,
                   block.WOOL.id, 3)
```

5. Enfin, comme le programme n'est pas encore terminé, ajoute un délai pour te laisser le temps d'observer son exécution :

```
time.sleep(5)
```

6. Il ne te reste plus qu'à enregistrer ton fichier et à exécuter le programme. Tu devrais voir apparaître trois lignes de blocs, chacune d'une couleur de laine différente : une ligne élevée à la verticale à partir de la position de ton joueur, une autre déployée à l'horizontale et une troisième partant en diagonale entre les deux précédentes.

PERCER LES SECRETS DU CODE

La fonction `drawLine()` utilise l'algorithme de Bresenham pour tracer le segment. Pour en savoir plus sur cet algorithme, tu peux te rendre sur la page : https://fr.wikipedia.org/wiki/Algorithme_de_tracé_de_segment_de_Bresenham.

Copyright © 2016 Eyrolles.

Dessiner des cercles

Tu n'es pas obligé de t'en tenir aux lignes, tu peux aussi utiliser `MinecraftDrawing` pour tracer des cercles en te servant de la fonction `drawCircle()` et en passant en paramètres le centre du cercle, un rayon et un type de bloc. Pour créer un cercle, tu peux donc utiliser la fonction suivante :

```
drawCircle(x, y, z, radius, blockType, blockData)
```

Pour obtenir le cercle de la figure 7-3, ajoute le code suivant à la fin de ton programme `segmentsCerclesEtSpheres.py` :

1. Retrouve avant tout la position de ton joueur en saisissant le code suivant :

```
pos = mc.player.getTilePos()
```

2. Maintenant, dessine un cercle 20 blocs au-dessus de ton joueur en le dotant d'un rayon de 20 blocs, comme ceci :

```
mcdrawing.drawCircle(pos.x, pos.y + 20, pos.z, 20,
                     block.WOOL.id, 4)
```

3. Enfin, ajoute un délai pour te laisser le temps d'observer l'exécution de ton programme tout en te déplaçant dans le jeu :

```
time.sleep(5)
```

4. Enregistre ton fichier et exécute le programme.

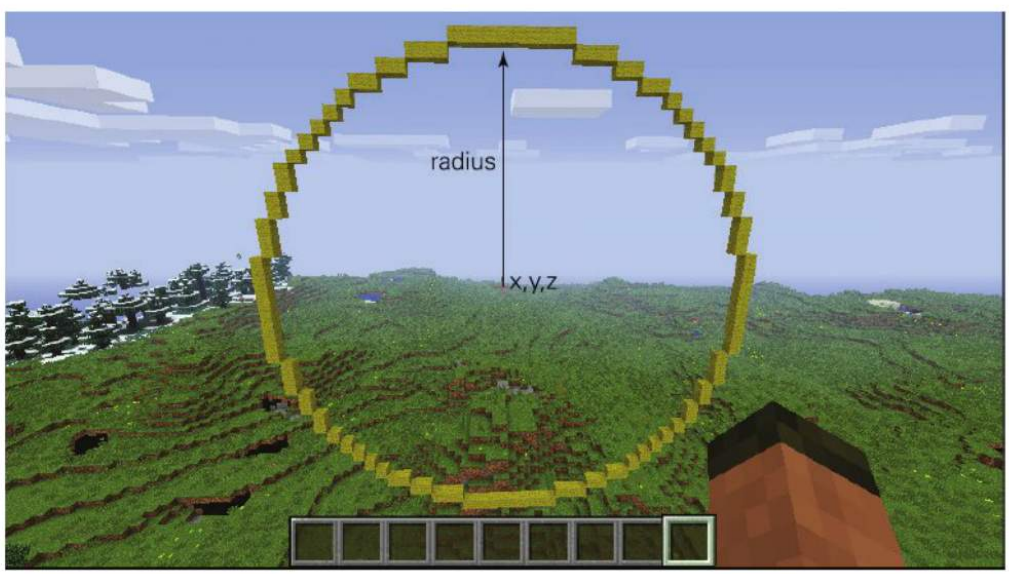


FIGURE 7-3 drawCircle trace un cercle d'un rayon donné autour d'un centre (x,y,z) donné.

Les segments apparaîtront en premier puis tu auras cinq secondes pour déplacer ton joueur avant que le cercle ne se dessine juste au-dessus des segments.

Copyright © 2016 Eyrolles.

PERCER LES SECRETS DU CODE

La fonction `drawCircle()` utilise l'algorithme de tracé d'arc de cercle par point milieu pour construire le cercle avec des blocs. Il s'agit d'une adaptation de l'algorithme de tracé de segment de Bresenham. Pour en savoir plus, tu peux consulter la page : `https://fr.wikipedia.org/wiki/Algorithme_de_tracé_d'arc_de_cercle_de_Bresenham`.

Dessiner des sphères

La fonction `drawSphere()` est similaire à celle de `drawCircle()` dans la mesure où elle utilise aussi un centre, un rayon et un type de bloc. Ainsi, pour créer une sphère, tu vas utiliser la fonction suivante :

```
drawSphere(x, y, z, radius, blockType, blockData)
```

Pour obtenir la sphère de la figure 7-4, ajoute le code suivant à la fin de ton programme `segmentsCerclesEtSpheres.py`.

1. Retrouve la position de ton joueur en saisissant :

```
pos = mc.player.getTilePos()
```

2. Dessine une sphère 20 blocs au-dessus de ton joueur avec un rayon de 15 blocs, comme ceci :

```
mcdrawing.drawSphere(pos.x, pos.y + 20, pos.z, 15,
                     block.WOOL.id, 5)
```

3. Enregistre ton fichier et exécute le programme.



FIGURE 7-4 drawSphere dessine une sphère à l'aide d'un rayon donné autour d'un centre (x,y,z) donné.

Copyright © 2016 Eyrolles.

Tu verras les segments et les cercles se dessiner petit à petit et tu auras cinq secondes pour déplacer ton joueur avant de voir apparaître la sphère au complet.

Si besoin, tu peux télécharger le code entier des segments, cercles et sphères sur le site Internet du livre à l'adresse : `www.editions-eyrolles.com/dl/14292`.



Les sphères sont des objets formidables pour provoquer des explosions dans Minecraft. En dessinant une sphère remplie d'`AIR`, tu peux retirer tous les blocs autour du centre de ta sphère, ce qui produira un « trou » dans le monde.

DÉFI



Maintenant que tu sais comment créer des formes simples, essaie de programmer tes propres chefs-d'œuvre en 3D en utilisant des segments, des cercles et des sphères.

Créer une horloge dans Minecraft

Maintenant que tu sais comment créer des cercles et des segments, tu visualises peut-être déjà la manière dont tu vas coder l'horloge de la figure 7-1. Le cadran n'est autre qu'un immense cercle et les aiguilles des segments. Mais les choses se compliquent un peu, car tu vas devoir calculer l'emplacement des segments et les mettre en mouvement.



Tu peux regarder une vidéo didactique sur la création de l'horloge Minecraft en te rendant sur le site Internet du livre : `www.editions-eyrolles.com/dl/14292`.

Dans cette section du chapitre, tu vas te servir de tes connaissances en **trigonométrie** pour calculer l'orientation des aiguilles et modifier leur angle de virage sur le cadran à l'aide des coordonnées x et y, comme sur la figure 7-5. Tu vas simuler le mouvement des aiguilles en les traçant avec des blocs, puis les effacer en utilisant des blocs d'`AIR` et les faire réapparaître dans une nouvelle position.

Copyright © 2016 Eyrolles.

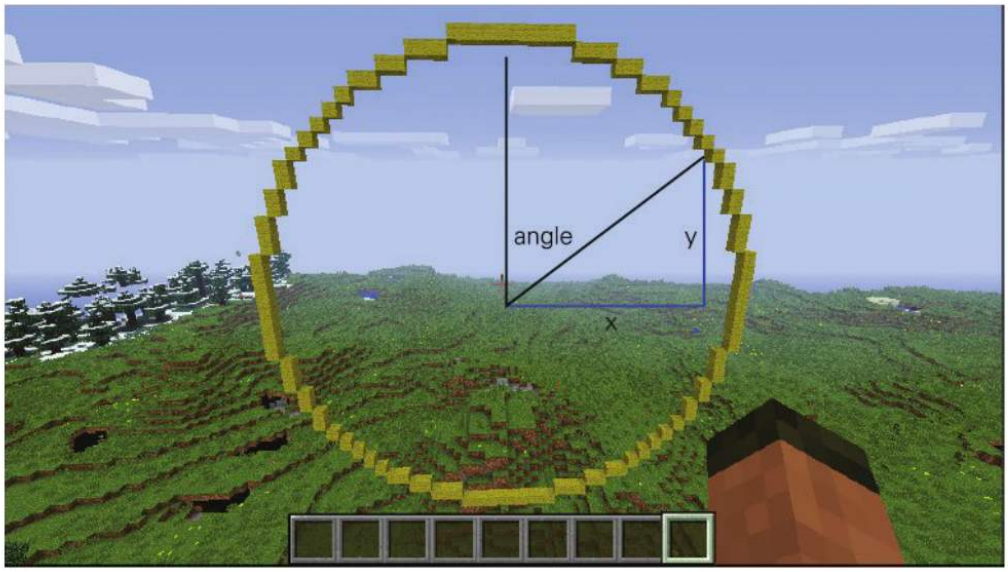


FIGURE 7-5 Calculer l'orientation des aiguilles de l'horloge.

La **trigonométrie** est une branche des mathématiques qui traite des relations entre distances et angles dans les triangles. Pour en savoir plus, consulte la page `https://fr.wikipedia.org/wiki/Trigonométrie`.



Pour créer ton horloge, suis ces étapes.

1. Crée un nouveau programme pour ton horloge Minecraft en ouvrant IDLE et en créant un nouveau fichier. Nomme ton programme `horlogeMinecraft.py` et enregistre-le dans le dossier `MyAdventures`.

2. Importe ensuite les modules `minecraft`, `block`, `minecraftstuff`, `time`, `datetime` et `math` en saisissant les lignes suivantes :

```
import mcpi.minecraft as minecraft
import mcpi.block as block
import mcpi.minecraftstuff as minecraftstuff
import time
import datetime
import math
```

3. Crée une fonction `trouverPointSurCercle()`. Lorsque tu passes le centre du cercle et l'angle des aiguilles de l'horloge en paramètres dans la fonction, elle te retourne la position des aiguilles comme sur la figure 7-5.

```
def trouverPointSurCercle(cx, cy, radius, angle):
    x = cx + math.sin(math.radians(angle)) * radius
```

Copyright © 2016 Eyrolles.

```
    y = cy + math.cos(math.radians(angle)) * radius
    x = int(round(x, 0))
    y = int(round(y, 0))
    return(x,y)
```

4. Connecte-toi à `Minecraft` et crée l'objet `MinecraftDrawing` :

```
mc = minecraft.Minecraft.create()
mcdrawing = minecraftstuff.MinecraftDrawing(mc)
```

5. Récupère la position de ton joueur en saisissant :

```
pos = mc.player.getTilePos()
```

6. Maintenant, tu vas créer des variables pour le centre de l'horloge (que tu placeras 25 blocs au-dessus du joueur), le rayon du cadran et la longueur des aiguilles de l'horloge :

```
centreHorloge = pos
centreHorloge.y = centreHorloge.y + 25
RAYON_HORLOGE = 20
LONGUEUR_AIGUILLE_HEURES = 10
LONGUEUR_AIGUILLE_MINUTES = 18
LONGUEUR_AIGUILLE_SECONDES = 20
```

7. Dessine ensuite le cadran de l'horloge en utilisant `drawCircle()` dans la ligne de code suivante :

```
mcdrawing.drawCircle(centreHorloge.x, centreHorloge.y,
                     centreHorloge.z,
                     RAYON_HORLOGE, block.DIAMOND_BLOCK.id)
```

8. Commence une boucle sans fin. Toutes les lignes de code qui vont suivre doivent être incluses à l'intérieur de cette boucle.

```
while True:
```

9. Tu vas ensuite devoir demander l'heure à ton ordinateur en utilisant la fonction `datetime.datetime.now()`. Puis tu diviseras le temps en heures, en minutes et en secondes. Comme ton cadran contient 12 heures et non 24, tu dois préciser à ton programme de soustraire 12 heures après midi. Par exemple, s'il est 14 h 00, ton programme devra indiquer 2 heures sur l'horloge. Pour cela, saisis les lignes de code suivantes :

```
    tempsActuel = datetime.datetime.now()
    heures = tempsActuel.hour
    if heures >= 12:
        heures = tempsActuel.hour - 12
        minutes = tempsActuel.minute
        secondes = tempsActuel.second
```

Copyright © 2016 Eyrolles.

10. Tu vas maintenant dessiner l'aiguille des heures. L'angle de cette aiguille sera de 360 degrés divisé par 12 heures et multiplié par l'heure actuelle. Trouve les coordonnées x et y de la pointe de l'aiguille en utilisant la fonction `trouverPointSurCercle()` et dessine l'aiguille en utilisant `drawLine()`, comme ceci :

```
        angleAiguilleHeures = (360 / 12) * heures
        aiguilleHeuresX, aiguilleHeuresY = ↩
trouverPointSurCercle(centreHorloge.x, centreHorloge.y,
                      LONGUEUR_AIGUILLE_HEURES,
                      angleAiguilleHeures)
        mcdrawing.drawLine(centreHorloge.x, centreHorloge.y,
                           centreHorloge.z,
                           aiguilleHeuresX, aiguilleHeuresY,
                           centreHorloge.z,
                           block.DIRT.id)
```

11. Dessine ensuite l'aiguille des minutes, qui se situera un bloc derrière l'aiguille des heures (`z-1`), comme ceci :

```
        angleAiguilleMinutes = (360 / 60) * minutes
        aiguilleMinutesX, aiguilleMinutesY = ↩
trouverPointSurCercle(centreHorloge.x,centreHorloge.y,
                      LONGUEUR_AIGUILLE_MINUTES,
                      angleAiguilleMinutes)
        mcdrawing.drawLine(centreHorloge.x, centreHorloge.y,
                           centreHorloge.z-1,
                           aiguilleMinutesX, aiguilleMinutesY,
                          centreHorloge.z-1,
                           block.WOOD_PLANKS.id)
```

12. Maintenant, ajoute la troisième aiguille un bloc devant (`z+1`) l'aiguille des heures :

```
        angleAiguilleSecondes = (360 / 60) * secondes
        aiguilleSecondesX, aiguilleSecondesY = ↩
trouverPointSurCercle(centreHorloge.x, centreHorloge.y,
                      LONGUEUR_AIGUILLE_SECONDES,
                      angleAiguilleSecondes)
        mcdrawing.drawLine(centreHorloge.x, centreHorloge.y,
                           centreHorloge.z+1,
                           aiguilleSecondesX,
                           aiguilleSecondesY,
                           centreHorloge.z+1,
                           block.STONE.id)
```

13. Patiente une seconde :

```
        time.sleep(1)
```

Copyright © 2016 Eyrolles.

14. Tu dois maintenant effacer l'heure en redessinant les aiguilles de ton horloge avec des blocs d'AIR. Pour cela, tu dois saisir :

```
        mcdrawing.drawLine(centreHorloge.x, centreHorloge.y,
                           centreHorloge.z,
                           aiguilleHeuresX, aiguilleHeuresY,
                           centreHorloge.z,
                           block.AIR.id)
        mcdrawing.drawLine(centreHorloge.x, centreHorloge.y,
                           centreHorloge.z-1,
                           aiguilleMinutesX,
                           aiguilleMinutesY,
                           centreHorloge.z-1,
                           block.AIR.id)
        mcdrawing.drawLine(centreHorloge.x, centreHorloge.y,
                           centreHorloge.z+1,
                           aiguilleSecondesX,
                           aiguilleSecondesY,
                           centreHorloge.z+1,
                           block.AIR.id)
```

15. Enregistre ton fichier et exécute le programme pour voir le fruit de tes efforts. Tu devrais voir apparaître l'horloge Minecraft juste au-dessus de ton joueur. Pour mieux l'admirer, déplace ton joueur tout en t'assurant de le placer du bon côté de l'horloge, sinon tu verras les aiguilles tourner dans le mauvais sens !

Au besoin, tu peux télécharger le code entier de l'horloge Minecraft sur le site Internet du livre : www.editions-eyrolles.com/dl/14292. Mais je te recommande vivement de saisir le code toi-même en suivant les étapes de ce chapitre, car tu apprendras bien mieux ainsi !

PERCER LES SECRETS DU CODE

La fonction `trouverPointSurCercle()` calcule un point (x, y) sur un cercle à partir de son centre de coordonnées (cx, cy), de son rayon, et d'un angle que tu as précisé (voir la figure 7-5).

1. On passe les données `cx`, `cy`, `radius` (rayon) et `angle` en paramètres d'entrée dans la fonction :

```
def trouverPointSurCercle(cx, cy, radius, angle):
```

2. Le point (x, y) sur le cercle est calculé par les fonctions `sin()` et `cos()` du module `math`. Le résultat est ensuite multiplié par le rayon :

```
    x = cx + math.sin(math.radians(angle)) * radius
    y = cy + math.cos(math.radians(angle)) * radius
```

Copyright © 2016 Eyrolles.

Il est nécessaire de passer des **radians** en paramètre dans les fonctions `math.sin()` et `math.cos()`. Ils permettent en effet de calculer des angles et offrent une alternative à la méthode la plus commune qui consiste à utiliser les degrés de 0 à 360. La fonction `math.radians()` est donc utilisée pour convertir les angles en radians.

1. En calculant les valeurs x et y, on tombe sur des décimales. Or, la fonction n'accepte que les nombres entiers, donc il faut utiliser les fonctions `round()` et `int()` pour arrondir le nombre décimal à la valeur entière la plus proche et ensuite le convertir en entier relatif :

```
x = int(round(x, 0))
y = int(round(y, 0))
```

2. Les valeurs x et y sont ensuite renvoyées au programme :

```
return(x,y)
```

La fonction renvoie les deux valeurs x et y en même temps. Il faut donc que le programme qui l'appelle lui fournisse deux variables.

Pour créer les aiguilles de l'horloge, tu dois passer par trois étapes.

1. Calculer l'angle de l'aiguille en divisant 360° par 12 ou par 60 selon qu'il s'agisse de l'aiguille des heures, des minutes ou des secondes. Puis multiplier le résultat par le nombre d'heures, de minutes ou de secondes :

```
angleAiguilleHeures = (360 / 12) * heures
```

2. Trouver ensuite les coordonnées (x, y) de la pointe de l'aiguille à l'aide de la fonction `trouverPointSurCercle()` :

```
aiguilleHeuresX, aiguilleHeuresY =
        trouverPointSurCercle(centreHorloge.x,
        centreHorloge.y, LONGUEUR_AIGUILLE_HEURES,
        angleAiguilleHeures)
```

   Comme la fonction renvoie deux valeurs, les deux variables `aiguilleHeuresX` et `aiguilleHeuresY` s'affichent lorsqu'elle est appelée.

3. Enfin, tracer un segment partant du centre de l'horloge jusqu'à la pointe de l'aiguille :

```
mcdrawing.drawLine(centreHorloge.x,
      centreHorloge.y, centreHorloge.z,
      aiguilleHeuresX, aiguilleHeuresY,
      centreHorloge.z, block.DIRT.id)
```

Copyright © 2016 Eyrolles.

DÉFI



Sur une montre normale, l'aiguille des heures suit le cours des minutes. Par exemple, s'il est 11 h 30, l'aiguille des heures se trouvera à mi-chemin entre 11 heures et midi. Le code que tu viens d'écrire pour créer ton horloge Minecraft n'intègre pas cette caractéristique. Au contraire, l'aiguille des heures reste bloquée sur le chiffre 11 jusqu'à la dernière seconde avant midi.

Essaye de changer le code de ton horloge pour qu'elle fonctionne comme une vraie montre en modifiant le calcul de l'angle des heures.

Dessiner des polygones

Un polygone est une figure en 2D composée de côtés droits qui se rejoignent à des intersections. Il peut posséder trois côtés (triangle) ou plus. Comme tu peux le voir à la figure 7-6, tu peux créer des polygones de toutes les sortes dans Minecraft.



FIGURE 7-6 Exemples de polygones dans Minecraft

Bien que ces figures soient en 2D, tu verras que les polygones peuvent s'avérer extrêmement utiles pour créer des objets en 3D, car tu peux construire toutes sortes d'objets en 3D en regroupant plusieurs polygones. Lorsque les polygones sont assemblés pour créer des objets en 3D, ils possèdent ce qu'on appelle des **faces**. Grâce à cela, tu vas pouvoir construire des structures fascinantes, comme le montre la figure 7-7. Il s'agit d'une reproduction des gratte-ciel de l'île de Manhattan, qui a été réalisée en assemblant plusieurs polygones. Pour en savoir plus sur les secrets de cette reproduction, tu peux te rendre sur le site : `www.stuffaboutcode.com/2013/04/minecraft-pi-edition-manhattan-stroll.html`.

Copyright © 2016 Eyrolles.



FIGURE 7-7 Reproduction des gratte-ciel du quartier de Manhattan à New York

Une **face** est l'une des surfaces planes qui composent un objet, comme le côté d'un cube ou le haut d'un tambour.



Tu peux créer des polygones ou des faces en utilisant la fonction `drawFace()` dans `MinecraftDrawing`. La fonction s'attend à recevoir une liste de points (x, y, z) qui, une fois connectés dans une séquence, vont permettre de créer un polygone entier. En passant la valeur `True` ou `False` en paramètre, tu indiqueras si la face est pleine ou vide. Enfin, le dernier paramètre consiste à savoir de quel type de bloc se compose la face (voir la figure 7-8).

```
drawFace(points, filled, blockType, blockData)
```

Crée un nouveau programme pour utiliser la fonction `drawFace()` et lance-toi dans la construction du triangle de la figure 7-8.

1. Tout d'abord, ouvre IDLE et crée un nouveau fichier. Nomme ton programme `polygone.py` et enregistre-le dans le dossier `MyAdventures`.
2. Importe les modules `minecraft`, `block` et `minecraftstuff` en saisissant les lignes suivantes :

```
import mcpi.minecraft as minecraft
import mcpi.block as block
import mcpi.minecraftstuff as minecraftstuff
```

Copyright © 2016 Eyrolles.



FIGURE 7-8 La fonction drawFace() a été utilisée pour construire un triangle à partir de trois points.

3. Connecte-toi à Minecraft et crée l'objet `MinecraftDrawing` :

```
mc = minecraft.Minecraft.create()
mcdrawing = minecraftstuff.MinecraftDrawing(mc)
```

4. Récupère la position de ton joueur en saisissant :

```
pos = mc.player.getTilePos()
```

5. Tu dois maintenant créer une liste pour stocker les points de ton polygone. Commence par saisir :

```
points = []
```

6. Puis ajoute trois points (x, y, z) à la liste. Une fois regroupés, ils te permettront de créer un triangle :

```
points.append(minecraft.Vec3(pos.x, pos.y, pos.z))
points.append(minecraft.Vec3(pos.x + 20, pos.y, pos.z))
points.append(minecraft.Vec3(pos.x + 10, pos.y + 20,
pos.z))
```

7. Utilise maintenant la fonction `MinecraftDrawing.drawFace` pour créer le triangle :

```
mcdrawing.drawFace(points, True, block.WOOL.id, 6)
```

8. Enregistre ton fichier et exécute le programme pour créer le triangle.

Copyright © 2016 Eyrolles.

PERCER LES SECRETS DU CODE

Les points de la face du triangle sont créés à l'aide de l'instruction `minecraft.Vec3(x, y, z)`. La fonction `minecraft.Vec3()` permet à l'API de Minecraft de stocker une série de coordonnées (x, y, z) ensemble. `Vec3` est l'abréviation de « vecteur 3D ».

Quant aux points de la face, ils sont ajoutés à la fin de la liste grâce à la fonction `append()`. Cette fonction permet en effet d'ajouter des éléments à la fin d'une liste.

DÉFI

Quelles autres figures peux-tu créer en utilisant `drawFace()` ? Pourquoi ne construirais-tu pas une figure à cinq côtés comme un pentagone ?



Pyramides

Cherche une photo de pyramide et observe-la bien. Que remarques-tu ? Ses côtés ont-ils tous la même forme ? Qu'ont-ils en commun et combien en possède-t-elle ?

Comme tu le sais probablement, chaque côté de la pyramide, excepté la base, est un triangle. Si les pyramides d'Égypte possèdent quatre faces (ou cinq si tu comptes la base), une pyramide peut en fait se composer de trois faces et plus. As-tu également remarqué que la base d'une pyramide rentre parfaitement dans un cercle ? Regarde la figure 7-9 pour comprendre ce que je veux dire.

Tu vas maintenant créer un programme qui, en appelant les fonctions `drawFace()` et `trouverPointSurCercle()`, va te permettre de construire une pyramide de la taille de ton choix, avec les dimensions et le nombre de faces que tu souhaites.

1. Tout d'abord, ouvre IDLE et crée un nouveau fichier. Nomme ton programme `pyramidesMinecraft.py` et enregistre-le dans le dossier `MyAdventures`.
2. Importe ensuite les modules `minecraft`, `block`, `minecraftstuff` et `math` en saisissant les lignes suivantes :

```
import mcpi.minecraft as minecraft
import mcpi.block as block
import mcpi.minecraftstuff as minecraftstuff
import math
```

Copyright © 2016 Eyrolles.

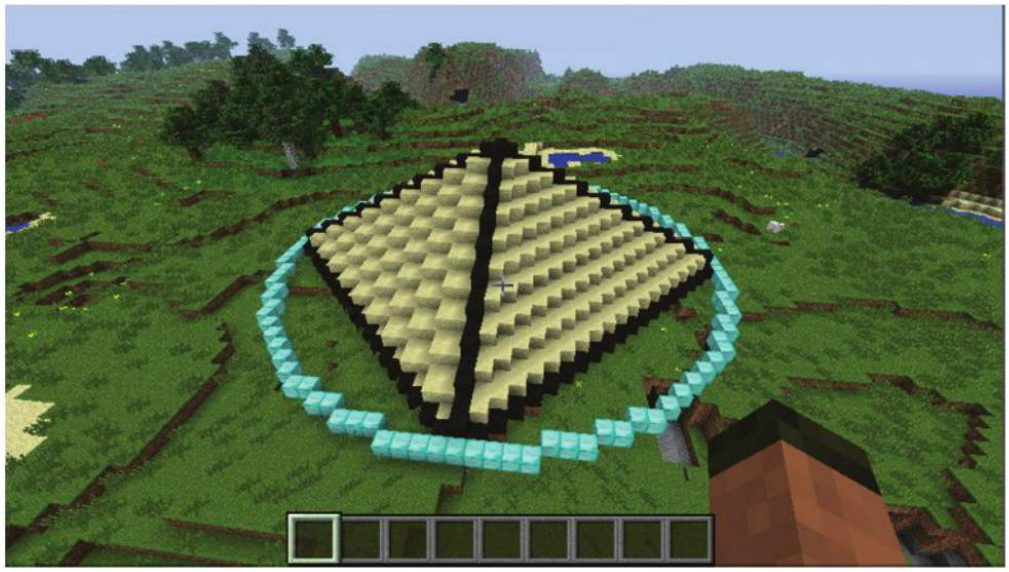


FIGURE 7-9 Une pyramide faite de triangles qui tient parfaitement dans un cercle

3. Crée la fonction `trouverPointSurCercle()`, qui sera utilisée pour calculer l'emplacement de chaque triangle sur le cercle :

```
def trouverPointSurCercle(cx, cy, radius, angle):
    x = cx + math.sin(math.radians(angle)) * radius
    y = cy + math.cos(math.radians(angle)) * radius
    x = int(round(x, 0))
    y = int(round(y, 0))
    return(x,y)
```

4. Connecte-toi à Minecraft et crée l'objet `MinecraftDrawing` :

```
mc = minecraft.Minecraft.create()
mcdrawing = minecraftstuff.MinecraftDrawing(mc)
```

5. Maintenant, définis des variables pour construire ta pyramide. Le centre de celle-ci correspondra à la position de ton joueur. Les valeurs que tu vas assigner à ces variables vont modifier la taille (donc le rayon), la hauteur et le nombre de côtés de ta pyramide. Saisis les données suivantes :

```
centrePyramide = mc.player.getTilePos()
RAYON_PYRAMIDE = 20
HAUTEUR_PYRAMIDE = 10
COTES_PYRAMIDE = 4
```

6. Introduis une boucle `for` dans ton code pour passer en revue tous les côtés de la pyramide, sans oublier d'indenter les lignes de code après le mot-clé :

```
for side in range(0, COTES_PYRAMIDE):
```

Copyright © 2016 Eyrolles.

Plus ta pyramide sera grande, plus ton programme prendra de temps à s'exécuter et à la construire dans Minecraft. D'ailleurs, si elle est trop grande, il se peut qu'elle dépasse la hauteur maximale du jeu. Alors, prends ton temps et commence par affecter des valeurs raisonnables. Cela dit, rien ne t'empêche de concevoir des pyramides géantes si tu le souhaites, mais tu devras redoubler de patience au moment de les construire, car cela prend du temps.



7. Pour chaque face de la pyramide, calcule les angles des côtés du triangle, puis utilise `trouverPointSurCercle()` pour trouver l'emplacement des coordonnées x, y et z sur le cercle. Pour connaître l'angle, tu dois diviser 360 degrés par le nombre total de côtés, puis multiplier le tout par le nombre de côtés à dessiner, comme tu peux le voir sur la figure 7-10. Saisis le code suivant :

```
    anglePoint1 = int(round((360 / COTES_PYRAMIDE) * side,0))
    point1X, point1Z =
trouverPointSurCercle(centrePyramide.x, centrePyramide.z,
                      RAYON_PYRAMIDE, anglePoint1)
    anglePoint2 = int(round((360 / COTES_PYRAMIDE) * (side +
                            1),0))
    point2X, point2Z =
trouverPointSurCercle(centrePyramide.x, centrePyramide.z,
                      RAYON_PYRAMIDE, anglePoint2)
```

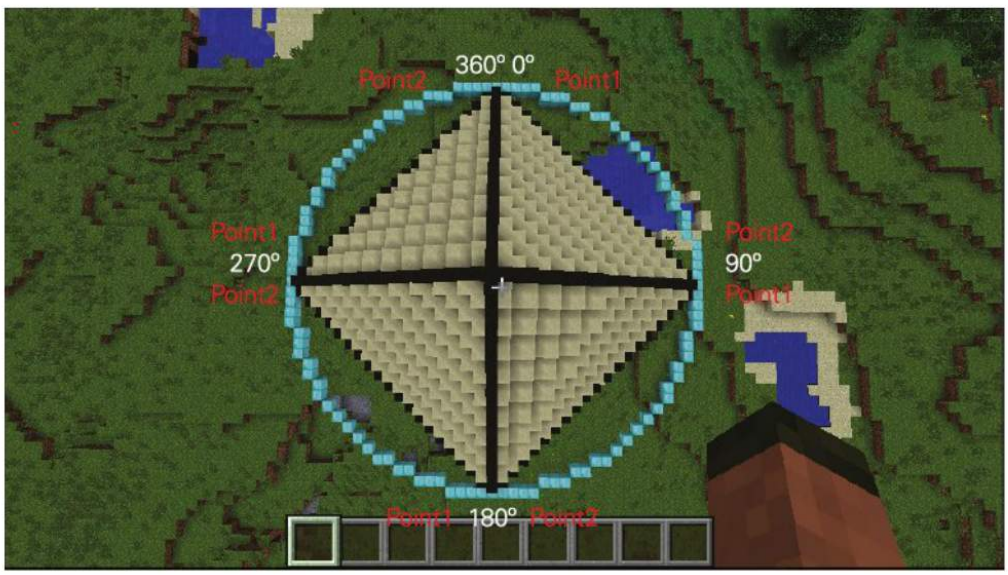


FIGURE 7-10 Les angles d'une pyramide à quatre côtés

Copyright © 2016 Eyrolles.

8. Place les points des triangles de la pyramide et utilise `drawFace()` pour en édifier les faces, comme ceci :

```
pointsTriangle = []
pointsTriangle.append(minecraft.Vec3(point1X,
                      centrePyramide.y, point1Z))
pointsTriangle.append(minecraft.Vec3(point2X,
                      centrePyramide.y, point2Z))
pointsTriangle.append(minecraft.Vec3(centrePyramide.x,
                      centrePyramide.y + HAUTEUR_
                      PYRAMIDE, centrePyramide.z))
mcdrawing.drawFace(pointsTriangle, True,
                      block.SANDSTONE.id)
```



Le grès (`block.SANDSTONE.id`) est un type de bloc très utile pour construire des pyramides, car il ressemble beaucoup au sable, mais n'est pas soumis à la gravité, donc il ne tombera pas s'il n'est soutenu par aucun bloc. Si tu construisais ta pyramide avec du sable, ton joueur serait enseveli et passerait des heures à essayer de s'en sortir.

9. Enregistre ton fichier et exécute le programme. Tu verras une pyramide apparaître au-dessus de ton joueur et le piéger à l'intérieur !

Ce programme est capable de créer des pyramides de toutes les tailles et dotées d'un nombre indéfini de faces. Essaye de modifier les variables de ta pyramide dans le programme et exécute-le à nouveau. La figure 7-11 te donne quelques exemples de pyramides assez impressionnantes.

Tu peux télécharger le code entier de cette pyramide Minecraft sur le site Internet du livre à l'adresse : `www.editions-eyrolles.com/dl/14292`.

DÉFI



La pyramide que tu as construite ne possède pas de base. Seras-tu capable de créer un polygone lui servant de base ? Cela ne devrait pas trop te poser de problèmes pour une pyramide à quatre côtés, mais si ton code est correct, il devrait aussi fonctionner pour une pyramide de cinq, six ou sept côtés.

Copyright © 2016 Eyrolles.



FIGURE 7-11 Des pyramides Minecraft

Explorer d'autres pistes avec les objets en 2D et en 3D

Grâce à la fonction `drawFace()`, tu peux créer n'importe quel type de polyèdre, c'est-à-dire une forme à faces planes comme la pyramide que tu viens de créer. Alors, pourquoi ne pas en construire plus ?

Tu trouveras des tas d'exemples de **polyèdres** et de bonnes idées sur les sites suivants :

- Maths is Fun (`www.mathsisfun.com/geometry/polyhedron.html`) ;
- Kids Math Games (`www.kidsmathgamesonline.com/facts/geometry/3d-polyhedronshapes.html`).

Copyright © 2016 Eyrolles.

Tableau de références	
Commandes	**Description**
`mcdrawing.drawLine(0, 0, 0, 10, 10, 10, block.DIRT.id)`	Tracer un segment entre deux points
`mcdrawing.drawCircle(0, 0, 0, radius, block.DIRT.id)`	Tracer un cercle
`mcdrawing.drawSphere(0, 0, 0, radius, block.DIRT.id)`	Créer une sphère

`tri = []`
`filled = True`
`tri.append(minecraft.Vec3(0,0,0))`
`tri.append(minecraft.Vec3(10,0,0))`
`tri.append(minecraft.Vec3(5,10,0))`
`mcdrawing.drawFace(tri, filled, block.DIRT.id)`
	Créer un polygone ou une face (par exemple, un triangle)

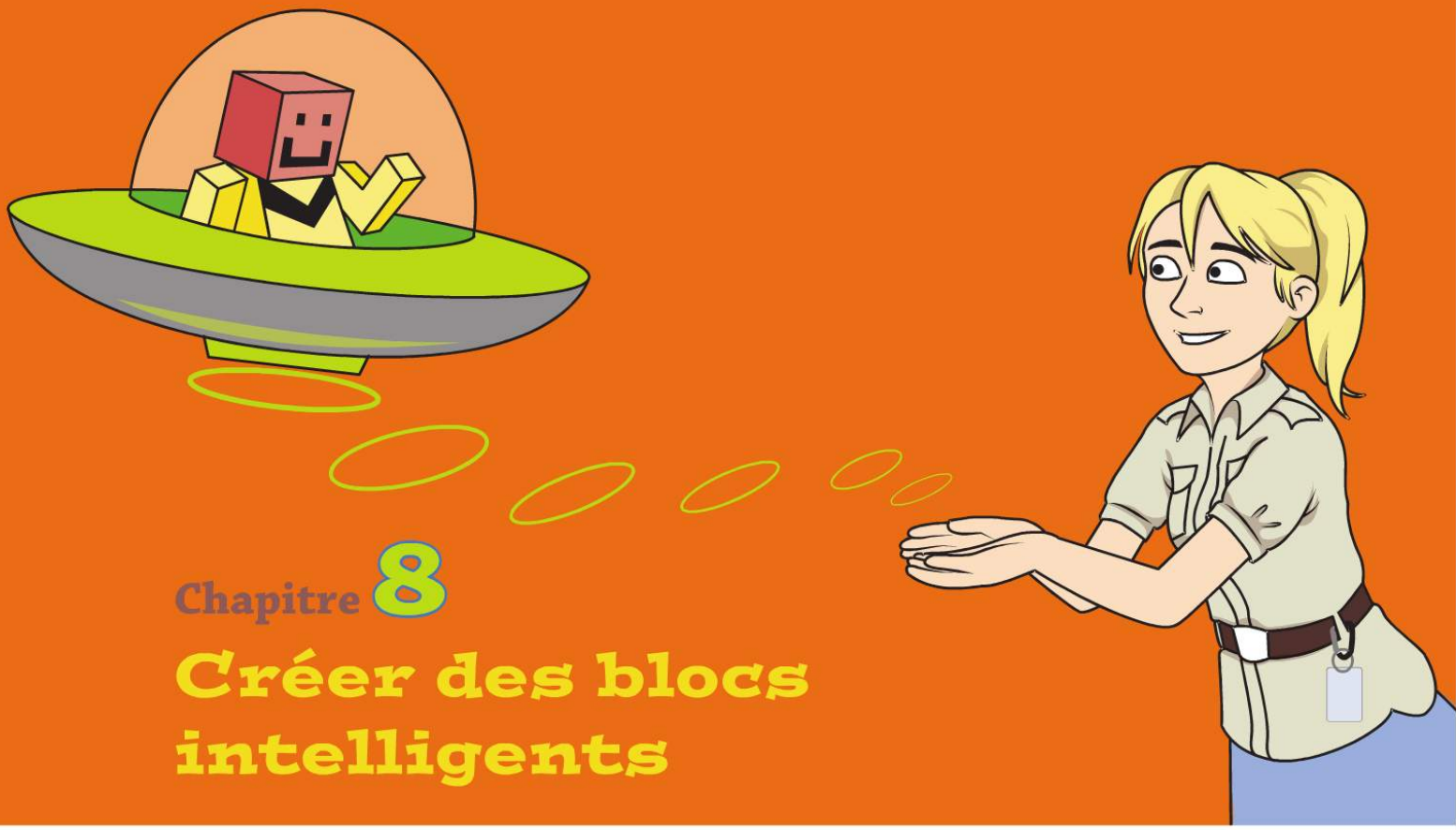


Niveau terminé : tu es passé maître dans l'art de la 3D et tu sais maintenant construire des superstructures. Chapeau bas, grand architecte !

Dans le prochain chapitre...

Dans le chapitre 8, tu vas apprendre à rendre les objets Minecraft intelligents, à te lier d'amitié avec un bloc et à empêcher une invasion d'extraterrestres !

Copyright © 2016 Eyrolles.



Chapitre 8

Créer des blocs intelligents

LES ORDINATEURS NE PENSENT PAS, ils sont incapables d'émettre la moindre pensée. Tout ce qu'ils savent faire, c'est exécuter les ordres que leur intiment les programmeurs. Tu peux toutefois t'arranger pour qu'ils fassent *comme* s'ils pensaient et prenaient des décisions de manière autonome. En programmant ton ordinateur de façon à ce qu'il « comprenne » les situations qui se présentent à lui et en inventant des règles pour qu'il « décide » des actions à mener en conséquence, tu vas t'ouvrir les portes d'un monde qui regorge d'aventures trépidantes !

Dans ce chapitre, tu vas te lier d'amitié avec un bloc et le programmer pour qu'il t'accompagne dans toutes tes pérégrinations, pourvu que tu ne t'éloignes pas trop de lui. Tu vas également fabriquer une soucoupe volante qui te prendra en chasse jusqu'à ce qu'elle puisse léviter au-dessus de ta tête et te hisser à bord avec son faisceau lumineux.

Tu vas donc apprendre à animer un bloc et à le doter de la capacité de tracer son propre chemin, en te servant du module Python `random` pour simuler son intelligence. Tu vas aussi créer des formes en utilisant les fonctions `MinecraftShape` du module `minecraftstuff`, inclus dans le kit de démarrage.

Copyright © 2016 Eyrolles.

Les blocs sont tes amis

Dans le vaste monde de Minecraft, on peut vite se sentir seul. Mais ton joueur n'est pas forcément voué à la solitude, tu as la possibilité de lui présenter un ami en animant un simple bloc qui l'accompagnera dans toutes ses aventures et lui fera la conversation (voir la figure 8-1).



FIGURE 8-1 Présente un ami à ton joueur pour l'accompagner dans ses aventures.

Pour programmer quelqu'un, tu dois d'abord te mettre à sa place et essayer de penser comme lui ou elle ! Lorsque ton joueur s'approchera de lui, il sera très heureux et mourra d'envie de le suivre dans ses aventures, puis il essayera de s'asseoir à ses côtés. Tant que tu resteras près de lui, il sera content et si tu décides de bouger, il suivra tes pas. Mais si tu t'éloignes trop, cela le rendra triste et il arrêtera de te suivre jusqu'à ce que tu reviennes et essayes de te réconcilier avec lui (pour cela, il te suffira de placer ton joueur à côté de lui).

Le programme de création de ton ami comporte deux parties.

- Les règles qui déterminent les actions que le bloc va entreprendre : cette partie du programme aide ton ami à se décider entre suivre ton joueur (la cible) ou rester immobile.
- Et le code qui permet de déplacer le bloc vers une cible : une fois que ton ami atteint sa cible, il utilise à nouveau les règles qui déterminent ses actions pour savoir ce qu'il va faire ensuite.

Pour se diriger vers la cible, il est plus logique que ton ami se déplace sur le sol au lieu de voler dans les airs ou de se creuser un chemin sous le sol. Pour cela, tu vas appeler la fonction `mc.getHeight(x,z)` et lui passer une position horizontale (x,z) en paramètres. Elle te renverra alors la coordonnée verticale (y) du premier bloc qui se trouve à la surface de la terre et qui n'est pas de l'`AIR`.

Copyright © 2016 Eyrolles.

Tu peux regarder une vidéo didactique sur la création de ton ami en te rendant sur le site Internet du livre : www.editions-eyrolles.com/dl/14292.



Commence par créer un nouveau programme pour ton ami.

1. Ouvre IDLE, clique sur File>New File pour créer un nouveau programme et enregistre le fichier dans le dossier MyAdventures en le nommant blocAmi.py.
2. Importe les modules minecraft, block, minecraftstuff, math et time en saisissant les lignes suivantes :

```
import mcpi.minecraft as minecraft
import mcpi.block as block
import mcpi.minecraftstuff as minecraftstuff
import math
import time
```

3. Tu vas tout d'abord créer une fonction te permettant de calculer la distance entre deux points. Tu connaîtras ainsi la distance qui te sépare de ton ami :

```
def distanceEntrePoints(point1, point2):
    dx = point2.x - point1.x
    dy = point2.y - point1.y
    dz = point2.z - point1.z
    return math.sqrt((dx*dx) + (dy*dy) + (dz*dz))
```

4. Maintenant, tu dois définir la distance à ne pas dépasser pour que le bloc continue à suivre ton joueur. Crée donc une constante pour enregistrer cette distance maximale. Pour ma part, j'ai choisi le nombre 15, ce qui veut dire que lorsque mon joueur se trouvera à 15 blocs de son ami, il arrêtera aussitôt de le suivre :

```
TROP_LOIN = 15
```

5. Crée les objets Minecraft et MinecraftDrawing :

```
mc = minecraft.Minecraft.create()
mcdrawing = minecraftstuff.MinecraftDrawing(mc)
```

6. Puis crée une variable pour enregistrer l'humeur de ton bloc. Dans ce chapitre, il apprendra à exprimer deux sentiments : sa satisfaction et sa tristesse. Dis-lui d'abord d'être "content" :

```
humeurDuBloc = "content"
```

Copyright © 2016 Eyrolles.

7. Place ton ami à quelques blocs de ton joueur en localisant ton joueur et en ajoutant 5 à sa coordonnée x. Puis appelle la fonction `getHeight()` pour trouver sa coordonnée y et fait en sorte que ton ami ait les pieds sur terre :

```
ami = mc.player.getTilePos()
ami.x = ami.x + 5
ami.y = mc.getHeight(ami.x, ami.z)
mc.setBlock(ami.x, ami.y, ami.z, block.DIAMOND_BLOCK.id)
mc.postToChat("<bloc> Salut l'ami !")
```

8. Crée une cible vers laquelle ton ami se dirigera. Pour cela, tu dois d'abord aligner la position de la cible sur celle de ton ami, afin qu'il reste à sa place lorsque le programme s'exécute :

```
cible = ami.clone()
```



Pour copier des emplacements dans Minecraft, il faut utiliser la fonction `clone()`, car les données de localisation que l'API de Minecraft renvoie sont des objets Python et diffèrent d'une variable normale. Par exemple, si tu avais utilisé le code `cible = ami` et que tu avais changé la valeur `ami.x` dans l'objet `ami`, la valeur `cible.x` aurait également changé dans l'objet `cible`.

9. Rédige une boucle sans fin, afin que ton programme s'exécute en continu. Remarque : tout le code qui se situe après cette boucle doit être indenté.

```
while True:
```

10. Récupère la position de ton joueur et calcule la distance qui le sépare de son ami à l'aide de la fonction `distanceEntrePoints()` :

```
    pos = mc.player.getTilePos()
    distance = distanceEntrePoints(pos, ami)
```

11. Établis les règles à suivre pour définir le comportement de ton ami. S'il est « content », demande-lui de comparer la distance entre la constante « trop loin » et sa propre position. Si la distance est inférieure à la constante, alors la cible doit correspondre à la position de ton joueur. Si elle est supérieure, alors tu dois modifier l'humeur de ton ami parce qu'il est « triste » (voir la figure 8-2) :

```
    if humeurDuBloc == "content":
        if distance < TROP_LOIN:
            cible = pos.clone()
        if distance >= TROP_LOIN:
            humeurDuBloc = "triste"
            mc.postToChat("<bloc> Reviens tu es trop loin. ↩
                          Je veux être avec toi !")
```

Copyright © 2016 Eyrolles.

12. Maintenant, tu dois indiquer à ton programme que si la distance qui sépare ton joueur du bloc ami est supérieure à la constante « trop loin » et si ton ami est « triste », alors il doit attendre que ton joueur se rapproche de son ami pour le faire changer d'humeur et lui faire dire qu'il est « content » à nouveau :

```
    elif humeurDuBloc == "triste":
        if distance <= 1:
            humeurDuBloc = "content"
            mc.postToChat("<bloc> Ah te revoilà ! Allons-y.")
```



FIGURE 8-2 Ton ami est triste.

13. Les lignes de code qui suivent vont faire bouger ton ami jusqu'à sa cible :

```
    if ami != cible:
```

14. Cherche ensuite tous les blocs qui se trouvent entre ton ami et sa cible à l'aide de la fonction `getLine()` dans `MinecraftDrawing` :

```
        blocsEnTravers = mcdrawing.getLine(ami.x, ami.y,
                                           ami.z,
                                           cible.x, cible.y,
                                           cible.z)
```

15. La fonction `getLine()` se comporte de la même manière que `drawLine()` (voir le chapitre 7). Mais au lieu de tracer un segment avec des blocs, elle renvoie une liste de points (x, y, z) qui constituent un segment entre les deux séries de coordonnées x, y, z passées en paramètres.

Copyright © 2016 Eyrolles.

16. Juste en dessous du code précédent, tu dois indiquer à ton programme de passer en boucle tous les blocs situés entre ton ami et sa cible et de faire avancer ton ami vers le joueur en marchant sur chaque bloc du segment qui se trouve entre eux :

```
        for blocEnTravers in blocsEnTravers[:-1]:
            mc.setBlock(ami.x, ami.y, ami.z, block.AIR.id)
            ami = blocEnTravers.clone()
            ami.y = mc.getHeight(ami.x, ami.z)
            mc.setBlock(ami.x, ami.y, ami.z,
                        block.DIAMOND_BLOCK.id)
            time.sleep(0.25)
        cible = ami.clone()
```

À la fin de la boucle `for`, ton ami atteindra sa cible. Alors, pour éviter qu'il se déplace à nouveau, définis la cible sur sa position actuelle.

Le programme déplace ton ami en l'effaçant de sa dernière position, c'est-à-dire en transformant l'identité du bloc où il se trouvait en `AIR`. Puis il actualise la position de ton ami sur le bloc suivant et la fait réapparaître sur cette même position.

Fais bien attention à la différence entre `blocEnTravers` et `blocsEnTravers`. La première variable contient le bloc où se trouve ton ami ; la seconde contient la liste de tous les blocs qui se trouvent entre la position de départ de ton ami et la cible vers laquelle il se dirige.



La vitesse à laquelle se déplace ton ami dépend du temps de veille du programme entre chaque déplacement : `time.sleep(0.25)`. Si cet intervalle est trop court, le bloc se déplacera trop rapidement et ton joueur ne pourra jamais s'en séparer. À l'inverse, s'il est trop long, le bloc se déplacera si lentement qu'il te rendra fou !

17. Ajoute un court délai à la fin de la boucle afin que ton programme ne surcharge pas Minecraft en activant constamment la boucle pour connaître la position de ton joueur :

```
time.sleep(0.25)
```

18. Maintenant, exécute le programme.

Ton ami apparaîtra juste à côté de ton joueur et le suivra jusqu'à ce qu'il considère qu'il est allé trop loin. Lorsque cela se produira, ton ami s'arrêtera net et ton joueur devra faire demi-tour et se placer à côté de lui pour qu'il se remette à le suivre.

Tu peux télécharger le code entier du programme sur le site Internet du livre à l'adresse : `www.editions-eyrolles.com/dl/14292`.

Copyright © 2016 Eyrolles.

PERCER LES SECRETS DU CODE

Pour calculer la distance entre ton ami et ton joueur, tu as créé une fonction que tu as nommée `distanceEntrePoints(point1, point2)`. Elle utilise le théorème de Pythagore pour calculer la distance exacte entre deux points. Tu as peut-être déjà cherché à savoir comment le théorème de Pythagore est appliqué lors du défi « Ajouter un signal lumineux » du chapitre 4.

Si ce n'est pas le cas et pour en avoir le cœur net, voici comment la fonction calcule la distance :

1. Elle soustrait les valeurs x, y et z du `point1` et du `point2` :

```
dx = point2.x - point1.x
dy = point2.y - point1.y
dz = point2.z - point1.z
```

2. Elle calcule ensuite les carrés des différences entre les points :

```
dx*dx
```

3. Puis elle additionne tous ces carrés :

```
(dx*dx) + (dy*dy) + (dz*dz)
```

4. Elle renvoie enfin la différence entre les deux points, qui équivaut à la racine carrée de la somme des carrés. La racine carrée est calculée à l'aide de la fonction Python `math.sqrt()` :

```
return math.sqrt((dx*dx) + (dy*dy) + (dz*dz))
```

Rends-toi sur le site `www.mathsisfun.com/algebra/distance-2-points.html` pour en savoir plus sur le théorème de Pythagore et pour apprendre à t'en servir afin de calculer la distance entre deux points.

Si ton programme parvient à déplacer ton ami vers ton joueur, c'est en calculant le nombre de blocs qui les séparent et en établissant une boucle qui va tous les parcourir :

```
for blocEnTravers in blocsEnTravers[:-1]:
```

Le paramètre `[:-1]` indique à Python de passer en revue tous les blocs de la liste `blocsEnTravers` sauf le dernier. Ainsi, il s'arrête à côté de ton joueur sans le bousculer.

La **racine carrée** d'un nombre est la valeur qui, multipliée par elle-même, permet d'obtenir le nombre en question. Par exemple, la racine carrée de 9 est 3, car 3 x 3 = 9.



Copyright © 2016 Eyrolles.

DÉFI



Ton ami se déplace toujours à la même vitesse, à savoir un bloc toutes les 0,25 seconde, soit 4 b/s (blocs par seconde). Mais un bon ami accélérerait le pas jusqu'à ce qu'il atteigne sa cible. Essaie de modifier ton programme pour que ton ami accélère à mesure qu'il se rapproche de toi.

Retourne dans le chapitre 4 et relève à nouveau le défi « Ajouter un signal lumineux », en utilisant cette fois-ci la fonction `distanceEntrePoints()` pour mieux estimer les distances.

Utiliser les nombres aléatoires pour rendre ton ami plus intéressant

L'ennui avec le programme que tu viens de créer, c'est que ton ami est trop **prévisible** ; il agit toujours de la même manière. Après quelques essais, tu t'habitueras à son comportement et pourras prédire ses moindres faits et gestes, ce qui deviendra vite ennuyeux. Pour que ton ami agisse comme un être vivant, il faut que tu le rendes un peu moins prévisible.



On dit que quelque chose est **prévisible** lorsqu'il est possible d'anticiper les événements qui vont se produire. Or, cela peut vite devenir un problème si tu souhaites faire comme dans la vraie vie, car les événements ne sont pas toujours prévisibles.

En utilisant les nombres aléatoires, tu as la possibilité de provoquer l'inattendu, c'est-à-dire que tu peux conférer à un objet la capacité d'agir de manière imprévisible. Pour cela, tu dois développer ton programme de façon à ce qu'il agisse selon des **probabilités**. Par exemple, tu peux lui demander d'exécuter une action toutes les 100 fois. Et en ajoutant des règles à exécuter à des intervalles différents, tu peux rendre ton programme encore plus imprévisible. Mais avant de reprogrammer ton ami avec des nombres aléatoires, penchons-nous sur le code qui permet de créer des nombres aléatoires et de vérifier quelques probabilités. Comme tu as déjà eu l'occasion de travailler avec les nombres aléatoires dans le chapitre 3, tu pourras ainsi te rafraîchir la mémoire.

Copyright © 2016 Eyrolles.

Une **probabilité** est un calcul qui permet de connaître la fréquence d'un événement. Par exemple, lorsqu'on joue à pile ou face, il y a 50 % de chances (1 sur 2) de tomber sur le côté face.



Le module Python `random` contient une fonction `random.randint (startNumber, endNumber)` qui est utilisée pour produire un nombre au hasard compris entre deux nombres donnés `startNumber` et `endNumber`.

Le code qui suit choisira un nombre au hasard entre 1 et 10 chaque fois que tu l'exécuteras. Si tu veux voir les résultats, tu devrais créer un nouveau programme Python :

```
import random
noAleatoire = random.randint(1,10)
print noAleatoire
```

En ajoutant une instruction `if` pour savoir lorsqu'un nombre équivaut à 10, tu crées une probabilité qui se vérifiera environ 1 fois sur 10 :

```
import random
if random.randint(1,10) == 10:
    print "Cela se produit environ une fois sur dix"
else :
    print "Cela se produit environ neuf fois sur dix"
```

Si tu exécutes le programme plus de 100 fois, tu t'attends à voir l'expression « Cela se produit environ une fois sur dix » s'afficher environ 10 fois (voir la figure 8-3). Mais il se peut que tu ne la voies apparaître que 9 ou 11 fois, voire pas du tout. C'est tout à fait imprévisible !

Si tu exécutais ce programme 100 fois, combien de fois verrais-tu apparaître la phrase « Cela se produit environ une fois sur dix » ? Il se peut que tu la voies s'afficher 10 fois, mais il se peut aussi que tu ne la voies pas s'afficher du tout, ou encore qu'elle apparaisse 100 fois !



Si tu utilises des nombres aléatoires et des probabilités dans le programme de ton ami, tu peux le rendre moins prévisible. Tu peux même aller jusqu'à rompre vos liens d'amitié !

Copyright © 2016 Eyrolles.

Ajoute des règles au programme de ton ami pour qu'il puisse décider, après avoir essuyé plusieurs déceptions, de ne plus suivre ton joueur, même s'il revient le voir et essaye de se réconcilier avec lui. Tu auras donc 1 chance sur 100 de ne plus le revoir s'il est « triste ».

1. Lance IDLE et ouvre le programme `blocAmi.py` qui se trouve dans le dossier `MyAdventures`.
2. Clique sur `File>Save As` et enregistre ton fichier en le nommant `blocAmiAleatoire.py`.

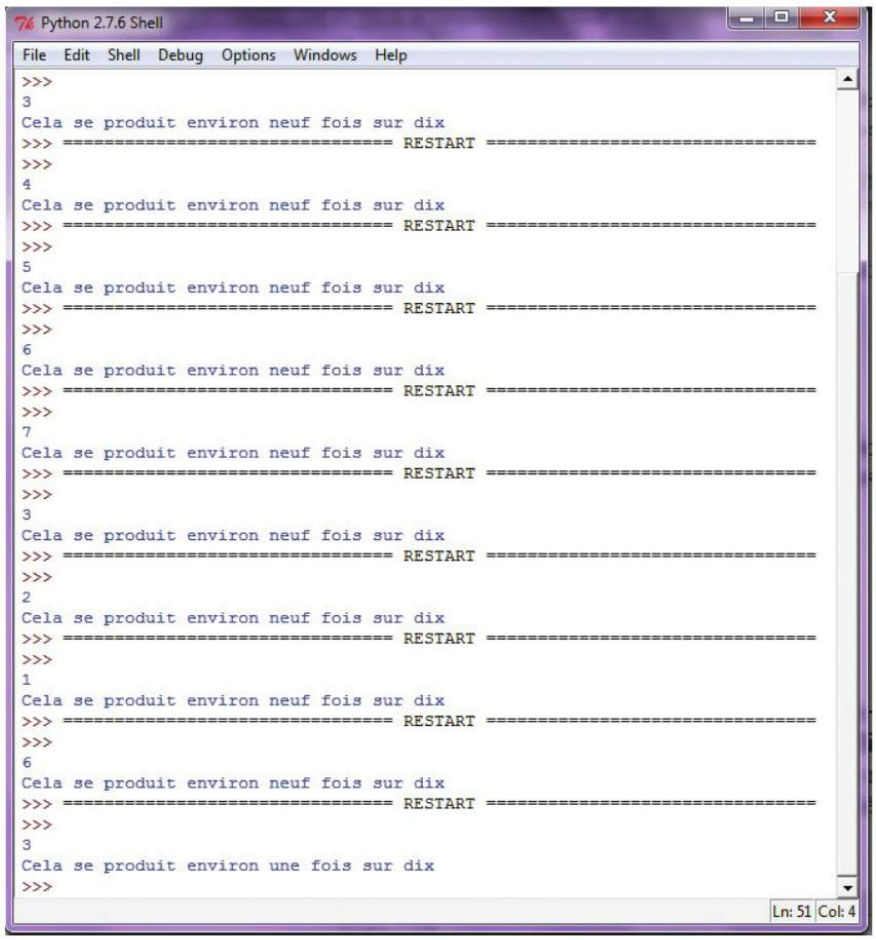


FIGURE 8-3 Les nombres aléatoires rendent ton ami plus imprévisible.

3. Ajoute le module `random` dans les instructions `import` au tout début du programme (il s'agit du code en **gras**) :

```
import mcpi.minecraft as minecraft
import mcpi.block as block
import mcpi.minecraftstuff as minecraftstuff
import math
import time
import random
```

Copyright © 2016 Eyrolles.

4. Ajoute un test de nombre aléatoire pour que ton bloc change d'humeur 1 fois sur 100 et déclare «ça suffit» lorsqu'il est «triste» :

```
    elif humeurDuBloc == "triste":
        if distance <= 1:
            humeurDuBloc = "content"
            mc.postToChat("<bloc> Ah, te revoilà ! Allons-y.")
        if random.randint(1,100) == 100:
            humeurDuBloc = "çasuffit"
            mc.postToChat("<block> Ça suffit! J'en ai marre!")
```

5. À présent, exécute le programme.

Si tu t'éloignes de ton ami et que tu restes suffisamment longtemps à distance, il finira par se lasser et ne considèrera plus ton joueur comme un ami. Dans ce cas, il dira : « Ça suffit ! J'en ai marre ! » et restera assis au même endroit sans bouger.

DÉFI

Modifie encore le programme pour que ton joueur ait 1 chance sur 50 de se faire pardonner s'il essaye de se réconcilier avec son ami alors qu'il est d'humeur «ça suffit».



Personnaliser un objet plus gros

Pour programmer ton ami, tu as utilisé un bloc que tu as doté de la capacité de se mouvoir et de suivre ton joueur. Mais comment t'y prendrais-tu pour faire bouger plusieurs blocs en même temps ou pour déplacer des objets plus volumineux comme une voiture ou un vaisseau spatial ?

C'est maintenant que les choses se corsent, car tu vas devoir suivre plusieurs blocs en même temps. Et chaque fois que tu voudras déplacer ton objet, tu devras transformer tous les blocs en blocs d'`AIR` avant de les faire réapparaître dans une nouvelle position. De plus, si ton objet comporte de nombreux blocs, il peinera à se déplacer correctement et avancera à une allure d'escargot.

Le module `minecraftstuff` contient une fonction intitulée `MinecraftShape`, qui a été écrite spécialement pour créer des objets mouvants. Elle y parvient en gardant la trace de tous les blocs de l'objet et, lorsqu'il se déplace, elle ne modifie que l'état de ceux qui ont bougé.

Copyright © 2016 Eyrolles.

Pour utiliser `MinecraftShape`, tu dois indiquer à ton programme à quoi ressemble l'objet en créant une liste des blocs qui le composent. Chaque bloc a une position précise (x, y, z) et une identité.

Sur la figure 8-4, tu peux voir un objet simple composé de sept blocs, dont les coordonnées sont indiquées en blanc. Ici, le terme « coordonnée » n'a pas le même sens que celui des coordonnées Minecraft que tu connais déjà. Comme tu peux le voir, les coordonnées du centre du cheval en bois sont 0,0,0 et celles des autres blocs sont définies par rapport au centre, d'où les coordonnées 1,0,0 pour le bloc qui se trouve juste à droite du centre.



FIGURE 8-4 Positionnement des blocs d'un objet Minecraft en forme de cheval

Tu vas maintenant créer un programme qui fait appel à la fonction `MinecraftShape` pour créer le cheval en bois de la figure 8-4 et le faire bouger :

1. Ouvre IDLE et clique sur `File>New File` pour créer un nouveau programme. Nomme-le `chevalEnBois.py` et enregistre-le dans le dossier `MyAdventures`.
2. Importe les modules `minecraft`, `block`, `minecraftstuff` et `time` :

```
import mcpi.minecraft as minecraft
import mcpi.block as block
import mcpi.minecraftstuff as minecraftstuff
import time
```

3. Crée l'objet `Minecraft` :

```
mc = minecraft.Minecraft.create()
```

4. Pour fabriquer ton cheval en bois, crée une liste de blocs. Utilise `ShapeBlock` pour leur attribuer des coordonnées relatives et un type de bloc :

Copyright © 2016 Eyrolles.

```
horseBlocks = [minecraftstuff.ShapeBlock(0,0,0,
                                         block.WOOD_PLANKS.id),
               minecraftstuff.ShapeBlock(-1,0,0,
                                         block.WOOD_PLANKS.id),
               minecraftstuff.ShapeBlock(1,0,0,
                                         block.WOOD_PLANKS.id),
               minecraftstuff.ShapeBlock(-1,-1,0,
                                         block.WOOD_PLANKS.id),
               minecraftstuff.ShapeBlock(1,-1,0,
                                         block.WOOD_PLANKS.id),
               minecraftstuff.ShapeBlock(1,1,0,
                                         block.WOOD_PLANKS.id),
               minecraftstuff.ShapeBlock(2,1,0,
                                         block.WOOD_PLANKS.id)]
```

Les coordonnées des blocs sont les mêmes que sur la figure 8-4.

5. Tu dois indiquer à `MinecraftShape` où créer le cheval en bois dans Minecraft. Récupère la position de ton joueur et ajoute 1 à ses coordonnées z et y pour que ton cheval ne lui tombe pas sur la tête :

```
horsePos = mc.player.getTilePos()
horsePos = horsePos.z + 1
horsePos = horsePos.y + 1
```

6. Crée ton cheval en te servant de `MinecraftShape`, et passe en paramètre de ton objet Minecraft les coordonnées de l'endroit où tu souhaites faire apparaître ton cheval ainsi que la liste des blocs qui le composent (`ShapeBlocks`) :

```
horseShape= minecraftstuff.MinecraftShape(mc, horsePos,
                                          horseBlocks)
```

7. Exécute le programme. Et voilà ! Tu devrais voir un cheval en bois apparaître à côté de ton joueur.

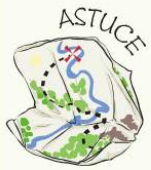
8. Modifie ton programme `chevalEnBois.py` pour que ton cheval puisse se déplacer, en ajoutant le code suivant à la fin du script :

```
for count in range(1,10):
    time.sleep(1)
    horseShape.moveBy(1,0,0)
horseShape.clear()
```

9. La fonction `moveBy(x,y,z)` (en français, « avance de ») indique à ton cheval d'avancer de tant de blocs, en fonction des coordonnées x, y et z passées en paramètre. Dans ce cas, `horseShape` avance d'un bloc sur l'axe (x). Quant à la fonction `clear()`, elle efface le cheval en transformant tous les blocs qui le composent en `AIR`.

10. Exécute ton programme et regarde ton cheval en bois galoper !

Copyright © 2016 Eyrolles.



Comme avec `moveBy(x,y,z)` et `clear()`, tu peux aussi demander à un objet d'avancer vers un endroit précis dans Minecraft, à l'aide de la fonction `move(x,y,z)`. De même, si l'objet a été effacé, tu peux le recréer à l'aide de la fonction `draw()`.

Tu peux télécharger le code entier de ce cheval en bois sur le site Internet du livre à cette adresse : `www.editions-eyrolles.com/dl/14292`.

Ce que tu viens de voir va beaucoup t'aider, car tu vas maintenant te lancer dans la construction d'un vaisseau spatial ! Plus tard, tu utiliseras ces objets à nouveau pour créer des obstacles.

Invasion d'extraterrestres

Des extraterrestres menacent d'envahir Minecraft. Un vaisseau spatial va descendre du ciel et léviter juste au-dessus de ton joueur, qui se trouvera alors en danger de mort. Ces extraterrestres ne viennent pas en tant qu'amis et ils ne renonceront pas avant d'être parvenus à leurs fins !

Dans le programme qui va suivre, tu vas te servir de `MinecraftShape` et des mêmes techniques de programmation que pour la création de ton ami en vue de confectionner un vaisseau spatial extraterrestre (voir la figure 8-5) qui sondera la surface de la terre à la poursuite de ton joueur pour essayer de léviter au-dessus de sa tête. Lorsqu'il y parviendra, il aspirera ton joueur pour le téléporter à bord !



FIGURE 8-5 Fabrique un vaisseau spatial pour pimenter le jeu !

Copyright © 2016 Eyrolles.

Comme pour le programme du cheval en bois, tu vas utiliser `MinecraftShape` pour créer ton vaisseau spatial en indiquant les coordonnées relatives de chaque bloc qui le constitue ainsi que leur type. Sur la figure 8-6, tu peux voir les coordonnées des blocs du vaisseau, vu de haut et de face.



FIGURE 8-6 Les coordonnées des blocs du vaisseau extraterrestre

L'exercice du vaisseau spatial t'apprend à créer des figures en 3D à l'aide de l'objet `MinecraftShape`. Grâce à cet objet, tu peux aussi bien créer des petites formes simples que de grandes figures complexes. Il te laisse toute la liberté d'utiliser des formes différentes.



Comme le programme que tu as écrit pour te créer un ami, le code que tu rédigeras pour simuler l'invasion extraterrestre comporte deux parties. La première contient les règles qui vont déterminer le comportement du vaisseau spatial et la deuxième correspond aux commandes qui vont conduire le vaisseau à ton joueur.

Lorsque le vaisseau prendra ton joueur en chasse, il se moquera de lui en publiant des messages sarcastiques dans le chat, tels que : « Tu ne vas pas passer ta vie à courir ». Ces messages sont piochés au hasard dans une liste de moqueries (voir la figure 8-7).

Les règles qui définissent le comportement du vaisseau extraterrestre seront exécutées selon trois modes différents :

- **Atterrissage :** ce mode initial s'exécute au démarrage du programme ; il commande au vaisseau spatial de descendre du ciel et de se poser au-dessus de ton joueur ;

Copyright © 2016 Eyrolles.

- **Attaque :** une fois l'atterrissage terminé, le vaisseau passera à l'attaque et poursuivra ton joueur jusqu'à ce qu'il parvienne à se caler juste au-dessus de sa tête pour le hisser à bord ;
- **Mission accomplie :** ce mode s'active dès que ton joueur est hissé à bord du vaisseau ; à ce moment-là, les extraterrestres renverront ton joueur sur terre, signant ainsi la fin du programme.

Dès que le vaisseau spatial aura capturé ton joueur, le programme construira un cachot et le jettera à l'intérieur en modifiant ses données de position (voir la figure 8-8). Les extraterrestres enverront alors un message au joueur avant de le renvoyer sur terre à l'endroit de sa capture et de faire disparaître le cachot.



FIGURE 8-7 Le vaisseau spatial te prend en chasse.

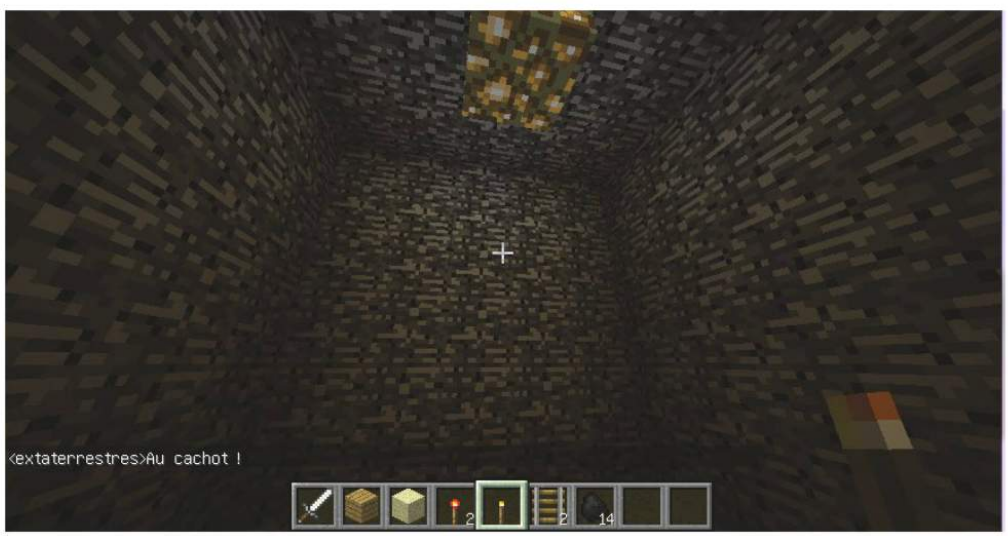


FIGURE 8-8 À l'intérieur du vaisseau spatial

Copyright © 2016 Eyrolles.

Suis ces étapes pas à pas pour écrire ton programme d'invasion extraterrestre.

1. Ouvre IDLE, clique sur `New>New File`, nomme ton fichier `invasionExtraterrestre.py` et enregistre-le dans le dossier `MyAdventures`.

2. Importe les modules `minecraft`, `block`, `minecraftstuff` et `time` :

```
import mcpi.minecraft as minecraft
import mcpi.block as block
import mcpi.minecraftstuff as minecraftstuff
import time
```

3. Crée la fonction `distanceEntrePoints()` :

```
def distanceEntrePoints(point1, point2):
    dx = point2.x - point1.x
    dy = point2.y - point1.y
    dz = point2.z - point1.z
    return math.sqrt((dx*dx) + (dy*dy) + (dz*dz))
```

4. Crée les constantes dont tu vas avoir besoin dans ton script. `HAUTEUR_DU_PLANE` correspond au nombre de blocs qui sépareront le vaisseau spatial de ton joueur lorsqu'il planera au-dessus de sa tête ; `MOQUERIES_DES_EXTRATERRESTRES` contient une liste des commentaires sarcastiques que les extraterrestres publieront sur le chat lorsqu'ils prendront ton joueur en chasse :

```
HAUTEUR_DU_PLANE = 15
MOQUERIES_DES_EXTRATERRESTRES = ["<extraterrestres>Tu ne vas
                                 pas passer ta vie a courir",
                                 "<extraterrestres>Inutile
                                 de résister",
                                 "<extraterrestres>Nous
                                 voulons juste être ton ami"]
```

   N'hésite pas à faire preuve d'imagination : modifie les railleries ou ajoutes-en d'autres !

5. Crée les objets `Minecraft` et `MinecraftDrawing` :

```
mc = minecraft.Minecraft.create()
mcdrawing = minecraftstuff.MinecraftDrawing(mc)
```

6. Commence par définir le point de départ des extraterrestres et leur mode d'action, à savoir 50 blocs au-dessus du joueur en mode « atterrissage » :

```
posExtraterrestres = mc.player.getTilePos()
posExtraterrestres.y = posExtraterrestres.y + 50
mode = "atterrissage"
```

7. Bâtis le vaisseau extraterrestre en te servant de `MinecraftShape` (réfère-toi à la figure 8-6 pour te rappeler comment faire) :

Copyright © 2016 Eyrolles.

```
blocsExtraterrestres =
  [minecraftstuff.ShapeBlock(-1,0,0,block.WOOL.id, 5),
   minecraftstuff.ShapeBlock(0,0,-1,block.WOOL.id, 5),
   minecraftstuff.ShapeBlock(1,0,0,block.WOOL.id, 5),
   minecraftstuff.ShapeBlock(0,0,1,block.WOOL.id, 5),
   minecraftstuff.ShapeBlock(0,-1,0,block.GLOWSTONE_BLOCK.id),
   minecraftstuff.ShapeBlock(0,1,0,block.GLOWSTONE_BLOCK.id)]
vaisseauExtraterrestre = minecraftstuff.MinecraftShape(mc, ↩
posExtraterrestres, blocsExtraterrestres)
```

8. Crée ensuite une boucle `while`, qui tournera jusqu'à ce que le vaisseau passe en mode `«missionaccomplie»` ou, pour le dire autrement, qui terminera sa course lorsque le programme passera au mode `«missionaccomplie»` :

```
while mode != "Mission accomplie":
```

9. Récupère la position de ton joueur à chaque tour de boucle :

```
    posJoueur = mc.player.getTilePos()
```

10. Le prochain bloc de code correspond aux règles que le programme va suivre pour déterminer l'action du vaisseau. Si le programme est en mode `«atterrissage»`, définis la cible juste au-dessus de la position de ton joueur et passe en mode `«attaque»` :

```
    if mode == "atterrissage":
        mc.postToChat("<extraterrestres> Nous ne venons pas
                      en amis alors sois gentil et panique !")
        cibleExtraterrestre = posJoueur.clone()
        cibleExtraterrestre.y = cibleExtraterrestre.y
                                + HAUTEUR_DU_PLANE
        mode = "attaque"
```

11. Si le programme est déjà en mode `«attaque»`, demande-lui de vérifier si le vaisseau spatial se trouve au-dessus de ton joueur. Si c'est le cas, hisse-le à bord du vaisseau et passe en mode `«missionaccomplie»`. Mais si le joueur a pris les jambes à son cou, alors définis la cible sur sa nouvelle position et publie un message sur le chat :

```
    elif mode == "attaque":
        # vérifie si le vaisseau se trouve bien au-dessus du
        joueur
        if posExtraterrestres.x == posJoueur.x and
           posExtraterrestres.z == posJoueur.z:
            mc.postToChat("<extaterrestres>Au cachot !")
            #construis un cachot
            mc.setBlocks(0,50,0,6,56,6,block.BEDROCK.id)
            mc.setBlocks(1,51,1,5,55,5,block.AIR.id)
            mc.setBlock(3,55,3,block.GLOWSTONE_BLOCK.id)
```

Copyright © 2016 Eyrolles.

```
            # hisse le joueur à l'intérieur
            mc.player.setTilePos(3,51,5)
            time.sleep(10)
            mc.postToChat("<aliens>Cet humanoïde n'a
                          vraiment aucun intérêt.
                          Renvoyez-le sur Terre !")
            time.sleep(2)
            # renvoie le joueur là où il se trouvait avant
            mc.player.setTilePos(posJoueur.x, posJoueur.y,
                                 posJoueur.z)
            # fait disparaître le cachot
            mc.setBlocks(0,50,0,6,56,6,block.AIR.id)
            mode = "Mission accomplie"
        else:
            # le joueur s'est enfui
            mc.postToChat(MOQUERIES_DES_EXTRATERRESTRES
                          [random.randint(0,len(MOQUERIES_
                          DES_EXTRATERRESTRES)-1)])
            cibleExtraterrestre = posJoueur.clone()
            cibleExtraterrestre.y = cibleExtraterrestre.y
                                    + HAUTEUR_DU_PLANE
```

Lorsque ton joueur est hissé à bord du vaisseau, il est jeté dans un cachot qui est construit 50 blocs au-dessus de l'endroit où le vaisseau a fait son apparition. Cela signifie que sa position est modifiée pour qu'il apparaisse à l'intérieur de la salle. C'est un peu comme dans la série Doctor Who : son TARDIS est bien plus grand qu'il n'y paraît ! Des messages sont ensuite publiés sur le chat avant que la position du joueur ne soit réinitialisée et que le cachot ne disparaisse.



Si le cachot est construit dans le ciel, juste au-dessus du lieu d'apparition du vaisseau, c'est avant tout par commodité. Cela dit, tu pourrais tout à fait le construire ailleurs, car dès que le joueur est renvoyé sur Terre, le cachot disparaît et tout redevient comme avant.

12. Si la position du vaisseau spatial ne correspond pas à la cible (comme dans la règle précédente), conduis le vaisseau jusqu'à la cible en saisissant ce code à la fin du script, que tu indenteras sous la boucle `while` :

```
    if posExtraterrestres != cibleExtraterrestre:
        blocsEnTravers = mcdrawing.getLine
                         (posExtraterrestres.x,
                         posExtraterrestres.y,
                         posExtraterrestres.z,
                         cibleExtraterrestre.x,
```

Copyright © 2016 Eyrolles.

```
                        cibleExtraterrestre.y,
                        cibleExtraterrestre.z)
    for blocEnTravers in blocsEnTravers:
        vaisseauExtraterrestre.move
                    (blocEnTravers.x, blocEnTravers.y,
                     blocEnTravers.z)
        time.sleep(0.25)
    posExtraterrestres = cibleExtraterrestre.clone()
```

13. À partir de cette ligne, le programme reviendra au début de la boucle `while` et lorsqu'il passera en mode `«mission accomplie»` et que la boucle sera terminée, il lira la dernière ligne du script pour faire disparaître le vaisseau :

```
vaisseauExtraterrestre.clear()
```

14. Exécute le programme et tiens-toi prêt à voir des extraterrestres envahir l'horizon !

Tu peux télécharger le code entier de l'invasion extraterrestre sur le site Internet du livre à l'adresse : `www.editions-eyrolles.com/dl/14292`.

PERCER LES SECRETS DU CODE

Chaque fois que le joueur est pris en chasse par les extraterrestres, un message sarcastique est choisi au hasard dans la constante `MOQUERIES_DES_EXTRATERRESTRES` et publié sur le chat.

```
mc.postToChat(MOQUERIES_DES_EXTRATERRESTRES[
  random.randint(0,len(MOQUERIES_DES_EXTRATERRESTRES)-1)])
```

`MOQUERIES_DES_EXTRATERRESTRES` est une liste de chaînes de caractères. La fonction `random.randint()` est utilisée pour choisir un élément dans une liste en générant un nombre entre 0 et le nombre de `MOQUERIES_DES_EXTRATERRESTRES` moins 1.

Si le chiffre 1 est soustrait au nombre de messages de la liste, la fonction `len()` renvoie le nombre d'éléments qu'elle contient (c'est-à-dire 3), mais tu ne dois pas oublier de référencer les éléments de la liste en commençant par 0 (soit 0,1,2).

DÉFI



Le vaisseau extraterrestre que nous avons créé est vraiment très simple. Essaye d'en confectionner un autre plus impressionnant et plus réaliste. Modifie ensuite le programme pour qu'il passe en mode « espion » lorsqu'il apparaît dans l'atmosphère. Il lévitera ainsi non loin de ton joueur et restera à l'affût jusqu'à ce qu'il décide de passer à l'attaque sans prévenir, de manière tout à fait aléatoire.

Copyright © 2016 Eyrolles.

Explorer d'autres pistes dans le domaine de la simulation

Dans ce chapitre, tu t'es servi d'algorithmes et de règles pour simuler un ami et une invasion d'extraterrestres. Tu pourrais pousser l'expérience un peu plus loin en simulant d'autres événements, tels que des volées d'oiseaux (ou de blocs), des vagues à la surface des océans de Minecraft ou encore un automate cellulaire créé à partir de blocs, comme dans le jeu de la vie de John Horton Conway.

Pour en savoir plus sur le jeu de la vie de Conway, consulte la page Wikipédia : https://fr.wikipedia.org/wiki/Jeu_de_la_vie.

Tableau de références

Commandes	Description
`import random`	Importe le module Python `random`.
`random.randint(début, fin)`	Crée un nombre aléatoire situé entre le premier et le dernier nombre d'une liste.
`import minecraftstuff`	Importe le module d'extensions `minecraftstuff` qui se trouve dans le kit de démarrage.
`mc.getHeight(x,z)`	Récupère la coordonnée de hauteur (y) du sol, c'est-à-dire le premier bloc qui se trouve à la surface de la terre, sur un axe horizontal (x,y) donné, et qui n'est pas composé d'`AIR`.
`mcdrawing = minecraftstuff. ↩ MinecraftDrawing(mc)`	Crée l'objet `MinecraftDrawing`.
`copieDePosition = position.clone()`	Crée une copie (un clone) d'une position Minecraft.
`mcdrawing.getLine(x1, y1, z1, x2, y2, z2)`	Récupère tous les blocs d'un segment.
`shapeBlocks = [ minecraftstuff.ShapeBlock(1,0,0, block.DIRT.id), minecraftstuff.ShapeBlock(0,0,1, block.DIRT.id)]`	Crée une liste de blocs `shapeBlocks` qui forment une figure `MinecraftShape`.
`shape = minecraftstuff. ↩ MinecraftShape(mc, pos, shapeBlocks)`	Crée une forme `MinecraftShape` à partir des blocs `shapeBlocks` passés en paramètre à une position donnée.
`shape.moveBy(x,y,z)`	Déplace une figure `MinecraftShape` en fonction des valeurs indiquées dans les coordonnées x, y et z.
`shape.move(x,y,z)`	Conduit une figure `MinecraftShape` à une position x,y,z.
`shape.clear()`	Efface la figure `MinecraftShape`.
`shape.draw()`	Dessine la figure `MinecraftShape`.

Copyright © 2016 Eyrolles.

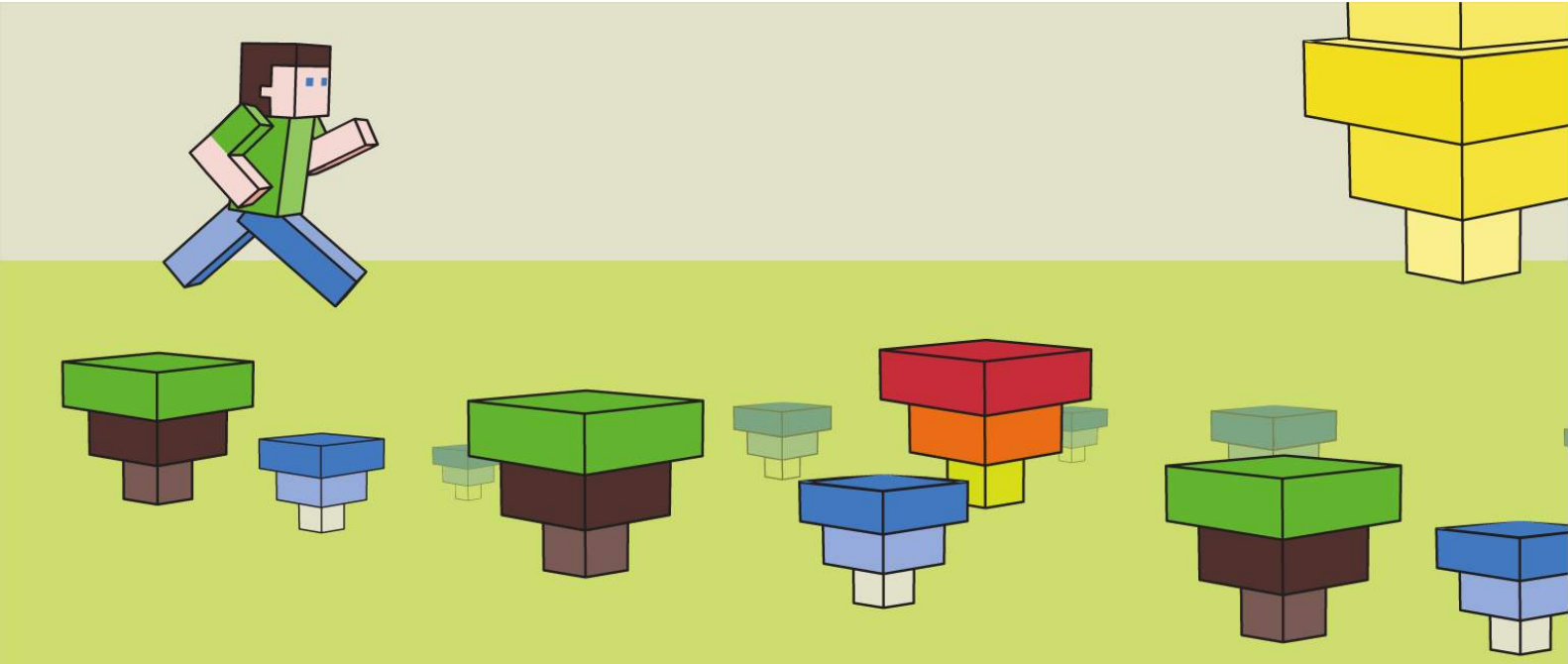


Niveau terminé : tu t'es fait kidnapper par tes propres automates !

Dans le prochain chapitre...

Dans le chapitre 9, tu vas utiliser toutes les compétences que tu as acquises pour créer un super jeu ! Tu vas organiser une course contre la montre pour récupérer des diamants. Mais attention ! Ton chemin sera semé d'embûches et certains objets malintentionnés seront déterminés à te barrer la route !

Copyright © 2016 Eyrolles.



Chapitre 9
La grande aventure

TE VOILÀ ENFIN prêt pour la grande aventure ! Tu vas maintenant pouvoir utiliser toutes les connaissances que tu as acquises pendant ton périple pour créer un mini-jeu entièrement fonctionnel dans Minecraft. Tu verras à quel point le monde de Minecraft est riche. Et tu te rendras compte qu'il est possible de réaliser des choses incroyables rien qu'en prenant et en reposant un bloc ou en localisant ton joueur dans le jeu.

Tu apprendras également à maîtriser la programmation parallèle pour exécuter plusieurs codes à la fois.

Ce projet peut s'étendre à l'infini. Et même si tu le termines, cela ne signifie pas pour autant que l'aventure s'arrête là. Bien au contraire, une fois que tu en seras venu à bout, c'est tout un univers de jeux encore plus stimulants qui s'ouvrira à toi !

Un jeu dans le jeu

Le jeu que tu vas créer dans ce chapitre s'appelle « Traversée périlleuse ». Le but du jeu est de récupérer le plus de diamants possible et de traverser l'arène avant la fin du compte à rebours, en évitant les nombreux pièges qui jonchent le sol et qui ont été conçus pour te ralentir.

Copyright © 2016 Eyrolles.

Chaque diamant que tu ajoutes à ton inventaire te rapporte des points. Une fois que tu auras traversé l'arène, ces points seront multipliés par le nombre de secondes qu'il te reste au compteur.

Pour traverser l'arène le plus vite possible, ton joueur devra franchir une rivière en sautant sur une plate-forme mouvante, passer sous un mur oscillant de bas en haut sans se faire écraser et éviter les trous qui apparaîtront aléatoirement sur le sol (voir la figure 9-1).



FIGURE 9-1 Lance-toi dans cette traversée périlleuse !



Tu trouveras une vidéo du jeu de la traversée périlleuse sur le site Internet du livre : `www.editions-eyrolles.com/dl/14292`.

Créer un jeu est un projet qui requiert beaucoup d'attention. Avant de commencer à jouer, tu dois construire, tester et assembler de nombreux éléments. Pour te faciliter la tâche, j'ai divisé les instructions en quatre parties, afin que tu puisses développer et tester ton programme pas à pas.

- Première partie : élaboration de la structure principale du jeu et construction de l'arène où se déroulera l'action.
- Deuxième partie : programmation des obstacles qui se dresseront devant le joueur dans le but de le ralentir.
- Troisième partie : introduction du concept de jouabilité dans le programme, avec l'ajout de niveaux de difficulté, d'un système de points et, bien sûr, de l'incontournable « Game over ».

Copyright © 2016 Eyrolles.

- Quatrième partie : utilisation des connaissances acquises dans le chapitre 5 et réutilisation du bouton-poussoir et de l'afficheur 7 segments pour créer un bouton de démarrage, un compteur de diamants ainsi qu'un compte à rebours.

Une fois que ton jeu sera terminé, tu pourras le développer à ta façon et t'amuser à le transformer pour te l'approprier davantage.

> Tu trouveras d'ailleurs dans ce chapitre toute une série de défis à réaliser pour compléter l'aventure. Attends toutefois de terminer la création de ton jeu pour les relever, car tu risquerais sérieusement de t'emmêler les pinceaux ! Tu auras tout le loisir de revenir dessus une fois que tu auras terminé la construction du jeu.



Première partie : construire l'arène

L'arène du jeu Traversée périlleuse est le lieu où toute l'action va se dérouler. Le joueur se trouve d'un côté et doit essayer d'atteindre l'autre côté en évitant les obstacles qui jonchent le sol et le ralentissent.

Sans les obstacles, l'arène est un rectangle tapissé de blocs d'`HERBE` et cerné de murs en `VERRE` (voir la figure 9-2). Pour en définir la largeur, la hauteur et la longueur (c'est-à-dire les axes x, y et z), tu vas utiliser des constantes. En modifiant les constantes, tu modifieras les dimensions et la forme de l'arène, et les obstacles s'adapteront automatiquement à l'espace restant. Quant au sol de l'arène, il devra être profond de trois blocs, car tu dessineras le lit de la rivière sur une profondeur de deux blocs.



FIGURE 9-2 Ton arène ressemblera à cela.

Copyright © 2016 Eyrolles.

Souviens-toi de la fonction `setBlocks()` que tu as utilisée pour construire une maison lors du chapitre 3. Tu vas l'utiliser ici pour construire ton arène.

Lance Minecraft, IDLE et le serveur Bukkit si tu es sur un PC ou un Mac. Tu t'es déjà certainement un peu familiarisé avec le processus, mais n'hésite pas à relire le chapitre 1 si tu as besoin de te rafraîchir la mémoire.

Commence par créer un nouveau programme pour ton jeu « Traversée périlleuse ». La première étape consiste à bâtir la structure du jeu.

1. Ouvre IDLE, crée un nouveau fichier, nomme-le `traverseePerilleuse.py` et enregistre-le dans le dossier `MyAdventures`.

2. Importe les modules nécessaires. Tu vas notamment utiliser un nouveau module, `thread`, que tu apprendras à connaître tout à l'heure lorsque tu construiras les obstacles de la deuxième partie :

```
import mcpi.minecraft as minecraft
import mcpi.block as block
import mcpi.minecraftstuff as minecraftstuff
import time
import random
import thread
```

3. Crée trois constantes x, y et z correspondant à la largeur, à la hauteur et à la longueur de ton arène :

```
ARENEX = 10
ARENEZ = 20
ARENEY = 3
```

4. Insère ensuite les fonctions de ton programme. Tu les développeras tout au long du chapitre, jusqu'à ce que tu termines le script de ton jeu :

```
def creerArene(pos):
    pass
def leMur(posArene, posZMur):
    pass
def laRiviere(posArene, posRiviereZ):
    pass
def lesTrous(posArene, posZTrous):
    pass
def poserDiamants(posArene, nombre):
    pass
```

   L'instruction `pass` ne sert qu'à indiquer que tu la remplaceras par du code par la suite.

5. Établis la connexion avec Minecraft :

```
mc = minecraft.Minecraft.create()
```

Copyright © 2016 Eyrolles.

6. Crée une variable booléenne qui deviendra `True` à la fin, mais qui restera `False` pendant toute la durée du jeu.

```
gameOver = False
```

Par la suite, tu ajouteras du code au programme principal et compléteras les fonctions que tu viens de créer.

> Tu peux essayer d'exécuter le programme maintenant. Tu verras qu'il ne se passe rien ! Il est toutefois utile de l'exécuter en l'état, car si aucune erreur ne s'affiche dans l'interpréteur de commandes Python, cela veut dire que ton script est correct.
>
> 
>

L'étape suivante consiste à mettre à jour la fonction `creerArene`, celle qui te permet de construire ton arène dans Minecraft.

1. Commence par rechercher la fonction `creerArene` dans le script que tu viens d'écrire en effectuant une recherche sur le code suivant :

```
def creerArene(pos):
    pass
```

   Supprime l'instruction `pass` qui est indentée en dessous de la fonction.

2. En l'indentant sous la ligne `def creerArene(pos):`, écris une ligne de code pour établir la connexion avec Minecraft :

```
    mc = minecraft.Minecraft.create()
```

3. Tu peux maintenant construire l'arène à l'aide de `setBlocks` en utilisant la position `(pos)` passée dans la fonction et en ajoutant les constantes `ARENEX`, `ARENEY` et `ARENEZ` que tu as définies plus haut (la figure 9-3 montre comment les constantes ont été ajoutées à la position pour créer l'arène) :

```
    mc.setBlocks(pos.x - 1 , pos.y, pos.z - 1,
                 pos.x + ARENEX + 1, pos.y - 3,
                 pos.z + ARENEZ + 1,
                 block.GRASS.id)
```

4. Pour créer les murs en verre de ton arène, construis un parallélépipède avec des blocs de type `VERRE` puis imbrique à l'intérieur un autre parallélépipède en blocs de type `AIR` :

```
    mc.setBlocks(pos.x - 1, pos.y + 1, pos.z - 1,
                 pos.x + ARENEX + 1, pos.y + ARENEY,
                 pos.z + ARENEZ + 1,
```

Copyright © 2016 Eyrolles.

```
                block.GLASS.id)
    mc.setBlocks(pos.x, pos.y + 1, pos.z,
                pos.x + ARENEX, pos.y + ARENEY,
                pos.z + ARENEZ,
                block.AIR.id)
```

5. La fonction `creerArene()` est maintenant terminée, mais il faut encore que ton programme principal l'appelle. Pour cela, ajoute les lignes suivantes à la fin du programme pour localiser ton joueur et affecter sa position à une variable que tu passeras en paramètre de la fonction `creerArene()` :

```
posArene = mc.player.getTilePos()
creerArene(posArene)
```

6. Il est temps d'exécuter ton programme ! Tu devrais voir l'arène se construire juste à côté de ton joueur, comme dans la figure 9-3.

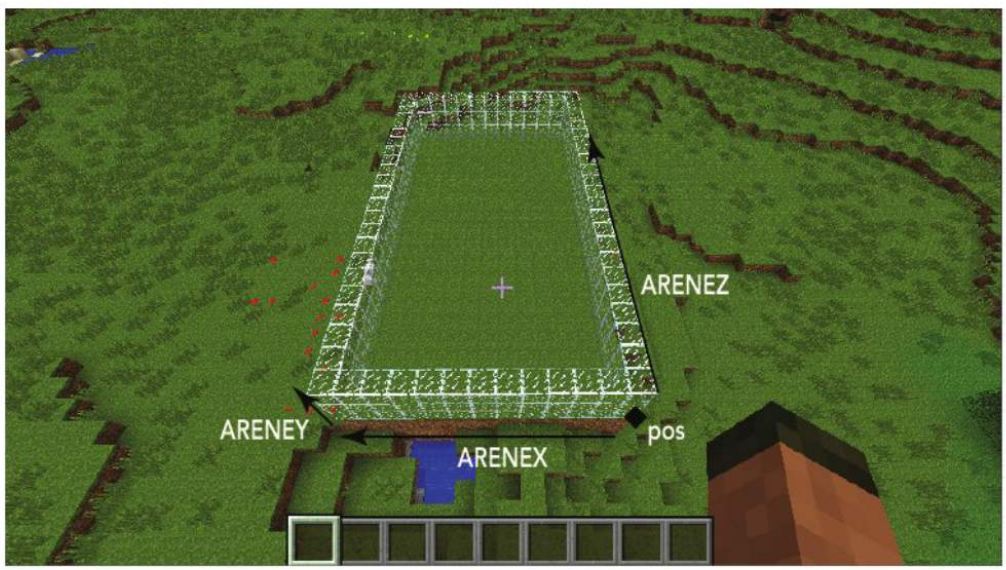


FIGURE 9-3 Création de l'arène de jeu

DÉFI



Ton arène est bien construite, mais elle est un peu triste ! Tu pourrais ajouter quelques éléments de décoration, histoire de l'égayer un peu. Pose par exemple des flèches au sol pour indiquer la direction à ton joueur ou accroche des objets au mur. Tu pourrais aussi lui construire un toit et l'orner de torches en feu !

Copyright © 2016 Eyrolles.

Deuxième partie : placer les obstacles

Ajouter de la difficulté à un jeu, c'est aussi le rendre plus intéressant. Un jeu qu'on termine vite et facilement devient rapidement ennuyeux. Dans la prochaine partie de ce chapitre, tu vas créer des obstacles pour ralentir ton joueur et l'empêcher d'atteindre facilement l'autre côté de l'arène.

Le mur

Le premier obstacle que tu vas créer est un mur en briques qui non seulement sépare l'arène en deux, mais qui en plus oscille de bas en haut. Lorsqu'il est au plus bas, il empêche ton joueur de passer. Tu dois donc attendre avant de le franchir (voir la figure 9-4).



FIGURE 9-4 Le mur que tu as érigé peut ralentir ton joueur.

Ériger un mur est une technique très pratique pour ralentir un joueur, mais s'il s'y prend bien, il peut se baisser rapidement et passer à l'épreuve suivante.

Tu vas créer ton mur en utilisant `MinecraftShape`, qui se trouve dans le module `minecraftstuff`, comme tu as appris à le faire dans le chapitre 8 lorsque tu as simulé l'invasion extraterrestre.

Mets à jour la fonction `leMur()` pour ériger l'obstacle, en suivant ces instructions.

1. Retrouve la fonction dans le script que tu as écrit en effectuant une recherche sur le code suivant :

```
def leMur(posArene, posZMur):
    pass
```

Copyright © 2016 Eyrolles.

Supprime l'instruction `pass` qui est indentée en dessous de la fonction.

2. En l'indentant sous la ligne `def leMur(posArene, posZMur):`, écris une ligne de code pour établir la connexion avec Minecraft :

```
mc = minecraft.Minecraft.create()
```

3. Il faut passer deux paramètres à la fonction `leMur()`. Le premier, `posArene`, est l'endroit où tu as construit l'arène ; le second, `posMurZ`, est la coordonnée z de l'arène où tu poseras le mur. Positionne ton mur en te servant de ces deux paramètres (voir la figure 9-5) :

```
posMur = minecraft.Vec3(posArene.x, posArene.y + 1,
                        posArene.z + posMurZ)
```

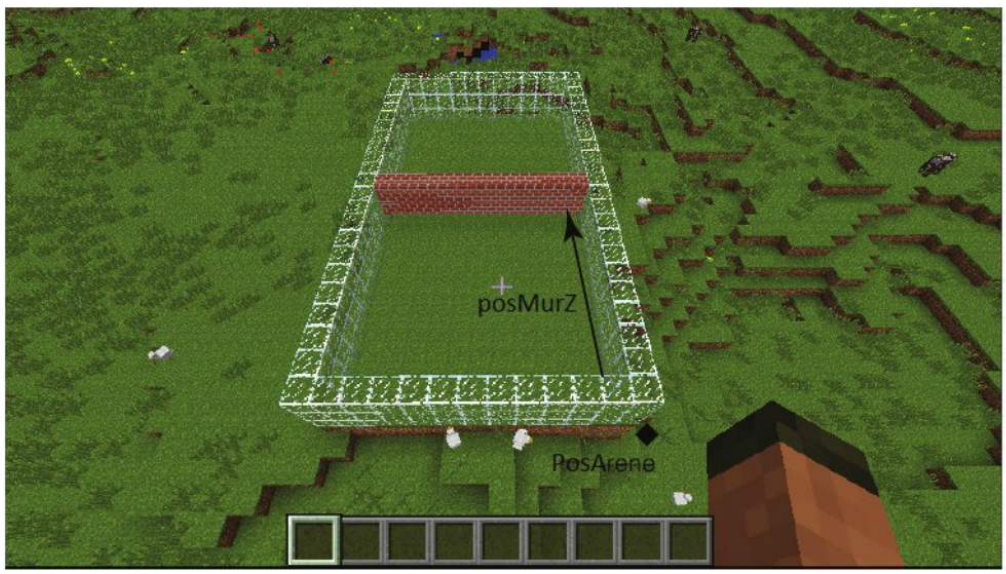


FIGURE 9-5 Positionnement du mur

4. Tu vas maintenant constituer la forme de ton mur en utilisant deux boucles `for` : une qui posera des briques à travers l'arène (x) et une autre qui posera des briques les unes sur les autres (y).

   Ensuite, tu vas ajouter ces boucles à la liste `blocsDuMur` :

```
blocsDuMur = []
for x in range(0,ARENEX + 1):
    for y in range(1, ARENEY):
        blocsDuMur.append(minecraftstuff.ShapeBlock
                          (x, y, 0, block.BRICK_BLOCK.id))
```

5. Dresse ton mur en passant les variables `posMur` et `blocsDuMur` que tu as créées aux points 3 et 4 :

Copyright © 2016 Eyrolles.

```
    formeDuMur = minecraftstuff.MinecraftShape(mc, posMur,
                                               blocsDuMur)
```

6. Ça y est, ton mur est érigé. Mais attends un peu, il est encore statique ! Pour le faire bouger de haut en bas, tu dois utiliser la fonction `moveBy()` de `MinecraftShape`, en ajoutant un petit intervalle de temps :

```
    while not gameOver:
        formeDuMur.moveBy(0,1,0)
        time.sleep(1)
        formeDuMur.moveBy(0,-1,0)
        time.sleep(1)
```

   Le code de la boucle `while` continuera de s'exécuter jusqu'à ce que la valeur `True` (ou `not gameOver`) soit affectée à la variable `gameOver`. Le mot-clé `True` sera affecté à la variable à la fin du jeu, lorsque le joueur terminera la partie ou bien la perdra.

7. La fonction `leMur()` est maintenant terminée. Tout ce qu'il te reste à faire est d'indiquer au programme principal de l'appeler et d'ajouter le code suivant à la fin du script :

```
MURZ = 10
leMur(posArene, MURZ)
```

   La constante `MURZ` retient la position `Z`, c'est-à-dire l'endroit où tu veux construire le mur dans l'arène.

8. Exécute le programme. Tu devrais voir le mur se déplacer de haut en bas au beau milieu de l'arène.

Tu n'as pas encore écrit le code qui te permet d'affecter la valeur `True` à la variable `gameOver`, alors ton programme ne s'arrêtera plus une fois que tu l'auras exécuté ! Pour l'arrêter, tu devras cliquer sur `Shell>Restart Shell` dans l'interpréteur de commandes Python.



PERCER LES SECRETS DU CODE

Lorsque tu as posé les briques du mur, tu as utilisé deux boucles `for` en les plaçant l'une à l'intérieur de l'autre. Ce procédé pratique est très courant en programmation. On parle alors d'imbrication et on dit que les boucles sont imbriquées. La première boucle `for (for x in range(0, ARENEX + 1):)` parcourt tous les blocs du mur à travers l'arène. La deuxième boucle `(for y in range(1, ARENEY):)` fait de même avec les blocs verticaux (y).

Copyright © 2016 Eyrolles.

Les variables x et y sont utilisées pour façonner chaque brique du mur selon une forme `ShapeBlock, minecraftstuff.ShapeBlock(x,y,0,block.LAY_BRICKS.id)`.

En imbriquant les boucles `for` et en utilisant les constantes `ARENEX` et `ARENEY`, tu t'assures que le mur parcourt toute la largeur de l'arène, quelle que soit sa taille.

Programmer plusieurs obstacles en même temps

Te voilà maintenant dans de beaux draps ! Le programme que tu viens de créer est bloqué. Il n'arrête pas de tourner et le mur est voué à osciller à l'infini. En effet, le programme que tu viens de rédiger est **séquentiel**, c'est-à-dire que chaque commande est exécutée l'une après l'autre et attend que la précédente se termine avant de démarrer. Si ton programme bloque, c'est parce qu'il ne passe pas à la commande qui se trouve après la boucle `while`.



Séquentiel signifie qui suit un ordre ou une séquence, c'est-à-dire qui exécute des instructions les unes à la suite des autres.

Mais alors, comment faire pour créer plusieurs obstacles et continuer le jeu si le programme reste bloqué sur le mur ?

Ne t'inquiète pas, il existe une solution. Elle s'appelle le *multithreading* ! Jusqu'à présent, tous les programmes que tu as écrits dans les chapitres précédents étaient linéaires, c'est-à-dire qu'ils étaient séquentiels et exécutaient les commandes les unes à la suite des autres.

Imagine ton programme comme un fil qui comporte plusieurs nœuds : il parcourt le fil d'un bout à l'autre, exécutant une commande à chaque nœud. Si tu insères une boucle, elle renverra ton programme vers un nœud précédent, et en insérant une commande `if`, ton programme pourra sauter une instruction. Mais il n'en reste pas moins qu'il ne pourra exécuter qu'une commande à la fois.

En revanche, si tu utilises la technique de la programmation parallèle, ou *multithreading*, tu demandes à ton programme de créer un nouveau thread (un nouveau fil), doté de ses propres commandes (ou nœuds). Ces commandes s'exécuteront simultanément à celles du programme d'origine, ce qui signifie que ton programme sera capable de faire deux choses en même temps, au lieu d'une seule (voir la figure 9-6).

Copyright © 2016 Eyrolles.

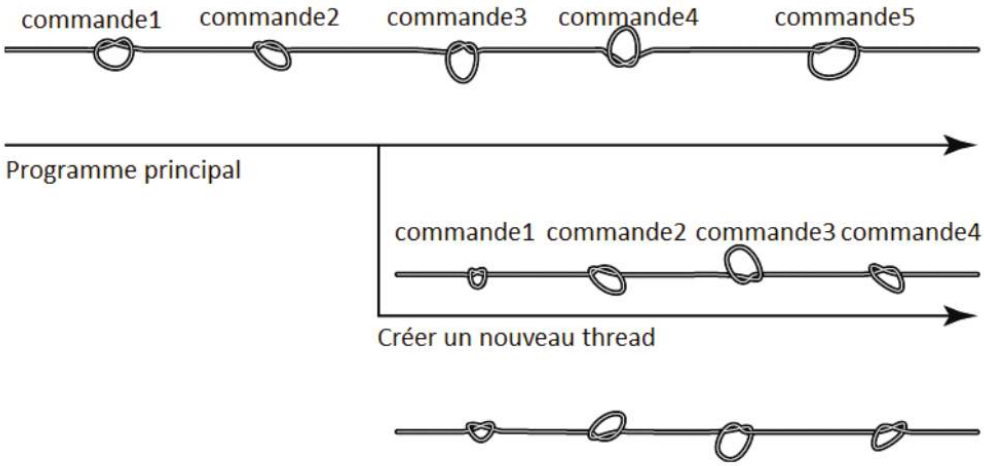


FIGURE 9-6 Quand tu crées plusieurs threads, ton programme peut exécuter plusieurs instructions à la fois.

Pour que tes obstacles fonctionnent tous en même temps, tu dois créer un nouveau thread pour chacun d'entre eux, ce qui veut dire que plusieurs programmes fonctionneront de manière simultanée.



Le multithreading est super utile en programmation, mais cette technique s'emploie à un niveau déjà très avancé et peut vite devenir difficile à gérer. Si tu veux en savoir plus sur la question, rends-toi sur le site : `www.tutorialspoint.com/python/python_multithreading.htm`.

Retournons à nos moutons. Pour activer le mur à l'aide d'un thread, tu dois modifier la ligne de code qui appelle la fonction `leMur()` et la remplacer par `thread.start_new_thread()`.

1. Supprime la dernière ligne du programme qui exécute la fonction `leMur()`, à savoir :

```
leMur(posArene, MURZ)
```

2. Insère la ligne suivante à la fin du script pour appeler la fonction `leMur()` dans un nouveau thread :

```
thread.start_new_thread(leMur, (posArene, MURZ))
```

3. Exécute le programme.

Tu devrais voir exactement le même résultat que tout à l'heure, le mur oscillant de haut en bas, à ceci près qu'il dispose de son propre thread. Tu peux désormais continuer à programmer la suite de ton jeu.

Copyright © 2016 Eyrolles.



Si tu souhaites arrêter un programme à instructions parallèles dans IDLE, clique sur `Shell>Restart Shell` dans l'interpréteur de commandes Python. Si tu appuies sur `Ctrl+F6`, cela n'arrête que le programme principal et ignore les instructions parallèles. Pour tout arrêter, tu dois donc cliquer sur `Shell>Restart Shell`.

PERCER LES SECRETS DU CODE

Pour exécuter la fonction `leMur()` dans un thread, tu utilises le code `thread.start_new_thread(leMur, (arene, MURZ))`. Le premier paramètre est le nom de la fonction (`leMur`) ; le deuxième paramètre contient les variables que la fonction requiert, à savoir `arene et MURZ`.

Si les variables, ou paramètres, à passer dans la fonction sont indiquées entre parenthèses – comme ceci : `(arene, MURZ)` –, c'est parce que l'instruction `start_new_thread` ne fonctionne qu'avec des **tuples**.

Toutes les fonctions de Python peuvent être appelées de cette façon. Si tu souhaitais appeler la fonction `print` suivante pour afficher un message à l'écran :

```
def printMessage(message)
    print message
```

tu pourrais tout aussi bien l'appeler dans un thread parallèle, comme ceci :

```
thread.start_new_thread(printMessage, ("Bonjour le monde
  de Minecraft",))
```

La virgule placée après le message `"Bonjour le monde de Minecraft"` indique à Python qu'il s'agit d'un tuple, mais d'un tuple qui ne contient qu'un élément.



Les **tuples** sont semblables aux listes que tu as utilisées dans les chapitres précédents. Mais la grande différence, c'est qu'une fois que tu en as créé un, tu ne peux plus le modifier, contrairement à la liste, à laquelle tu peux ajouter ou retirer des éléments. En programmation, on appelle les objets non modifiables des objets « immuables ». À l'inverse, ceux qui peuvent être modifiés sont dits modifiables. Pour en savoir plus sur les tuples en Python, rends-toi sur le site : `www.tutorialspoint.com/python/python_tuples.htm`.

Copyright © 2016 Eyrolles.

Creuser le lit de la rivière

La prochaine étape consiste à creuser une rivière qui va traverser l'arène dans la largeur, mais qui sera trop large pour que le joueur puisse la franchir en sautant. Heureusement, il pourra emprunter un pont pour la traverser. Seul ennui... le pont ne tient pas en place ! Il se déplace d'amont en aval, donc le joueur devra faire preuve d'une grande habileté pour le sauter afin d'atteindre l'autre rive sans tomber dans l'eau (voir la figure 9-7).

Si par inadvertance ton joueur tombe à l'eau, il sera renvoyé à l'entrée de l'arène. Tu écriras ces instructions dans la troisième partie du jeu. Pour l'heure, tu vas te contenter de créer la rivière et le pont mobile.



FIGURE 9-7 Le joueur doit franchir la rivière.

Tu dois d'abord creuser le lit de la rivière en enlevant une partie du sol de l'arène et en ajoutant des blocs d'eau au fond. Pour créer le pont, tu vas utiliser `MinecraftShape`. Et pour le faire bouger d'un côté à l'autre, tu vas utiliser la même méthode que celle que tu as employée pour créer le mur flottant.

Mets à jour la fonction `laRiviere` pour créer l'obstacle de la rivière en suivant ces étapes.

1. Retrouve la fonction `laRiviere` dans le script que tu as écrit en effectuant une recherche sur le code suivant :

```
def laRiviere(posArene, posRiviereZ):
    pass
```

   Supprime l'instruction `pass` qui est indentée en dessous de cette fonction.

Copyright © 2016 Eyrolles.

2. En l'indentant sous la ligne `def laRiviere(posArene, posRiviereZ):`, écris une ligne de code pour établir la connexion avec Minecraft :

```
mc = minecraft.Minecraft.create()
```

3. Crée deux constantes dont les valeurs correspondent à la largeur de la rivière (`LARGEURDELARIVIERE`) et à la largeur du pont (`LARGEURDUPONT`) :

```
    LARGEURDELARIVIERE = 4
    LARGEURDUPONT = 2
```

4. Maintenant, creuse le lit de la rivière à travers l'arène en te servant des paramètres `posArene` et `posRiviereZ` (la position Z correspond à l'axe de l'arène le long duquel tu vas creuser la rivière) :

```
    mc.setBlocks(posArene.x, posArene.y - 2, posArene.z +
                 posRiviereZ, posArene.x + ARENEX,
                 posArene.y, posArene.z + posRiviereZ +
                 LARGEURDELARIVIERE - 1,
                 block.AIR.id)
    mc.setBlocks(posArene.x, posArene.y - 2, posArene.z +
                 posRiviereZ, posArene.x + ARENEX,
                 posArene.y - 2, posArene.z + posRiviereZ +
                 LARGEURDELARIVIERE - 1,
                 block.WATER.id)
```

Pour créer la rivière, tu dois utiliser `setBlocks()` et creuser un trou d'`AIR` dans le sol, puis ajouter une couche d'`EAU` au fond.

5. Repère l'endroit où tu vas placer ton pont. Pour obliger ton joueur à effectuer un saut, tu dois le placer au beau milieu de la rivière, loin de la rive :

```
    posPont = minecraft.Vec3(posArene.x, posArene.y,
                             posArene.z + posRiviereZ + 1)
```

6. Crée les blocs de la figure du pont en t'inspirant de la méthode que tu as employée pour ériger le mur, c'est-à-dire en utilisant deux boucles `for` imbriquées, une qui correspond à la largeur du pont (x) et l'autre à la largeur de la rivière (z) :

```
    blocsDuPont = []
    for x in range(0, LARGEURDUPONT):
        for z in range(0, LARGEURDELARIVIERE - 2):
            blocsDuPont.append(minecraftstuff.ShapeBlock(x,
                                                         0,
                                                         z,
                                                block.WOOD_PLANKS.id))
```

Lors de la création du pont, tu soustrais 2 à la `LARGEURDELARIVIERE` pour laisser un bloc d'espace entre la rive et le pont. Cela signifie que le joueur devra sauter sur le pont puis sur l'autre rive pour franchir l'obstacle (voir la figure 9-8).

Copyright © 2016 Eyrolles.



FIGURE 9-8 Comme le pont est plus petit que la rivière, ton joueur est obligé de sauter.

7. Façonne ton pont en passant en paramètre les variables `posPont` et `blocsDuPont` que tu as définies aux étapes 5 et 6 :

```
formeDuPont = minecraftstuff.MinecraftShape(mc, posPont,
                                             blocsDuPont)
```

8. Pour que le pont avance d'un bloc sur l'eau, tu dois calculer le nombre de mètres que le pont doit effectuer pour aller d'un bout à l'autre de l'arène, soit la largeur de l'arène moins la largeur du pont :

```
metres = ARENEX - LARGEURDUPONT
```

9. Fais dériver le pont d'un bout à l'autre de l'arène en utilisant deux boucles `for` (l'une allant à droite et l'autre à gauche) et la fonction `MinecraftShape.moveBy()` pour le faire bouger d'un bloc à chaque mètre. N'oublie pas d'ajouter un petit intervalle de temps entre chaque mètre :

```
while not gameOver:
    for left in range(0, metres):
        formeDuPont.moveBy(1,0,0)
        time.sleep(1)
    for right in range(0, metres):
        formeDuPont.moveBy(-1,0,0)
        time.sleep(1)
```

10. La fonction de la rivière est terminée. Tu dois maintenant l'appeler à partir d'un nouveau thread lié au programme principal. Pour cela, ajoute le code suivant à la fin du script :

```
RIVIEREZ = 4
thread.start_new_thread(laRiviere, (posArene, RIVIEREZ))
```

Copyright © 2016 Eyrolles.

11. Il est temps d'exécuter le programme ! Ton arène est coupée en deux par le mur mobile et la rivière qui la traverse est surmontée d'un pont qui fait un va-et-vient.



Entre dans l'arène et essaye de traverser le pont. Si tu maîtrises bien les commandes, tu devrais y parvenir assez facilement, mais lorsque tu devras te dépêcher de récupérer des diamants dans l'arène en un temps record, franchir cet obstacle sera une autre paire de manches ! Il se peut que tu manques le coche plus d'une fois.

Creuser les trous

Les derniers obstacles que tu dois construire dans ton jeu Traversée périlleuse sont des trous, qui se forment de manière aléatoire et se refermeront toutes les secondes pour réapparaître ailleurs (voir la figure 9-9).

Pour obtenir des positions aléatoires, tu appelleras la fonction `randint()`. Des blocs de `LAINE NOIRE` apparaîtront furtivement avant que le sol ne se creuse pour prévenir ton joueur et lui donner le temps de s'échapper. Pour créer les trous, tu changeras les blocs du sol en blocs d'`AIR`. Ils resteront dans cet état quelques secondes avant de redevenir des blocs d'`HERBE` et de réapparaître ailleurs.



FIGURE 9-9 Les trous se forment, ton joueur doit faire attention à ne pas tomber dedans !

Comme pour la rivière, si ton joueur tombe dans un trou, il sera renvoyé à l'entrée de l'arène. Tu écriras cette partie du code seulement dans la troisième partie de ce chapitre.

Copyright © 2016 Eyrolles.

Tu vas maintenant actualiser la fonction `lesTrous` pour creuser les obstacles.

1. Retrouve la fonction `lesTrous` dans le script que tu as écrit en effectuant une recherche sur le code suivant :

```
def lesTrous(posArene, posZTrous):
    pass
```

   Supprime l'instruction `pass` qui est indentée en dessous de la fonction.

2. En l'indentant sous la ligne `def lesTrous(posArene, posZTrous):`, écris une ligne de code pour établir la connexion avec Minecraft :

```
mc = minecraft.Minecraft.create()
```

3. Crée deux constantes dont les valeurs correspondent au nombre de trous à creuser (`TROUS`) et à leur largeur (voir la figure 9-10) :

```
    TROUS = 15
    LARGEURDESTROUS = 3
```

4. Crée ensuite la boucle `while` des trous, qui continuera de tourner jusqu'à la fin du jeu :

```
    while not gameOver:
```

   Le reste du code de la fonction `lesTrous` devra être indenté sous cette boucle `while`.

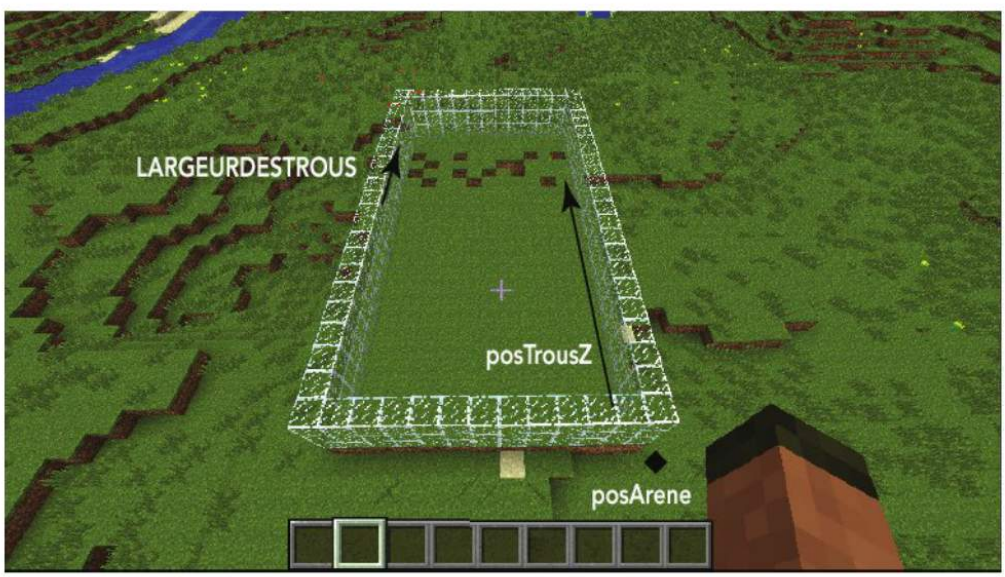


FIGURE 9-10 Définis la position et la largeur des trous.

Copyright © 2016 Eyrolles.

5. Creuse les trous à des endroits différents en utilisant la fonction `random.randint()`, qui définira leurs coordonnées x et z au hasard (la coordonnée y correspond au positionnement de l'arène) et ajoute-les à une liste Python :

```
trous = []
for count in range(0,TROUS):
    x = random.randint(posArene.x,
                       posArene.x + ARENEX)
    z = random.randint(posArene.z + posTrousZ,
                       posArene.z + posTrousZ
                       + LARGEURDESTROUS)
    trous.append(minecraft.Vec3(x, posArene.y, z))
```

6. Parcoure toutes les coordonnées de la liste de trous et transforme les blocs en blocs de `LAINE NOIRE` :

```
for trou in trous:
    mc.setBlock(trou.x, trou.y, trou.z,
                block.WOOL.id, 15)
    time.sleep(0.25)
```

En colorant le sol en noir, tu préviens ton joueur qu'un nouveau trou va se former et lui donnes ainsi une chance de l'éviter.

7. Pour ouvrir les trous, transforme les blocs situés sous leurs emplacements en blocs d'`AIR` à l'aide de la fonction `setBlocks()` :

```
for trou in trous:
    mc.setBlocks(trou.x, trou.y, trou.z,
                 trou.x, trou.y - 2, trou.z,
                 block.AIR.id)
time.sleep(2)
```

Une fois que tu as formé les trous, ajoute un délai de deux secondes pour qu'ils restent dans cet état un petit instant.

8. Pour les refermer, utilise la même boucle que celle qui t'a servi à les ouvrir, mais en les transformant cette fois-ci en blocs d'`HERBE` :

```
for trou in trous:
    mc.setBlocks(trou.x, trou.y, trou.z,
                 trou.x, trou.y - 2, trou.z,
                 block.GRASS.id)
time.sleep(2)
```

Le programme va alors revenir au début de la boucle `while` et créer une nouvelle série de trous.

9. Ta fonction `trous` est maintenant terminée. Ajoute le code suivant à la fin du programme pour l'appeler :

Copyright © 2016 Eyrolles.

```
TROUSZ = 15
thread.start_new_thread(lesTrous, (posArene, TROUSZ))
```

La constante `TROUSZ` correspond à l'axe z de l'arène où l'obstacle des trous sera créé.

10. Exécute le programme. Comme avant, tu peux voir l'arène avec les obstacles du mur et de la rivière, mais tu devrais aussi voir des trous se creuser et se refermer à des endroits différents et de façon aléatoire de l'autre côté de l'arène.

Lance-toi dans l'arène et essaye de voir si tu peux la traverser plusieurs fois en évitant les obstacles et sans tomber dans les pièges.

Tu peux en outre modifier les constantes du jeu pour te familiariser avec. Essaye par exemple d'agrandir l'arène ou de l'élargir, déplace les obstacles ou rends-les plus difficiles à franchir en accélérant leurs mouvements.



PERCER LES SECRETS DU CODE

Les fonctions qui créent les obstacles sont comme des mini-programmes qui fonctionnent indépendamment les uns des autres grâce au multithreading. Avec cette technique, il est très facile de créer une nouvelle instruction parallèle pour ajouter un obstacle. Peut-être préfères-tu avoir deux murs plutôt qu'un ? Appelle la fonction `leMur` à nouveau en attribuant au premier mur une autre coordonnée z et tu verras un second mur apparaître et osciller de haut en bas, à l'instar du premier :

```
MUR2Z = 13
thread.start_new_thread(leMur, (posArene, MUR2Z))
```

DÉFI

Tu n'es pas obligé de t'en tenir là. En utilisant les mêmes méthodes que pour le mur, la rivière et les trous, tu peux créer des obstacles en tous genres ! Invente par exemple une cage qui apparaît de façon aléatoire et enferme ton joueur par surprise, ou encore plusieurs plates-formes sur lesquelles ton joueur doit sauter pour rejoindre l'autre côté de l'arène.



Copyright © 2016 Eyrolles.



Et si les obstacles que tu as créés sont un peu trop simples à franchir ou trop difficiles à surmonter, tu peux tout à fait en modifier la difficulté en affectant des valeurs différentes aux constantes. Tu peux notamment augmenter ou diminuer les intervalles de temps pour ralentir ou accélérer le mouvement des obstacles. Pour accélérer les mouvements du pont, modifie le code `time.sleep(1)` dans les deux boucles `for` de gauche et de droite et abaisse le temps à `time.sleep(0.5)`.

Troisième partie : la jouabilité

Te voilà maintenant prêt à entamer la troisième partie de ta grande aventure. Ton but ici est d'introduire un peu de jouabilité, c'est-à-dire que tu vas transformer ton arène, qui n'est pour l'instant qu'une simple course d'obstacles, en véritable jeu à niveaux dont tu ne pourras plus te passer !

Pour y arriver, il faut que ton jeu soit suffisamment captivant et exigeant, que tu donnes un but à ton joueur et que tu lui promettes une récompense.

Le but du jeu consistera à récupérer tous les diamants dispersés au hasard dans l'arène, tout en essayant de franchir les obstacles en un temps donné. À chaque niveau terminé et pour chaque diamant récupéré, ton joueur cumulera des points. Plus il passera les niveaux rapidement et plus il gagnera de points. Le but du jeu consiste donc à passer tous les niveaux et à obtenir le plus de points possible.

Ton jeu comportera trois niveaux, dont la difficulté augmentera à mesure que ton joueur les complétera, car tu ajouteras à chaque fois plus de diamants et réduiras le compte à rebours.

Démarrer le jeu

Dans cette partie, tu vas poser les jalons de ton jeu et créer des constantes qui détermineront le nombre de diamants ainsi que les limites de temps à attribuer à chaque niveau.

Ton programme se compose de deux boucles principales.

- La boucle de jeu : elle effectuera des tours jusqu'à la fin du jeu (`while not gameOver`). À cette étape, tu vas paramétrer chacun des niveaux et les initialiser.
- La boucle de `niveau` : elle effectuera des tours jusqu'à la fin de chaque niveau, c'est-à-dire jusqu'à ce que tu termines un niveau ou le jeu tout entier (`while not gameOver and not niveauTermine`). Elle permettra également de renvoyer le joueur à l'entrée de l'arène s'il tombe dans la rivière ou dans un trou, de supprimer les diamants dès qu'il les frappe et de vérifier le temps qu'il lui reste.

Copyright © 2016 Eyrolles.

Dans cette section du chapitre, il est important que tu fasses attention à l'indentation du code et le places au bon endroit, sous la boucle de `jeu` ou sous la boucle de `niveau`. Sinon ton programme ne fonctionnera pas correctement.



Tu vas maintenant créer la structure du jeu ainsi que les deux boucles principales, en respectant les étapes suivantes pour pouvoir ajouter le code nécessaire à la fin du script.

1. Saisis les trois constantes suivantes à la suite de ton programme. Elles correspondent aux niveaux du jeu, au nombre de diamants qui seront parsemés dans l'arène et au temps (en secondes) qui sera accordé au joueur pour terminer chaque niveau :

```
NIVEAUX = 3
DIAMANTS = [3,5,9]
DELAIS = [30,25,20]
```

   Les constantes `DIAMANTS` et `DELAIS` sont des listes Python qui contiennent toutes les deux trois éléments ; ces éléments sont des valeurs réparties sur les trois niveaux. Par exemple, dans le premier niveau, le joueur doit récupérer trois diamants ; il dispose de 30 secondes pour le faire et pour parvenir de l'autre côté de l'arène.

2. Crée deux autres variables pour enregistrer les points cumulés par le joueur et le niveau auquel il se trouve :

```
niveau = 0
points = 0
```

3. Crée la boucle `jeu`. Rédige un commentaire à cet endroit pour te souvenir de son emplacement, car tu devras indenter du code sous cette boucle un peu plus tard dans ce chapitre.

```
# boucle du jeu
while not gameOver:
```

   Au début du programme, la valeur `False` est affectée à la variable `gameOver`. On lui affectera la valeur `True` lorsque le temps sera écoulé ou lorsque le joueur aura terminé le jeu. Cette variable est également utilisée dans les fonctions relatives aux obstacles. Lorsqu'on lui affecte la valeur `True`, elle désactive tous les obstacles.

4. Sous la boucle `jeu`, modifie la position de ton joueur pour le placer à l'entrée de l'arène, prêt à commencer :

```
    mc.player.setPos(posArene.x + 1, posArene.y + 1,
                     posArene.z + 1)
```

Copyright © 2016 Eyrolles.

5. Active le compte à rebours du niveau en récupérant l'heure actuelle et en l'affectant à une variable :

```
    debut = time.time()
```

6. Affecte la valeur `False` à la variable de fin du niveau et crée la boucle de niveau. Comme pour la boucle `jeu`, rédige un commentaire pour retrouver facilement cette boucle, car tu indenteras du code dedans un peu plus tard :

```
    niveauTermine = False
    # boucle de niveau
    while not gameOver and not niveauTermine:
```

7. Sous la boucle `niveau`, ajoute un petit délai. Celui-ci est nécessaire, car le programme s'exécute pendant toute la durée du jeu et, sans lui, le programme utiliserait toute la puissance de ton ordinateur :

```
        time.sleep(0.1)
```

8. Exécute le programme. Tu ne verras pas de grand changement s'opérer pour l'instant, car ces nouvelles lignes n'ont pour effet que de renvoyer ton joueur à l'entrée de l'arène. En revanche, cela te permettra de vérifier que tout fonctionne correctement et que ton script ne comporte aucune erreur.

DÉFI



Lorsque ton joueur est renvoyé à l'entrée de l'arène, il recommence à partir de l'angle de droite. Mais ne serait-il pas plus pratique de le renvoyer au milieu ? Modifie le code pour recentrer l'entrée de l'arène, de façon à ce que ton joueur recommence chaque niveau à partir du milieu.

Récupérer des diamants

L'objectif principal du jeu est de récupérer des diamants. Tu vas maintenant les programmer pour qu'ils apparaissent à des endroits différents dans l'arène (voir la figure 9-11). Pour les prendre, ton joueur devra les « frapper », ou plutôt effectuer un clic droit sur le bloc en tenant son épée.

Les diamants disparaîtront dès qu'ils auront été frappés. Et une fois qu'il aura collecté tous les diamants, ton joueur pourra terminer le niveau en se rendant de l'autre côté de l'arène.

Copyright © 2016 Eyrolles.

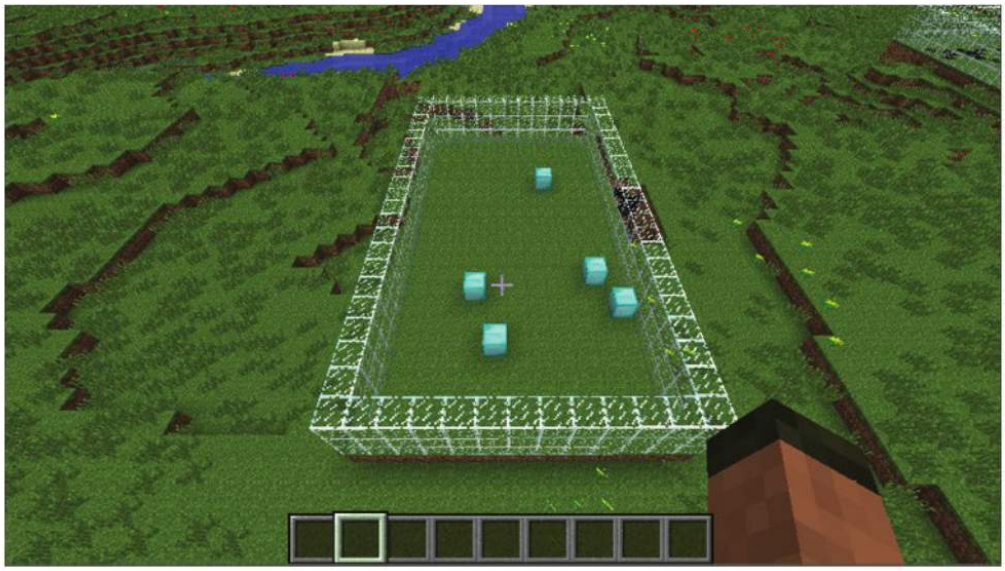


FIGURE 9-11 Les diamants apparaissent à des endroits différents.

Mets à jour ta fonction `poserDiamants()` pour l'appeler sous la boucle `jeu` en suivant ces quelques étapes.

1. Pour retrouver la fonction `poserDiamants()` dans le script, effectue une recherche sur le code suivant :

```
def poserDiamants(posArene, nombre):
    pass
```

   Supprime l'instruction `pass` qui est indentée en dessous de la fonction.

2. En l'indentant sous la ligne `def poserDiamants(posArene, nombre):`, écris une ligne de code pour établir la connexion avec Minecraft :

```
mc = minecraft.Minecraft.create()
```

3. Pour créer les diamants, tu dois prendre un bloc au hasard dans l'arène, récupérer ses coordonnées x et z, puis le transformer en bloc de diamant :

```
    for diamant in range(0, nombre):
        x = random.randint(posArene.x, posArene.x + ARENEX)
        z = random.randint(posArene.z, posArene.z + ARENEZ)
        mc.setBlock(x, posArene.y + 1, z,
                    block.DIAMOND_BLOCK.id)
```

4. Il faut maintenant que tu appelles la fonction `poserDiamants()` au début de la boucle `jeu`. Ainsi, à chaque fois que ton joueur passera à un niveau supérieur, une nouvelle poignée de diamants sera jetée dans l'arène. Dans le corps de la boucle jeu, ajoute le code suivant juste en dessous de la boucle `while` :

Copyright © 2016 Eyrolles.

```
# boucle du jeu
while not gameOver:
    poserDiamants(posArene, DIAMANTS[niveau])
    diamantsRestants = DIAMANTS[niveau]
```

Dans ces instructions, tu viens également de créer la variable `diamantsRestants`. Elle a pour fonction de retenir le nombre de diamants que le joueur doit encore récupérer.

5. Exécute le programme. Comme tu n'en es qu'au premier niveau, tu devrais voir trois diamants apparaître à différents endroits dans l'arène.

Une fois les diamants créés, tu peux passer à l'écriture du code qui va te permettre de gérer les frappes de ton joueur. Pour cela, tu utiliseras la fonction `pollBlockHits`, qui, comme tu as pu le voir dans le chapitre 4, transforme les blocs de `DIAMANT` en blocs d'`AIR` lorsque ton joueur les frappe.

Ajoute le code suivant sous la boucle `niveau` pour transformer le bloc de diamant en bloc d'`AIR` si ton joueur le frappe.

1. En l'indentant sous la boucle `niveau`, appelle la fonction `pollBlockHits` pour récupérer tous les événements de frappe :

```
    # boucle de niveau
    while not gameOver and not niveauTermine:
        frappes = mc.events.pollBlockHits()
```

2. Passe en revue tous les événements de frappe à l'aide d'une boucle et récupère l'identité du bloc qui a été frappé :

```
        for frappe in frappes:
            typeDeBlocFrappe = mc.getBlock(frappe.pos.x,
                                           frappe.pos.y,
                                           frappe.pos.z)
```

3. Vérifie si le type de bloc que ton joueur a frappé est un bloc de diamant. Si c'est le cas, transforme-le en `AIR` et soustrais le chiffre 1 à la variable `diamantsRestants` :

```
            if typeDeBlocFrappe == block.DIAMOND_BLOCK.id:
                mc.setBlock(frappe.pos.x, frappe.pos.y,
                            frappe.pos.z, block.AIR.id)
                diamantsRestants = diamantsRestants - 1
```

La variable `diamantsRestants` sera utilisée pour vérifier si tous les diamants ont été récupérés une fois que ton joueur atteindra l'autre bout de l'arène.

4. Exécute le programme. Si tu frappes les blocs de diamant, ils devraient normalement se transformer en `AIR` et disparaître.

Copyright © 2016 Eyrolles.

Souviens-toi que pour frapper un bloc, tu dois brandir une épée et faire un clic droit avec ta souris.



DÉFI

Et si tes diamants étaient plus difficiles à atteindre ? Tu pourrais par exemple t'amuser à les rendre mobiles en les faisant osciller de haut en bas et ne permettre à ton joueur de les frapper que lorsqu'ils sont en l'air.



Plus une seconde à perdre !

Si tu n'imposais aucune limite de temps à ton joueur, le jeu serait bien trop simple, voire complètement ennuyeux. En introduisant une limite de temps, tu rends ton jeu bien plus stimulant. Tu attribues de ce fait à ton joueur un objectif en lui demandant de récupérer tous les diamants et d'atteindre l'autre côté de l'arène avant la fin du compte à rebours. Et s'il ne reste plus de temps, le jeu se termine.

Le prochain exercice consiste à développer ta boucle de niveau pour vérifier le temps qu'il reste et, s'il est épuisé, affecter la valeur `True` à la variable `gameOver`.

1. À la suite de ta boucle `niveau`, indique à ton programme de calculer le nombre de secondes restantes dans le niveau en question :

```
# boucle de niveau
while not gameOver and not niveauTermine:

    secondesRestantes = DELAIS[niveau] - (time.time()
                                          - debut)
```

2. Si le résultat tombe en dessous de 0 seconde, affecte la valeur `True` à la variable `gameOver` et publie un message dans le chat :

```
    if secondesRestantes < 0:
        gameOver = True
        mc.postToChat("Trop tard...")
```

3. Exécute le programme. Au bout de 30 secondes (comme il s'agit du premier niveau), ton programme devrait s'arrêter et le message « Trop tard... » devrait s'afficher dans le chat (voir la figure 9-12).

Copyright © 2016 Eyrolles.



FIGURE 9-12 Indique à ton joueur qu'il ne lui reste plus de temps.

PERCER LES SECRETS DU CODE

Le calcul du nombre de secondes restantes par niveau s'effectue dans cette ligne de code :

```
secondesRestantes = DELAIS[niveau] - (time.time() - debut)
```

Pour construire ta variable `secondesRestantes`, tu dois récupérer le nombre de secondes allouées au niveau dans la constante `DELAIS` :

```
DELAIS[niveau]
```

Puis tu soustraits cette valeur au nombre de secondes qui se sont écoulées depuis le début du niveau, c'est-à-dire l'heure actuelle (`time.time()`) moins l'heure à laquelle le niveau a débuté (`debut`) :

```
(time.time() - debut)
```

Localiser le joueur

Ton joueur termine un niveau lorsqu'il a récupéré tous les diamants et est arrivé de l'autre côté de l'arène. S'il tombe dans la rivière ou dans un trou, il revient au début de l'arène. Pour que ces événements puissent se produire, ton programme doit pouvoir localiser ton joueur.

Une fois que ton programme a repéré ton joueur, il peut soit affecter la valeur `True` à la variable `niveauTermine`, s'il a récupéré tous les diamants, soit changer sa position pour le renvoyer à l'entrée de l'arène.

Ta prochaine tâche va consister à développer la boucle `niveau` pour localiser ton joueur et soit le renvoyer à l'entrée de l'arène, soit terminer le niveau.

Copyright © 2016 Eyrolles.

1. Sous la boucle `niveau`, récupère les données de position de ton joueur :

```
    # boucle de niveau
    while not gameOver and not niveauTermine:

        pos = mc.player.getTilePos()
```

2. Vérifie la hauteur (coordonnée y) à laquelle se trouve ton joueur pour savoir si elle est inférieure à la hauteur du sol. Si c'est le cas, c'est qu'il a dû tomber dans la rivière ou dans un des trous. Tu dois donc le renvoyer au début :

```
        if pos.y < posArene.y:
            mc.player.setPos(posArene.x + 1, posArene.y + 1,
                             posArene.z + 1)
```

3. Vérifie que ton joueur est arrivé au bout de l'arène et qu'il a récupéré tous les diamants. Pour cela, tu dois comparer la coordonnée z de sa position à celle de l'arrivée. Si elles sont identiques, alors tu peux affecter la valeur `True` à la variable `niveauTermine` :

```
        if pos.z == posArene.z + ARENEZ and ↩
                    diamantsRestants == 0:
            niveauTermine = True
```

   Dès que la variable `niveauTermine` indique `True`, cela marque la fin de la boucle `niveau`. Les diamants sont alors recréés et le jeu recommence.

4. Exécute le programme. Le jeu devrait recommencer une fois tous les diamants collectés et l'autre rive atteinte. En revanche, si ton joueur tombe dans la rivière ou dans un trou, il devrait être renvoyé au début de l'arène.

Pour l'instant, ton jeu ne comporte qu'un seul niveau, à savoir le premier, qui est aussi le plus simple des trois. Je te conseille vivement de commencer à t'entraîner, car le prochain niveau va te donner du fil à retordre !



Indiquer la fin d'un niveau de jeu et calculer des points

Dès lors que ton joueur termine un niveau, le programme doit calculer le nombre de points qu'il a cumulés et passer au niveau suivant. Ton joueur ne marquera les points en question que s'il termine le niveau. Pour chaque diamant ramassé, ton joueur obtient un point, multiplié par le nombre de secondes restantes.

Pour cela, tu dois écrire le code suivant. Ce nouveau code doit être indenté en dessous de la boucle `jeu`, à la fin de la boucle `niveau`.

Copyright © 2016 Eyrolles.

1. En te plaçant après la boucle `niveau`, sur le même alinéa que la boucle `jeu`, vérifie que ton joueur a bien terminé le niveau :

```
# boucle du jeu
while not gameOver:
     [code]
    # boucle de niveau
    while not gameOver and not niveauTermine:
     [code]

    if niveauTermine:
```

2. S'il l'a terminé, calcule le nombre de points qu'il a cumulés et affecte le résultat à la variable `points`, avant de l'annoncer sur le chat :

```
        points = points + (DIAMANTS[niveau] * ↩
                           int(secondesRestantes))
        mc.postToChat("Niveau termine - Points = " ↩
                      + str(points))
```

3. Passe au niveau suivant en ajoutant 1 à la variable de `niveau` :

```
        niveau = niveau + 1
```

4. S'il s'agit du dernier niveau, affecte la valeur `True` à la variable `gameOver` et publie un message de félicitations dans le chat :

```
        if niveau == niveaux:
            gameOver = True
            mc.postToChat("Bravo ! Tu as terminé tous les ↩
                          niveaux.")
```

   La constante `NIVEAUX` enregistre le nombre total de niveaux que le jeu possède.

5. À présent, exécute le programme.

Ton jeu est presque terminé ! Lorsque ton joueur aura récupéré tous les diamants et qu'il se sera frayé un chemin jusqu'à l'autre bout de l'arène avant la fin du chronomètre, le jeu passera au niveau suivant. Il deviendra d'autant plus difficile à gagner, car ton joueur devra ramasser plus de diamants et disposera de moins de temps pour y arriver. Mais si malgré tout il vient à bout de tous les niveaux, il aura gagné le jeu et tu pourras le féliciter !

DÉFI



Essaye d'ajouter davantage de niveaux. Tu peux par exemple ralentir la cadence des premiers niveaux en donnant plus de temps à ton joueur pour moins de diamants, puis augmenter la difficulté petit à petit pour finir avec des niveaux bien plus ardus.

Copyright © 2016 Eyrolles.

Afficher le message « Game over »

La toute dernière chose qu'il te reste à faire pour que ton jeu soit complet, c'est d'ajouter un message à la suite de ton script pour indiquer au joueur que le jeu est fini, en lui donnant son score final (voir la figure 9-13). Pour cela, il te suffit d'ajouter ces quelques lignes de code à la fin de ton programme :

```
mc.postToChat("Jeu terminé - Points = " + str(points))
```



FIGURE 9-13 Indique au joueur que le jeu est fini et affiche son score.

J'ai gagné 156 points. Arriveras-tu à battre mon record ?



DÉFI

Enregistre les résultats dans un fichier sur ton ordinateur et crée un tableau de classement pour afficher les progrès de ton joueur à la fin du jeu.



Tu peux télécharger le programme entier du jeu « Traversée périlleuse » sur le site Internet du livre à l'adresse : www.editions-eyrolles.com/dl/14292.

Copyright © 2016 Eyrolles.

Tu n'es pas obligé de donner une fin à ton jeu ! Ce programme est conçu pour être développé, alors amuse-toi à en modifier les paramètres, à y introduire de nouveaux obstacles et à créer une arène encore plus impressionnante. Tu peux donner libre cours à ton imagination.

Quatrième partie : ajouter un bouton-poussoir et un afficheur

Le programme « Traversée périlleuse » comporte quelques problèmes. Pour commencer, le jeu démarre automatiquement, sans attendre que ton joueur soit prêt. Et il n'affiche pas d'information utile, comme le nombre de diamants à récupérer et le temps qu'il te reste.

Dans la dernière partie de ce chapitre, tu vas réutiliser le détonateur que tu as fabriqué dans le chapitre 5 et le transformer en bouton-poussoir. Tu pourras l'actionner dès que tu voudras commencer une partie. Tu utiliseras aussi l'afficheur 7 segments pour afficher le nombre de diamants restants. Le séparateur décimal de l'afficheur clignotera lorsque tu seras arrivé à cinq secondes de la fin d'un niveau.

Ce dont tu as besoin

Tu auras besoin exactement des mêmes composants que dans le chapitre 5, avec les mêmes branchements que le détonateur que tu as fabriqué. Tu dois donc te doter d'une plaque de montage, d'un afficheur 7 segments et d'un bouton-poussoir. Tu dois ensuite les connecter à ton Raspberry Pi (figure 9-14) ou à un circuit Arduino (figure 9-15). Au besoin, pour la configuration du dispositif, je t'invite à relire le chapitre 5.

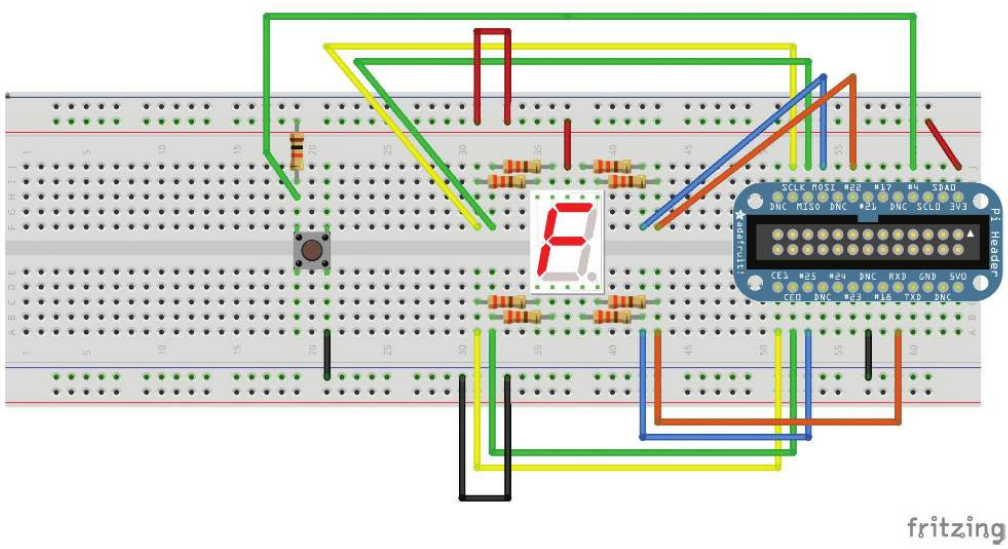


FIGURE 9-14 Schéma du circuit pour Raspberry Pi

Copyright © 2016 Eyrolles.

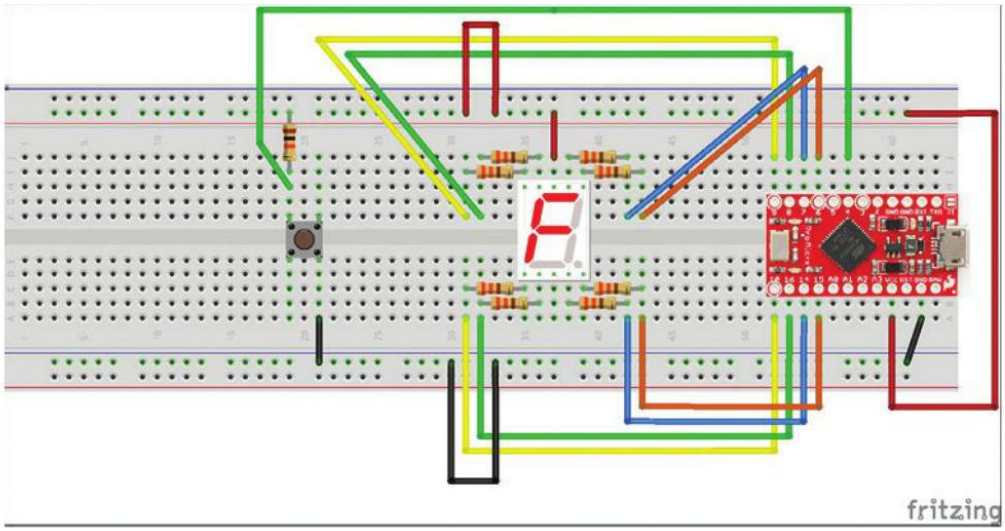


FIGURE 9-15 Schéma du circuit pour Arduino

Préparer ton matériel

Pour ajouter un bouton et un afficheur 7 segments, tu dois tout d'abord t'occuper d'importer les bons modules, de configurer le GPIO et de créer des constantes pour chacune des broches.

Modifie le programme de ton jeu pour configurer le dispositif et commence à jouer une fois le bouton enclenché.

1. Importe le module de l'afficheur 7 segments sous les instructions d'import existantes :

```
import anyio.seg7 as display
```

2. Importe le module GPIO pour ton dispositif et crée des constantes pour les broches que tu utiliseras :

Sur Raspberry Pi

```
import RPi.GPIO as GPIO
BOUTON = 4
STONES_LED = [10,22,25,8,7,9,11,15]
```

Sur Arduino

```
import anyio.GPIO as GPIO
BOUTON = 4
STONES_LED = [7,6,14,16,10,8,9,15]
```

Copyright © 2016 Eyrolles.

3. Prépare ton matériel.

```
GPIO.setmode(GPIO.BCM)
GPIO.setup(BOUTON, GPIO.IN)
ON = False # anode commune, indiquer True pour un afficheur
           # à cathode commune
display.setup(GPIO, STONES_LED, ON)
```

Si tu utilises un afficheur 7 segments à cathode commune (comme expliqué dans le chapitre 5), écris `ON = True`.

4. Juste avant la boucle `jeu`, ajoute le code d'enclenchement du bouton :

```
mc.postToChat("Appuie sur le bouton pour commencer")
while GPIO.input(BOUTON):
    time.sleep(0.1)
    while not gameOver:
```

Le programme va alors créer l'arène et les obstacles, puis attendre que ton joueur appuie sur le bouton. Dès qu'il appuiera dessus, le programme lancera la boucle du jeu, introduira ton joueur dans l'arène et démarrera la partie.

5. Il est important de terminer ton programme correctement et de nettoyer le port GPIO. Pour cela, ajoute le code suivant à la fin du script :

```
GPIO.cleanup()
```

6. Exécute le programme et teste ton nouveau bouton de démarrage.

DÉFI



Seras-tu capable de modifier ton programme de façon à te servir du bouton pour recommencer une partie au lieu d'attendre que le programme ne s'exécute à nouveau ?

Faire le décompte des diamants

L'afficheur 7 segments va afficher le nombre de diamants restants que ton joueur doit récupérer. Il se mettra à jour grâce à la fonction `display.write`.

Pour ce faire, tu dois modifier ton programme « Traversée périlleuse » afin d'actualiser l'afficheur dès que des diamants apparaissent dans l'arène et dès que ton joueur en frappe un :

1. Après avoir appelé la fonction `poserDiamants` dans la boucle `jeu` et une fois que les diamants ont été semés dans l'arène, mets l'afficheur à jour :

Copyright © 2016 Eyrolles.

```
    poserDiamants(arenaPos, DIAMANTS[niveau])
    diamantsRestants = DIAMANTS[niveau]
    display.write(str(diamantsRestants))
```

La méthode `display.write()` attend qu'on lui passe une chaîne de caractères en paramètre. On convertit donc le nombre de diamants restants en chaîne de caractères à l'aide de `str()` avant de le passer dans la fonction.

2. Dès que ton joueur frappe un diamant et que le nombre de diamants dans l'arène diminue de 1, tu mets à jour l'afficheur :

```
        frappes = mc.events.pollBlockHits()
        for frappe in frappes:
            typeDeBlocFrappe = mc.getBlock(frappe.pos.x,
                                           frappe.pos.y,
                                           frappe.pos.z)
        if typeDeBlocFrappe == block.DIAMOND_BLOCK.id:
            mc.setBlock(frappe.pos.x, frappe.pos.y, frappe.
                        pos.z, block.AIR.id)
            diamantsRestants = diamantsRestants - 1
            display.write(str(diamantsRestants))
```

3. À la fin du script, tu dois t'assurer que l'affichage disparaisse. Avant de nettoyer le GPIO, saisis le code suivant pour l'effacer :

```
display.clear()
GPIO.cleanup()
```

4. Maintenant, exécute le programme. Le nombre de diamants restants devrait s'afficher et diminuer chaque fois que ton joueur en frappe un, jusqu'à ce qu'il tombe à 0.

L'afficheur 7 segments ne s'élève que jusqu'à 9. Si tu as choisi de modifier ton programme pour semer plus de 9 diamants dans l'arène, tu devras ajouter une instruction `if` au script pour que ton afficheur ne se mette à jour que lorsque le nombre de diamants à récupérer sera passé en dessous de 10.



Indiquer le temps restant

La toute dernière étape de ce chapitre consiste à mettre l'afficheur à jour pour indiquer le temps au joueur. Pour cela, tu vas activer le séparateur décimal de l'afficheur. La LED du séparateur décimal s'affichera pour indiquer à ton joueur qu'il ne lui restera plus que cinq secondes pour terminer le niveau.

Mets à jour le programme « Traversée périlleuse » pour activer la LED du séparateur décimal pendant les cinq dernières secondes du niveau.

Copyright © 2016 Eyrolles.

1. Après le calcul du temps restant dans boucle `niveau`, vérifie s'il reste moins de cinq secondes. Si c'est le cas, active le séparateur décimal, sinon laisse-le éteint :

```
secondesRestantes = DELAIS[niveau] - (time.time()
                                      - debut)
if secondesRestantes < 5:
    display.setdp(True)
else:
    display.setdp(False)
```

2. Exécute le programme. Lorsqu'il ne te restera plus que cinq secondes, le séparateur décimal s'allumera.

DÉFI



Au début de chaque niveau, utilise l'afficheur pour indiquer au joueur le niveau dans lequel il entre, avant de lui indiquer le nombre de diamants qu'il doit récupérer.

Tableau de références

Commandes	Description
`import thread`	Importe le module de threading de Python.
`thread.start_new_thread(fonction, (variable1, variable2))`	Appelle une fonction au sein d'un même thread.
`import time`	Importe le module de temps de Python.
`tempsActuel = time.time()`	Récupère l'heure actuelle.

Continuer l'aventure dans Minecraft et explorer les pistes qui s'offrent à toi

Minecraft t'offre le cadre idéal pour donner libre cours à ta créativité et partir à l'aventure. Sans compter que tu maîtrises maintenant davantage le jeu grâce à la programmation et que tu peux user de ton imagination sans bornes ! Alors, que faire maintenant que tu es venu à bout de ce livre ?

Voici quelques idées et ressources que j'ai recensées pour te donner un peu d'inspiration.

- Il n'y a rien de mieux que de créer des jeux pour t'entraîner en programmation. Jette un œil à `www.classicgamesarcade.com` pour trouver quelques idées.

Copyright © 2016 Eyrolles.

- Minecraft peut se jouer à plusieurs. Tu pourrais en profiter pour programmer des jeux destinés à plusieurs personnes.
- Interagir avec des objets électroniques te permet de connecter le monde de Minecraft au monde réel. Repousse les limites de tes connaissances en lisant À l'*aventure avec Arduino* ! http://www.editions-eyrolles.com/Livre/9782212143140/a-l-aventure-avec-arduino.

Il existe tout un tas de sites web qui répertorient des données en libre accès *(open data)*. Pourquoi n'y ferais-tu pas un tour ? Tu pourrais t'amuser à intégrer Minecraft à des plates-formes telles que Twitter (dev.twitter.com) ou à des données météorologiques en temps réel, comme celles de Météo-France (www.data.gouv.fr/fr/organizations/meteo-france/#datasets) !



Niveau terminé : tu viens de terminer ta plus grande aventure dans Minecraft !

Copyright © 2016 Eyrolles.

Copyright © 2016 Eyrolles.



Annexe A

Pour aller plus loin

NOUS ESPÉRONS que les chapitres de ce livre t'auront inspiré autant d'idées que des bribes de code, et qu'elles t'auront fait acquérir de nouvelles compétences pour pousser la grande aventure de la programmation dans Minecraft un peu plus loin. Tu es maintenant libre de continuer à explorer d'autres pistes en faisant jouer ton imagination ! Si tu ne sais pas encore par où commencer ou si tu as une idée de projet, mais que tu ne sais pas encore comment la concrétiser, voici quelques pistes intéressantes que tu peux creuser pour t'inspirer.

Sites Internet

Internet regorge de sites pratiques sur Minecraft, à tel point qu'il est parfois difficile de s'y retrouver. C'est pourquoi Martin et David t'ont préparé une liste de sites web utiles pour apprendre, jouer et programmer dans Minecraft :

Minecraft

- `www.editions-eyrolles.com/dl/14292` – Il s'agit du site Internet de ce livre. Il inclut un chapitre complet en guise de bonus (l'Ascenseur Minecraft), des petits guides de référence, des badges, des scripts complets et des vidéos téléchargeables pour chacun des projets de ce livre.

Copyright © 2016 Eyrolles.

- www.stuffaboutcode.com – C'est le blog de Martin : un vrai succès ! Il contient toute une section dédiée à Minecraft avec des tonnes d'idées de projets et d'expériences à réaliser tout seul. Martin y héberge également un guide sur l'API Minecraft : la référence en la matière. Tu n'en trouveras pas de meilleure ailleurs !

- http://arghbox.wordpress.com – Craig Richardson développe régulièrement de fabuleux programmes pour Minecraft en soutien aux nouveaux programmes d'enseignement en informatique. Prends le temps de feuilleter son livre de programmation open source sur Minecraft et découvre sa fameuse télécommande Minecraft fabriquée à partir de fruits !

- www.minecraftforum.net – C'est le forum officiel de Minecraft. Tu y trouveras de l'aide sur tous les sujets en lien avec Minecraft, tels que les nouvelles fonctionnalités, l'installation des serveurs ainsi que des conseils pour les modes Créatif et Survie.

- http://minecraft-fr.gamepedia.com/Minecraft_Wiki – Minecraft Wiki est un site collaboratif qui regroupe des informations sur Minecraft, ce qui veut dire que tu peux aussi contribuer à l'enrichir !

- www.scarabcoder.com – Il s'agit du blog de Nicholas Harris, un jeune programmeur qui a appris à coder en utilisant Minecraft sur Raspberry Pi. Il écrit des articles sur ses aventures dans la programmation et la technologie. Ne manque pas ses publications sur les plug-ins Bukkit.

- www.reddit.com/r/MCPI – Reddit est un site web communautaire de partage de liens où les internautes sont invités à voter pour ou contre. On y voit régulièrement apparaître des projets de programmation Minecraft.

- http://mcpipy.wordpress.com et https://github.com/brooksc/mcpipy – Ces sites regroupent toute une collection de ressources de programmation Minecraft. Tu peux les trouver ailleurs, mais bien des gens (y compris David) ont entendu parler de Martin O'Hanlon pour la première fois sur ces sites et sur les pages Reddit.

- www.minecraftmaps.com – Minecraft te permet de charger et de sauvegarder des cartes géographiques, qui ne sont autres que des aperçus complets de mondes Minecraft. Ce site web a réalisé un immense inventaire de cartes communautaires que tu peux charger dans ton monde Minecraft et continuer de développer dans ton jeu.

- http://minecraft.curseforge.com – C'est un site open source d'hébergement et d'échange (un peu comme sourceforge, mais dédié à Minecraft), où les joueurs peuvent héberger leurs projets et les télécharger depuis www.curse.com.

- http://minecraft-seeds.net – Les mondes de Minecraft sont générés par un algorithme intégré au jeu. Des numéros sont générés aléatoirement à partir d'un numéro d'origine. Si tu saisis le numéro d'un monde que tu connais, celui-ci apparaîtra dans ton jeu. Ce site recense plusieurs listes de numéros (ou *seeds*) que tu peux saisir pour créer des mondes spécifiques dans Minecraft.

- www.mcedit.net et www.worldpainter.net – Ces outils sont téléchargeables gratuitement et te permettent de concevoir et de dessiner tes propres cartes à l'aide d'outils d'édition.

Copyright © 2016 Eyrolles.

- http://mcreator.pylo.si – Doté d'une interface graphique avec pointeur, MCreator permet de générer des mods pour Minecraft sans programmation. Tu peux t'amuser à créer de magnifiques mods Minecraft comme de nouveaux types de blocs, des mobs, des armures, des commandes et bien d'autres éléments.
- www.firetechcamp.com – Fire Tech Camp organise des classes de codage au Royaume-Uni pour les adolescents, pendant leurs vacances scolaires. On y enseigne des disciplines aussi fascinantes que la conception de jeux vidéo, la construction dans Minecraft, la programmation en Python, Arduino et bien d'autres cours spécialement conçus pour les adolescents férus de technologies.

Python

- https://www.python.org et https://docs.python.org/2 – Ce sont les sites officiels de documentation et de téléchargement du langage de programmation Python.
- www.codecademy.com/tracks/python – C'est un cours en ligne gratuit. Tu peux suivre les cours toi-même pour apprendre petit à petit le langage de programmation Python.
- http://inventwithpython.com/chapters – Il s'agit d'un livre (en anglais) gratuit consultable en ligne et qui contient de super projets Python pour t'apprendre à programmer dans ce langage.
- https://docs.python.org/2/library/idle.html – Ce site est le guide d'utilisation officiel de l'interface de programmation IDLE. Comme il s'agit d'un guide, il est rédigé de manière très précise et ne contient pas d'images. Nous préférons utiliser ces tutoriels, créés par Anne Dawson : www.annedawson.net/Python_Editor_IDLE.htm
- www.geany.org – IDLE est une petite interface de programmation très simple d'utilisation. Mais après un certain temps, tu auras probablement envie de passer à une interface plus complète. Essaye Geany à la place, c'est un environnement bien plus sympa à utiliser.
- http://blog.whaleygeek.co.uk – Il s'agit du blog de David. Il y publie toute une panoplie de projets Python et Raspberry Pi, des trucs et astuces, ainsi que des jeux de cartes mémoire pour te rappeler les principes de la syntaxe en Python et sur Minecraft. Imprime-les et conserve-les dans ta besace au cas où tu aurais besoin de te rafraîchir la mémoire !

Bukkit

- http://wiki.bukkit.org/Plugin_Tutorial – Le serveur Bukkit que tu as utilisé dans ce livre a été développé de manière collaborative et fonctionne avec des plug-ins. Ces plug-ins sont écrits en Java et tu peux en télécharger pour à peu près tout et n'importe quoi. Tu peux même créer tes propres plug-ins ! Pour savoir comment faire, rends-toi sur ce site web.

Copyright © 2016 Eyrolles.

- http://dev.bukkit.org/bukkit-plugins/raspberryjuice – Le RaspberryJuice que tu utilises dans ce livre est en fait un plug-in Bukkit qui expose l'intérieur de l'API Minecraft à travers l'interface MPCI qu'exploite Raspberry Pi.
- http://www.skpang.co.uk – Mr S.K.Pang est un bon revendeur de composants électroniques et de projets intéressants au Royaume-Uni. L'enseigne vend également une version pré-assemblée et préprogrammée de Sparkfun Arduino Pro Micro, ainsi qu'un T-Cobbler d'Adafruit pré-assemblé avec une étiquette pratique qui te permet de différencier chaque broche GPIO. Si tu souhaites terminer plus rapidement les chapitres 5 et 9, ainsi que le chapitre bonus, essaye de te procurer ces dispositifs pré-assemblés. Il existe des sites équivalents français comme www.selectronic.fr, www.conrad.fr ou encore www.lextronic.fr.
- http://www.sparkfun.com – Sparkfun propose une très belle gamme de produits électroniques et des guides de prise en main très bien écrits qui te permettront de faire tes branchements en toute simplicité. Il est l'éditeur du circuit Arduino Pro Micro que tu as probablement utilisé dans les chapitres 5 et 9 ainsi que dans le chapitre 10 (bonus), mais tu dois savoir que la plupart des produits en vente sur ce site ne sont ni pré-assemblés ni présoudés. Tu devras donc faire un petit effort supplémentaire pour t'en servir.
- http://www.maplin.co.uk – Maplin Electronics est un bon détaillant de produits électroniques.

Autres techniques et astuces d'automatisation

La programmation en Python n'est pas l'unique manière qui existe pour automatiser des tâches dans Minecraft. Le jeu dispose également de blocs de redstone et de commande, ainsi que d'étiquettes. Il est possible de combiner chacun de ces blocs pour actionner certaines parties de ton monde et les automatiser.

- http://minecraft.gamepedia.com/Tutorials/Command_Block – Les blocs de commande peuvent être déclenchés pour automatiser des tâches à ta place, en les connectant par exemple à une balise de redstone. Tu peux construire tout un tas d'objets grâce à cela. Jette un œil aux dispositifs pièges ou aux téléporteurs sur cette page de tutoriels.
- www.minecraftforum.net/topic/1969520-17-using-summon-give-datatags-in-map-making-tutorials – Ce site web montre comment utiliser les *Named Binary Tags* (NBT ou « étiquettes nominatives » en français) avec les commandes. Les NBT sont le format de sauvegarde interne de plusieurs types de données Minecraft. Sans programmer, tu peux exécuter des commandes avec des étiquettes et ainsi configurer toute une série d'objets Minecraft, tels que des blocs et des entités.
- www.minecraft101.net/redstone/redstone101.html – Le redstone est l'équivalent Minecraft de l'électricité. Simple en apparence, il te permet en fait de construire des dispositifs très complexes si tu assembles plusieurs petits circuits.

Copyright © 2016 Eyrolles.

Projets et tutoriels

Inventer ses propres projets est vraiment très grisant, mais un petit coup de pouce est parfois le bienvenu quand on est en panne d'inspiration. Les sites web répertoriés ici rassemblent une belle collection de projets d'exemple, qui ont pour la plupart été développés par des passionnés et des codeurs de la communauté.

- `http://minecraft.gamepedia.com/Tutorials/Advanced_redstone_circuits` – Ce tutoriel montre comment créer une immense structure automatisée en assemblant plusieurs petits circuits de Redstone et t'apprend même, pas à pas, à fabriquer un ordinateur entier à partir de Redstone !
- `https://learn.adafruit.com/search?q=minecraft` – Adafruit possède une magnifique collection de tutoriels dont les instructions ont été rédigées par ses équipes et par d'autres membres de la communauté. Elle a par ailleurs une nouvelle section dédiée aux projets Minecraft et à l'électronique, qui a l'air de bien se remplir.
- `www.stuffaboutcode.com/2013/09/minecraft-ordnance-survey-map-rastrack.html` – Ce projet est le premier sur lequel David et moi avons travaillé, alors que nous ne nous étions même pas encore rencontrés ! Il s'agit d'un *mashup*, c'est-à-dire un savant mélange de programmes reliés les uns aux autres. Nous avons récupéré des informations sur le site web Rastrack de Ryan Walmsley (`www.rastrack.co.uk`) et des données de cartes en libre accès sur le site de l'Ordnance Survey, puis nous avons recréé le Royaume-Uni dans son intégralité dans Minecraft en inondant le ciel de framboises à l'effigie de Raspberry Pi ! Si tu es curieux d'en savoir plus, regarde la vidéo que j'ai publiée sur mon blog.
- `http://hackaday.com/2013/01/30/controlling-minecraft-with-a-raspberry-pi` – Tu peux aussi prendre le contrôle de dispositifs électroniques depuis Minecraft rien qu'en utilisant Bukkit avec Redstone. En communiquant avec Raspberry Pi, un plug-in Bukkit te permet de prendre le contrôle et de téléguider des dispositifs électroniques à partir de leviers et de balises dans Minecraft.
- `www.instructables.com/howto/minecraft` – Le site web instructables possède une immense collection d'instructions très claires qui t'expliquent comment construire des projets pas à pas. Le site dispose désormais d'une section entièrement dédiée à Minecraft.

Copyright © 2016 Eyrolles.

Vidéos

Comme beaucoup de jeux aujourd'hui, Minecraft est très visuel. L'une des meilleures façons de trouver des idées et de l'inspiration consiste d'ailleurs à regarder ce que font les autres. La plupart des vidéos qu'on trouve sur Internet ont été créées par des blogueurs et des passionnés devenus célèbres grâce à cette activité, d'autant plus qu'ils publient des vidéos (en anglais) chaque semaine, voire chaque jour.

- `http://minecraft.gamepedia.com/Tutorials/Elevators` – Dans le chapitre bonus qui se trouve sur le site Internet du livre, on t'explique comment programmer un ascenseur Minecraft en parfait état de marche. Or, il existe plusieurs manières de construire un ascenseur dans Minecraft. Celles-ci sont toutes très bien expliquées dans ce wiki et sont accompagnées de superbes vidéos montrant l'ascenseur en action.
- `www.youtube.com/user/sethbling` – Seth Bling dispose d'une chaîne YouTube très active dans laquelle il publie d'innombrables vidéos de projets et d'idées pour Minecraft. Ne manque surtout pas celle sur les Jeux Olympiques au TNT !
- `www.youtube.com/user/stampylonghead` – Strampy publie au moins une nouvelle vidéo Minecraft par jour sur sa chaîne YouTube et s'amuse beaucoup en construisant de superbes structures.
- `www.youtube.com/user/scarabcoder` – Nicholas Harris a créé une bonne chaîne YouTube proposant de nombreux projets intéressants. Il est également l'auteur d'un livre électronique sur la programmation dans Minecraft que tu peux acheter sur Amazon.
- `www.youtube.com/user/SimplySarc` – Jette un œil à la chaîne de SimplySarc, notamment sa vidéo sur son appareil photo « Vanilla Camera » (`www.youtube.com/watch?v=NyMHCabq_rs`) qui t'explique en détail comment créer une caméra dans Minecraft à l'aide de blocs de commande, sans recourir à aucun mod !
- `www.youtube.com/watch?v=7t4bH7Z-Yt4` – Dans cette vidéo extraordinaire, Martin O'Hanlon en personne explique comment transformer des blocs en bombes prêtes à exploser rien qu'en les frappant, grâce à un programme Python.
- `www.youtube.com/user/ThatMumboJumbo` – MumboJumbo réalise une série YouTube sur les techniques de construction en s'armant de Redstone, de pistons et de leviers. Tu verras que lui aussi construit un ascenseur Minecraft, mais en n'utilisant que des pistons. Pour savoir comment il a fait, jette un œil à sa vidéo : `www.youtube.com/watch?v=jQulvbivtYI`.
- `www.youtube.com/user/HiFolksImAdam` – En vrai prestidigitateur, Adam utilise les blocs de commande pour créer d'impressionnants effets visuels. Il a même construit un TARDIS (une machine à voyager dans le temps), quant à lui beaucoup plus grand de l'intérieur que de l'extérieur !

Copyright © 2016 Eyrolles.

- www.youtube.com/user/AsdjkeAndBro – Asdjke et Bro se sont amusés à développer une multitude de petits jeux dans Minecraft. L'un de nos favoris est la bataille navale : www.youtube.com/watch?v=6AbPlT-cAm8.

Livres

Il est assez agréable de pouvoir compter sur un livre imprimé de temps à autre et de l'avoir à portée de main. Tu peux le feuilleter quand tu veux, y introduire des notes colorées ou marquer les pages qui t'intéressent, tout en progressant dans tes aventures Minecraft. David et moi avons beaucoup appris de ces auteurs et nous pensons qu'ils peuvent t'apporter bien d'autres idées de projets et de sources de divertissement que celles que nous t'enseignons dans ce livre. Nous te recommandons tout particulièrement le livre *À l'aventure avec Arduino* de Becky Stewart (Eyrolles) : il est tout indiqué si tu veux continuer l'aventure sur PC ou sur Mac en utilisant les mêmes circuits électroniques que dans les chapitres 5 et 9 de ce livre, ainsi que dans le chapitre bonus. Le circuit Arduino Pro Micro que tu t'es procuré est compatible avec tous les projets que Becky développe dans son livre. Il te suffira de suivre les instructions du site Sparkfun pour réinitialiser le micrologiciel du circuit avant de te mettre en route pour de nouvelles aventures ! Attention, seuls les deux derniers ouvrages de cette liste sont en français.

- *Minecraft for Dummies* de Jacob Corderio (Wiley, 2013)
- *Minecraft: The Official Beginners' Handbook* (Egmont, 2013)
- *Minecraft: The Official Redstone Handbook* (Egmont, 2013)
- *Minecraft: Awesome Building Ideas for You* (Createspace, 2014)
- *Learn to Program with Minecraft Plug-ins* d'Andy Hunt (Pragmatic Bookshelf, 2014)
- *Adventures in Raspberry Pi* de Carrie Anne Philbin (Wiley, 2013)
- *Adventures in Python* de Craig Richardson (Wiley, 2014)
- *À l'aventure avec Arduino* de Becky Stewart (Eyrolles, 2015)
- *Python pour les kids* de Jason R. Briggs (Eyrolles, 2015)

Copyright © 2016 Eyrolles.

Copyright © 2016 Eyrolles.

Index

Symboles

!= 109
() 105
[] 105, 106, 110
\ 85, 196
64
= 53
== 53
2D 179
2D et 3D 205
3D 179
.id 84
{}(langages C et C++) 196
\n 188

A

abs() 119
afficheur
  7 segments 147, 148
    brancher 149
    circuit 151
  listes 152
algorithme
  de Bresenham 209
  de tracé d'arc de cercle par point milieu 211
and (mot-clé) 49
anode 138
  commune 149
anyio 134
API 40
  définition 42
appeler une fonction 103, 208
  définition 104
append() 107, 221
Arduino 128
  connecté à la plaque d'essais 140
  LED 143
  LED et résistance raccordées 138
  pôle négatif 140
  pôle positif 140
Arduino Pro Micro 128
as (mot-clé) 141

B

BCM 142
bloc 69
  AIR 114
  block.AIR.id 75
  construire plusieurs blocs 70
  créer 68
  donnée supplémentaire 83
  .id 84
  ID 75
  intelligent 227
  liste dans fichier de données 170
  WOOL 83
block.AIR.id 75
booléen 193
boucle
  de jeu 45
  externe 176
  for 71, 177
  imbriquée 174
    définition 176
  infinie
    définition 45
  interne 176
  while not gameOver 268
  while not gameOver and not niveauTermine 268
  while True 46
boucle infinie 45
bouton
  raccordé à un GPIO via un circuit Arduino 158
  raccordé à un Raspberry Pi 156
  raccordé un circuit Arduino 156
bouton-poussoir 131
break (instruction) 193
broches GPIO 133, 134, 139, 145
  définition 133
Bukkit 21
  arrêter 29
  définition 21

C

casse 32
cathode 138
  commune 149
cercles 210
chaîne de caractères 44
  définition 44
classe 207
clear() 239
clone() 230
commentaires
  # 64
constante, définition 59
Constructeur BMP de Minecraft 166
coordonnée 38
  absolue 70
    définition 70
  afficher 39
  définition 38
  relative 69
    définition 70
courant 131
  définition 132
CraftBukkit 6, 12
Créatif (mode) 7
CSV (fichier) 171
  définition 171
  module d'importation 178
  répertoire de fichiers 200

Copyright © 2016 Eyrolles.

D

datetime.datetime.now() 214
def (mot-clé) 85
diode électroluminescente 131
    définition 131
donnée supplémentaire 83
donneur d'indices 167
drawCircle() 210, 211
drawFace() 219
drawLine() 208
drawSphere() 211

E

EDI. Voir IDLE
elif (mot-clé) 109
else (instruction)
    indentation 100
else, instruction 62, 63

F

face 218
    définition 219
fichier
    CSV 171
fichier de données
    métadonnées 180
fichiers de données 165
    fin de ligne 167
    intérêt 166
finally (mot-clé) 145
fin de ligne (fichier) 167
    définition 167
fonctions 84, 86
    abs() 119
    append() 221
    clear() 239
    clone() 230
    datetime.datetime.now() 214
    définition 85
    drawCircle() 210, 211
    drawFace() 219
    drawLine() 208
    drawSphere() 211
    getBlock() 98
    getLine() 231
    int() 192, 217
    len() 246
    math.cos() 217
    math.radians() 217
    math.sin() 217
    math.sqrt() 233
    MinecraftShape 237
    moveBy() 239
    open() 170
    print() 43
    raw_input() 154, 192
    readlines() 177
    round() 217
    setBlock() 98
    split() 177
    str() 281
    strip() 168
for
    mot-clé 85
    variable de comptage 72
for (boucle) 71, 114, 177
    construire plusieurs blocs 72
    construire une immense tour 73
for/in (instruction) 114

G

gameOver (variable) 257
General Purpose Input/Output 133
géorepérage 55
    définition 56
getBlock() 98
getLine() 231
glob (module et fonction) 201
GPIO 8, 143
GPIO.cleanup() 144
Graphical User Interface (GUI) 18

I

IDLE 12, 16
if
    else 63
    mot-clé 85
if (instruction)
    indentation 100
if, instruction 48, 49
indentation 46
    définition 47
    Python vs C et C++ 196
index 105
    définition 106
instanciation 207
instruction
    break 193
    définition 38
    else 62, 63
    for/in 114
    if 48, 49, 63
    print 42
    range 73
    raw_input 77
    setBlocks 75
    sleep 60
int() 192, 217
interface
    définition 42

K

KeyboardInterrupt 145
kit de démarrage
    installer sur Mac 24
    installer sur PC 22
    minecraftstuff 206

L

LED 131, 132, 139, 144
    à 7 segments 150
    allumer avec un ordinateur 138
    Arduino 143
    astuce 141
    contrôler 137
    définition 131
    et résistance raccordées 138
len() 246
line (variable) 177
liste Python 104, 107, 110
    [] 110
    définition 104

M

math.cos() 217
math.radians() 217
math.sin() 217
math.sqrt() 233
métacaractère 202
    définition 202
métadonnées 180
    définition 180
Minecraft
    API 167
    à propos de 3
    blocs 38
    configurer pour PC et Mac 21
    coordonnées 39
    démarrer 26
    hsitoire de 4
    installer 11
    launcher 28
    wiki 11
Minecraft Pi 6, 12
Minecraft Pi Edition 17, 18
MinecraftShape 237
MinecraftStuff 7
minecraftstuff (module) 206
module 206
Mojang 4
moveBy() 239
multithreading 258

N

nombre
    aléatoire 234

Copyright © 2016 Eyrolles.

nombres aléatoires 91
définition 92
produire 91
nombres pseudo-aléatoires 92
None (valeur) 117

O

open() 170

P

paramètres 208
passer des valeurs en paramètre 208
pilotes (configurer) 134
Pi-T-Cobbler 139
plaque d'essais
connectée à Arduino 140
plaque d'essais 131
définition 130
plug-in
définition 21
RaspberryJuice 21
pôle
négatif 140
positif 140
polyèdre 225
polygones 218
faces 218
pop() 107
pop (méthode) 107
postToChat 45
pos (variable) 69
pos, variable 42
prévisible (définition) 234
print, instruction 42, 43, 44
probabilité (définition) 234, 235
programmation
parallèle. Voir multithreading
programmation orientée objet 207
programme 37
arrêter 35
créer 30
définition 38
exécuter 33
prototype
définition 129, 130
pyramides 221
Python 12, 15
= 53
== 53
boucle imbriquée 176
constante 59
contrôler l'afficheur 7 segments 151
définition 15
en savoir plus 22
fonctions 84
indentation 9, 46, 47
installer 21
installer sur Mac 24
installer sur PC 22
lignes trop longues 196
liste 104, 107, 110
définition 104
listes 176
module 153
sensible à la casse 32
télécharger et installer 23, 25
versions 2 et 3 77, 193

R

racine carrée (définition) 233
radians 217
random 91, 92, 235
range (instruction) 72, 73
RaspberryJuice 21
RaspberryJuice (plug-in) 12
Raspberry Pi 11
à propos de 16
configurer 17, 18
démarrer 19
installer Minecraft 18
LED et résistance raccordées 138
Pi-T-Cobbler 139
Raspberry Pi B+ 128
raw_input() 154, 192
raw_input (instruction) 77
readlines() 177
répertoire de fichiers 200
résistance 131
définition 132
de tirage 157
et LED raccordées 138
retour chariot. Voir fin de ligne
return (mot-clé) 191, 192
définition 192
root 145
round() 217

S

segments 208
sensible à la casse 32
séquentiel (définition) 258
setBlock() 70
setBlocks (instruction) 75
sleep, instruction 60
sphères 211
split() 177
str() 281
strip() 168
sudo (mot-clé) 145
Survie (mode) 7
syntaxe 54
définition 55

T

tension 131
définition 131
théorème de Pythagore 233
trigonométrie 212
définition 213
try/finally 145
tuple 111, 260
définition 260

V

variable 42
booléenne
définition 193
False 193
True 193
de comptage 72
définition 72
de contrôle de boucle 73
définition 43
globale 88
définition 88
pos 42, 69
variables
line 177

W

while
mot-clé 85
while not gameOver and not niveauTermine (boucle) 268
while not gameOver (boucle) 268

Copyright © 2016 Eyrolles.

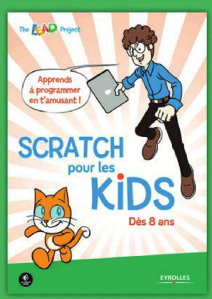
APPRENDS À CODER DÈS 5 ANS !

NOUVELLE MATIÈRE DU PROGRAMME, DU CE1 À LA 3E !

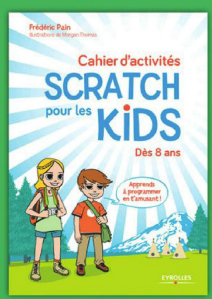
SCRATCH



Dès 5 ans
160 pages – 15,90 €



Dès 8 ans
160 pages – 15,90 €

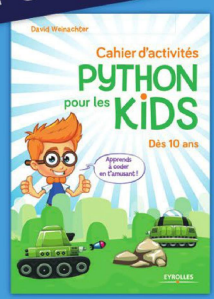


Dès 8 ans
64 pages – 12 €

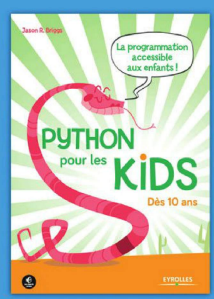


Dès 10 ans
288 pages – 25 €
À paraître en septembre

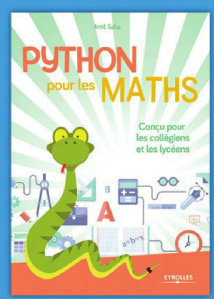
PYTHON



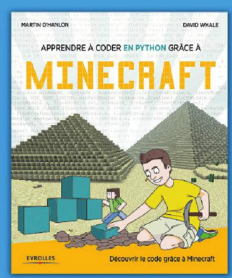
Dès 10 ans
64 pages – 12 €



Dès 10 ans
332 pages – 22,90 €

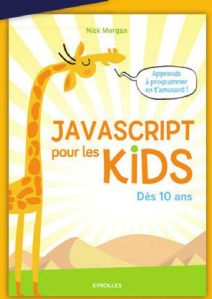


Dès 10 ans
300 pages – 22,90 €
À paraître en juin

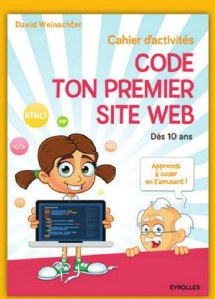


Dès 10 ans
350 pages – 24,90 €
À paraître en juin

WEB

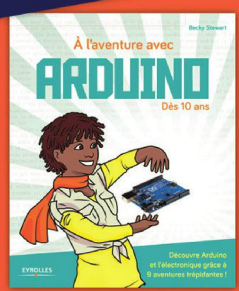


Dès 10 ans
364 pages – 22,90 €

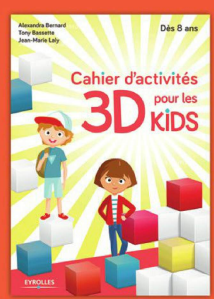


Dès 10 ans
152 pages – 19,90 €
À paraître en septembre

MAKERS



Dès 10 ans
352 pages – 25 €



Dès 8 ans
64 pages – 12 €
À paraître en septembre

EYROLLES

Pour en savoir plus : www.editions-eyrolles.com/go/coderaveceyrolles