Raspberry Pi Activity 3: Catch the Baby!

In this activity, you will design a simple game. You will need the following items:

- Raspberry Pi B v2 with power adapter;
- LCD touchscreen with power adapter and HDMI cable;
- Wireless keyboard and mouse with USB dongle;
- USB-powered speakers (optional);
- Wi-Fi USB dongle; and
- MicroSD card with NOOBS pre-installed.

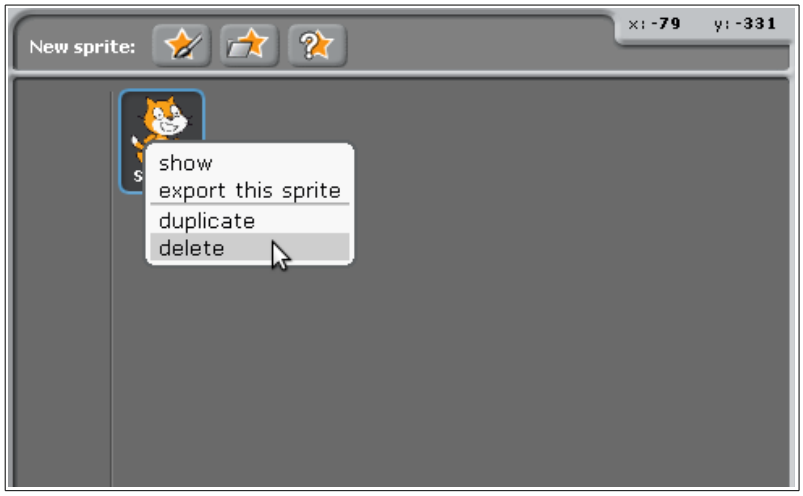
The goal of this activity is to take you through the process of making a simple game in Scratch. Various constructs will be utilized and discussed (e.g., selection, repetition).

Catch the Baby!

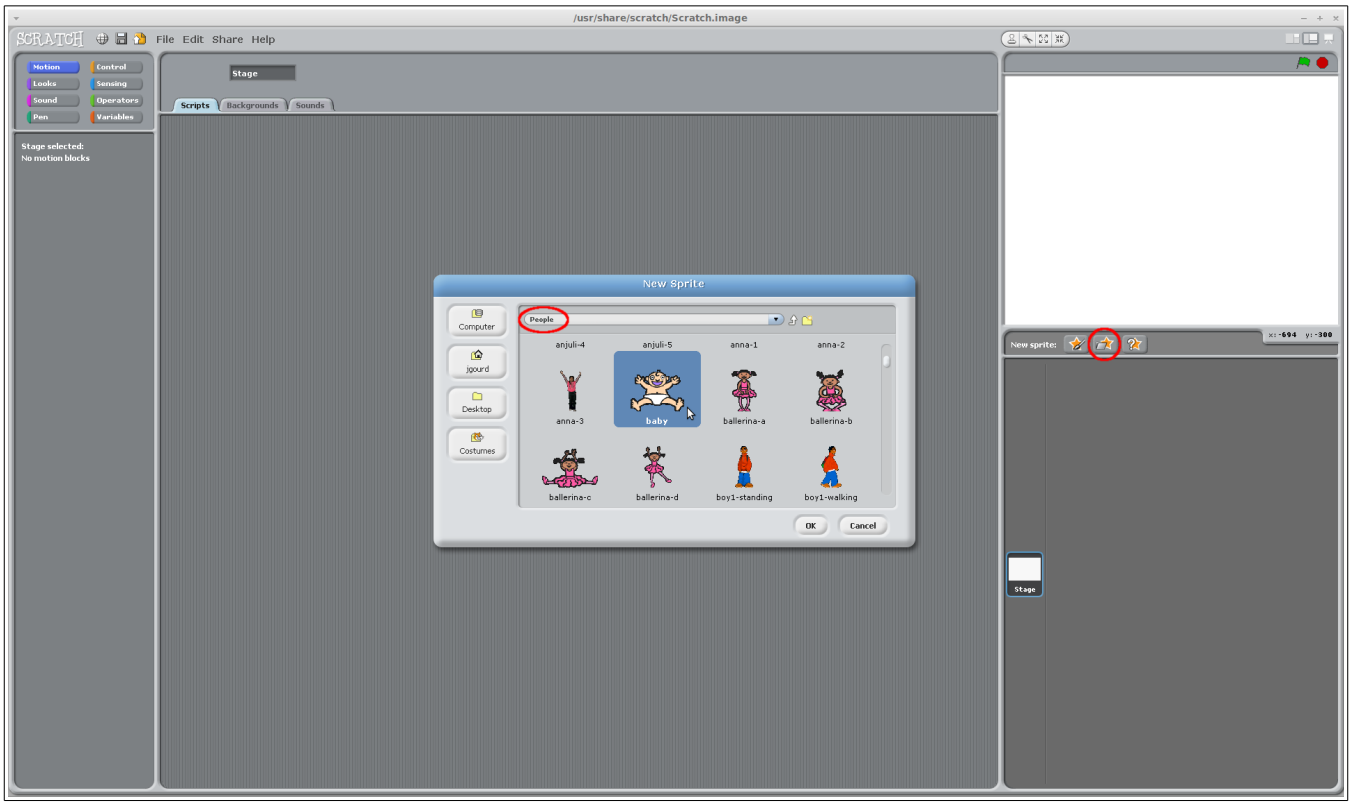
The objective of the game *Catch the Baby!* is, well, to use a trampoline to catch a falling baby. Each time the baby is successfully caught by the trampoline, the player's score increases. The baby is randomly placed somewhere at the top of the screen, and then quickly descends to the bottom. The player can control the trampoline with the left and right arrow keys to position it below the falling baby. Technically, the baby can barely touch the trampoline to be saved. Here is a screenshot of the stage at the start of play:

The game

Start a new Scratch project. Remove the default cat sprite by right-clicking on the cat and selecting **delete**:

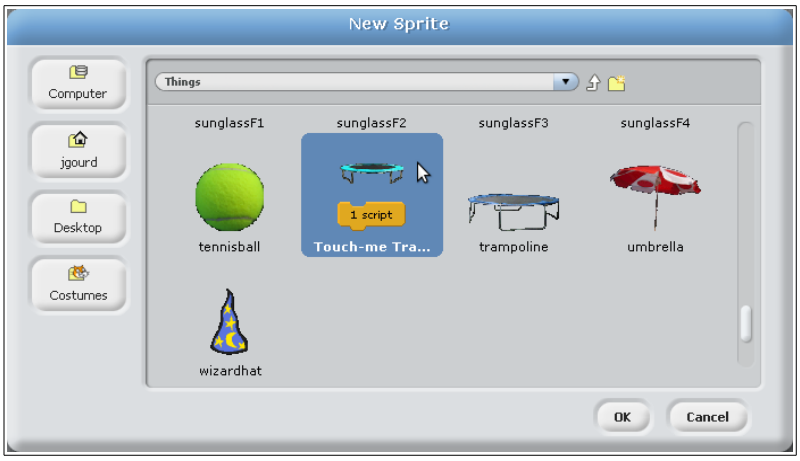


This is a good time to add the two sprites that you will need: the baby and the trampoline. To add the baby, click on the **choose new sprite from file icon**, then browse through the **People** folder and select the baby:



Rename the baby to something more appropriate, like **Baby**.

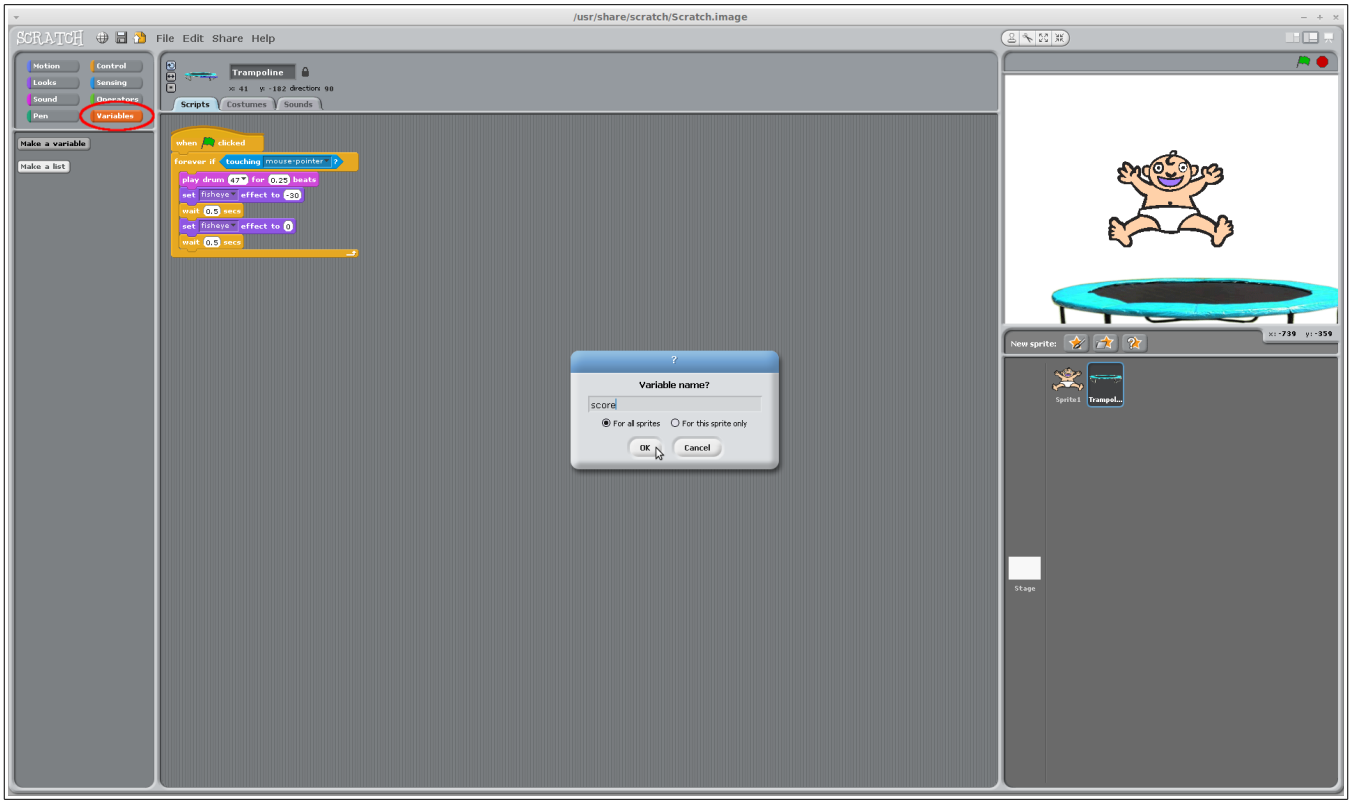
To add the trampoline, go through the same process, but browse through the **Things** folder and select the light blue trampoline:



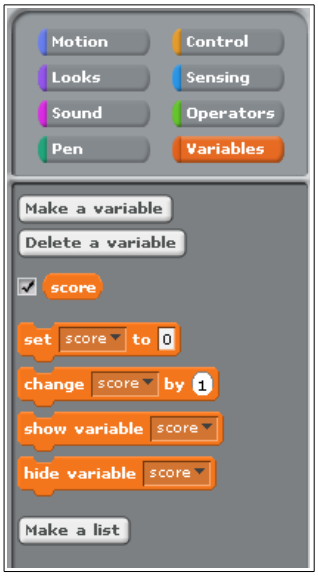
Note that the trampoline has a script associated with it. That is, it comes with a preloaded program that plays a short drum sound and changes the sprite if it is collide with. We will make use of some of this later. For now, resize the baby and trampoline sprites so that they are smaller (as in the image at the beginning of this activity).

Variables

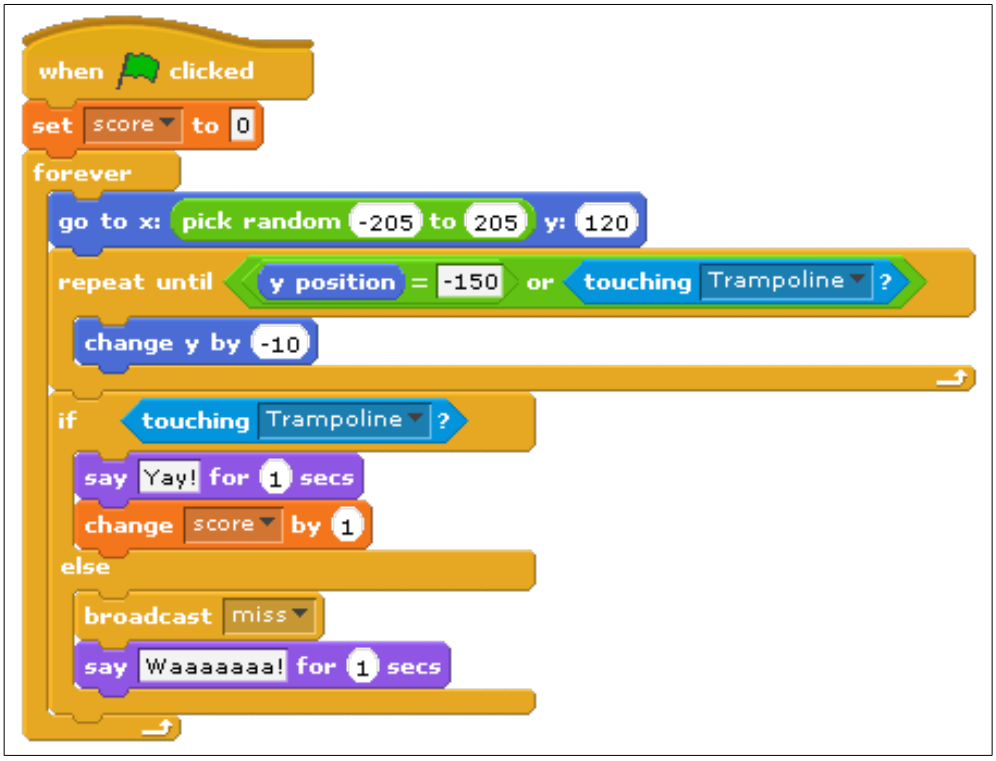
We will need one variable in our game: `score`. Add it now through the **variables** blocks group:



This adds the variable and allows us to modify it as we wish:



Make sure the baby sprite is selected in the sprites list and implement the following script for it:

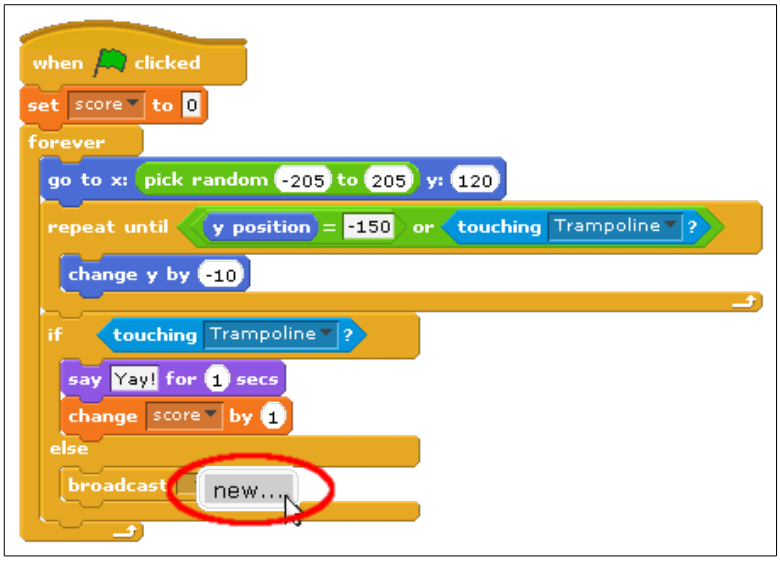


Let's explain what's going on here. This script runs when the green flag is clicked (i.e., when the game is started). The first statement sets the score to 0. Then, a group of statements is repeated forever (well, at least until the stop sign is clicked by the user).

The grouped statements in the **forever** construct first instruct the baby to move to a random position at the top of the screen (where y=120). By experimenting, it was calculated that the leftmost position for the baby should be at x=-205 and the rightmost at x=205.

At this point, a **repeat-until** construct is entered. Note that this is, in effect, repetition within repetition! The repeat-until condition instructs the baby to move down 10 pixels (**change y by -10**) until its is at position y=-150 or until it is touching the trampoline. In effect, it is instructing the baby to move down until it either collides with the trampoline or it reaches the bottom of the stage. Once either of these conditions occurs, the repeat-until construct is exited.

The next statement is a selection statement in the form of an **if-else** construct. The script is now going to potentially do two different things, depending on whether or not the baby has collided with the trampoline. If it has (i.e., **touching Trampoline**), then it will say, "Yay!" for a bit, and the score will be incremented (since the baby was successfully caught by the trampoline). Otherwise (**else**), it will cry. Note the **broadcast miss** statement. This is a useful way to send another sprite a message. Any sprite can broadcast a message that other sprites can receive. In this case, the goal is to notify the trampoline that it has missed the baby. Adding a broadcast message is as simple as adding the block and creating a new message using the block's arrow:



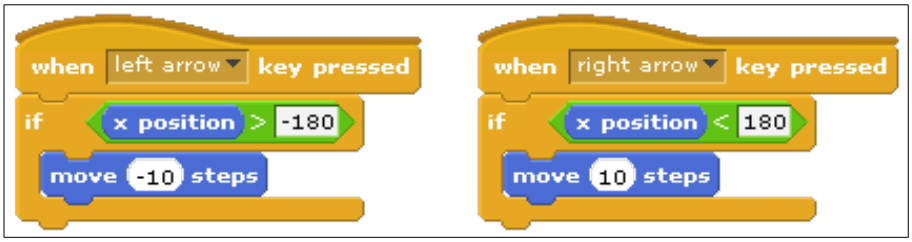
That's it for the baby! Now click on the trampoline in the sprites list and change the existing script to the following new script:

This change instructs the trampoline to move to the bottom-left corner of the stage when the green flag is clicked. It then repeats a set of statements forever, but only when it has collided with the baby (i.e., if **touching Baby**). If so, it first, it alters the trampoline sprite a little bit applying a *fisheye* filter (which makes the sprite appear to bend a little). A drum sound is then played, which is followed by a small delay, an undo of the *fisheye* filter, and another small delay.

Add another script to the trampoline as follows:

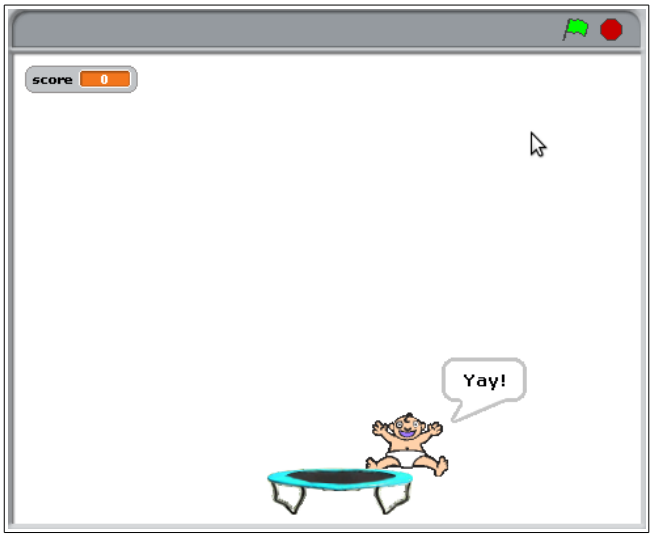
This script instructs the trampoline to say something appropriate when it receives the broadcasted message **miss**. Recall that this message was defined in the baby's script earlier. So the baby can broadcast the message which is then received by the trampoline. That is, if the baby reaches the bottom of the stage (i.e., the trampoline missed the baby), it broadcasts this message that the trampoline receives. This alerts the trampoline that it has missed the baby, and it utters an appropriate message.

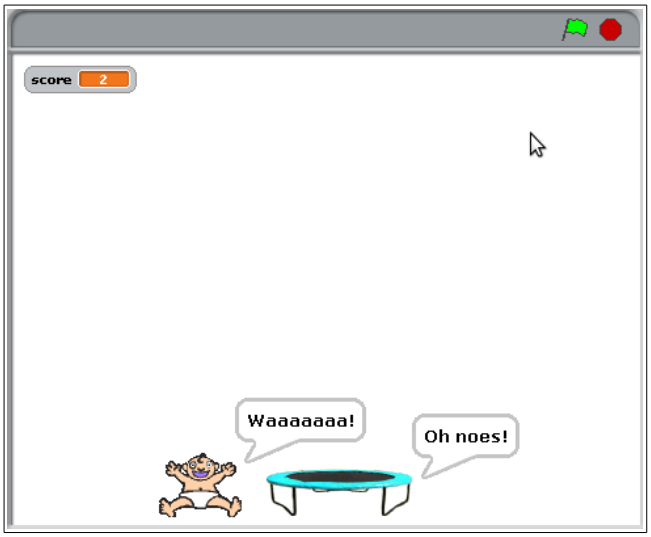
The last thing to add is the ability to move the trampoline with the left and right arrow keys. We can do this by adding the following two scripts:



These scripts instruct the trampoline to move 10 steps (to the left or right) when the arrow keys are pressed. In order to prevent the trampoline from going beyond the left or right border of the stage, however, additional **if** statements are added. These selection constructs prevent the trampoline from moving any further toward a border if it is already at one. For example, take a look at the left-arrow script. The left-most position for the trampoline is at x=-180. This was determined by testing (i.e., moving the trampoline with the mouse and capturing the x coordinate). The script checks to see if the trampoline's x-position is to the right of the established left-most position x=-180 (i.e., its **x-position > -180**). If so, it allows the sprite to move to the left; otherwise, it simply ignores the keypress.

At this point, you should be able to play the game by clicking on the green flag icon.





Click on the red stop sign icon to end the game.

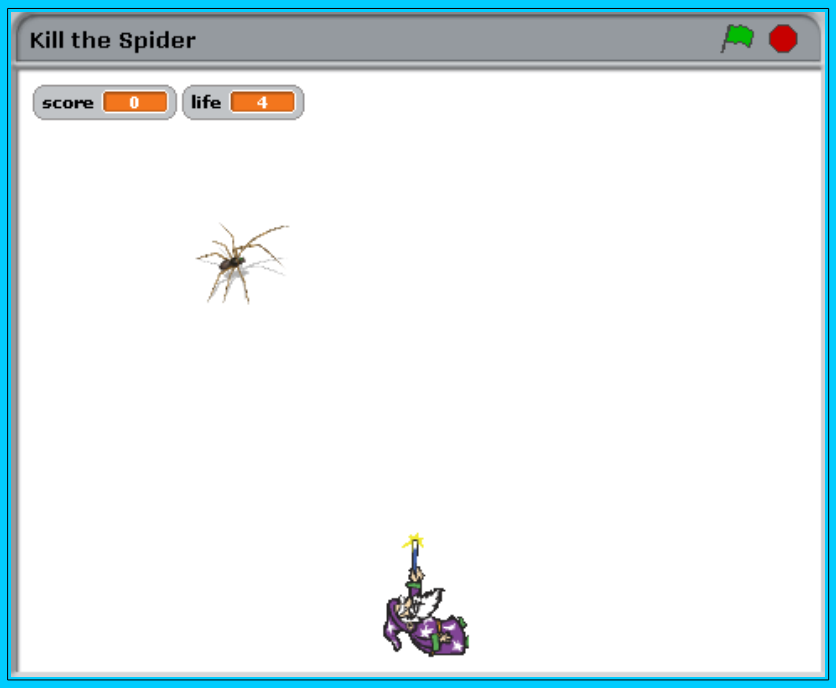
Improvements
Although the game ends here, improvements can be made. For example, the baby could bounce in a random direction once it collides with the trampoline. Maybe it can defy the laws of gravity and move from side-to-side as it falls down. Experiment a bit.

Homework: Kill the Spider!

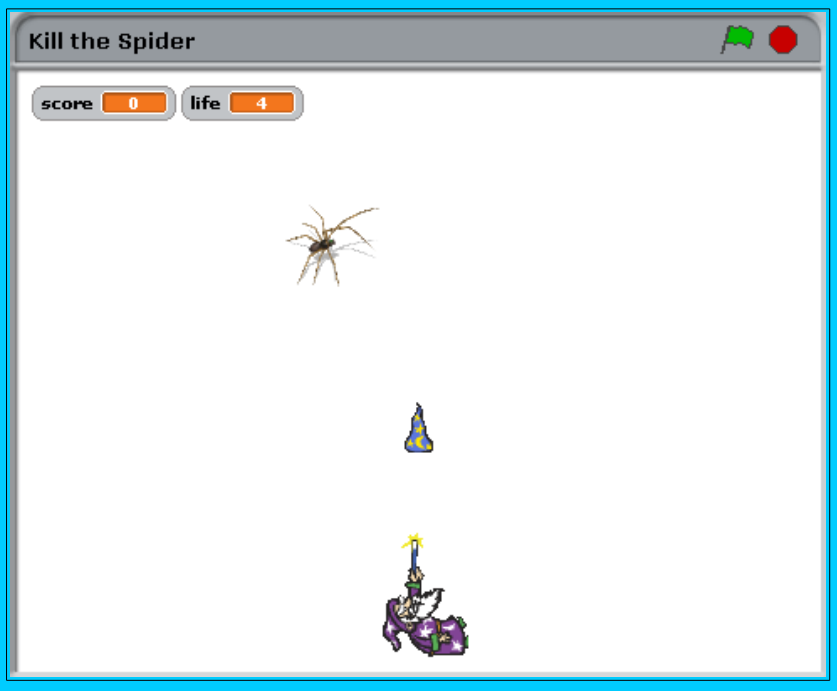
In this activity, you will design a game based on *Catch the Baby!* Of course, it will be a bit different. **You are to submit your Scratch v1.4 (not v2.0!) file (with a .sb extension) through the upload facility on the web site.**

This game has two characters: a wizard and a spider. The wizard stands at the bottom of the stage and can move horizontally (just like the trampoline). The spider is placed to the left of the stage. At the beginning of the game, the wizard has five lives and a score of 0. Like the baby, the spider moves around the stage; however, it only does so horizontally. It is initially placed to the left of the stage at some random vertical position above the wizard. It then moves to the right until it reaches the right side of the stage.

Spiders are scary and therefore should not be allowed to live. So the wizard can, of course, kill the spiders by shooting a wizard hat out of his wand. The player can make a hat shoot out of the wizard's wand by pressing the space bar. The hat starts at the wizard's wand and moves upward until it either collides with the spider or reaches the top of the stage. If the hat collides with the spider, the player's score increases by 1, and the spider reappears to the left of the stage at another random vertical position above the wizard for another round. If the spider is able to reach the right side of the stage, the wizard loses a life (since the spider was left to live and that is worthy of losing a life). The game ends when the wizard uses up all of his lives. Here's what the stage looks like during the game:



Here's what it looks like when the wizard shoots a hat:



And here's what it looks like when the wizard kills a spider:

Code your game in Scratch according to the description above. Feel free to add extra features or embellishments (although this is not a requirement). At minimum, your game should feature a wizard that can shoot a hat from his wand in order to kill spiders. The wizard should move via the left and right arrow keys, and should shoot a hat from his wand via the space bar. If the hat collides with a spider, the

spider should say something appropriate, another spider should reappear to the left of the stage, and the game should continue. Each time the wizard kills a spider, the score should increase by one. Each time a spider reaches the right of the stage, the wizard should lose one life. Start your game with five lives.

A note about the spider sprites: You can use your own spider image (e.g., a JPG or PNG). You can find some on the Internet through Google image search. Feel free to edit the image as you see fit. In Scratch, you can access the image and add it as a sprite the same way that you normally do to add sprites already in Scratch (i.e., via the **choose new sprite from file** icon). You will need to browse to the location where your spider image was saved. The spider sprite used in the example above is actually made up of 16 individual images so that it can appear to be crawling as it moves across the stage. The first image was selected as the sprite. The other 16 were added as separate costumes by selecting the costumes tab above the scripts area.