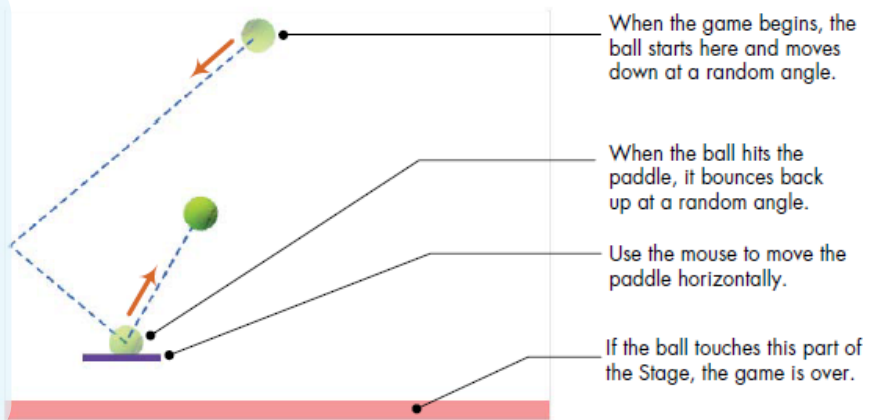


# Your First Scratch Game - PONG

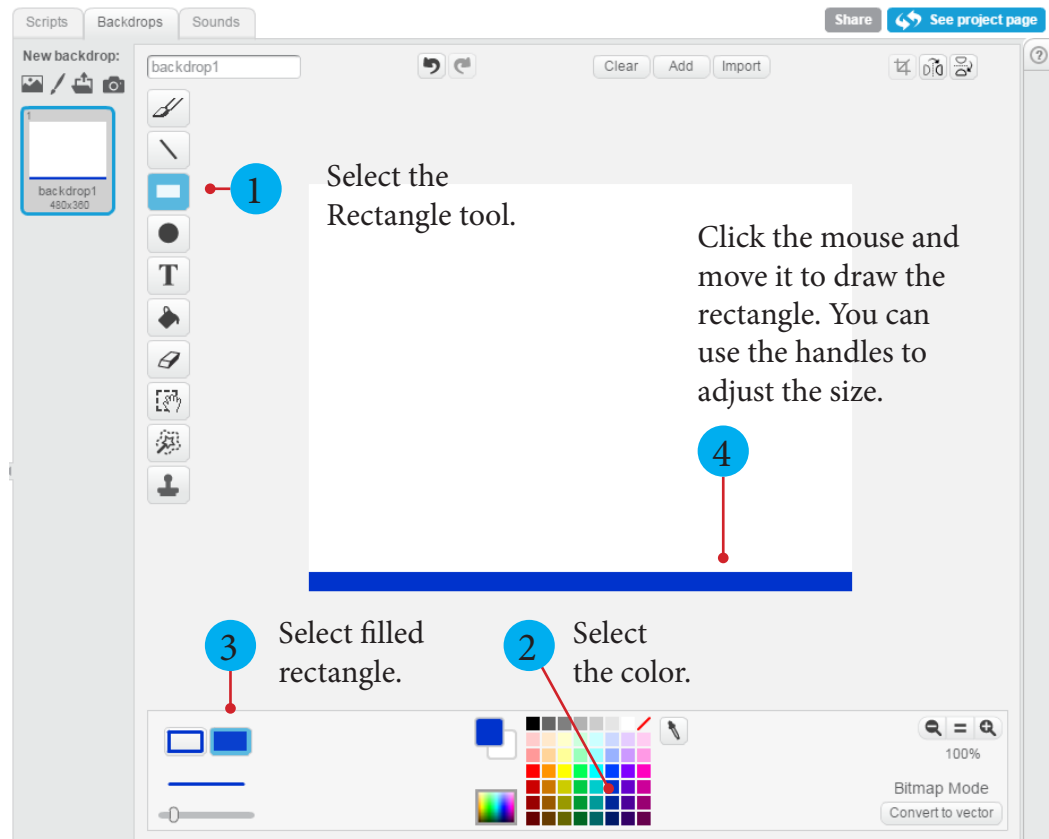
In this module, you'll create a single-player game in which players will move a paddle to keep a bouncing tennis ball from hitting the floor, based on the classic arcade game Pong.

As shown in the picture on the right, the ball starts at the top of the Stage and moves down at some random angle, bouncing off the edges of the Stage. The player moves the paddle horizontally (using the mouse) to send the ball back up. If the ball touches the bottom of the Stage, it's game over.




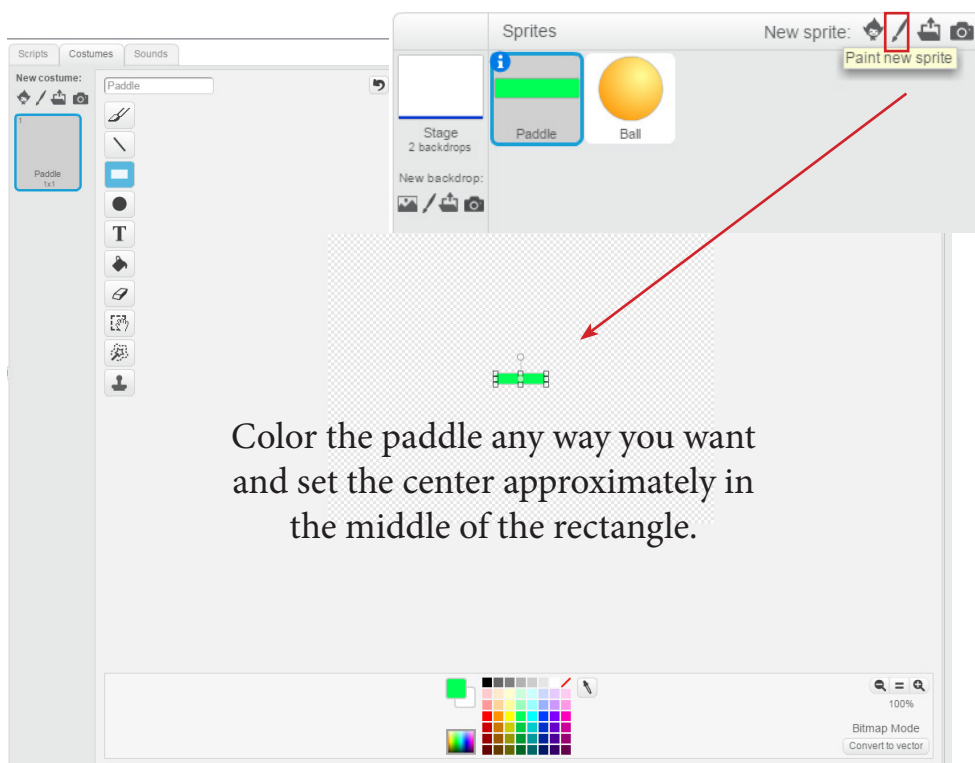
## Step 1. Prepare the backdrop

To detect when the ball misses the paddle, we'll mark the bottom of the Stage with a certain color and use the touching color ? block (from the Sensing palette) to tell us when the ball touches that color. Click the thumbnail of the Stage to select it and then go to the **Backdrops** tab. Follow the steps below to draw a thin rectangle at the bottom of the Stage's backdrop.



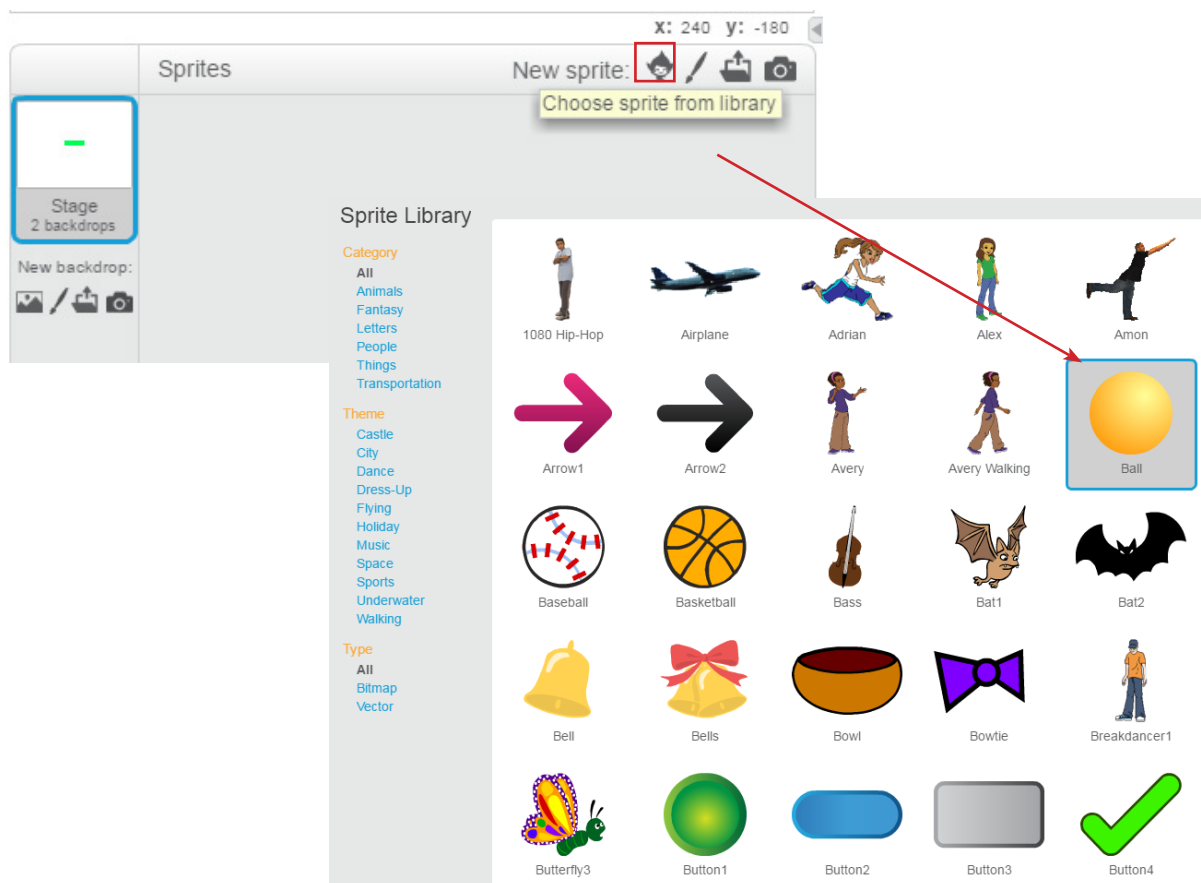
## Step 2. Add the Paddle

Click the Paint new sprite button  above the Sprite List to add the Paddle sprite to your project. Since the paddle is just a thin, short rectangle, repeat what you did in Step 1. Rename the sprite to something that explains what it is; I called it Paddle



## Step 3. Add the Ball

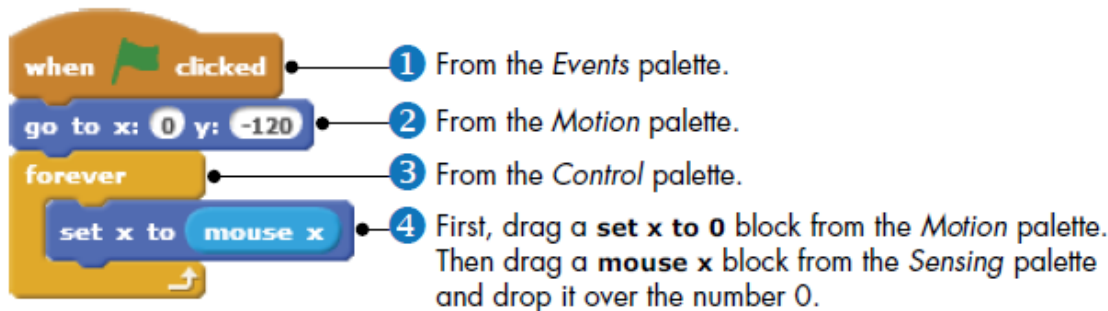
Choose sprite from library above the Sprite List to import one. In the dialog that appears, click the Things category and select the Tennis Ball image to add that sprite to your project. Rename the sprite as Ball.



## Step 4: Start the Game and Get Your Sprites Moving

As the designer for this game, you'll decide how players can start a new round. For example, the game could begin when you press a key, click a sprite on the Stage, or even clap or wave your hands (if you have a webcam). The green flag icon (located above the Stage) is another popular option, which we'll use here. The idea is simple. Any scripts that start with the when green flag clicked trigger block start running when you press that button. The flag turns bright green and stays that way until the scripts finish.

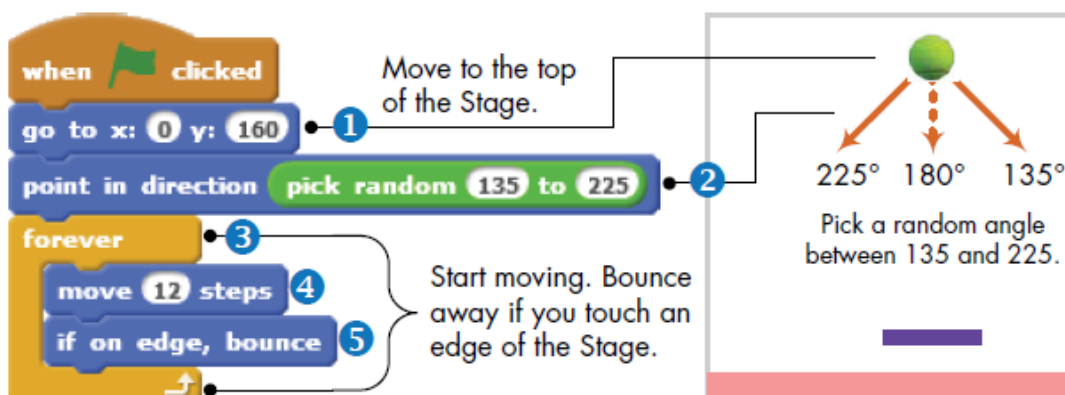
### PADDLE SPRITE: The script for the Paddle sprite



When the green flag is clicked, the go to x: y: block v sets the paddle's vertical position to -120, just in case you previously moved it with the mouse. The paddle should hover just above the pink rectangle at the bottom of the Stage, so if your rectangle is thicker, change its position number to something that works for your design.

The script then uses a forever block w to constantly check the mouse position. We'll move the paddle back and forth by matching the paddle's x-position to that of the mouse x. Run the script (by clicking the green flag icon) and try moving your mouse horizontally; the paddle should follow. Click the stop icon next to the green flag to stop the script.

### BALL SPRITE: The first part of the Ball sprite script



First, we move the ball to the top of the stage u and make it point down at a random angle using the pick random block v (from the Operators palette).

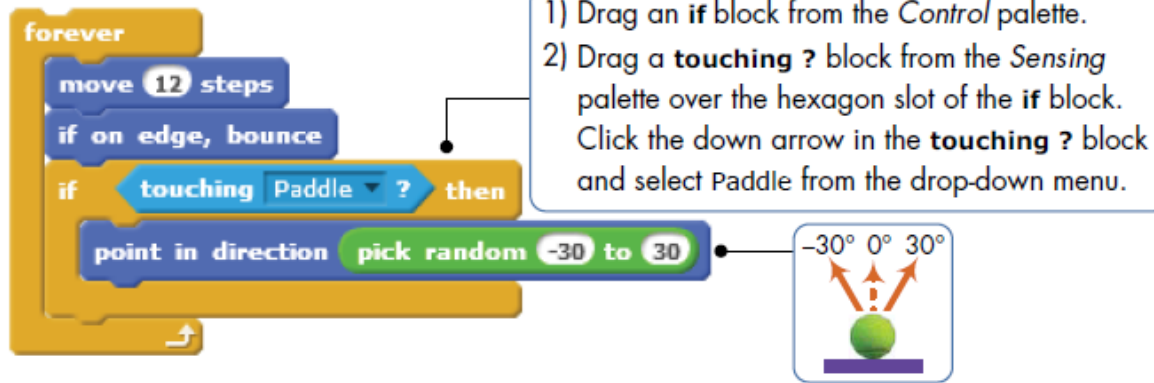
The script then uses a forever block w to move the ball x across the Stage and bounce y off the edges. Click the green flag to test what you've written so far. The ball should move in a zigzag pattern, and the paddle should still follow your mouse.

### TRY IT OUT

Replace the 12 inside the move block with different values, run the script, and watch what happens. This should give you an idea of how to make the game easier or harder to play. Click the stop icon when you're done.

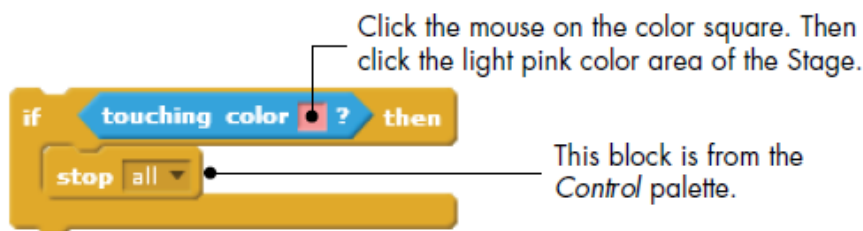


### BALL SPRITE: Make the ball travel upwards when



We can modify the forever block we just created so the ball travels upward when it hits the paddle. When the ball and paddle touch, we command the ball to point in a random direction between  $-30$  and  $30$ . When the forever block goes for the next round, it will execute the move block, which will now cause the ball to go up. Click the green flag again to test this part of the game. Click the stop icon when you are sure the ball is bouncing off of the paddle as it's supposed to.

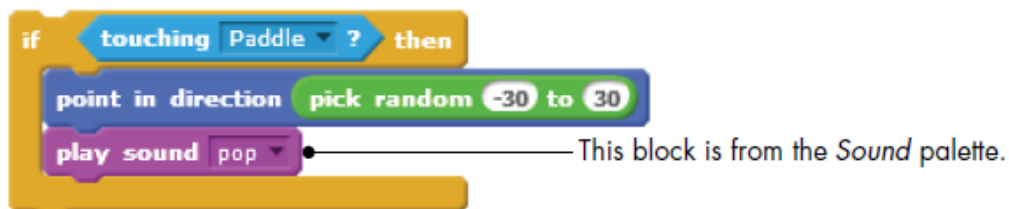
### BALL SPRITE: Stop the game when the ball touches the bottom of the Stage



Add the script to the Ball sprite, either right before or after the if block. You'll find the touching color ? block in the Sensing palette and the stop block in the Control palette.

## Step 5: Spice It Up with Sound

### BALL SPRITE: Stop the game when the ball touches the bottom of the Stage



Double-click the ball on the Stage to select it and then select the Sounds tab. Click the Choose sound from library button to add a sound to the Ball sprite. In the dialog that appears, select the Effects category, choose the pop sound, and click OK to add it to the Sounds tab. After that, go back to the Scripts tab and insert a play sound block (from the Sound palette)

**CONGRATULATIONS** Your game is now complete

## 1. CHALLENGE TASK

🕒 try duplicating the Ball sprite to have two (or more) balls in your game and see how that changes the way you play! Show others how you modified your game



## EXTRA TIME: ARITHMETIC OPERATORS

Scratch supports the four basic arithmetic operations of addition (+), subtraction (-), multiplication (\*), and division (/). The blocks used to perform these operations, called operators, are shown in the picture below. Since these blocks produce a number, you can use them as inputs to any block that accepts numbers, as demonstrated in this figure.

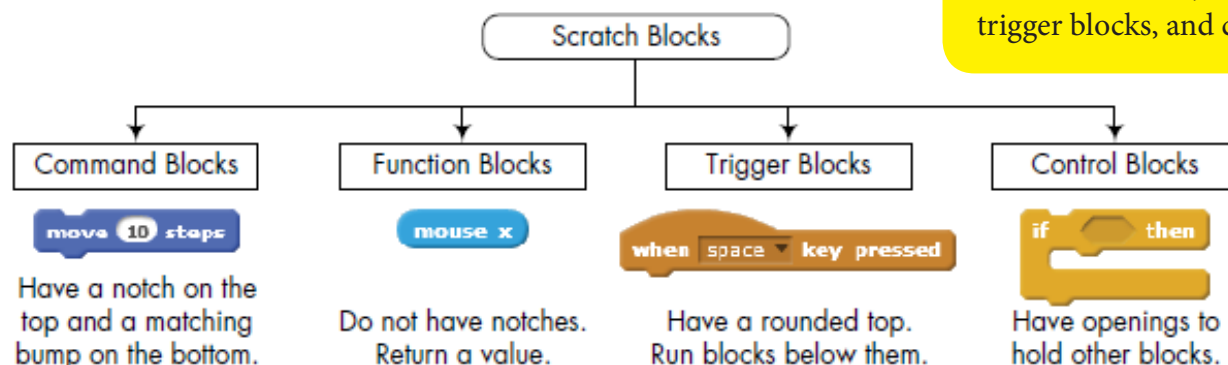
Example				
Operator				
	Addition	Subtraction	Multiplication	Division

1. Use the say command and the appropriate blocks from the Operators palette to calculate the following:

- The square root of 32
- The sine of 30°
- The cosine of 60°
- The result of rounding 99.459

REFERENCE: Use extra time to get familiar with different scratch blocks

As shown in the graph, Scratch has four kinds of blocks: command blocks, function blocks, trigger blocks, and control blocks.



**Command** blocks and **control** blocks (also called stack blocks) have bumps on the bottom and/or notches on the top. You can snap these blocks together into stacks

**Function** blocks can't form a layer of a script alone; instead, they're used as inputs to other blocks. The shapes of these blocks indicate the type of data they return. For example, blocks with rounded ends report numbers or strings, whereas blocks with pointed ends report whether something is true or false ->>>>>

**Trigger** blocks, also called hats, have rounded tops because they are placed at the top of a stack. Trigger blocks connect events to scripts. They wait for an event—such as a key press or mouse click—and run the blocks underneath them when that event happens.

	Function blocks with rounded ends report numbers or strings.
	Function blocks with pointed ends report Boolean (true/false) values.