**Game Over!**

#

In this lesson, you’ll extend the monkey game from the last two weeks. You’ll make three improvements. First, the monkey will have a fixed amount of time to try to get more bananas than the thief; if the monkey has more bananas, he wins! If the monkey has fewer bananas, he loses! Second, when the monkey jumps and moves through the air, he won’t change costumes like he does when he is walking. Third, the monkey will get angry when the thief or bad bananas steal his.

You’ll learn about a lot of new Scratch blocks today. You’ll learn about the **broadcast message** and **When I receive message** control blocks. The broadcast message block is needed when one Sprite needs to tell another Sprite something (e.g., that the game is over or that it detected it is touching something else).

You’ll learn about the **if <question?> else** block. This block is has a second part to it compared to the **if** block; if the answer to the question is true, then the blocks in the first group are run; if the answer to the question is false, then the blocks in the second **else** group are run.

Finally, we’ll practice more with variables. Previously, we used variables to remember something that we wanted the user to be able to see, like their points or number of bananas they caught. Today, we will use variables to remember something that our program has to remember, but the user doesn’t directly care about.

This lesson has three steps:

1. Add a timer that is counting down every second. When the timer reaches zero, the game is over. If the monkey has more bananas than the thief, show a You Win background; otherwise, show a You Lose background.
2. If the monkey is jumping, don’t make the monkey change costumes as he moves to the left and right. Track when the monkey is jumping with a new variable.
3. Tell the monkey when his bananas are stolen. When this happens, make him show in some way that he is unhappy.

## Track how much time is left

We want the user to be able to play this game for 30 seconds. When the 30 seconds are over, we’ll figure out if the monkey won or lost.

There are different ways to track how much time has passed in Scratch. One easy way to do that is to create our own new variable that is decremented every second. How can we do that?

Remember to initialize the variable. Where should these scripts go? Since the timer is for the whole game, let’s have the Stage run these scripts.

We don’t want the loop to run forever, we only want it to run until time is out. How can we do this? With a **repeat until** loop. What is the question we want to ask to figure out when we should stop the loop? When **time left = 0.**

When the time left is zero (i.e., when the repeat loop is done), what do we want to have happen? We want to show a game over screen. Let’s draw two versions of the screen, one for when the player wins and one for when they lose.

Now, when the time left is zero, what do we want to ask? We want to ask who won. How do we find out who won?

We can use **if else statements** to find this out. The question we should ask is **Bananas < Stolen Bananas?** If the answer is true, what should we show? Switch the background to the one showing that the player lost. If the answer is false, what should we show? Switch to the background showing that the player won. This block should go in the **else** part of the **if else** question.

How does the game work now? It looks pretty good the first time we play, but to start everything out right, you’ll need to switch to the regular background when the player clicks the green flag. You can use the block **stop all scripts** at this point so the bananas don’t keep falling.

**Advanced:** You might not like that the monkey and bananas still show up. You can use broadcasts and receives as described below to fix this.

## Don’t walk while in air

The monkey looks a little silly walking when he is up in the air. Lets make it so he doesn’t switch costumes when he is jumping. But, how are we going to do this?

One script is making the monkey jump while another script is switching the costumes. How can the jumping script tell the walking script that the monkey is jumping? It can do this with a **variable**; let us call this variable **Jumping.** The jumping variable records for us that the monkey is jumping. Let’s make this variable not for all sprites, but for just the monkey, since it is only the monkey that is jumping and no one else cares.

We are going to set the Jumping variable to a special value when the monkey is jumping and to a different value when the monkey is not jumping. Computer scientists like to use the special value of 1 to mean something is True and 0 to mean that something is False. So, lets use 1 and 0. Where should we put the block **set Jumping to 1**? Where should we put the block **set Jumping to 0?** Don’t forget to initialize!

Then, the walking code can check the Jumping variable; if the Jumping variable is set to 1, then the walking scripts know the monkey is jumping and it should not switch the costumes. If Jumping is 0, then the monkey should switch the costumes. How can we do this? Use the **if block** and ask the question **Jumping = 0?**  Only if that is true should the monkey **switch costumes.**

Your code should work now just fine.

## Make the Monkey Sad

Right now, when the rotten bananas or the thief touches the monkey, the monkey doesn’t react in any way. Maybe you want the monkey to say something or change how it looks when these bad actions happen.

Right now, the rotten bananas and thief are asking if they are touching the monkey. You might think you should have the monkey ask the same type of question. Unfortunately, this doesn’t always work correctly. Sometimes, if you have two different Sprites ask the same question, one might answer True when the other answers False, even though you want them to always get the same answer. The two Sprites get different answers because they ask their questions at slightly different times: the banana might happen to ask the question when it is touching the monkey, but then the monkey might ask when it is NOT touching the banana. Weird things can happen in your program when different Sprites get different answers. (Computer scientists call this a **race condition** because there is a race condition in your program and depending upon who wins the race, you will get different results.)

So, instead of having both Sprites ask the same question, just one Sprite should ask the question and then it should tell the other Sprite the answer. A direct way to tell a Sprite something is to **broadcast** a message to them. Broadcasting a message is a lot like yelling something out so every Sprite can hear what you are saying. Some Sprites may choose to ignore your messages, but others will want to pay attention and do something special when they **receive** that message. You can send different signals with different names. Make sure you name your messages something descriptive.

In this program, lets have the rotten bananas **broadcast** the message **Sad Monkey** when they detect they are touching the monkey. (Note that you probably just want to **broadcast** without **broadcast and wait.**) Where should that block be inserted?

The monkey can now receive that message. You can start off a new script whenever the monkey hears that message with the **Event** block **When I receive Sad Monkey**. What should happen when the monkey receives this message? You can do more interesting things, but lets begin with just saying “Oh no” for 2 seconds.

Which other Sprites should broadcast this message? Lets have the thief broadcast this message too. Where should the block be inserted?

Test this out now. Does everything work as desired?

**Advanced:** If it bothers you that the Sprites all show when the Game is Over, you might want to tell the Sprites to all hide after switching to the Game Over backgrounds. You can do this by broadcasting a **Game Over** message and having every Sprite **hide** after **When I receive Game Over.** Note that you’ll need to make show the Sprites call **show** when the game starts (i.e., when the user **Clicks the Green Flag.)**

# Things to Remember

An **if** block asks a question and then executes some code **if and only if** the answer to the question was true. An **if else** block extends this so that it executes some other code when the answer to the question is false. In an **if else** block, one of the two groups of blocks will always be run.

A **variable** is a way for your program to remember something. A variable might hold a value that the user wants to see (e.g., the number of points they have) or it might not. Variables can record the answer to a question inside of your program: for example, is the monkey jumping in the air right now? Computer scientists like to use the value 1 to mean True and the value 0 to mean False.