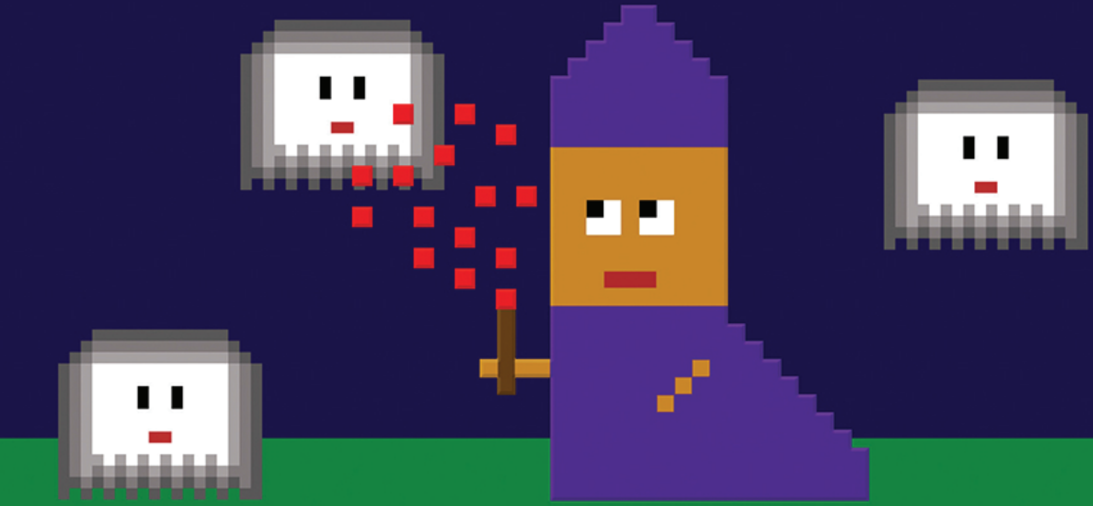
Sample Chapter

Hello Scratch!

Learn to Program by Making Arcade Games



Gabriel Ford, Sadie Ford, and Melissa Ford

MANNING

Hello Scratch!

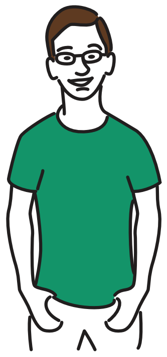
by Gabriel Ford, Sadie Ford, and Melissa Ford

Sample Chapter 7

Brief contents

7

Using conditionals to build your fixed shooter



You've probably figured out by now that the way people invent new video games is to look at old video games. If a video game is a big success, other game companies try to imitate it, making small changes to the visuals or goals of the game while keeping the same mechanics. That's what happened when Intellivision noticed the popularity of Atari's *Asteroids*.

Asteroids was released in 1979 and was a huge success. The player controlled a space cannon flying in outer space that could spin around and shoot at the rocks falling around them. Intellivision made its version—*Astrosmash*—in 1981, moving the space cannon to the ground, the colorful rocks to the sky, and adding a few extra enemies such as spinning bombs.

But don't think that Intellivision is just a copycat. *Asteroids* was based on the game *Spacewar!* And *Spacewar!* probably would have been based on something else if it wasn't one of the first video games. The game industry has a long history of creating similar games, which makes it easy for players to pick up a new game and immediately understand their goal and how to move their sprite.

Like *Astrosmash*, the goal of Wizards vs. Ghosts is to shoot the ghosts coming at you with sparks from your wand while trying not to get hit by any of the apparitions. The ghosts won't break apart like the rocks in

Asteroids or *Astrosmash* but instead disappear from the screen when blasted by the wizard's spell, as shown in figure 7.1.

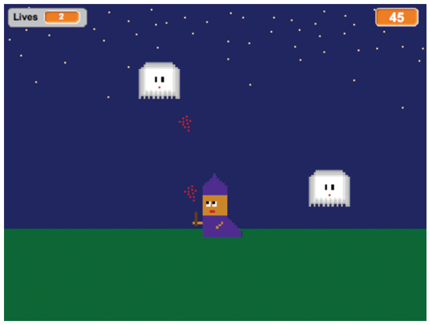


Figure 7.1 In Wizards vs. Ghosts, the player moves the wizard back and forth while blasting ghosts with his wand. The player loses a life whenever he touches a ghost, loses points whenever a ghost touches the ground, and gains points whenever he blasts one away.

Wizards vs. Ghosts is a fixed shooter, which means the sprite doing the blasting (in this case, the wizard) can only move back and forth on one fixed plane on the screen. You can't make the wizard fly above the ghosts, and you can't get him to hide in the grass. He can only move across the ground while fighting the ghosts. Additionally, the scenery is fixed and remains the same—a big, open field—throughout the whole game rather than changing as the player moves through a larger universe. The player is looking at the action from the side as if it is happening in front of them.

There are lots of types of shoot-em-ups, all of them named based on the way you're viewing the action or the abilities of the main sprite, such as side-scrolling shooters (games where the action makes the player keep flying in the same direction), rail shooters (where the game moves the sprite so you can concentrate on the single action of blasting away the enemy), and multidirectional shooters (where the main sprite can move in any direction).

You'll notice that all of these types of games include the word *shooter*. Violence plays a big role in video games, and it's common to have the player fighting against an enemy. Although games can be a safe way to explore something you would never do in real life (such as shoot at rocks!), you also don't have to make your game violent. The most creative fixed shooters move away from guns and cannons and use other methods for fighting the enemy. For instance, in this game you'll use a magic wand to blast your enemy, the ghosts.

In this chapter, you will learn

- How to use conditionals to check whether the player is pressing the spacebar
- How to use loops to keep sprites continuously moving
- How to use the same variable to reward and remove points

If you didn't get your letter from Hogwarts, here is your chance to not only get to be a wizard, but to rid the world of pesky ghosts like Peeves. Open up your Wizards vs. Ghosts project where you made your sprites in chapter 6 and get ready to code.

Preparing to program

You are steps away from being ready to write the code for this program, but there are a few small tasks you must complete before you're ready to go.

Missing sprites

If you skipped chapter 6, either flip back and complete it or go to the Manning site and download the background and sprites for Wizards vs. Ghosts. The directions for importing are the same as chapter 5. You should have a wizard, a ghost, a group of sparks, and a barrier line, as well as the nighttime background.

Preparing the Stage

You'll need to increase the size of your wizard and ghost. The wizard needs eight clicks with the Grow tool from the Grey Toolbar, and the ghost needs six clicks. The sparks and line are both the correct size.

If your wizard is not already standing on the grass, click and drag the wizard on the Stage to pull him onto the green third of the screen. You don't need to move any other sprite because they will all be coded into position.

Programming the wizard

The wizard is the equivalent to the space cannon in *Astrosmash*. You will move him back and forth along the grass using the arrow keys while you press the spacebar to fire spells from his wand. Three programs power the wizard: a movement script, a life deducting script, and a game ending script. All scripts in this section are applied to the wizard, so go put the blue box around the wizard sprite in the Sprite Zone and don't move the blue box until you program the ghost in the next section. Remember, the names or values on *your* blocks may be slightly different from time to time, so use the completed script images to make sure you choose the correct block.

Making a movement script

The wizard is standing in the grass, but he can't go anywhere. It's going to be hard to catch that falling ghost in figure 7.2 if he's unable to move.

You need to write a script that will give the player the ability to move the wizard left and right when the arrow keys are pressed.

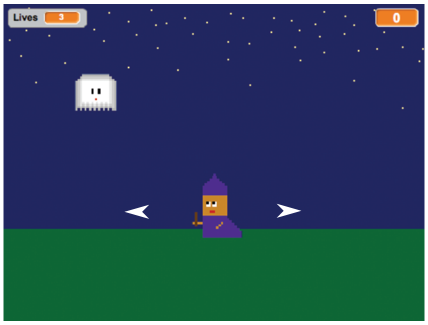


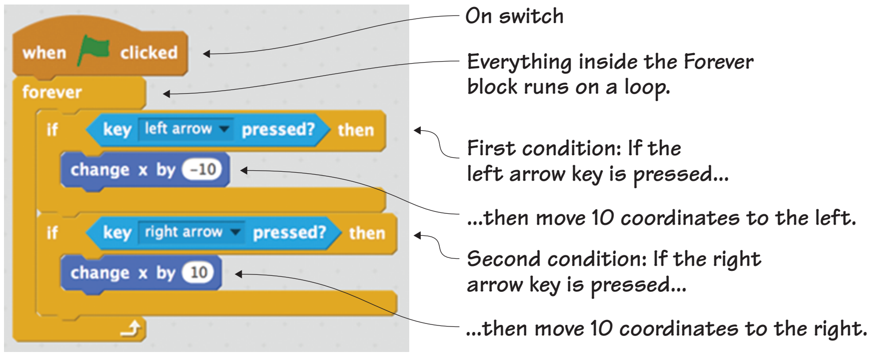
Figure 7.2 The wizard is currently unable to move beneath the ghosts in order to shoot sparks at them.

To give the wizard the ability to move

1. Start with a When Flag Clicked (Motion) block.
2. Snap a Forever (Control) block underneath to start a loop.
3. Move two If/Then (Control) blocks inside the Forever block. Stack them atop one another. You're going to set two conditions.
4. Slide a Key Space Pressed (Sensing) block into each of the empty, hexagonal spaces in the If/Then block. Change the top Key Space Pressed block to Left Arrow using the drop-down menu and the bottom Key Space Pressed block to Right Arrow. This sets what will happen if the left or right arrow keys are pressed on the keyboard.

5 Place a Change X by 10 (Motion) block inside each If/Then block. Change the number in the first Change X by 10 block to –10. This will move the wizard 10 coordinate spaces to the left. You don't need to make any changes to the second Change X by 10 block because positive numbers move toward the right.

The first script is now complete. Does your script match the one in figure 7.3?



```
when [green flag] clicked
forever
    if <key [left arrow] pressed?> then
        change x by -10
    if <key [right arrow] pressed?> then
        change x by 10
```

On switch
Everything inside the Forever block runs on a loop.
First condition: If the left arrow key is pressed...
...then move 10 coordinates to the left.
Second condition: If the right arrow key is pressed...
...then move 10 coordinates to the right.

Figure 7.3 The movement script allows the player to move the wizard left and right along the grass.

ANSWER THIS

WHY 10 COORDINATES?

Question: why have the wizard move ten spaces at a time instead of one space?

Answer: you know that negative numbers move to the left and positive numbers move to the right, but you also need to think about the speed at which you're setting the wizard to move. The lower the number, the slower the wizard will move across the Stage. The higher the number, the faster the wizard will move. Jumping eight coordinate spaces at a time ensures that the wizard doesn't miss a falling ghost by overshooting the space, nor does he inch forward at a glacial speed. Ten is good for this game, though you can experiment with other numbers at the end of the chapter.

You've set the wizard to jump 10 coordinates at a time. Remember, the higher the coordinate number, the faster the sprite will move.

Making a life deducting script

Currently, the ghosts can swoop straight through our wizard without anything happening. But the ghost is the wizard's sworn enemy! Shouldn't something happen if the ghost touches the wizard, as in figure 7.4?

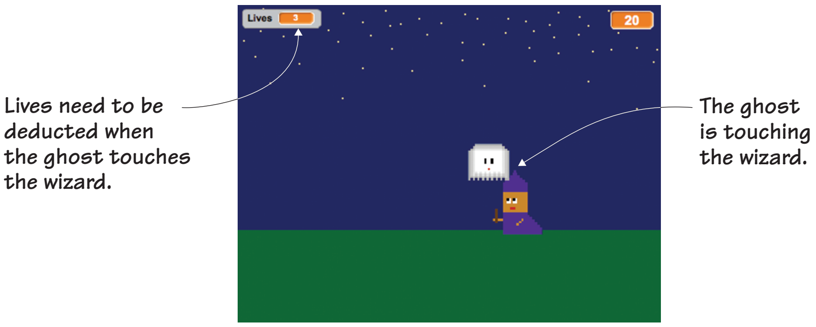

Lives 3
20
Lives need to be deducted when the ghost touches the wizard.
The ghost is touching the wizard.

Figure 7.4 The ghost is touching the wizard, but nothing is happening.

You need to write a script that will deduct a "life" every time a ghost touches the wizard. The player will start out the game with three lives, or turns.

To make this life deducting script

1. Start with a When Flag Clicked (Events) block.
2. Add a Forever (Control) block underneath to start a loop.
3. Slide an If/Then (Control) block inside the Forever block to set a condition.
4. Place a Touching (Sensing) block in the empty hexagonal space on the If/Then block. Use the drop-down menu to set the sprite to Ghost. You've now set a condition: if the wizard is touching the ghost sprite.
5. Go to Data and create a variable called Lives. Keep it on the default For All Sprites setting. Position the variable in the top left corner of the Stage.
6. Drag a Change Lives by 1 (Data) block inside the If/Then block. Change the 1 to a –1 because you want to deduct, not add, a life.

You've now set the action that will happen if the condition is met: change the value of the variable Lives by –1.

7 Slip a Wait 1 Secs (Control) block under the Change Lives by –1 block. Change the 1 to a 2, which will give the game two seconds to get rid of the ghost so the same ghost doesn't glitch and remove multiple lives at a time.

You now have the script, shown in figure 7.5, that will remove one of the player's three chances whenever a ghost sprite touches the wizard.

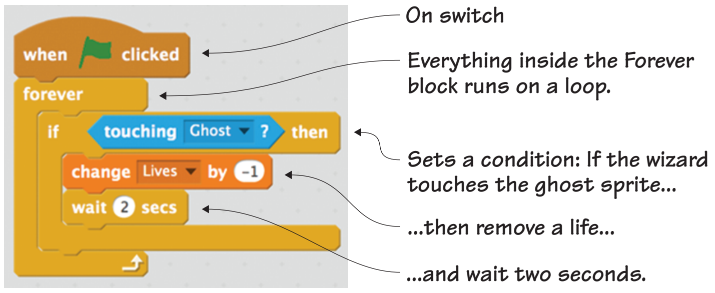

```
when [flag] clicked
forever
  if touching Ghost ? then
    change Lives by -1
    wait 2 secs
```
On switch
Everything inside the Forever block runs on a loop.
Sets a condition: If the wizard touches the ghost sprite...
...then remove a life...
...and wait two seconds.

Figure 7.5 The completed life deducting script removes one of the player's three chances when a ghost touches the wizard.

Making a game ending script

This is the final script for the wizard, and it solves a big problem: the game currently doesn't have an end point. Sure, you're planning on giving the player three chances to get the ghosts, but currently, they can keep going long after they've used up their three chances, like the scoreboards in figure 7.6.

This script will check how many lives the player has left, and once it is less than 1, it will stop the game.

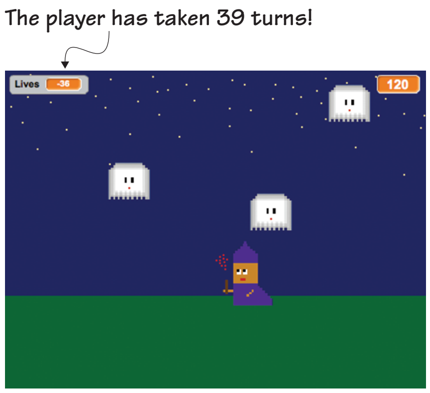


Figure 7.6 The game can go on forever unless you set an end point.

To make the game ending script

1. Start with a When Flag Clicked (Events) block.
2. Add a Forever (Control) block underneath to start a loop.
3. Slide an If/Then (Control) block inside the Forever block to set a condition.
4. Place a Square < Square (Operators) block inside the empty hexagonal space in the If/Then block. Fill the left square with a Lives (Data) block and type a 1 in the right square. You've set a condition: if the value of the variable Lives is less than 1, then do *something*. You'll define that "something" with the next step.
5. Slide a Stop All (Control) block inside the If/Then block. That is the action that will happen if the condition is met: all the scripts will stop, and the game will end.

The game ending script shown in figure 7.7 is now complete. Check your work against the image because this is the final script for the wizard sprite.

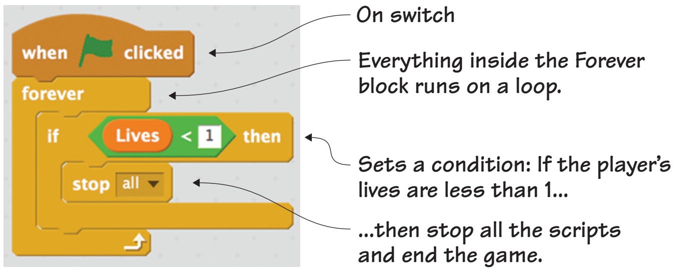

when clicked
forever
if Lives < 1 then
stop all
On switch
Everything inside the Forever block runs on a loop.
Sets a condition: If the player's lives are less than 1...
...then stop all the scripts and end the game.

Figure 7.7 The completed game ending script tells Scratch when to end the game of Wizards vs. Ghosts.

There are three scripts for the wizard sprite and three loops. Wizards vs. Ghosts is a fast-paced game, and loops enable a set of actions to run indefinitely with the click of a single on switch—in this case, the When Flag Clicked block.

The ghost will have its own set of loops because you'll want the ghosts to continuously generate and fall during the game.

Programming the ghosts

Once again, you only have one ghost, but I keep using the plural: ghosts. This is your clue that you're about to make many copies of the ghost, by cloning them with code. The ghosts are the equivalent to the asteroids that fall during *Astrosmash*, and they require three scripts: a positioning script, a cloning script, and a movement script. Put the blue box around the ghost in the Sprite Zone and leave it there for all three scripts.

Making a positioning script

Where do ghosts hang out when they're not swooping down and haunting the wizard? In this game, they live in the sky, so they're in the perfect position to descend on our wizard, as you can see in figure 7.8.

This script will make the ghost sprites generate at a random point at the top of the Stage.

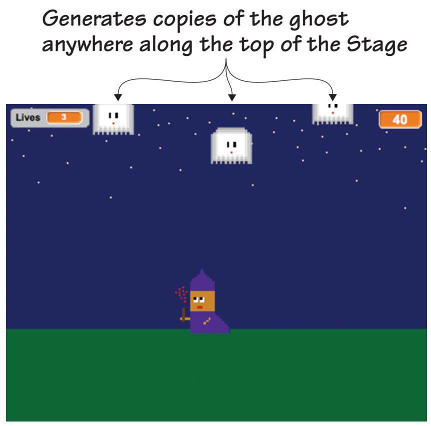

Generates copies of the ghost anywhere along the top of the Stage
Lives 3
40

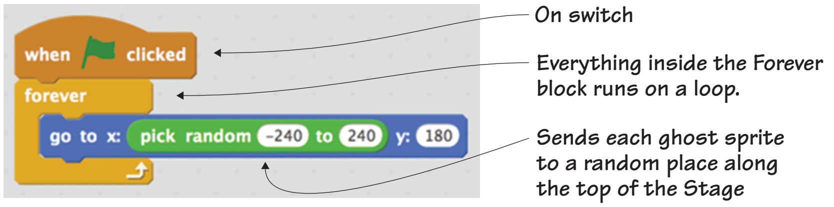
Figure 7.8 Watch out, wizard! The ghosts are coming down from the top of the Stage!

To create the positioning script

1. Start with a When Flag Clicked (Events) block.
2. Add a Forever (Control) block underneath to start a loop.
3. Place a Go to X/Y (Motion) block inside the Forever block. This will set the coordinate (or point on the screen) where the ghost will go. But you want the ghosts to pop out from different places at the top of the Stage, not the same place, so you're going to slip a block into the X slot.
4. Put a Pick Random 1 to 10 (Operators) block inside the first circle for the X coordinate, as in figure 7.9. This will allow you to set a range of coordinates and have Scratch choose a different one each time a ghost forms. Change the two numbers in the Pick Random block to –240 and 240 to indicate the range of coordinates along the

top of the Stage. Type 180 in the Y coordinate slot because you want all the ghosts to generate from the top of the Stage and not random spots along the Y-axis.

You now have your first script for your ghosts and yet another script using a loop. This time, the loop in figure 7.9 is causing Scratch to choose a new spot for each ghost sprite to use as a starting point at the top of the Stage.



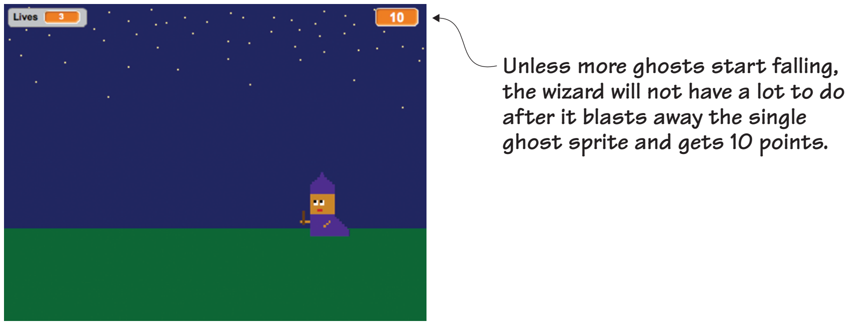
```
when [flag] clicked
forever
    go to x: (pick random -240 to 240) y: 180
```

On switch
Everything inside the Forever block runs on a loop.
Sends each ghost sprite to a random place along the top of the Stage

Figure 7.9 The completed positioning script sets a random starting point for each ghost sprite.

Making a cloning script

You currently only have one ghost. It's going to be a brief game unless you make some more. Right now, once the wizard blasts away that one ghost (as seen in figure 7.10), he's not going to have a lot to do.



Lives 3
10
Unless more ghosts start falling, the wizard will not have a lot to do after it blasts away the single ghost sprite and gets 10 points.

Figure 7.10 The wizard scores 10 points for shooting down the ghost and then stands in the field indefinitely with nothing more to do.

You can make the game a lot more exciting by cloning the ghost and making the enemy constantly regenerate new copies during the game. This script will make copies of the ghost sprite, one every second.

To make this cloning script

1. Start with a When Flag Clicked (Events) block.
2. Slide a Hide (Looks) block underneath the When Flag Clicked block. This block hides the ghost while it generates so the player can't predict where the ghost will start falling.
3. Add a Forever (Control) block underneath to start a loop.
4. Place a Create Clone of Myself (Control) block and a Wait 1 Secs (Control) block inside the Forever block, one on top of the other. The task happens on a continuous loop while the game is in action, creating a clone and then pausing for a second.

The brief script in figure 7.11 gives the wizard plenty of ghosts to fight during the game.

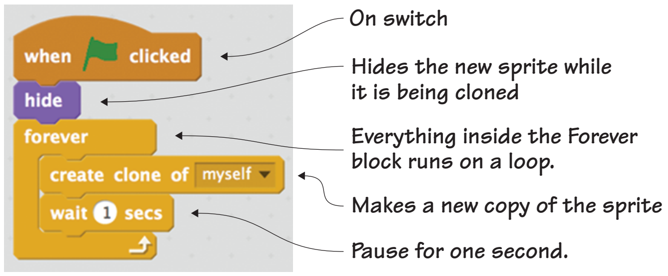

when clicked
hide
forever
create clone of myself
wait 1 secs
On switch
Hides the new sprite while it is being cloned
Everything inside the Forever block runs on a loop.
Makes a new copy of the sprite
Pause for one second.

Figure 7.11 The completed cloning script creates new copies of the ghost sprite.

ANSWER THIS **WHAT DOES THE ONE SECOND DELAY DO?**

Question: the script ends with a Wait 1 Secs block, but do you need it?

Answer: let's put it this way: would you rather fight an army of ghosts that are coming at you one per second, or would you rather deal with a million ghosts at once? Be kind to the little wizard and give him a fighting chance against the ghosts by staggering them one per second. Without that delay, the ghost copies will start generating one on top of the other and falling in clumps. You can change the length of the delay by typing a new number in the bubble. Changing the 1 to a 2 would make the game a little easier for young players. Changing the 1 to a .5 would make the game more intense.

Making a movement script

You've made a lot of ghosts, and they're positioned at the top of the Stage, but right now they're only hiding. Our little wizard is waiting nervously at the ready, as you can see in figure 7.12.

This movement script will do double duty, not only sending the ghosts down from the top of the Stage but setting up a scoring system, too.

This is the final script for the ghost, and it's a long one. Check your script against the one in the book every few steps. This script uses two variables and a complicated loop to keep track of where the ghost is on the screen and whether it is touching the wizard or the sparks:

The ghosts are hidden at the top of the Stage. You need to make them visible and start moving.

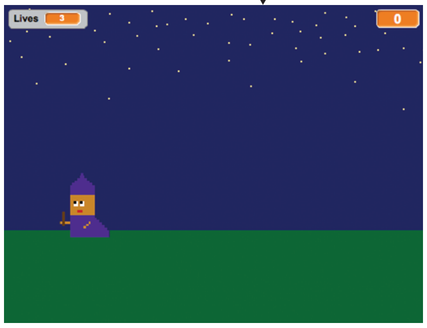


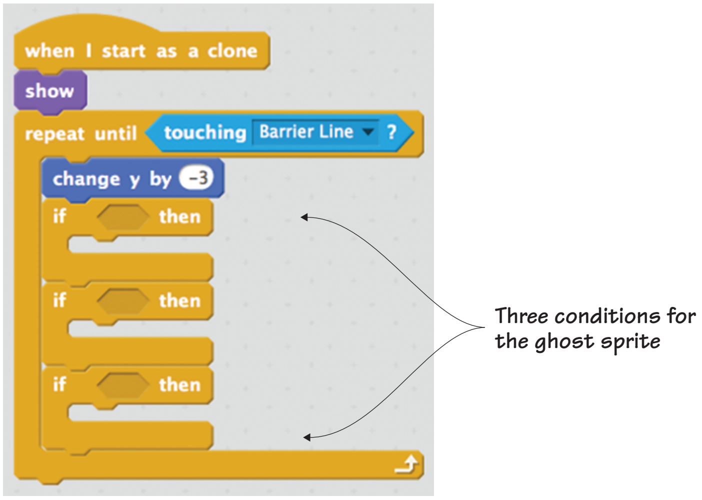
Figure 7.12 The wizard is ready, but the ghosts have no way to move.

1 Start with a When I Start as a Clone (Control) block. I bet you thought I was going to send you to the Events menu, but this script only kicks into action if the ghost clone has been made by the last script.
2 Snap a Show (Looks) block under the When I Start as a Clone block. This will make the clone visible.
3 Place a Repeat Until (Control) block under the Show block. This is a new kind of loop. You're going to have an action repeat until a condition is met.

4 Drag a Touching (Sensing) block into the empty space on the Repeat Until block. Use the drop-down menu to set it to Barrier Line. This means that everything you put inside the Repeat Until block will keep happening until the ghost sprite reaches the bottom of the screen and therefore touches the barrier line.

5 Slide a Change Y by 10 (Motion) block and place it inside the Repeat Until block. Change the 10 to –3. Any negative number will make it move down the Stage, and –3 is a good speed for the ghost. You can play with this number once your game is done. Low numbers, such as –1, will make the ghost fall slower, and higher numbers, such as –5, will make the ghost fall faster.

6 Stack three If/Then (Control) blocks underneath the Change Y by –3 block, as seen in figure 7.13. You're going to set up three conditions.

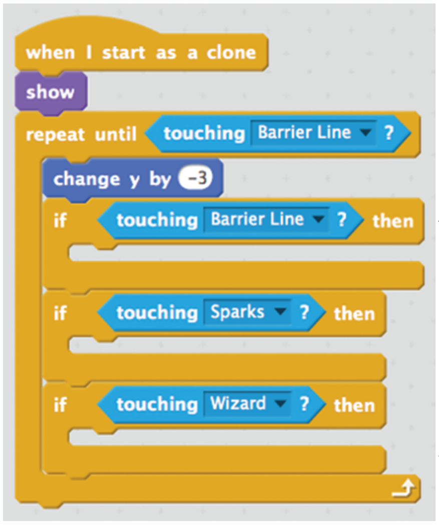


```
when I start as a clone
show
repeat until <touching [Barrier Line] ?>
    change y by -3
    if <> then
    if <> then
    if <> then
```

Figure 7.13 Stacking three If/Then blocks inside the Repeat Until block allows the game maker to set three different conditions.

7 Grab three Touching (Sensing) blocks and place one each inside the empty hexagonal space in the If/Then blocks. Use the drop-down

menu to change the first one to Touching Barrier Line, the second one to Touching Sparks, and the third one to Touching Wizard, as shown in figure 7.14.



```
when I start as a clone
show
repeat until <touching [Barrier Line] ?>
    change y by (-3)
    if <touching [Barrier Line] ?> then
    
    if <touching [Sparks] ?> then
    
    if <touching [Wizard] ?> then
    
```

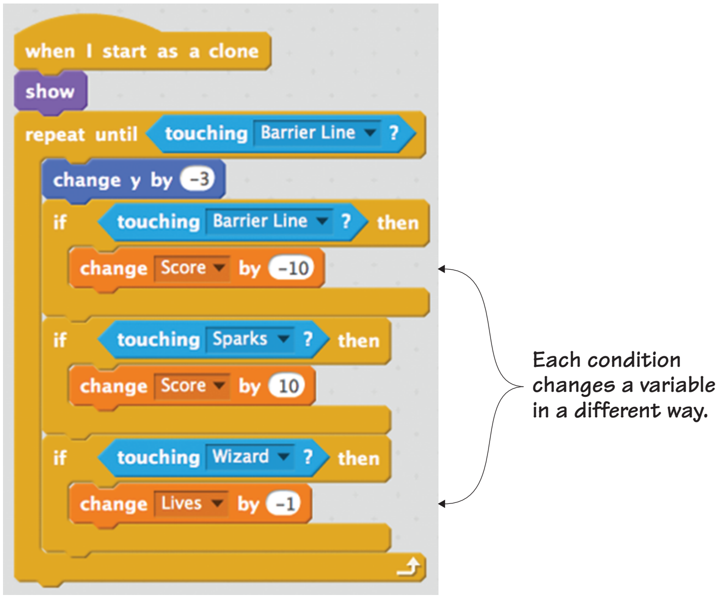
Different actions will happen depending upon whether the ghost is touching the barrier line, sparks, or wizard.

Figure 7.14 The loop contains three different conditions: if the ghost is touching the barrier line, sparks, or wizard.

8 Click Data and make a variable called Score. Position the variable in the top right corner of the Stage. Right-click (or control-click, if you're using a Mac) the score bubble to bring up the menu. Choose Large Readout so the score appears as a number on the screen.

9 Drag three Change Score by 1 (Data) blocks and place one in each of the If/Then blocks. Keep the variable set to Score in the first block, but change the value to –10. If the ghost reaches the barrier line without being shot down by the sparks, it will deduct 10 points from the player's score. Keep the variable set to Score in the second block, but change the value to 10. If the sparks make contact with the ghost, it will add 10 points to the player's score. Change the variable to Lives in the third block and change the value to –1. If the

ghost touches the wizard, it will remove a life. You can see all these variables and their values in figure 7.15.



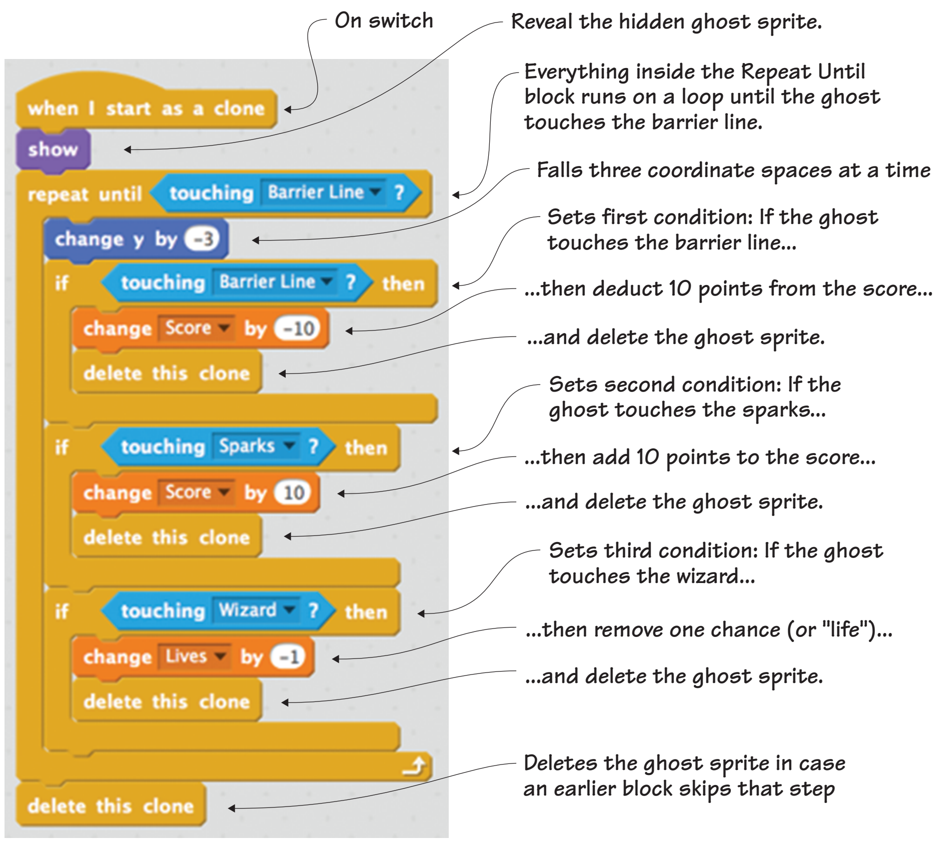
```
when I start as a clone
show
repeat until <touching [Barrier Line] ?>
    change y by (-3)
    if <touching [Barrier Line] ?> then
        change [Score] by (-10)
    if <touching [Sparks] ?> then
        change [Score] by (10)
    if <touching [Wizard] ?> then
        change [Lives] by (-1)
```

Figure 7.15 Each condition changes the variable and its value.

10 Slide a Delete This Clone (Control) block underneath each of the Change Score by –10 blocks (or whatever the variable and value is on the block). This will delete the copy of the ghost after it touches the barrier line, sparks, or wizard.

11 Place a fourth Delete This Clone (Control) sprite underneath the Repeat Until Touching Barrier Line block at the very bottom of the script. Just in case the If/Then inside the Repeat Until block misses deleting the sprite, this backup block will get rid of the ghost so you don't have copies lingering at the bottom of the screen.

You can see in figure 7.16 that it's a long script, but it gives the ghosts the ability to move and it sets up a scoring system.



```
when I start as a clone
show
repeat until <touching [Barrier Line ▾] ?>
    change y by (-3)
    if <touching [Barrier Line ▾] ?> then
        change [Score ▾] by (-10)
        delete this clone
    if <touching [Sparks ▾] ?> then
        change [Score ▾] by (10)
        delete this clone
    if <touching [Wizard ▾] ?> then
        change [Lives ▾] by (-1)
        delete this clone
delete this clone
```

On switch
Reveal the hidden ghost sprite.
Everything inside the Repeat Until block runs on a loop until the ghost touches the barrier line.
Falls three coordinate spaces at a time
Sets first condition: If the ghost touches the barrier line...
...then deduct 10 points from the score...
...and delete the ghost sprite.
Sets second condition: If the ghost touches the sparks...
...then add 10 points to the score...
...and delete the ghost sprite.
Sets third condition: If the ghost touches the wizard...
...then remove one chance (or "life")...
...and delete the ghost sprite.
Deletes the ghost sprite in case an earlier block skips that step

Figure 7.16 *The completed movement script not only allows the ghosts to fall but also sets up a scoring system.*

The Repeat Until is another type of loop. It's one that comes with a condition rather than being open-ended.

ANSWER THIS **HOW DO YOU DECIDE THE NUMBERS IN A SCORING SYSTEM?**

Question: each game sets a guideline for how points are gained or lost, but how do you decide how many points to reward or remove?

Answer: this is a decision in the hands of the game maker, and there are no right or wrong answers. In fact, feel free to change the values in the Change Score blocks. Maybe you want to deduct only 10 points if the player misses the ghost but reward 20 points if the player gets the ghost. Or you can take the opposite path, deducting more points than you reward. Think about whether you want to make it easy, medium, or hard to rack up points.

The ghost is now complete. It's time to look at the sparks and set up scripts that will allow them to shoot from the wizard's wand.

Programming the sparks

The wizard is always holding his wand, but nothing comes out of it until he casts a spell. You need to code this sprite so the sparks will shoot up toward the ghosts when the player presses the spacebar on the keyboard. Once again, you'll use cloning to make unlimited sparks for your wizard to use. The sparks for the wand use four scripts: a positioning script, a cloning script, a movement script, and a clone deletion script. Place the blue box around the red sparks in the Sprite Zone.

Making a positioning script

The wizard can move back and forth, so he can dodge out of the way of the falling ghosts. But right now, he can't cast a spell and get *rid* of the ghosts. The wand in figure 7.17 is dormant—only a little stick of wood.

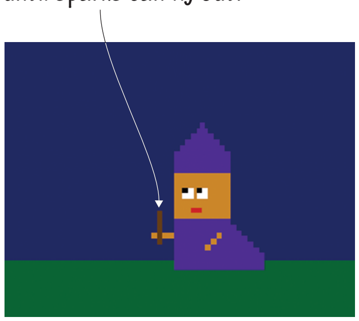


Figure 7.17 The wand is part of the wizard sprite, but the sparks operate with their own scripts.

This script will send the sparks to the wand. The wand is drawn as part of the wizard sprite, whereas the sparks are a separate sprite. This script brings the two sprites together.

To create the positioning script

1 Start with a When Flag Clicked (Events) block.
2 Slide a Hide (Looks) block under the When Flag Clicked block. This will keep the sparks invisible until the player is ready to use them.
3 Add a Forever (Control) block underneath to start a loop.

4 Drag a Go to Mouse Pointer (Motion) block inside the Forever block. You don't want the sparks to go to the mouse pointer. You want them to go to the wizard, so open the drop-down menu and choose the Wizard option.

Figure 7.18 shows the whole script. It's a small but important script that gives the wizard a lot of power in his fight against the ghosts.

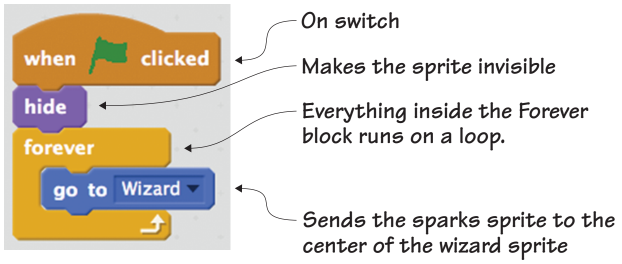

```
when [flag] clicked
hide
forever
    go to Wizard
```
On switch
Makes the sprite invisible
Everything inside the Forever block runs on a loop.
Sends the sparks sprite to the center of the wizard sprite

Figure 7.18 The completed positioning script sends the sparks to the wizard's wand.

FIX IT MY SPARKS ARE COMING FROM THE WIZARD'S HEAD! When you sent the sparks to the wizard, you sent them to the center of the sprite, which could be anywhere on the wizard and not necessarily where you want them, such as coming from the wand. Remember how the sprite is the size of that canvas, with some parts in color and other parts transparent? The center of the canvas is the center of the sprite. Knowing that, you can draw your sprite so it's near that little grey plus sign always seen in the center of the Art Editor (figure 7.19).

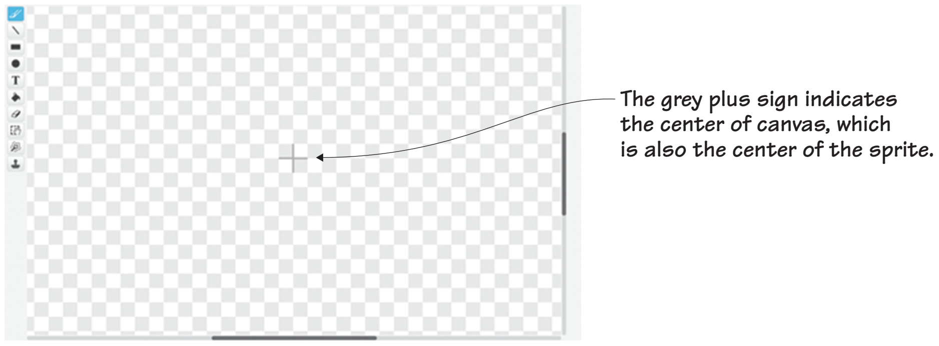

The grey plus sign indicates the center of canvas, which is also the center of the sprite.

Figure 7.19 The light grey plus sign marks the center of both the canvas and the sprite.

But don't worry if you drew your sprite somewhere else on the canvas. You can always center it. Go to the Sprite Zone and click the wizard. Navigate to the tab marked Costumes in the middle of the Block Menu. Look at the plus sign in the top right corner of the Art Editor, marked in figure 7.20. Click it, and a cross should appear on the canvas.

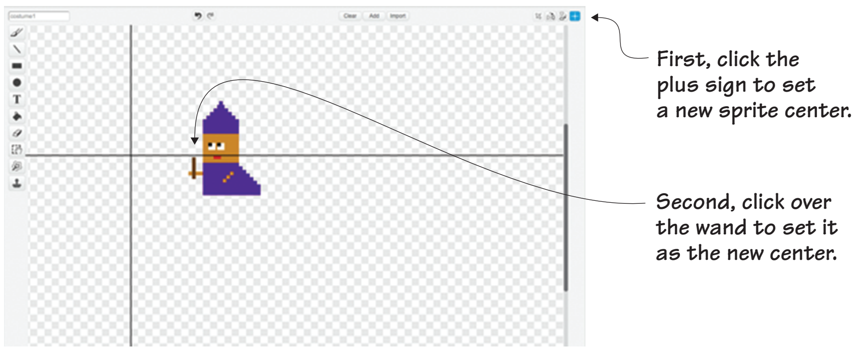

First, click the plus sign to set a new sprite center.
Second, click over the wand to set it as the new center.

Figure 7.20 First click the plus sign in the top right corner, and then click a point on the screen to set a new center.

Zoom in using the magnifying glass to enlarge the wizard if you have trouble seeing his wand. Now click the top of the wand (also marked in figure 7.20) to set that space as the new center. You should see the grey plus sign on the canvas over the wand, as in figure 7.21. Now the sparks will come out of the wand and not the wizard's head.

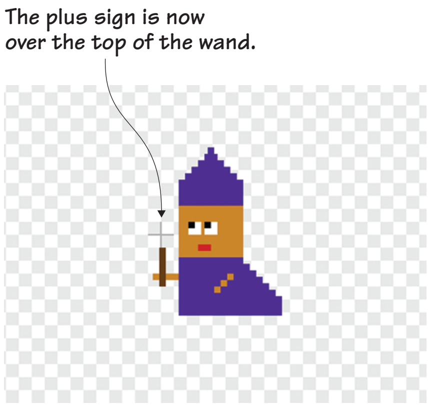


Figure 7.21 The center of the sprite is now the space over the wand, which is where the sparks will be sent from once you clone the sprite.

Making a cloning script

Right now, you have one spark. But the wizard is going to need an unlimited number of sparks in order to fight the ghosts so the sparks can continuously shoot from the wand, as in figure 7.22.

This script will clone the sparks that will come out of the wand. It will make a new copy of the sparks every time the spacebar is pressed. This is the only script so far that doesn't contain a loop.

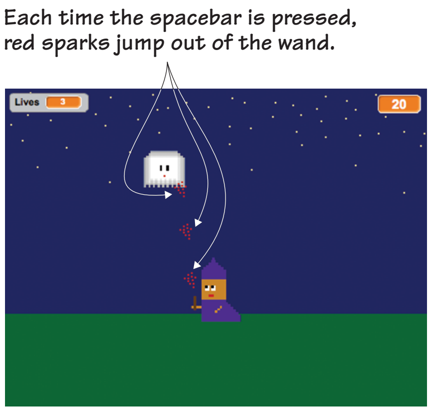

Each time the spacebar is pressed, red sparks jump out of the wand.
Lives 3
20

Figure 7.22 The wand emits sparks every time the spacebar is pressed.

If you checked the center of the wizard sprite with the last Fix It, make sure you go back to the Sprite Zone and put the blue box around the sparks. You're still programming the sparks sprite. Click the Scripts tab to continue programming.

To create the cloning script

1. Start with a When Space Key Pressed (Events) block. This is a different type of on switch, and it runs the next part of the script whenever the player presses the spacebar on the keyboard.
2. Slide a Create Clone of Myself (Control) block under the When Space Key Pressed block. This creates a clone of the sparks every time the player presses the spacebar.

And that's it! The whole script in figure 7.23 is only two blocks long. One block sets the starting point (pressing the spacebar) and the other block clones the sprite.

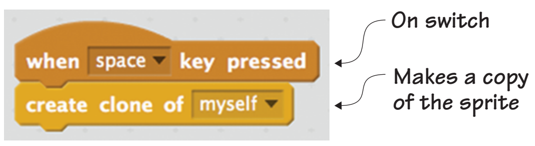

```
when space key pressed
create clone of myself
```

Figure 7.23 The completed cloning script is brief but is an important part of the game.

Making a movement script

Right now the sparks are ready to come out of the wand, but the sprite has no instructions to move. In figure 7.24, the sparks are piling up on the tip of the wand, unable to shoot toward the ghosts above.

This script will make the sparks that you clone shoot upward, out of the wand. To make the movement script

1 Start with a When I Start as a Clone (Control) block.
2 Snap a Show (Looks) block under the When I Start as a Clone block. This will make the clone visible.
3 Place a Forever (Control) block under the Show block. You're starting a loop that will put the sparks into continuous motion until a condition is met.
4 Drag a Repeat Until (Control) block inside the Forever block. It's a double loop! The outer loop (Forever block) controls how many times the inner loop (Repeat Until block) runs through its actions. The inner loop will run, and then the outer loop will check where things are with the script and run again. Putting a loop inside a loop is called a *nested loop*. In this case, the inner loop sets up a situation for the individual copy of the sparks, and then the outer loop applies those tasks to all the copies made of the sparks.
5 Place a Hexagon or Hexagon (Operators) block inside the empty hexagonal space on the Repeat Until block.
6 Put a Touching (Sensing) block in each of the empty hexagons in the green Operators block. Using the drop-down menu, choose Ghost for the first block and choose Edge (as seen in figure 7.25) for the second block. This sets a limit for the loop: repeat the actions inside until the sparks are touching either the ghost or an edge. In this case, the edge is always going to be the one at the top of the Stage because the sparks go straight up.

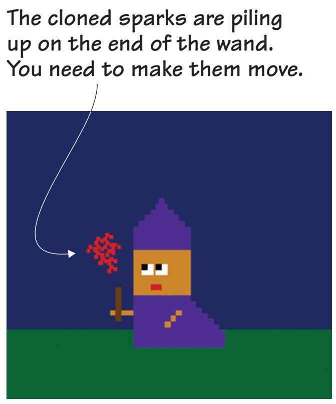

The cloned sparks are piling up on the end of the wand. You need to make them move.

Figure 7.24 New copies of the sparks are being cloned, but they have nowhere to go.

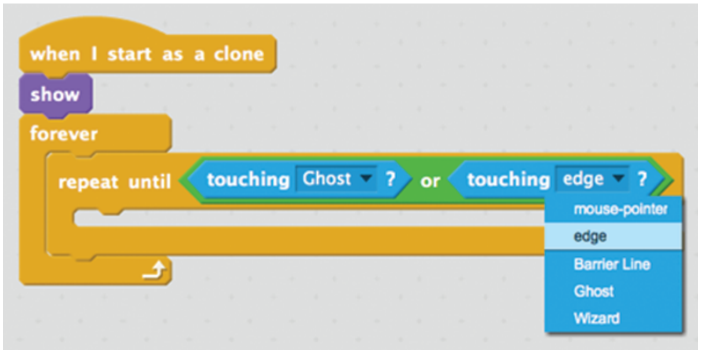


Figure 7.25 Use the drop-down menu to set the two limits for the Repeat Until block.

7 Place a Change Y by 10 (Motion) block inside the Repeat Until block. You can keep it as 10 because this will make the sparks slightly faster than the ghosts. Because it's a positive number, the sparks will go up.

8 Slide an If/Then (Control) block under the Change Y by 10 block, as in figure 7.26. You're about to start setting a new condition that will apply to the individual sparks.

9 Put a Touching (Sensing) block in the empty hexagonal space in the If/Then block. Use the drop-down menu to set the option to Ghost.

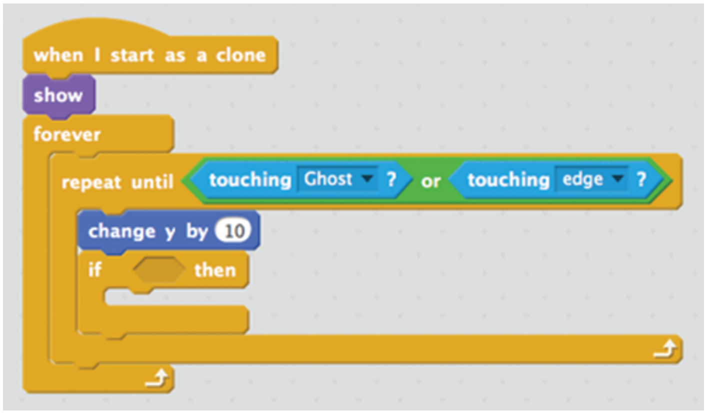


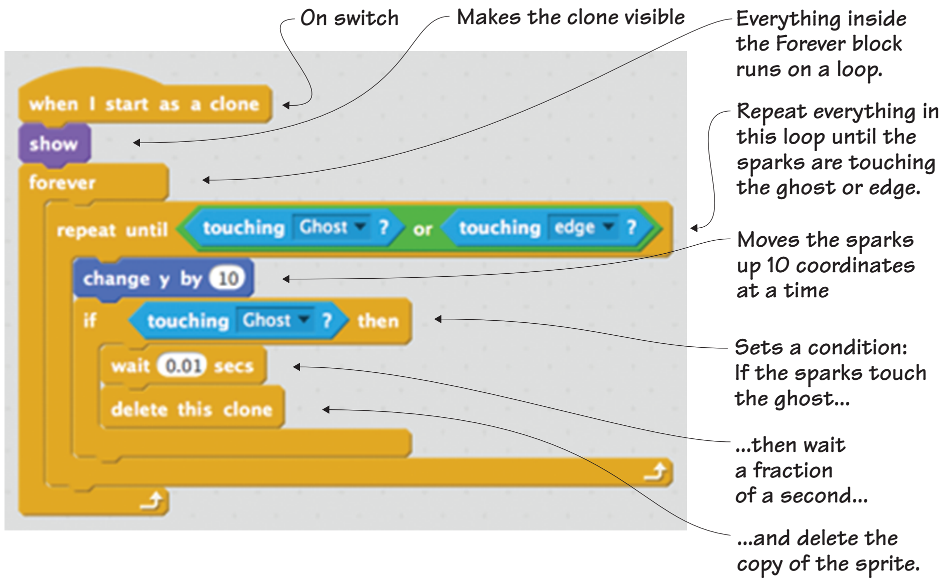
Figure 7.26 The If/Then block goes under the Change Y by 10 block.

This condition looks at what happens if the sparks make their target and hit a ghost.

10 Snap a Wait 1 Secs (Control) block inside the If/Then block. A second is a long time, so change that number to 0.01. You only need a fraction of a second pause to ensure that Scratch has enough time to recognize that the sparks have touched the ghost.

11 Slide a Delete This Clone (Control) block underneath the Wait 0.01 Secs block. If the sparks make contact with the ghost, you want the sparks to disappear from the screen.

The movement script shown in figure 7.27 sends the sparks shooting upward out of the wand and sets up what happens next if the sparks hit a ghost.



```
when I start as a clone
show
forever
    repeat until <<touching Ghost ?> or <touching edge ?>>
        change y by 10
        if <touching Ghost ?> then
            wait 0.01 secs
            delete this clone
```

On switch
Makes the clone visible
Everything inside the Forever block runs on a loop.
Repeat everything in this loop until the sparks are touching the ghost or edge.
Moves the sparks up 10 coordinates at a time
Sets a condition: If the sparks touch the ghost...
...then wait a fraction of a second...
...and delete the copy of the sprite.

Figure 7.27 The completed movement script contains a nested loop—a loop inside a loop.

Making a clone deletion script

You set up a way for the sparks to disappear from the screen if they hit the ghost, but what about all the sparks that miss the ghost? After a few minutes of playing, the top of your Stage will be a sparks graveyard, as in figure 7.28.

This script will make the sparks disappear when they hit the top of the Stage. This task has been separated out from the last script to give you space to create two different situations when the sparks hit the ghost or the top of the screen.

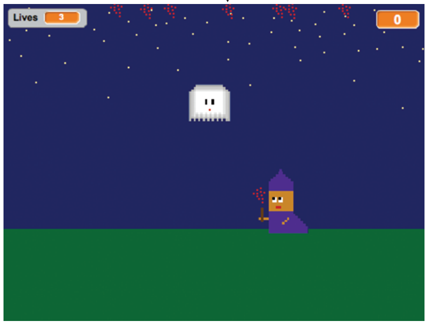
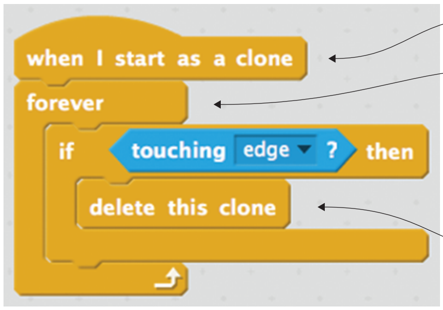

The sparks need to delete when they reach the top of the Stage.
Lives 3
0

Figure 7.28 The sparks are collecting at the top of the Stage.

To make the clone deletion script

1 Start with a When I Start as a Clone (Control) block.
2 Add a Forever (Control) block underneath to start a loop.
3 Place an If/Then (Control) block inside the Forever block to set a condition.
4 Put a Touching (Sensing) block inside the empty hexagonal space in the If/Then block and use the drop-down menu to change the option to Edge. This will detect any edge, though the sparks will only come in contact with the top of the Stage because they move upward.
5 Drag a Delete This Clone (Control) block inside the If/Then block. This will remove the copy of the sparks from the screen.

Figure 7.29 shows the final script for the sparks. Does your script match the one in the figure?



```
when I start as a clone
forever
    if touching edge ? then
        delete this clone
```

On switch
Everything inside the Forever block runs on a loop.
Sets a condition: If the sparks are touching the top of the screen...
...then delete the sparks.

Figure 7.29 The completed clone deletion script detects the edge of the Stage.

Programming the odds and ends

You have two more scripts to write, both of them brief. One applies to the line, and the other script applies to the background. Yes, even the background is programmable in Scratch.

Making a positioning script for the line

Although there is a bottom edge on the Stage, marked by a Y coordinate number of –180, you're going to use a boundary line to ensure that Scratch always knows when the ghosts have reached the bottom of the screen. Right now your line is probably somewhere random on the screen, as in figure 7.30.

This script will position the line at the bottom of the Stage. Place the blue box around the line sprite in the Sprite Zone.

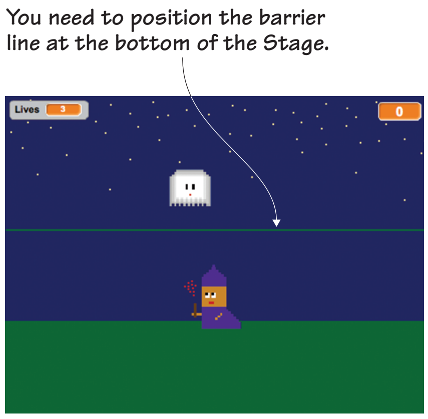

You need to position the barrier line at the bottom of the Stage.
Lives 3
0

Figure 7.30 It's going to be hard for those sparks to reach the ghost with the barrier line in the way.

To make the line positioning script

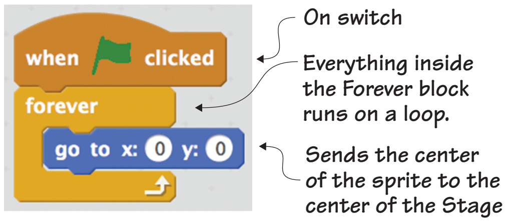
1. Start with a When Flag Clicked (Events) block.
2. Add a Forever (Control) block underneath to start a loop.
3. Place a Go to X/Y (Motion) block inside the Forever block and change both numbers to zero (0). This will send the center of the sprite to the center of the Stage.

ANSWER THIS WHY SET THE X AND Y COORDINATES TO ZERO WITH THE BARRIER LINE?

Question: isn't X:0 and Y:0 the center of the Stage? Why use those numbers if you want the line to be at the bottom of the screen?

Answer: remember that every sprite comes with a transparent background. The barrier line sprite may look like a tiny two-dimensional line, but the sprite is the size of the whole Stage (or Art Editor). To imagine this with real objects, cut out a tiny strip of black paper and put it at the bottom of a square piece of plastic wrap. That's how Scratch processes your sprite. Because you drew the line at the bottom of the Art Editor, exactly where you wanted it to be for the game, you can use that as the transparent center of the sprite, which matches the center of the Stage.

The script in figure 7.31 sends the line to the bottom of the Stage because the line was drawn at the bottom of the Art Editor. It is matching the center of the sprite with the center of the Stage, and therefore the line is positioned exactly where it is drawn.



```
when [green flag] clicked
forever
    go to x: 0 y: 0
```

On switch
Everything inside the Forever block runs on a loop.
Sends the center of the sprite to the center of the Stage

Figure 7.31 The completed positioning script sends the line to the bottom of the Stage.

Making a scoring script for the background

You can also program the background, which may sound a little odd. It's not that you want your background to be able to run around the screen, but there are times when you'll want to tuck pieces of code into the backdrop. Sprites aren't the only programmable piece in your game.

Sometimes assigning too many starting tasks to a single sprite can slow down a game. Imagine your game as a swim race, and all the sprites are the swimmers. They all have a starting task, lining up on the side of the pool and getting into position. They can do their single task and be ready at the same time. But what if you asked one swimmer to pass out goggles to all the other swimmers before the race as well as get in position? Now that swimmer is going to be delayed and unable to start the race at the same time as the other swimmers.

That's what can happen if you load one sprite with too many "getting ready" tasks that start a game. Because you're already giving almost every sprite a starting task—from cloning to positioning—you can give this setting-the-initial-score task to the backdrop. While the sparks are cloning and the ghosts are hiding up at the top of the screen, your game can simultaneously set the score to 0 and the lives (or game chances) to 3. And now every part of your game is ready to run at the same time.

To program the backdrop, navigate to the Sprite Zone and click the picture of your backdrop on the left side so the blue box is around the thumbnail, as it is in figure 7.32.

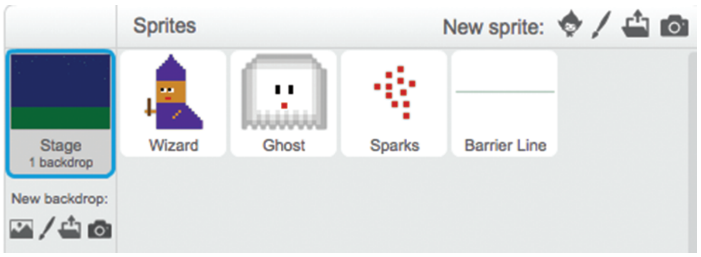


Figure 7.32 The blue box is around the Stage thumbnail in the Sprite Zone.

You will now build your scoring script in the Script Zone, the same as programming any sprite. You need to set the starting values of your Lives and Score variables. The value of Lives should be set to three to give the player three chances to get a high score fighting the ghosts. The value of Score should be set to zero (0), as in figure 7.33, so the player can start building their points as they shoot down ghosts.

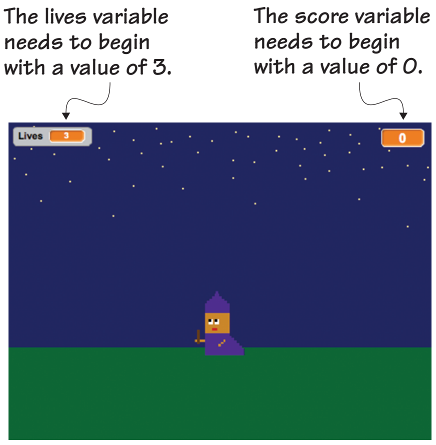

The lives variable needs to begin with a value of 3.
The score variable needs to begin with a value of 0.
Lives 3
0

Figure 7.33 The two variables of Lives and Score need to be set before the game begins.

You may notice as you look at the Block Menu that certain blocks are missing. Don't worry—you still have them. Scratch doesn't display them with the backdrop to streamline the menu, including only the blocks you're able to use. Click around the menus for a moment and see what's missing. All those blocks will be back when you return to programming sprites; they're only missing when you program the backdrop because the backdrop can't move or touch anything.

To make the scoring script

1. Start with a When Flag Clicked (Events) block.
2. Move two Set Score to 0 (Data) blocks and stack them, one on top of the other, under the When Flag Clicked block. Open the drop-down menu on the top block and set it to Lives. Type a 3 in the value box. This will give the player three chances to fight the ghosts. Open the drop-down menu on the bottom block and set it to Score. Type a zero (0) in the value box. This will give your player a blank slate for points when the game opens.

This final script should match the one in figure 7.34.

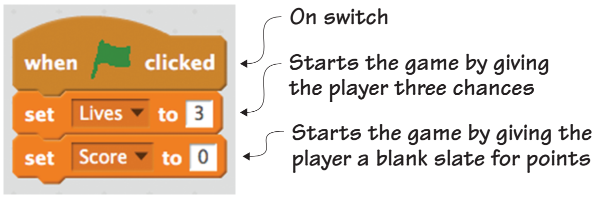

when clicked
set Lives to 3
set Score to 0
On switch
Starts the game by giving the player three chances
Starts the game by giving the player a blank slate for points

Figure 7.34 The completed scoring script sets the initial values for the two variables in the game.

Your game is finished and ready to be played! Click the flag and start blasting those ghosts, and then keep reading in case you run into problems with your game.

Troubleshooting your game

Our game went off without a hitch, but that doesn't mean that *your* game will work the same way. Use the ideas in this section to get your game working properly.

Checking your scripts

The first thing to do whenever a game doesn't work according to plan is to go back through the chapter and check your scripts against our scripts. Look carefully at each block because there are many similarly named blocks. Also look at values: do your numbers match the ones in the book?

Sprites not centered

You learned how to center your sprites when writing the scripts for the sparks. Turn back a few pages and reread those instructions if sprites are not behaving properly. If it's not a layering issue, it may be a centering issue, especially when it comes to the sparks lining up with the wizard's wand.

Eliminating blocks

Sometimes you may be inclined to delete a block that seems redundant, wondering if it's necessary for making the game. But wait! There is always a reason a block is included in a game, or code is broken up into two or more chunks, so please go back and follow the directions in the book if your game isn't working properly. An example is the seemingly redundant loop inside a loop that comes with the spark's movement script. When the Forever block is removed, it causes sparks to randomly generate in strange places on the screen and then shoot off on their own accord. Unless you're looking to produce phantom sparks in your game, stick to our instructions, because we've already troubleshot for you.

Learning in action

These challenges once again play with the existing code, making the game function in new ways. By playing with values, you can see how the script controls the various sprites.

Play with the code

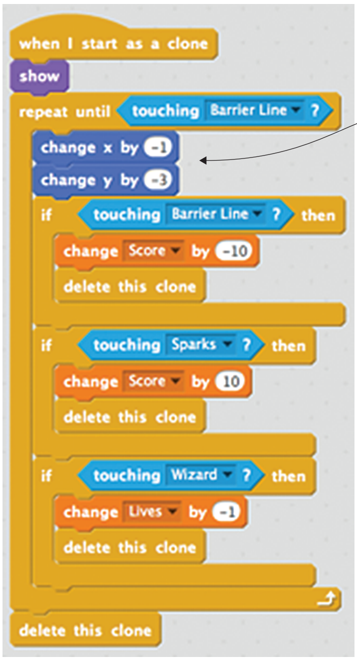
This fixed shooter is also a reflex-testing game, and you can play with the values to make the game harder or easier to play.

CHALLENGE What happens if you slow down the wizard but speed up the ghosts? Remember, the lower the number, the slower the sprite will move. The higher the number, the faster the sprite will move. How high can you score when the ghosts can fall 20 coordinates at a time but the wizard can only glide 2 coordinates at a time?

This is your second game, and you're ready to go beyond playing with values and tweaking the speed. Can you figure out how to change the way the ghosts move?

CHALLENGE Each ghost currently moves in a straight line from the place where it generates at the top of the screen toward the bottom of the Stage. What if you want the ghost to move at a diagonal, so it's more difficult to position the wizard underneath the target?

You can do this by adding a Change X by 10 block to the ghost movement script, as in figure 7.35. By slipping in the extra block and changing the value to a much smaller number, such as –1, the ghost will gently drift to the left as it falls. Additionally, you can change those values to have the ghost fall sharply to the side.



```
when I start as a clone
show
repeat until <touching [Barrier Line ▾] ?>
    change x by (-1)
    change y by (-3)
    if <touching [Barrier Line ▾] ?> then
        change [Score ▾] by (-10)
        delete this clone
    if <touching [Sparks ▾] ?> then
        change [Score ▾] by (10)
        delete this clone
    if <touching [Wizard ▾] ?> then
        change [Lives ▾] by (-1)
        delete this clone
delete this clone
```

Adding the Change X by -1 block with the Change Y by -3 block makes the ghost move at an angle instead of straight down.

Figure 7.35 An extra block in the ghost's movement script causes it to move in a diagonal line instead of straight down.

What did you learn?

Before you go back to your game and get a high score blasting down ghosts, take a moment to reflect which common computer science ideas from chapter 3 were used in this game:

- Using an on switch for every script in the game, including the more unusual When I Start as a Clone block and the When Space Key Pressed block

- Moving the ghosts down and the sparks up with X and Y coordinates
- Writing conditional statements to detect if the value of the Lives variable is at zero (0) so the game will end
- Setting up a loop (in all but two scripts!) to accomplish many tasks, including cloning the ghosts and giving the wizard unlimited sparks for his wand
- Using a variable to keep track of the number of ghosts the wizard blasts down with his wand
- Working with touching blocks and Booleans to check if the sparks make contact with a ghost
- Cloning all the ghosts and sparks for the game from a single copy of each sprite

Once again, you put into action seven out of eight common programming ideas. You'll continue to put these computer science ideas into action in future games. Additionally, you learned

- How to make multiple clones occur in the same game
- How to apply loops to solve many varied coding tasks
- How to use a single variable to increase or decrease, depending upon the situation in the game, to make a two-way point system
- How to make a fixed shooter

Okay, go blast away some ghosts. But after you're done getting your high score, turn the page because it's time to make another game. This one's based on the Atari game *Breakout*. You're back to working with the ball and paddle format, only this time the game is one player, and you need to remove pieces of a soccer net while bouncing a ball against a shoe paddle. Goal!

KIDS/BEGINNING PROGRAMMING

Hello Scratch!

Gabriel Ford, Sadie Ford, and Melissa Ford

Can 8-year-olds write computer programs? You bet they can! In Scratch, young coders use colorful blocks and a rich graphical environment to create programs. They can easily explore ideas like input and output, looping, branching, and conditionals. Scratch is a kid-friendly language created by MIT that is a safe and fun way to begin thinking like a programmer, without the complexity of a traditional programming language.

Hello Scratch! guides young readers through five exciting games to help them take their first steps in programming. They'll experiment with key ideas about how a computer program works and enjoy the satisfaction of immediate success. These carefully designed projects give readers plenty of room to explore by imagining, tinkering, and personalizing as they learn.

What's inside

- Learn by experimentation
- Learn to think like a programmer
- Build five exciting, retro-style games
- Visualize the organization of a program

Written for kids 8–14. Perfect for independent learning or working with a parent or teacher.

Kids know how kids learn. *Sadie* and *Gabriel Ford*, 12-year-old twins and a formidable art and coding team, wrote this book with editing help from their mother, author *Melissa Ford*!

Free eBook
SEE INSERT

"Brilliant writing on the art and science of game making using Scratch! Applicable to anyone new to game development ... solid examples throughout the book."

—Peter Lawrence, SAS

"Very well written. There were so many things to learn, I wished there had been more chapters in the book!"

—Khaled Tannir, dataXper

"A great book for learning how to program your own games while enjoying quality family time with your kids."

—Gonzalo Huerta-Cánepa
Universidad Adolfo Ibáñez

"An excellent guide through the world of Scratch."

—Karim Alkama, student

To download their free eBook in PDF, ePub, and Kindle formats, owners of this book should visit www.manning.com/books/hello-scratch

ISBN-13: 978-1-61729-425-9
ISBN-10: 1-61729-425-X
53499
9 781617 294259

MANNING $34.99 / Can $46.99 [Including eBOOK]