第3章 函数

代码由许多命令组成，将实现一个特定功能的命令组合在一起就是函数。有了函数的帮助，在编程时就无须多次输入重复的命令，只需要将这些重复的命令定义成函数，需要使用时直接调用就可以了。本章介绍函数的定义和调用，以及如何使用函数嵌套，通过关卡案例，大家可以更好地掌握函数的使用方法。

3.1 定义和调用


老师，基本命令我已经全学会了。


太棒了！ 老师想问你一个问题。


什么问题？


用 left() 命令如何表达向右转呢？好好想想！


我想到了，使用 3 次左转的命令就实现向右转的效果。


真棒，没错。


我有个问题，如果每次用到向右转，是不是都要使用 3 次向左转呢？


这个问题问得很好，今天给大家介绍一个新知识——函数，通过函数就能解决这个问题。

函数就是将实现特定功能的一组命令组合在一起。在下面的代码中，使用 3 个 left() 命令实现了向右转功能，这就是一个函数。函数有函数名称，这个函数名称为 TurnRight，实现功能的命令组合在大括号中。这个给函数命名和实现具体功能的过程称为“函数定义”。

代码

函数定义	函数调用

void TurnRight(){
left();
left();
left();
}
	TurnRight();

函数被“定义”好以后，就可以使用了。以刚刚定义的 TurnRight() 函数为例，在输入 TurnRight() 后，计算机系统会找到 TurnRight() 的函数定义，执行函数大括号内的 3 个 left() 命令。这个使用函数的过程就是“函数调用”。

有了 TurnRight() 函数定义和函数调用，以后在实现右转功能时就不用再写 3 次左转命令了，直接调用 TurnRight() 函数就可以，这就是使用函数的好处。

函数定义	函数调用

void 函数名称 ()
{
命令 1;
命令 2;
⋮
}
	函数名称 ();

函数定义一般由 4 部分组成，第一部分是 void，表示这是一个“函数”；第二部分是函数名称，例如 TurnRight；第三部分是小括号，小括号里面可以放置函数的参数，参数的内容会在后面的章节中介绍；第四部分是函数主体，就是需要执行的命令集合，写在大括号里面，例如 3 个左转 left() 命令。

函数调用可以理解为函数的使用。函数调用包含函数名称、小括号和分号 3 部分内容。当进行函数调用时，计算机就会根据函数名称找到函数定义，执行大括号中的“命令集合”，例如调用 TurnRight() 函数，主角就会根据函数定义的内容，执行命令集合中的 3 次左转命令。

知识小课堂

函数书写规范：

1. 函数名称都是英文字母或英文字母 + 阿拉伯数字，不能写成汉字，不能带标点符号；

2. 函数名称区分大小写，TurnRight 、Turnright、turnright 代表不同的函数，因此书写函数名称以及调用时，函数名称大小写要一样；

3. 函数名如果由多个单词组成，单词之间不要有空格，单词首字母用大写字母；

4. 函数名称后面的小括号一定要加上；

5. 函数定义都要包含基本命令，因此大括号不能丢，而且要成对出现。

函数命名大家了解了吗？找一找下面的代码哪些地方出错了？

代码

```
Turn Rights (){
    left();
}}
```

以上代码有 3 处错误，大家都找出来了吗？

（1）缺少函数关键字 void；

（2）Turn 和 Right 中间不应有空格；

（3）大括号多了一个。

老师，使用函数要比多次重复输入命令好多了。

如果有好几处都要书写命令实现向右转，函数就会很方便。

哦，是这样呀。

函数只需要定义一次，就可以被多次使用，这样就不用每次都写很多的基础命令。

老师，如果我定义了函数，但是没有用，会不会有问题呢？

如果定义了函数但没有调用，程序就不会读取函数，是没有问题的。

使用函数的好处？

函数可以简化代码书写。例如定义了右转函数，每次需要右转的时候直接调用右转函数即可，无须每次都写多个 left() 命令。

3.2　函数嵌套

老师，函数的定义和调用我已经理解了。

非常好。

老师，我可以先定义一个函数，然后在另一个函数定义中调用之前的函数吗？

可以呀，这就是函数嵌套。

函数嵌套是指函数被定义后，在另外一个函数中调用被定义的函数。

伪代码

```
void 转身函数 (){
   left();
   left();
}

void 新函数 (){
   move();
   take();
   转身函数 ();
   move();
}
```

用一个例子说明什么是函数嵌套。首先定义一个“转身函数”，函数体里用两个 left() 命令实现转身功能。再定义一个“函数”，这个函数不仅使用了基础命令，还调用了自定义的“转身函数”。

这个例子说明，可以在一个函数中调用另一个函数，这就是函数嵌套。为了方便辨认，这里使用了中文的函数名，属于伪代码，在后面章节会给大家介绍。大家在定义函数的时候不能使用中文函数名，需要使用英文的函数名。

3.3 关卡案例

老师，学会了函数，在编写代码时方便了许多。

是的，可以多次使用，也可以放到别的函数中使用。

老师，函数要比基本命令难一些，我还要慢慢理解。

没关系，下面进行一些关卡练习，能够帮助大家更好地掌握函数运用。

3.3.1 命令组合

关卡说明

关卡编号：2-1

关卡难度：*

通关条件：南瓜 1 个，开关 0 个。

关卡目标：使用命令组合来右转。

关卡 2-1 场景图如图 3.1 所示。

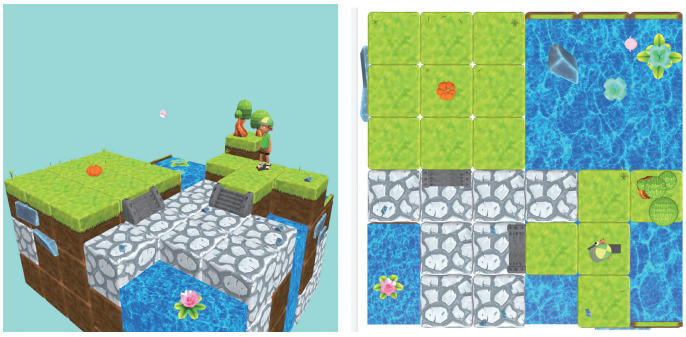

图 3.1　关卡 2-1 场景图

根据第 2 章学习的基本命令收集到南瓜，大家已经掌握的命令包括移动命令 move()、左转命令 left() 、切换开关命令 toggle() 和收集南瓜命令 take()。在这个关卡中，不能直接使用右转命令 right()，大家依旧要多次使用左转命令 left() 实现右转的功能。

代码

```
move();
move();
move();
left();
left();
left();
move();
move();
move();
take();
```

通过观察不难发现，主角向前移动 3 格后需要向右转。因为没有右转命令，只能使用 3 次左转命令来实现右转功能。最后再向前移动 3 格到达南瓜所在砖块，这样就可以收集到南瓜了。关卡 2-1 通关路线图如图 3.2 所示

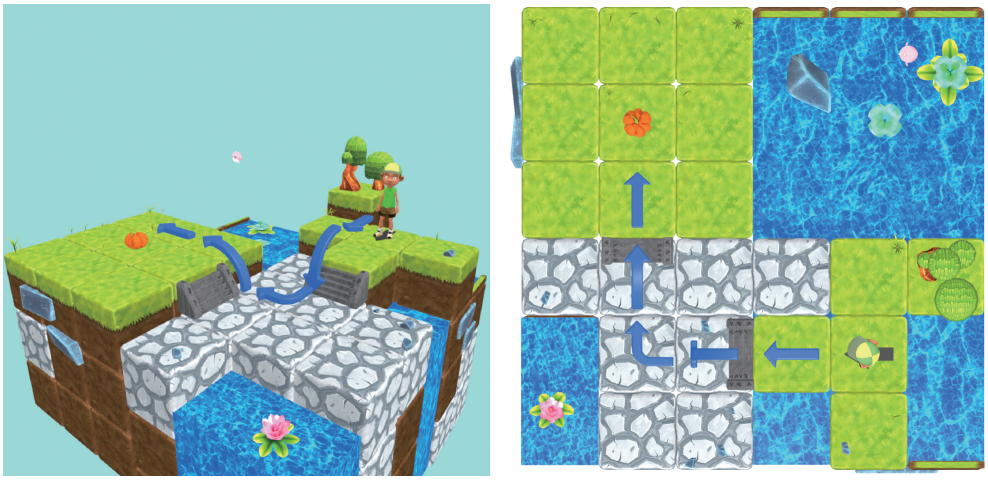

图 3.2　关卡 2-1 通关路线图

3.3.2 创建函数

关卡说明

关卡编号：2-2

关卡难度：*

通关条件：南瓜 0 个，开关 2 个。

关卡目标：定义函数来实现主角右转。

关卡 2-2 场景图如图 3.3 所示。

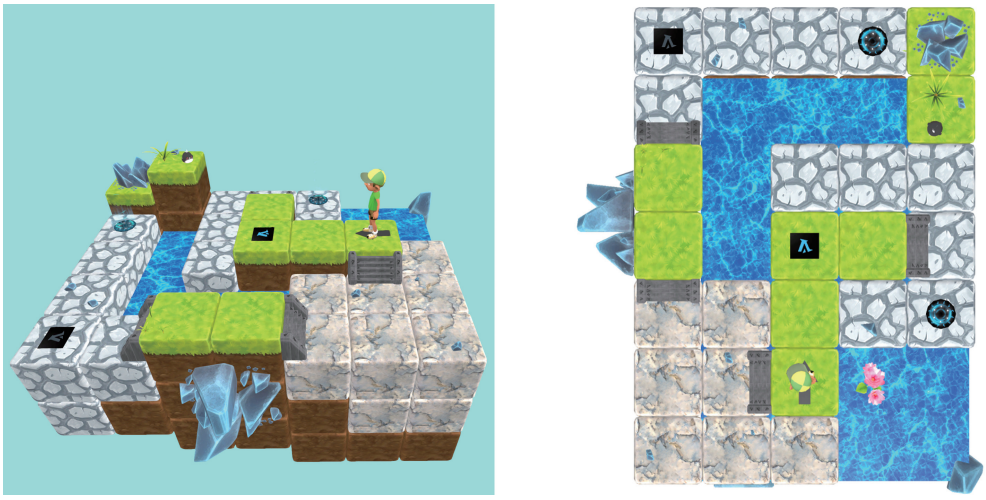

图 3.3 关卡 2-2 场景图

关卡的任务是打开 2 个开关，其中有 1 处开关已经打开，需要到达另一个没有打开的开关位置。主角的移动过程中需要多次右转，本关依旧不能使用右转命令 right()，这里最好的办法是自定义一个表示右转的函数，在需要右转的时候调用自定义的右转函数。

知识小课堂

C# 代码中的字符和括号都是英文字符，在代码编辑器中输入代码时，常常因为误输入中文字符导致代码编译错误。

代码

```
void TurnRight(){
    left();
    left();
    left();

}

move();
move();
TurnRight();
move();
move();
TurnRight();
move();
TurnRight();
move();
move();
move();
toggle();
```

首先定义右转 TurnRight() 函数，通过 3 个左转命令 left() 实现右转。在定义完右转函数后，一起来梳理主角的移动步骤。

第一步：向前移动 2 格；

第二步：向右转；

第三步：向前移动 2 格；

第四步：向右转；

第五步：向前移动 1 格到达传送门；

第六步：传送到另一个传送门后向右转；

第七步：向前移动 3 格到达开关的格子；

第八步：打开开关，完成任务。

关卡 2-2 通关路线图如图 3.4 所示。

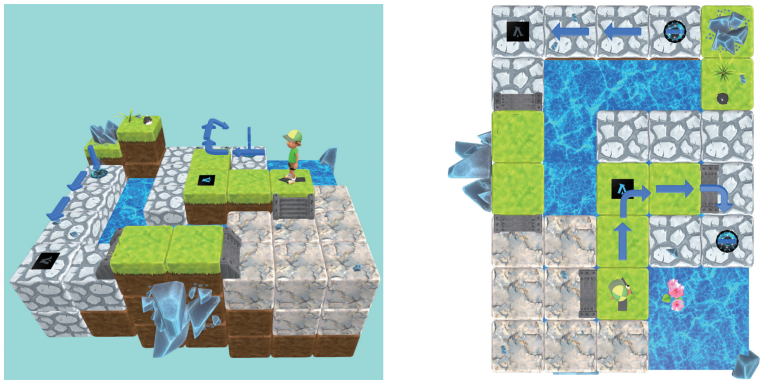

图 3.4　关卡 2-2 通关路线图

这里需要提醒一下，函数定义和函数调用的代码要一起输入代码编辑器中才可以运行。

3.3.3 重复模式

关卡说明

关卡编号：2-3

关卡难度：**

通关条件：南瓜 4 个，开关 4 个。

关卡目标：定义函数处理重复的模式。

关卡 2-3 场景图如图 3.5 所示。

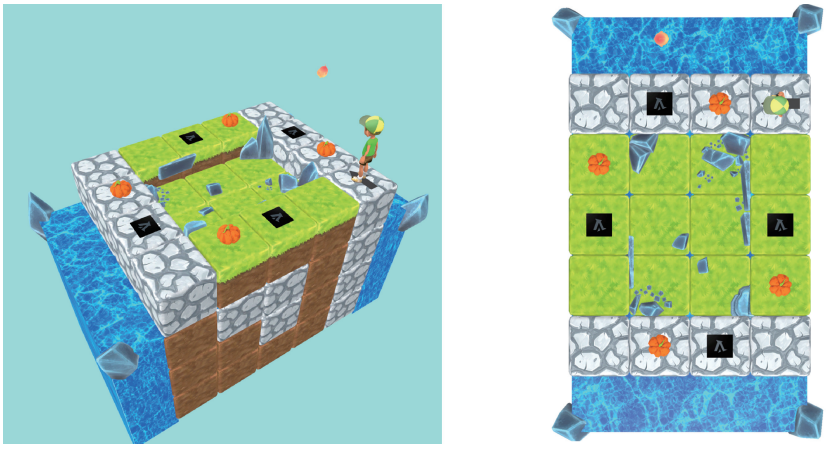

图 3.5　关卡 2-3 场景图

在本关中，每个南瓜旁边都有个开关。如果能把“收集南瓜”和“打开开关”的代码写在一个函数里面，就可以在适当的时候多次调用这个函数，避免重复编写代码。

知识小课堂

在以后的关卡中，如果发现有很多重复的模式（模式是指由多个基本命令按固定顺序组成，表达一个复杂的动作或命令），就可以将这个模式定义成函数，多次调用函数要比重复写相同的代码好得多。

代码

```
void TakeTwo(){

  take();
  move();
  toggle();
```

```
    move();
}

move();
TakeTwo();
left();
move();
TakeTwo();
move();
left();
move();
TakeTwo();
left();
move();
TakeTwo();
```

先定义 TakeTwo() 函数，这个函数可以先移动 2 个格子，并收集南瓜和打开开关。在 4 个边上收集南瓜和打开开关时，都可以调用 TakeTwo() 函数。

在定义完函数后，这里给出主角通关的步骤。

第一步：向前移动 1 格；

第二步：调用 TakeTwo() 函数；

第三步：向左转；

第四步：向前移动 1 格；

第五步：调用 TakeTwo() 函数；

第六步：向前移动 1 格并左转；

第七步：向前移动 1 格；

第八步：调用 TakeTwo() 函数；

第九步：向左转向；

第十步：向前移动 1 格；

第十一步：调用 TakeTwo() 函数，完成任务。

关卡 2-3 通关路线图如图 3.6 所示。

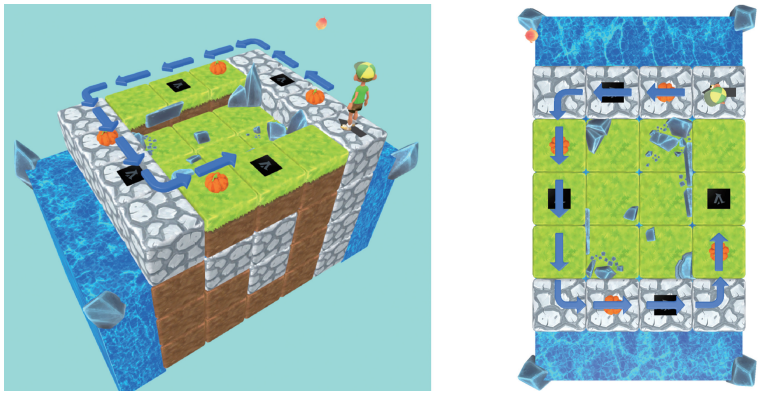

图 3.6　关卡 2-3 通关路线图

3.3.4　九个南瓜

关卡说明

关卡编号：2-4

关卡难度：**

通关条件：南瓜 9 个，开关 0 个。

关卡目标：识别重复的收集模式，并定义函数。

关卡 2-4 场景图如图 3.7 所示。

本关卡中，需要收集 9 个南瓜，这些南瓜分布有什么特点呢?

这些南瓜排列成 3 行 3 列，是不是可以将每列 3 个南瓜分为一组，然后将收集这一组 3 个南瓜的代码定义为一个函数。

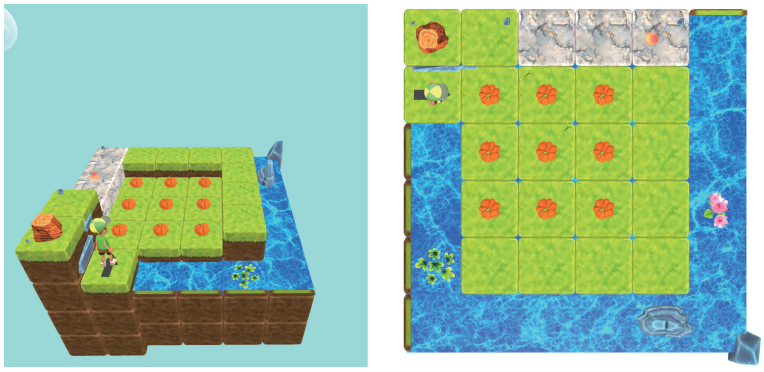

图 3.7 关卡 2-4 场景图

代码

```
void GetThree(){
  take();
  move();
  take();
  move();
  take();
}

move();
GetThree();
right();
move();
right();
GetThree();
left();
move();
left();
GetThree();
```

首先定义一个函数 GetThree()，用来收集一行上的 3 个南瓜，然后按照图 3.8 所示的运动方向逐行收集南瓜。

在定义完函数后，给出如下的核心步骤。

第一步：向前移动 1 格；

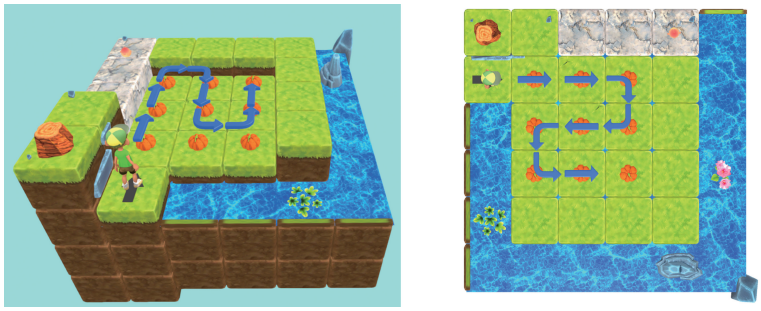

图 3.8　关卡 2-4 通关路线图

第二步：调用 GetThree() 函数；

第三步：向右转；

第四步：向前移动 1 格，并向右转；

第五步：调用 GetThree() 函数；

第六步：向左转；

第七步：向前移动 1 格，并向左转；

第八步：调用 GetThree() 函数，完成任务。

当然，大家也可以想出自己的方法，利用函数收集到所有的南瓜。但务必使用函数减少重复的代码。

3.3.5　嵌套调用

关卡说明

关卡编号：2-5

关卡难度：*

通关条件：南瓜 4 个，开关 0 个。

关卡目标：从一个函数调用另一个函数。

关卡 2-5 场景图如图 3.9 所示。

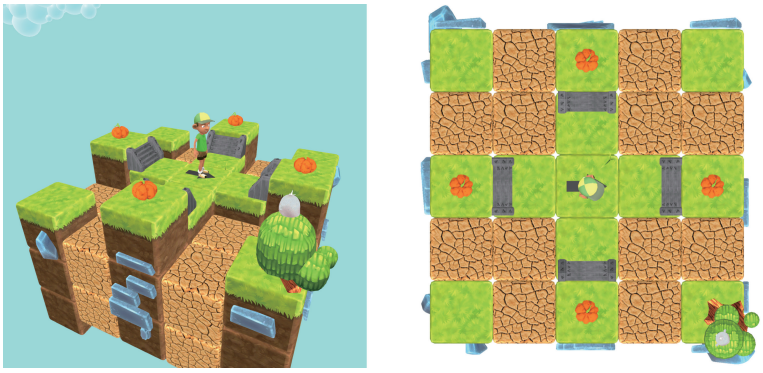

图 3.9　关卡 2-5 场景图

本关卡中，需要收集 4 个南瓜，这些南瓜分布有什么特点呢？

这些南瓜分布在不同的方向上，从主角位置到达每个南瓜的距离相同，在收集南瓜后还要转身回到起始位置，再去下一个南瓜位置收集。

代码

```
void TurnBack(){
  left();
  left();
}

void OneLine(){
  move();
  move();
  take();
  TurnBack();
```

```
    move();
    move();
}

OneLine();
OneLine();
right();
OneLine();
OneLine();
```

在每次收集南瓜后都要转身，将转身的动作定义成函数 TurnBack()。另外，从中心位置去收集南瓜然后回到中心位置，每个方向的这个过程都是相同的，因此，将这个过程定义成函数 OneLine()。

TurnBack() 函数调用 2 次左转命令 left() 实现转身的效果，大家可以操作一下转身。

OneLine() 函数，首先向前移动 2 格，在收集南瓜时调用 TurnBack() 函数转身，然后向前移动 2 格，回到起始位置。

主角要完成收集目标所需的动作如下：首先在第一个方向收集南瓜，然后回到起始点，再到对面方向收集南瓜，然后又回到起始点，先左转，进行第三个方向收集南瓜，回到起始点，再进行最后一个方向的南瓜收集。

关卡 2-5 通关路线图如图 3.10 所示。

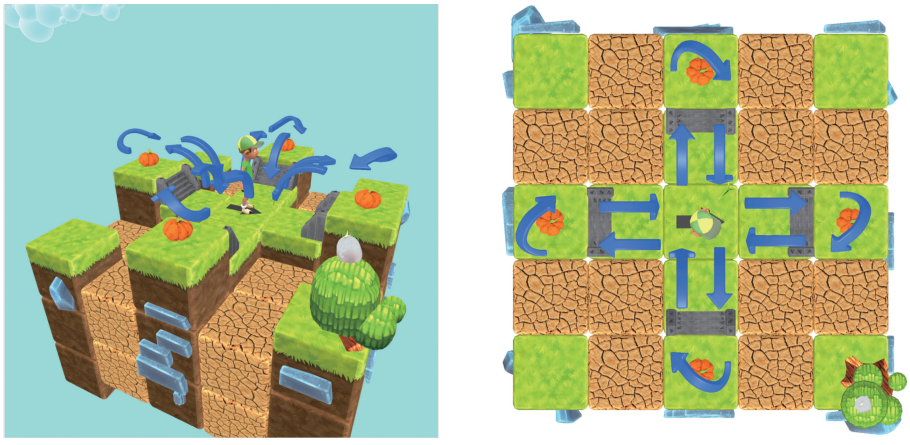

图 3.10　关卡 2-5 通关路线图

知识小课堂

通过函数把一个较大的问题分解成较小问题的方式，称为“问题分解”，这种方式有利于解决关卡难题，让代码更加简单易读。

3.3.6 高低阶梯

关卡说明

关卡编号：2-6

关卡难度：**

通关条件：南瓜 6 个，开关 0 个。

关卡目标：通过多个函数来分解问题。

关卡 2-6 场景图如图 3.11 所示。

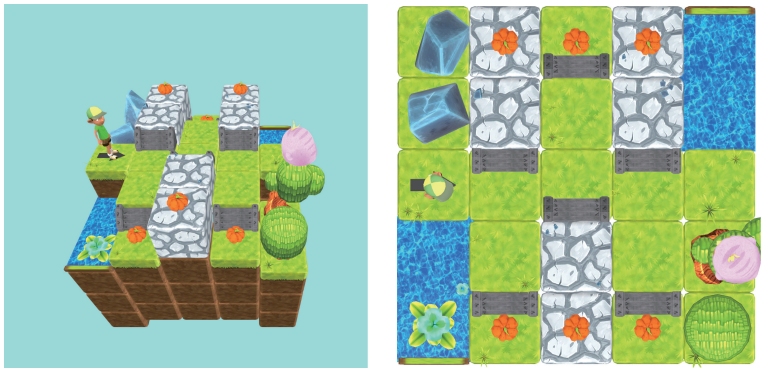

图 3.11 关卡 2-6 场景图

本关卡中，需要收集 6 个南瓜。这些南瓜分布有什么特点呢？读者可以思考一下！

接下来，一起找找南瓜分布有哪些相似的模式。如果以主角所站砖块画一个中心线，两边的南瓜分布是对称的，就是说从中心线到南瓜的距离是相同的。

知识小课堂

函数的命名小常识：在给函数命名时，可以基于函数的用途和目的给函数命名，尽量用有意义的单词组合，这样以后看到函数的时候，就可以知道这个函数的用途。

代码

```
void TakeTurnRound(){
   move();
   move();
   take();
   left();
   left();
   move();
   move();

}

void TakeOneLine(){
  left();
   TakeTurnRound();
   TakeTurnRound();
```