

For the Win Instructor Guide

Game Coding with Scratch

Preparation

Make sure that Scratch Desktop is installed and up to date. <https://scratch.mit.edu/download>

Load the file **BasketballHoop.png** onto student computers.

Set up the classroom so that students can see the instructor's screen.

You may wish to open up a blank Scratch project on each student computer to save class time.

If permitted by the host institution, consider inviting a parent or volunteer to document the event with pictures while you teach.

Make copies of the handouts, parent letter and survey for each student.

Advertise the class, and encourage participants to bring USB drives to save their games.

As Students Arrive

Ask them about their coding and game playing experiences. If students have created with Scratch or other programming languages, ask them about their projects.

Mark down students names on a seating chart, so you can easily refer to them later.

Starting Class

Introduce yourself and welcome the students.

If this is the first in a coding club series, explain that today's class will focus on some basic programming and game making concepts, and that we will continue to build on what we learn today through each advancing lesson.

Explain that we will be using Scratch, *a free, visual programming language from MIT*, that they can continue to use after the class.

Play a Game: Programmer Says

Gather the group in a circle and play a round of Programmer Says. Like participants in Simon Says, computers can only do that which the programmer commands. Begin with simple instructions and then add in some commands that we'll later replicate in the coded game:

Repeat Until ()

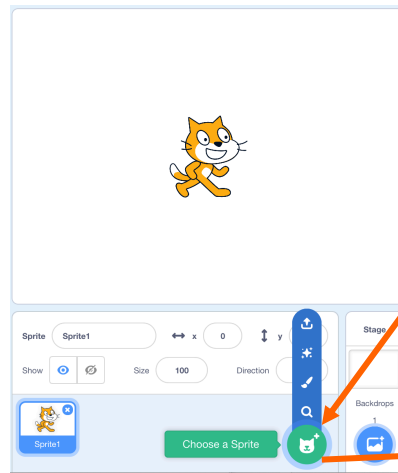
Programmer says, "repeat hopping on one foot until I clap my hands."

Forever (If ()Then)

Programmer says, "for the rest of the game, anytime I touch my ear, you stop what you're doing and crouch to the floor."

Make a Game: For the Win!

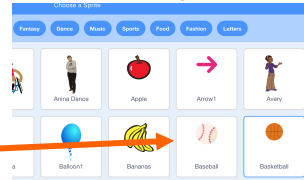
Step 1



Have students open Scratch.

A cat sprite, affectionately known as Scratchy, automatically loads in each new project and will be our basketball player for now. Students may customize this choice later.

Open the sprite library and select the Basketball sprite.



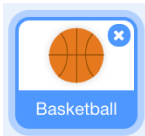
Discuss sprites.

Point out to students that Scratchy is labeled Sprite 1 and ask them what they think the term sprite means. Discuss as a group that the noun sprite has at least three meanings:

1. a small magical creature, like an elf or fairy
2. a standalone two-dimensional computer graphic that can be manipulated as part of a larger scene
3. a reddish-orange flash caused by electrical discharge about thunderstorms.

There is, of course, also a proper-noun version of the word that refers to a copyrighted carbonated beverage. In Scratch, *Sprites are visual elements that we can program and control.* Each of our sprites has its own scripting area where we tell it what to do.

Step 2

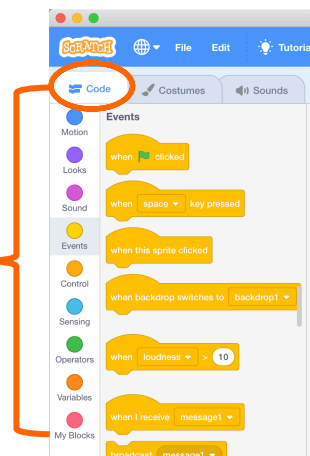


Make sure each student has the basketball sprite selected.

We're going to give it some scripts.

The *Block Palette* contains bits of code that snap together to form scripts.

Make sure the Code tab is selected.



Select the Events palette.

Each script is triggered by an event.

Click and drag **When (Green Flag) Clicked** into the scripts area.

The user will click the green flag above the stage panel to start playing the game and initiate the script.

Select the Control palette and attach a **Forever** block to the script.

A **Forever** block causes the blocks it contains to loop continuously from the time the program starts until the time it ends if the script is interrupted.

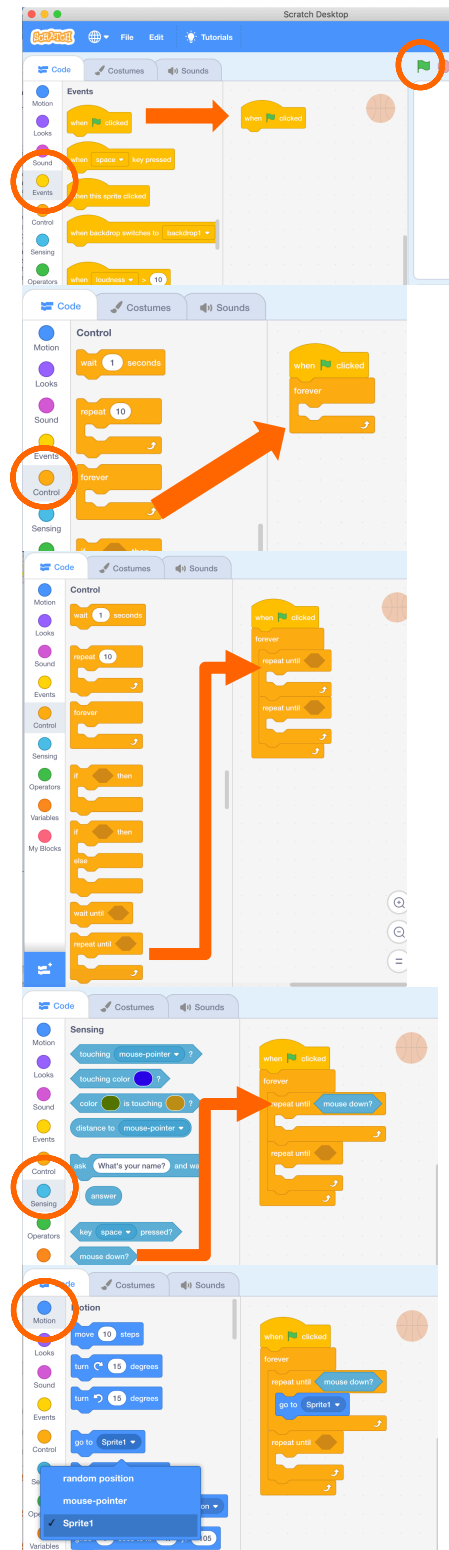
Stack two **Repeat Until ()** blocks inside the Forever block.

A **Repeat Until ()** block causes the blocks it contains to loop continuously until a specified condition is met. Remind students of the repeat until command used in Programmer Says.

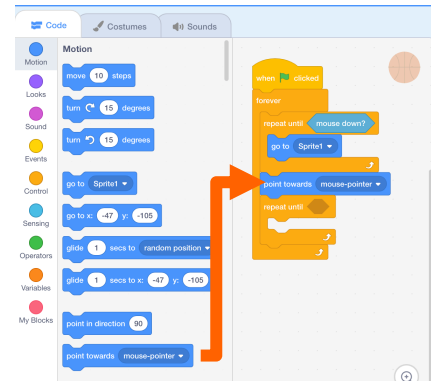
From the Sensing palette, drag a **Mouse Down?** block into the space beside the first **Until**.

From the Motion palette, drag a **Go to (sprite1)** block into the first **Repeat Until** loop.

Now, in our video game, the first Repeat Until () will tell the basketball to go to Scratchy until the mouse is clicked. Next, we need to tell it what to do once the mouse is clicked.

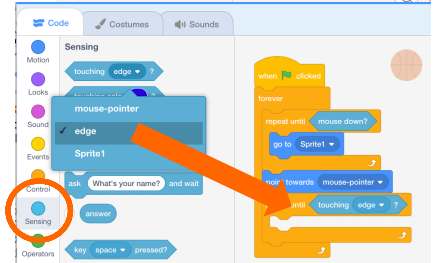


From the **Motion** palette, drag a **Point Towards (mouse-pointer)** block between the two **Repeat Until ()** blocks so that the basketball knows to aim toward wherever the player places the mouse on the screen.

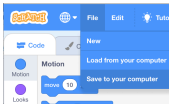
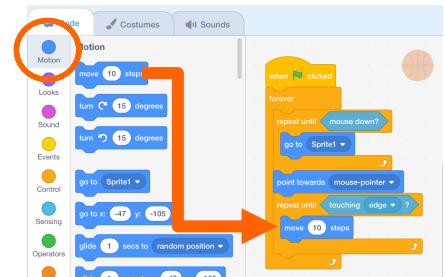


From the **Sensing** palette, drag a **Touching (edge)** block into the space beside the bottom **Until**.

This will tell the basketball to continue this loop until it reaches a boundary of the game window.



From **Motion**, drag a **Move (10) Steps** block into the bottom **Repeat Until ()** loop.



Go to **File > Save to Computer** and then test it by clicking the green flag. Each time the user clicks the screen the basketball should aim toward the mouse click. Once the ball reaches the edge it should seem to disappear and then return to Scratchy.

Step 3

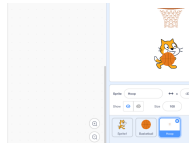
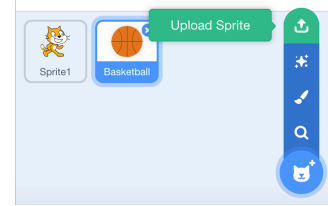


Hoop Motion Script

Upload Sprite

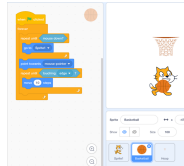
Navigate to BasketballHoop.png

Click OK.



Don't Panic!

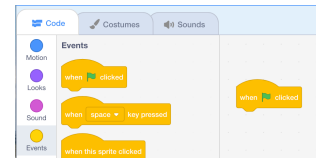
When the Hoop sprite is selected the code we previously created seem to vanish. That is because each script is tied to a specific sprite.



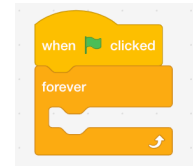
If we click on the Basketball sprite again, we'll see our old scripts are still there. Demonstrate to students, and then make sure everyone reselects the Hoop.

This script will make the Hoop move around the stage, giving players a moving target.

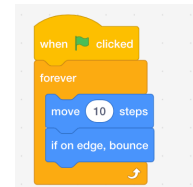
From the **Events** palette, select a **When (Green Flag) Clicked** block



From the **Control** palette add a **Forever** loop
Remember, the forever loop will cause the steps inside of it to happen over and over until the game ends.



From **Motion**, add
Move (10) Steps
and
If on Edge, Bounce

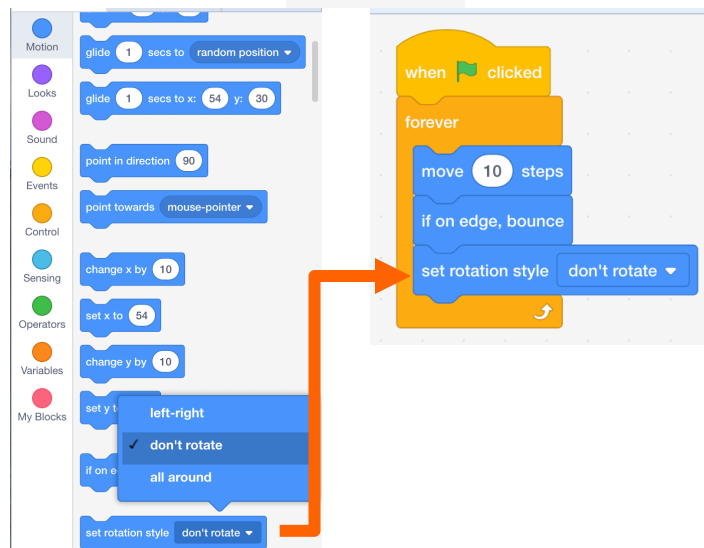


Save and Test.

The hoop should move back and forth, but something funny happens. It flips upside down each time it hits the edge. Let's fix that!

From **Motion**, add
Set Rotation Style (Don't Rotate)

Save and test. The hoop should stay upright now when it bounces.



Step 4 Score with Hit Script



With this script, each time the player successfully hits the hoop with the ball, the score will increase.

From the **Events** palette, select a **When (Green Flag) Clicked** block

Place it separately to start a new script.

Variables

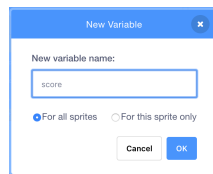
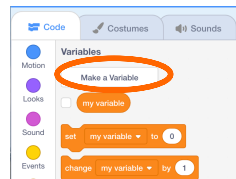
Go to the **Variables** palette.

Ask students if anyone has ever heard of a variable. What do they think the term means?

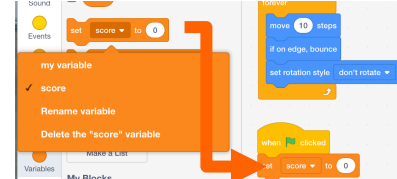
Ask students what the score of a basketball game would be when it first starts, and then what it would be after one team makes a regular basket. The variable we call *score* changes from 0-0 at tipoff to 0-2, then 2-2, 2-4, etc... The value assigned to the variable *score* keeps changing until the game is over.

A **variable** is a changeable value assigned to a letter, word or symbol. For our game, we'll create a variable called *score*.

Select a **Make a Variable** Name it *score*



Add **Set (score) to (0)**



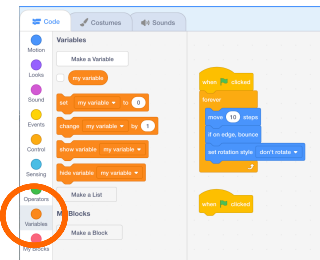
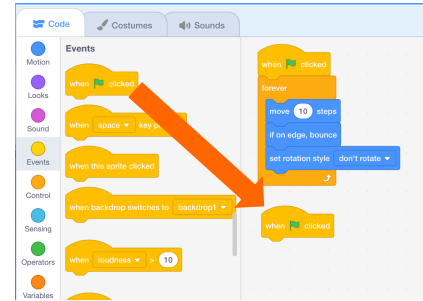
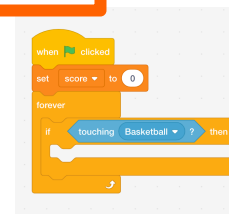
From the **Control** palette add a **Forever** loop

Add and **If () Then** loop inside of Forever

Remind students of the how we used an If/Then statement in programmer says. If the condition was True that the programmer was touching her ear, than they were to stop and drop. The same thing happens with this script. If the condition that we place between if and then is true, then the script does what is inside. i.e. if the Basketball is touching the Hoop then we get points.



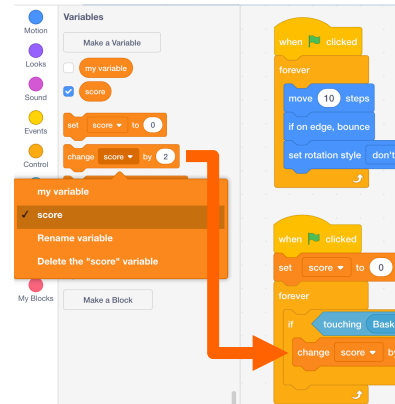
From the **Sensing** palette add a **Touching (Basketball)** block



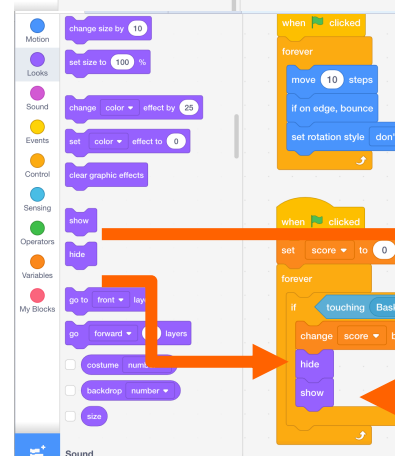
Return to the **Variables** palette and add **Change (score) by (2)**

Make sure to select **score** from the dropdown

Click in the white circle to replace 1 with 2



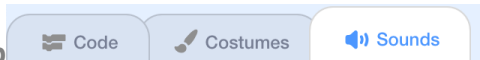
From the **Looks** palette scroll down to add **hide** and **show**



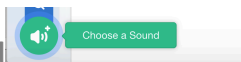
Sounds

Young students tend to get carried away with Scratch's built in ability to record sounds. If this issue arises, guide students to use a prerecorded sound for now, but assure them that they will have customization time later.

Select the **Sounds** Tab



Click **Choose a Sound**

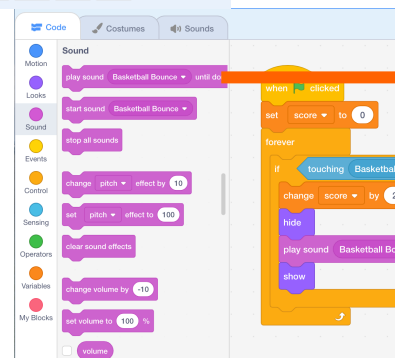


Select **Basketball Bounce** from the **Sports** tab or another brief sound



From the **Sounds** palette, add

Play Sound (Basketball Bounce) Until Done between the **Hide** and **Show**

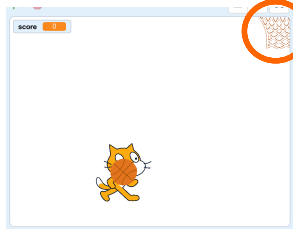


Save and Test

The hoop should disappear and the score increase each time the basketball find its target.

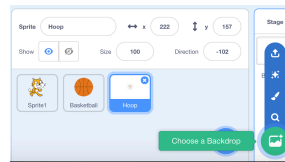
Step 5 Random Reappear with Coordinates

Currently the hoop's location on reappearance is pretty predictable, since it will maintain the same trajectory.



The first thing we can do to fix this is to **move the Hoop to a corner of the stage**. Now, it will bounce off of the top and side, causing it to move diagonally. **Save and test.**

The other thing we'll do is to use coordinates to randomly relocate the Hoop before each appearance.



Choose a Backdrop

Select XY-grid (students can customize this later)

Discuss Coordinates

Ask if anyone knows what the X and Y lines represent?

Explain that Scratch displays sprites based on the Cartesian Coordinate System in which each point on a plane has two values, an x and a y, to describe its exact position.

Click on the basketball sprite.

Change basketball's coordinates to (0,0).

Explain to students that we can think of the basketball as though it is moving along a tightrope made up of points. It is currently at the zero point on the x-axis. If we wanted to get it to the 100th point, we could change the x value accordingly.

Change basketball's coordinates to (100,0).

Explain to students that if we wanted to get the basketball to the 200th point on the line, we would need to "Change X by 100".

Change basketball's coordinates to (200,0).

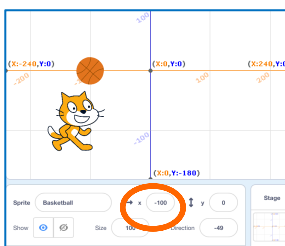
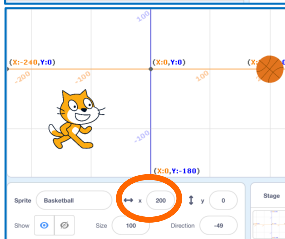
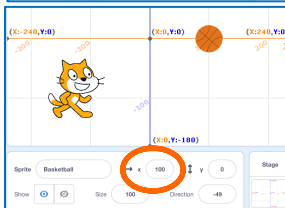
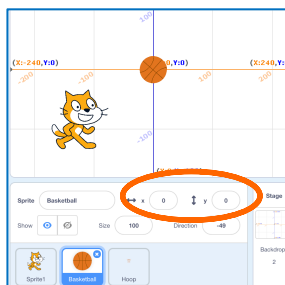
Ask students, if we wanted to move the basketball back to the left, would we add or subtract points? Explain that we would subtract, or "Change x by negative 100."

Change basketball's coordinates to (100,0), then (50,0), then (0,0).

Explain that a sprite's x coordinate can be any number on the line or axis, even less than 0.

Change basketball's coordinates to (-100, 0), (-200, 0).

Change basketball's coordinates back to (0,0)





Ask students what we would do if we wanted to move the basketball up the y-axis?

Change basketball's coordinates to (0, 100), explaining that we would "Change y by (100)"

If we wanted to move it back down, would we add or subtract from the y value?

Change basketball's coordinates to (0, 0), then (0, -100), (0, -200)



Pick Random

We'll use coordinates to make the hoop reappear at random locations.

Make sure Hoop is selected.

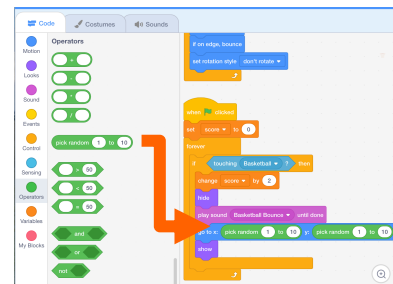
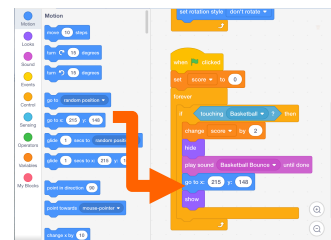
From **Motion**, add **Go to X () Y()** below **Play Sound Until Done**.

The prefilled values for x and y are based on where the hoop is right now. We'll change these to reset to a different random value each time the Hoop hides and then reappears.

From **Operators**, place a **Pick Random (1) to (10)** in each coordinate placeholder.

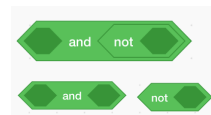
Looking at our X/Y Backdrop, we can see that the farthest left coordinate on the axis is -240 and the farthest right is 240, enter those x values:

go to x: pick random -240 to 240 y: pick random 0 to 180 For y, set the lowest value to 0. The highest can be 180.



Save and Test

The Hoop should bounce around the stage, disappearing when the basketball collides with it, but there's a little problem. If the Hoop goes over the cat holding the ball, it still disappears and the player gets points. Let's change that so the player has to shoot to score.



From **Operators**, drag **() And ()** & **Not ()** blocks into the Code area, but don't attach to anything.

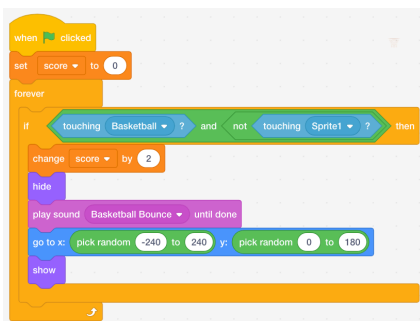
Place the **Not ()** in the right parameter of **And ()**



Move the **Touching Basketball** into the left **() And** parameter.

From **Sensing**, add a **Touching Sprite 1** to the **Not()** parameter.

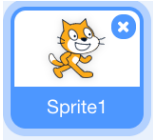
Click on the "and" to drag the whole thing into the **if () then** parameter



Save and Test. The Hoop should be able to pass over the cat holding the ball.

Step 6 Timer

*Skip this step if the class period is coming to an end.



Make sure that the cat (Sprite1) is selected.

Remember the other scripts will seem to hide, but they are still attached to their respective sprites.

From **Events** start a new script with **When Green Flag Clicked**

Go to **Variables**, and **Make a Variable** called **time** for All Sprites, click OK

Add Set (time) to (30) to the script

Students may make the game period shorter or longer, but we want to keep it fairly brief for testing purposes. They can add time later.

From **Control** add a **Forever** loop

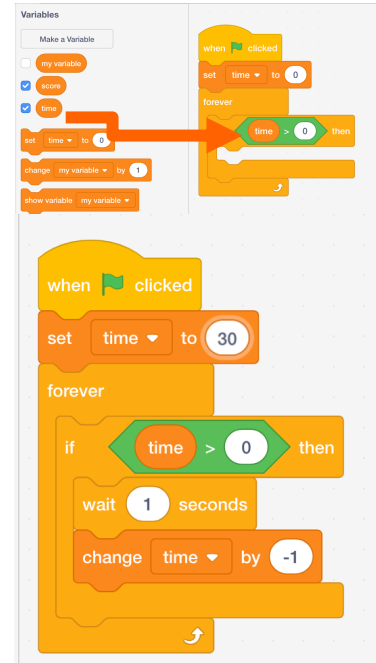
Place an **If () Then** inside of the Forever

We want to subtract 1 from time each second until time runs out, so from **Operators** place a greater than (**>**) between if () then.

From **Variables**, place **time** in the left side, and change the right side to **0**.

Place **Change (time) by (-1)** inside the if () then.

From **Control** place a **Wait (1) Second** above the change time block.



Save and Test

The timer should count down and stop at zero. The problem now is that players can keep playing once time runs out. Let's fix that!



Select the Hoop

Add an **if () then** inside the Forever

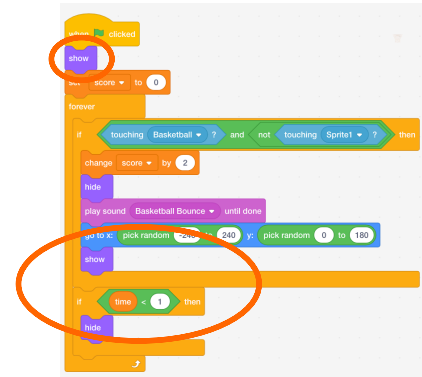
Place a **Less Than (<)** as the condition

Place **time** on the left side and **1** on the right

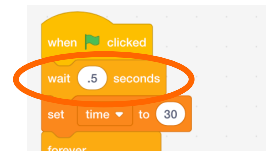
From **Looks**, place a **hide** inside the if () then.

This will make the Hoop disappear when time runs out, so players can no longer score.

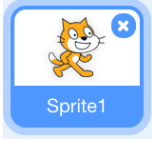
Remember to add a **show** under **When Green Flag Clicked** so that the hoop reappears for a new game.



If the time does not reset to 30 before the Hoop tests that $time > 1$, the Hoop will stay hidden. We can fix this by adding a **wait (.5) seconds** before the timer starts in the first script of the cat.

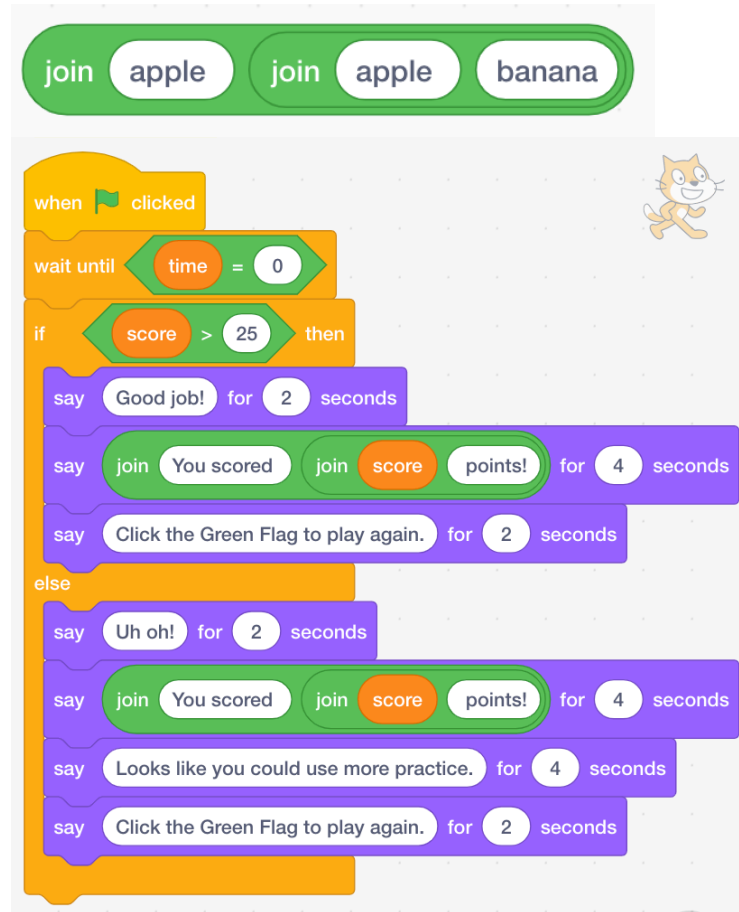


Customization Options



Add Win/Lose Message

It would be nice to let players know how they did! We can have the cat tell them. Make sure the cat is selected.

A Scratch script for a cat sprite. It starts with a green flag clicked event, followed by a wait until time = 0 block. Then an if-then-else conditional block. The 'if' block checks if score > 25. The 'then' branch has three 'say' blocks: "Good job!" for 2 seconds, "join You scored join score points!" for 4 seconds, and "Click the Green Flag to play again." for 2 seconds. The 'else' branch has four 'say' blocks: "Uh oh!" for 2 seconds, "join You scored join score points!" for 4 seconds, "Looks like you could use more practice." for 4 seconds, and "Click the Green Flag to play again." for 2 seconds. A small cat sprite icon is visible in the top right of the script area.

```
when green flag clicked
wait until time = 0
if score > 25 then
  say Good job! for 2 seconds
  say join You scored join score points! for 4 seconds
  say Click the Green Flag to play again. for 2 seconds
else
  say Uh oh! for 2 seconds
  say join You scored join score points! for 4 seconds
  say Looks like you could use more practice. for 4 seconds
  say Click the Green Flag to play again. for 2 seconds
```



Select Custom Backdrop