Introducing Scratch

1 – Squash Game

Scratch is a fun and easy way to get started with the important principles of computer programming. Creating projects in Scratch allows users to become familiar with common programming concepts such as variables and control structures. Unlike most programming languages though, with scratch there is no need to learn any complicated commands as it is all visual. Programs are created by dragging blocks in to a script area in a logical and simple way.

Not only is Scratch a great way to get started with basic programming, it's also free!

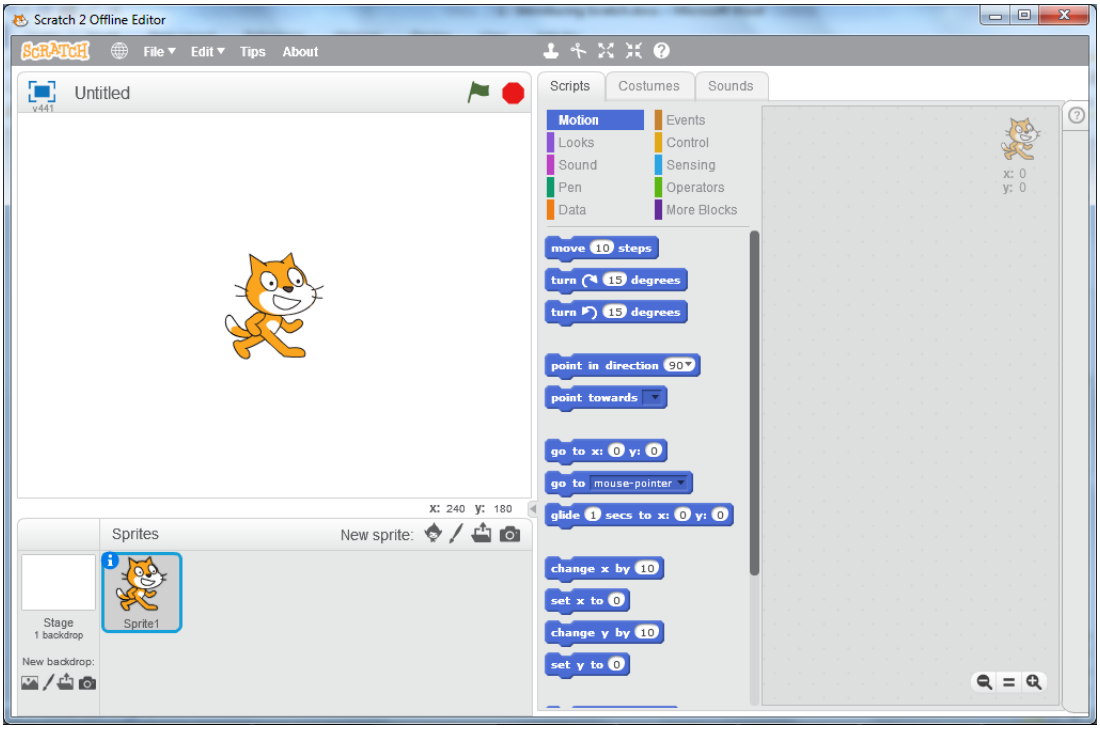
You can use Scratch through your web browser at https://scratch.mit.edu/. You can also download Scratch for Windows, Linux and Mac to use offline from http://scratch.mit.edu/download. The images shown in these exercises are from the offline version.

In the following exercises we'll be using Scratch to create a few simple programs. At the end of each exercise there is a section for more advanced students to add a few enhancements to their program.

To begin with we'll make a small squash game.

Exercise 1. Setting up the Stage

When you open Scratch, there are three main sections to the screen.



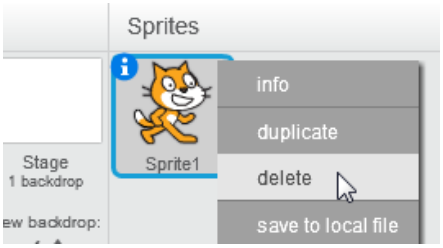
On the left we have the **Stage** where all the action takes place, with your **Sprite List** below that.

In the middle section we have the **Blocks Palette**. This is where you find instructions for making things happen in your program.

In the right is the **Scripts Area**. This is where you piece programming instructions together.

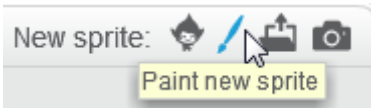
Characters and objects on your stage are known as **Sprites**. You can import sprites or draw your own sprites for use in your programs. Each time you start a new Scratch project you will begin with the Cat sprite but we won't use that one for now.

1. The sprites list in the bottom left section of the screen currently only includes the cat (labelled something like Sprite1). **Right-click** on the sprite and then select **Delete** to remove it.

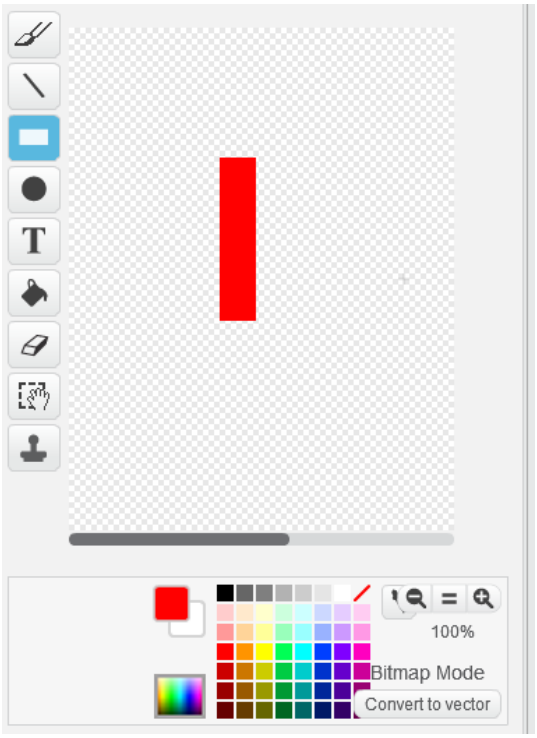


Exercise 2. Creating the Bat

1. Click the **Paint new sprite** button at the top of the **Sprites Area**.



The right section of the screen will change to an editing area for creating your new sprite. We are going to draw a rectangle that will be used as the bat in our bat and ball game.



2. Select a colour from the colour palette at the bottom and then select the **Rectangle** drawing tool.
3. Next to the colour palette, select the filled shape option.

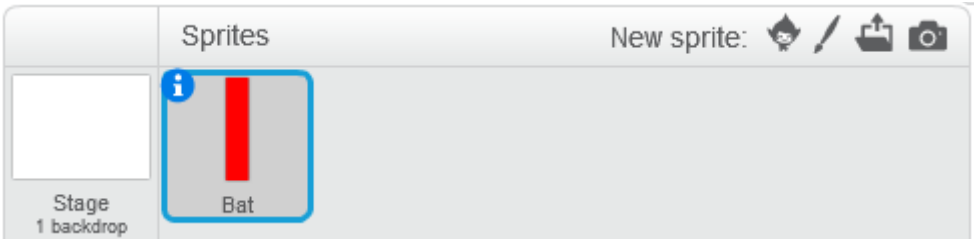
1. Draw a rectangle similar to the example to the right. As you draw the sprite, you will see a copy of the rectangle appear in the stage area.
2. Click the Set costume center button in the top right corner of the editing area. 
3. Drag the cross to tell the program where the centre of your sprite will be. This can be very important in some programs.

4. In the **Sprites area**, there is a small **i** symbol in the corner of the sprite's icon. Click on that symbol to view the sprite's information.

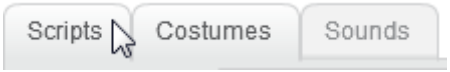
4. Change the name of the sprite to Bat.

5. Click the  icon to finish viewing the sprite information.

Before we add any programming blocks, it is important to make sure your **Bat** sprite is still selected in the **Sprites area**. And programming instructions we add will apply to whatever sprite we currently have selected.

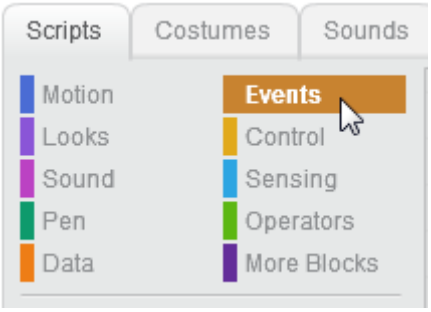


6. Click on the **Scripts** tab at the top to change from sprite edit view to the sprite scripts view.



Script blocks are grouped in to various colour coded categories. The top section of the scripts are has buttons for selecting each category.

7. Select **Events** from the Scripts categories.



When you create a program in Scratch, you can run the program by clicking the green flag  icon at the top of the stage. The first script block is used to specifying what will happen when the program starts.

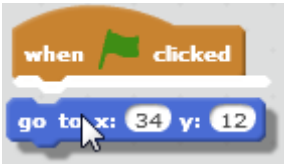
8. Drag the **when**  **clicked** block in to the blank script area. Place it near the top so there is plenty of room to put other blocks under it.

We will add an instruction that tells the program to move the bat to a certain part of the stage (a starting point) every time the program begins.

9. Select the Motion category.

The script blocks are designed to fit together like lego. You can attach a block to the bottom of another block which means the program will do them one after another.

10. Find the go to block from the **Motion** category and attach it to the bottom of the **when**  **clicked** block in the scripts area.

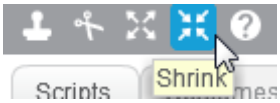


The **x:** value is for the horizontal position (sideways) while the **y:** value is for the vertical position (up and down). For each value, 0 is right in the middle.

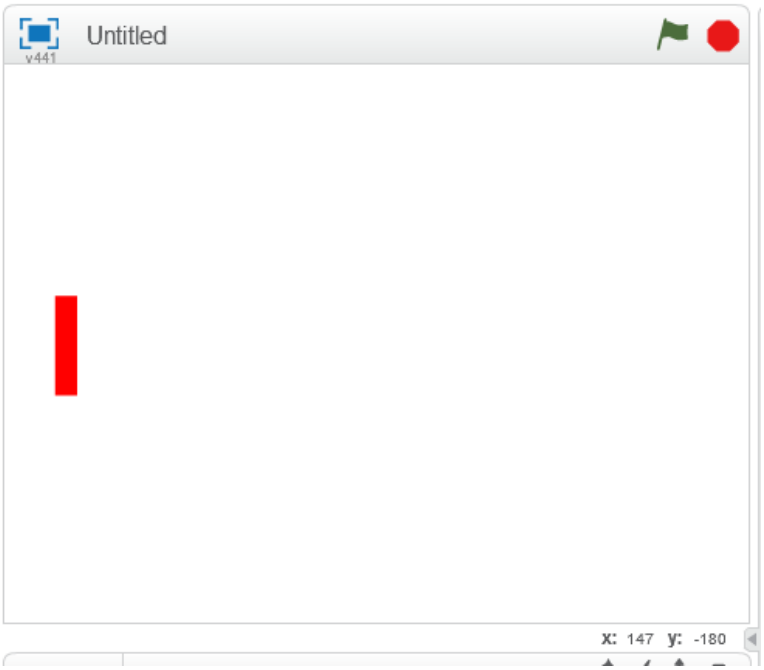
11. Change the **x:** value to -200. Since it is a negative number, that means it will be to the left of the middle. In fact that will be pretty close to the left edge of the stage. Change the **y:** value to 0 so that it will be right between the top and bottom of the stage.

12. Test your program by clicking the  button at the top of the stage. Your bat sprite should move to the left of the stage.

Now we've got the bat starting in the right spot, we'll get it the right size. At the top is a row of buttons that includes a Grow and Shrink button.



13. You can click on the **Grow** or **Shrink** icons and then click the sprite you want to change the size of. Use the grow and shrink buttons until you bat is a similar size to the example below.



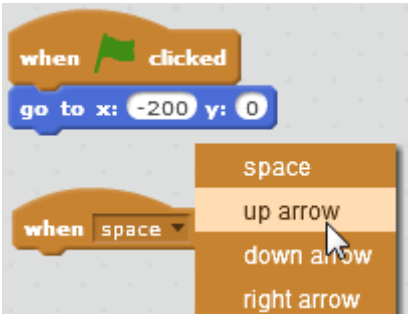
Exercise 3. Controlling Sprites

1. Make sure your Bat sprite is still selected in the Sprites area.
2. Select the Events category from the blocks palette.

The second control block is **when ____ key pressed** which allows you to tell the program what to do when a certain key is pressed. It is usually set to the spacebar key by default.

We will set up our program so that pressing the up arrow key will move the bat upwards and pressing the down arrow key will move the bat downwards.

3. Move the **when ____ key pressed** in to the blank space below your existing blocks.
4. Change the option to **up arrow**.



5. Add another similar block further down (leaving a bit of room between them) and change the key to **down arrow**.
6. Select the **Motion** category.
7. Find the **Change y by 10** block and add it to the bottom of the **when up key pressed** block.

Now every time you press the up arrow on your keyboard, the bat will move a distance of 10 pixels upward.

8. Add a similar block to the down arrow block, except that this one needs to have a negative number as show. Now when you press the down arrow, the bat will move 10 pixels downward. If you hold down the up or down arrow, the bat will move continuously in that direction

Tip You can test any group of connected blocks in the script area by clicking on the group.

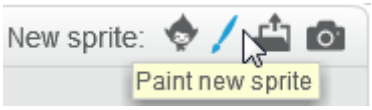
Before we continue it would be a good idea to save our project. Saving in Scratch is the same as in most programs.

9. Press **Ctrl S** or select **Save** from the **File** menu to save your file. Enter a suitable file name such as *Bat and Ball*.

Exercise 4. Creating the Ball

The ball in our game is going to be a simple circle that will bounce around the screen.

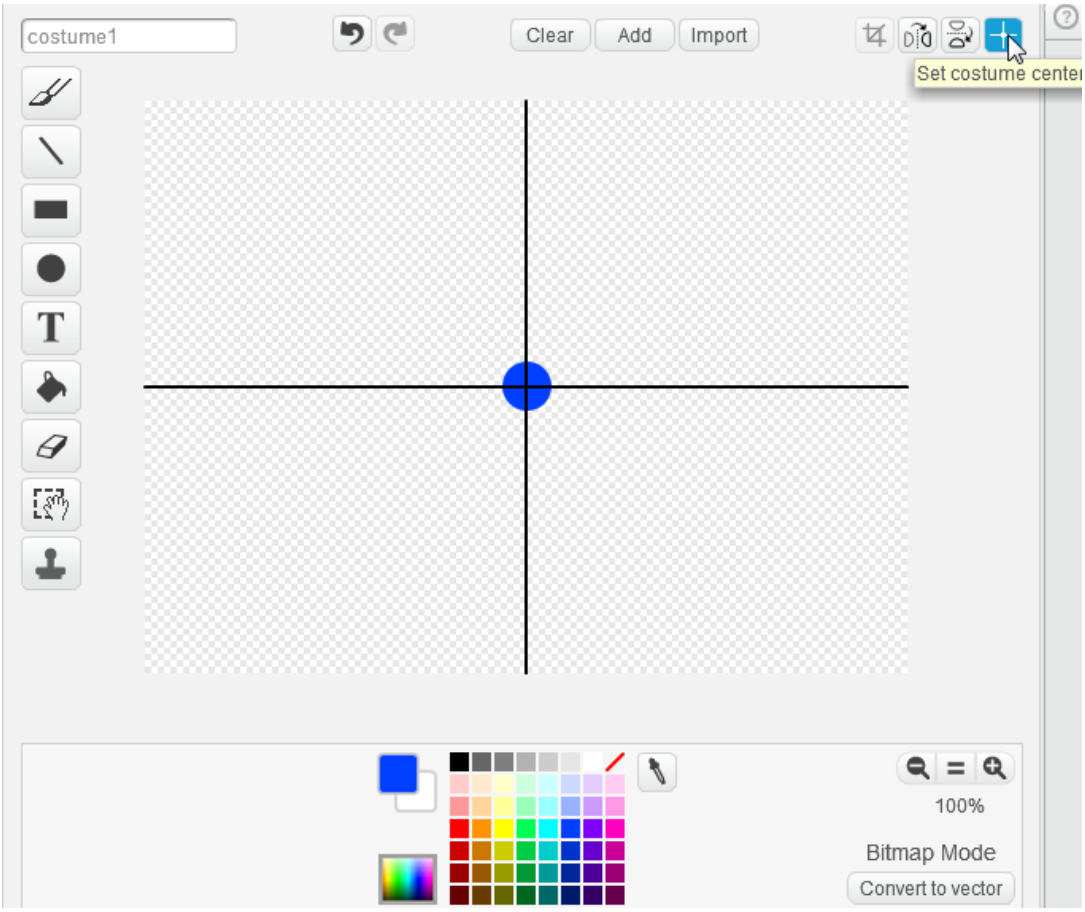
1. Click on the **Paint new sprite** button.



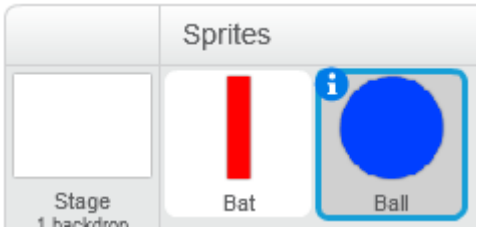
2. Draw a circle in your choice of colour that is a similar size to the example below.

Tip When you draw a circle with the oval tool, hold down **shift** while you draw it to make sure the height and width are equal.

3. Make sure the **costume centre** is set to the middle of the circle.



4. The new sprite will be in the Sprite are next to your Bat sprite. Change the name of the new sprite to *Ball* (remember to click the **i** icon in the corner of the sprite's icon to change its name).



5. Make sure the Ball sprite is still selected and click the Scripts tab to view the **Scripts Area**.

6. Select the **Events** category and add a 

block to your scripts area.
7. From the Motion category, add a **go to x: y:** block under the previous block. Set the **x** number to 200 (without the minus sign this time) and set the **y** number to 0.

Now each time the button is clicked, the *Bat* and *Ball* sprite will both move to the specified locations. We will now add some additional instructions to get the ball moving. You can test this by moving both of them to different positions and then clicking the button. Make sure the Ball sprite is selected again before continuing.

8. Make sure you are still in the **Motion** category and add the **move 10 steps** block to the bottom of the existing blocks.
9. Click the button to test the block. The ball will move once and then stop. We want the ball to keep on moving so we will need to add a control block.
10. Remove the **move 10 steps** block by dragging it back in to the Blocks palette to the left (make sure you drag only that last block and not the entire group. Dragging a block will also move any blocks attached underneath). This is how you delete a block
11. From the **Control** category find the **Forever** block and add that to your existing blocks (under the go to block). Anything inside a forever block will keep on repeating for as long as the program is running.
12. From the **Motion** category locate the **move 10 steps** block. This time instead of placing it on the bottom you will need to place it inside the forever block.
13. From the **Motion** category add an, **if on edge, bounce** block right underneath the **move 10 steps** block. It should look like the example below.

14. Click the button to test your program. The ball should now keep on moving and bounce when it hits the edge of the stage.
15. Click the button to stop the program.

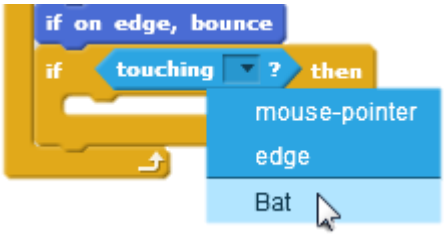
Currently the ball is ignoring the bat since we haven't told it what to do when it touches the bat. That is our next job.

We will add a block which contains a condition. This block will check to see if the ball is touching the bad. If it is touching the bat, then it will run some additional blocks which we will add.

16. Under the **Control** category, find the **if __ then** block (it is listed below **forever** block). Add this block below the **if on edge, bounce** block.

The blank space next to the word if is where we put the condition the block is looking for. In this case, we want it to check to see if the ball is touching the bat.

17. Select the **Sensing** category and locate the **touching** block, at the top of the list.
18. Drag the touching block to the small gap in the **if** block. The arrow in the touching block allows you to select from the objects in your program. Click the arrow and then select *Bat* from the list of options.



Now we can add blocks that specify what will happen when the ball touches the bat.

19. From the **Motion** category select the 

block and drag it to the middle of the **if** block. Change the number to 180.
20. Test the program and the ball will now bounce back each time it touches the bat.

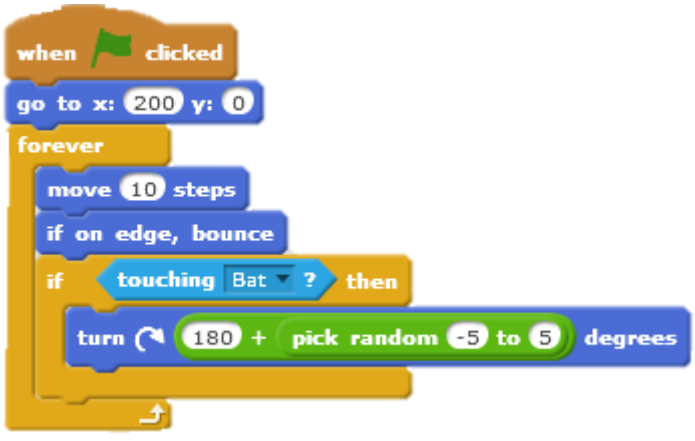
Going back in exactly the same direction is a little boring so we'll make it change direction slightly each time. We can do that by making the direction of the turn slightly different each time using a random number block.

21. The **Operators** category is where we find all our number related blocks. Select the **+** block from the Operators category 

and drag it to the space where the number *180* is. Type *180* in the first space.

The second number, which will be added on to 180 will be randomly generated.

22. Locate the pick random 1 to 10 block and drag it in to the second gap. Change the numbers to -5 and 5. The whole block should now be adding any random amount from -5 to 5 on to the 180º turn.
23. Test your program. Now each time the ball touches the bat it will bounce back on a slightly different angle.



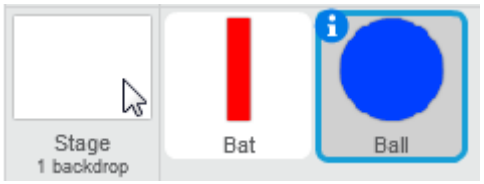
24. Save your completed program and test it.

Exercise 5. Additional things to try

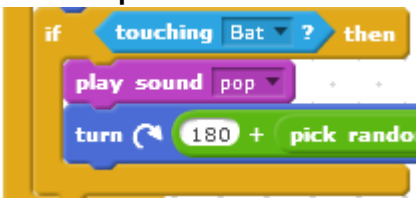
Here's a few things you can add to make the program even better

- **Set the starting direction**. Each time you run the program it might keep on moving at the same angle it was last time you ran it. To prevent this you can add a block which will make it face the same way every time the program starts. You can do that by adding a block right under the When clicked block. Make sure the angle is set to - 90 and not just 90 if you want it moving to the left.

- **Change the stage background**. Vanilla a bit boring? Click on the stage in the Sprites list. From the top of the **Scripts** area click on the **Backdrops** tab. Click the **Choose backdrop from library** button and then choose a background picture. Scratch includes a variety of background pictures grouped under categories such as Sports and Outdoors. You can also paint a background if you choose to.



- **Add a sound** when the ball hits the bat. Select the Ball sprite. Before you can use a sound you need to import one. Click the **Sounds** tab at the top of the **Script** area. Click the **Choose sound from library** button and you can choose one of the sounds that comes with Scratch (such as the pop sound in the effects folder) or another sound you have on your computer.
You can then go to the **Sounds** category and add a **play sound** block.



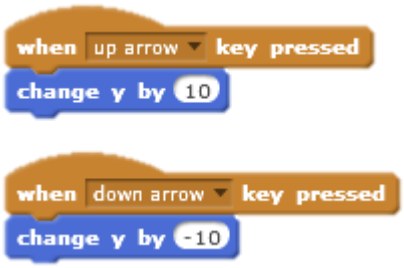
- **Change the difficulty**. You can make the game harder or easier by changing some of the numbers in your blocks.
    - Change how much the angle randomly changes in your 

 block
    - Change the speed of the ball by changing the number in your 

 block
    - In the bat scripts, change how fast your bat moves by changing the numbers in your 

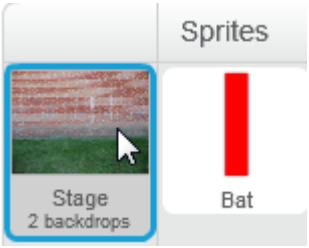
 block.

- **Improved movement**. At the moment there is a pause when you change direction and the movement isn't very smooth. You can fix that by putting your movement controls inside a forever loop like in the following example. Try it and see the difference.

Replace	With
	

Exercise 6. Adding Scoring

Firstly we will need to add a variable. The variable will be a container for a number that can change – in this case, our score.



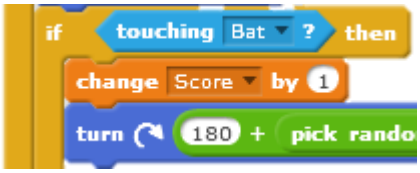
1. Select the **Data** category.
2. Click the **Make a Variable** button.
3. For the variable name enter *Score* and click **OK**.

A score block will appear on your stage and a selection of script blocks related to that variable will appear in the scripts area.

You can move the variable anywhere on the stage you like. You can also right-click the variable on the stage to switch between different views but we'll leave it on **Normal Readout** for now.

4. Select the **Ball** sprite from the sprite list.

We already have a section that tells the program what to do when the ball touches the bat. We'll add an instruction that makes the score increase by one every time this happens.

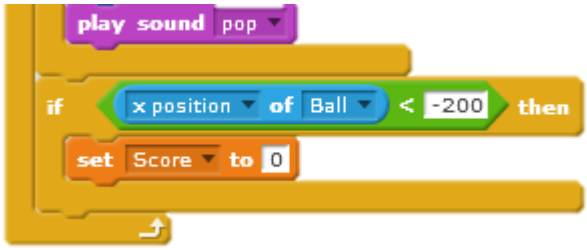


5. In the **Data** category, find the **change score by 1** block and add it to the **if touching bat** section you already have in your script area.
6. Test your program. Each time the bat touches the ball, the score will increase by 1. Now we'll add some blocks that will set the score to 0 when the ball misses the bat.

7. Add another **if** ___ **then** block underneath the current one. We will add a condition that checks to see if the **x** position (left and right) of the ball has gone past a certain point. If it has gone beyond a certain point, we will know that it has missed the bat.
8. From the **Operators** category, select a **less than** block  and add it to the blank space in the **if** ___ **then** block.

Tip Can't remember which one is the greater than symbol and which one is the less than symbol? The less than symbol **<** looks a lot like a letter **L** that's been tilted. **L** for **L**ess than.

9. From the Sensing category, select a **x position of ball** block  and add it to the first blank space in the **less than** block.
10. In the second blank space, enter the number -200 (with the minus sign).
11. From the **Data** category, find a **set score to 0 block** and place it inside the **if** ___ **then** block. The completed section should look like the example shown to the right.

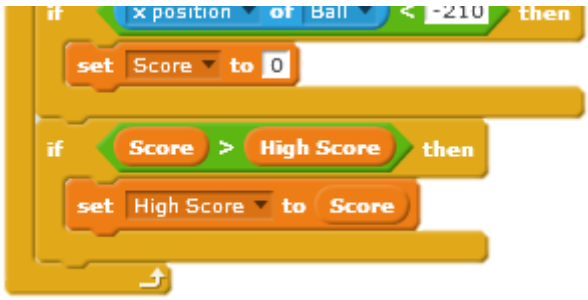


12. We will add one last block that sets the score to 0 when the is clicked. Otherwise every time you start a new game, the score will continue from the previous game.
13. Add another **set score to 0 block** just under the When clicked block.
14. Test your game. The score should now increase by one every time the ball touches the bat and return to 0 every time it misses the bat. The score should also return to 0 each time the is clicked.

Exercise 7. Adding a High Score

There's no point in getting a great score if it's going to vanish as soon as you miss. A high score variable will allow you to keep track of your best result.

1. Select the **Data** category.
2. Click **Make a Variable** and enter **High Score** for the variable name.
3. Move it to a convenient location on your stage so that it isn't overlapping your Score variable or getting in the way when you play.
4. Select the **Ball** sprite.
5. Use what you have already learned to add additional blocks as shown below.



Congratulations! In this simple exercise you have demonstrated some key programming concepts including:

- Working with Variables – the Score and High Score.
- Working with Control structures
  - Sequence – steps in a program happening one after another
  - Selection – choices within the program such as the ones using **If** blocks.
  - Iteration – steps being repeated in a program such as the ones in the forever block.

Tip You can test your program in full screen mode by clicking the button in the top left corner of the stage. You can then press the same button or press **Esc** to exit full screen mode.