

SHALL WE LEARN  
SHALL WE LEARN

# Scratch Programming

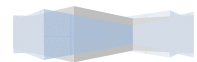
By Jessica Chiang

1

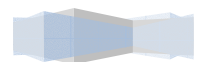


## Content Index

Introduction: What is Scratch? .....	4
Lesson 1: Introducing Scratch and Creating Sprite .....	5
Step 0: Learn What Can You Do with Scratch.....	5
Step 1: First Look at Scratch .....	6
Step 2: Create Your First Sprite .....	8
STEP 3 CREATE BACK, LEFT SIDE, and RIGHT SIDE VIEWS.....	11
Lesson 2: Animating a Sprite (Dance) .....	15
Step 1: Dance .....	15
Step 2: Finishing and Testing .....	18
Step 3: Kick Up a Notch.....	19
Lesson 3: Adding Music (Dance to the Beat).....	21
Step 1: Dancing and Popping .....	21
Step 2: Hip-Popping .....	22
Step 3: Wrapping it up.....	24
Lesson 4: Create a Music Sprite .....	25
Step 1: Create a Music Sprite .....	25
Step 2: Move Music Loop to the Music Sprite .....	28
Step 3: Turn Music On and Off.....	30
Lesson 5: Working with the Stage .....	33
Step 1: Create a Simple Scene .....	33
Step 2: Adding Simple Movement Scripts to Sprites .....	34
Step 3: Do the same for Donut Man and Donut .....	36
Step 4: Test.....	38
Lesson 6: Create Music with Music Tool Kit & Audacity .....	39
Step 1: Create a Tune using Sound Tool Kit.....	39
Step 2: Create a Sound Clip using Audacity .....	41
LESSON 7: TURN YOURSELF TO A SPRITE USING GIMP .....	48
Step 1: Get GIMP .....	48
Step 2: Load a Photo of Yourself to GIMP .....	49
Step 3: Select Yourself in the Photo Using Free Select Tool.....	50
Step 4: Delete the Background.....	53
Step 5: Save the Result Image .....	55
Step 6: Import Yourself to Scratch.....	57
LESSON 8: PROJECT – MANIKIN DANCE.....	61
Step 1: Create the Manikin Parts .....	61
Step 2: Connect Body Parts Together .....	63



Step 3: Create Multiple Dancing Postures.....	66
Step 4: Make It Dance .....	72
Lesson 9: The Pong Game .....	74
Step 1: Understand how the Pong Game works.....	75
Step 2: X-Y Coordinate System and Rotation in Scratch .....	78
Step 3: Modify the Pong Game .....	80
LESSON 10: STORIES TO ANIMATION PART I.....	82
Step 1: Create a Story Line .....	82
Step 2: Create Sprites .....	83
Step 3: Creating the Village Background .....	84
LESSON 11: STORIES TO ANIMATIONS PART II.....	101
Step 1: Get to Know Your Stage .....	101
Step 2: Put Storyboard Together .....	103
Step 3: Create Scene Transition Using Broadcast Messages.....	105
LESSON 12: SCROLLING PLATFORM GAME – Game Design.....	117
LESSON 13: SCROLLING PLATFORM GAME – SPRITES .....	123
LESSON 14: SCROLLING PLATFORM GAME – GAME RULES .....	135
LESSON 15: Scrolling Platform Game – Platforms.....	144
Lesson 16: Mini Mario Game Part 5 – Scrolling Intro .....	158
Lesson 17: Platform Game Wrapup .....	167
Step 1: Create a Variable to Represent Scrolling Amount.....	167
Step 2: Make Platforms Scroll.....	168
Step 3: Modify the <b>Brick</b> and <b>CoinToPass</b> to Stop Them from Following <b>Mario</b> .....	171
Step 4: Upgrade the <b>Bullet</b> and Turn <b>Brick</b> to a Mini Platform .....	174
CONCLUSION .....	177



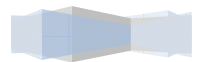
## Introduction: What is Scratch?



Scratch is developed by the [Lifelong Kindergarten group](#) at the [MIT Media Lab](#). Here is how Scratch [website](#) describes Scratch.

*"Scratch is a new programming language that makes it easy to create your own interactive stories, animations, games, music, and art -- and share your creations on the web."*

I started to use Scratch about one and half year ago and am amazed at how well it's designed. I've been a professional programmer for more than 8 years and started to use Scratch to teach my 9-year-old boy and his classmates to program. I also used it to create small educational software for kids. I am amazed at how well Scratch is designed and how fun it is to use. With this series, I aim to guide you, step-by-step, to go from creating animating Sprites (Lesson 1 to Lesson 4), the Stage (Lesson5), using Scratch multimedia tools(Lesson 6 and Lesson 7), creating 2D animations (Lesson8 to Lesson9), then finally to creating a scrolling platform Scratch Game (Lesson 10 to Lesson15). To showcase your creation, I will show you how to share your projects, both on Scratch site and on your own Google Site to display (Lesson 16).





## Lesson 1: Introducing Scratch and Creating Sprite

Have you heard of Scratch? No, not what you do to your itch, but Scratch from MIT the famous school for the curiously brainy people? If you have not heard, seen, or played with Scratch, then you've been missing out. Because it is a log of FUN!!!

*Step 0: Learn What Can You Do with Scratch*

Scratch is a programming language for all, even for kids. In fact, Scratch, unlike all other programming languages, is designed first and foremost for kids. Because it's designed for kids, it's very easy to learn and use. They can create animations like never before. For older kids or teens, they can create single-level or multi-levels Scratch games.

But Scratch is not just for kids or teens. Teachers and adults can use Scratch to create effective education tools such as math quiz, physics simulation, and educational videos.

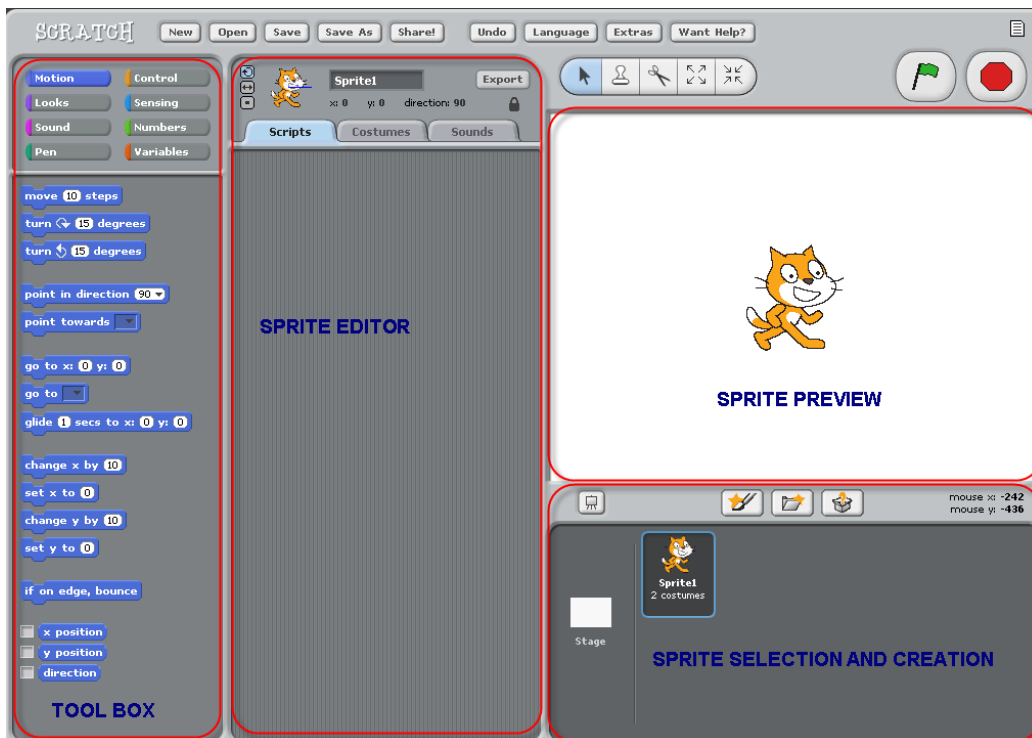
Since I knew about Scratch from a coworker, I have been using Scratch, teaching Scratch, and now writing on Scratch. As you can tell, I just cannot get enough of Scratch. I have two school-age boys and I've been looking for ways to quickly create games and animations to help them learn. Scratch is what I've been looking for and more. It's just a tool so awesome, so fun, and so easy to use and master, that I am sure, once you start, you will be just like me: Can't Get Enough!

5

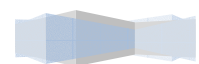
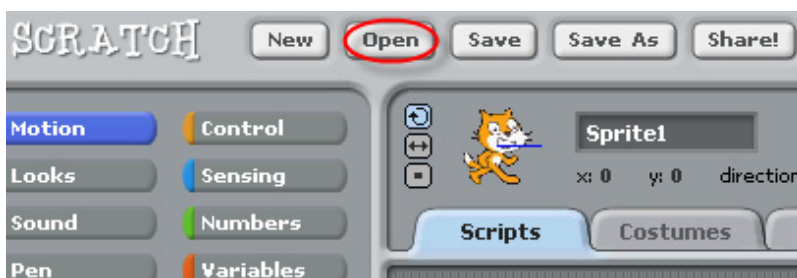
Without further ado, let's start using Scratch!

*Step 1: First Look at Scratch*

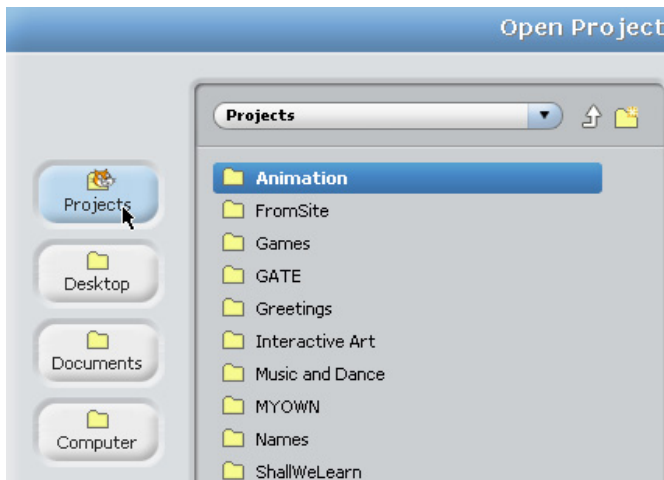
Start Scratch Program. Sprite Preview lets you preview a project. Sprite Selection and Creation lets you select an existing Sprite or create a new Sprite. Sprite Editor lets you edit a Sprite's Scripts, Costumes, and Sounds. Tool Box is like a bucket of Lego; it provides single script blocks for building combo script blocks.



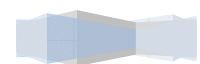
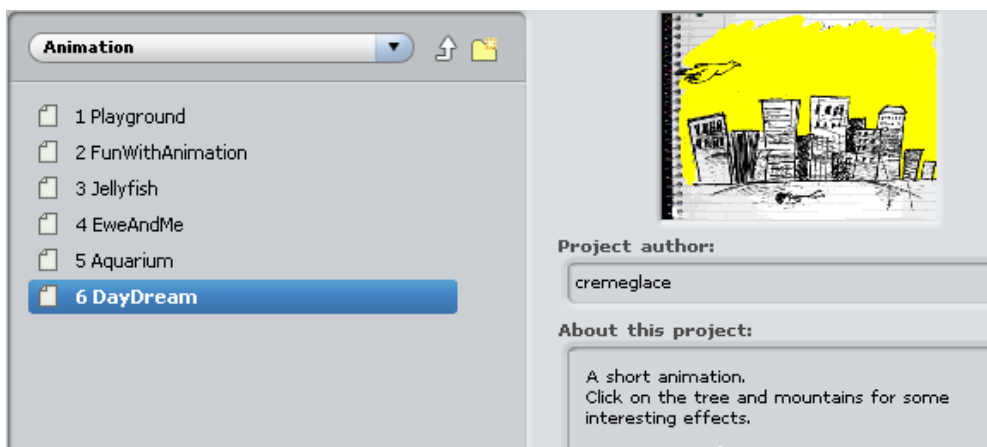
Let's take a look at available sample projects. Click "Open" on the top of the Scratch window.




If "Projects" not already selected, click "Projects" to go to the default Scratch projects directory. You'll want to try "Animation" and "Games".















My favorite from the sample projects are "Daydream" from Animation folder. Try opening it. I am sure you will like it too.

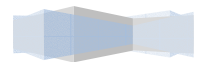





*Step 2: Create Your First Sprite*


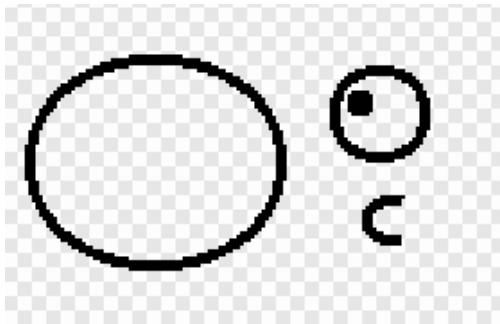
Now let's create our first sprite. Delete the cat sprite by right click on "sprite1" and select "Delete". Then click the  ("Create new sprite" button) to open the Paint Editor. I created this sprite using these tools:


 Paintbrush	 Line Tool
 Eraser	 Text Tool
 Fill Tool	 Select Tool \
 Rectangle Tool	 Stamp Tool
 Ellipse Tool	 Eyedropper

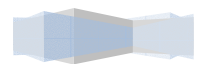
Click Ellipse Tool () and click hollow mode (). Draw three circles.




Click Ellipse Tool () and click solid mode ( 

); draw a little dark solid circle inside the medium circle, which will be the eye. Use Eraser tool () to trim the smallest circle; this will be the ear.

Click Stamp Tool () button and select the eyeball to copy. Drag the eyeball copy to where you want the new eyeball to be. Do the same for the ear to make two ears.




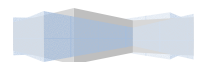
Use Fill Tool () to fill the face and the eye ball.





Click Select Tool () button and select both eyes; move them to the face.





Click Select Tool () button and select left ear; drag it to its place.



Click Select Tool (  ) button and select the right ear; click  to flip it. Then drag the right ear to its place.



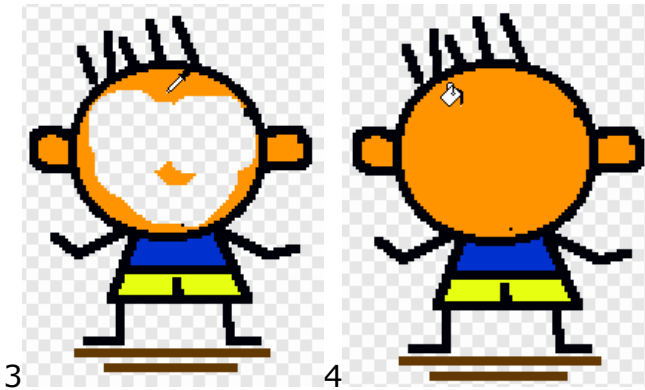
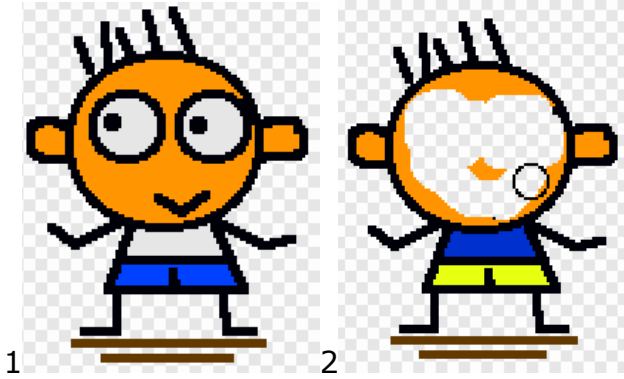
Click Eyedropper Tool (  ) and click the face to copy the face color. Click Fill Tool (  ) and fill both ears with the face color.



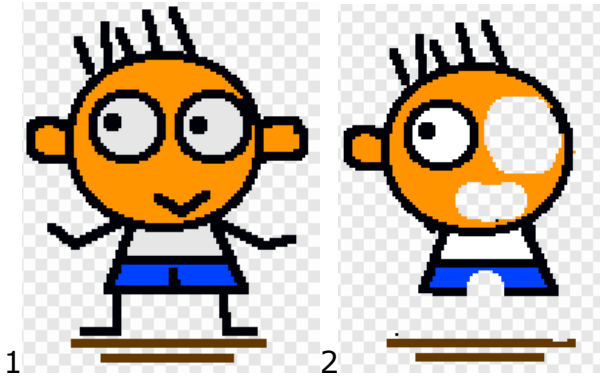
Once you are happy with your own Virtual Me, click OK to save. Rename the costume to "front". This is the front view.

### *STEP 3 CREATE BACK, LEFT SIDE, and RIGHT SIDE VIEWS*

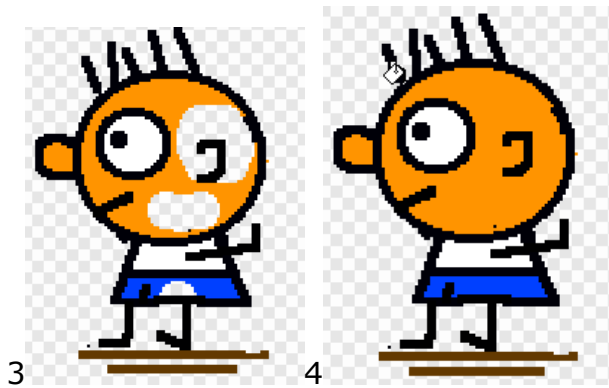
To create back view, make a copy of costume "front" (click "Copy" button next to costume "front"). Use Erase tool to erase eyes and mouth. Use Eyedropper tool to copy the face color. Then use Fill Tool to paste the color in the empty area. Click OK to save. Rename this costume as "back".




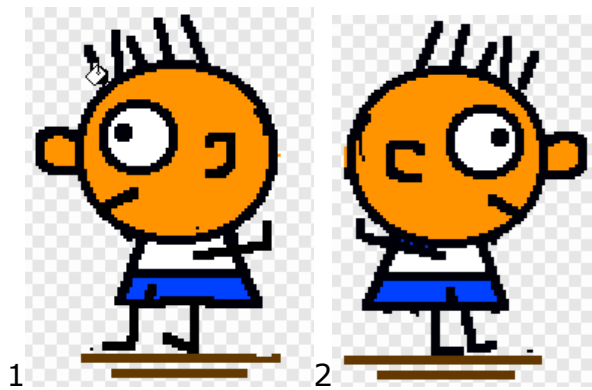
Then copy the costume "back". Erase extra body parts. Redraw body parts. Refill color of the face and the pants using Eyedropper Tool and Fill Tool. Click OK to save and rename this costume as "facing left"



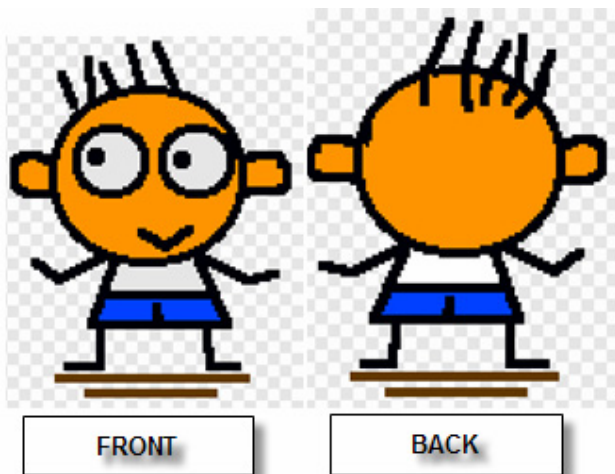


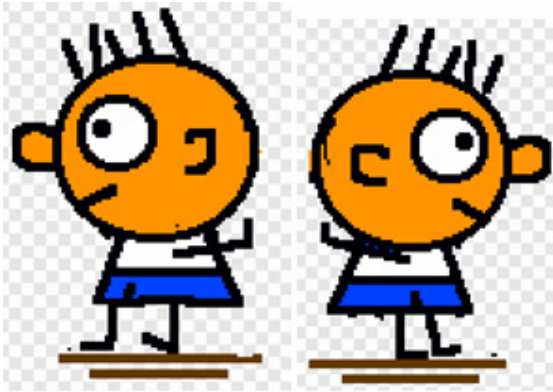


Copy the costume "facing left" and click  to flip the figure horizontally. Click OK to save and rename the costume as "facing right".



We've just created four costumes for the same sprite: "front", "back", "facing left" and "facing right".





FACING LEFT

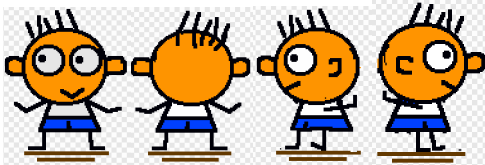
FACING RIGHT

THIS CONCLUDES TODAY'S LESSON. IN NEXT LESSON WE WILL MAKE HIM DANCE.



## Lesson 2: Animating a Sprite (Dance)

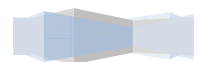
In Lesson 1, we created a sprite and also created four costumes: "front", "back", "facing left", and "facing right".



In this lesson, we will make our sprite dance.

### *Step 1: Dance*

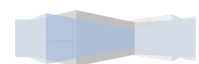
Open the "dance" project that we created in Lesson 1. Click "Sprite1" and change its name to "dancer1".



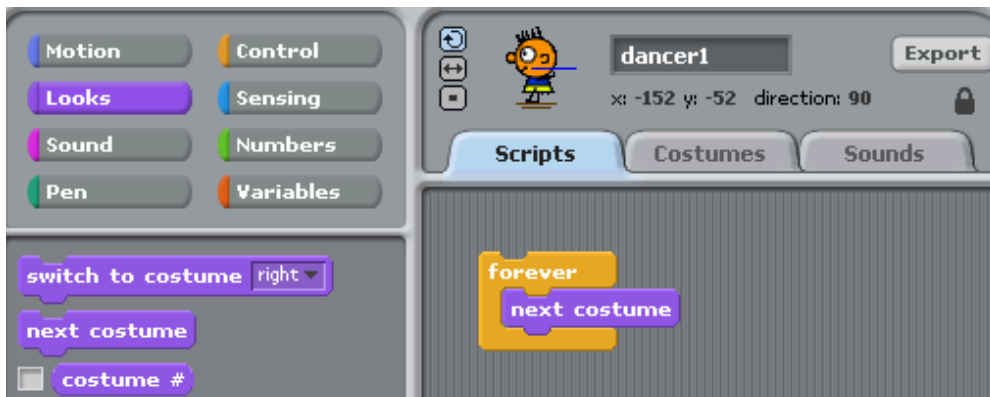
Click "Scripts" tab. Sprite "dancer1" doesn't have any script blocks. Click "Control".



Then drag out "forever" script block and drop it in the script editor.



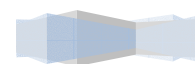
Click "Looks". Drag out "next costume" block and drop it inside "forever" block to form a combo block.



Now double click this combo block to see dancer1 spin like crazy!





We need to slow him down! To do so, click "Control" again. Drag "wait 1 secs" and drop it under "next costume". Now double click the combo block again.




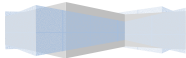


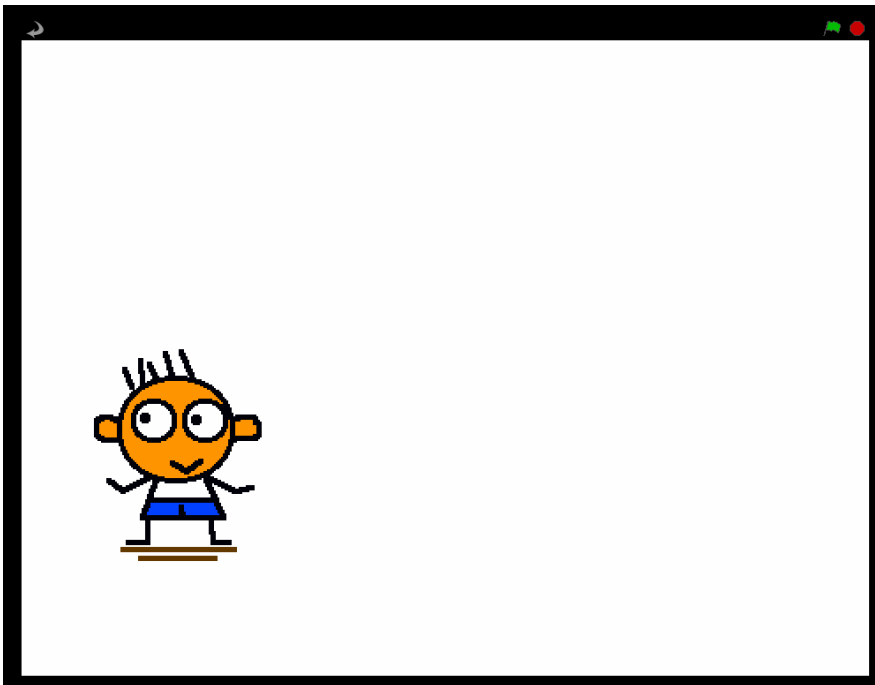
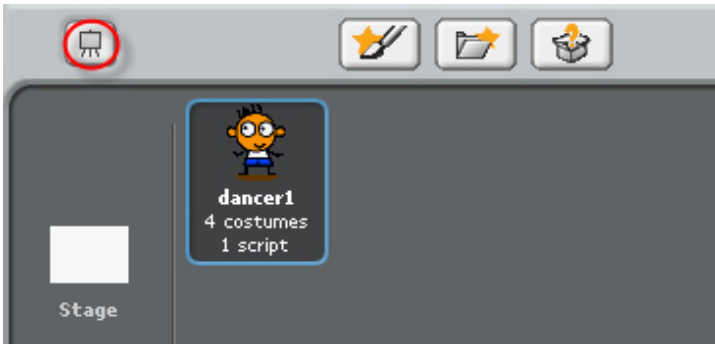
*Step 2: Finishing and Testing*

We are almost done with creating our first animation. To wrap it up, drag “when  clicked” and drop it above “forever” block. Now click the  button to start the animation. switch



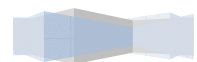
We are done with the first animation. Try viewing in full-screen mode or “presentation mode” by clicking .

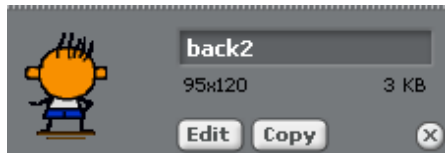




### *Step 3: Kick Up a Notch*

You can add more costumes or make dancer1 dance faster. I added two extra costumes: "front2" and "back2". I also change "wait 1 secs" to "wait 0.5 secs" to make dancer1 dance faster.








## Lesson 3: Adding Music (Dance to the Beat)

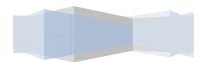
In this lesson 2, we make our sprite dance. In this lesson, I will show you how to make him dance to the beat. And not just to the beat but to the Hip-Pop beat!

### *Step 1: Dancing and Popping*

Open project "dance". Select sprite "dancer1" and click "Sound" button.



Drag "play sound pop" block and drop it under "next costume" block. Now click  to play.





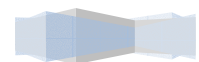
### Step 2: Hip-Popping

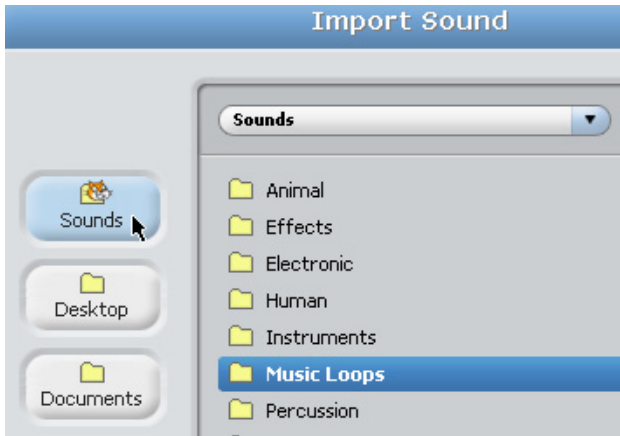
Fun, huh? But try listening to it for 5 minutes – it gets a bit boring. One can only take this much “popping”. This calls for a twist to the music clip. Let’s fancy it up!

Go to Sounds Tab, and click “Import” button. When “Import Sound” window opens, click “Sounds” button.

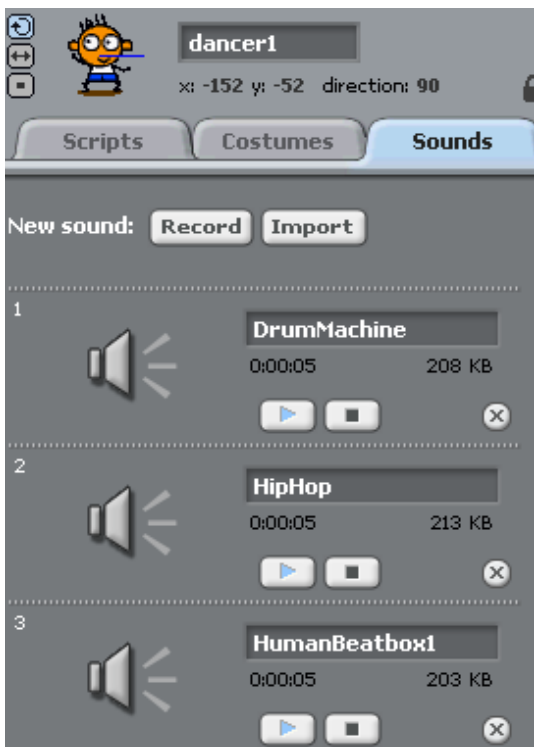


Select “Music Loops” folder.





Click each clip to listen to the demo. "DrumMachine", "HipHop", and "HumanBeatbox1" are among my favorites. So I think they are cool enough for our first loop. Import all three. Delete the less interesting "Pop". Now the Sounds editor for sprite "hip-pop" should now look like the snapshot below.




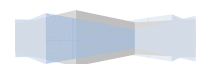
Go back to Scripts Tab. First drag a **forever** block from Control Tool Box and drop it anywhere in the editing box. Then drag a **play sound \_\_\_\_\_ until done** block and drop it inside the **forever** block. Drag two other **play sound**

\_\_\_\_\_ **until done** blocks and drop it under first **play sound** \_\_\_\_\_ **done** block. Select sound for each block to your liking. Double click this combo block to test.



*Step 3: Wrapping it up*


Remove the "play sound pop" block from the motion combo block. Then add a "when  clicked" on top of the music combo block.

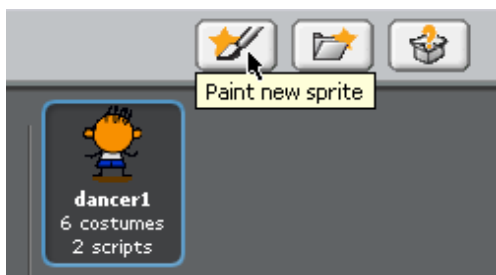


## Lesson 4: Create a Music Sprite

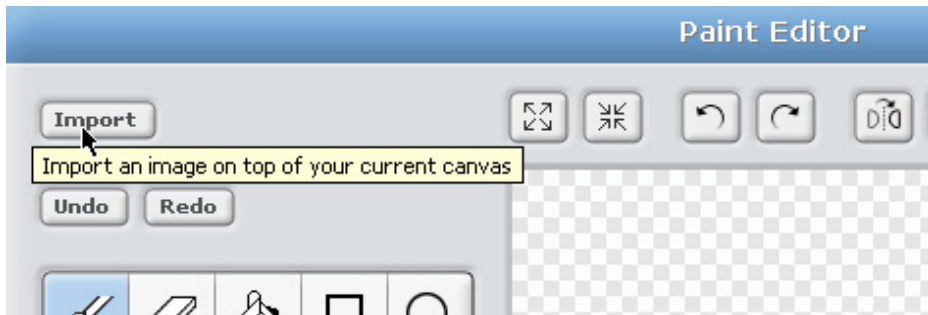
In this lesson, I will show you how two sprites can communicate with each other via variables. We will use the dancer sprite created from Lesson 3 but extract its music playing scripts to a new sprite call Hip-Pop.

### *Step 1: Create a Music Sprite*

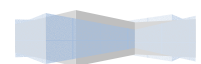
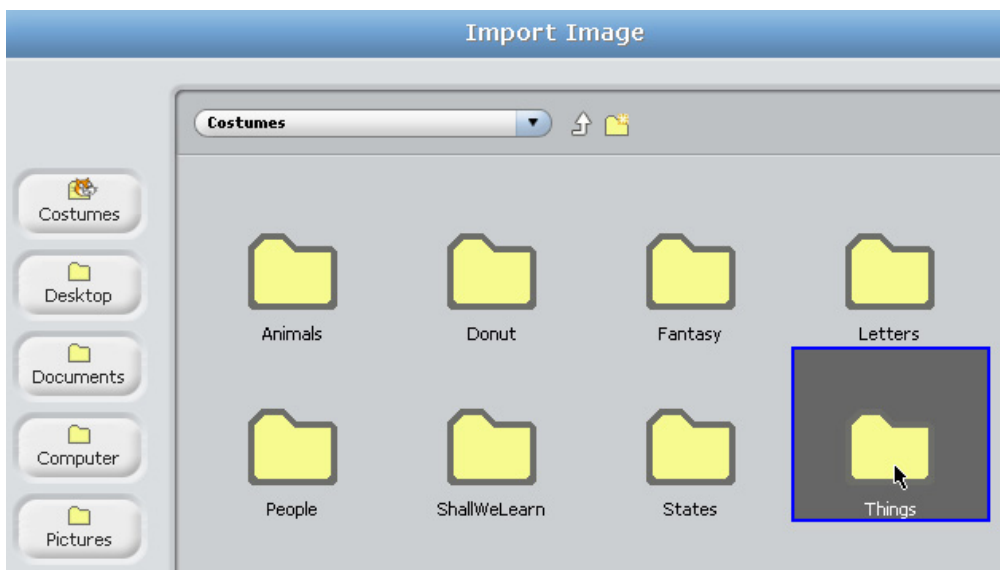
Let's create a Music Sprite or a sprite that controls playing and stopping of background music. To create a new sprite, click  to open Paint Editor.



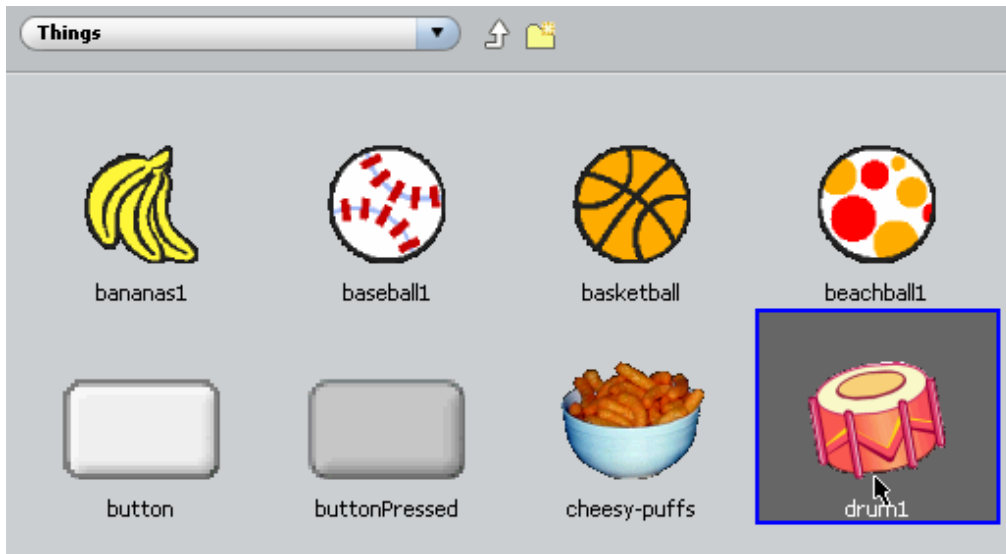
Instead of drawing our own, this time we will import an image. Click "Import".



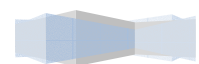
Click "Costumes" if it's not already selected. Then open "Things" folders.



Select "drum1" or whatever image you like. Then click "OK" to save.



Rename this new sprite to "hip-pop". It's seems big for me so I shrink it with "Shrink Sprite" tool.



Then use "Move" tool to move the "hip-pop" sprite to wherever you see fit.

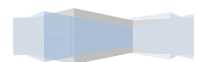


I place it next to dancer1's feet.



*Step 2: Move Music Loop to the Music Sprite*

Drag the Music Combo Block from dancer1's Scripts Tab.





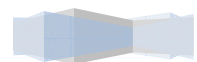
Drop it to be over sprite "hip-pop" icon. Wait until the "hip-pop" sprite is highlighted before releasing the mouse.



Sprite "hip-pop" now has the music combo block from "dancer1".



Delete this same combo block from "dancer1" so there will not be echoing.





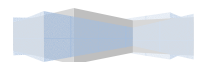
### *Step 3: Turn Music On and Off*

To turn music on and off, we need to create a virtual switch. Like a real switch, a virtual switch can be on or off. To create a virtual switch, we use Variables. Select "Variables" Tool Box, and click "Make a variable" button. Then enter "switch" for Variable name.





I would like this switch to work this way:



When the animation first starts, the music is playing and the switch is on. Click the "dancer1" sprite to turn it OFF, and click the "hip-pop" sprite to turn it ON again. If the switch is ON, music is ON; if the switch is OFF, music is OFF.

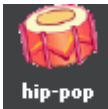




For this switch to work, we need following blocks:

one  block from  tool kit

one  block from  tool kit

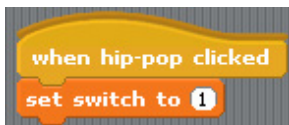
one  block from  tool kit



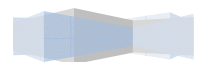
For  sprite, we will add three new combo blocks. The first combo block turns the switch ON (set music on to 1) when  is clicked.



The second combo block turns the switch ON (set music on to 1) when "hip-pop" is clicked.



Finally, the third combo block keeps checking if the switch is OFF (switch = 0). If OFF, then stops all sounds.





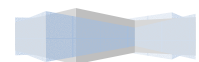
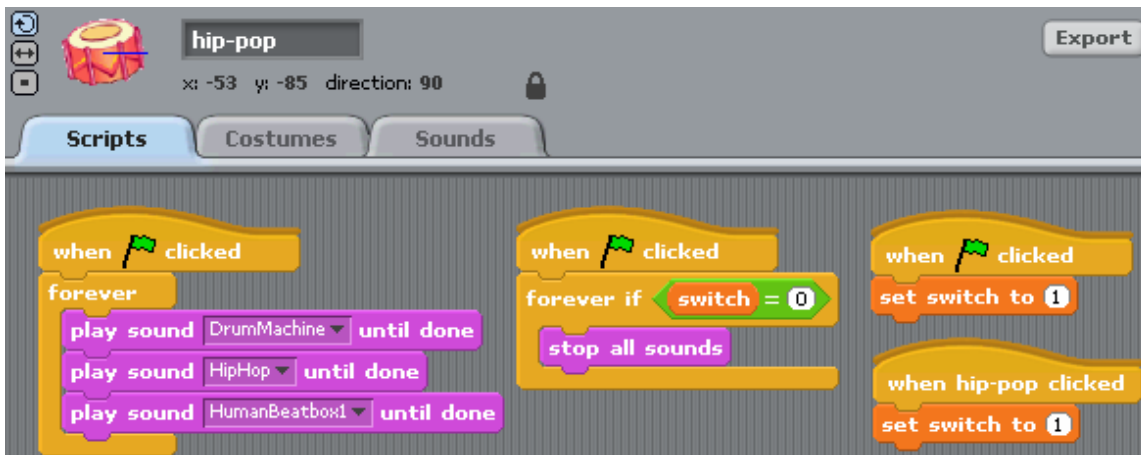
For `dancer1` sprite, we will add one combo block. This combo block will turns the switch OFF if “dancer1” is clicked.



The Scripts tab of your “dancer1” sprite should look like this.



The Scripts tab of your “hip-pop” sprite should look like this.



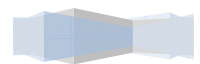
## Lesson 5: Working with the Stage

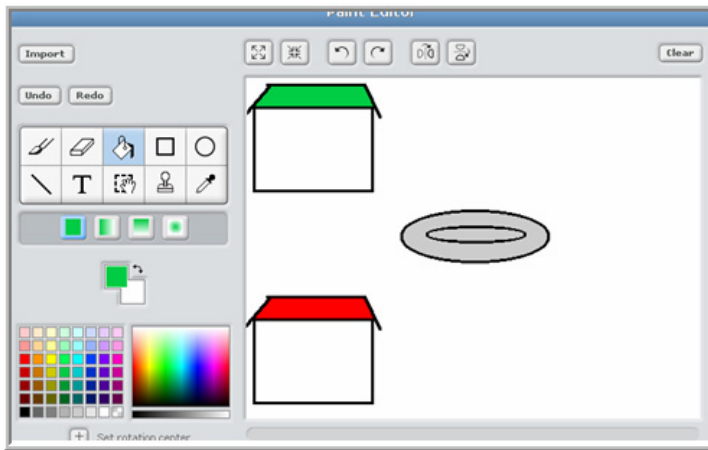
A scene, like a movie scene, sets the tone of the story and provides a platform for the characters in the story.

### *Step 1: Create a Simple Scene*

To create a scene, we change the costume of "Stage". We will add two houses and a plate to the "Stage". To do so, double click the Stage icon to select. Then click "Costumes" and select the only costume, "background1" and hit "Edit" button.

Change the costume by adding two houses and a plate. When done, hit "OK" button.





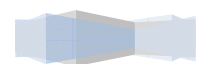
### Step 2: Adding Simple Movement Scripts to Sprites

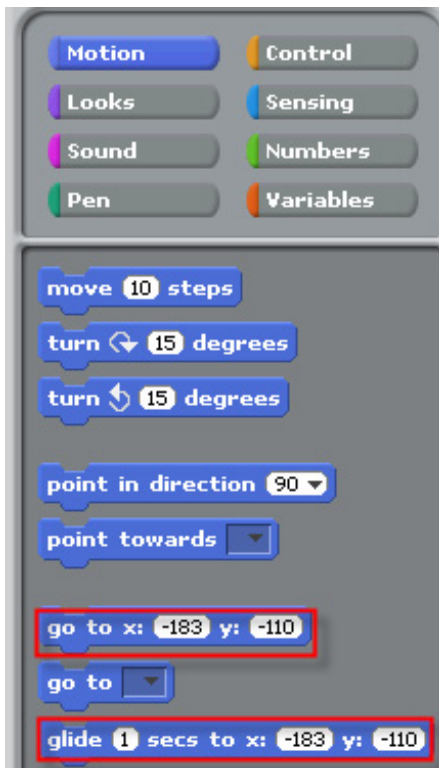
If you have not already, create three sprites: Mr. Meow, Donut Man, and Donut. Now let's add scripts so that all sprites will show up in the right places when the animation starts (when the Start Flag is clicked).

Move Mr. Meow around on the stage and its x-axis and y-axis values would be updated accordingly.



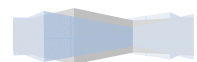
A cool feature is that when sprites are moved, the Motion blocks in the Tool Box are updated as well.





To build Mr. Meow's scripts, select Mr. Meow from the Sprites Area, then click "Scripts" tab.

From the Tool Box, click the "Control" button and drag "when Start Flag clicked" block to drop it in Scripts work area.



Then click "Motion" button in Tool Box Selection and drag "glide 1 secs to x:-183 y:-110" to the Stage. Then attach it under the "when Start Flag clicked" block.

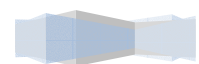


Then click "Looks" button in Tool Box Selection, and drag "switch to costume1" to the Stage. Then attach it under the "glide 1 secs to x:-183 y:-110".

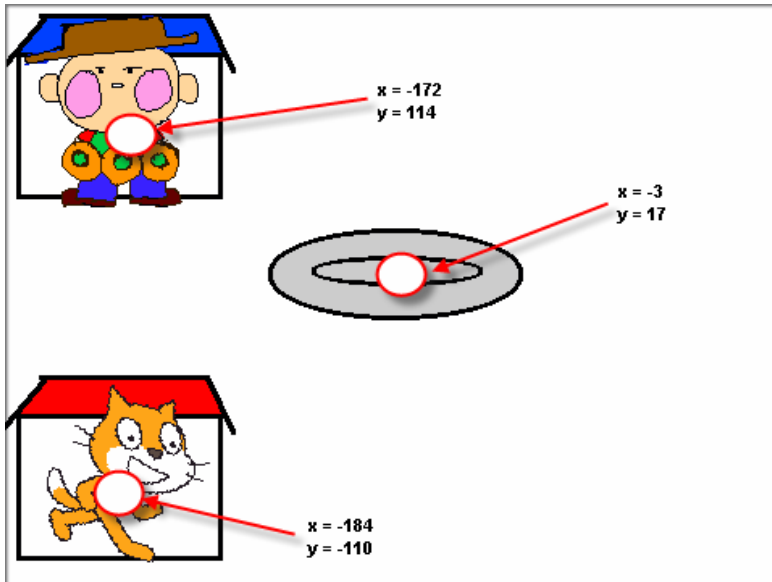


*Step 3: Do the same for Donut Man and Donut*

In the same way, create the following script for the Donut Man and the Donut. Drag each sprite to a spot you like. I dragged my sprites as shown in the snapshot below but yours will probably be a bit different than mine. It's OK. Or you can enter the x-y values by hand so they are exactly the same as mine.



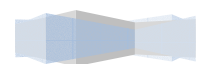




Below is what result scripts should look like. Again, your x-axis and y-axis values might be different than mine - it's OK. Mr. Meow will glide to x:-184 y:-110 after 1 second and then switch to costume "costume1".



Donut Man will glide to x:-172 y:114 after 1 second and then switch to costume "regular".




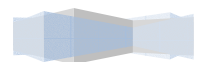


Donut will glide to x:-3 y:17 after 1 second and then just show.



#### Step 4: Test

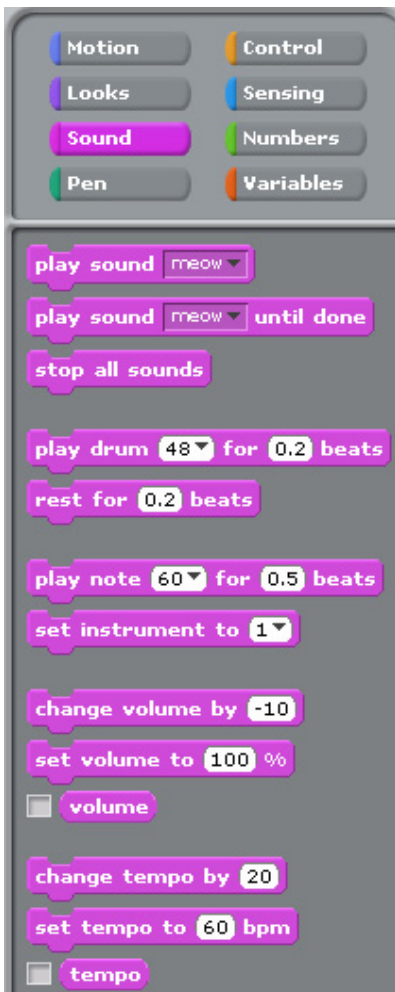
TEST: Move sprites around in the Stage and clicking  to see all of the sprites gliding back to their positions.



## Lesson 6: Create Music with Music Tool Kit & Audacity

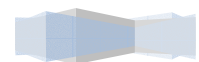
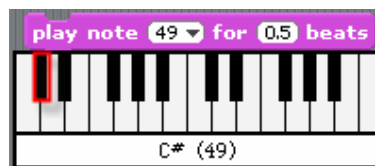
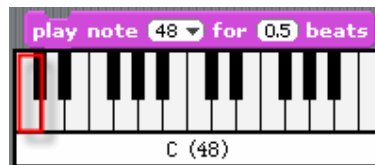
In lesson 6, I will show you how to create a tune using Scratch's Sound Tool Kit. Moreover, I will also show you how to use Audacity® to create a Scratch sound clip from a MP3 or WAV file.

*Step 1: Create a Tune using Sound Tool Kit*




To add a music script to a sprite, select a sprite and click Sounds Tab.

Drag out a "play note ? for ? beats". Click the down arrow to bring up the keyboard.



To create a simple tune that goes Do-Re-Mi, make a script combo block like the one below. DOUBLE CLICK this block to test it.

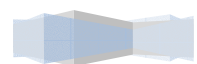
```
play note 48 for 0.5 beats
play note 50 for 0.5 beats
play note 52 for 0.5 beats
```

To add pause between notes, use the  block.

```
play note 48 for 0.5 beats
rest for 0.2 beats
play note 50 for 0.5 beats
rest for 0.2 beats
play note 52 for 0.5 beats
```

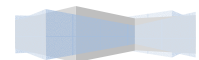
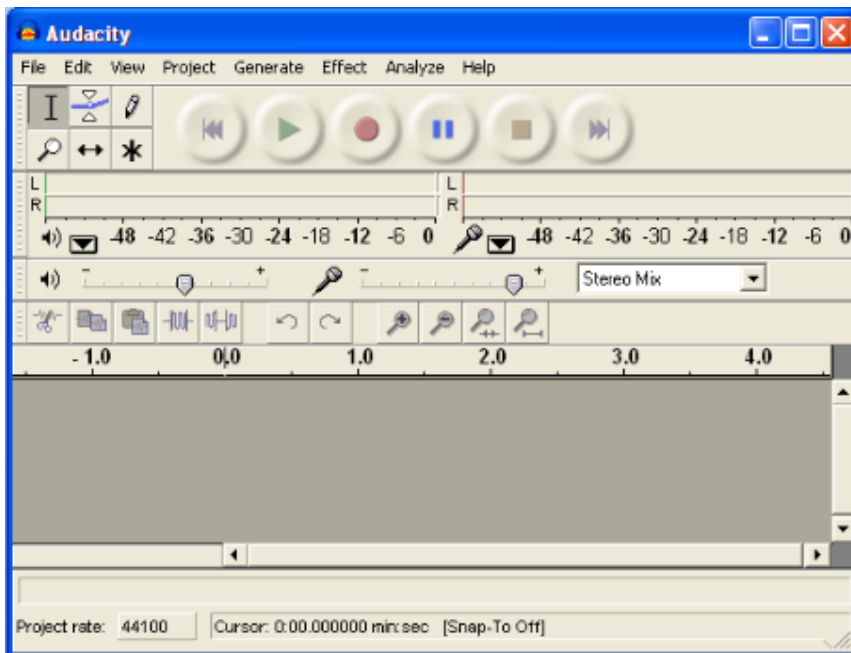
This is the script which plays the tune of **Twinkle, Twinkle Little Star**.

```
play note 60 for 1 beats
play note 60 for 1 beats
play note 67 for 1 beats
play note 67 for 1 beats
play note 69 for 1 beats
play note 69 for 1 beats
play note 67 for 2 beats
rest for 0.2 beats
play note 65 for 1 beats
play note 65 for 1 beats
play note 64 for 1 beats
play note 64 for 1 beats
play note 62 for 1 beats
play note 62 for 1 beats
play note 60 for 2 beats
```



## Step 2: Create a Sound Clip using Audacity

Creating a song using Sound Tool Kit is fine but it takes too much time. Let me clue you in on this wonderful tool called Audacity. [Audacity®](http://audacity.sourceforge.net/) (<http://audacity.sourceforge.net/>) is free, open source software for recording and editing sounds. It is available for Mac OS X, Microsoft Windows, GNU/Linux, and other operating systems.



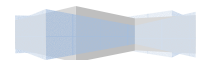
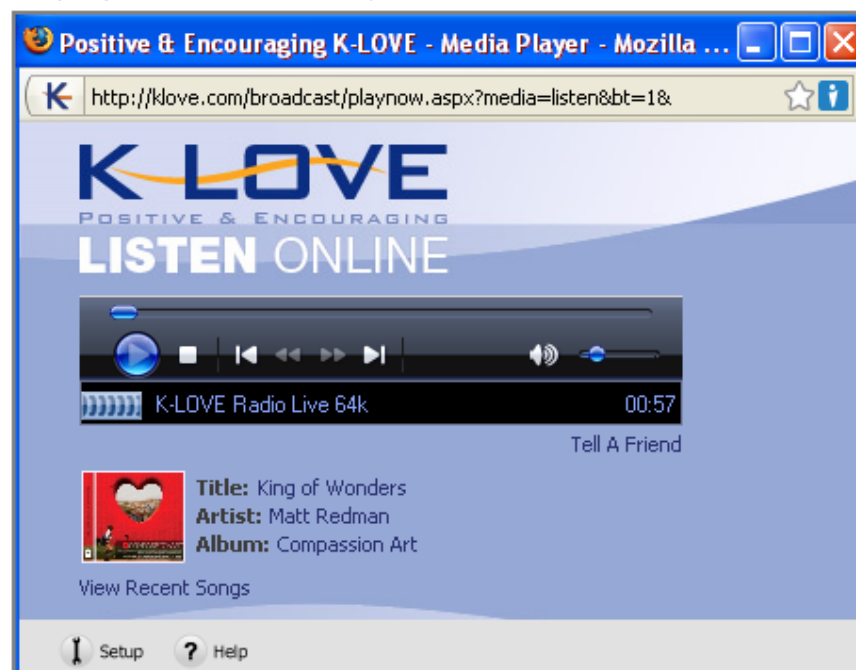
Audacity provides many cool features but I am only going to show you this one trick today. I will show you how to record music played on your computer and turn it to a Scratch sound clip or a wav file (no file converter required!).

Start by playing your favorite song using your favorite

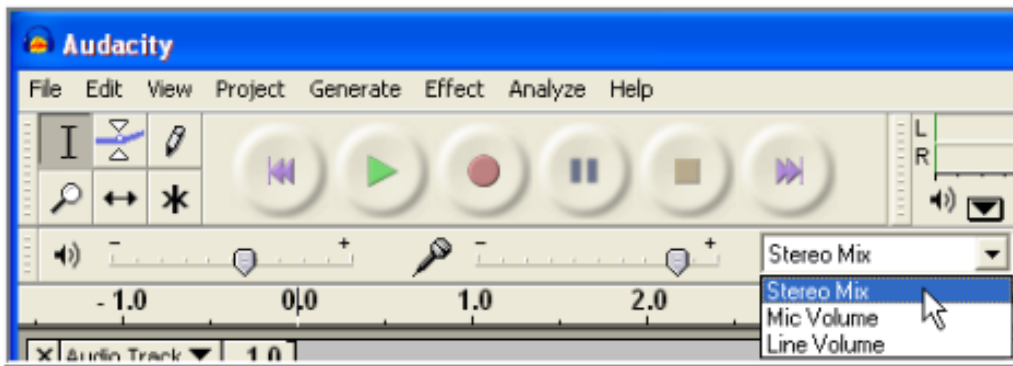


player.

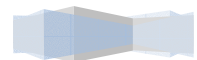
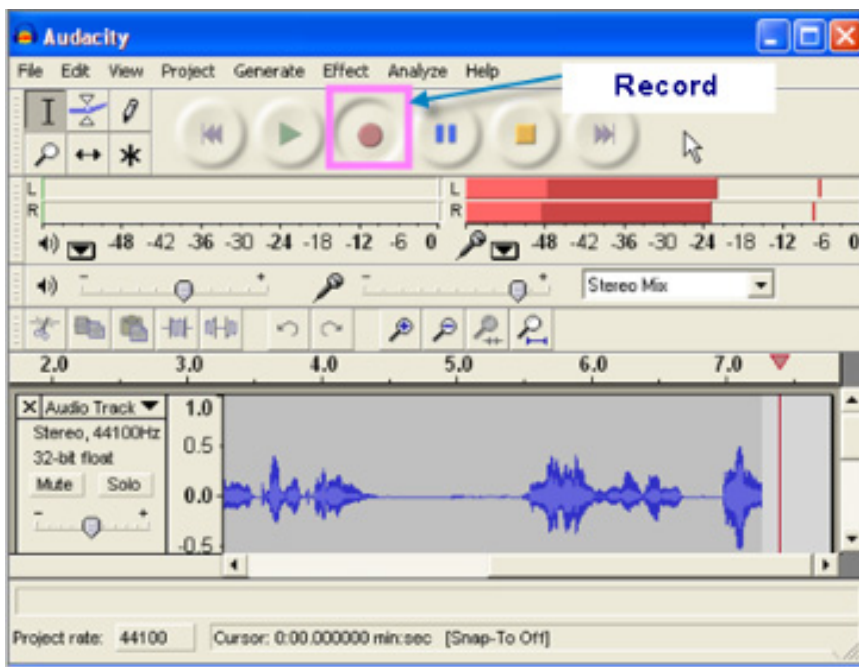
Or play online streaming music.



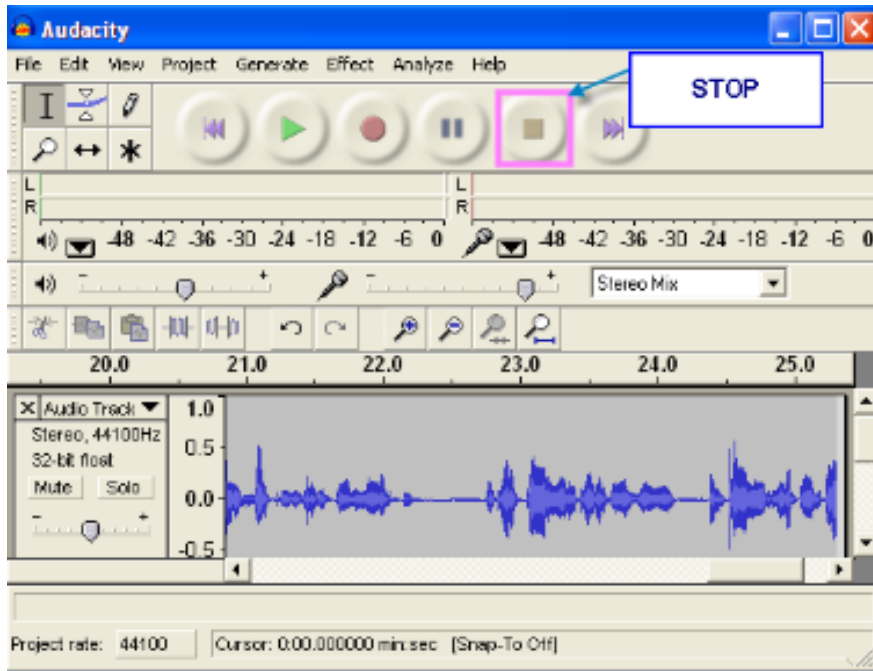
To record the music played on your computer, select "Stereo Mix".



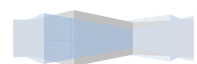
Then hit the "Record" button. Sound waves will start to appear.



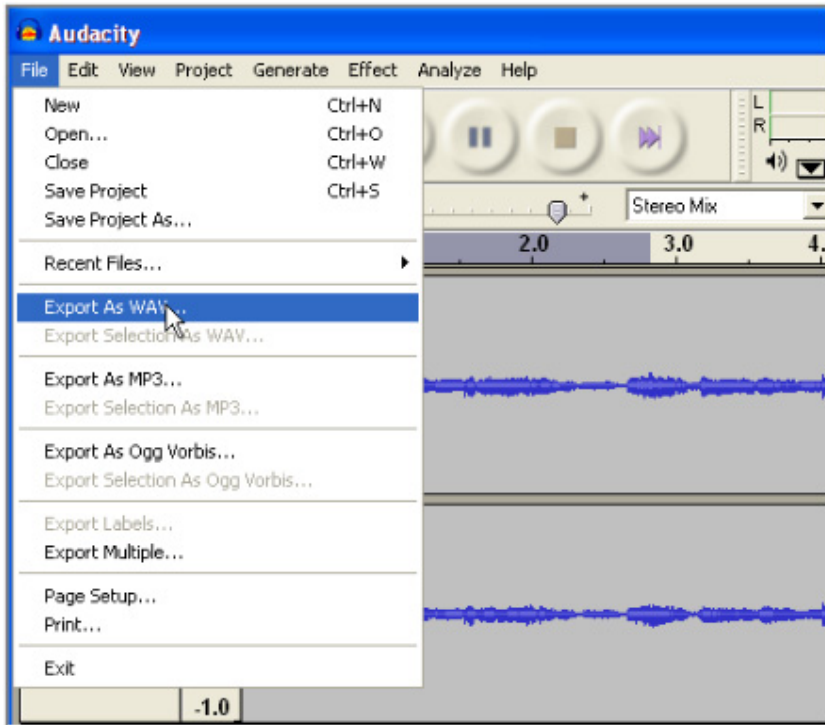
Once you've record enough music, click STOP to stop recording.



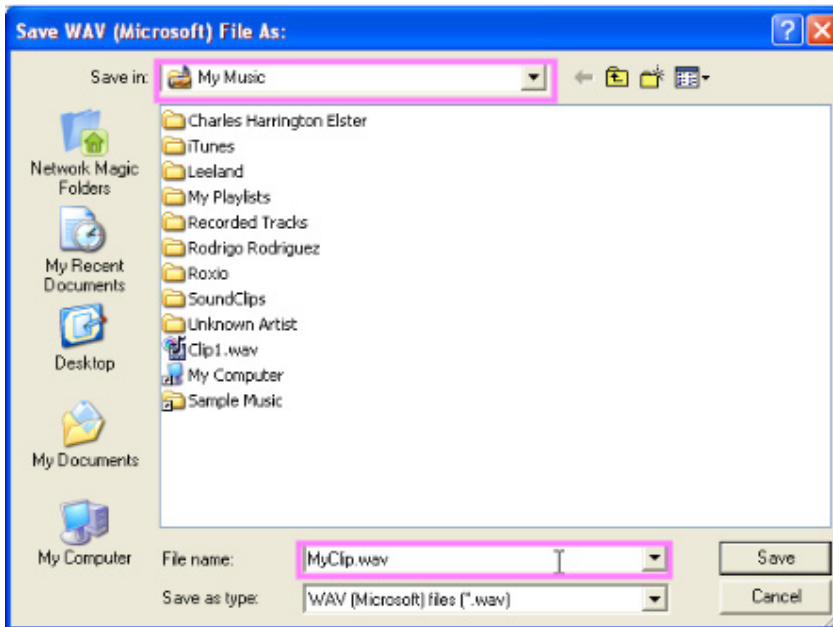
To export the recording as WAV file to be used in a Scratch project.  
Select **"File"->"Export As WAV"**.



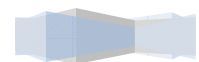




Save the recording in "My Music" as "**MyClip.wav**".

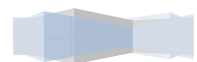


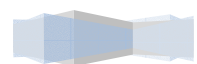
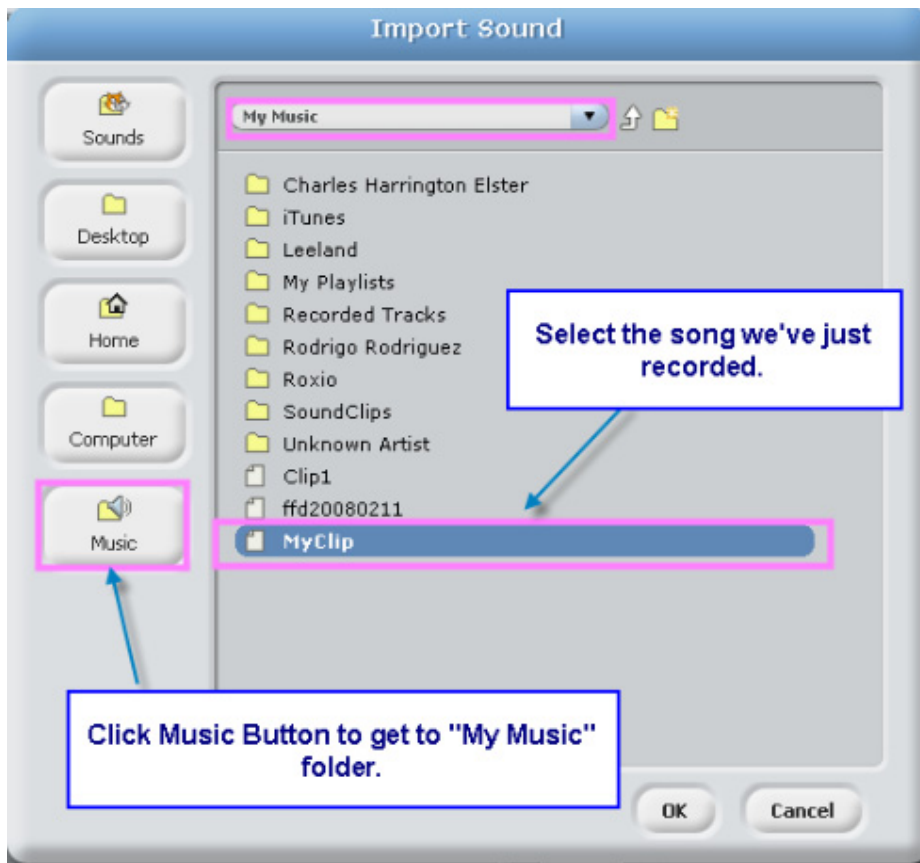
To import to a sprite, select the Sounds Tab and click "**Import**".





Click "Music" button to go to **"My Music"** folder and select the clip we've just recorded.



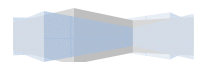
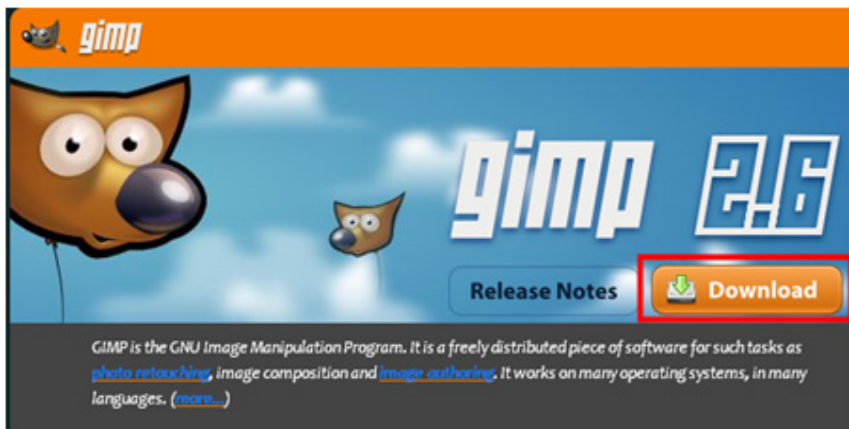


# LESSON 7: TURN YOURSELF TO A SPRITE USING GIMP

This project is great for a lazy Saturday or Sunday afternoon. I will show you how to turn yourself into a Sprite and to do something cool in a Scratch program. If you have followed the Scratch Programming Series, then this lesson will be pretty simple for you but it'll be a lot of fun!! What you need is a photo of yourself, GIMP, and Scratch.

## *Step 1: Get GIMP*

GIMP is a free image editing software that you can download from [GIMP site \(http://www.gimp.org\)](http://www.gimp.org). You might have heard of Adobe Photoshop, a great commercial product (but quite costly)

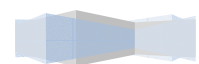
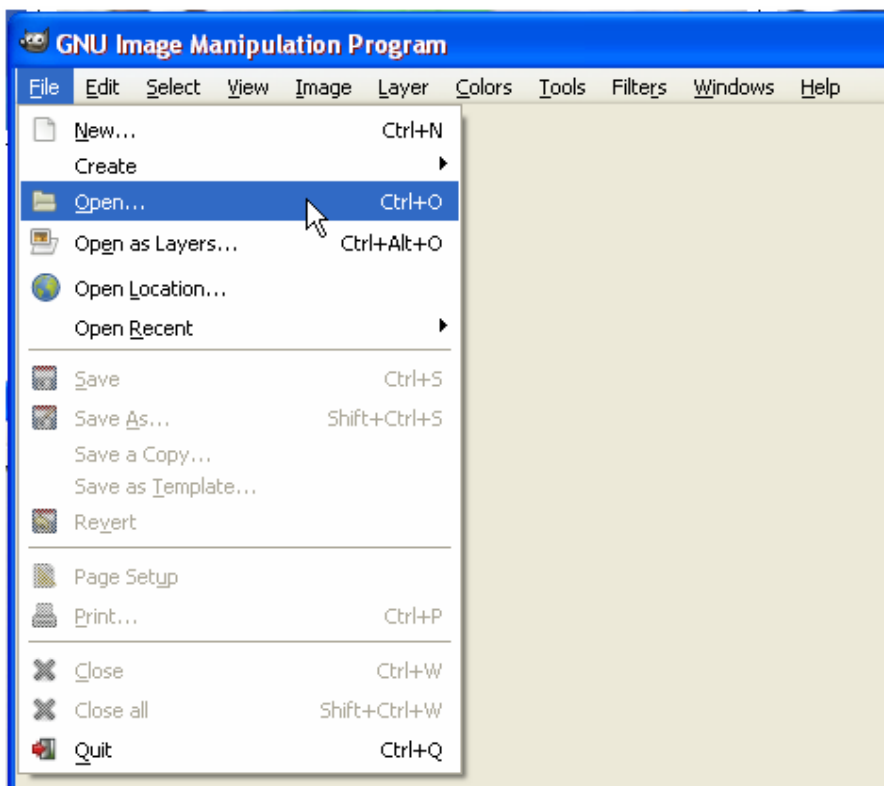


## Step 2: Load a Photo of Yourself to GIMP

These are three pictures I took of my son who was bravely trying a handstand (and as you can see, he needs a bit more practice). I will use the first photo. Get a picture of yourself and let's start!



First, open the photo by going to "Open". Then select the photo.



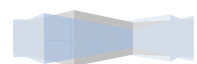
This is what GIMP interface looks like after the photo has been loaded.



*Step 3: Select Yourself in the Photo Using Free Select Tool*

What we are doing today is to simply crop the people image we want and delete the background. Once an image is cropped, we can export

50

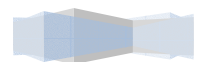


it and then use it in Scratch as a sprite or as an image in a funny background. I will show you how to do both in this lesson.

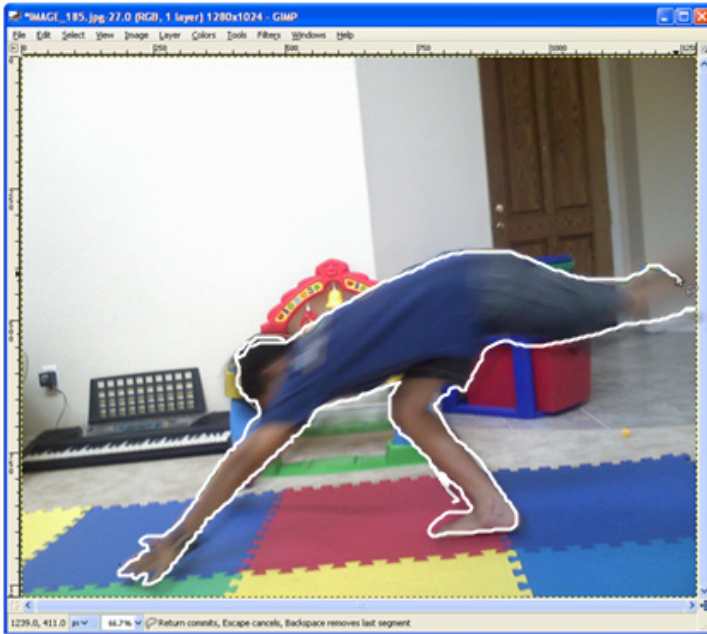
Use the Free Select tool to select the figure. Each time you click, a blue circle will be added around the figure. Keep doing so until you've made a FULLY ENCLOSED area around the figure.



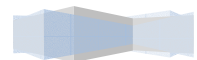
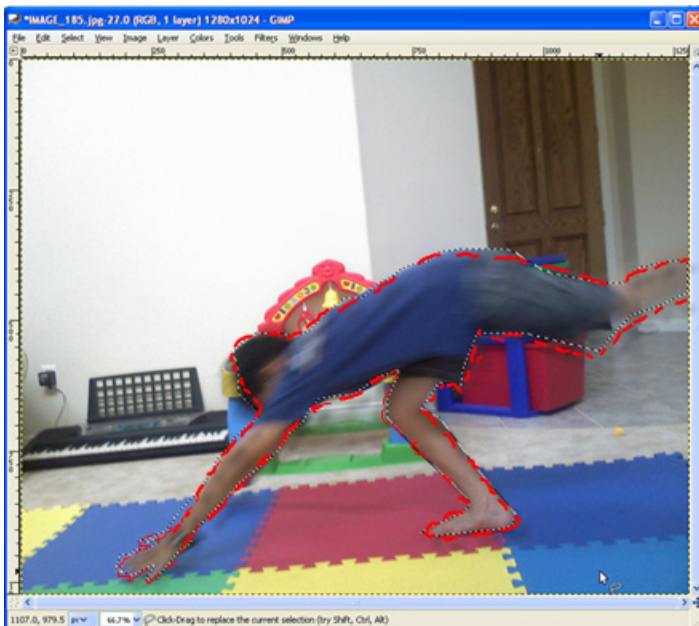
While selecting an image by clicking around it, you would see a thin white line around it as you click.







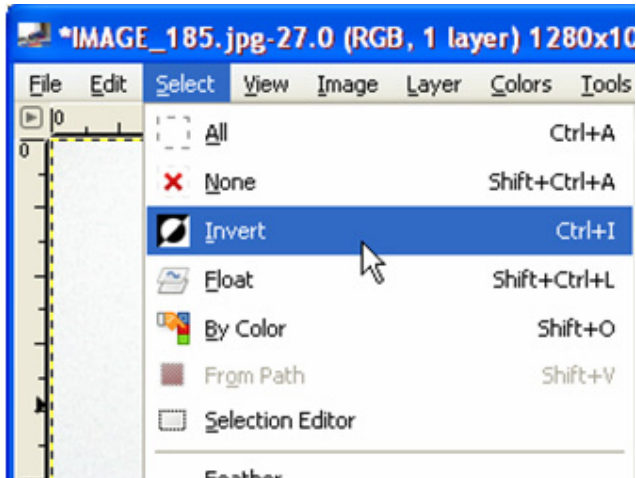
When the selection is completed (a fully enclosed area is made), then a dotted line will blink on and off around the selected figure. The dotted line is shown below in red to emphasize the selection but in reality, it's white dotted line.



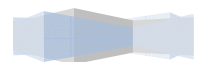


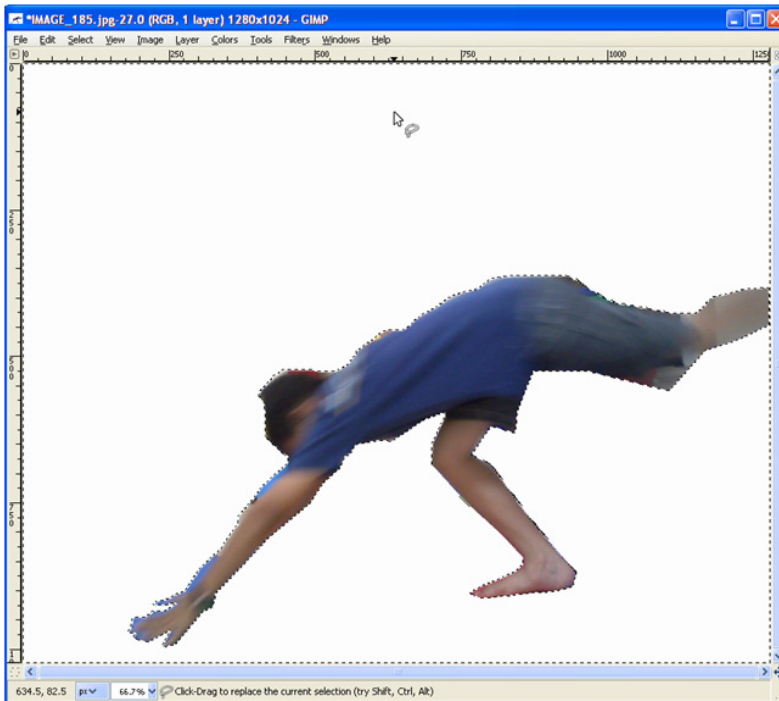
*Step 4: Delete the Background*

Then do "Select"->"Invert" to select the background and hit the "Delete" button on your keyboard to delete it.

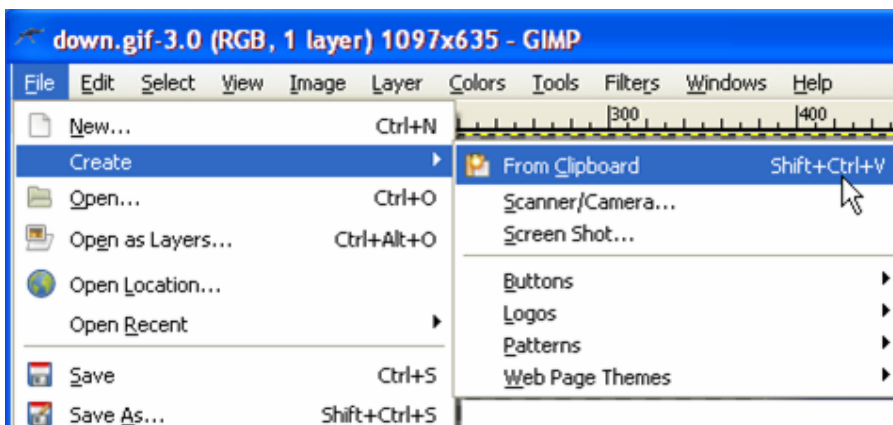


Now the selected background will become blank or filled with white.

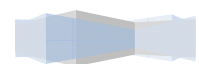


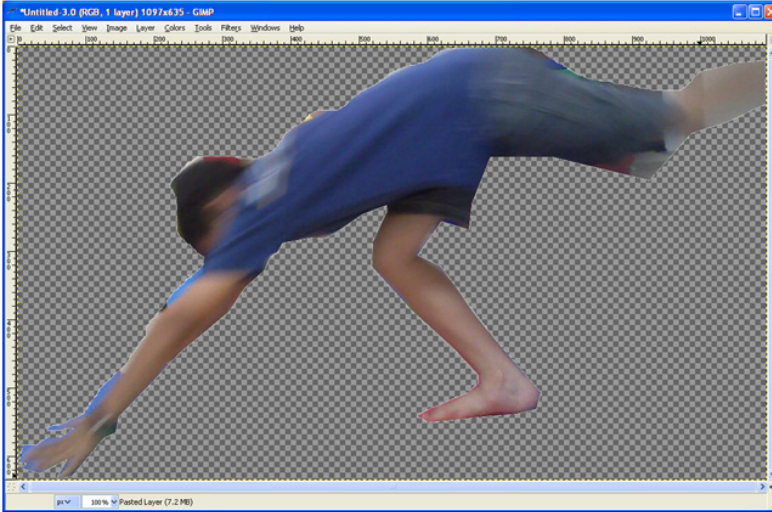


Select the figure by doing "Select"->"Invert" again to select the image. And then click "Edit"->"Copy" to copy the figure. Create a new image with this copied figure by doing "File"->"Create"->"From Clipboard".



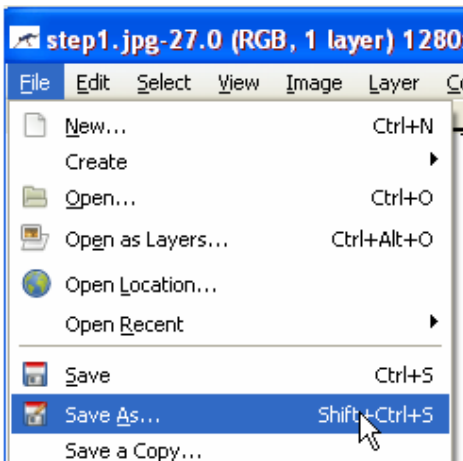
You should see yourself and no background (transparent background).



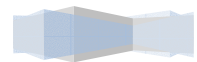


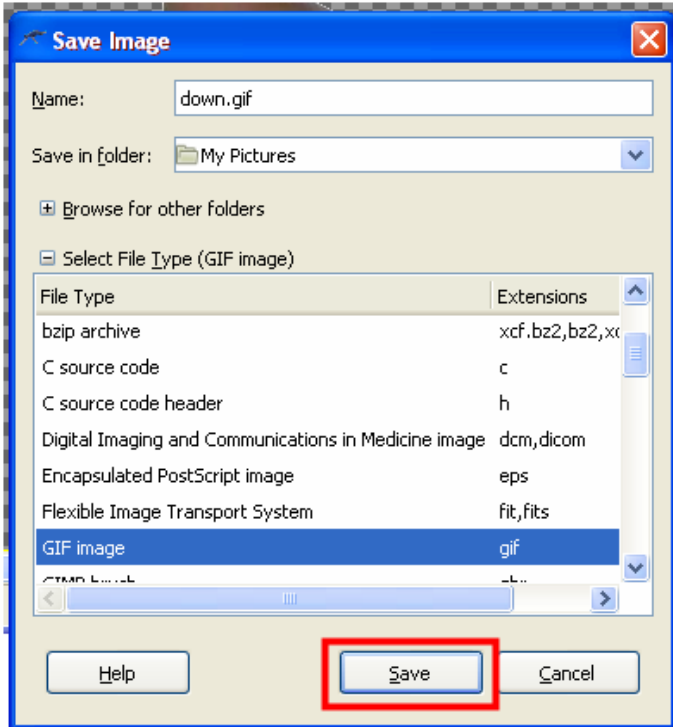
*Ste 5: Save the Result Image*

Now we are ready to save. Go to "File"->"Save As..."

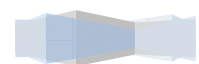
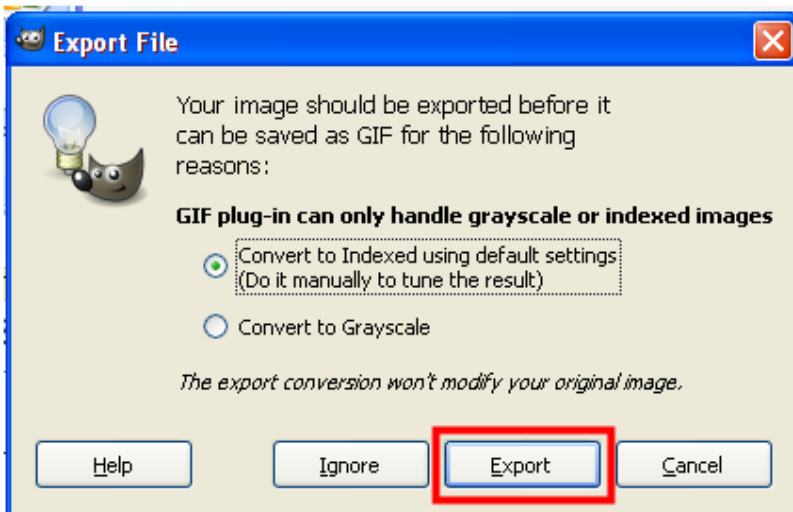


Select the GIF image file type and name it "down.gif". Then click "Save".

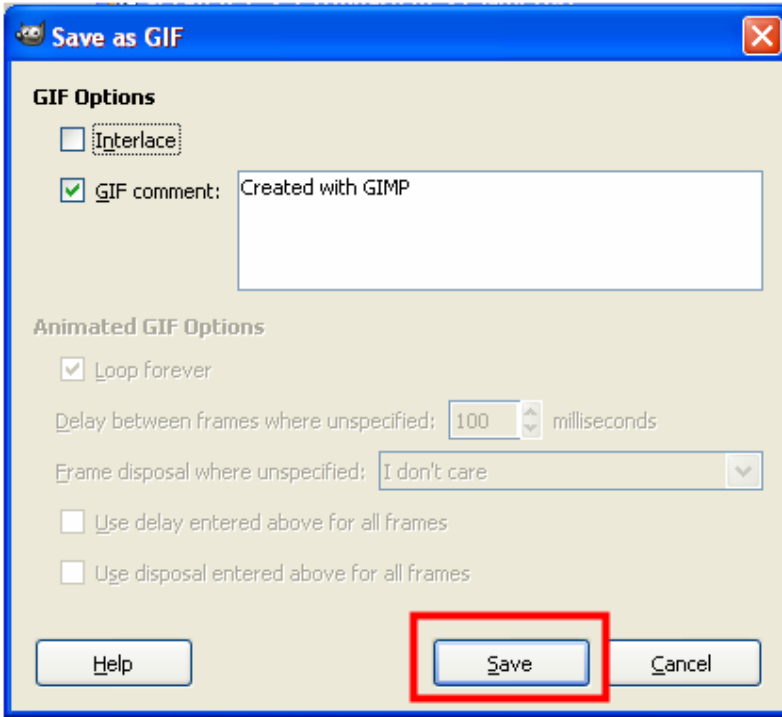




Click "Export" to continue.

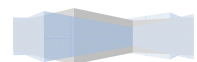


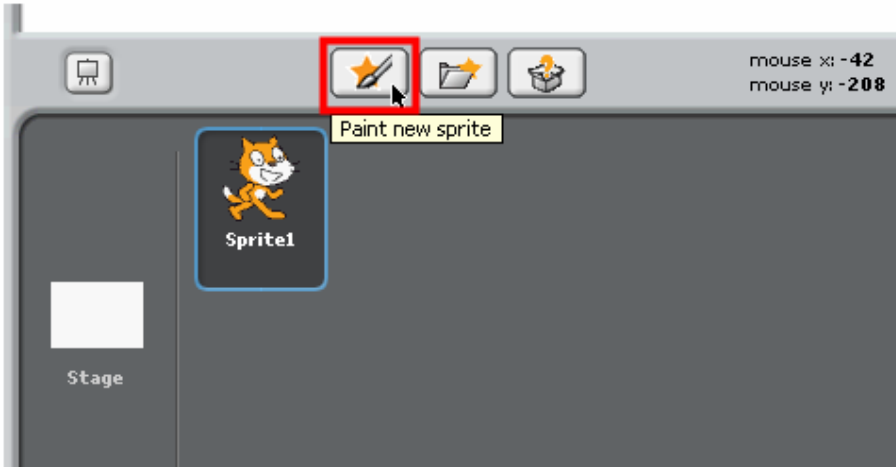
Keep the default selection and click "Save" to finish saving.



### *Step 6: Import Yourself to Scratch*

Now we are ready to create a new Sprite using the saved image. Click "Paint New Sprite".

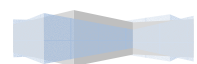


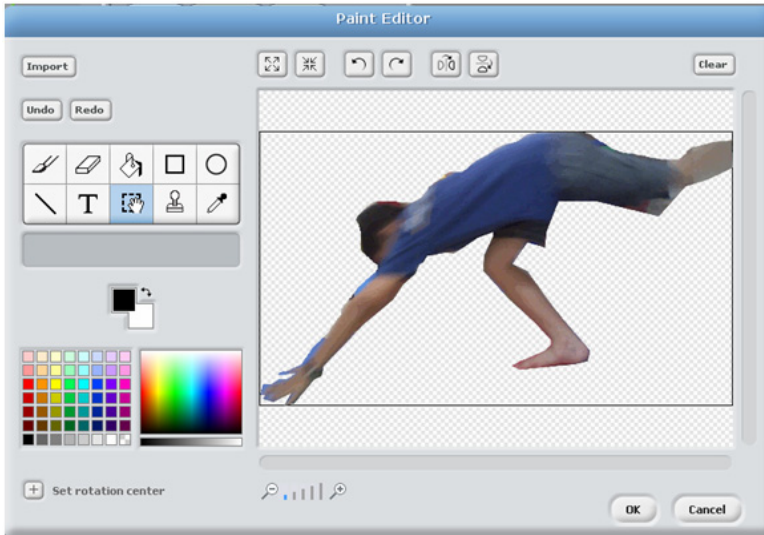


Select "down.gif" and click "OK".

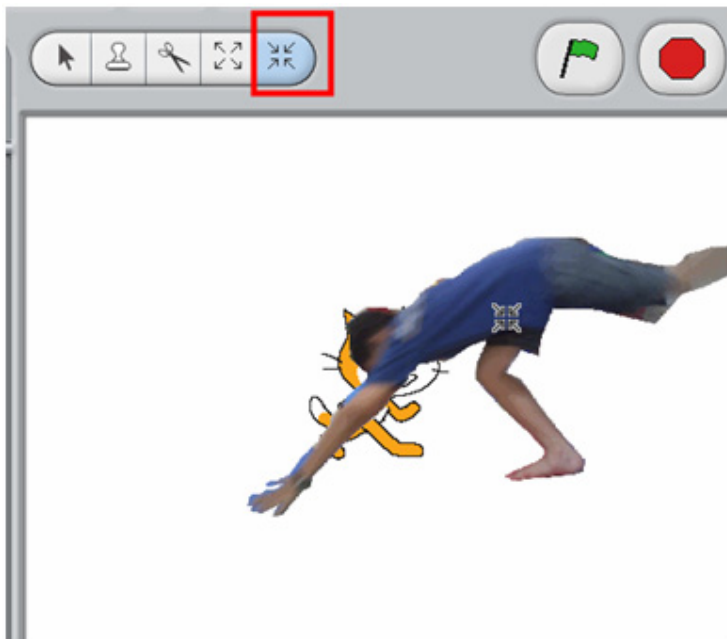


Click "OK" to save the new Sprite.

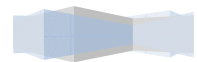




If the sprite is too big, use the Shrink tool.



We are done! Now you know how to a person in a picture to a sprite.







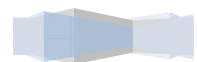
## LESSON 8: PROJECT – MANIKIN DANCE



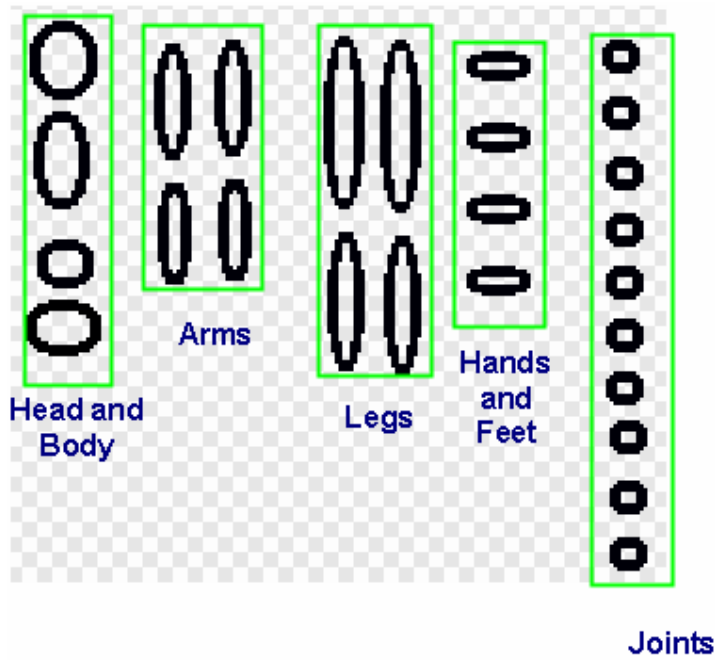
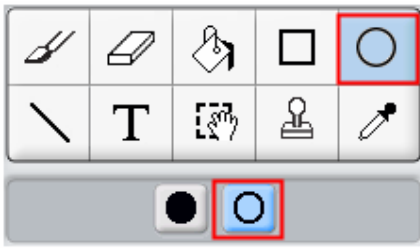
You probably have a wooden manikin in your house or you've seen one in an Art store. Today we are going to create a Scratch manikin and teach him/her to break dance. Why break dance? It's cool and easier to animate.

*Step 1: Create the Manikin Parts*

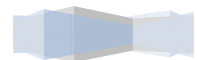
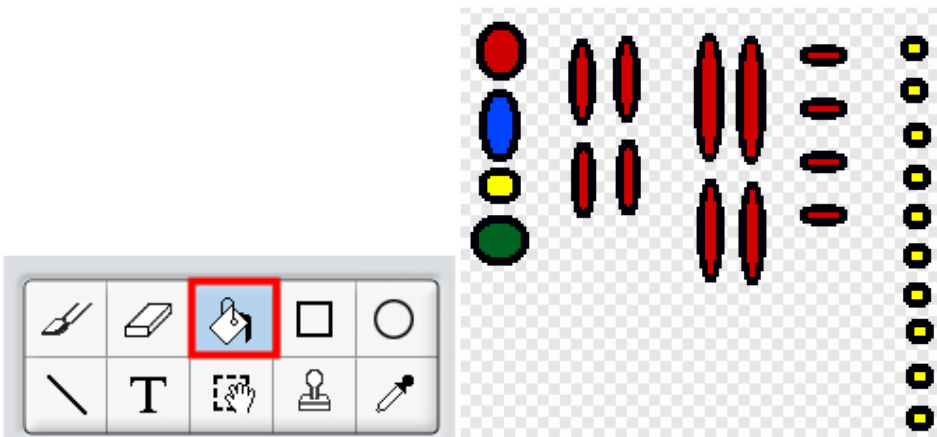
Let's Start! First, create a new Sprite by clicking the "Creating a New Sprite".



Select the Ellipse Tool with Hollow mode. Create circles for different body parts.

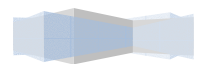
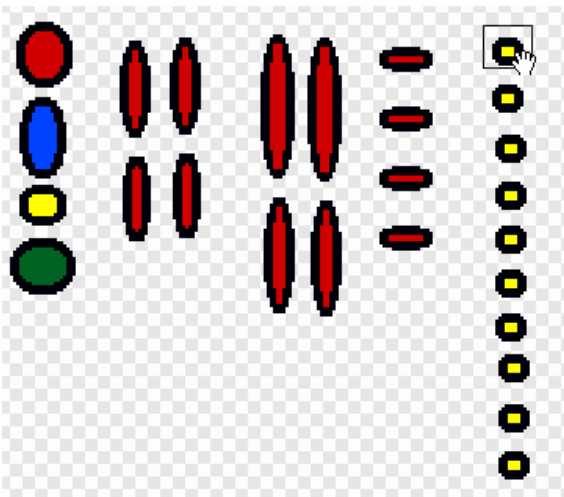


Color each body part using Fill Tool.

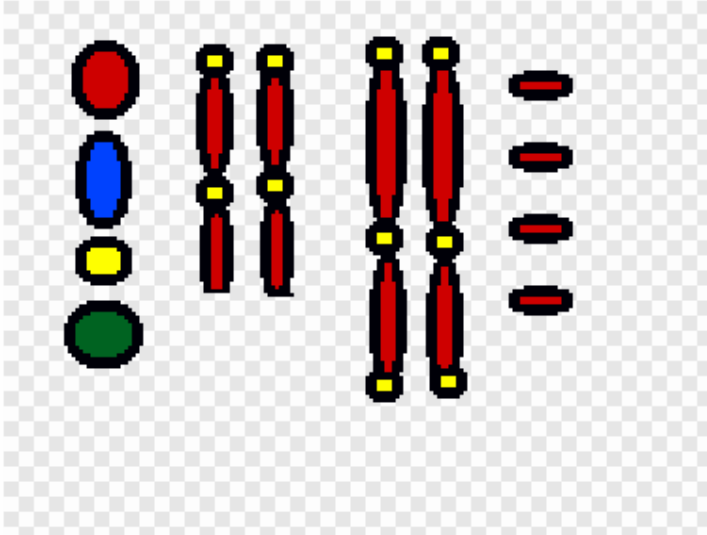


## Step 2: Connect Body Parts Together

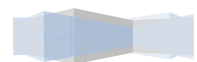
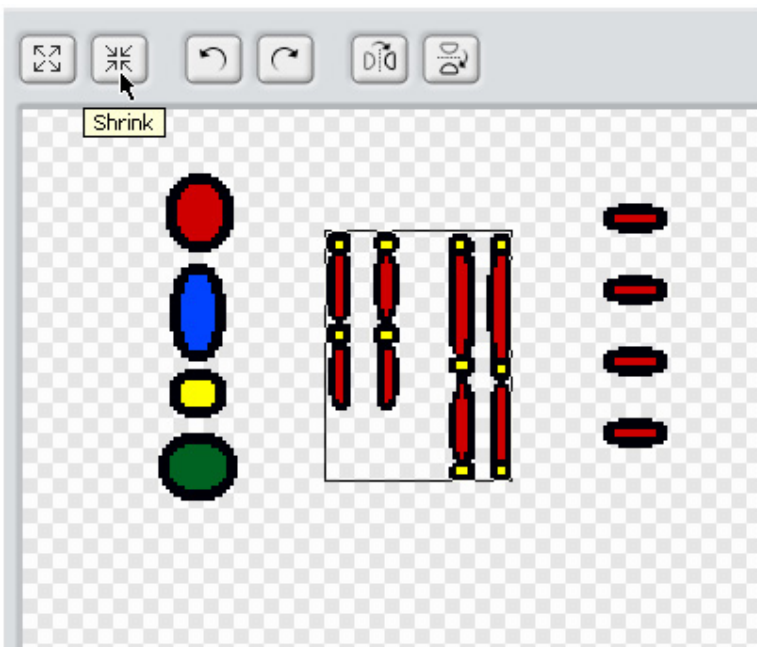
Then use Select Tool to select each body part and connect with each other.



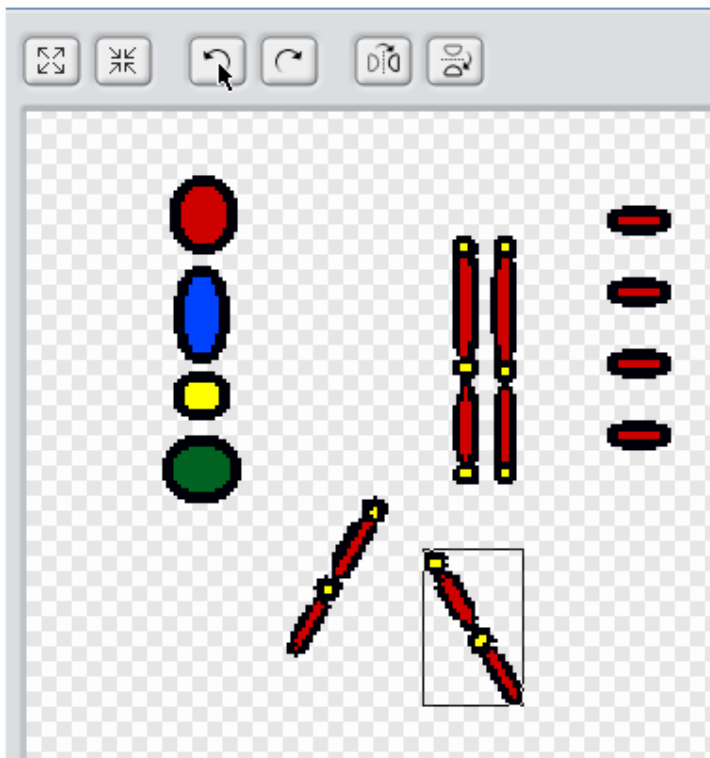
This is the snapshot of arms and legs



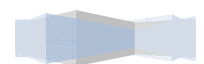
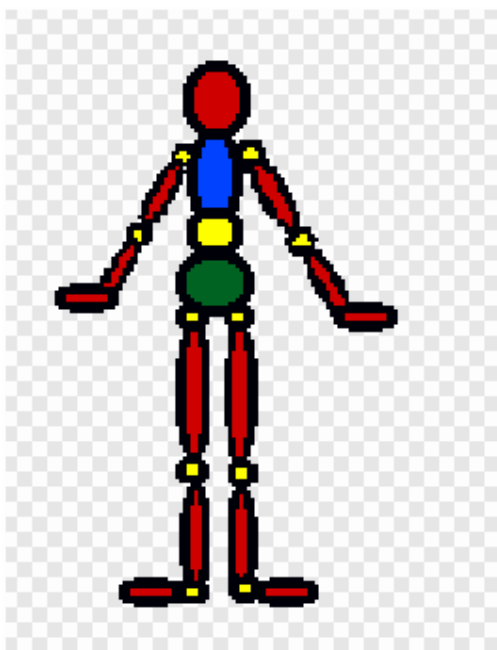
Use Select Tool and Rotation Tool to rotate the body parts and connect them to form a body. You can also Shrink Tool to shrink the arms and legs for better proportion.



Use the Select Tool to first select the arms Rotate Tool to rotate the arms.

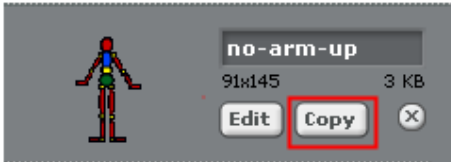


Connect all body parts to form our Manikin!

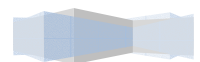
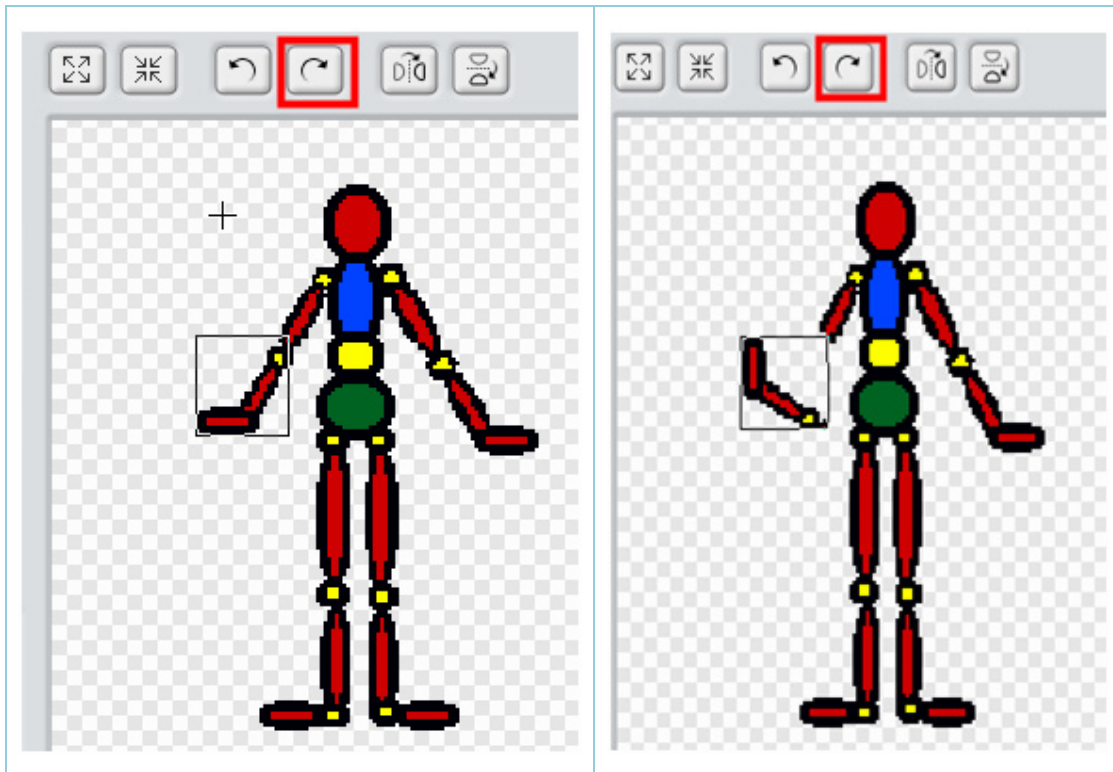


### Step 3: Create Multiple Dancing Postures

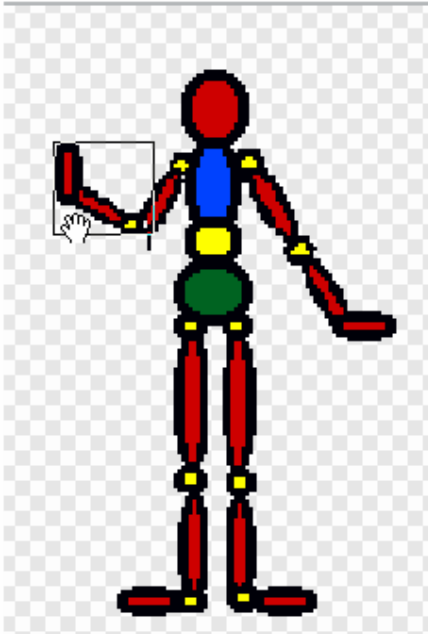
Click OK to save and rename the costume to "no-arm-up".



Use the Select Tool to select the forearm and then use the Rotate Tool to rotate.



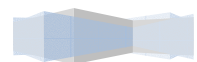
Drag the arm to reconnect it to the body.



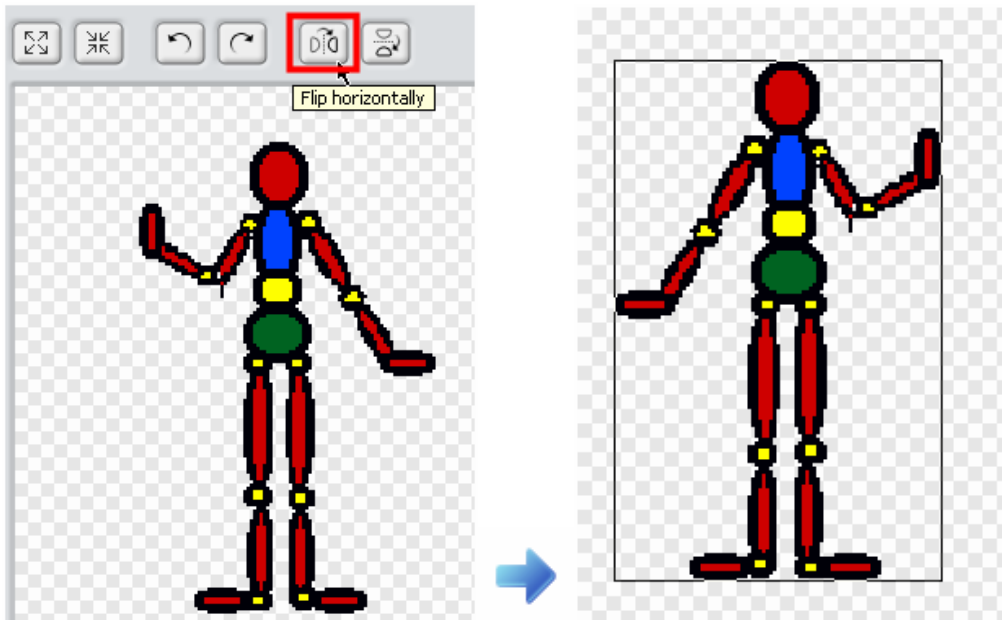
Then rename the costume to "left-hand-up". Click "Copy" to make a copy.



This is for costume "right-hand-up". Click "Edit".



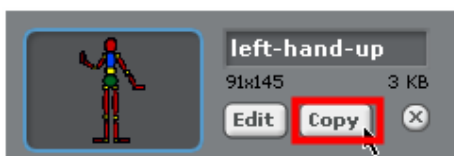
Click the "Flip horizontally" icon to flip the manikin horizontally.



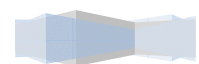
Rename the costume to "right-hand-up".



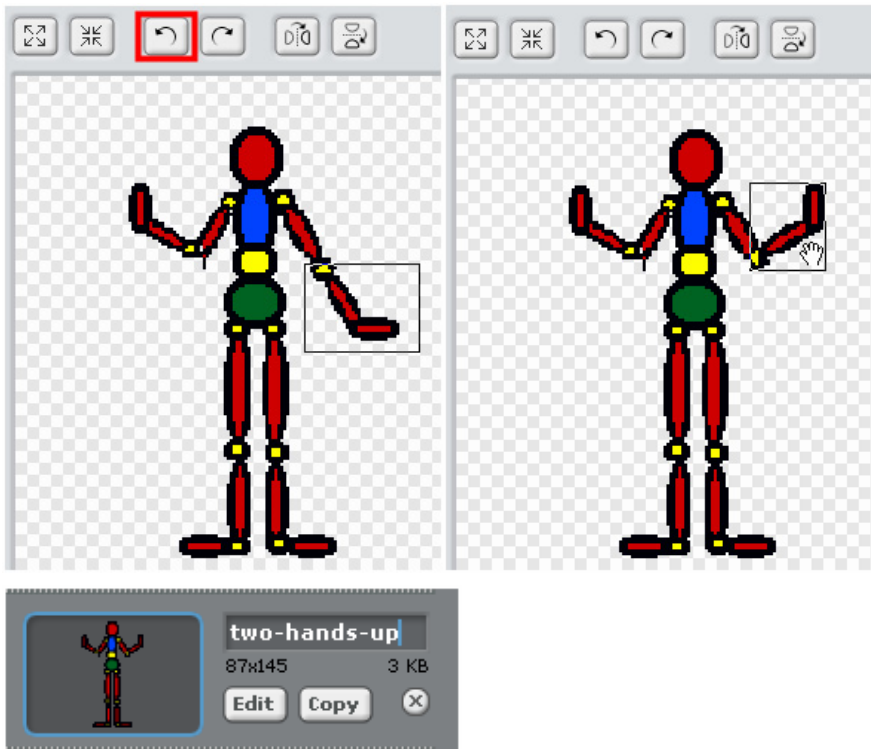
Next, we will create a costume named "both-hands-up".



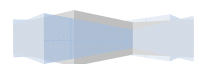
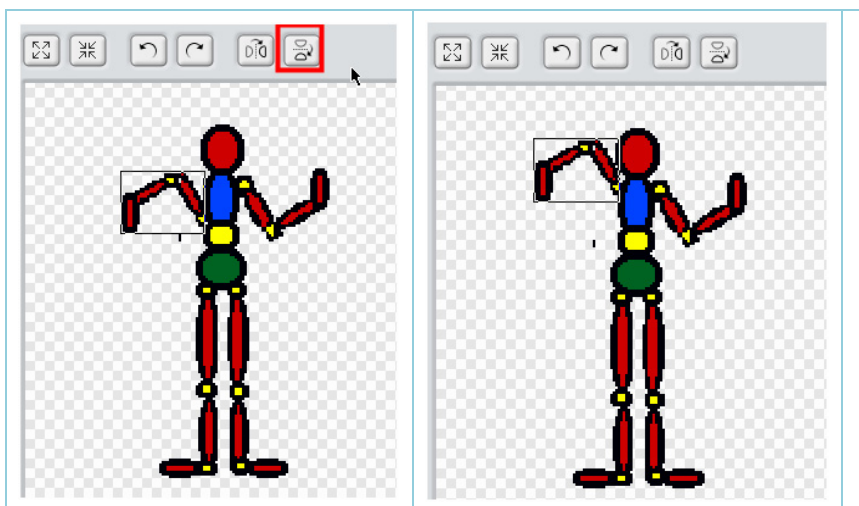
Use Select Tool to select the left arm and then use Rotate Tool to rotate it up. Reconnect the left arm to finish up this costume. Rename the costume to "two-hands-up".







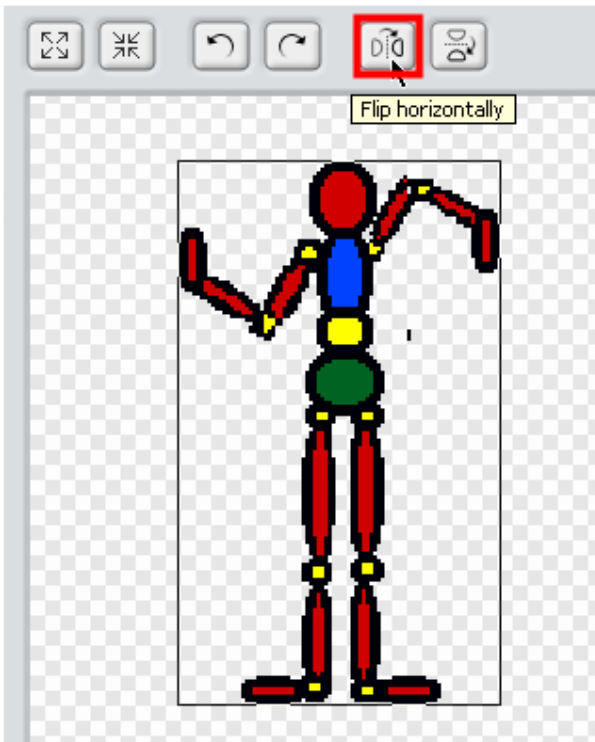
Next, we will create a costume named "right elbow up". Use Select Tool to select and Flip Tool to flip.



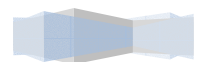
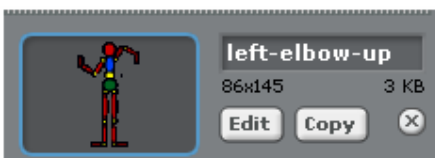
Rename this costume to "one-elbow-up" and then click "Copy" to make another copy.



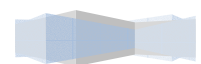
Edit "one-elbow-up2" by clicking the "Edit" button. Then click "Flip horizontally" to flip the manikin.



Then rename this costume to "left-elbow-up".



This is the final lists of costume for the manikin sprite.



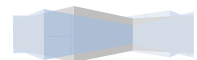
### Step 4: Make It Dance

Now we are ready to make it dance!

Drag "when the green flag clicked", "forever", and "wait ? secs" blocks from the Control Tool Kit to the Stage. Then drag "next costume" from the Look Tool Kit to the Stage.



Build the combo block as followed.



## Step 5: Test

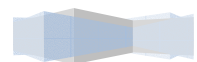
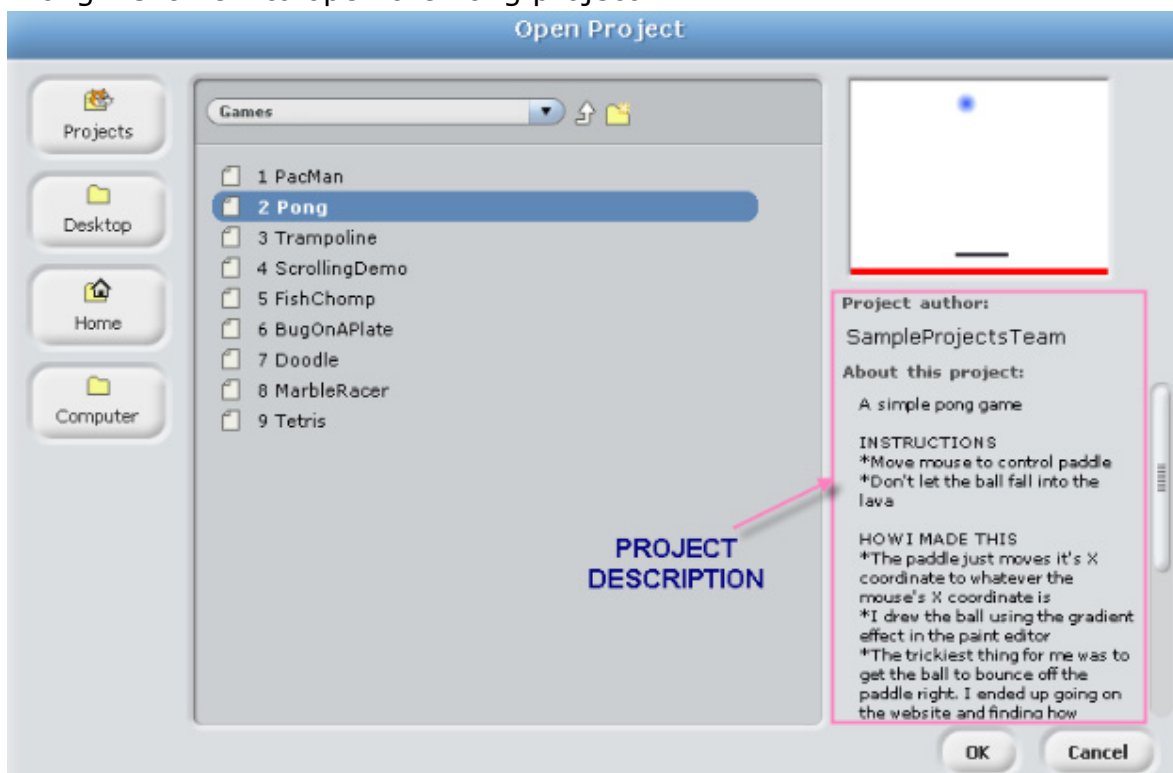
Ready to dance! Click the green flag to test.

To kick it up a notch, add more dance moves such as handstand or other floor moves.



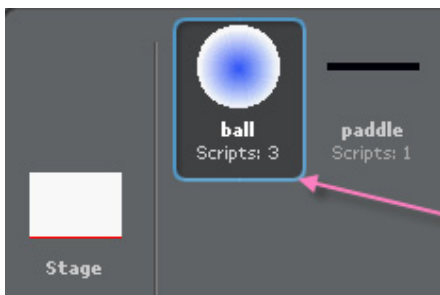
## Lesson 9: The Pong Game

In this lesson, we will modify a sample game named "Pong". We will add score, levels, and more balls to the game. To open the "Pong" game, click "Open" and browse to Projects->Games, and then select "Pong". Click OK to open the Pong project.

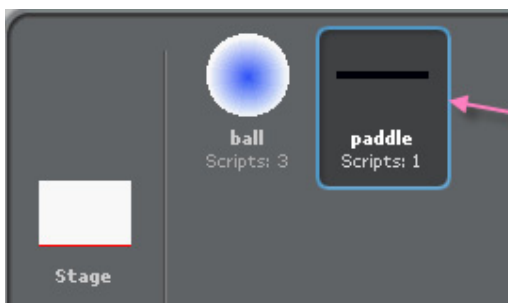


*Step 1: Understand how the Pong Game works*

There are two sprites: ball and paddle. The ball sprite moves randomly by its own, whereas the paddle sprite is moved by user moving the mouse.

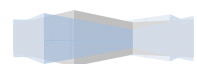


The ball sprite will move randomly.

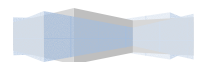
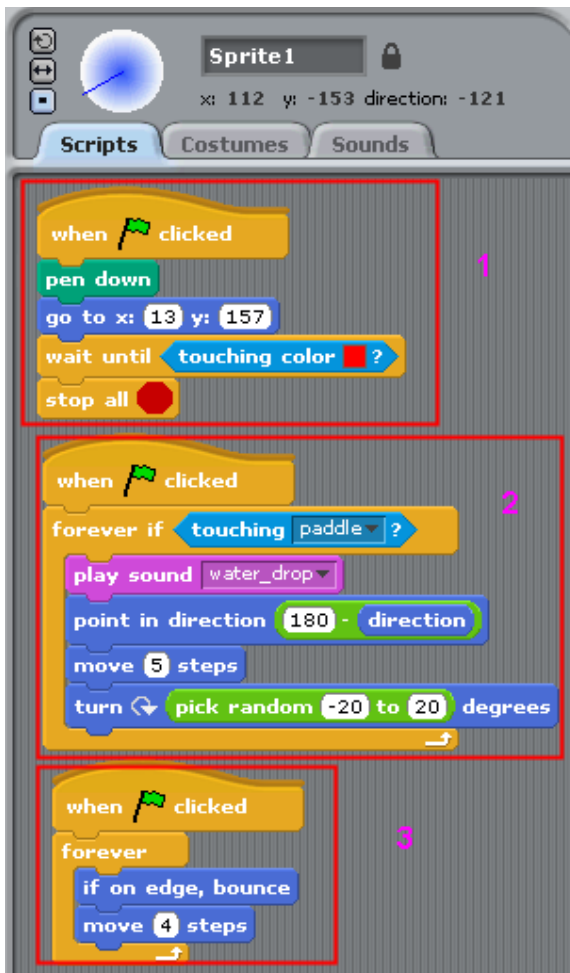


The paddle sprite is controlled by a human user.

Click the green flag to try the game. Use the paddle to hit the ball as it falls down. You should soon find that if the ball hits the bottom of the Stage, the game would end.

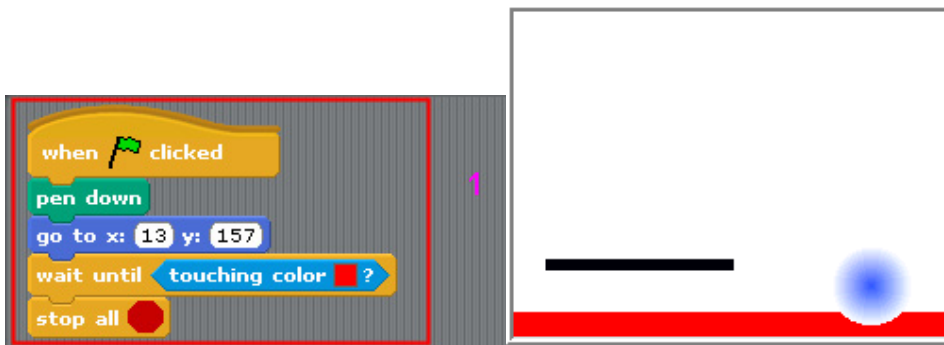


The ball sprite has three combo blocks. You can think of a sprite as a **robot** and each combo block as **a separate motor**. Just like a robot is controlled by many motors, a sprite can be controlled by many combo blocks. In Computer Science language, each combo block is **a thread of execution**, working independently of the other threads.

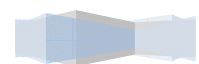
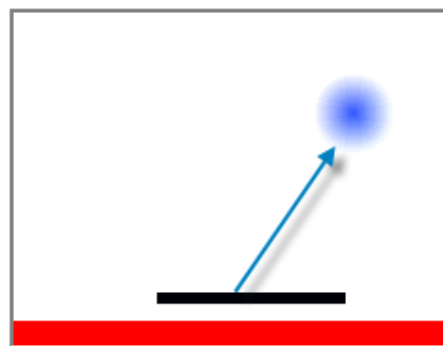
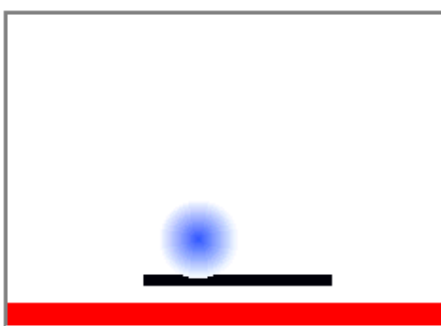




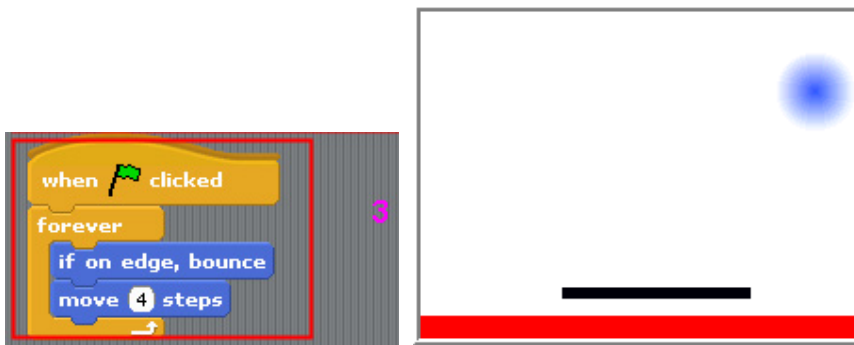
In the picture shown left, first combo block controls how the game would end - if ball touches red, then game ends.



The middle combo block checks whether the ball has touched the paddle and takes action when it does.

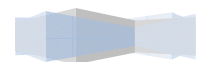
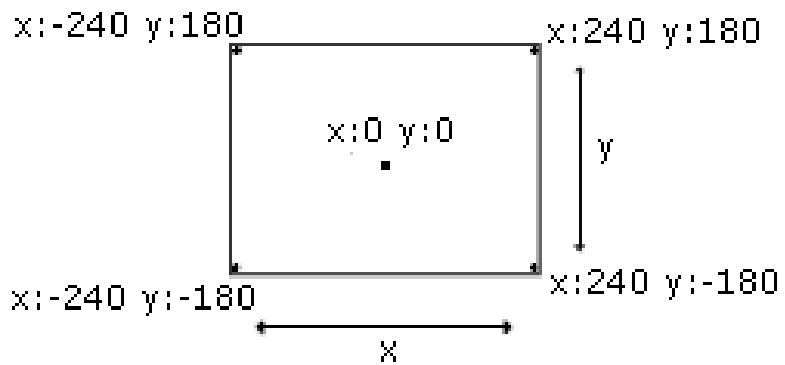



The last combo block checks whether the ball has hit the edge and takes action when it does.



*Step 2: X-Y Coordinate System and Rotation in Scratch*

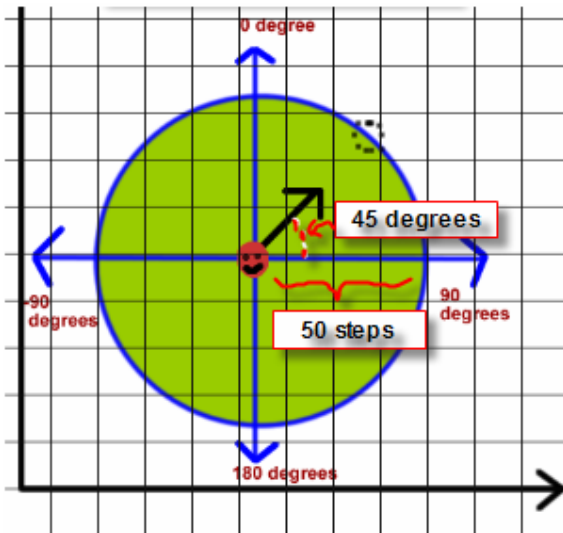
The image below shows how a sprite moves in the Scratch stage which is based on the X-Y coordinate system.



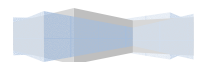
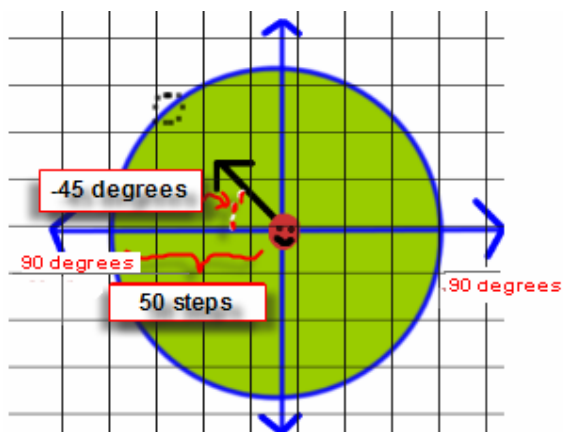
This sprite  moves according to this script:

```
point in direction 45
move 50 steps
```

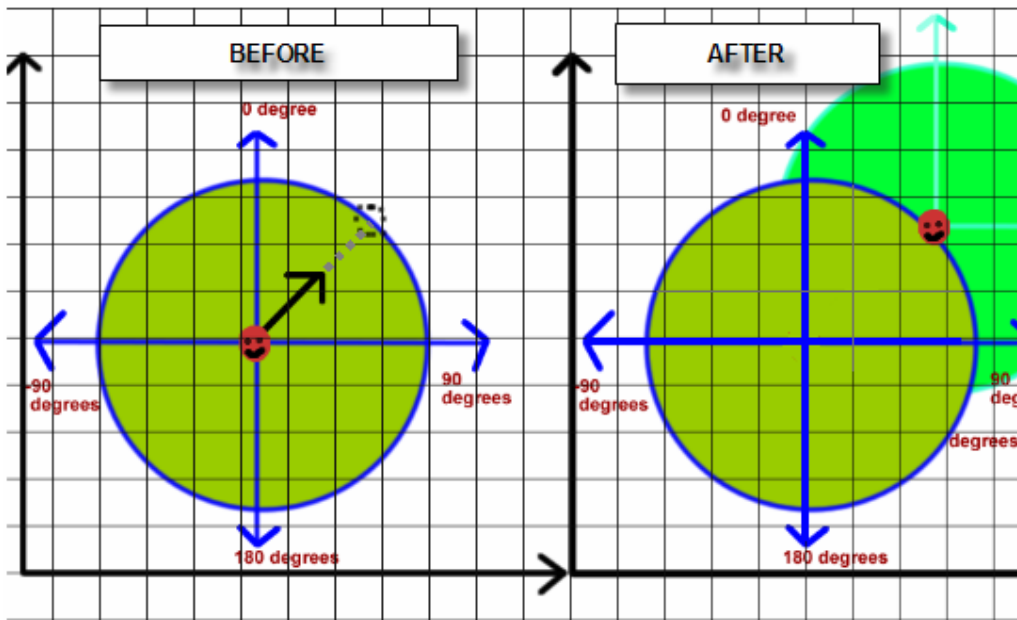
It works as shown in the image below. The red dot sprite first turns 45 degrees from the y-axis and then move 50 steps



Just to make it more clear, this is what it looks like if the red dot first turns -45 degrees then move 50 steps.

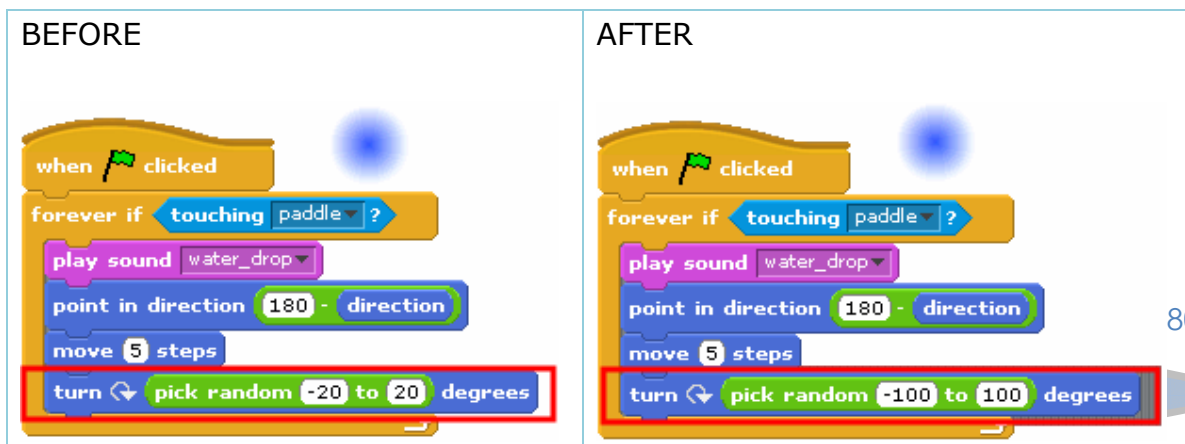


This is the location of the red dot sprite before and after it turns 45 degree and moves 50 steps.



### Step 3: Modify the Pong Game

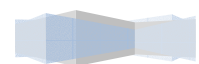
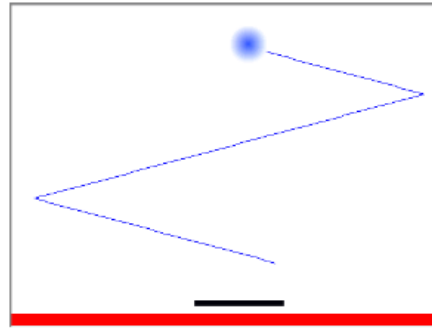
Try changing the randomness by changing the value in **pick random ? to ?** block. To make it more random and the game more difficult, increase the range of the degree change.



Also, you can make the ball leave trace of its movement by using the Pen Tool Kit.

```
when clicked
  clear
  go to x: 13 y: 157
  wait until touching color
  stop all

when clicked
  forever if touching paddle
    pen down
    play sound water_drop
    point in direction 180 - direction
    move 5 steps
    turn pick random -100 to 100 degrees
```



## LESSON 10: STORIES TO ANIMATION PART I

In this lesson, we will create an Animation. We will first create a story line. Based on the story line, we will create sprites and scenes, and finally add scripts to put together the story.

In this lesson, we will create a Cartoon Animation. We will first create a story line. Based on the story line, we will create sprites and scenes, and finally add scripts to put together the story.

### *Step 1: Create a Story Line*

The first step to create an animation is to have an idea. It could be from a joke you've heard or a story you recently read. Or better yet, it could be entirely your own. I've created a really simple story as shown below.

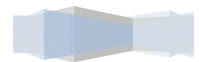
In a small village far, far, AND far away, there lived a little boy who dreamed big.

Bobby was a 9-year-old kid who got bullied all the time by this crazy bully and his sidekick.

Then he met Master Meow, the Kung Fu master. Bobby was under intense training for many months.

Then Bobby entered Kung Fu competition at school and defeated the bully.

From this simple story line, we have this "ingredient list" for our animation:



Sprites	Bobby, bully, sidekick , Master Meow
Scenes	Village, School, Training Ground, Fighting Stage

*Step 2: Create Sprites*

From this simple story line, we need these sprites:

Sprites	Bobby, bully, sidekick , Master Meow
---------	--------------------------------------

By now, I trust that you should be pretty comfortable drawing your own sprites, so we will just create our sprites by importing images from Scratch library.

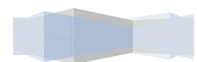
Create or import a sprite and name it "Bobby". Add costumes: "front", "right", "left kick" and "right kick".



Add a new sprite and name it "bully". Add costumes: "up", "down", and "defeated".



Add another new sprite and name it "sidekick". Add costumes: "tease" and "sad".





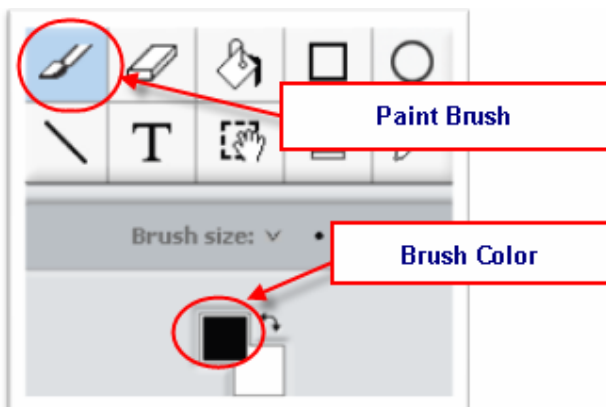
Add yet another new sprite and name it "Master Meow". Just give him one costume: "happy".



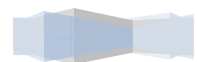
To create scenes, just add backgrounds for Stage. In the next step, we will create four backgrounds: village, school, training ground, and fighting stage.

### *Step 3: Creating the Village Background*

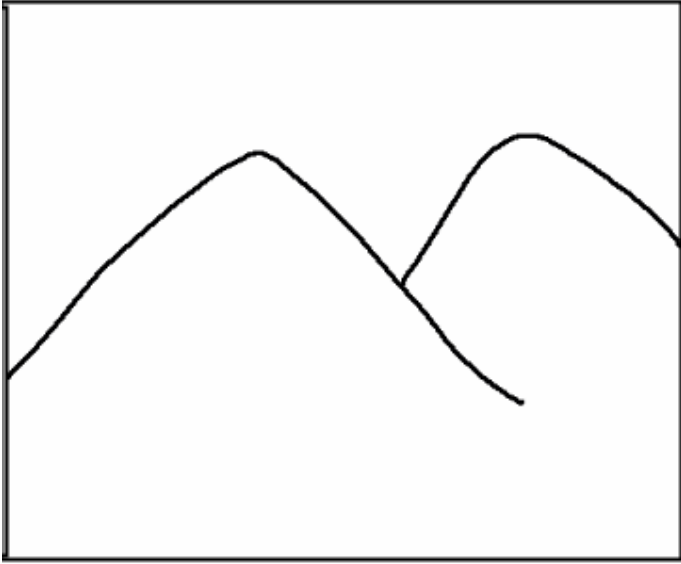
Draw outline of mountains using Paint Brush tool. You can change the default Brush Color.



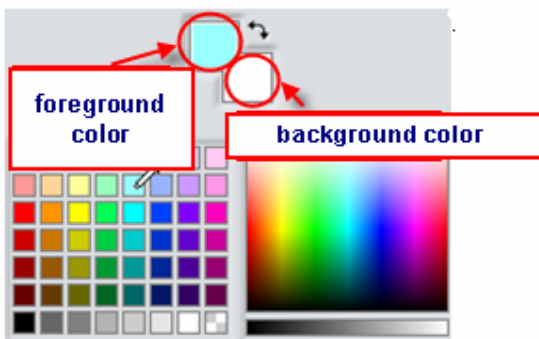
Make sure to draw it all the way to the side of the canvas.



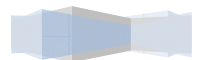


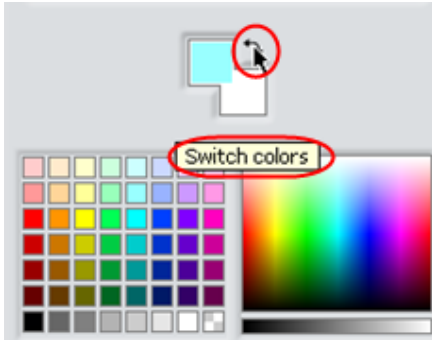


We need light blue for sky and light yellow for the sun light. To select two colors, click first either Eyedropper Tool or Fill Tool to turn mouse arrow to an eyedropper icon. Then select the light blue as foreground color.

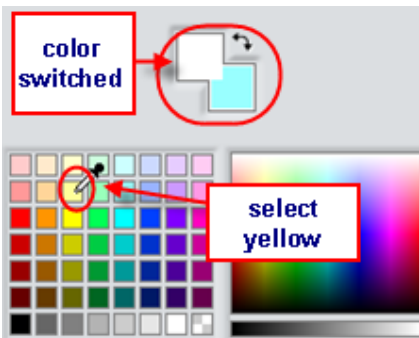


Click the two-ways arrow to switch colors, so that blue would become background color.

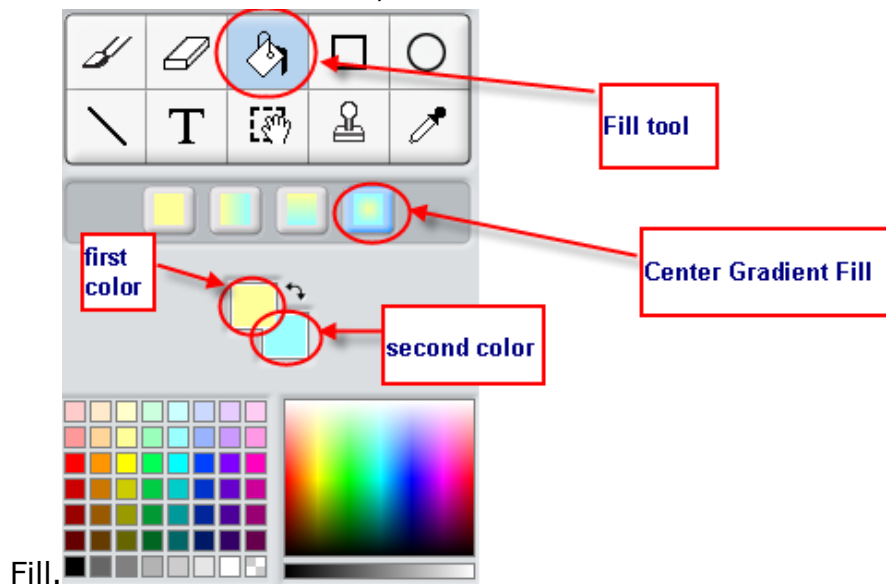




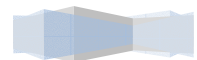
Blue has become the background color. Select light yellow to be the foreground color.



Click the Fill Tool button, and select Center Gradient

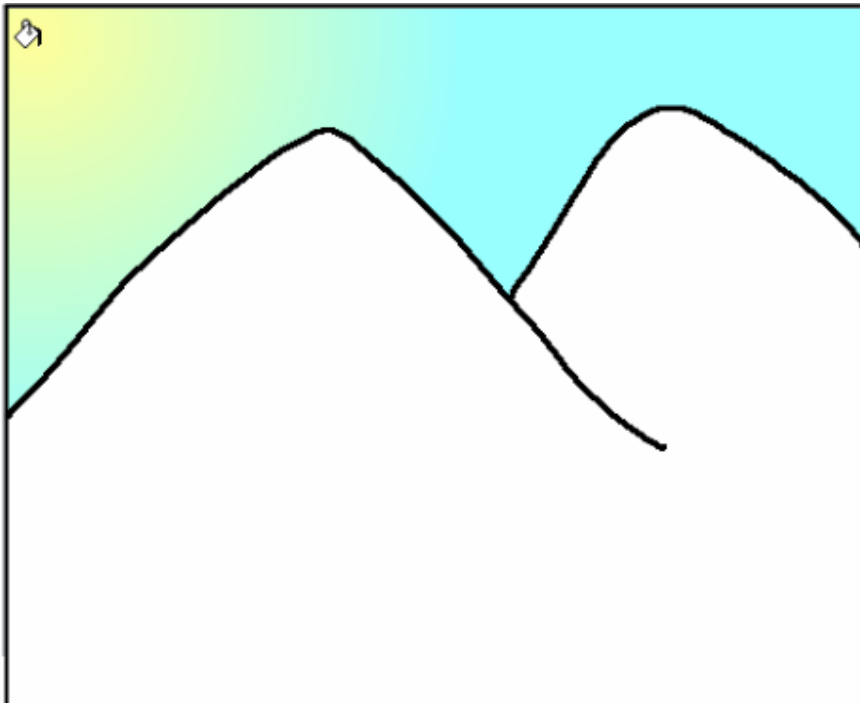


Bring the little paint bucket icon to where you like the sun to be. I like the sun to be at the upper left hand corner.

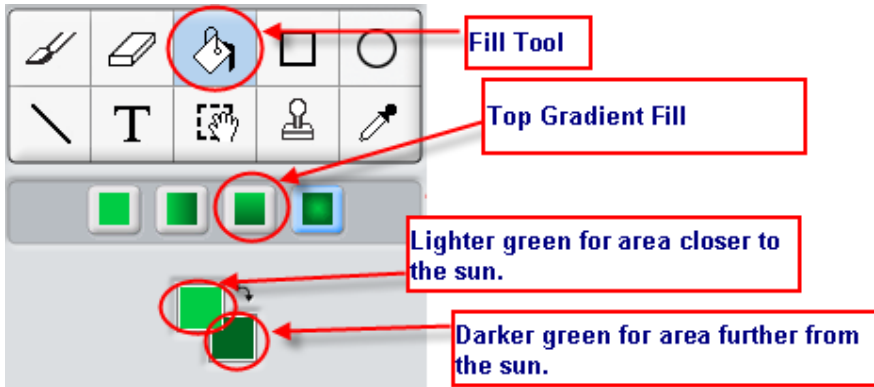




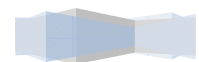
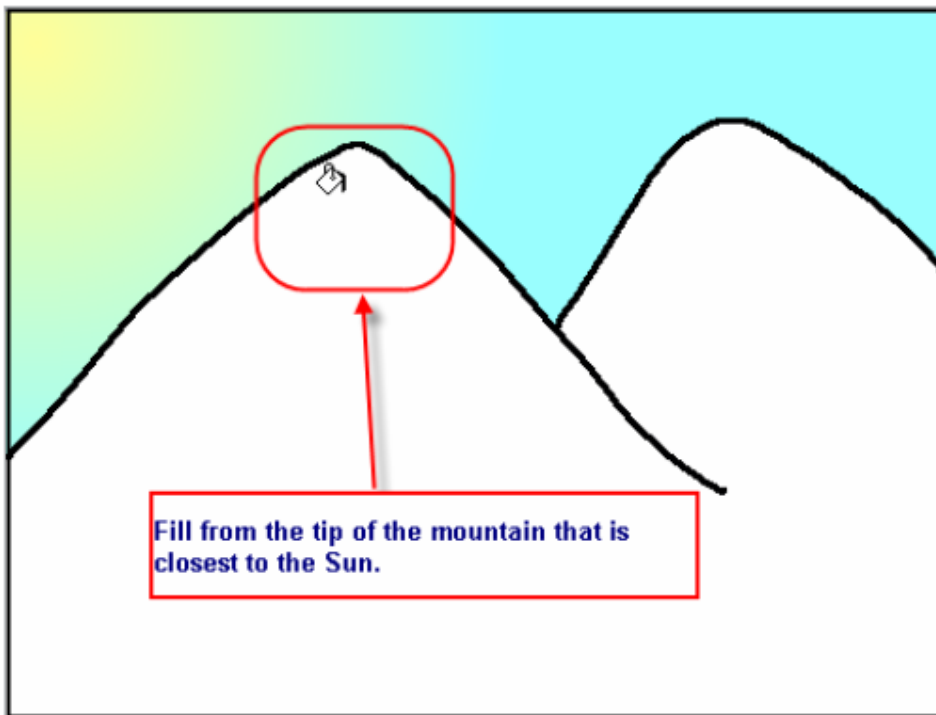
After filling, the Canvas should look like this.



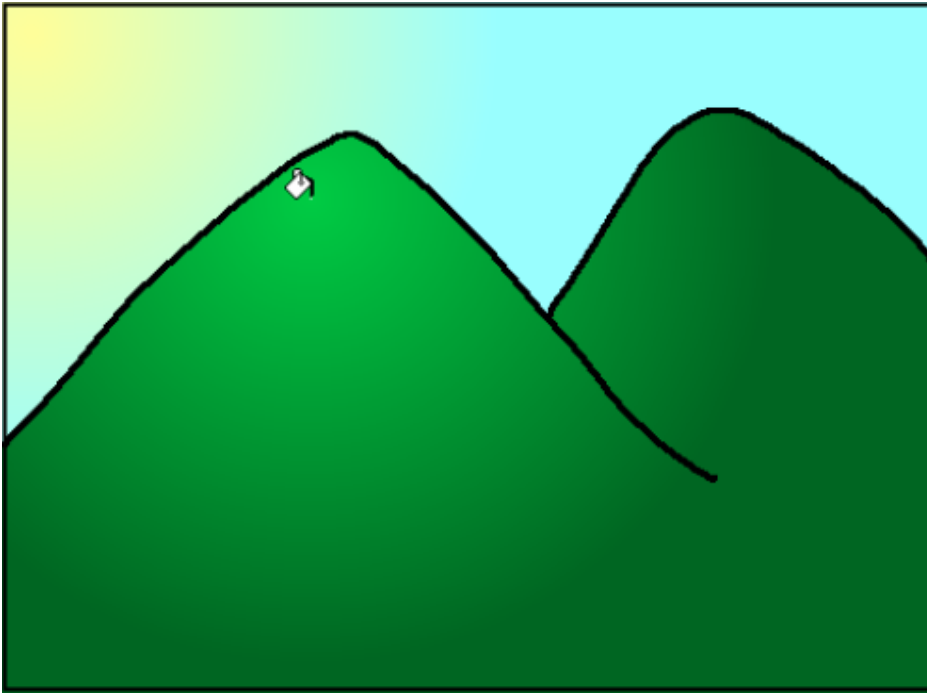
Next, we will fill the mountains in the same way. We will first select two color tones: light green for mountaintops, and darker green for area further away, such as foot of the mountains or the mountain in the shadow of the first mountain. But this time, we will use Top Gradient Fill.



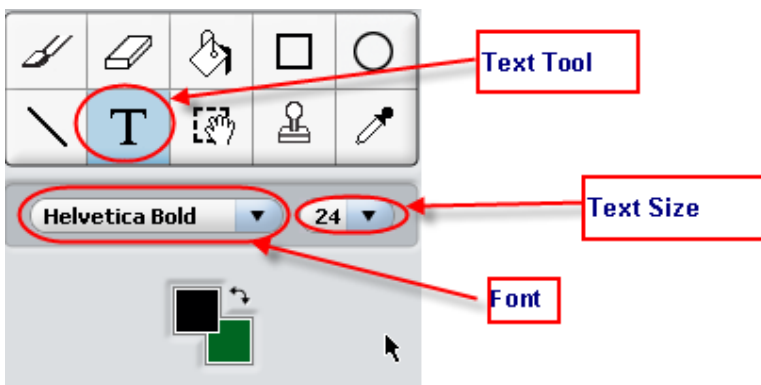
Bring the paint bucket icon to the tip of the mountain that is closer to the Sun and click to fill.



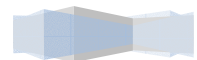
This is the result of the color fill.



To make it obvious to viewers that this is the village scene, I also added story line in text to the Canvas. Select Text Tool and click anywhere in the Canvas.

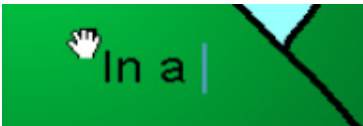


To start typing, click Text Tool, and just start typing. You would see a tiny black square box. It's the *text handle*.





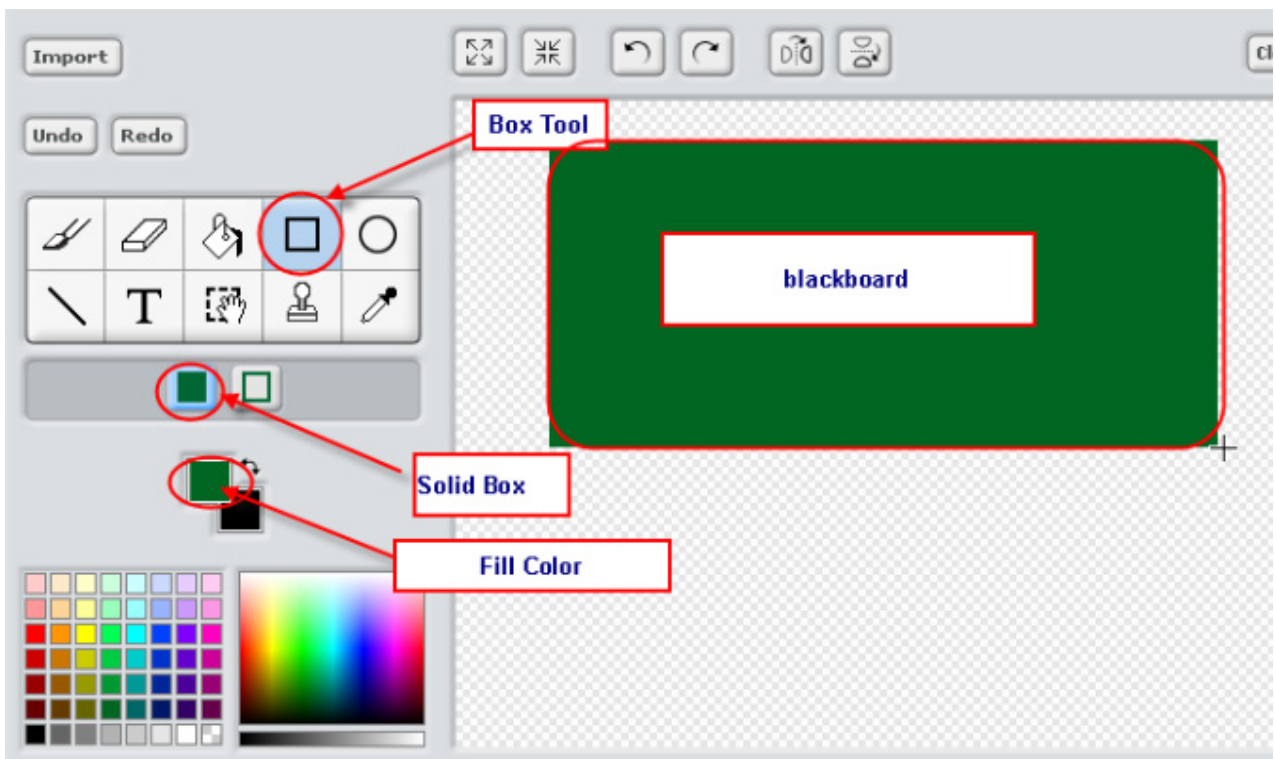
To move the text, move your mouse over to where the text handle is and drag it.



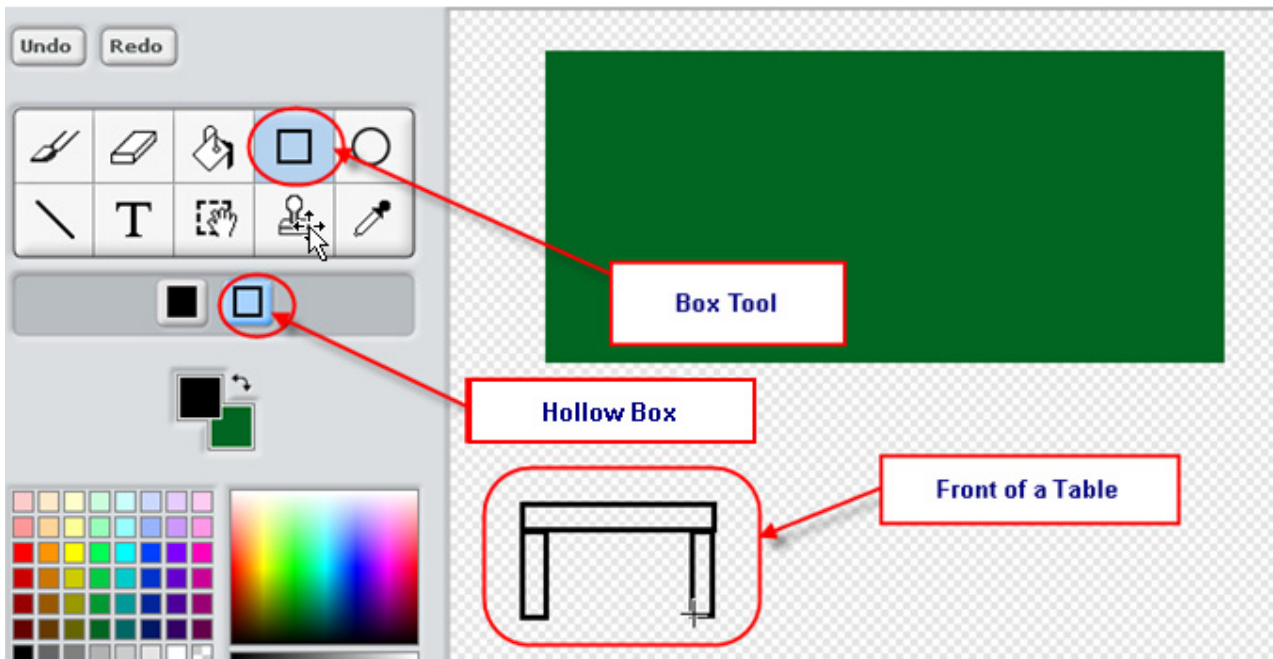
#### **Step 4: Creating the School Background**

For school background, we will draw a old-school blackboard and several wooden chairs.

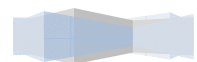
Select Box Tool and Solid Box mode. Make sure the Fill Color is dark green. Draw a solid green box.

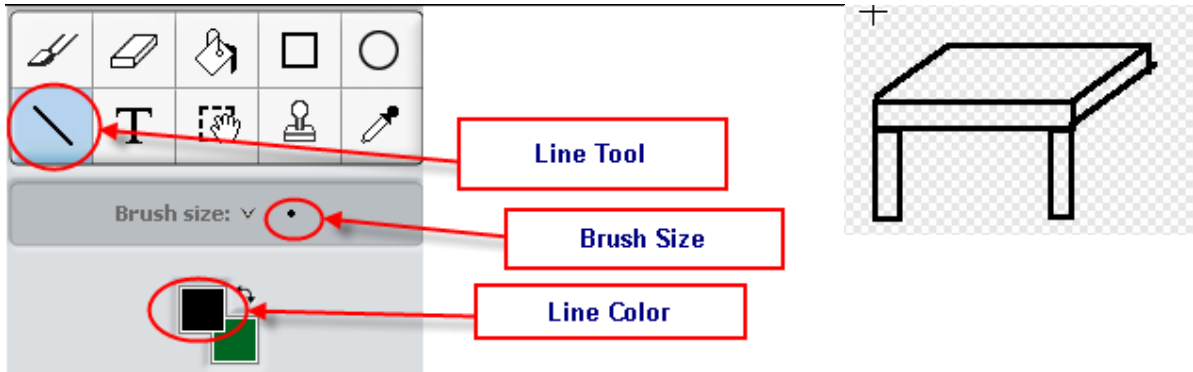


Switch color so the main color will be black. Then select Box Tool and Hollow Box mode. Create front of a table like shown below: one horizontal rectangle on top as side of the table and two vertical rectangles as legs.

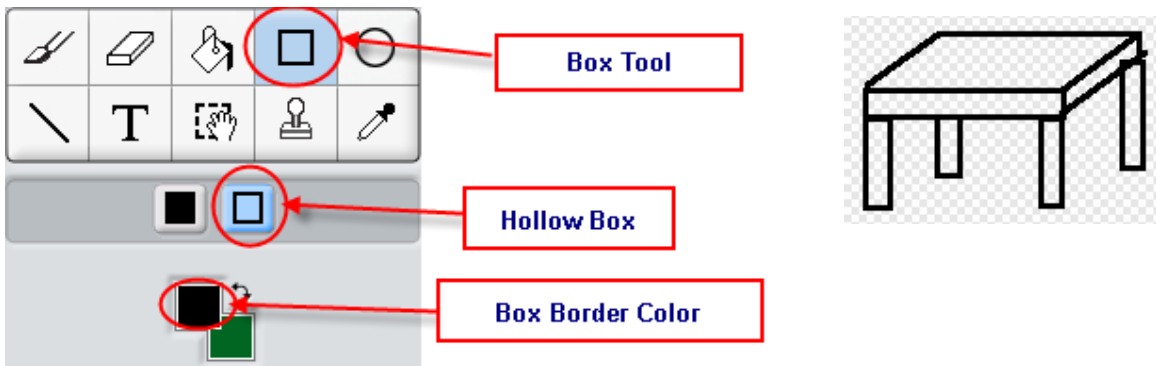


Draw the table top and the side using Line Tool.

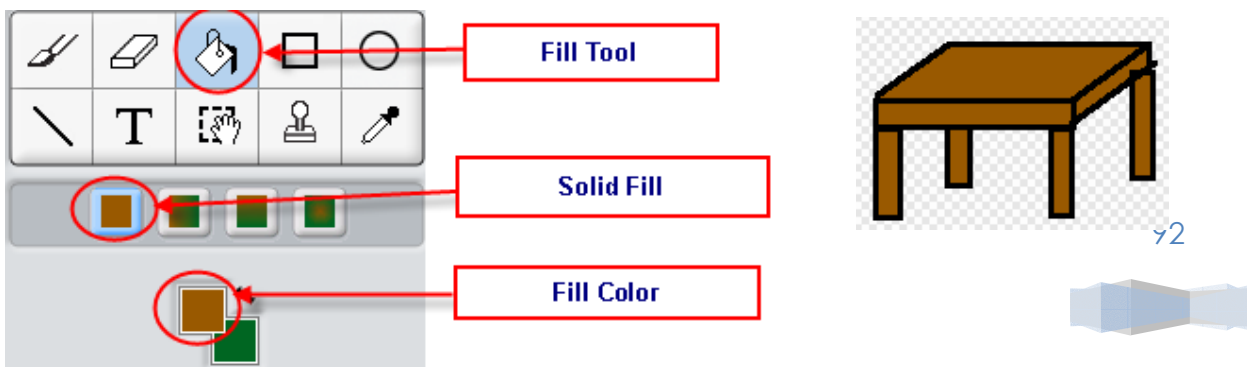




Then finish up the table outline by adding another two legs using Box Tool in Hollow Box mode.

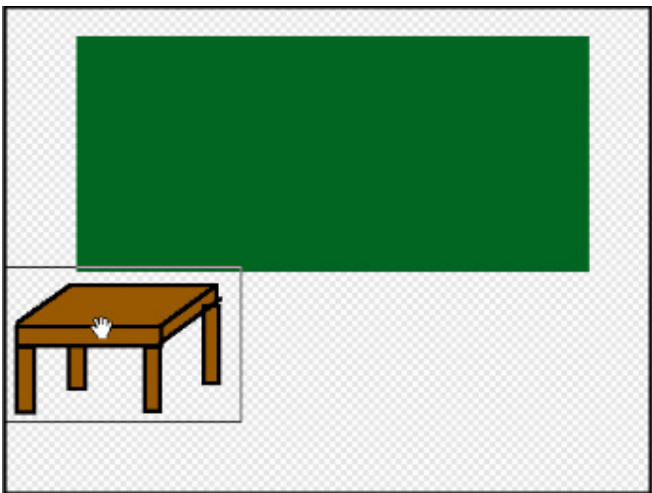
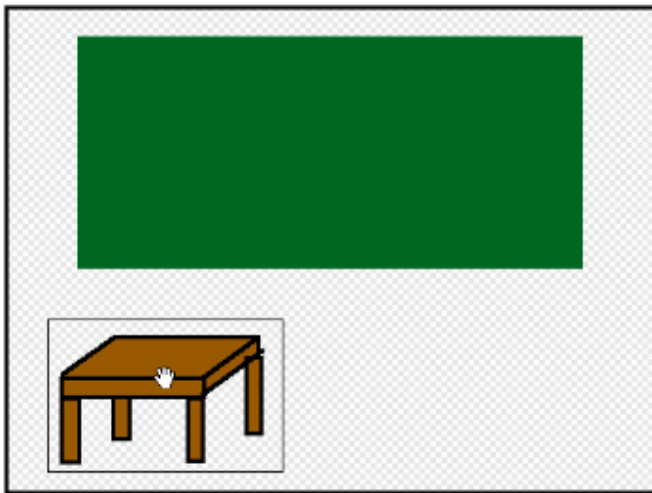
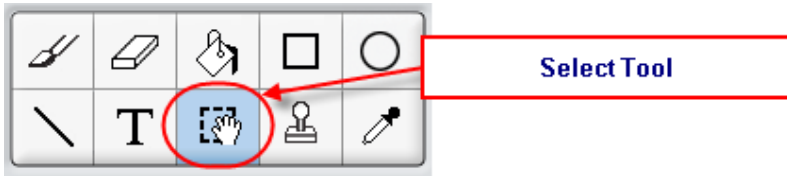


Finally, fill the table with wooden color using Fill Tool with Solid Fill.

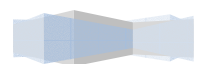


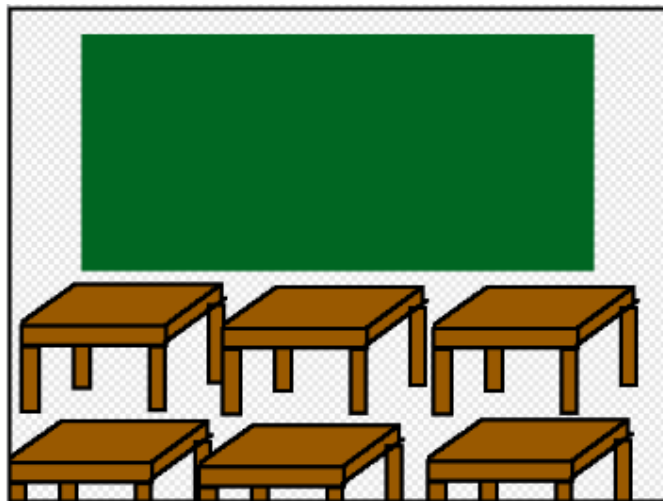
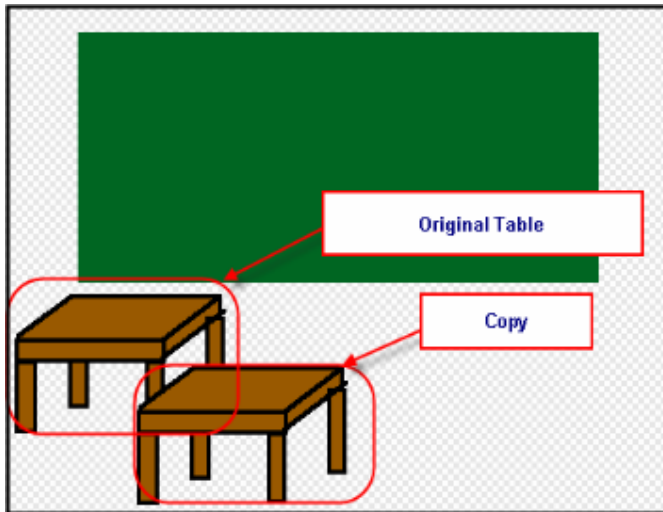


To move this table, use Select Tool to draw a box around the table and drag it.



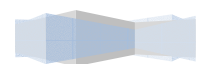
Select Stamp Tool and draw a box around the original table to make a copy. Drag the copy to where you like and click to drop the copy. Repeat until you have enough tables that this look like a classroom. I know there are not chairs but hey, tables are good enough.



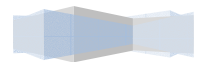
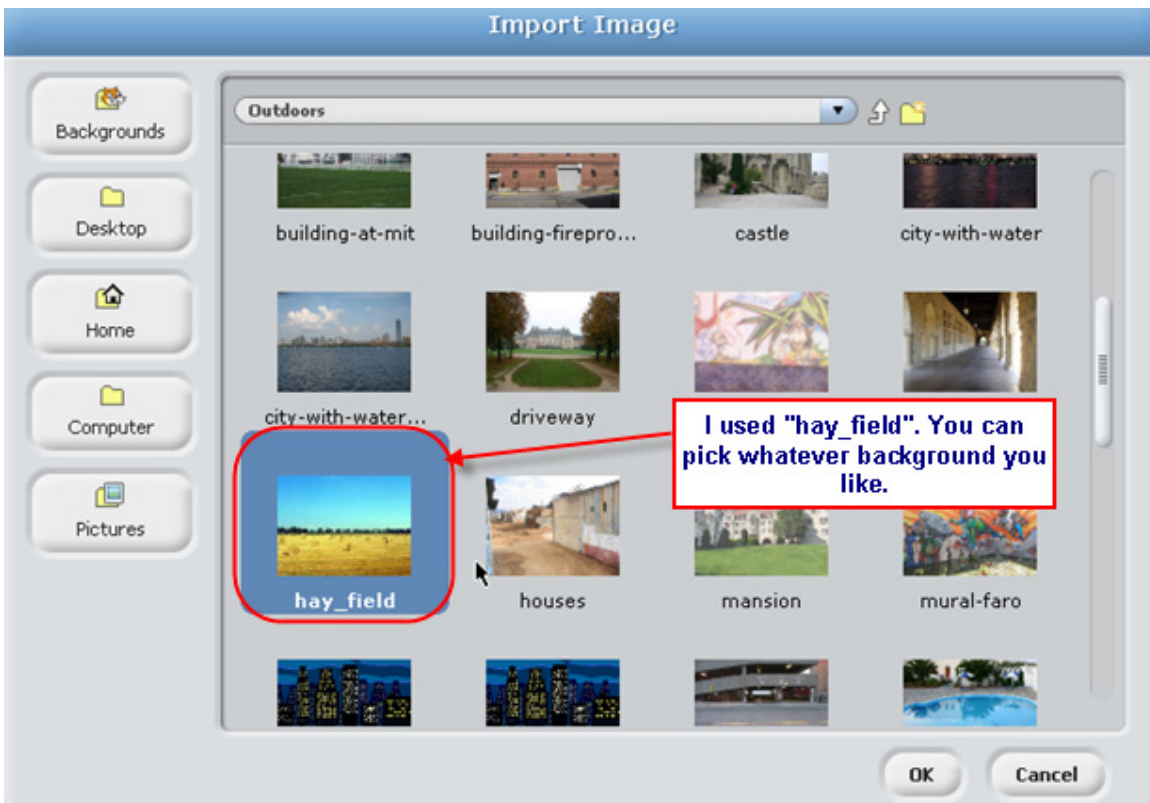
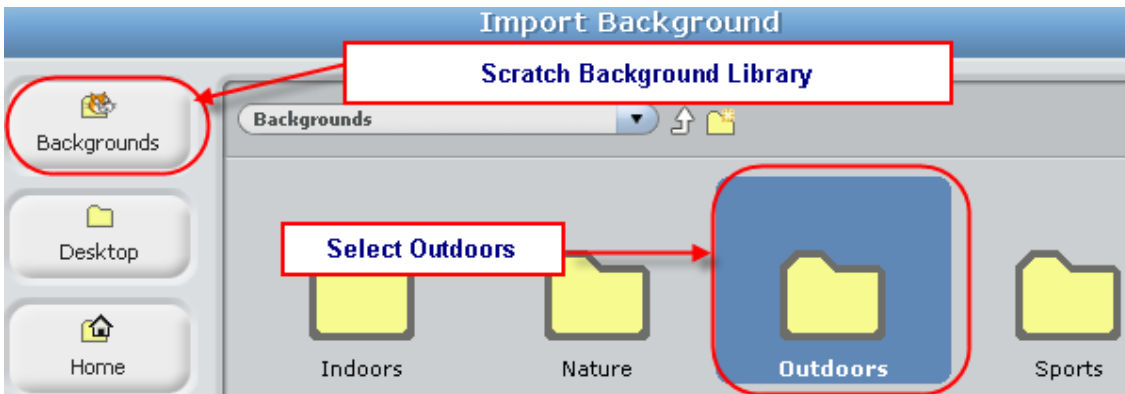


### **Step 5: Importing the Training Ground Background**

Training ground background is the easiest of all. We will just import it from Scratch Background Library and rename it "training field".

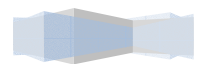
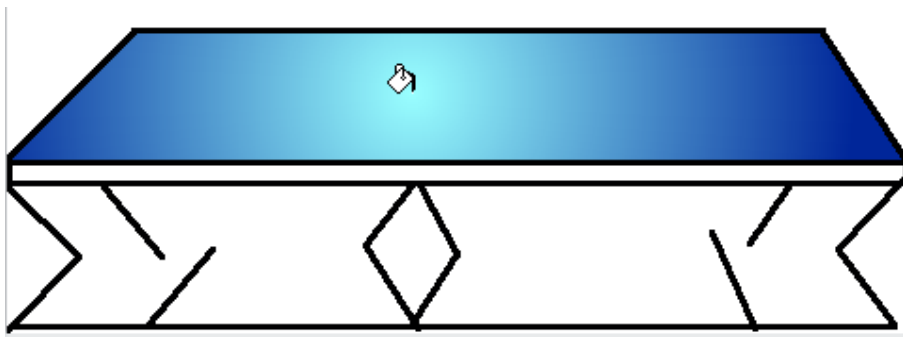
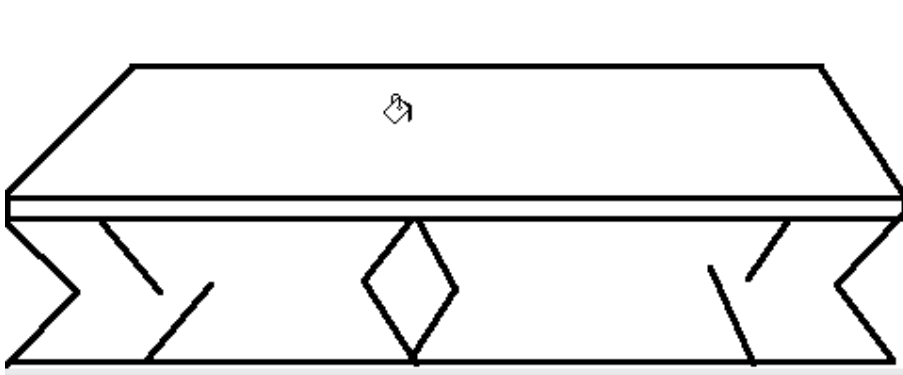
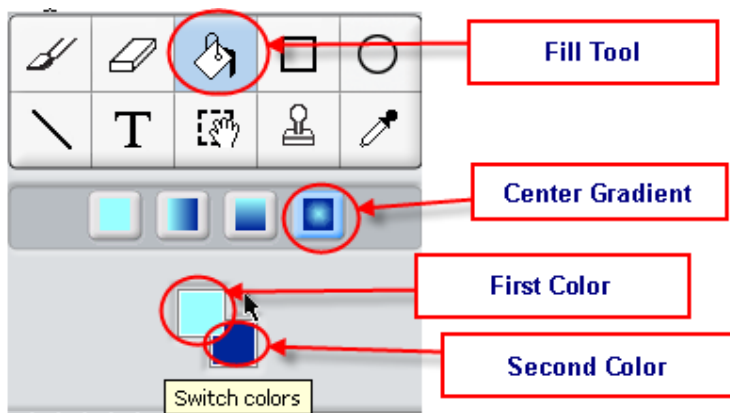


New background: Paint **Import** **Import from an image**

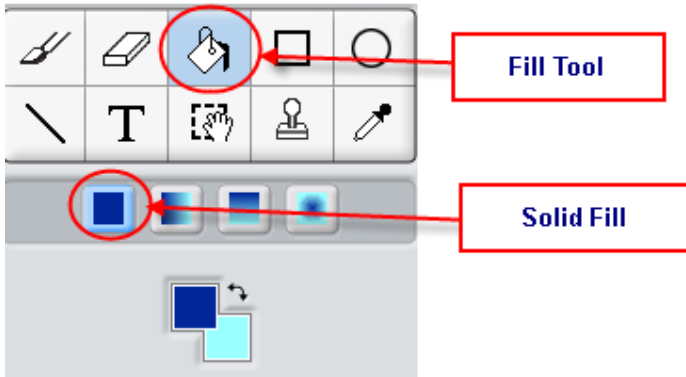


## **Step 6: Creating the Fighting Stage Background**

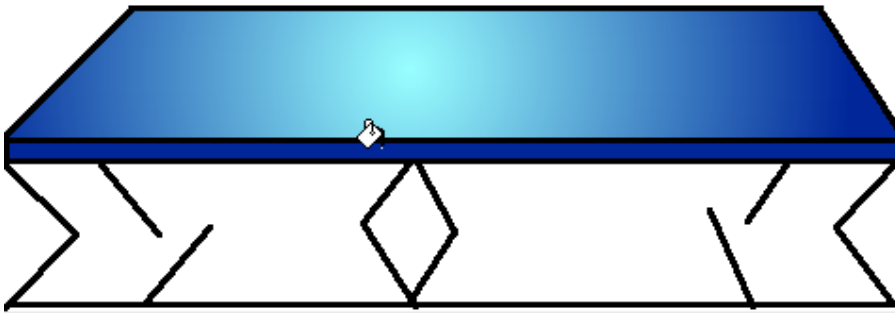
First use Box Tool and Line Tool to create a fighting stage. Select the Fill Tool and the Center Gradient Fill. Then select light blue as first color and dark blue as second color.



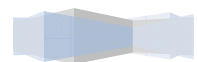
Switch two colors and then select Solid Fill.

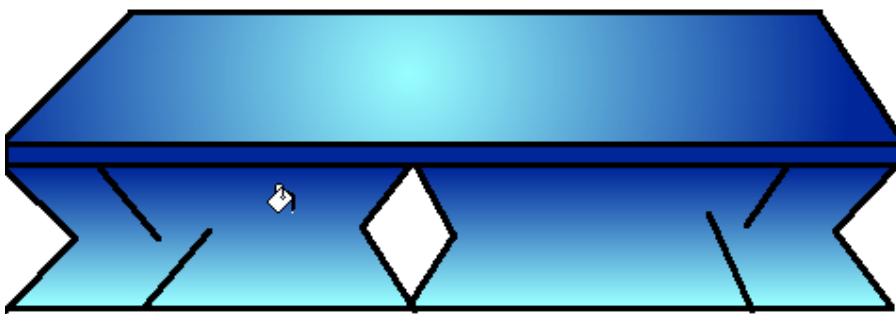
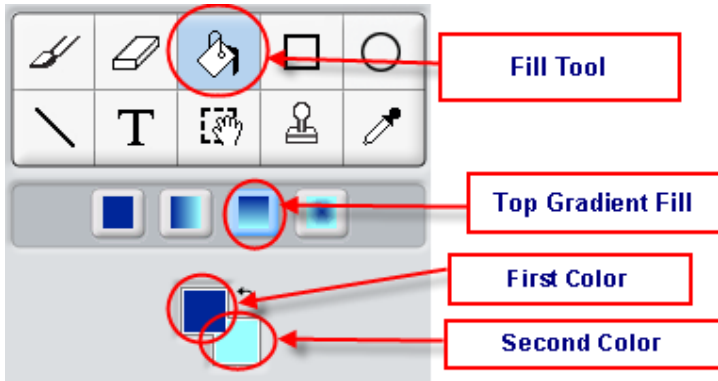


Fill the rectangular side with solid blue.

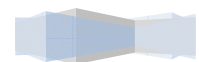
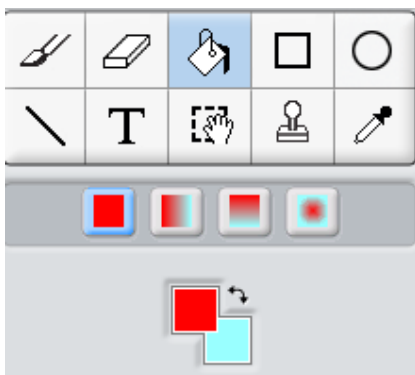


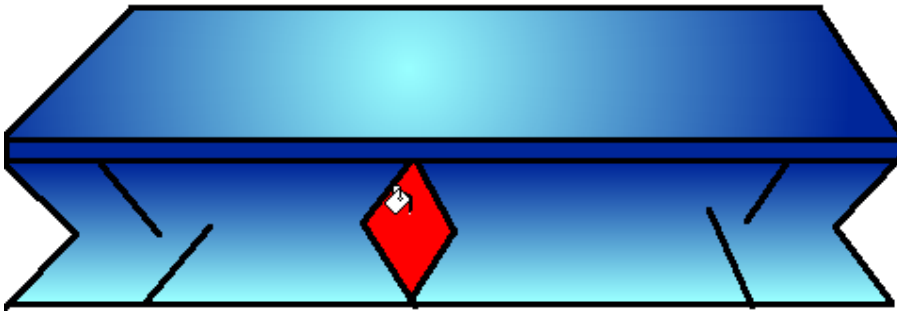
Fill the rest, except the diamond, with blue gradient fill. Use Fill Tool in Top Gradient Fill mode.



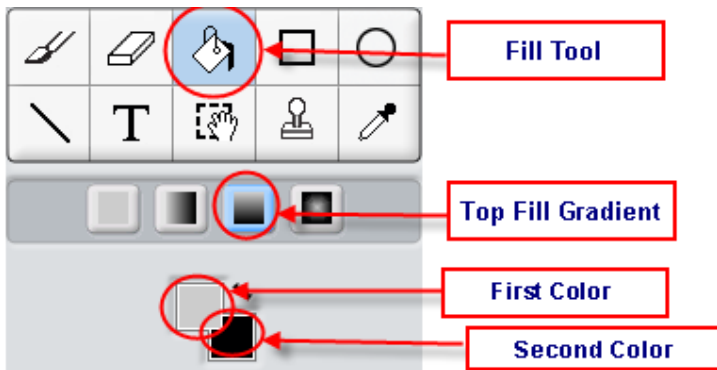


Change first color to red and fill the diamond.

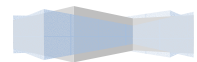


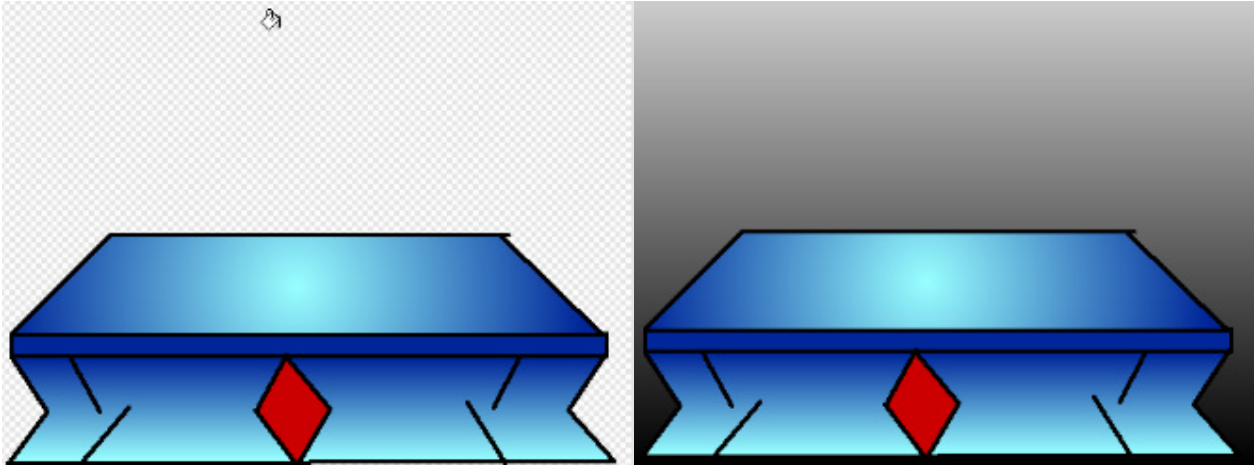


Next we will create shadowing effect by using light grey and black. Select light grey as first color and black as second color. Then select Top Fill Gradient.



Fill from the top of the Canvas.





We are done for this lesson. Go take a break and we will add scripts in next lesson for scene transitions and sprite interactions.





## LESSON 11: STORIES TO ANIMATIONS PART II

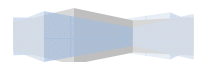
In this lesson, I will show you how to add scene transition to your animation using broadcast message and "wait" control blocks.

### *Step 1: Get to Know Your Stage*

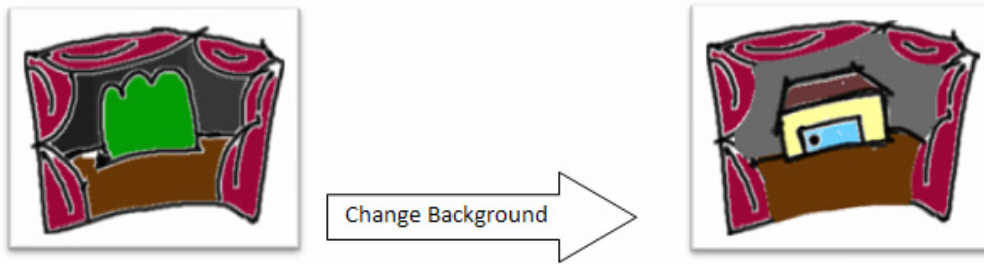
A Stage is like a Sprite. A Stage can change its look, just as a Sprite. A Stage can also think (such as doing math) and feel (such as checking if a sprite is touching another on Stage), just like a Sprite. Both Stages and Sprites can send and receive "messages", which will be covered in this lesson.

The main difference between a Stage and a Sprite is that a Stage cannot move. Other than that, a Stage is just like a Sprite. I am already repeating myself but the following illustration should make this painfully clear.

A Sprite changes costumes.



A Stage changes backgrounds.



Both a Sprite and a Stage can think and feel.



Moreover, both Stages and Sprites can send and receive "messages".

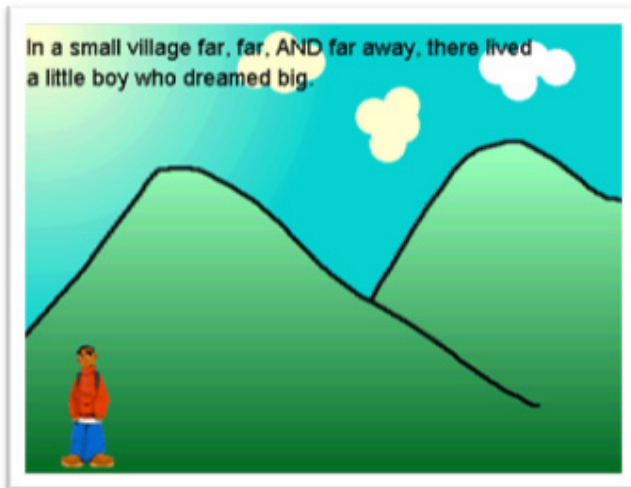


Now that we have gotten to know our stage a bit better, next we will create backgrounds for the stage and

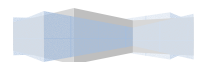
*Step 2: Put Storyboard Together*

First, we need to design the story board.

Scene 1 (the **village** background): Bobby stood in front of a remote village



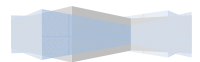
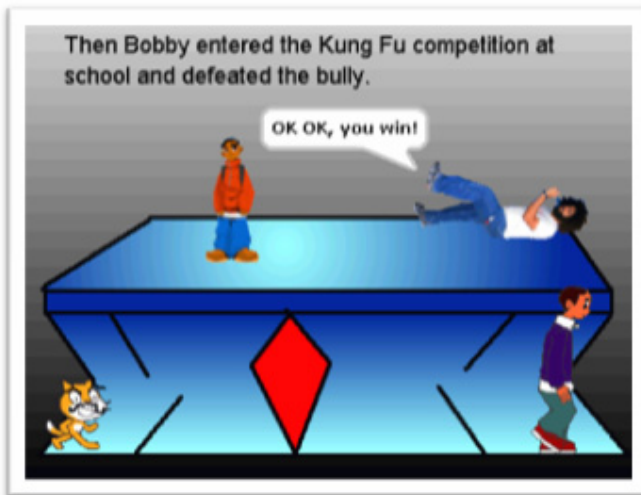
Scene 2 (school): Bobby was in class, being teased at.



Scene 3 (training ground): Master Meow trained Bobby.



Scene 4 (fighting stage): Bobby fought with the Bully and won!



### *Step 3: Create Scene Transition Using Broadcast Messages*

Next, we will use Broadcast Messages to let actors (sprites) know when and where to move.

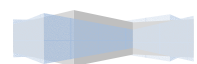
#### ***Introducing Broadcast Messages***

Each scene has a set of sprites and each sprite needs to know where to go in a scene. Moreover, some sprites do not show up in some scenes. To let sprites know when and where they will go as scenes change, we will use Broadcast Messages.

A *Broadcast Message* is a message sent by a Sprite or a Stage, and can be received by all Sprites. We will let Stage send out Broadcast Messages. Click Control button from the Tool Kit.

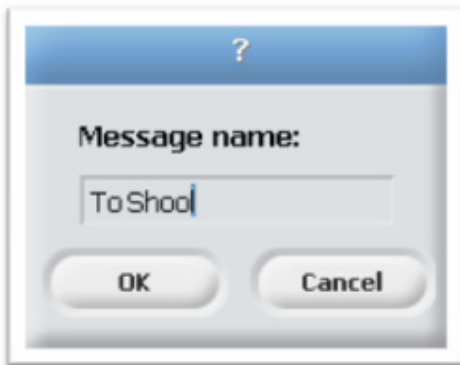
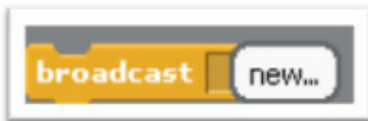


Drag out a "broadcast" block from Control Tool Kit and drop it in Stage's Script Editor. Click the little down arrow and select a message or create a new message.

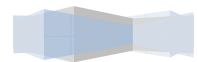




To create a new message, click the "new..." from the popped bubble and then enter "school" as message name. Click "OK" to save this new Broadcast Message.



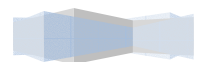
In total, we need these messages: "To School", "To Village", "To Training Ground", "To Fighting Stage", "Done Laughing", and "Done Training".

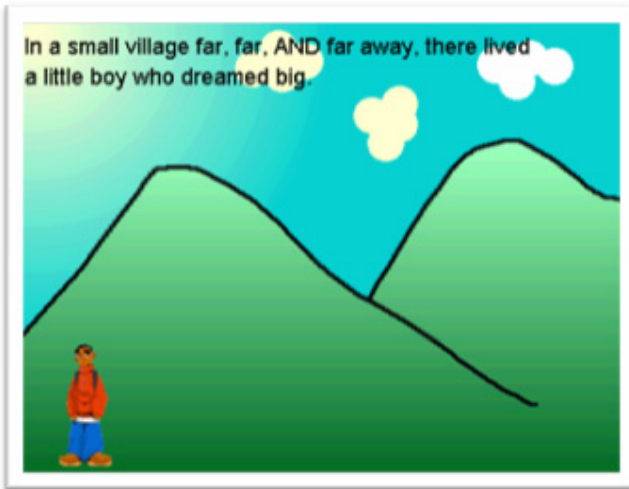
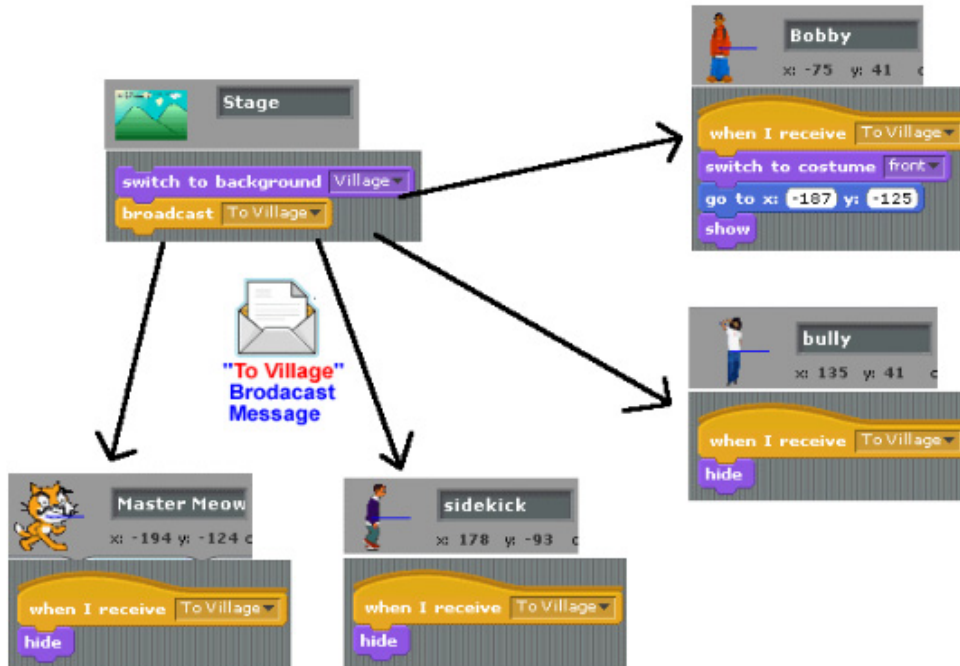




## VILLAGE SCENE

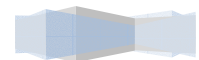
Stage would change to "Village" background and sends out a "To Village" broadcast message (to everyone). Upon receiving the "To Village" message, Bobby shows up on Stage; all other sprites just hide.





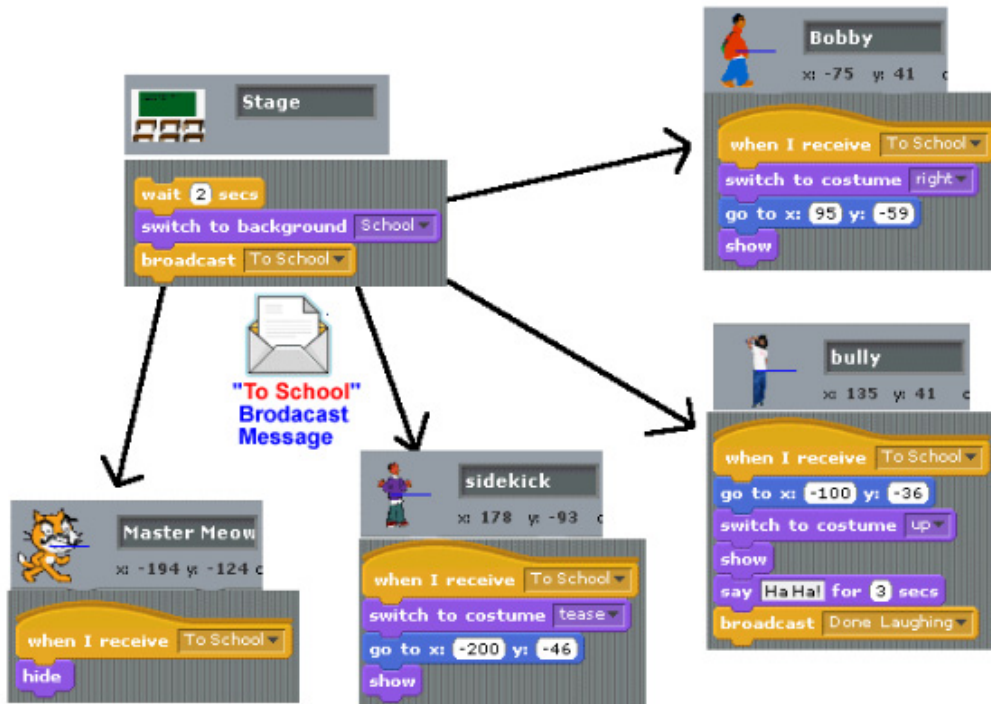
## SCHOOL SCENE

Then the Stage would wait two seconds, change its background to "School", and then broadcast a "To School" message. Upon receiving the "To School" message, Bobby, bully, and sidekick all show up on Stage. They also would change to appropriate costumes: Bobby would change to "right" costume, bully to "up", and sidekick to "tease".

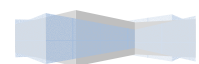




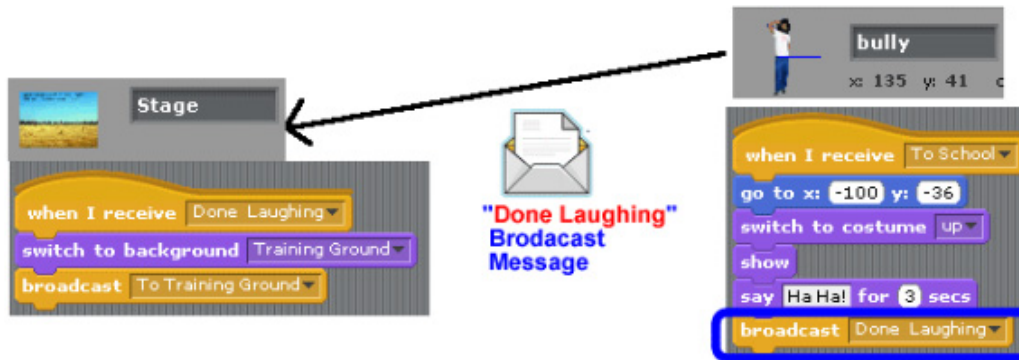
Moreover, the bully Sprite would say "Ha Ha" for two seconds. I know it's not really serious bullying, but hey, verbal abuse is abuse nevertheless. Once finished laughing, the bully would send out a "Done Laughing" message.



The result snapshot of the Stage of this scene looks like this:

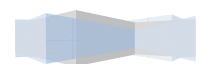
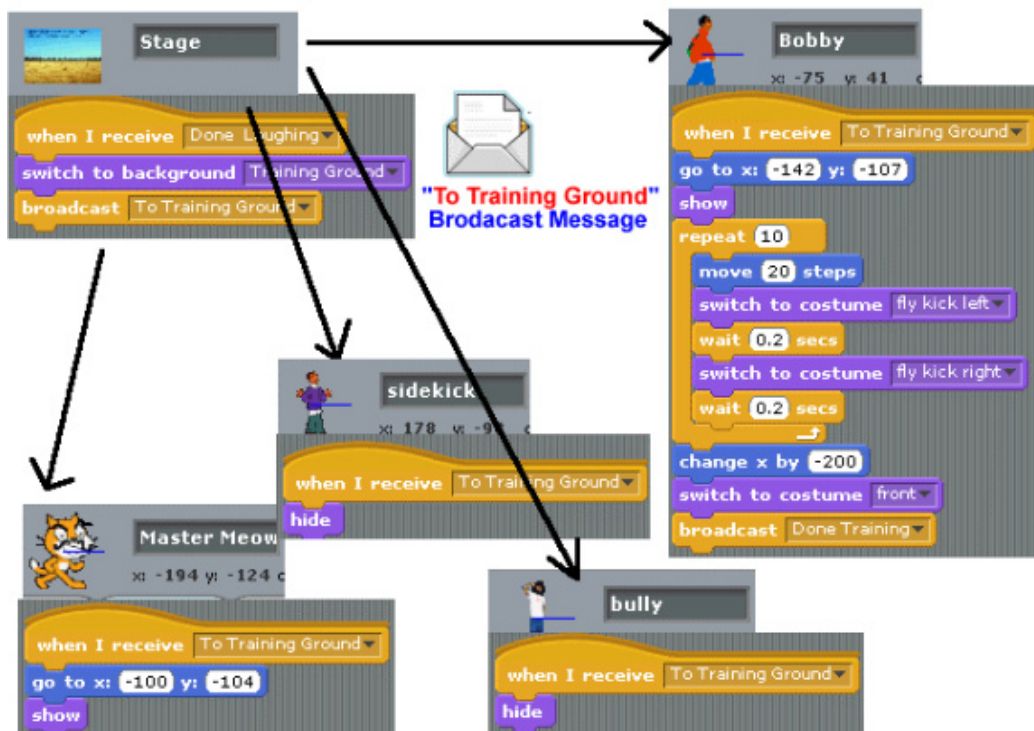


Upon receiving this "Done Laughing" message, Stage would change to "Training Ground" background and broadcast a "To Training Ground" message.



### TRAINING SCENE

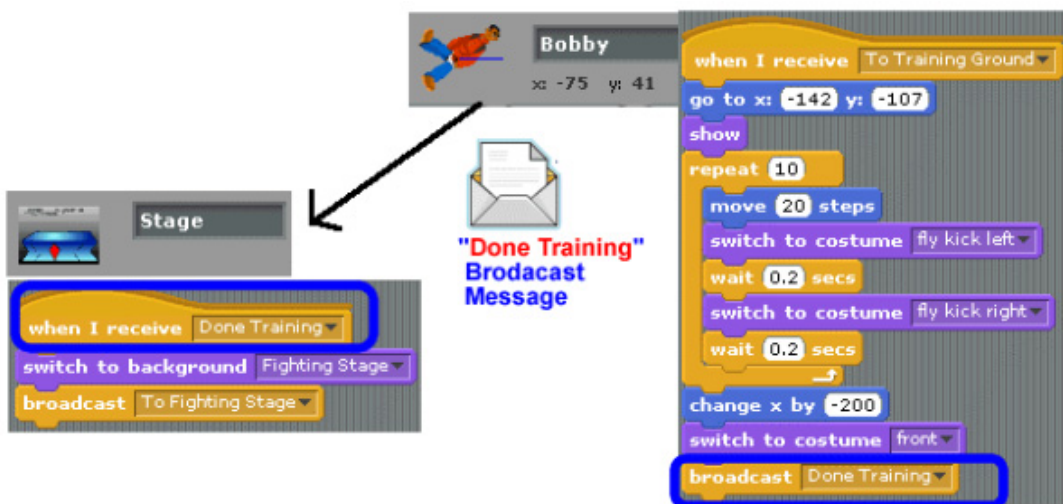
At the training scene, Bobby would spin-kick across the field for several times and then sends out a "Done Training" message.



The result snapshot of the Stage of this scene looks like this:

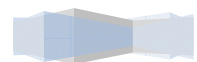


Upon receiving this "Done Training" message, Stage would change to "Fighting Stage" background and broadcast a "To Fighting Stage" message.

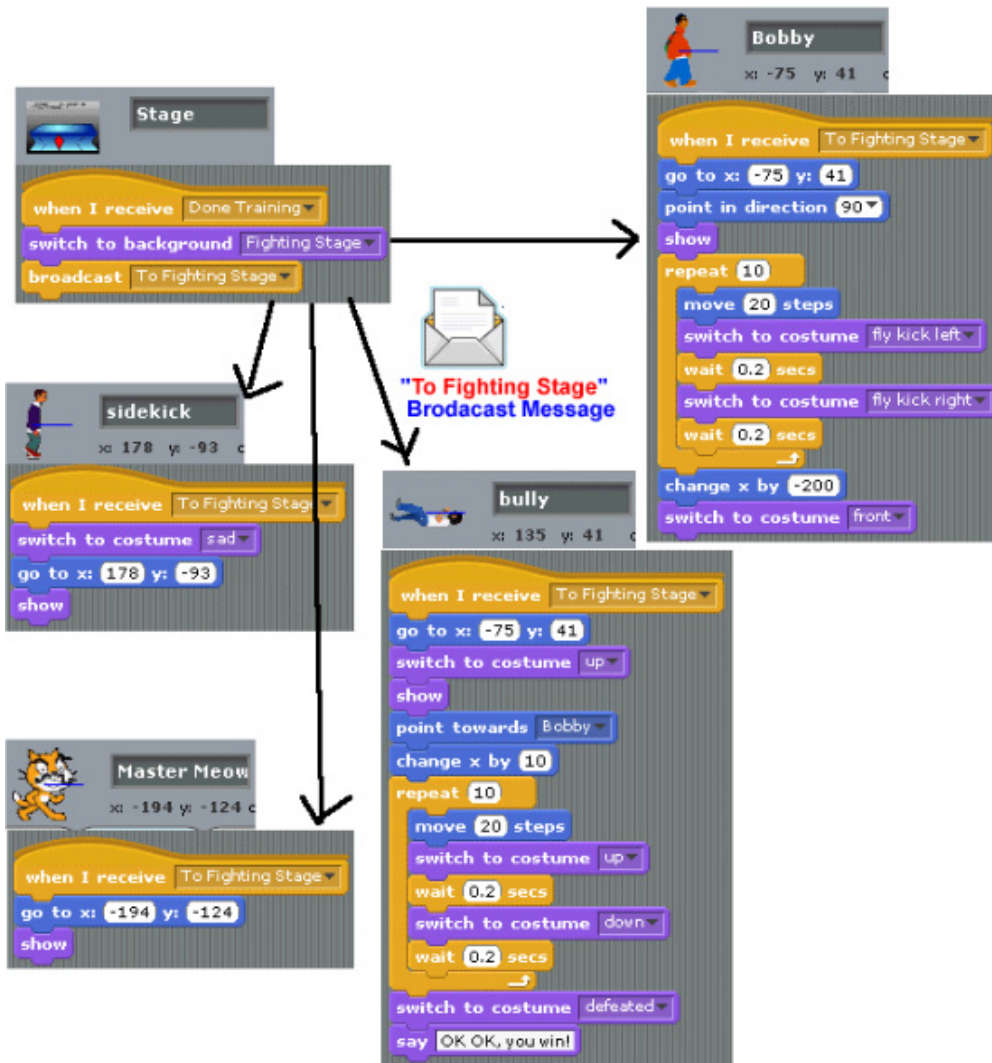


## FIGHTING SCENE

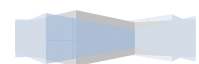
At the fighting scene, Bobby would kick (by repeatedly changing between "fly kick left" and "fly kick right" costumes) Bully's until Bully

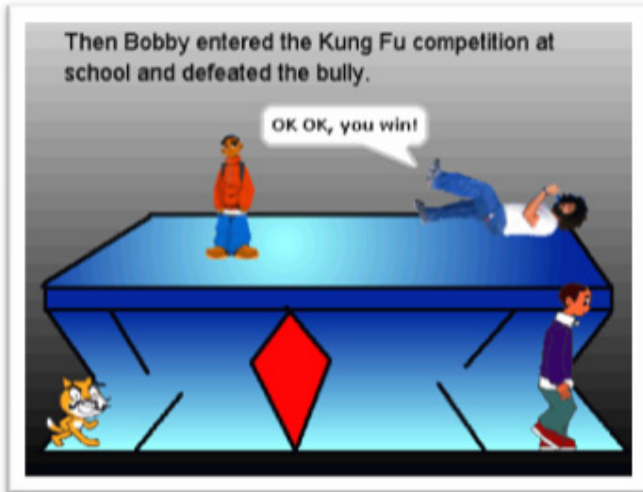


is defeated. To animate the bully being kicked, just repeatedly change the bully's costumes between "up" and "down". Once the bully has been kicked badly enough, he would say "OK OK You Win".



This is the snapshot of the Stage at the Fighting Scene:



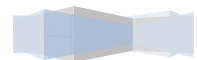


### 3: Put Together Scripts for Each Sprite

Below are snapshots of scripts for all sprites. The Stage's scripts handle changing backgrounds.



Master Meow's scripts are very easy because he just shows and hides.

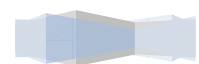




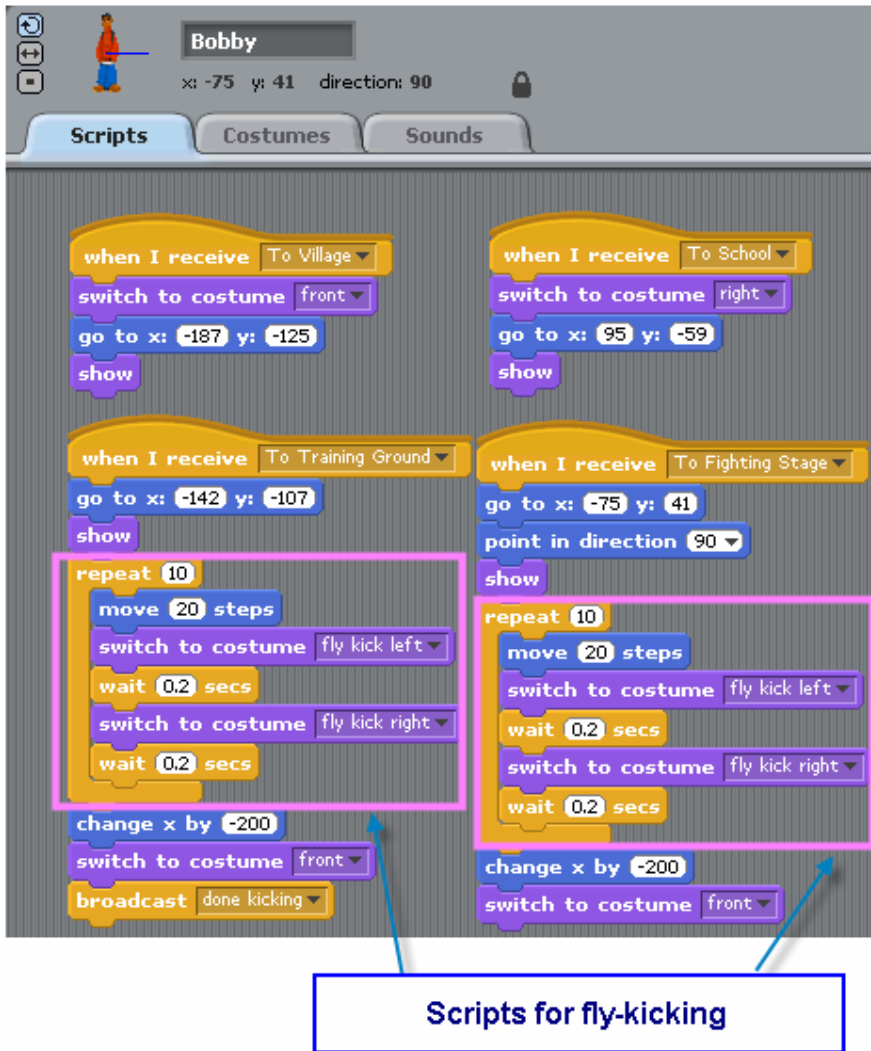
Sidekick's scripts are also very easy because he just shows and hides.



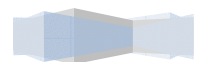
Bobby's scripts involve showing, hiding, as well as fly-kicking.

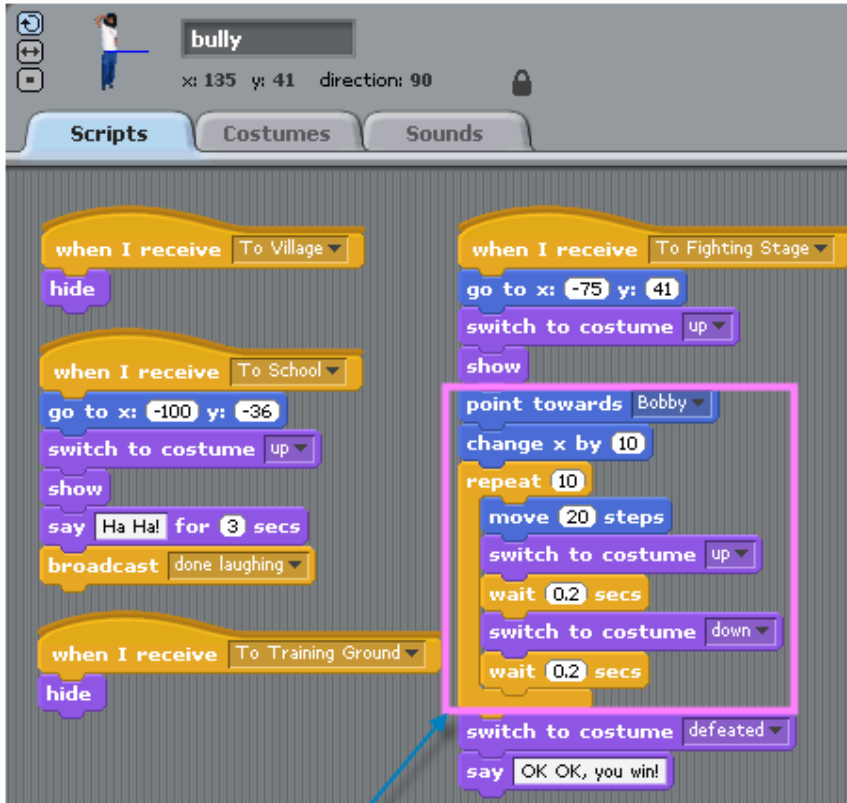






Besides showing and hiding, the bully's scripts involve laughing and being kicked down by Bobby.





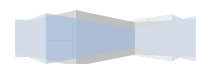
Fighting with Bobby and Being Kicked Down

Download the result project for this lesson [HERE](#).

You may download the PDF version of the lesson [HERE](#).



TEST TIME: Whew! We are done. Go ahead and test your first animation. This concludes Lesson 9.





## LESSON 12: SCROLLING PLATFORM GAME – Game

### Design

In lesson 10 through 14, I will cover what you need to learn to make a mini Super Mario game.



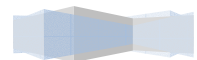
We will cover the following:

- Designing a Game (Lesson 10)
- Creating Game sprites (Lesson 11)
- Adding Game Rules to Game Sprite (Lesson 12)
- Changing Background when Mario Moves (Lesson 13)
- Managing Score and Levels (Lesson 14)

After following these classes, you should feel very comfortable creating your own game. Let's start!

Here is a basic table that we need to fill in:

Game Name	Mini Mario
Sprites	List of sprites: name, costumes, sounds, movements
Sprite Interaction	How sprites interact in this game
Backgrounds	The list of background doodles



	here
Scores and Levels	The score counter rule and level advancement rule here

Let's expand each item to its own table:

<b>GAME NAME</b>	
------------------	--

<b>SPRITES</b>	<b>Name</b>	<b>Look</b>	<b>Sound</b>	<b>Movement</b>

<b>SPRITE INTERACTION</b>	<b>Sprite1</b>	<b>Sprite2</b>	<b>Interaction</b>

<b>BACKGROUNDS</b>	<b>Name</b>	<b>Look</b>

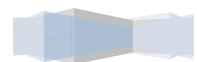
<b>SCORE AND LEVELS</b>	<b>How</b>	<b>How many points</b>

<b>LEVELS</b>	<b>Level</b>	<b>Requirement</b>	<b>Starting Background</b>	<b>Ending Background</b>

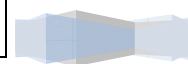
You may download the PDF form of above table [HERE](#).


Let's fill in each table for this game:










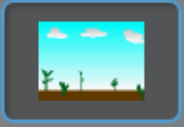
<b>GAME NAME</b>	Mini Mario
------------------	------------

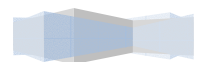



Name	Look	Sound	Movement/ Action
Mario 	-Walk1 -Walk2 -Jump Up -Jump Down	-Jump -Die -Enter -Score -Grow -Shrink	-Walk -Jump -Die -Grow -Shrink
Fruit Platter 	-Delicious	None	None
Princess 	-Surprised -Happy	-Happy tune	-Jump up and down for joy
Brick 	-Regular -Cracked	-Crushed	-Crushed
Bat 	-Fly1 -Fly2	-Wing flap	-Fly
Coin 	-Spin1 -Spin2 -Spin3	-Bling-bling	-Spin



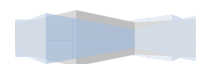
Crab 	Walk1 Walk2	-None	-Walk
--	----------------	-------	-------

Sprite1	Sprite2	Interaction
Mario 	Brick 	If Mario hand hits Brick, then Brick would crack.
Mario 	Coin 	If Mario touches Coin, Coin would disappear and Mario would score 1 point.
Mario 	Crab 	If Mario touches Crab, he would die.
Brick 	Coin 	When Brick cracks, Coin would show spinning above Brick
Mario 	Stage 	When Mario hits the bottom of the stage, he would die.



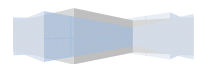
Name	Description
Level1_1 	Solid ground, several bricks (to be added from sprites), crabs as enemies
Level1_2 	Holes in ground, several bricks, crabs as enemies
Level1_Passed 	Fruit platter
Level2_1 	Solid ground, several bricks, bats as enemies
Level2_2 	Holes in ground, several bricks, bats as enemies
Level2_Passed 	Princess

Level	Requirement for this Level	Starting Background	Ending Background



1	When the game is started	Level1_1	Level1_2
2	When Mario grabs the Fruit Platter in Level 1	Level2_1	Level2_2


Whew!! We are done for the design stage. This may seem a lot of work but it's always a good idea to lay out the design in as much detail as possible for your masterpiece, whether it is an animation or a game. Once you take the time to design your game, the building part should be straight forward.





## LESSON 13: SCROLLING PLATFORM GAME – SPRITES

In Lesson 2, we will create all necessary sprites for Mini Mario game. Especially, we will focus on Mario and show how to make him walk, jump, and squat.

### 1. Create Fruit Platter sprite

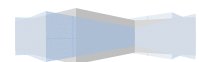
Name	Look	Script	How to Create
1. Fruit Platter 	<i>-Delicious</i>	None	Import costume from library

Import the Fruit Platter image by clicking  then select

COSTUMES->THINGS. Scroll to select fruit\_platter  and click OK to save. Rename the costume as *Delicious*. Save this sprite as Fruit Platter.


### 2. Create Princess sprite


Name	Look	Script	How to Create
2. Princess	<i>-Surprised</i> <i>-Happy</i>	-Jump for Joy	1. Import first costume from library 2. Create another




			costume based on the first costume 3. Create Jumping for Joy script
---	--	--	--

Import the image from Scratch library by clicking  then select

COSTUMES->PEOPLE. Scroll to select squaregirl  and click OK

to save. Rename this costume as *Surprised*  and modify


Surprised to become *Happy* .

Add a script to keep changing to next costume.



Save this sprite as Princess.

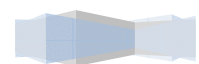
### 3. Create Coin sprite

Name	Look	Script	How to Create
3. Coin 	- <i>Spin1</i> - <i>Spin2</i> - <i>Spin3</i>	-Spin	1. Create custom costumes 2. Import sound 3. Add Spin script

Create three costumes:

*Spin1* , *Spin2*,  and *Spin3* .

Add a script to create the spinning effect:







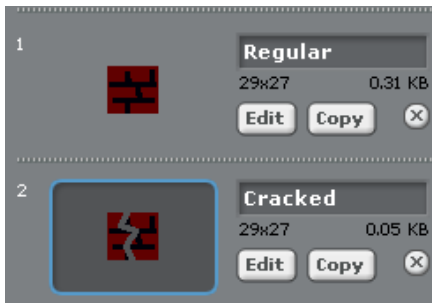


Save this sprite as Coin.

#### 4. Create Brick Sprite


Name	Look	Script	How to Create
4. Brick 	-Regular -Cracked	-Crushed	1. Create custom costumes 2. Import sound 3. Create Crushed script

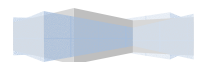
Use  to create a new Sprite which has two costumes: *Regular* and *Cracked*.



Save this sprite as Brick.

#### 5. Create Bat Sprite

Name	Look	Script	How to Create
5. Bat 	-Fly1 -Fly2	-Fly	1. Import both costumes from the library 2. Import sounds 3. Add Fly script



Import the image by clicking  then select




COSTUMES->ANIMALS. Scroll to select *Fly1* and *Fly2*. Save them as *Fly1* and *Fly2*.

Add a script to simulate the wing flapping:

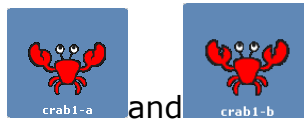


Save this sprite as Bat.

### 6. Create Crab sprite

Name	Look	Script	How to Create
6. Crab 	-Walk1 -Walk2	-Walk	1. Import Walk1 from library 2. Create Walk2 based on Walk1 3. Create Walk script

Import the image from Scratch library by clicking  then select



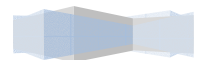
COSTUMES->ANIMALS. Scroll to select *Walk1* and *Walk2*. Save them as *Walk1* and *Walk2*.


Add a script to simulate its claws opening and closing:




Save this sprite as Crab.

### 7. Create Mario sprite



Name	Look	Script	How to Create
7. Mario 	-Walk1 -Walk2 -Jump Up -Jump Down -Squat Down	-Walk -Jump -Die -Grow -Shrink	1. Create custom costumes 2. Import sounds 3. Create action script blocks

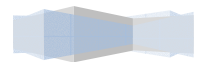
### 7.1 Create Mario's Costumes

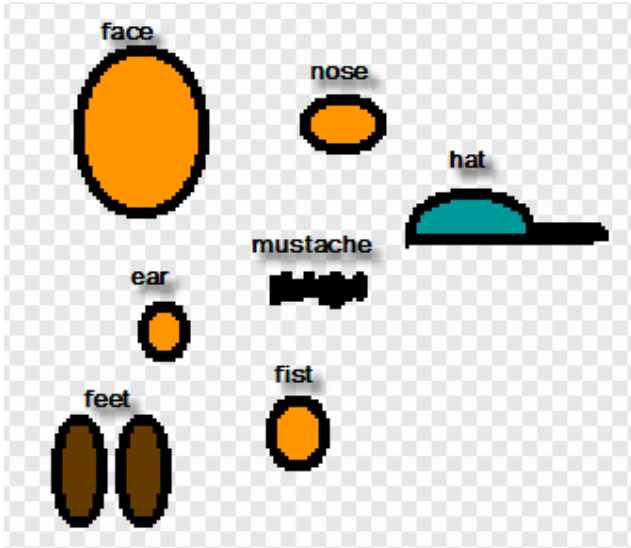
Name	Look
7. Mario 	-Walk1 -Walk2 -Jump Up -Jump Down -Squat Down

To create Mario's hat:

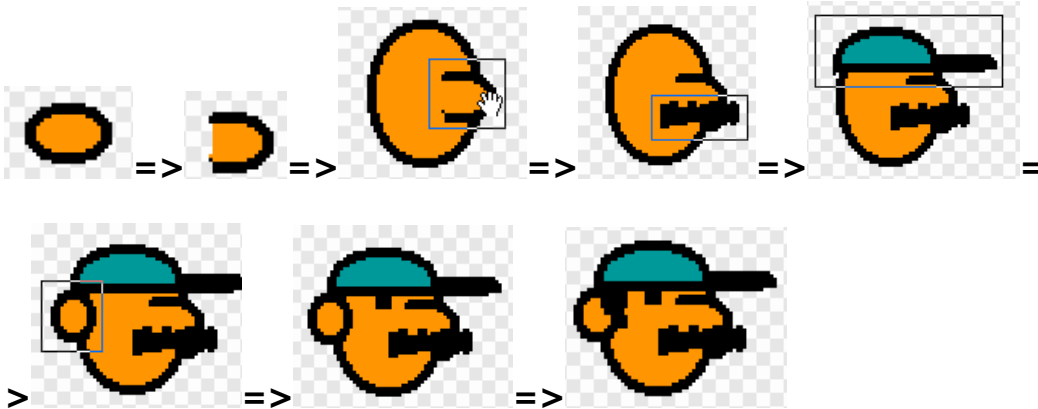


You should have all these pieces before moving on.

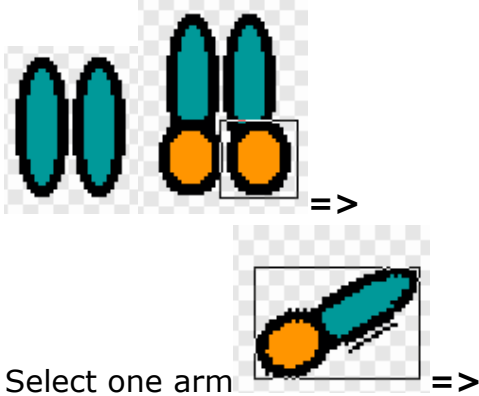




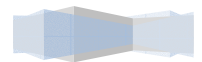
TO PUT MARIO'S HEAD TOGETHER:



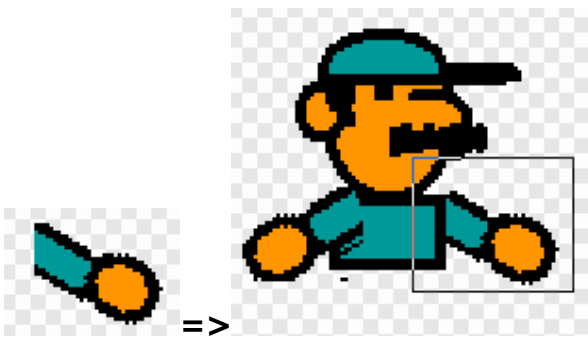
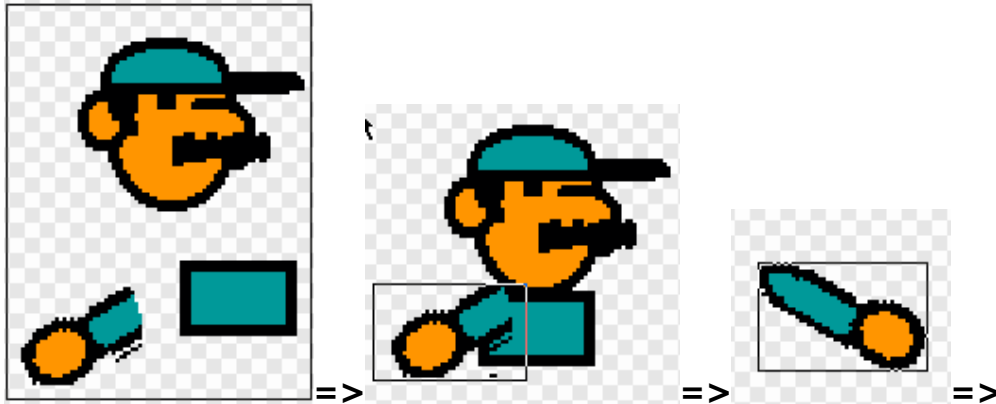
TO PUT MARIO'S UPPER BODY TOGETHER:



Select one arm =>



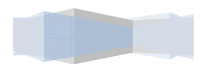
Trim the top  =>





TO CREATE MARIO'S LOWER BODY:




Select feet  =>



Use Shrink Tool  to shrink the feet  =>

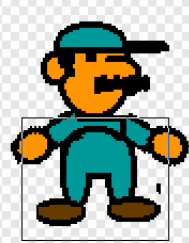
Connect feet to legs  =>

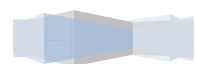
Move feet to connect to the butt  =>

Erase the line  =>

Fill with the same color 

TO PUT MARIO'S LOWER BODY WITH THE REST OF HIS BODY :

Connect lower body to the upper body  =>





Erase lines =>



Fill with color.

TO ROTATE ONE LEG:



Select a leg =>



Rotate it =>

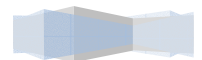


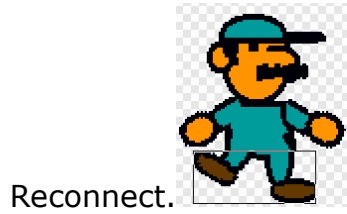
Reconnect

TO CREATE COSTUME **WALKING2**:



Select both legs. =>





I've also created other costumes: *Jumping Up*, *Jumping Down*, and *Squat Down*.



Jumping Up



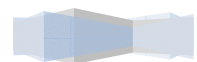
Jumping Down



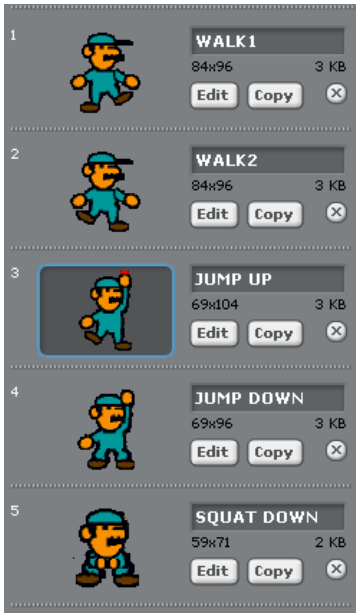
Squat Down

]

The complete list of Mario's costumes is shown below:








## 7.2 Create Mario's Scripts

Let's create **Walk** script and **Jump** Script and add more in Lesson 12,

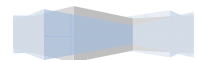
Name	Script
8. Mario 	-Walk -Jump

### CREATE WALK SCRIPT:

Add a script that makes Mario show up at a starting location when the game starts:



Add a script that makes Mario move right when the right arrow is clicked:



Add a script that makes Mario move left when the left arrow is clicked:



```
when left arrow key pressed
switch to costume WALK2
change x by -20
```

Add a script that makes Mario squat down when the down arrow is

clicked:



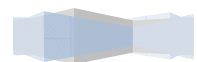
```
when down arrow key pressed
switch to costume SQUAT DOWN
```

### CREATE JUMP SCRIPT

Add a script that makes Mario jump up when the up arrow is clicked:



```
when up arrow key pressed
switch to costume JUMP UP
change y by 30
wait 0.2 secs
switch to costume JUMP DOWN
change y by -30
```



# LESSON 14: SCROLLING PLATFORM GAME – GAME

## RULES

In this lesson, I will show you how to create additional sprites and scripts to apply the game rules. We will add scripts based on Sprite Interaction Table. Moreover, we will create random motion scripts for Mario's enemies.







### 1. Create Scripts based on Sprite Interaction Rules

#### 1.1 Rule: How Mario can Score

The rule for Mario to score is as followed:

If Mario hand hits Brick, then Brick would crack. When the Brick cracks, Coin would start spinning above it. When Coin disappears, Mario gets one point.

The Sprite Interaction Table below lists all relevant interactions for this rule.

Sprite1	Sprite2	Interaction Rule
Mario 	Brick 	If Mario hand hits Brick, then Brick would crack.
Coin 	Mario 	Each time a coin appears, Mario scores 1 point.
Brick 	Coin 	When Brick cracks, Coin would show spinning above Brick

You may have noticed that the JUMPING UP costume has a bit of RED on the fist. It's not for decoration. The RED bit is added so that the Brick sprite can know when Mario's fist has hit it. I call this color (RED in this case) a *sensitive color* for the Brick, and the Brick is a *color-sensitive sprite* of the RED color. A sensitive color of a color-sensitive sprite is a color which such a sprite is sensitive to and responds to when touched. A good sensitive color for a color-sensitive sprite is a color that is not yet used by any sprite in the project. In the case of Brick, which contains burgundy and black, it can have all other colors as sensitive colors. I just picked RED. You can certainly use BLUE.



The Brick has burgundy and black.



I picked RED to be one sensitive color.



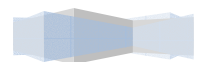
You can certainly use BLUE.

For Brick Sprite:

To simulate Brick cracking when hit by Mario, add the script as shown at right to Brick so that it would show when game starts and would cracked when touched by RED, one of Brick's sensitive color. Also, the Brick will send out a message, *coin\_show*, so that Coin knows when to appear.



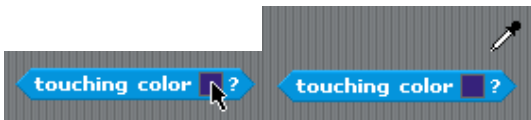
NOTE: To copy the color of the RED from Mario's fist, first select Mario's JUMPING UP costume so Mario shows up on Stage in JUMPING



UP costume.



Select Brick's script panel and drag out a "touching color ..." block. Click the color box (arrow cursor would turn into an eyedropper) and move the eyedropper

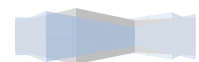


Move the eyedropper to the Stage to copy the RED on Mario's fist. The "touching color ..." block would change to reflect the color just copied.



For Coin Sprite:

Create a for-this-sprite-only variable called `coins_left` and set it to 1 when game starts. Hide Coin when game starts and check if `coins_left` is greater than 0 when it receives a `coin_show` message. If YES, then show Coin, spin it (simulated by changing costumes), hide it, send out a `coin_collected` message, decrease `coins_left` by 1, and finally stop script. If NO (`coins_left` is less or equal to 0), then just do nothing.






For Mario Sprite:

Add the script shown at right to Mario's script so that the score would go up one when he collects a coin (when received a coin\_collected message).



**1.2 Rule: If Mario is attached or touched by a Bat or a Crab, he would die.**

The Sprite Interaction Table below lists all relevant interactions for this rule.

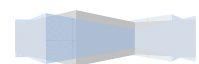
Sprite1	Sprite2	Interaction Rule
Mario 	Crab 	If Mario touches Crab, he would die.
Mario 	Bat 	If Mario touches Bat, he would die.

First add a new costume call *Dead*.

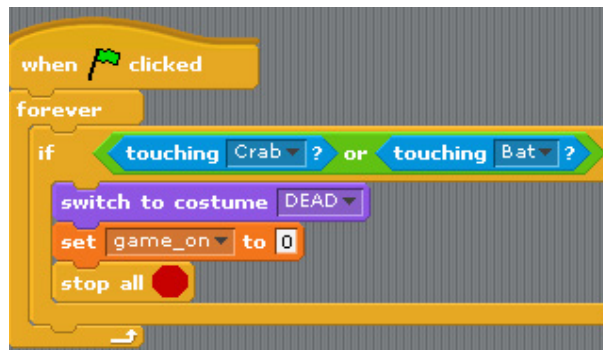


For Mario Sprite:

Create a variable called game\_on and set it to 1 when game starts



Add the script shown at right to Mario so that he would die when touched by a Crab or a Bat. To die, Mario switches costume to DEAD, sets game\_on to 0, and then stop all scripts.



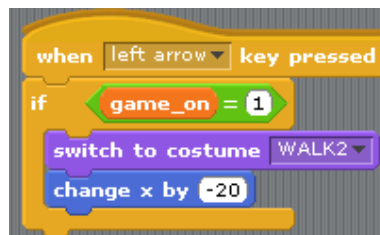
```
when clicked
  forever
    if touching Crab ? or touching Bat ?
      switch to costume DEAD
      set game_on to 0
      stop all
```

Modify the script to move right so that Mario responds to right arrow key click only when game\_on variable is 1.



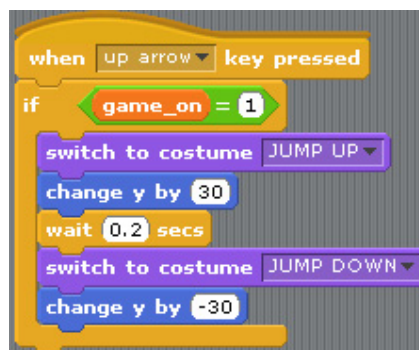
```
when right arrow key pressed
  if game_on = 1
    switch to costume WALK1
    change x by 20
```

Modify the script to move left so that Mario responds to left arrow key click only when game\_on variable is 1.

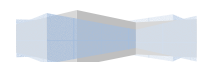


```
when left arrow key pressed
  if game_on = 1
    switch to costume WALK2
    change x by -20
```

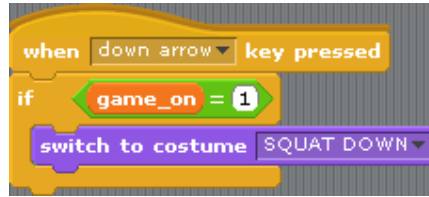
Modify the script to jump so that Mario responds to up arrow key click only when game\_on variable is 1.



```
when up arrow key pressed
  if game_on = 1
    switch to costume JUMP UP
    change y by 30
    wait 0.2 secs
    switch to costume JUMP DOWN
    change y by -30
```



Modify the script to squat so that Mario responds to down arrow key click only when game\_on variable is 1.



## 2. Create Motion Scripts for Mario's Enemies


In this section, we will add script to Mario's enemies (Crab and Bat for now). We want these enemies to move randomly. To do so, we will be using the "pick random X to Y" from Math Tool Kit.




### 2.1 Rule: Crab Randomly Crawls to Left and Right

Since a real Crab can only move side to side, but not up and down, we will make Crab move toward Mario, but with some randomness in its direction when deciding to turn left or right.

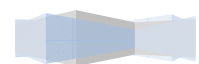
To do so, we need a way to pick left or right randomly. Create a

variable named *dir* and set it to the value of .

The  block returns a random value between 0 and 1.

And the  block returns the round number of this random value. If the random number is smaller than 0.5, then the result would be 0, but if the random number returned is greater or equal to 0.5, then the result would be 1. This way, there is 50% chance the result would be 1, and 50% chance the result would be 0.

To use the result to determine the direction, use the "if ... else ..." block.





```
if <round pick random 0 to 1 = 0>
  point in direction 90
else
  point in direction -90
```

Remember that in Scratch, direction 90 is right -90 is left.

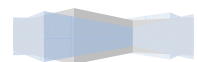
```
point in direction 90
```

(90) right  
(-90) left  
(0) up  
(180) down

Let's put Crab's script together so that it moves randomly to left and to right, and that it bounces if it hit the edge.

```
when clicked
  go to x: 17 y: -110

when clicked
  forever
    wait 0.1 secs
    if <round pick random 0 to 1 = 0>
      point in direction 90
    else
      point in direction -90
    move 25 steps
    if on edge, bounce
```



## 2.2 Rule: Bat Flies Toward Mario, with Some Randomness

Unlike Crab, Bat flies so it should move in all four directions. But there's much space on Stage so Bat may never touch Mario if it just flies randomly. What we want is to have Bat fly toward Mario but randomly adjusts its direction. To have it move toward Mario, use "point towards X" block and "move X steps" block.



To keep flying toward Mario when game starts, expand the script to look like this:



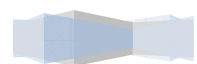
If you try it now, Bat would shoot straight toward Mario and the game will end before you have time to move Mario! What we need to add is a bit "wondering" by adding directional randomness. Just like Crab, Bat should randomly pick a direction to turn, but unlike Crab, Bat should pick a direction from -90 to 180, the whole range of rotation in Scratch.



To do so, place this block immediately below "Move towards Mario":

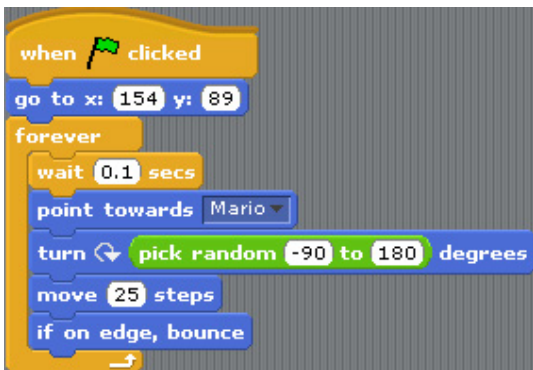


The updated script would look like this:



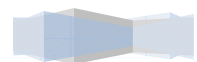


Don't forget to add the "if edge, then bounce" block. So the final script for Fly motion should be:



### !!TEST TIME!!

Now we are ready to test! Click Green Flag to start. Crab should crawl side to side, and Bat should fly in zig-zag motion. Try moving Mario under Brick and click up arrow key to jump to hit it. Coin should appear above Brick and disappear shortly after. The score would go up by 1. Also try move Mario toward Bat or Crab to "commit suicide"; when Mario touches either Bat or Crab, all should stop.




## LESSON 15: Scrolling Platform Game – Platforms

If you've have played one or more [platform games](#) (video games characterized by jumping to and from suspended platforms), you should've seen scrolling platforms. In this lesson, I will show you how to add platforms and make them scroll. Let's do it!


Platform Sprites are usually immobile sprites that look like stage backgrounds but can interact with other sprites on stage. Moreover, they often are located at the bottom of the Stage and they could look like ground, wall, grass field, or brick pavement. Today, I will show you how to turn our simple Mini Mario game from Lesson 12 to a basic platform game using Platform Sprites.

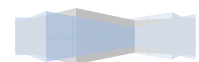
---

### Step 1: Create Platform Sprites

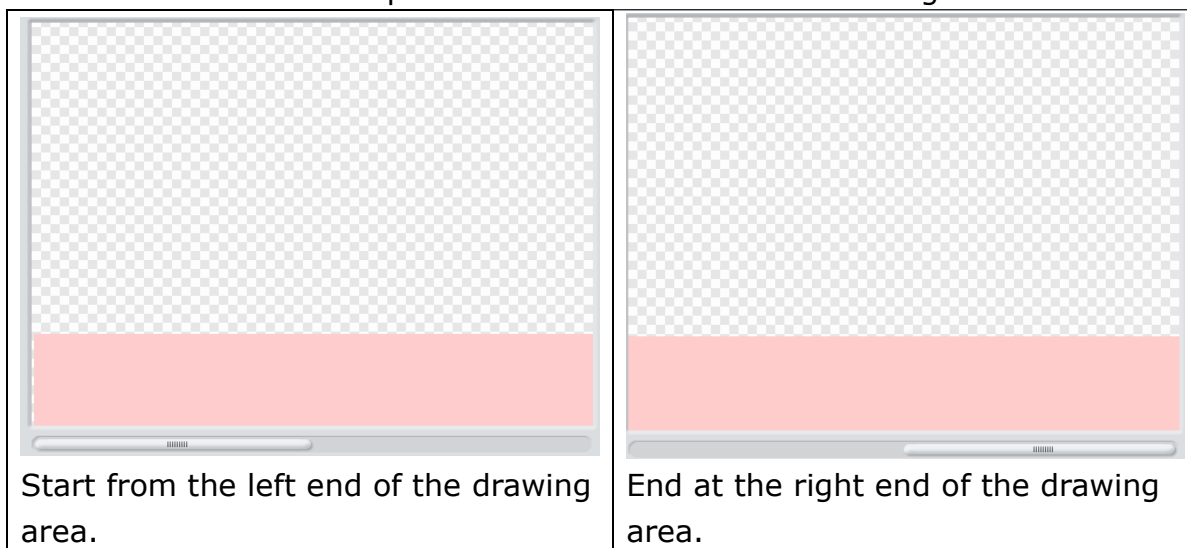
To create platform sprites that look like brick pavement, first click the "Paint new sprite" button  to open the costume editor. Then use

square tool  in solid mode   with pink

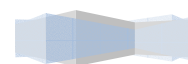
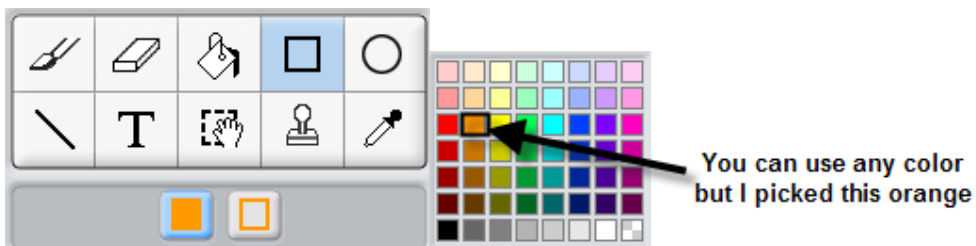
 **pick this color**  
color as shown below.

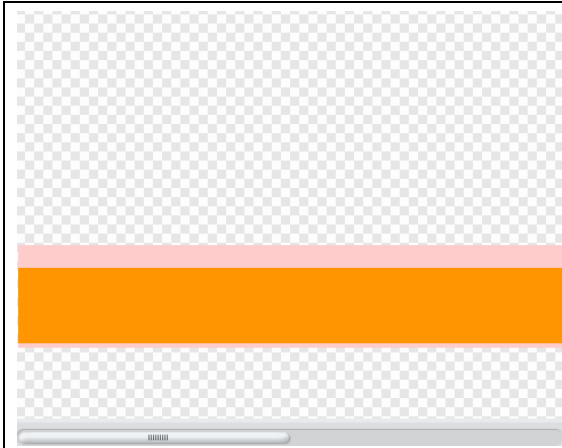


Create a wide bar that spans the whole width of the drawing area.

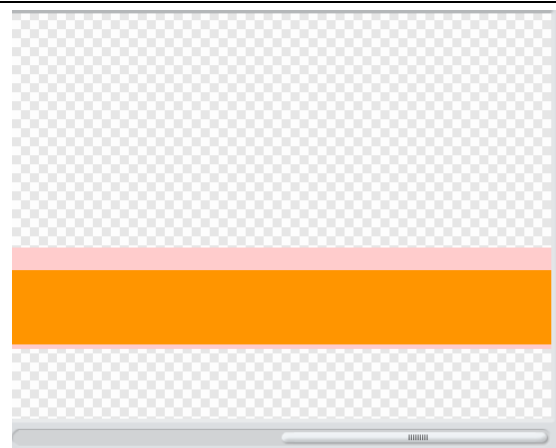


Draw a wide bar which is enclosed in the pink wide bar. You can pick any color.



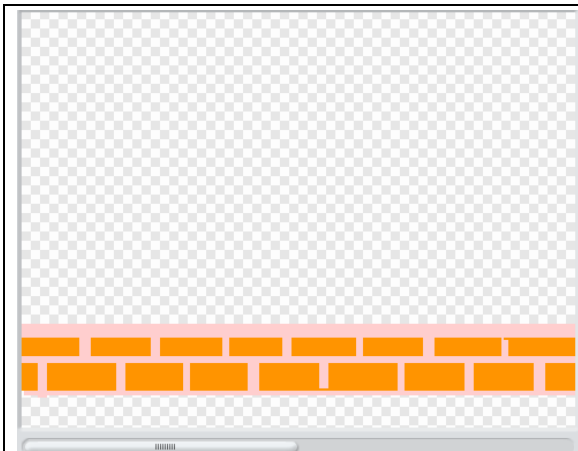
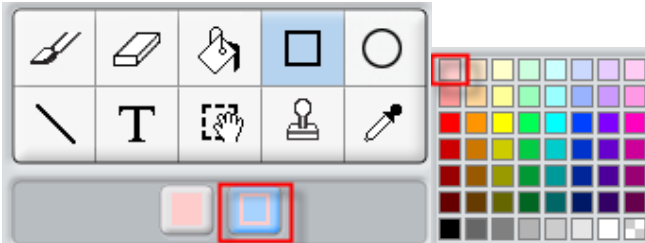


Start from the left end of the drawing area.

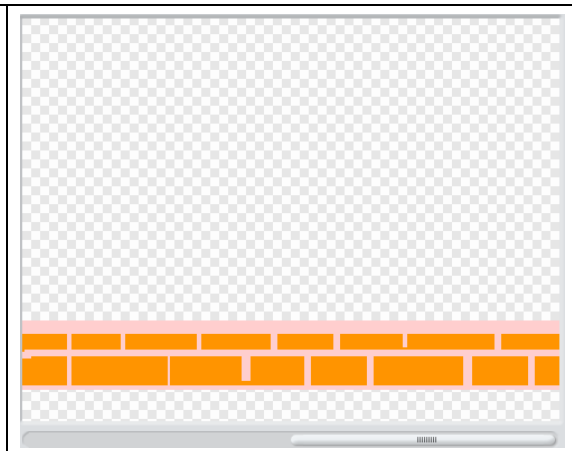


End at the right end of the drawing area.

Then use pink hollow box to create grouts.



Start from the left end of the drawing area.



End at the right end of the drawing area.

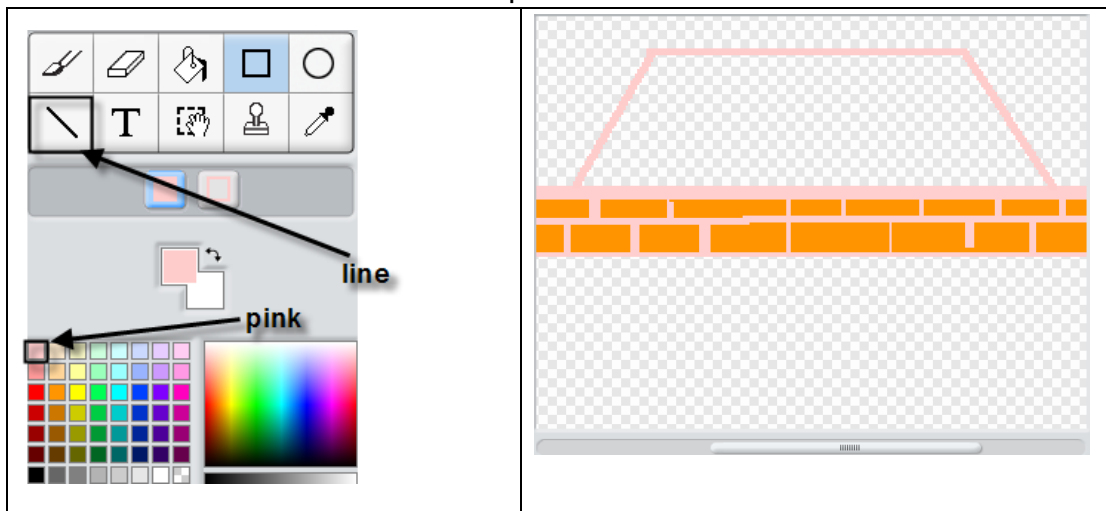
Click OK to save and rename this costume as Level 1. Then click Level1's "Copy" button.



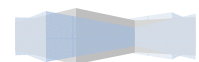
Rename the copy to Level 2 and click its "Edit" button.

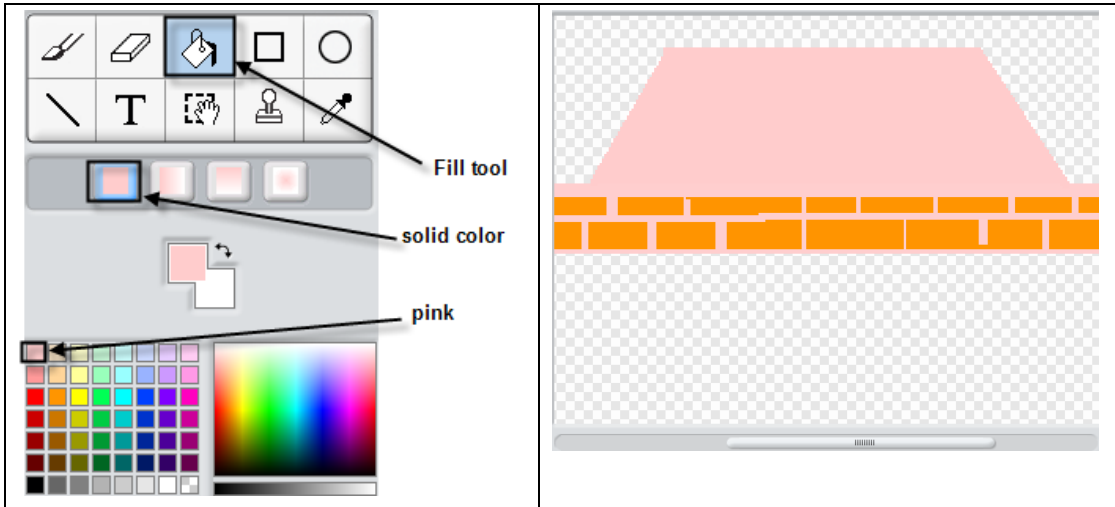


Draw the outline of an elevated platform.

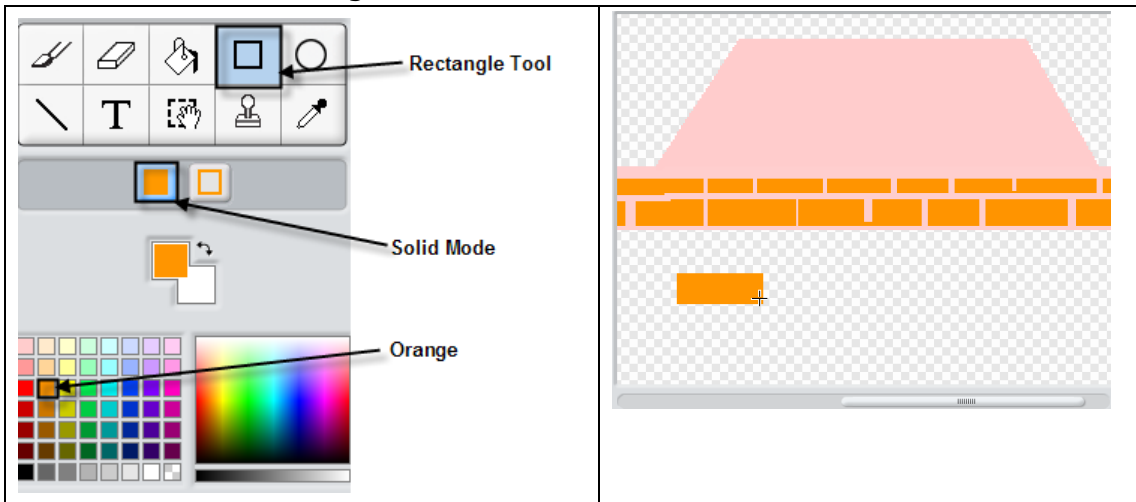


Then fill this platform with the same pink color.

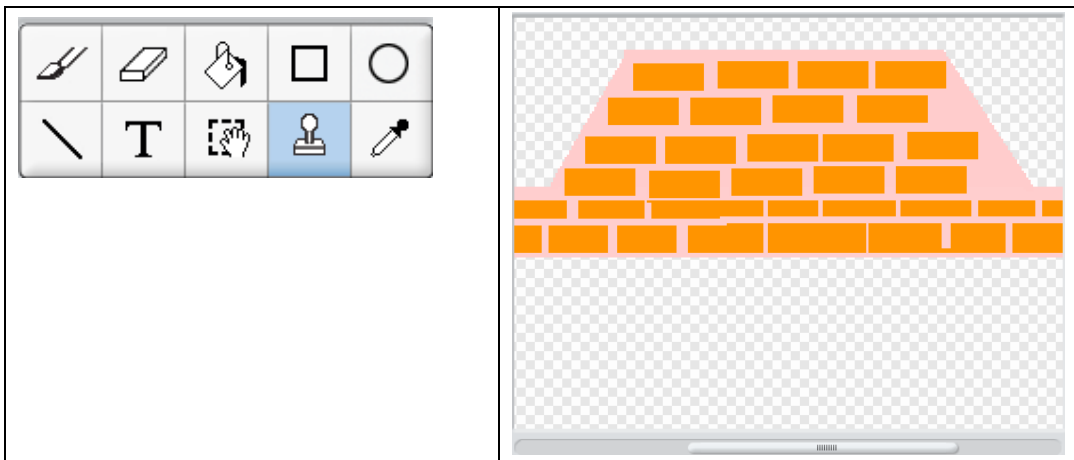




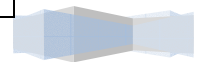
Create a brick in orange.



Then use Stamp Tool to make copy of this brick and stack them as below. Then click OK to save.



Change the sprite name to "Platform".





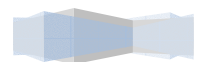


Then move Platform to the bottom of the Stage.



With Platform sprite selected, click Scripts Tab, and then select Motion Tool Kit. Drag out "go to x: 2 y:-164" block to the Scripts panel. Note that the x and y values of your "go to x: ? y:?" might be a bit different than 2 and -164. It's OK – these numbers are filled in for you automatically by Scratch when you move a sprite on the Stage.

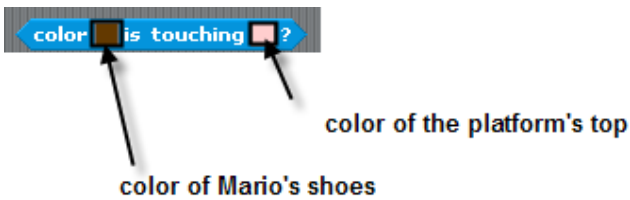
Build a combo block as followed so that Platform sprite shows at the bottom of the Stage every time the game starts.



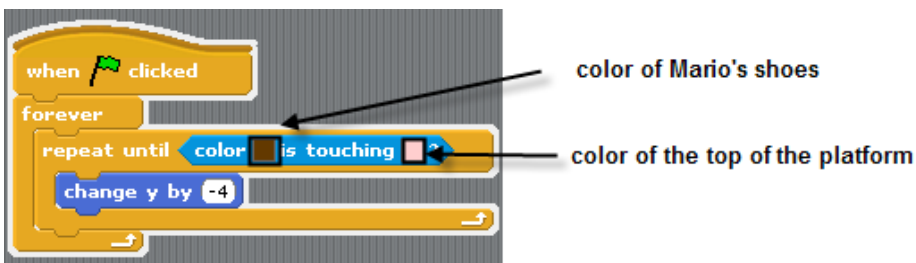


## Step 2: Interact with Platform Sprites

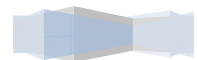
Next, we need to modify Mario's script so that he would stand on the platform. We will be using "color X is touching Y" block. We will use "color brown is touching pink" because Mario's shoes are brown and the platform's top is pink.

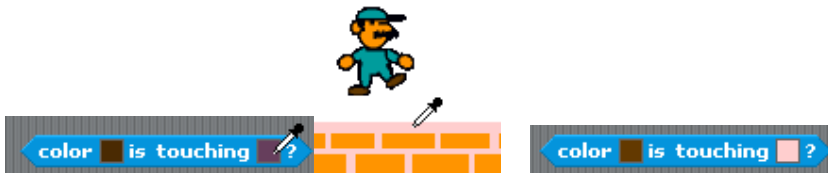


Create a combo block that repeatedly moves Mario down (change y by -4) until Mario's shoe color is touching the color of the platform's top.



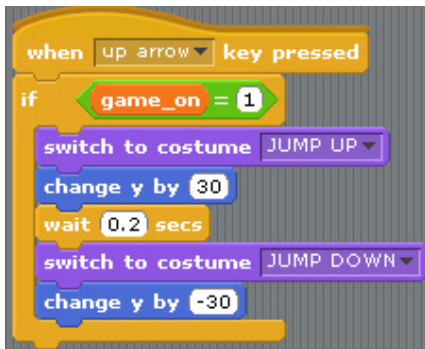
To copy these two colors, first click the first color box and then click Mario's shoes with the eyedropper. Do the same to copy the color pink.





With this combo block, we need to change the combo block that does the jumping.

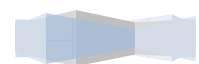
This is what we had from Lesson 12 – jumping involves going up (increase y) as well as going down (decrease y).



But since now we've added the combo block that moves Mario down until his shoes hit the platform, jumping now only needs to involve going up. Moreover, make Mario jump higher by increase the change from 30 to 60, so that Mario can jump over Crab.



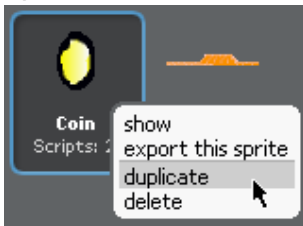
Also, I modified where Mario shows in the beginning so he will fall down from the sky, just like the original Mario game.



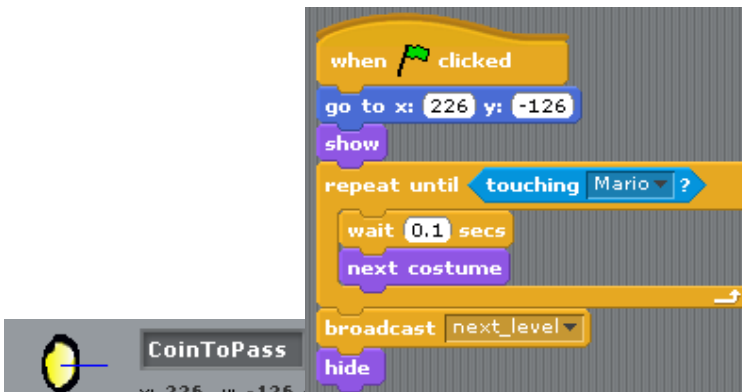
**TRY IT!** Start the game now and you should see Mario fall down from above and landed right on top of the Platform sprite. Try jump around and see Mario fall back down on the Platform.

### Step 3: Add a Level to the Game

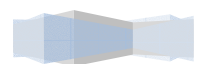
Next, let's kick it up a notch by adding a level to the game to make it a two-level game. To do so, let's add a special coin such that if Mario collects it then he would be moved to the next level. Right click Coin sprite and select duplicate.



Rename the duplicate sprite to CoinToPass. Delete all script in CoinToPass and add the following script block so that when CoinToPass sprite is touched by Mario, it would send out a broadcast message called "next\_level".

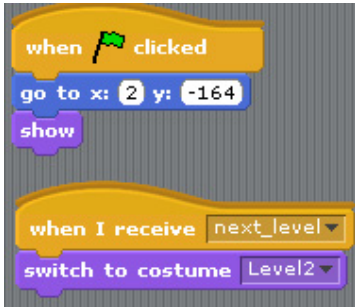


The Platform sprite would catch the "next\_level" message and change its costume to "Level2".






The Platform sprite's scripts now look like this:

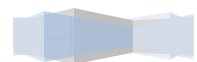
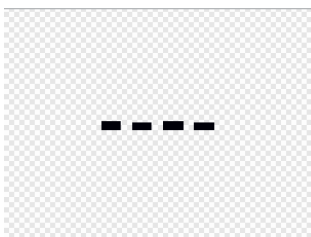


**TRY IT!** Now you can test your jumping skill. Note that you can do multiple jumps (by hitting UP multiple times) so it looks like Mario is flying or sky-walking. I just thought that it's nice to let him have a bit more power.

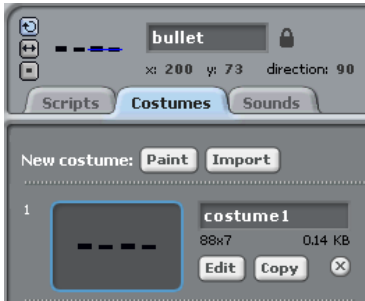
#### Step 4: Bullets

If you are like me, then you probably find this game pretty tough. If not, then you can skip this step. I guess you are pretty good at sky-walking.... But for me, my Mario needs a bit help to fend off his enemies. Let's give him bullets, shall we?

Click  to create a new sprite and create a simple bullet trace that looks like this:



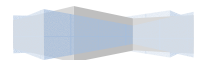
Save this sprite as "Bullet".

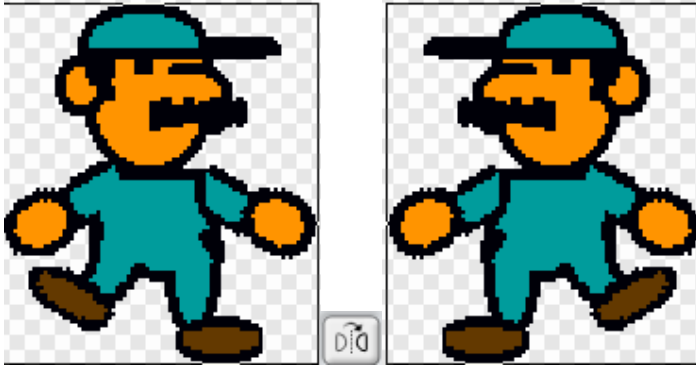


To tell bullet which direction to fly to, create a variable called "facing\_right" and set it to true (set to 1) when Mario turns right and to false (set to 0) when he turns left. The updated relevant scripts look like this:



Besides script, let's change the costume "WALK2" to face left. Click costume WALK2's edit button. Then click the horizontal flip button to flip the image.



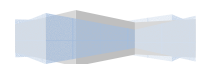


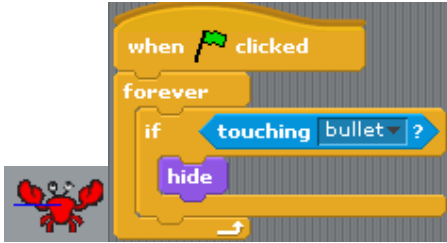
Now let's update Bullet's script so that, when space key is pressed, it flies out from Mario to the right if Mario faces right and to the left if he faces left. Also, Bullet will keep flying until it touches the edge of the Stage, then it would hide.

```
when clicked
hide

when space key pressed
if game_on = 1
go to Mario
show
if facing_right = 1
point in direction 90
else
point in direction -90
repeat until touching edge ?
move 10 steps
hide
```

Finally, let's add a combo block to Bat and Crab's scripts so that they disappear when hit by Bullet.

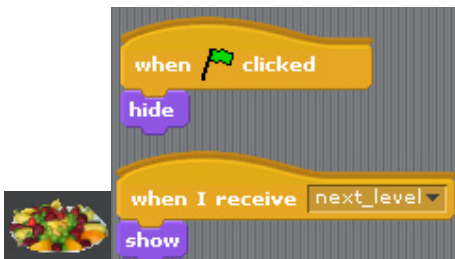




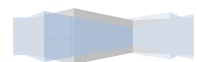
**TRY IT!** Try attacking by hitting the space key. Liberating eh?

### Step 5: Rewards

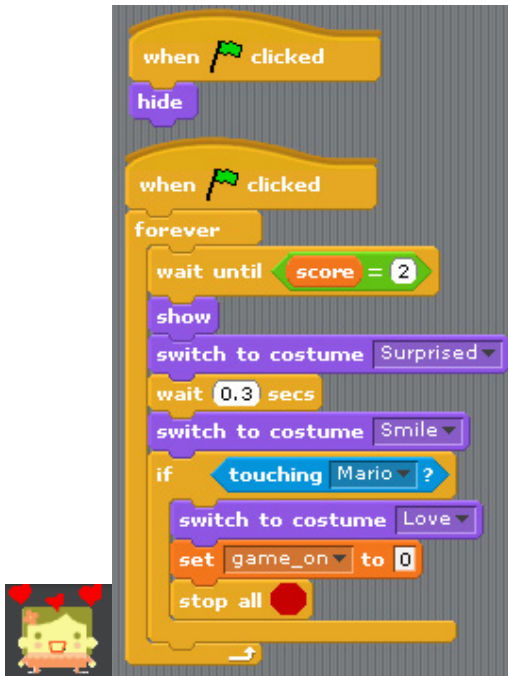
Finally, let's reward Mr. Mario for his hard work. The first prize is Fruit Platter. Update its scripts so that it shows on the top of the Stage when the first Level is cleared.



Update Princess' scripts so that she shows up only after Mario has scored two points. When Mario finally met or "touched" Princess, she would have hearts all above her head and then the game would end.

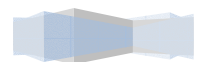






**TRY IT!** Try scoring two points by getting all coins, then move close to Princess to see her fall in love with Mario.

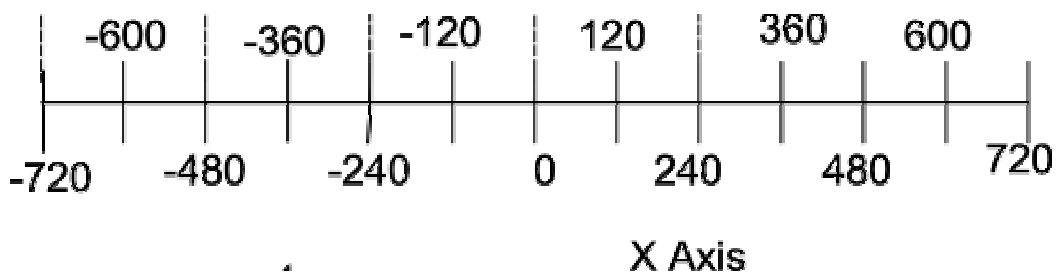
THIS CONCLUDES LESSON 13. IT'S TIME FOR MY [BOBA TEA BREAK](#).



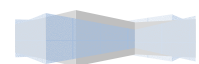
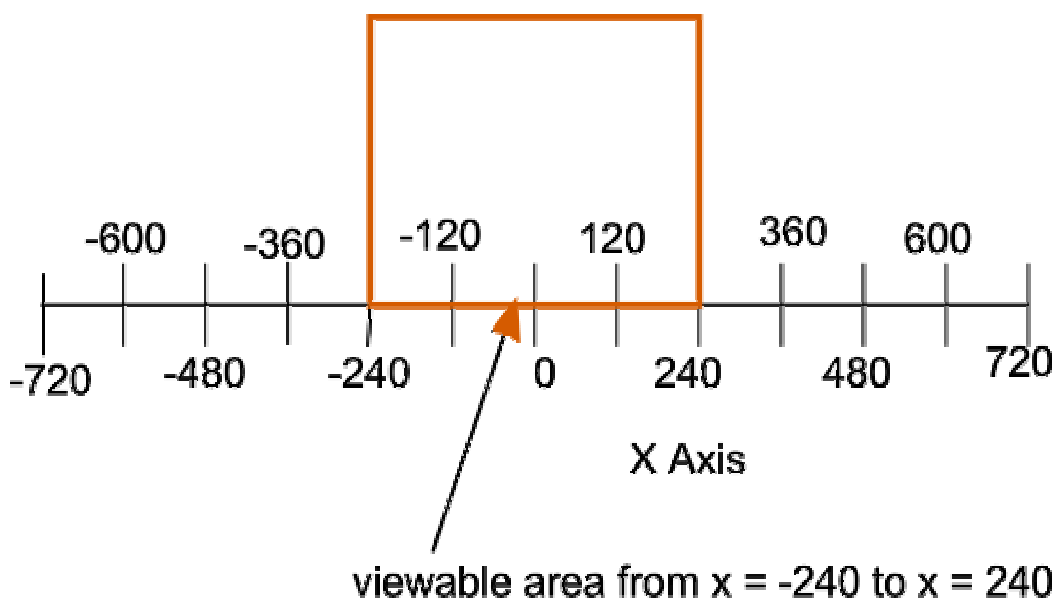
## Lesson 16: Mini Mario Game Part 5 – Scrolling Intro

The trick to have the scrolling platform effect in a Scratch game is to add multiple Platform Sprites. A Platform Sprite is an immobile sprite that looks like a stage background but it can interact with other sprites on stage.

Think of the Stage as a viewable area of an infinitely long strip. This viewable area starts from  $x = -240$  and ends at  $x = 240$ . We will just consider the one-dimensional reference frame and ignore the y-axis for now. The reference frame does not change when sprites move.



The viewable area also does not change when sprites move.



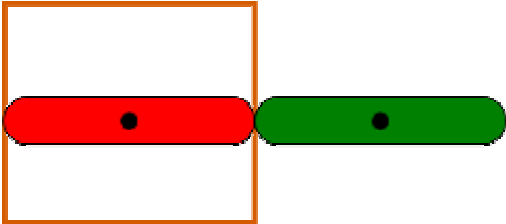
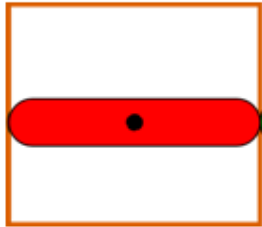
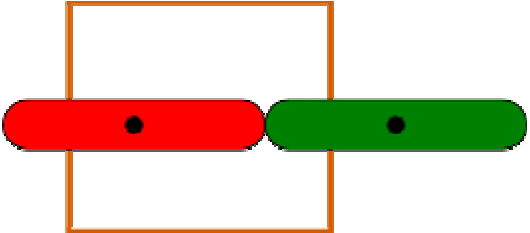
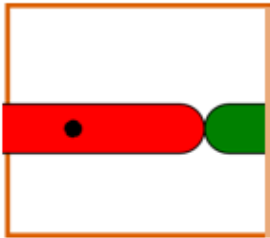
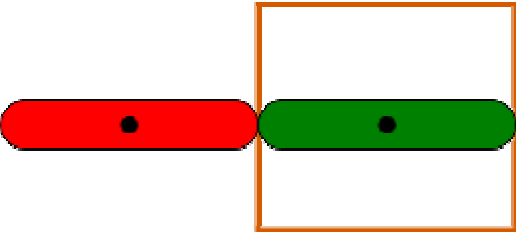
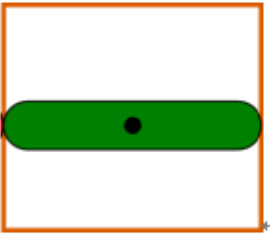
To illustrate how scrolling works, let's use these two platform sprites that each spans 480.

Platform 0: 

Platform 1: 

Line them up side by side, starting from the left edge of the Stage (the viewable area):



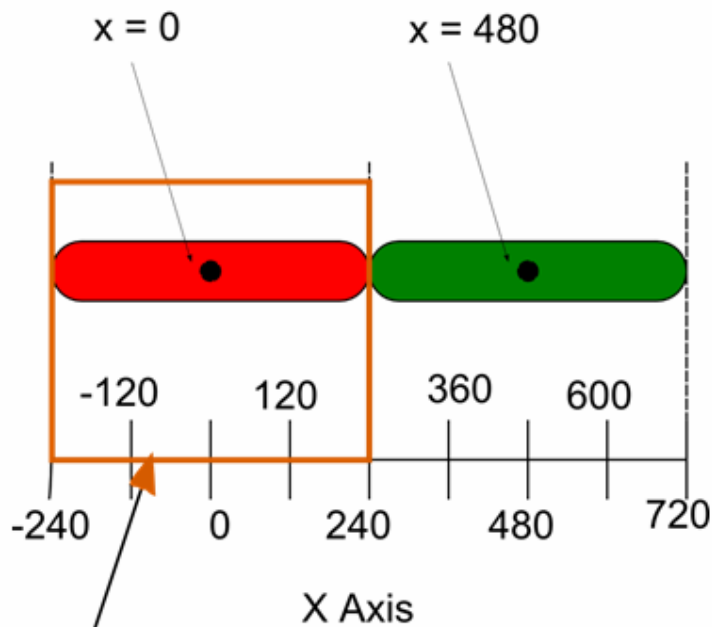
The Whole Strip	What's Shown on Stage
<p>Before Scrolling</p> 	
<p>Scroll to the right by 120</p> 	
<p>Scroll to the right by 480</p> 	

When the game first starts, the red platform's center is located at  $x = 0$  and takes up space from  $x = -240$  to  $x = 240$ ; the green platform's center is located at  $x = 480$  and takes up space from  $x = 240$  to  $x = 720$ . Both have width of 480.

$$240 - (-240) = 480 \quad \leftarrow \text{width of the red platform sprite}$$

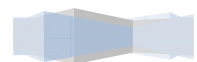
$$720 - 240 = 480 \quad \leftarrow \text{width of the green platform sprite}$$

The orange box represents the Stage and only area inside the orange box is viewable. The Stage is 480 in width (x-axis) and 360 in height (y-axis). Only the x-axis is shown because we will only be scrolling left or right.



viewable area from  $x = -240$  to  $x = 240$ .

If we let `STAGE_WIDTH` be a global variable for the width of the Stage, and `PLATFORM_INDEX` be a local variable of each platform sprite,



which specifies where the platform is on the strip, then we can represent the starting positions for both sprites as  $PLATFORM\_INDEX * STAGE\_WIDTH$  ( \* is the multiplication sign).

$$STARTING\_POSITION_{RED} = PLATFORM\_INDEX * STAGE\_WIDTH = 0 * 480 = 0$$

$$STARTING\_POSITION_{GREEN} = PLATFORM\_INDEX * STAGE\_WIDTH = 1 * 480 = 480$$

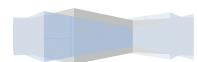
Imaging you are driving a car and are moving forward (scrolling to the right). Though the car is moving forward, but looking from inside the car, the landscape is moving backward relative to you.

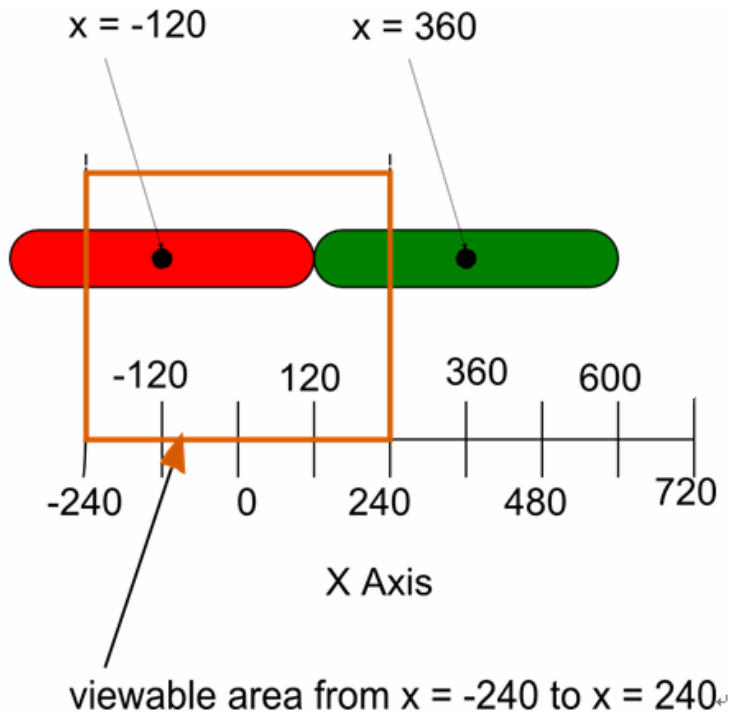


In the same way, to move forward from the user's perspective, we move the platforms backward. Therefore, if the user scrolls to the right by 120, then the positions for both platform sprites should be:

$$POSITION_{RED} = STARTING\_POSITION_{RED} + SCROLL\_X_{PLATFORM} = 0 + (-120) = -120$$

$$POSITION_{GREEN} = STARTING\_POSITION_{GREEN} + SCROLL\_X_{PLATFORM} = 480 + (-120) = 360$$

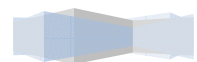


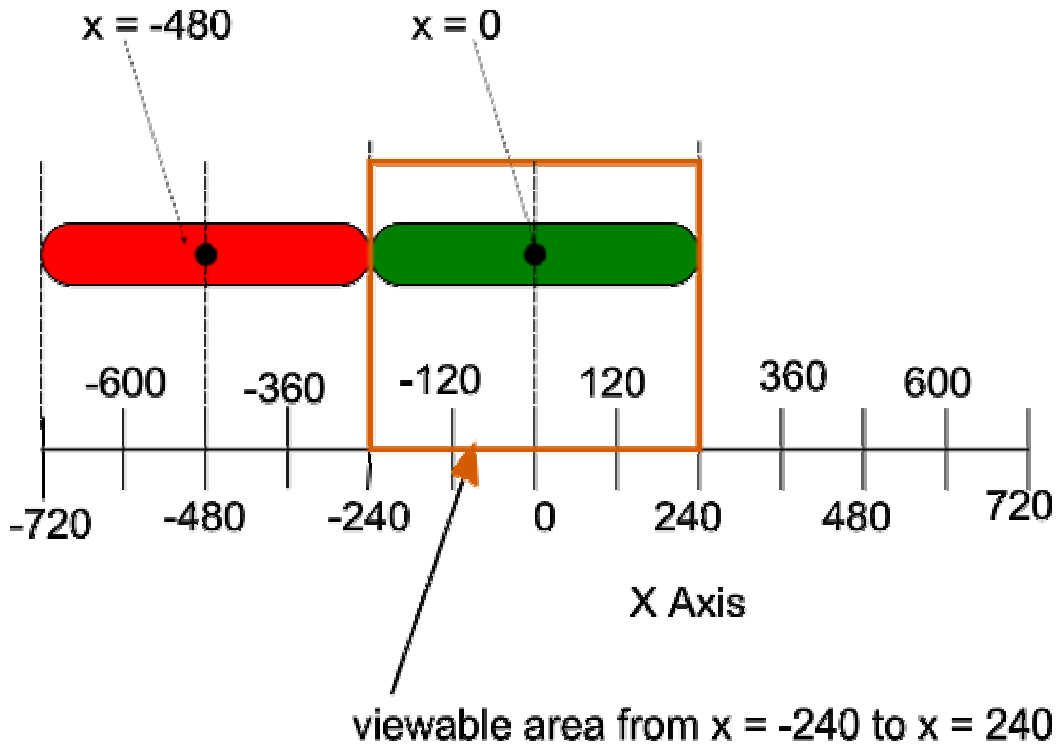


Moreover, if the user scrolls to the right by 480, then the positions for both platform sprites should be:

$$\text{POSITION}_{\text{RED}} = \text{STARTING\_POSITION}_{\text{RED}} + \text{SCROLL\_X}_{\text{PLATFORM}} = 0 + (-480) = -480$$

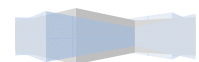
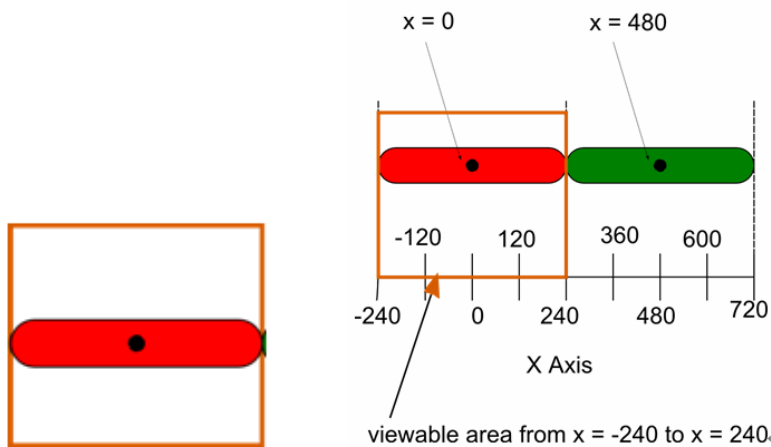
$$\text{POSITION}_{\text{GREEN}} = \text{STARTING\_POSITION}_{\text{GREEN}} + \text{SCROLL\_X}_{\text{PLATFORM}} = 480 + (-480) = 0$$



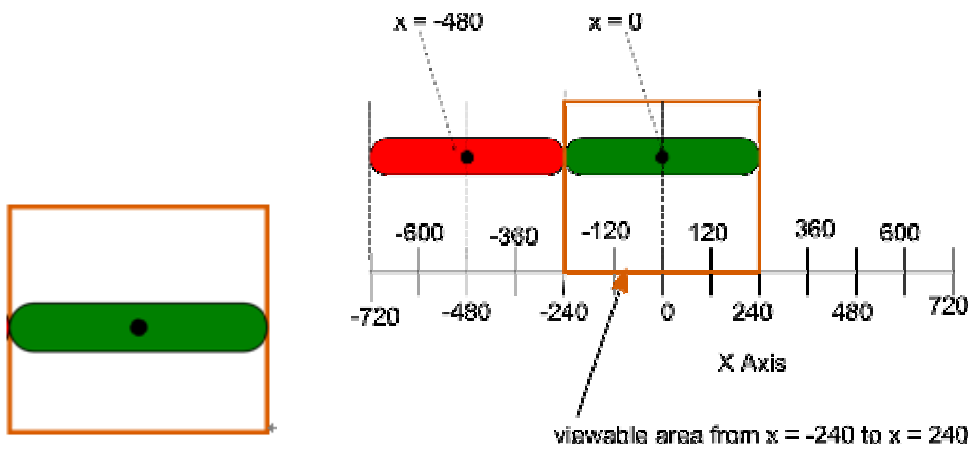


We also need to hide a platform sprite if its center is larger than 480 or smaller than -480.

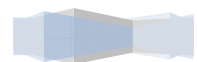
When the game first starts, the green platform is hidden because its center is at 480.



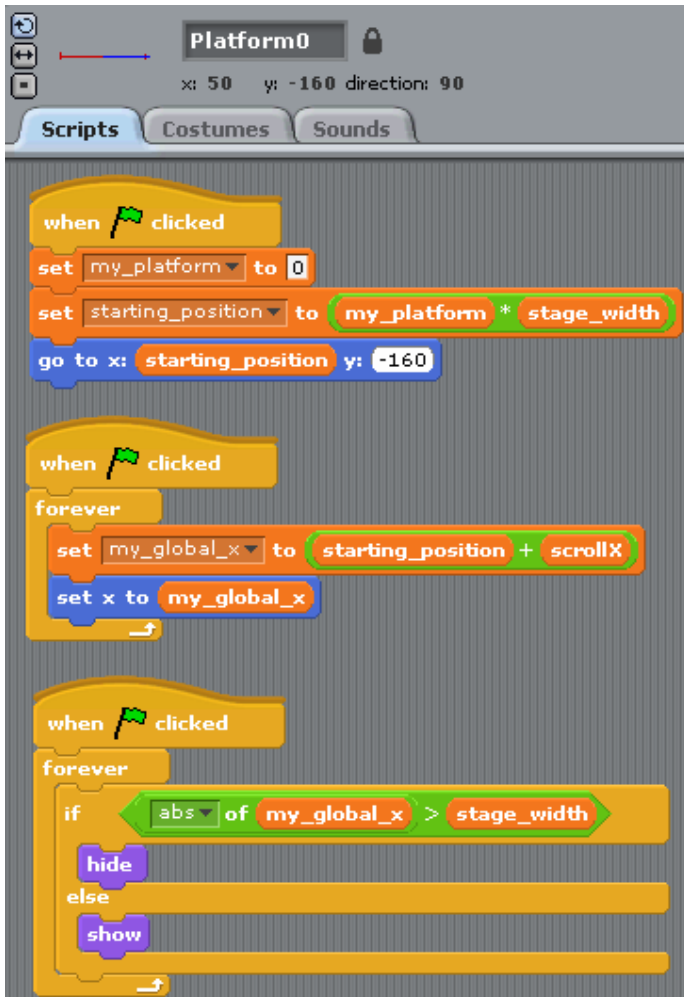
After scrolling to the right by 480, the red platform is hidden because its center is now at -480.



Now we are ready to put the scripts together. Below is the script for the red platform or Platform0.








The green platform or Platform1 has exactly the same script as the red platform or Platform0, except that its my\_platform is 1 instead of 0.



Platform1 

x: 462 y: -160 direction: 90

Scripts Costumes Sounds

```
when clicked
  set my_platform to 1
  set starting_position to my_platform * stage_width
  go to x: starting_position y: -160

when clicked
  forever
    set my_global_x to starting_position + scrollX
    set x to my_global_x

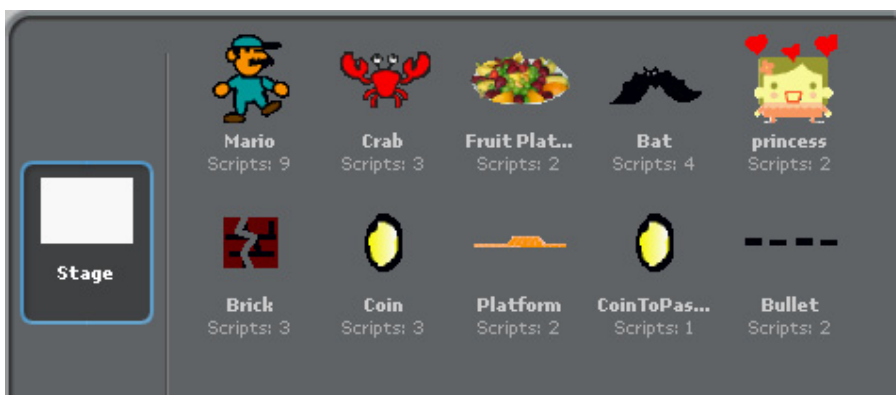
when clicked
  forever
    if abs of my_global_x > stage_width
      hide
    else
      show
```



# Lesson 17: Platform Game Wrapup

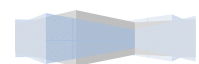
In this lesson, we will wrap up the Mini Mario game by making it scroll, among other things. Also, please go to <http://shallwelearn.com> to download MiniMarioLesson13.sb from Scratch Programming Lesson 13.

This is the snapshot of the sprite list when the project is opened.

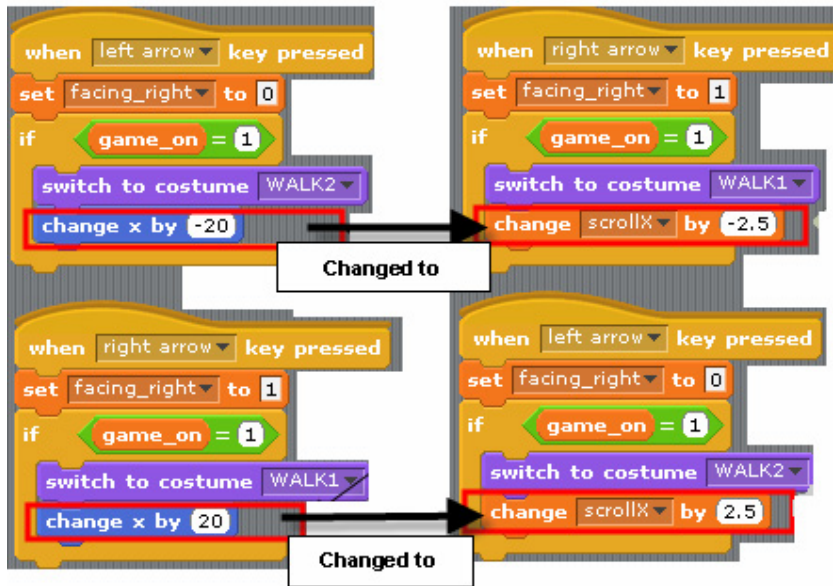


## *Step 1: Create a Variable to Represent Scrolling Amount*

Create a global variable called scroll to represent the amount of scrolling. When Mario moves to the right, the scrolling amount decreases, and when it moves to the left, the scrolling amount increases. If you are unfamiliar of how scrolling works in Scratch or in general, please refer to Shall We Learn Scratch Programming Lesson 14.



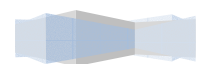
Delete “change x by ...” and replace with “chang scollX by -2.5”.

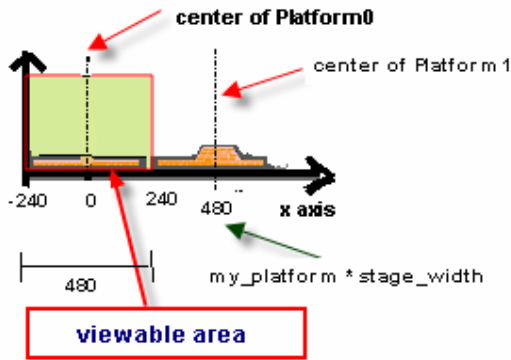


### Step 2: Make Platforms Scroll

I am going to split **Platform** sprite into two sprites: **Platform0** and **Platform1**.

I will line these two platform sprites side-by-side. When the game starts, **Platform0** is located at  $x = 0$  or  $0 * \text{the stage width} (=480)$ . **Platform1** is at  $x = 480$  or  $1 * \text{the stage width}$ .





**stage\_width = 480**  
**my\_platform for Platform0 = 0**  
**my\_platform for Platform1 = 1**

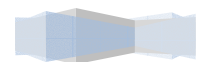
Rename **Platform** to **Platform0**. Delete all scripts from **Platform0** and add the following scripts, which are very similar to what we built in Lesson 14, except that I added a global variable call **platform0\_global\_x**.

```

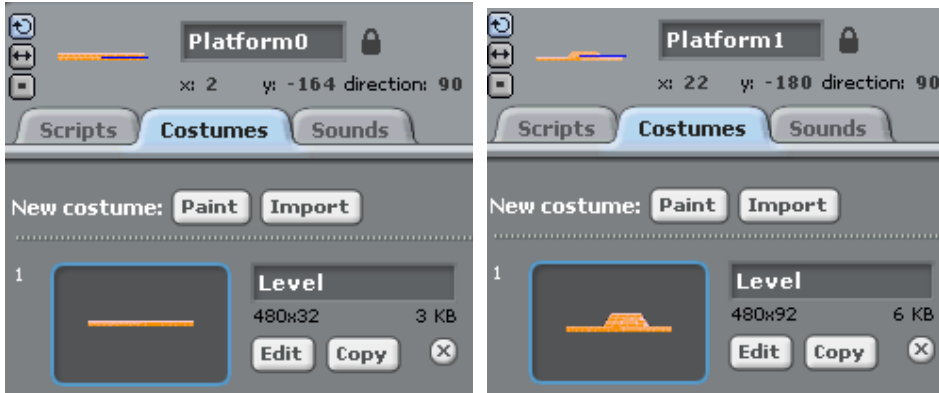
when clicked
  set my_platform to 0
  set platform0_global_x to 0
  go to x: my_platform * stage_width y: -164
  show

when clicked
  forever
    set platform0_global_x to scrollX + my_platform * stage_width
    set x to platform0_global_x

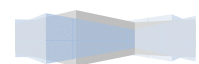
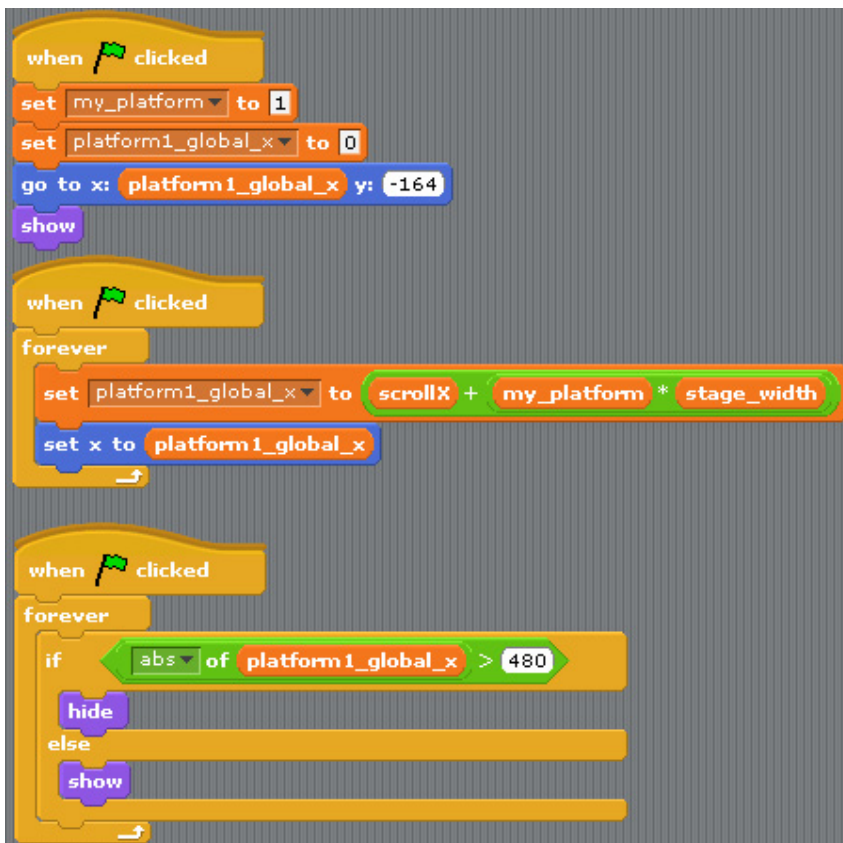
when clicked
  forever
    if abs of platform0_global_x > 480
      hide
    else
      show
  
```



Then make a copy of **Platform0** and name the copy as **Platform1**. Delete costume **Level2** from **Platform0** and rename costume **Level1** to **Level** and delete costume **Level1** from **Platform1** and rename costume **Level2** to **Level**.



Then, as shown below, for **Platform1**, set `my_platform` to 1 when game starts. Then create a new global variable called `platform1_global_x` and place it in the place of `platform0_global_x`.

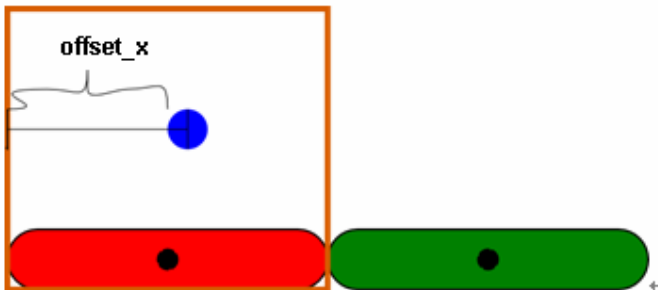




**TEST TIME:** Now test the game. You can scroll but there is an obvious problem. Both the brick and two coins keep following Mario so Mario could never get to them. We will fix it.

*Step 3: Modify the **Brick** and **CoinToPass** to Stop Them from Following **Mario***

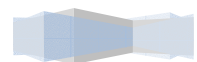
To keep the brick and coins from following **Mario**, we need to make them scroll, just like the platform sprites. We will copy the scripts pertaining to scrolling from platform sprites to both Brick and Coin sprite. But we need to make two changes. The first is to add a local variable call *offset\_x*; this variable stores the distance from the left boundary of the platform to the sprite.

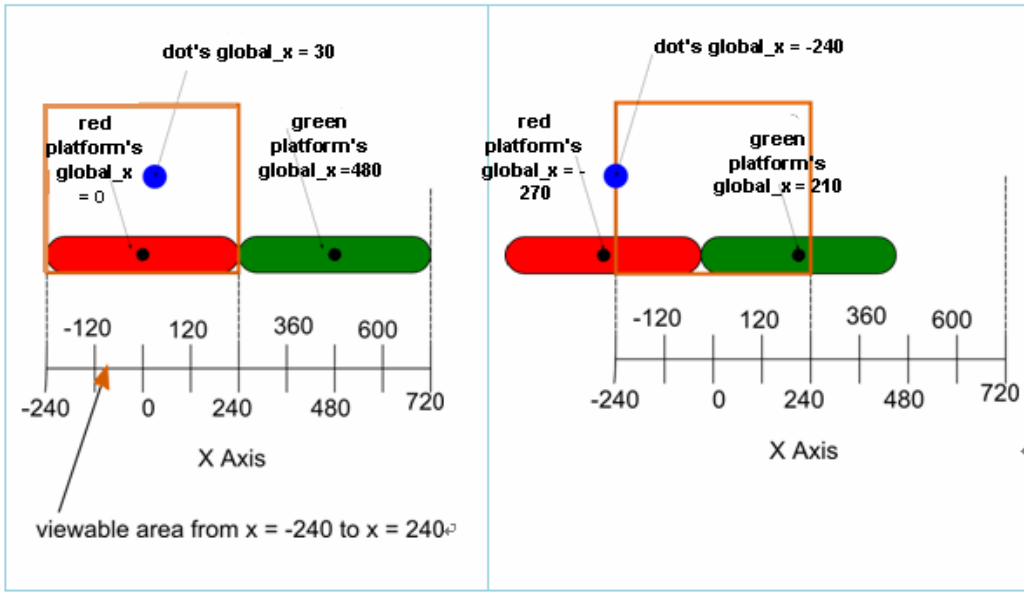


The location of the dot sprite will be

$$\text{STARTING\_POSITION}_{\text{DOT}} = \text{PLATFORM\_INDEX} * \text{STAGE\_WIDTH} + \text{OFFSET\_X}$$

$$\text{POSITION}_{\text{DOT}} = \text{STARTING\_POSITION}_{\text{DOT}} + \text{SCROLL\_X}_{\text{PLATFORM}}$$



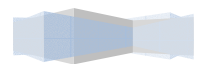


The second change is to change the show/hide boundary from `STAGE_WIDTH` (=480) to `STAGE_WIDTH/2` (=240). The graph below shows the snapshot of the stage before and after the scrolling to the right by 270 (`scrollX = -270`).

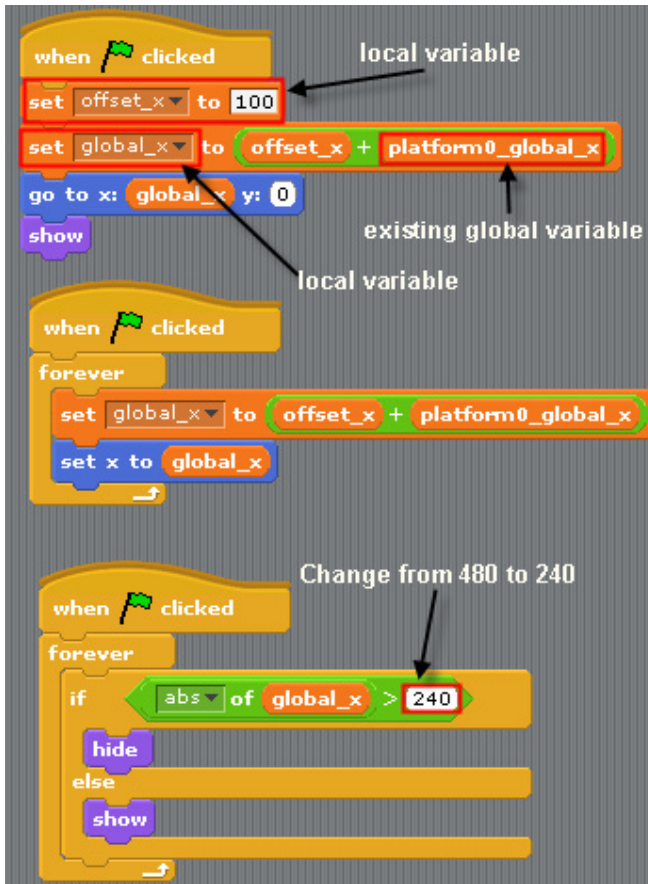
$$\begin{aligned} \text{STARTING\_POSITION}_{\text{DOT}} &= \text{STARTING\_POSITION}_{\text{RED\_PLATFORM}} + \text{OFFSET\_X} \\ &= \text{PLATFORM\_INDEX} * \text{STAGE\_WIDTH} + \text{OFFSET\_X} \\ &= 0 * 480 + 30 = 30 \end{aligned}$$

$$\begin{aligned} \text{POSITION}_{\text{DOT}} &= \text{POSITION}_{\text{RED\_PLATFORM}} + \text{OFFSET\_X} \\ &= \text{STARTING\_POSITION}_{\text{RED}} + \text{SCROLL\_X}_{\text{PLATFORM}} + \text{OFFSET\_X} \\ &= 0 * 480 + (-270) + 30 = 30 \end{aligned}$$

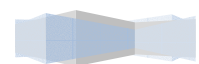
The snapshot below shows **Brick's** script blocks relevant to scrolling.

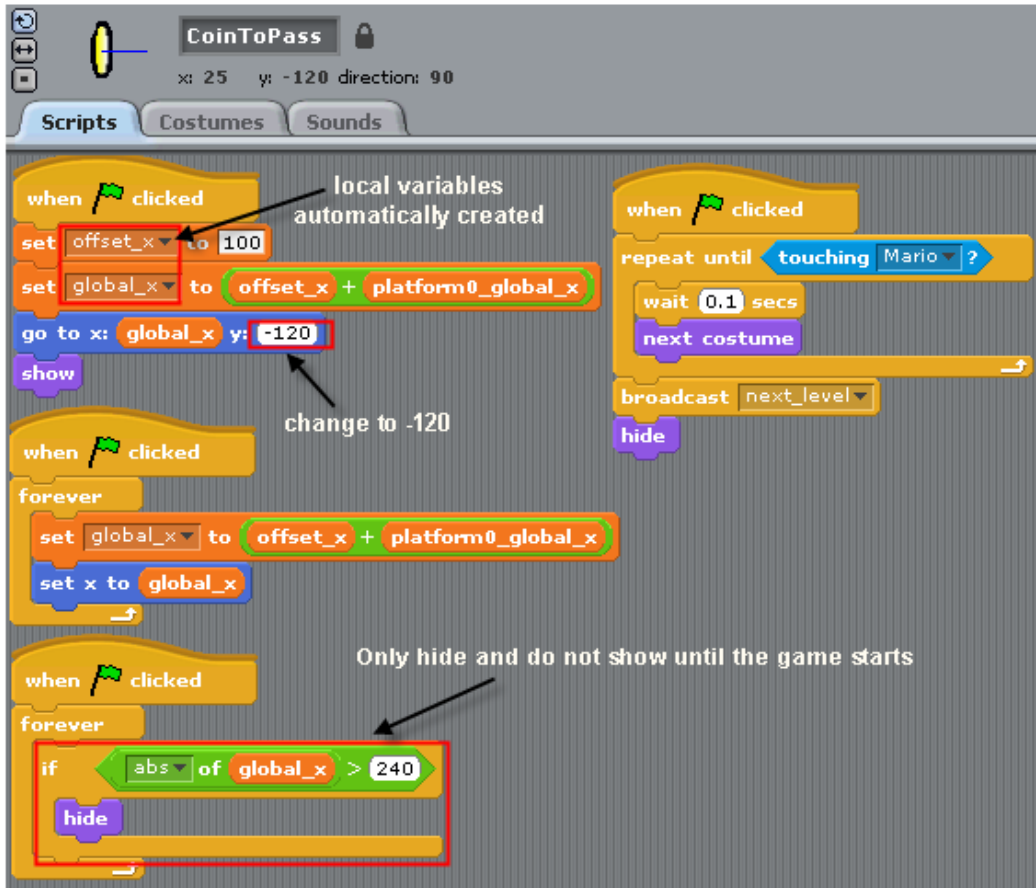




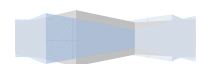


Next, we will change **CoinToPass** sprite. First remove all motion-related scripts and copy all scrolling related scripts from **Brick** to **CoinToPass**.



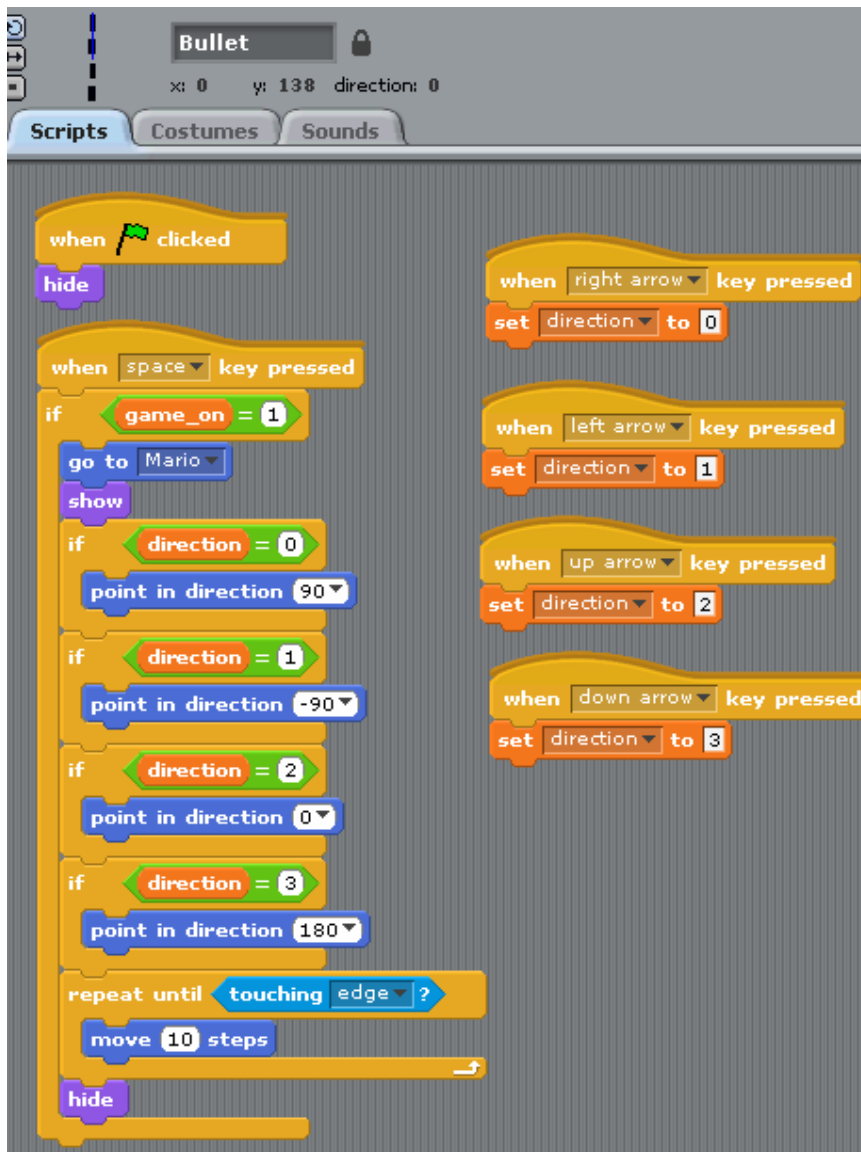


*Step 4: Upgrade the **Bullet** and Turn **Brick** to a Mini Platform*

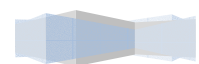


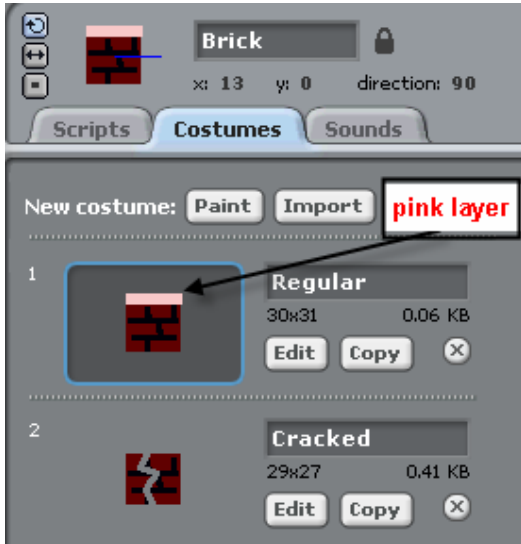
If you would like, you can upgrade the **Bullet** so that you can shoot from

all four directions. Below is the updated script for the **Bullet**.



To turn the **Brick** into a mini platform, just add a top pink layer (the same pink used to create the top layer of the **Platform0** and **Platform1** sprites).

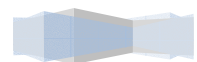




Then **Mario** can stand on the **Brick** as well.



This concludes Lesson 15, the last lesson of Mini Mario Game series. I hope you have enjoyed this series and leave with enough know-how to make your own platform games!



## CONCLUSION

This concludes Shall We Learn Beginning Scratch Programming. I hope you have enjoyed reading this book. If you would like to learn more, please read Shall We Learn Intermediate Scratch Programming.

For project downloads and more scratch or other programming lessons, go to <http://shallwelearn.com>

