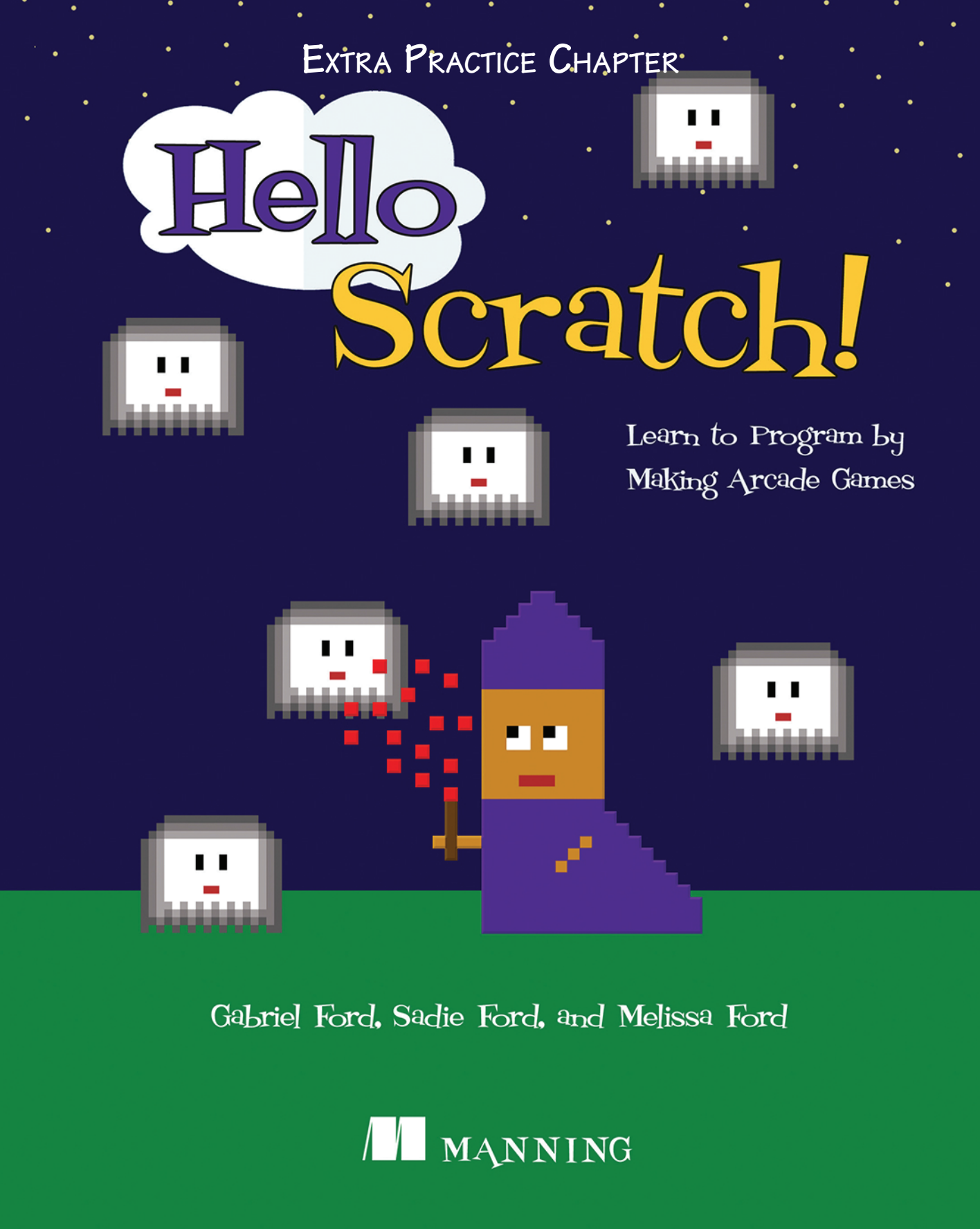
Extra Practice Chapter

Hello Scratch!

Learn to Program by Making Arcade Games



Gabriel Ford, Sadie Ford, and Melissa Ford

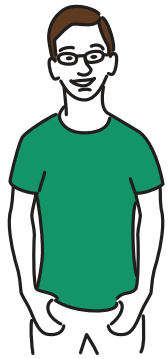
MANNING

Hello Scratch!

by Gabriel Ford, Sadie Ford, and Melissa Ford

Extra Practice Salad Catch—Coding

Extra Practice
Salad Catch—Coding



Have you ever played baseball? If you've been up at bat, you know that you get a split second to decide whether or not to swing at the ball and then make a connection if you do decide to go for a hit. Baseball, like many early video games, is all about your reaction time.

Take, for instance, Atari's *Avalanche*, one of the first reflex-testing video games (also called a *twitch* game), which came out back in 1978. The player controlled a stack of lines, which were used to catch falling rocks. A few years later, in 1981, Activision took *Avalanche*'s simple black-and-white graphics and remade the game as the more colorful *Kaboom!* In that game, a criminal is dropping bombs, and it's up to the player to slide the bucket left and right to catch them before they hit the ground and explode. Think for a second: I bet you can come up with many examples of reflex-testing games—from *Galaga* to *Tetris*.

Reflex-testing games force the player to note what is happening on the screen and take action...quickly. Are rocks falling? Catch them! Are bombs dropping? Don't let them hit the ground. In the game you're about to make, Salad Catch, vegetables are flying through the air, and you need to collect them in a wooden salad bowl—the same goal as *Kaboom!*

A player's reaction time is the most important part of a reflex-testing game, and this is something to keep in mind as you make Salad Catch. How hard do you want the game to be? By playing with the values in the

game, you will be able to make the carrots drop slowly or speed up like the bombs in *Kaboom!* during the game.

You'll use the salad bowl and carrots seen in figure 1 to make Salad Catch, as well as the barrier line positioned at the bottom of the Stage.

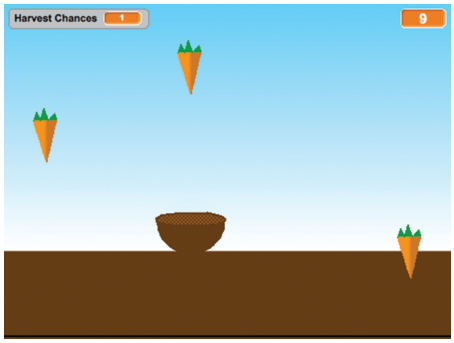


Figure 1 In Salad Catch, the player moves the bowl to catch the carrots, which drop from the top of the Stage. The game tracks how many carrots are caught as well as how many fall outside the bowl.

Salad Catch is easily customizable and is a good place for you to pause and think about your intended player. You don't want to make the game too challenging, or the player may get frustrated and stop playing. You also don't want to make the game too easy and have the player rack up hundreds of carrots while practically falling asleep.

In this section, you will learn

- How to use cloning to take one sprite and turn it into an infinite number of sprites
- How to generate sprites mid-game
- How to set up more than one variable to track many kinds of information simultaneously in the game

Open up your Salad Catch project where you made your sprites and get ready to snap some blocks together.

Preparing to program

You're almost ready to write the code. There are a few steps you need to take to prepare the Stage.

Missing sprites

If you skipped the last practice section, either flip back and complete it or go to the Manning site and download the background and sprites for Salad Catch. The directions for importing are the same as chapter 5. You should have a carrot, a salad bowl, and a barrier line, as well as the background.

Preparing the stage

Keep the bowl at its current size, but shrink the carrot by six clicks using the Shrink tool in the Grey Toolbar. Move the bowl until it's resting on top of the dirt, as in figure 2. If it's too low, seeing the bowl will be difficult because both the ground and the outside of the bowl are the same shade of brown. You don't need to move the carrot or the line because those will be positioned with code.

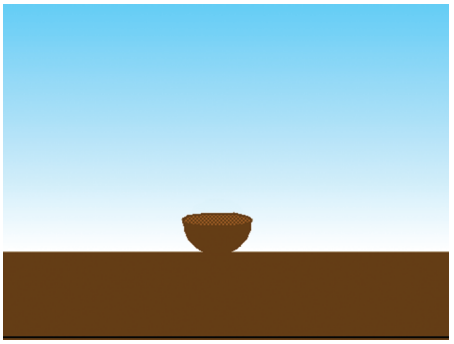


Figure 2 The bowl is placed directly on top of the brown dirt ground.

Programming the salad bowl

The salad bowl is the equivalent to the bucket in *Kaboom!* You will move it back and forth to catch the falling carrots. The bowl has a single task, and it is programmed with a single movement script. Make sure the blue box is around the salad bowl sprite in the Sprite Zone. Remember, the names or values on *your* blocks may slightly differ from time to time, so use the completed script images to make sure you chose the correct block.

Making a bowl movement script

Oh no! The bowl is going to miss the falling carrots in figure 3 because it can't move. You need to code the bowl so it can move left and right over the dirt.

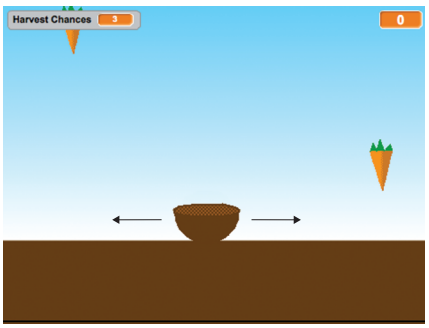


Figure 3 The bowl currently can't move to catch the carrots. It needs to be coded to move left and right.

This script will allow the bowl to move side to side when the player presses the arrow keys on the keyboard.

To program the salad bowl

1. Navigate to the Block Menu and click the Scripts tab.
2. Click Events and drag a When Flag Clicked block to the Script Area. Release it near the top of the workspace.
3. Switch to Control and slide a Forever block underneath the When Flag Clicked block so they snap together. This Forever block will create a loop, which will make the contents of the block run indefinitely.
4. Place two If/Then blocks inside the Forever block, one on top of the other.
5. Go to Sensing and drag a Key Space Pressed block into each of the empty hexagonal spaces in the If/Then block, as in figure 4. This will set up a conditional statement where a certain action happens if the designated key is pressed.

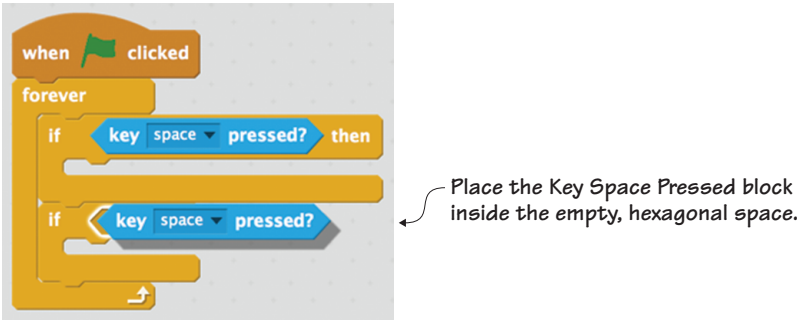


Figure 4 Place the Key Space Pressed sensing block inside the empty space in the If/Then control blocks.

6 Change the word Space to Left Arrow in the top sensing block and Right Arrow in the bottom sensing block using the drop-down menu.

7 Click Motion and drag two Change X by 10 blocks, placing one inside each If/Then block. Remember, your Change X by 10 block may have a different number listed. It is still the same block.

8 Change the value in the top Change X by 10 block to –8. This will move the bowl eight coordinate spaces to the left each time the left arrow key is pressed, because negative X-coordinate numbers move left.

9 Change the value in the bottom Change X by 10 block to 8. This will move the bowl eight coordinate spaces to the right each time the right arrow key is pressed, because positive X-coordinate numbers move right.

The first script is complete. Although it's simple, it does an important task, allowing the player to move the bowl to catch the carrots. Does your script match the one in figure 5?

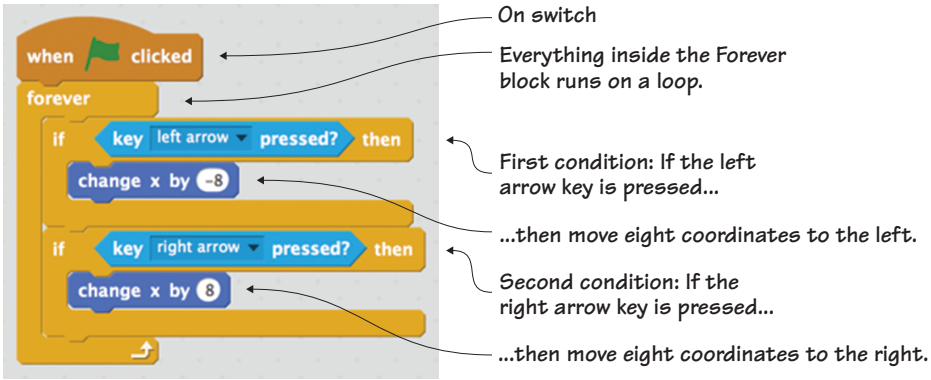


Figure 5 The completed movement script for the salad bowl

Like the script you wrote back in chapter 5 for the pans in Breakfast Wars, this script is powered by a conditional statement that checks whether the player is pressing the left or right arrow keys on the keyboard. If they are, it moves the salad bowl eight coordinate spaces to the left or right. If they're not, nothing happens.

Programming the carrot

The carrots are the equivalent of the bombs in *Kaboom!* Notice I used the plural, *carrots*. But wait a minute...you only have a single carrot sprite. This game will require many carrots, but you won't use the Duplicator tool in the Grey Toolbar. Instead you'll code the duplicates using cloning.

The bulk of the coding in this game is applied to the carrot sprite. You will create a cloning script, clone movement script, project stopping script, and scoring script. Keep the blue box around the carrot sprite in the Sprite Zone until you have finished writing all four scripts.

Making a cloning script

Right now, you only have one carrot. But Salad Catch requires many carrots. Notice that multiple carrots are falling in figure 6.

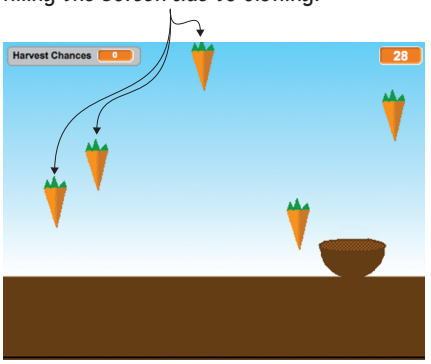


Figure 6 The cloning script generates a copy of the carrot whenever the game needs to drop a new carrot sprite.

This script will generate new copies of the carrot sprite mid-game. Additionally, it will also set a few values for the game, such as the number of tries the player has to collect the carrots (called Harvest Chances in this game), as well as track how many carrots land in the bowl. But the bulk of this script clones the carrot sprites.

It is a long script with 20 steps. Go slowly and check your work against the provided images. Make sure the blue box is around the carrot in the Sprite Zone.

To create this cloning script

1. Navigate to the Block Menu and click the Scripts tab to access all of your blocks.
2. Click Events and drag a When Flag Clicked block to the Script Area. Release it near the top of the workspace.

3 Click Data and get ready to make four variables. All will be applied to all sprites, so make sure you choose that option each time. Click Make a Variable and name the first one Carrots. Click Make a Variable again and name the second one Delay. Make a third called Harvest Chances and a fourth called Speed. Leave the boxes next to Carrots and Harvest Chances checked so their value will appear on the Stage. Leave the boxes next to Delay and Speed unchecked so their value is hidden.

4 Slide the Harvest Chances variable on the Stage to the top left corner and slide the Carrots variable on the Stage to the top right corner. You can change the look of the boxes by right-clicking them (or control-clicking on a Mac). In figure 7, the Harvest Chances box is untouched, but I changed the Carrots box to the large readout option by right-clicking the variable box on the Stage so it presents as a single number (currently zero).

5 Drag four Set Delay to 0 blocks (your blocks may have a different variable listed) and snap them one after the other underneath the When Flag Clicked block in the Script Area.

6 Change the first block to Harvest Chances using the drop-down menu. Change the value from 0 to 3. This means that the player will have three chances to catch the carrots, and they will lose one of these chances each time a carrot misses the bowl and hits the ground.

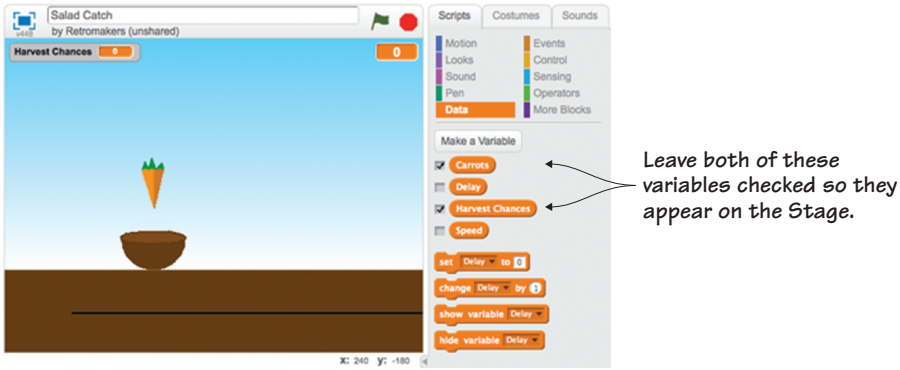


Figure 7 Four variables track information in the game, but only two variable boxes are checked so their values show on the screen.

7 Change the next block to Carrots using the drop-down menu. Keep the value at zero (0) because the player will start each game with zero carrots, though that value will increase during the game.

8 Change the next block to Speed using the drop-down menu. Set that value to –5. That –5 means that the carrots will fall down five coordinate spaces at a time. But the speed of the carrots will change with each new level of the game because the value of that variable will change during the game. When it becomes –6 or –7 in the future, the carrots will fall faster.

9 Change the last block to Delay using the drop-down menu. Set that value to 1. This will leave a one-second delay between each carrot so the player doesn't have dozens of carrots dumped at once. Your script should currently match figure 8.

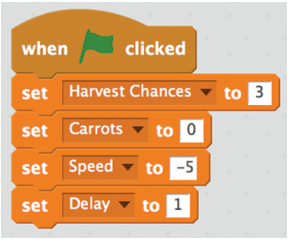


Figure 8 Each of the four variables is set with a different starting value.

10 Switch to Looks and drag a Hide block to the bottom of your script in the Script Area. This Hide block applies to the carrot sprite, and the reason you're hiding it is that you don't want the player to be able to see where the carrot spawns when Scratch clones a new carrot. They can't predict where it will be until it begins falling.

11 Click Motion and drag a Set Y to 0 Block underneath the Hide block. Change the 0 to 180 to make the carrots fall from the top of the Stage.

12 Go to Control and grab a Forever block. Place it at the bottom of the script to create a loop.

13 Place a Repeat 10 block inside the Forever block. Change that number to 20. Why 20 instead of 10? Each level will drop 20 carrots before the speed at which they are dropped increases.

14 Return to Motion and drag a Set X to 0 block inside the Repeat 20 block. But wait! You're going to do something interesting with this block.

15 Go to Operators and grab a Pick Random 1 to 10 block. Slide it into the bubble that currently contains the zero (0) in the Set X to 0 block,

as shown in figure 9. By the way, this is a good time to check the order of your blocks against our blocks in the figure.

16 Change the two numbers in the Pick Random block to –240 and 240. This will make the carrots randomly spawn anywhere across the top of the Stage (because –240 and 240 are the farthest left and farthest right X-coordinates on the Stage).

17 Go back to Control and drag a Create Clone of Myself. Put it under the Set X to 0 block inside the Repeat 20 block. This is the block that will create a duplicate of the carrot sprite.

Figure 9 You can place another block inside any writable space on another block. For instance, placing the Pick Random 1 to 10 block inside the number bubble on the Set X to 0 block tells Scratch to randomly choose the X position for the carrot sprite each time it generates.

18 Drag a Wait 1 Secs block and place it under the Create Clone of Myself block (still inside the Repeat 20 block). You're going to change that 1-second delay to an ever-changing number by placing a variable block inside that number bubble.

19 Click Data and drag the Delay variable block (it has rounded edges and looks like a little orange pill) into the space containing the 1 in the Wait 1 Secs block. This will cause Scratch to wait a shorter and shorter amount of time over the course of the game as the value of that variable changes. This means that the game will get harder as that variable value changes, and carrots will spawn with a smaller delay between each one when the player gets to each new level.

20 Remain in Data and drag two Change Delay by 1 blocks. Place them underneath the Repeat 20 block, but inside the Forever block. Using the drop-down menu, change the first block to speed (Change Speed

by 1) and make sure the second block is delay (Change Delay by 1). Make the value of Change Speed by 1 to –1 because you want the carrot to fall slightly faster each time by covering more Y-coordinate spaces at a time. Change the value of Change Delay by 1 to –0.1. This slightly shortens the delay time between each new carrot by a tenth of a second each time the player moves to a new level.

That is the very long cloning script. Check your script against the script in figure 10 and make sure they match.

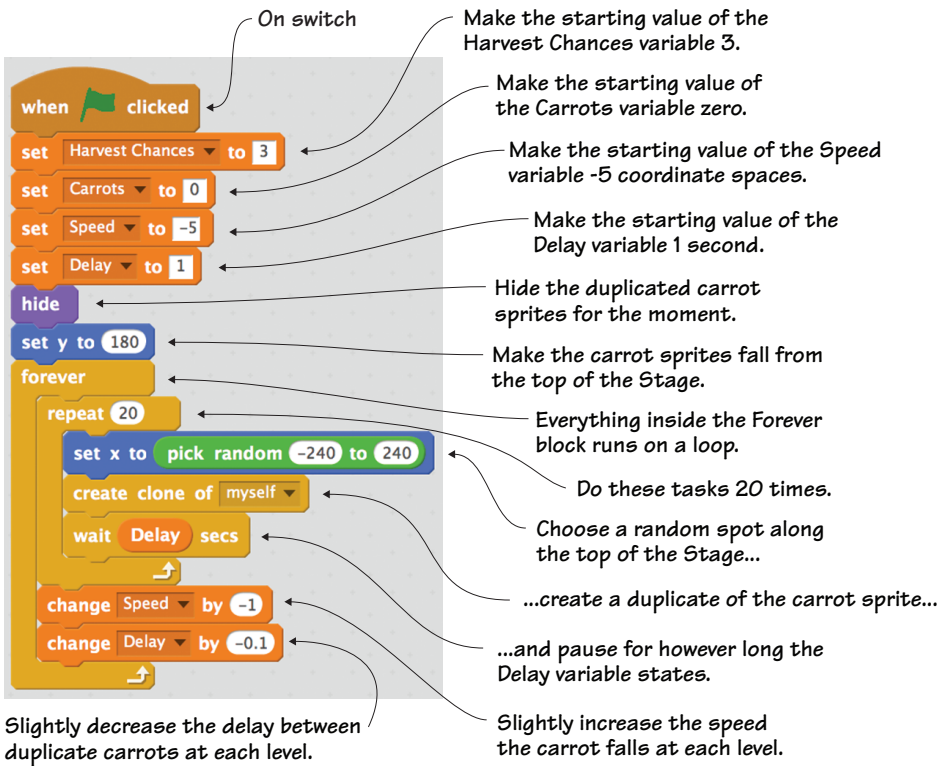


Figure 10 The completed cloning script covers a lot of ground, from setting up variables to telling Scratch how to clone the carrot sprite.

Great job! This is a complicated script that completes many tasks in the game. It sets up variables that will give the player a set number of

chances, keep score, and make the game increase in difficulty at each level. But this script is called a *cloning* script because it also tells Scratch to duplicate the carrot sprite but keep the copies invisible for the time being. It tells Scratch to spawn those carrots in random places across the top of the Stage and to duplicate with a smaller and smaller delay between carrots as the player moves up levels. Finally, after each group of 20 carrots, it changes the value of the Speed and Delay variables in order to make the game harder. This complicated script accomplishes a big chunk of the work of this game.

Making a clone movement script

You now have carrot sprites being generated, but they're invisible and frozen at the top of the Stage. It's going to be a pretty boring game if the carrots remain that way, as in figure 11.

The carrots are being cloned at the top of the screen, but they're still invisible.

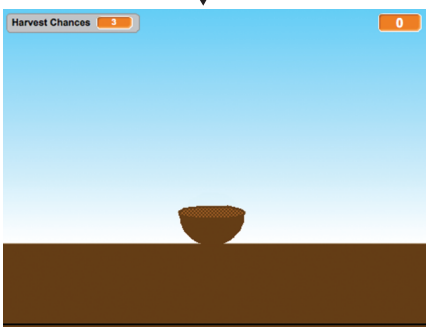


Figure 11 Where are the carrots? The cloned carrots need to become visible and start moving.

This script will make the carrots visible (because it's difficult to catch invisible carrots!) and make them start falling down the Stage toward the bottom of the screen.

To make this clone movement script

1. Click Control and slide a When I Start as a Clone block into the Script Area. Start this new script somewhere near (but not touching!) the existing script in your workspace. This block is still an on switch, but it only works if there is a clone of the carrot sprite. If the duplicate sprites haven't been made, this script won't run.
2. Switch to Looks and snap a Show block to the When I Start as a Clone block. This will cause the carrot sprite clones to become visible after they generate at the top of the Stage.
3. Return to the Control block menu, choose a Repeat Until block, and place it under the Show block.

4 Go to Sensing and choose the Touching block. Slide it into the empty hexagonal space in the Repeat Until block. Use the drop-down menu to change the option to Barrier Line. The action inside will happen on a loop until the carrot sprite touches the barrier line sprite.

5 Switch to Motion and grab a Change Y by 10 block. Place it inside the Repeat Until block.

6 Navigate to the Data block menu. You're going to place another block inside the space currently housing a number. Drag the speed variable inside the number bubble, covering up the 10. Look at figure 12 to see this block entering the space. By replacing the static number with an ever-changing variable value, the carrot will drop at an increased rate as the game continues.

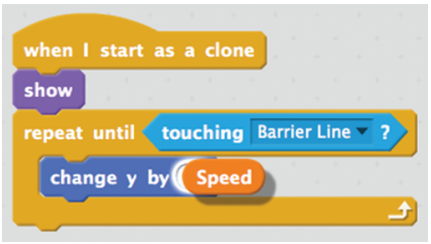


Figure 12 The variable replaces the number, making the carrot sprite increasingly fall faster along the Y-coordinate line as the game continues to higher levels. Remember that the value of the variable speed increases by 1 extra coordinate every 20 carrots. Carrots start falling at five coordinates at a time and increase to six coordinates, seven coordinates, and so on.

7 Drag a Change Delay by 1 block and slip it under the Repeat Until block. Use the drop-down menu to change the option to Harvest Chances (which is how many turns the player has) and the value to –1. This removes a turn every time a carrot hits that barrier line.

8 Return to Control and snap a Delete This Clone block under the Change Harvest Chances by –1 block. This will remove the clone once it reaches the bottom of the Stage. If not, you'll be able to see the little green tops of the clones pile up at the bottom of the screen.

Your carrot sprites can now fall down the screen toward the bowl if your script looks like the one in figure 13. Of course, it's up to the player to be able to move fast enough to catch them.

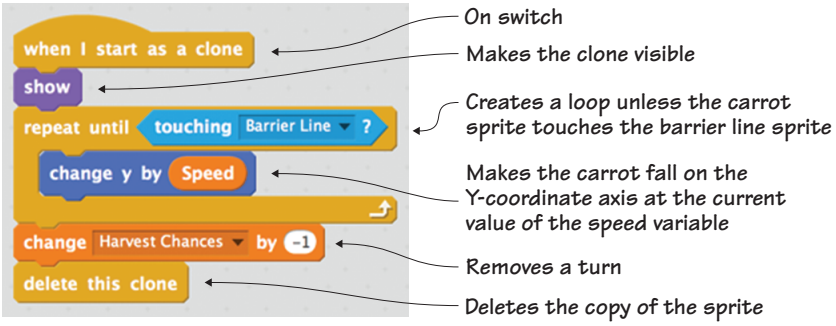


Figure 13 The completed clone movement script gets the carrots to drop from the top of the Stage.

Making a project stopping script

Right now the carrots can fall, disappear after they touch the barrier line, and even be used to take away a turn if they don't land in the salad bowl, but there's no way for the game to end. This means you can run into negative Harvest Chances, as in figure 14, which is a little odd.

Without a stopping point, the player can go into negative turns.

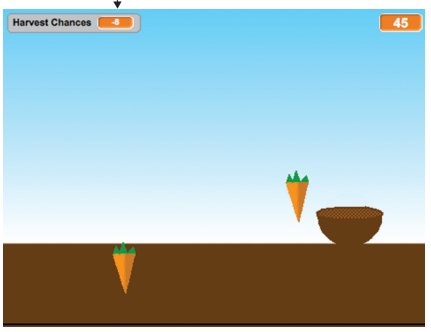


Figure 14 The player can continue indefinitely, with the turn counter showing a negative number.

This script will tell the game when you stop. Keep the blue box around the carrot in the Sprite Zone since you're still programming it.

To make the project stopping script

1. Go to the Events menu and drag a When Flag Clicked block to the Script Area. Release it near the top of the workspace, near (but not touching!) the other scripts.
2. Switch to Control and slide a Forever block under the When Flag Clicked block. Yes, you are once again setting up a loop so the actions inside the block will occur over and over again.
3. Put an If/Then block inside the Forever block so you can set a condition.

4 Navigate to Operators and drag a Square = Square block. Put it inside the empty hexagonal space in the If/Then block.

5 Go to Data and fill the left square in the Square = Square block with the Harvest Chances variable. Type zero (0) into the right square. The condition is set: if the value for the Harvest Chances variable is zero, then do this next action.

6 Return to the Control menu and slide a Stop All block inside the If/Then block. This will stop all the scripts, making it game over.

The small script in figure 15 ends the game.

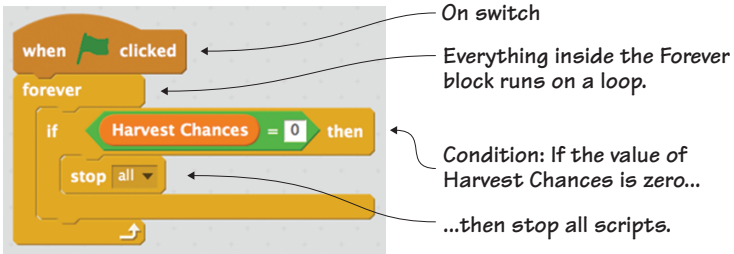


Figure 15 The completed project stopping script ends the game.

There is one last script to make for the carrot, so you can keep score of how many carrots land in the bowl.

Making a scoring script

You start off the game with zero carrots in your bowl, and that number in figure 16 is going to always remain zero unless you set up a way to track the score.

This script will give the player a point, increasing the carrot count in the top right corner of the Stage every time a carrot touches the salad bowl sprite.

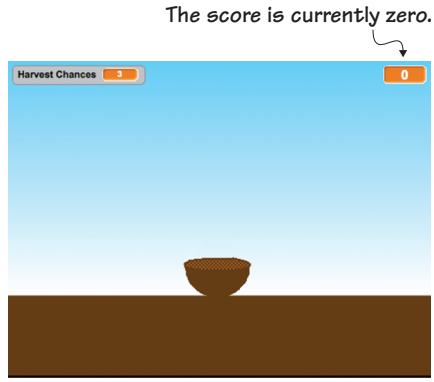


Figure 16 This next script will allow the carrot count to increase during the game whenever a carrot lands in the salad bowl.

To make the scoring script

1. Click Control and slide a When I Start as a Clone block into the Script Area near the other three scripts in your workspace. This block is an on switch. It only works if there is a clone of the carrot sprite. If the duplicate sprites haven't been made, this script won't run.
2. Snap a Forever block under the When I Start as a Clone block to set up a loop.
3. Drag an If/Then block and place it inside the Forever block to set up a conditional statement.
4. Click Sensing and drag a Touching block into the empty hexagonal space. Use the drop-down menu to set it to Salad Bowl.
5. Navigate to Data and choose a Change Delay by 1 block. Slide it inside the If/Then block. Use the drop-down menu to choose Carrots. Keep the number in the number bubble at 1.
6. Return to the Control menu and grab a Delete This Clone block. Place it inside the If/Then block, underneath the Change Carrots by 1 block.

That is the whole script, as seen in figure 17. This script sets up a condition: if the carrot sprite touches the salad bowl sprite, increase the value of the Carrots variable by 1 (meaning, give the player one point) and delete the carrot sprite from the screen.

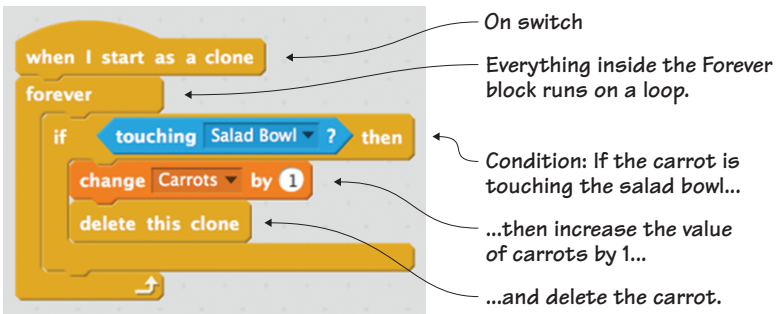


Figure 17 The completed scoring script tracks how many carrots land in the bowl.

ANSWER THIS

WHY DO YOU HAVE TO DELETE THE SPRITE?

Question: this is the second time you've included a Delete This Clone block. Why bother deleting the clones?

Answer: if you don't delete the clone after it touches the salad bowl, it will either linger in the air, or, even worse, continue down the Stage where it will run into the barrier line. This will cause the game to delete a turn even though the carrot was caught in the bowl! Therefore, delete the clone so it disappears once it has been counted.

Programming the odds and ends

The barrier line is a programmable sprite, serving as the "net" at the bottom of the screen. Although you could technically use the Y-coordinate –180 to indicate the bottom of the Stage, you'll get a better result from Scratch by programming your game to interact with another sprite rather than a space on the screen. Make sure the blue box is around the barrier line in the Sprite Zone.

Making a positioning script

Currently you have a line drawn, and you may see it in the middle of the dirt area on the Stage. It could even be higher than the salad bowl, as it is in figure 18, which means the carrot will disappear before it can reach its target.

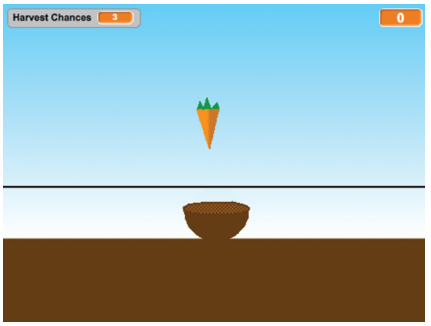


Figure 18 The line needs to be positioned at the bottom of the Stage or it could potentially stop the carrot from reaching the salad bowl.

This final script will position the line at the bottom of the Stage.

To make the positioning script

1. Navigate to the Block Menu and click the Scripts tab.
2. Click Events and drag a When Flag Clicked block to the Script Area. Release it near the top of the workspace.
3. Go to Control and slide a Forever block underneath the When Flag Clicked block.

4 Switch to Motion and drag a Go to X/Y block. Place it inside the Forever block. This will keep the line always positioned in the same place for the entire game.

5 Set both values in the Go to X/Y block to zero (0). This will center the line at the bottom of the Stage. If you imported the line sprite or drew it anywhere else on the canvas, change the Y position to –180 to make sure the line is at the bottom of the Stage.

It's a brief script, but you'll need it if you want the game to be playable. Look at figure 19 and make sure your script looks the same.

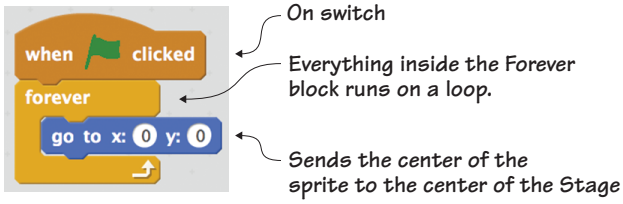


Figure 19 The completed positioning script sends the line to the correct place on the Stage.

Look at you go! Salad Catch is now ready to be played. Press the green flag and get ready to catch some carrots.

Troubleshooting your game

Our game went off without a hitch, but that doesn't mean that *your* game will work the same way. Your sprites may be larger or smaller than our sprites.

Checking your scripts

The first thing to do whenever a game doesn't work according to plan is go back through the section 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 section?

Fixing layering issues

Make sure the salad bowl is on the top layer of your game if the game isn't working correctly. Do you remember how to do that? Navigate to the Stage and click the salad bowl sprite. Slide it a centimeter to the left or right and then put it back into place. By clicking and moving the sprite, you bring it up to the top layer on the Stage.

Carrots falling off the screen

What if your carrots are generating half off the screen? Another simple fix: return to the cloning script and look at the –240 and 240 in the Pick Random block. Change those numbers to –200 and 200, which are 40 coordinate spaces from the far left side and far right side of the screen. The carrots will still fall randomly across the top of the Stage, but they'll now have a slight margin on either side.

Learning in action

These challenges 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

Remember, reflex-testing (or twitch) games are all about speed. You made your game highly playable, with the carrots falling around the same speed that the bowl moves. You also made your game slowly ramp up in difficulty. But if you play with a few values, you can change all of that for the player.

CHALLENGE Right now, your carrot and bowl are mostly in sync with one another, moving around the same speed. But what happens if you play with the number in the movement script for the bowl? Try making the number higher or lower. Do you overshoot the carrots by moving too many coordinate spaces at a time? Do you not reach the carrots in time because you're moving too slowly?

CHALLENGE Can you make the carrots fall faster or spawn faster? Which values would you have to play with in the cloning script in order to make the game super challenging from the beginning by infusing the carrots with super speed?

CHALLENGE Right now your game is perfect for an older kid or an adult, but can you make a version of the game that would be fun for younger gamers? How can you slow things down by playing with the variable values or give the player more chances to catch the carrots? Remember, reflexes get better with age and your reaction time peaks when you become an adult. Make your game playable for someone in kindergarten.

What did you learn?

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

- Using an on switch for every script in the game, even the unique When I Start as a Clone block
- Positioning the barrier line with X and Y coordinates
- Writing conditional statements to check how many Harvest Chances (turns) the player has
- Setting up a loop to create new copies of the carrot sprite until the game ends
- Creating four variables that track information, from the player's score to the number of turns they have left
- Working with touching blocks and Booleans to make the clone disappear when it touches the barrier line
- Cloning all the carrots for the game from a single carrot sprite

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 many copies of the carrot sprite by writing cloning into the program
- How to increase the difficulty level over the course of the game by using variables that change value during the game
- How to track points (or, in this case, carrots caught) by using a variable

You are ready to return to the last part of the book, "Coding and Playing Games." The games are going to get a little more complicated, and

you'll also notice that some of the reminders that are constantly repeated in the book are phased out. By now you should know your way around the workspace, where to drop your scripts in the Sprite Area, and when to move the blue box to a new sprite in the Sprite Zone. The training wheels are off as you move toward making a fixed shooter based on the Intellivision game *Astrosmash*.

This game will turn you into a wizard, blasting ghosts with your magic wand. Like Salad Catch, it also involves falling objects (though ghosts instead of carrots), only this time you'll be shooting them down instead of catching them.

KIDS/BEGINNING PROGRAMMING

Hello Scratch!

Gabriel Ford, Sadie Ford, and Melissa Ford

Free eBook
SEE INSERT

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*!

"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]