Приключение 7
Наделение блоков способностью мыслить

КОМПЬЮТЕРЫ НЕ УМЕЮТ ДУМАТЬ. Они вообще не способны мыслить и делают лишь то, что скажет программист. Однако вы можете придать компьютеру вид мыслящей и принимающей решения конструкции, запрограммировав «понимание» происходящего и снабдив компьютер правилами выбора «решения», что делать дальше. Тем самым вы откроете для себя массу интересного.

В этом приключении вы запрограммируете друга-блок, который будет следовать за вами повсюду, не упуская вас из виду. Также вы создадите летающую тарелку, преследующую персонажа, которая, оказавшись над ним, попытается телепортировать его внутрь.

Еще вы узнаете, как заставить блок двигаться и выбирать самый короткий путь, а также как с помощью модуля `random` придать компьютеру вид мыслящего существа. Кроме того, вы создадите фигуры с помощью функций `MinecraftShape` из модуля `minecraftstuff` (включен в начальный комплект инструментов).

Ваш друг-блок

Minecraft может быть пустынным миром. Но ваш персонаж не обязан томиться в одиночестве: вы можете создать блок, повсюду следующий за ним, который будет разговаривать и станет его другом (рис. 7.1).

Чтобы запрограммировать блок, ставший другом персонажу, вам нужно научиться думать, как он. Он будет счастлив находиться рядом с персонажем, всюду ходить за ним и стараться быть как можно ближе. Если персонаж отойдет, друг-блок последует за ним. Если персонаж уйдет слишком далеко, блок загрустит, прекратит движение и будет стоять на месте, пока персонаж не вернется и не обнимет его (встанет рядом с блоком).

Программа друга-блока делится на две части:

- **Правила, с помощью которых блок решает, что делать дальше**: эта часть программы поможет блоку решить, стоит ли продолжать идти за персонажем (целью) или лучше остановиться.
- **Программный код, перемещающий блок за персонажем**: достигнув цели, блок снова будет опираться на правила, желая определить, что делать дальше.

Рис. 7.1. Создайте друга-блока, который будет сопровождать вашего персонажа в его приключениях

Следуя за целью, он должен перемещаться по поверхности земли, не летать по воздуху и не прокладывать себе путь под землей! Добиться этого можно, вызвав функцию `mc.getHeight(x,z)` и передав ей горизонтальные координаты (`x`, `z`). В ответ функция вернет вертикальную координату (`y`) первого сверху блока, тип которого отличается от `AIR`.

На сайте книги *www.wiley.com/go/adventuresinminecraft* есть видеоурок, где показано, как создать друга-блока.

Напишите новую программу для создания друга-блока:

1. Откройте редактор IDLE, создайте новый файл, выбрав в меню пункт File ▸ New File (Файл ▸ Создать файл), и сохраните его в папке *MyAdventures* с именем *BlockFriend.py*.
2. Импортируйте модули `minecraft`, `block`, `minecraftstuff`, `math` и `time`:

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

3. Первое, что нужно сделать, — создать функцию для вычисления расстояния между двумя точками. С ее помощью блок определит, как далеко он находится от персонажа:

```
def distanceBetweenPoints(point1, point2):
  xd = point2.x - point1.x
  yd = point2.y - point1.y
  zd = point2.z - point1.z
  return math.sqrt((xd*xd) + (yd*yd) + (zd*zd))
```

4. Теперь надо решить, как далеко должен оказаться персонаж, чтобы блок перестал за ним следовать. Создайте константу для хранения расстояния, которое «слишком далеко». Я выбрал число 15, то есть, когда блок и персонаж окажутся на расстоянии 15 блоков друг от друга, друг-блок прекратит следовать за персонажем:

```
TOO_FAR_AWAY = 15
```

5. Создайте объекты `Minecraft` и `MinecraftDrawing`:

```
mc = minecraft.Minecraft.create()
mcdrawing = minecraftstuff.MinecraftDrawing(mc)
```

6. Создайте переменную для хранения настроения блока. В данном приключении блок счастлив (happy) либо расстроен (sad). Присвойте переменной значение `"happy"` (счастлив):

```
blockMood = "happy"
```

7. Создайте блок на расстоянии нескольких блоков от персонажа, для чего сначала определите его позицию, затем добавьте 5 к координате `x` и с помощью функции `getHeight()` найдите координату `y`, чтобы блок оказался на поверхности земли:

```
friend = mc.player.getTilePos()
friend.x = friend.x + 5
friend.y = mc.getHeight(friend.x, friend.z)
mc.setBlock(friend.x, friend.y, friend.z,
            block.DIAMOND_BLOCK.id)
mc.postToChat("<block> Hello friend")
```

8. Создайте цель для блока, к которой он будет стремиться. Первоначально определите координаты цели, совпадающие с текущими координатами блока, чтобы тот не начал движение сразу после запуска программы:

```
target = friend.clone()
```

> Чтобы скопировать координаты в мире Minecraft, нужно использовать функцию `clone()`: функции Minecraft API возвращают координаты в виде объектов языка Python, отличающихся от обычных переменных. Например, если выполнить копирование инструкцией `target = friend` и потом изменить значение `friend.x` в объекте `friend`, значение `target.x` в объекте `target` тоже изменится.
>
> 
>

9. Начните бесконечный цикл, чтобы программа выполняла его вечно. (Обратите внимание: весь программный код ниже принадлежит этому циклу.)

```
while True:
```

10. Получите координаты персонажа и вычислите расстояние между персонажем и другом-блоком с помощью функции `distanceBetweenPoints()`:

```
pos = mc.player.getTilePos()
distance = distanceBetweenPoints(pos, friend)
```

11. Примените правила, чтобы блок определил, что делать дальше. Если он «счастлив» (happy), предложите ему сравнить расстояние до персонажа с константой «слишком далеко». Если расстояние меньше, чем «слишком далеко», переместите цель в позицию персонажа, а если больше — измените настроение на «расстроен» (sad) (рис. 7.2):

```
if blockMood == "happy":
  if distance < TOO_FAR_AWAY:
    target = pos.clone()
  elif distance >= TOO_FAR_AWAY:
    blockMood = "sad"
    mc.postToChat("<block> Come back. You are too far ↩
away. I need a hug!")
```

Рис. 7.2. Друг-блок расстроен

12. Теперь надо указать, что в противном случае (когда расстояние не меньше, чем «слишком далеко») блок «расстроен» и останавливается, чтобы дождаться, пока персонаж подойдет к нему на расстояние одного блока (расстояние объятий), после чего настроение блока изменится на «счастлив»:

```
elif blockMood == "sad":
  if distance <= 1:
    blockMood = "happy"
    mc.postToChat("<block> Awww thanks. Let's go.")
```

13. Следующий фрагмент кода перемещает блок к его цели:

```
if friend != target:
```

14. Найдите все блоки между другом и его целью с помощью функции `getLine()` объекта `MinecraftDrawing`:

```
blocksBetween = mcdrawing.getLine(
  friend.x, friend.y, friend.z,
  target.x, target.y, target.z)
```

Функция `getLine()` действует примерно так же, как функция `drawLine()` (см. Приключение 6). Только вместо строительства линии из блоков она возвращает список координат (`x`, `y`, `z`) блоков, находящихся на прямой линии между двумя указанными заданными множествами координат `x`, `y`, `z`.

15. Непосредственно под предыдущим кодом добавьте цикл, который выполнит итерации по всем блокам между другом и целью и переместит друга к цели:

```
for blockBetween in blocksBetween[:-1]:
  mc.setBlock(friend.x, friend.y, friend.z, block.AIR.id)
  friend = blockBetween.clone()
  friend.y = mc.getHeight(friend.x, friend.z)
  mc.setBlock(friend.x, friend.y, friend.z,
              block.DIAMOND_BLOCK.id)
  time.sleep(0.25)
target = friend.clone()
```

Когда цикл `for` завершится, друг-блок достигнет своей цели. Поэтому скопируйте координаты блока в координаты цели. В данном случае блок прекратит движение.

Программа перемещает блок, очищая его в предыдущей позиции (это объясняет, почему устанавливается тип блока `AIR`), меняя координаты друга-блока на координаты следующего блока в линии и воссоздавая блок в этой позиции.

Обратите внимание на разницу в именах переменных `blockBetween` и `blocksBetween`: первая хранит блок, в котором находится друг, вторая — список всех блоков между позициями, где друг находился первоначально и куда он должен переместиться.

Скорость движения блока определяется задержкой после перемещения друга к следующему блоку — `time.sleep(0.25)`. Если задержка слишком короткая, блок будет двигаться слишком быстро, и персонаж не сможет от него убежать, если чересчур долгая — блок будет двигаться медленно и раздражать вас своей медлительностью.

16. Добавьте небольшую задержку в конце цикла, чтобы программа не давала слишком большую нагрузку на игру Minecraft, постоянно запрашивая позицию персонажа:

```
time.sleep(0.25)
```

17. Сохраните и запустите программу.

Друг-блок появится недалеко от персонажа и будет следовать за ним, пока они не окажутся слишком далеко друг от друга. Когда это случится, друг-блок остановится, и персонажу придется подойти к нему, встать рядом, прежде чем тот продолжит «преследование».

Полный код программы можно загрузить на сайте книги www.wiley.com/go/adventuresinminecraft.

УГЛУБЛЯЕМСЯ В КОД

Для вычисления расстояния между другом-блоком и персонажем вы написали функцию с именем `distanceBetweenPoints(point1, point2)`. Вычисление точного расстояния между двумя точками производится на основе теоремы Пифагора. Если помните, в Приключении 4 вам уже предлагали подумать, как применить теорему Пифагора, в разделе «Добавление вывода подсказок».

Вычисление расстояния выполняется, как описано ниже:

1. Находится разность между значениями `x`, `y`, `z` в `point1` и `point2`:

```
xd = point2.x - point1.x
yd = point2.y - point1.y
zd = point2.z - point1.z
```

2. Вычисляются квадраты разностей координат точек:

```
xd*xd
```

3. Вычисляется сумма квадратов разностей:

```
(xd*xd) + (yd*yd) + (zd*zd)
```

4. Вызывающей программе возвращается расстояние между двумя точками, которое вычисляется как **квадратный корень** из суммы квадратов разностей с помощью функции `math.sqrt()`:

```
return math.sqrt((xd*xd) + (yd*yd) + (zd*zd))
```

Посетите страницу www.mathsisfun.com/algebra/distance-2-points.html, чтобы познакомиться с теоремой Пифагора и узнать, как ее можно использовать, чтобы найти расстояние между двумя точками.

Друг-блок перемещается в сторону персонажа, отыскивая блоки между ним и персонажем, а затем в цикле двигается по этим блокам:

```
for blockBetween in blocksBetween[:-1]:
```

Здесь `[:-1]` сообщает интерпретатору Python, что цикл должен обойти все блоки в списке `blocksBetween`, пока не окажется на блоке рядом с персонажем, а не у него на голове.