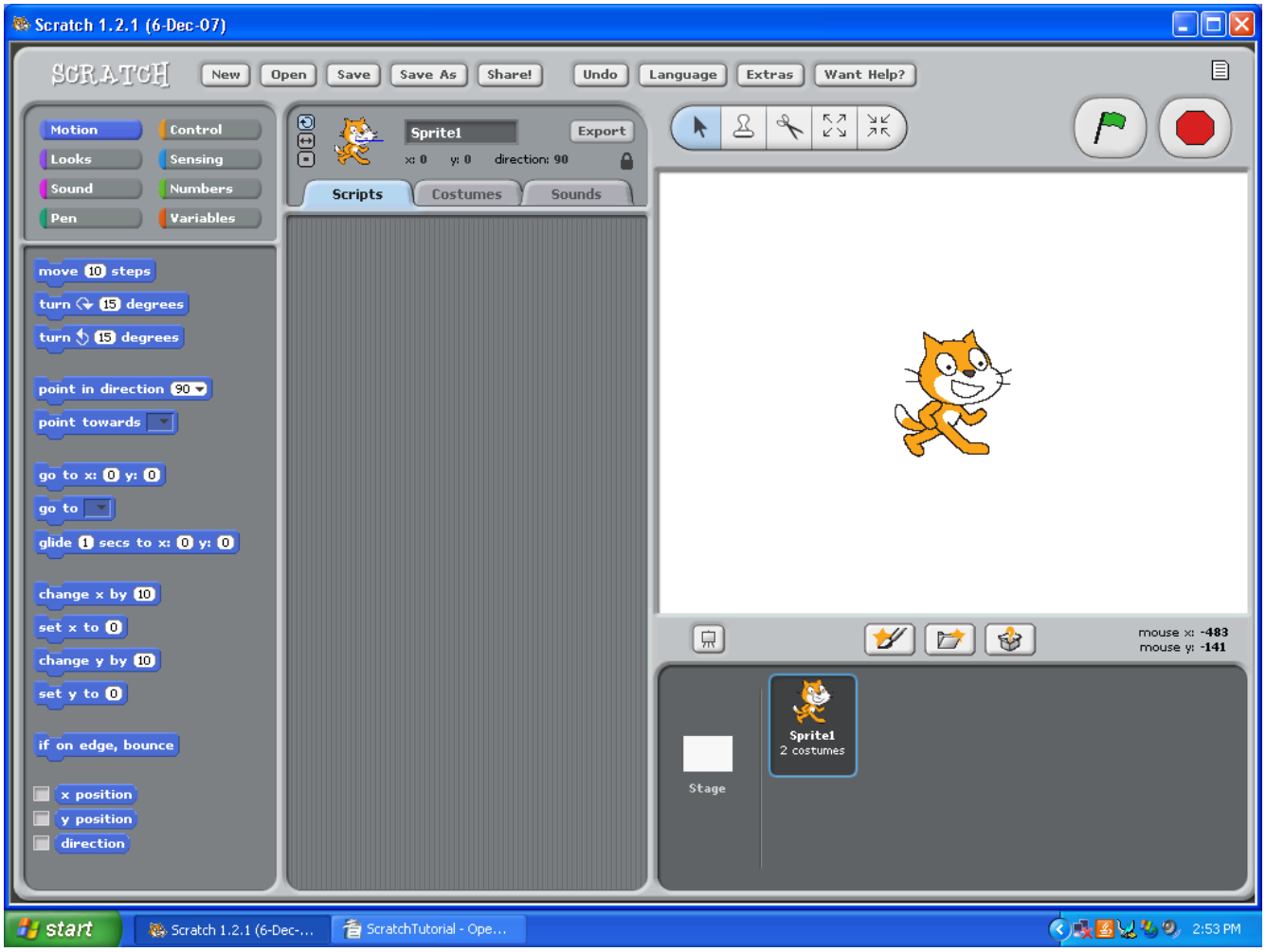
Breakout Tutorial

Yay! Who doesn't like the game Breakout? And now you can make your own!

The goal of the game is to keep the ball from hitting the ground and bounce the ball against the bricks at the top until they are all gone. So let's start!

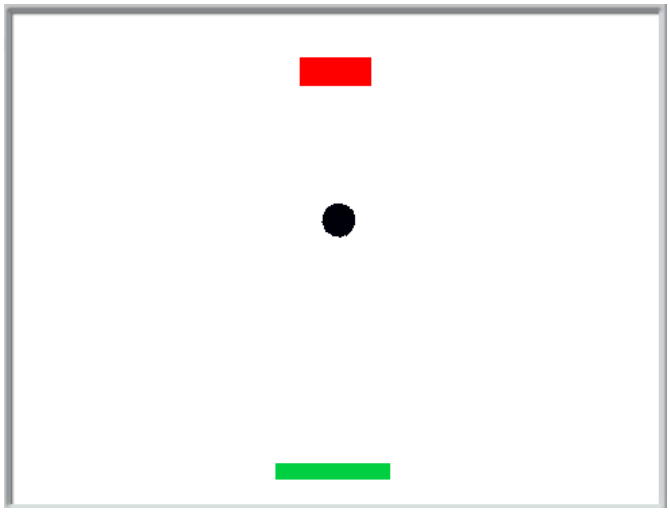
Go ahead and open a new Scratch. Once you've opened Scratch, you should see a screen like this:



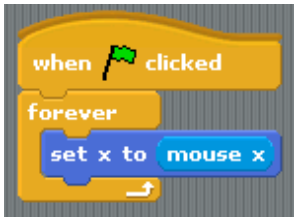
Note: I will almost always be using Windows for my examples. I have used Scratch on Macs before, and there is almost no difference. The only differences that you may find are just locating files, slightly different window management, and other things that are different for all programs on Macs vs Windows. So don't worry. These tutorials should work the same way.

First, we can make the sprites that we will need. We need a paddle to bounce the ball off of, a ball to bounce, and just one brick to start off with. Remember, the sprites need to be small enough to make the game challenging, but big enough to be fun to play.

Here's my screen:



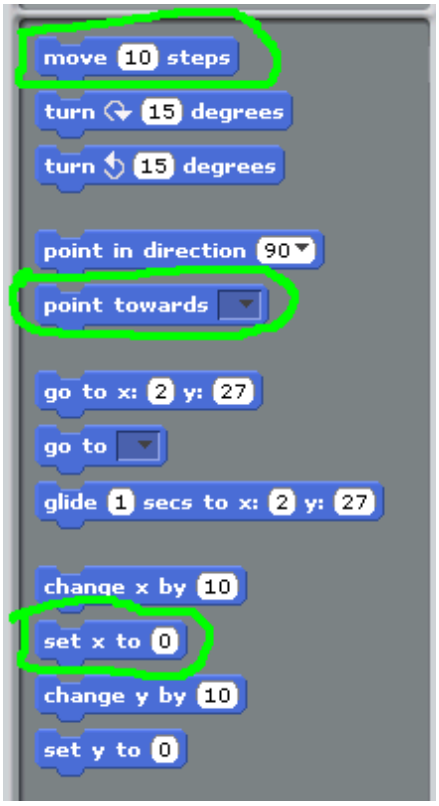
Once you have made these parts, open up code section of the paddle sprite. We need to make the paddle follow the user's mouse so that they can control the game. To do this, just make it so that the paddle always sets its x coordinate to that of the mouse. You can use the "set x to" block from the motion tab, and drag the "mouse x" oval from the sensing area into the brick. Make sure you don't forget to make it repeat this forever, otherwise it will only follow the mouse once when the program starts, then stop!



Congrats! You have finished making the paddle!

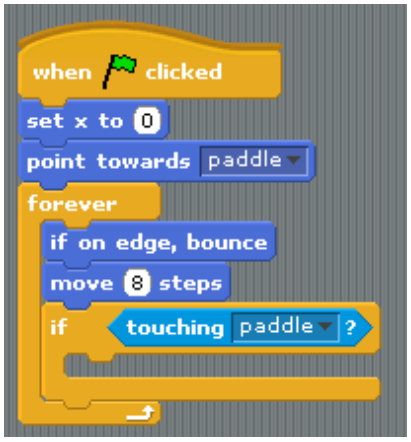
The next thing we need to make our program do is to make the ball move. To do this, we first want to center it in the middle of the screen by setting x to 0 and y to 0 with bricks from the motion section. We then want to give the ball a direction to move in. It makes sense for the ball to first hit the paddle, so that the user isn't completely surprised by the ball. To do this, add a "point towards" block, and choose to point towards the paddle.

Now that we have set up the initial state of the ball, we need to tell it how to move during the game. We want the ball to keep moving forward, unless it hits a wall or the paddle. To do this, add a forever loop to the bottom of the stack and inside ask the sprite to keep moving 8 steps forward. Try this and see how it works.

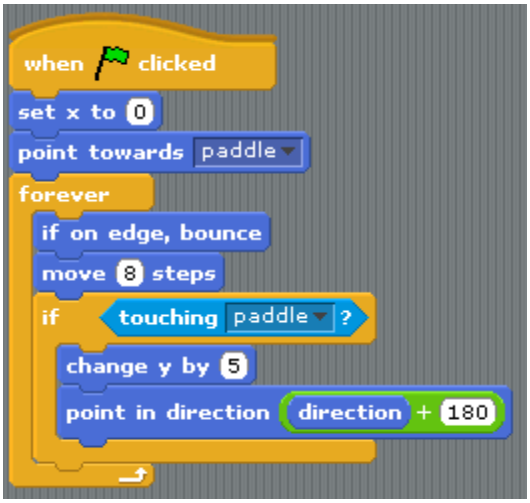


As you probably noticed, the ball goes off the side of the screen and doesn't come back! To make the ball bounce when it hits the edge of the screen, just add the brick "if on edge, bounce" from the motion tab above the "move 8 steps" brick. Now try your program.

Now the ball will bounce off the edges, but it still doesn't do anything when it hits the paddle. We need to change this! The first thing the ball will need to do is to check if it is touching the paddle. To do this, add an if statement inside the forever loop, then go to the sensing tab and choose the "touching <>?" diamond. Select "paddle" and put it inside the if statement.

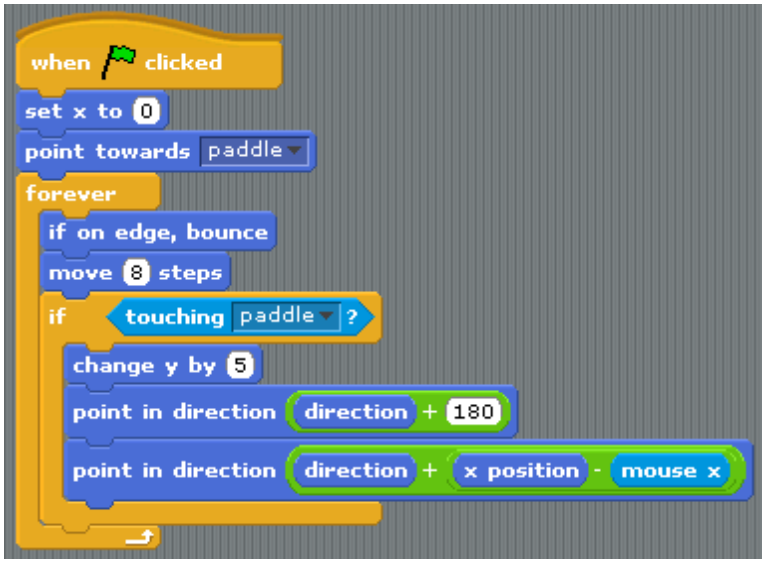


Whenever the ball hits the paddle, we want the ball to move up 5 pixels so that it doesn't keep hitting the paddle, and also to change directions. To do this, use a "change y by" block to change y by 5, moving the ball up a little bit. We can then reverse the direction by setting the direction to direction + 180 (in other words, go back in the direction that it came from). Try your game now!



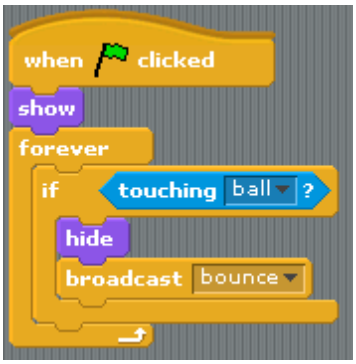
You may have noticed that the ball always just returns to where it came from at the same angle. We want our user to have a little bit of control over the angle, and be able to change where it goes depending on where the ball hits the paddle. For this, we will just tweak the direction a little bit depending on where the ball hits relative to the paddle. Because the paddle has the same x coordinate as the mouse, we can just get the x coordinate from the mouse.

Add another "point in direction" block after the first one. In this one, we will set the direction to "direction + (x position – mouse x)". This makes it so that the further away from the center of the paddle you hit the ball, the more ball will go at an angle closer to that wall. This is because the if the ball hits to the left of the middle, it will change the direction by a negative amount, and if it hits the ball to the right of the middle, it will change by a positive number. This will let the user control the ball.

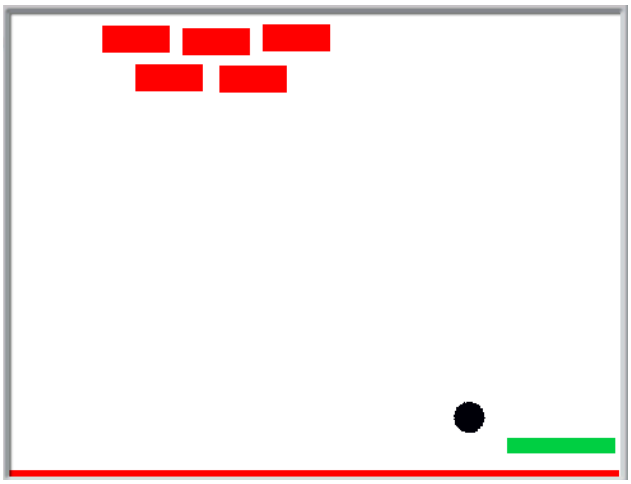


Now that we have our ball and our paddle working, we can work on the bricks. All the bricks have to do is to tell the ball when they have been hit so that the ball can bounce off of them, then disappear. Start by making sure that the brick is visible when the program starts. Then have the brick check forever if it has been hit by the ball. When it has been hit by the ball, we want it to first disappear, then tell the ball that it hit a block so that it will bounce.

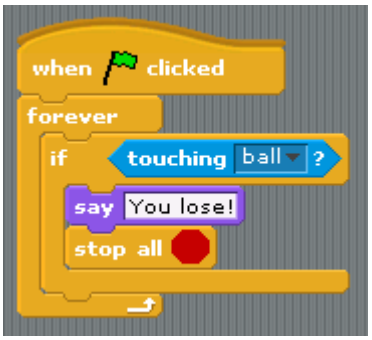
To do this, drag a starting block to the middle and put a "show" block from the looks tab underneath it so that every time you play the game the ball will be made visible. After that, add the forever loop and ask the brick to keep checking if it is touching the ball (you can find the "touching ball" diamond under the sensing category). Once we know that the brick is touching the ball, we can ask it to hide itself (using the brick found under the looks tab). Once the brick has disappeared, we can ask the ball to bounce by adding a broadcast brick to the bottom. Create a new message, probably called "bounce" to send after the block has hidden itself.



Now we need the ball to respond to the message to bounce off of bricks. To do this, start with a "when I receive" block, and ask the ball to change its direction by 180 degrees (go back in the direction it came from).



Now there is only one thing left to do: detect if the ball hits the bottom of the screen. To do this, you can make a long rectangular sprite at the bottom of the screen beneath the paddle. You can then ask it to keep looking if the ball every hits it. When this happens you can either subtract a life and start over or end the game.



Great! You now have a complete game of breakout!

For more ideas on upgrades, check the next page!

EXTRAS!!!

Here are some extra things you might want to add:

CHECKING WHEN ALL OF THE BRICKS ARE GONE:

To do this, create a variable for holding the number of bricks. When the game starts, have every brick increase the amount of this variable by one, so that you know the total number of bricks. Every time a brick hides itself, ask it to decrease the number by one. When the variable reaches zero again, you know that all of the bricks are gone and you can go on to the next level.

CHANGE COLOR ON THE BOUNCE

To do this, just ask the ball to either change its color by a certain amount (this brick can be found under the looks tab) or to change to the next costume after you create costumes of multiple colors.

OTHER THINGS YOU CAN ADD

Other good upgrades are adding sounds and keeping score when the user destroys bricks.

Multiple levels are always good.