# Introducing Scratch

## 2 – Bat & Ball Game

In this exercise we will create a game that will include sprites that move and interact with each other. We will also make use of common programming **Control Structures** such as **Sequence** (instructions happening one after another), **Selection** (decisions) and **Iteration** (loops). We will also make use of **Variables** which are an essential way to work with data in computer programs.

We are going to make a bat and ball / squash game.

## 1. Creating the Bat

1. If you still have a project open from the previous exercise, make sure it is saved and then go to the File menu and select New.

2. If Scratch asks you if you want to replace the contents of the current project, click OK.
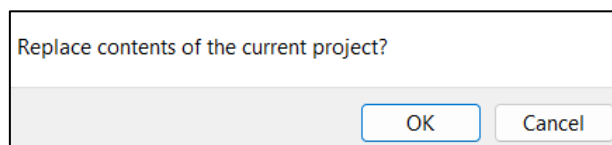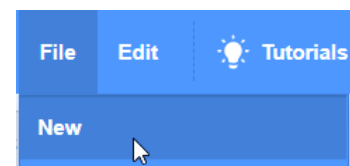
3. The sprites list in the bottom right section of the screen currently only includes the cat (labelled something like *Sprite1*). We won't be using the cat for this project. If the cat sprite is not already selected, click on it. You will see a small bin icon in the corner of the sprite. Click this bin icon to delete the sprite.
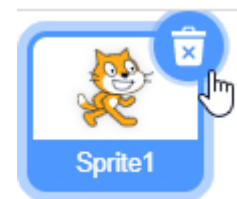
In the bottom of the sprites area is a choose a sprite button. Moving your mouse over it brings up a list of ways to create a new sprite including painting one, importing a picture or choosing one from Scratch's built in library.

1. Move your mouse over the choose a sprite icon and click Paint.
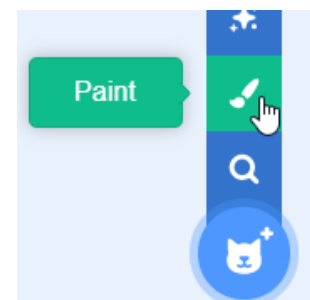
The middle section of the screen will change to an editing area for creating your new sprite. We are going to draw a rectangle that will be used as the bat in our bat and ball game.

2. Click the **Fill** colour icon above the editing area.

3. Use the colour, saturation and brightness sliders to select a colour for your bat.
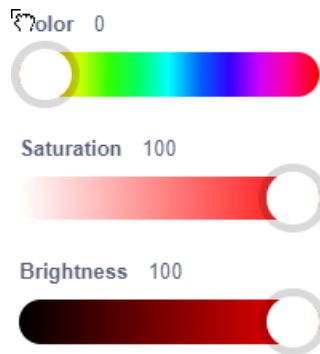
4.  Click on the fill icon again or anywhere else outside the colour options to close the options.

5.  Click the **Outline** colour icon.

6.  We will leave the outline colour as black, which might help to make the bat more visible later on when we change the background. You can use the option next to the Outline colour to choose how thick the outline will be. 2 should be a good thickness.

7.  Click the **Rectangle** icon to the side of the editing area.

8.  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.

Notice that there is a small crosshair shape in the middle of the sprite editing area. This indicates where the centre of the sprite is. In many programs it is important to make sure the middle of your sprite lines up with this cross. Otherwise, your program might not work as intended. For example, if the program thinks the middle of the sprite is somewhere near the bottom of your bat, then that can affect the way things move in the game later on.

9.  Click the Select tool to the left of the editing area.

10. Drag the rectangle you have just drawn to make sure its centre lines up with the centre of the editing area.
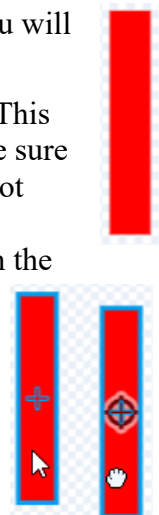
| Note | A sprite can contain more than one image. Each image is referred to as a **Costume**. Some programs might require a sprite to show different costumes. |
|---|---|

In the **Sprites area**, the section across the top that shows information and additional options about the current sprite. One of the options is the name of the sprite which would currently be something similar to Sprite2. You can call sprites whatever you want but your program will often need to refer to sprites by name, so it is helpful to give each sprites names that make it easy to identify them in your code.

11. Change the name of the sprite to **Bat**.



Before we add any programming blocks, it is important to make sure your **Bat** sprite is still selected in the **Sprites area**. Any programming instructions we add will apply to whatever sprite we currently have selected.
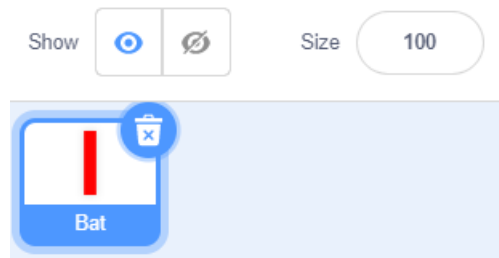
12. Click on the **Code** tab at the top to change from sprite edit view to the sprite code view.

**Note**    you can go back and make further changes to your sprite if needed by clicking on the Costumes tab again.
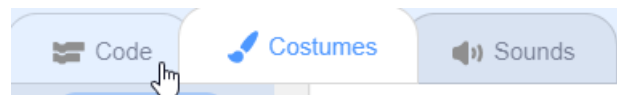


Code blocks are grouped into various colour coded categories. Buttons for selecting these categories are shown down the side of the code blocks.

13. Select **Events** from the code categories.

When you create a program in Scratch, you can run the program by clicking the **go** button at the top of the stage. The first code block in the events category is used to specify what will happen when the program starts.

14. Drag the **when** 🏳 **clicked** block in to the blank code 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.

15. Select the **Motion** category.

The code 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.

16. Find the **go to** block from the **Motion** category and attach it to the bottom of the **when** 🏳 **clicked** block in the code area.
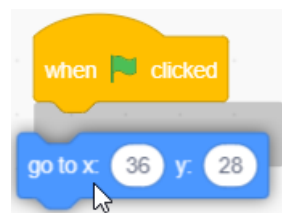


The numbers that are already in that code block are based on the current position of this sprite. You can see the same numbers at the top of the sprites area while the sprite is selected. The x value is for the horizontal (sideways) position. The y value is for the vertical (up and down) position. 0 is right in the middle so that for the x coordinates, positive numbers are to the right while negative numbers are to the left. For the y axis positive numbers are toward the top while negative numbers are toward the bottom.

17. 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.
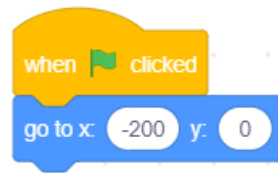


18. Test your program by clicking the 🏳 button at the top of the stage. Your bat sprite should move to the left of the stage.

19. Now we've got the bat starting in the right spot, we'll get it the right size. At the top the sprite area is a Size option. Change it to 75 and press **Enter**. The sprite on the stage will display at 75% of its normal size.
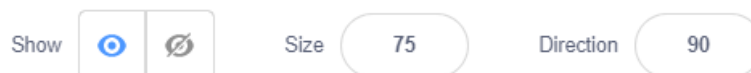


20. Make adjustments to the sprite's Size option until it looks similar on the stage to the example shown below.



## Exercise 1.   Controlling Sprites

1.  Make sure your **Bat** sprite is still selected in the **Sprites** area.

2.  Select the **Events** category in the code blocks on the left.

The second Events 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** into 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 arrow pressed** block. Remember that y position refers to vertical (up and down) direction.
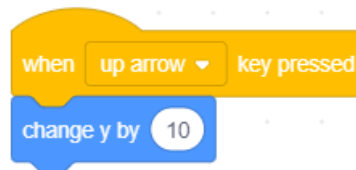


Now every time you press the up arrow on your keyboard, the bat will move a distance of 10 pixels upward.

8. Add a similar block to the down arrow block. Since this one is moving down instead of up it needs to have a negative number as shown to the right. 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 individual block or any group of connected blocks in the code area by clicking on them. |
|---------|---------|

Before we continue it would be a good idea to save our project.

9. From the **File** menu at the top select **Save to your computer**. Select a suitable save location and enter a suitable file name such as *Bat and Ball*.
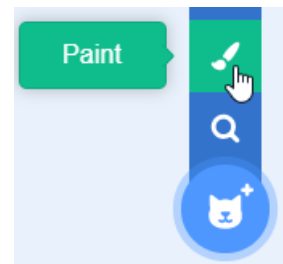
From now on save your project regularly.
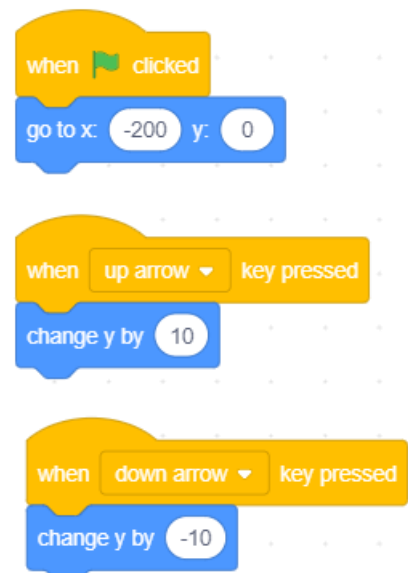
## Exercise 2.   Creating the Ball

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

1.  Select **Paint** from **Choose a Sprite**.

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 **sprite centre** is lined up with the middle of the circle.
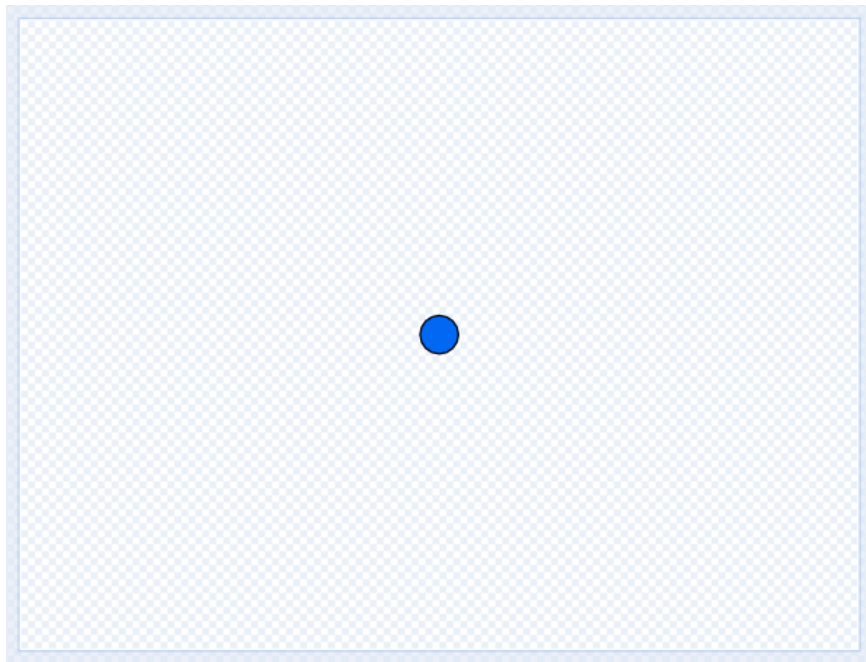
4.  The new sprite will be in the Sprite area next to your Bat sprite. Change the name of the new sprite to *Ball*.

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

6.  Select the **Events** category and add **when** 🏁 **clicked** block to your code area.

7.  From the Motion category, add a **go to x: y:** block under the **when** 🏁 **clicked** 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. The bat will start on the left while the ball will start on the right. You can test this by dragging both of them to different positions on the stage and then clicking the 🏳 button. Make sure the Ball sprite is selected again before continuing.

We will now add some additional instructions to get the ball moving.
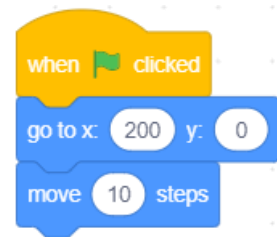
8.  Make sure you are still in the **Motion** category and add the **move 10 steps** block to the bottom of the existing blocks.

9.  Click the 🏳 button to test the block. The ball will move once and then stop. We want the ball to keep on moving repeatedly so we will need to add a control block.

10. **Remove** the **move 10 steps** block by dragging it back into the Blocks section 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 so that it is also inside the forever block. Since they are both inside the forever block and both will be repeating it won't matter which of the two is first. Your code should look like the example below.
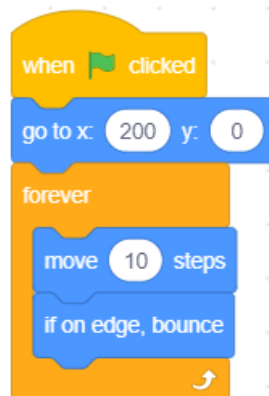
14. Click the 🏳 button to test your program. The ball should now keep on moving and bounce when it hits the edge of the stage.

15. Click the Stop 🔴 button next to 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 an **IF** condition. This block will check to see if the ball is touching the bat. 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 the **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.
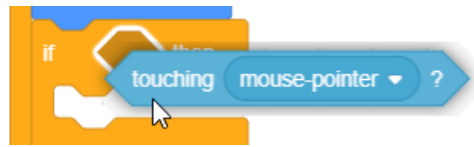


19. The arrow in the touching block allows you to select from the objects in your program. Click the arrow and then select *Bat* from the list of options.
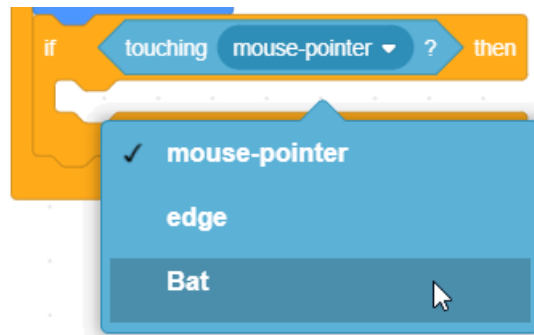


Now we can add blocks that specify what will happen when the ball touches the bat.

20. From the **Motion** category find the **turn degrees** block. There is one for clockwise turning and another for counter-clockwise turning. At the moment it won't matter which one you choose.

21. Drag it to the middle of the **if** block. Change the number to 180.
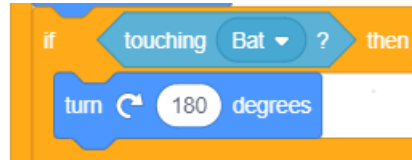


22. Test the program and the ball will now bounce back each time it touches the bat. You can use the up and down arrow keys to move the bat out of or into the path of the ball.

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

23. The **Operators** category is where we find all our number related blocks. Select the **pick random** block from the Operators category and drag it to the space where the number *180* is. We want the number to be randomly picked from a range a bit above 180 and a bit below 180.
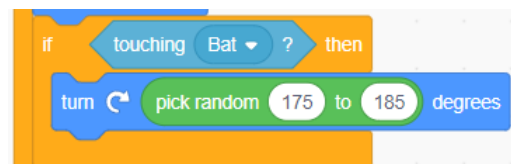


24.  Type *175* in the first space and *185* in the second space.

25. Test your program. Now each time the ball touches the bat it will bounce back on a slightly different angle.

26. Save the changes to your program and test it.

## Exercise 3.   Additional things to try

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

▪ **Set the starting direction**. A sprite always has a direction it is facing in, though that might be hard to tell with a circle. You can see the current direction the sprite is facing in at the top of the sprite options. You can even see this amount changing while you test your program.

Each time you run the program it might keep on moving at the same angle it was moving in the 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 since that will point left.
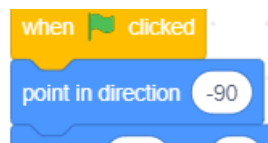
▪ **Change the stage background**. Vanilla a bit boring? To the right of the Choose a Sprite icon is a **Choose a Backdrop** icon. Click this icon to choose a backdrop from the library. Scratch includes a variety of background pictures grouped under categories such as **Sports** and **Outdoors**. You can also paint a background of your own if you choose to. Make sure you use a backdrop that isn't too close the colour of your sprites, otherwise you will have trouble seeing them.

▪  **Add a sound** when the ball hits the bat. Select the **Ball** sprite. Click the **Sounds** tab at the top of the **Code** area. Any sounds currently contained in this sprite are listed on the left. Your sprite might already have the default "pop" sound available but there are plenty of additional built-in sounds to choose from. At the bottom of the sounds list is a **Choose a Sound** button. Click this button and you can choose one of the sounds that comes with or another sound you have on your computer.
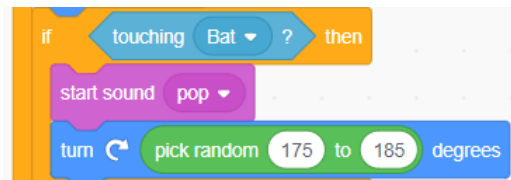
You can then go to the **Sounds** category in the **Code** area and add a **start sound** block.

▪ **Change the difficulty**. You can make the game harder or easier by changing some of the numbers in your blocks.

  o  Change how much the angle randomly changes in your `pick random 175 to 185` block

  o  Change the speed of the ball by changing the number in your `move 10 steps` block

  o  In the bat code, change how fast your bat moves by changing the numbers in your `change y by 10` block.

## Exercise 4.   Smoother Movement

You might have noticed that the bat doesn't move very smoothly when you press the up and down arrow keys. This is because each time you press up or down, you are activating different groups of code blocks.

It can run a lot more smoothly if you contain all of your movement within the same group of blocks. We will do this by adding some code to the **when** 🏳 **clicked** block which will check to see if the up or down keys have been pressed and then move accordingly.

Change the code for your bat so that the code shown on the left is replaced with the code shown on the right.
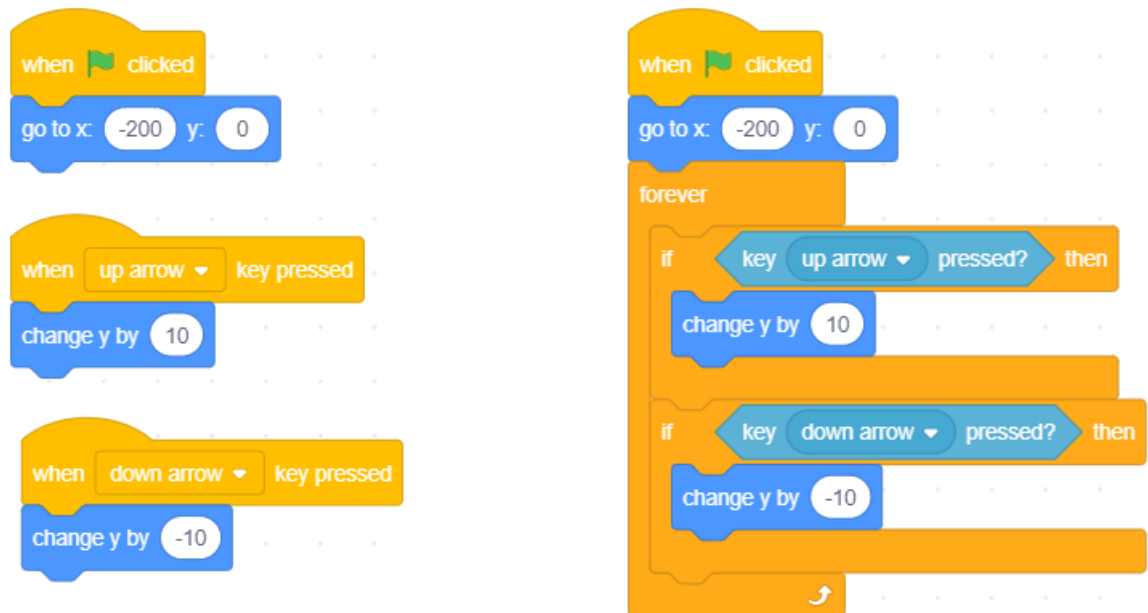


Test your program again by clicking the 🏳 button and then use your up and down arrow keys to see the how much more smoothly the bat now moves.

## Exercise 5.   Adding Scoring

**Variables** are frequently used in computer programs when we need a certain bit of information stored for use in the program. In this game we will use a variable to contain the score. The program code will update the value of this variable where needed.

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 **Variables** category.

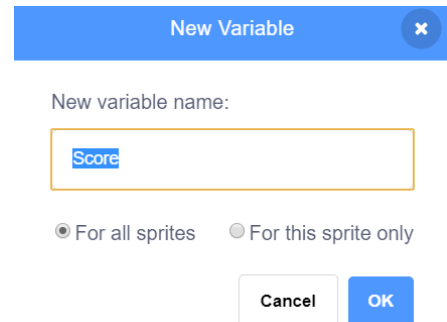2.   Click the **Make a Variable** button at the top of the code blocks.

3.   For the variable name enter *Score.* Make sure **For all sprites** is selected and click **OK**.

A score showing the value of your variable will appear on your stage. 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 area.

We already have a section that tells the program what to do when the ball touches the bat. We'll add an instruction to that same section that makes the score increase by one every time the ball touches the bat.
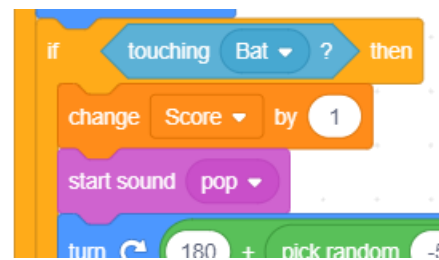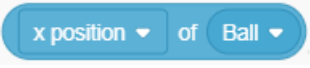
5.   In the **Variables** category, find the **change __ by 1** block. Use the list to change it to **change score by 1** and add it to the **if touching bat** section you already have in your code area.

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.   Select the **Backdrop** in the **Stage** area to the right of the sprites area.

8.   Add some code to the stage so that it checks to see if the **x** position (left and right) of the ball has gone past a certain point. If it has gone beyond -200 where the bat is, we will know that it has missed the bat.

9.   Add the following code blocks in the **stage** code area.
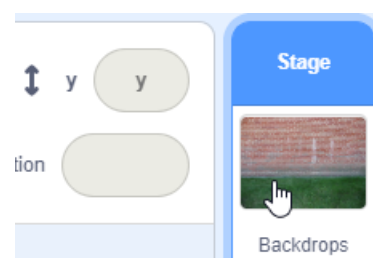
10. From the **Operators** category, select a **less than** block and add it to the condition space in the **if ___ then** block.

**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.

11. From the Sensing category, select the ___ of ___ block. `backdrop # ▼ of Stage ▼`

12. Change the second part of the block to **Ball** and then change the first part of the block to **x position**. `x position ▼ of Ball ▼`

13. Add it to first part of the less than block.

14. In the second blank space, enter the number -200 (with the minus sign).

15.  From the **Variables** category, find a **set ___ to 0 block** and change the variable in the block to **score**. Place it inside the **if ___ then** block. The completed section should look like the example shown to the right.

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.

16. Add another **set score to 0 block** just under the **When 🚩 clicked** block.
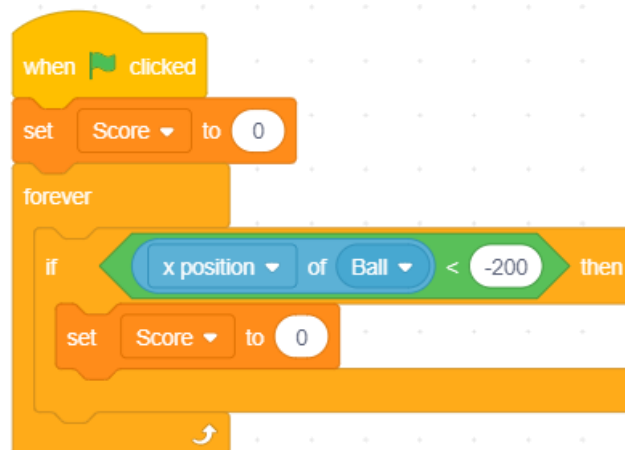


17. Test your game. The score should now increase by one every time the ball touches the bat and return to 0 every time it misses the bat. The score should also return to 0 each time the 🚩 icon is clicked.

## Exercise 6.   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.  Click on the **Ball** sprite in the sprites list.

2.  Select the **Variables** category.

3.  Click **Make a Variable** and enter **High Score** for the variable name.

4.  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.

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

   o   Sequence – steps in a program happening one after another

   o   Selection – choices within the program such as the ones using **If** blocks.

   o   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. |
|-----|---|
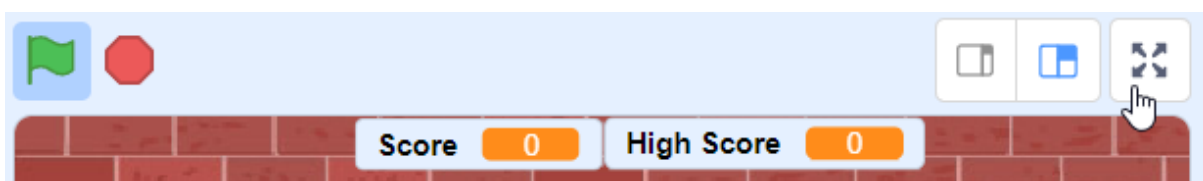
## Exercise 7.   Code Reference

### Completed Bat Sprite

```
when [flag] clicked
go to x: -200 y: 0
forever
    if < key (up arrow ▼) pressed? > then
        change y by 10

    if < key (down arrow ▼) pressed? > then
        change y by -10
```

### Stage

```
when [flag] clicked
set (Score ▼) to 0
forever
    if < (x position ▼) of (Ball ▼) < -200 > then
        set (Score ▼) to 0
```

### Completed Ball Sprite

```
when [flag] clicked
go to x: 200 y: 0
point in direction -90
forever
    move 10 steps
    if on edge, bounce
    if < touching (Bat ▼) ? > then
        change (Score ▼) by 1
        start sound (pop ▼)
        turn ↻ (pick random 175 to 185) degrees

    if < (Score) > (High Score) > then
        set (High Score ▼) to (Score)
```

## Exercise 8.   Extra Challenge

Try changing your single player game into a 2 player **Pong** style game.



Sample code on the next page but try creating your own version first.

## Blue Player

```
when 🏳 clicked
go to x: -210 y: 0
forever
    if   key w ▾ pressed? then
        change y by 10
    if   key s ▾ pressed? then
        change y by -10
```

## Red Player

```
when 🏳 clicked
go to x: 210 y: 0
forever
    if   key up arrow ▾ pressed? then
        change y by 10
    if   key down arrow ▾ pressed? then
        change y by -10
```
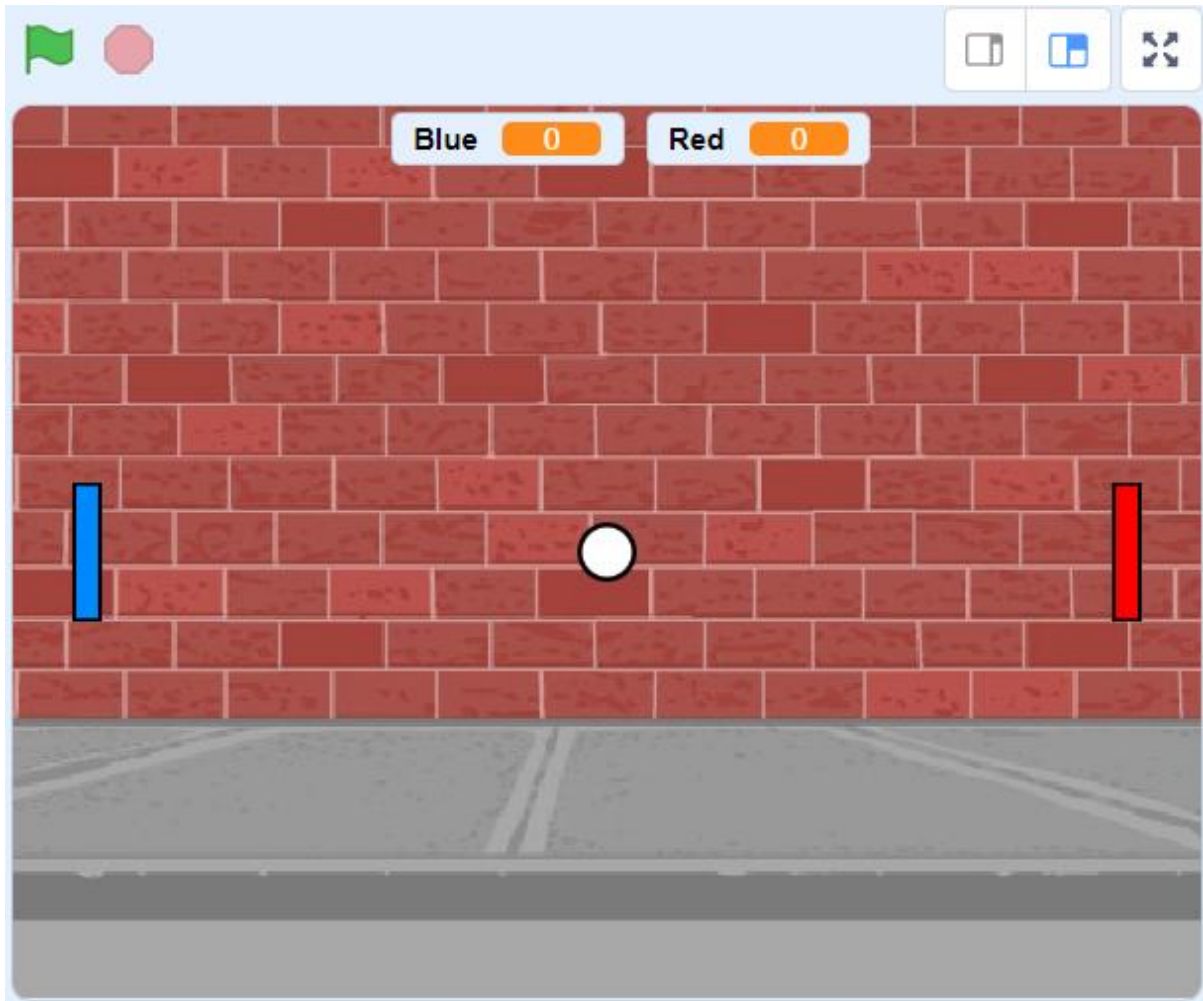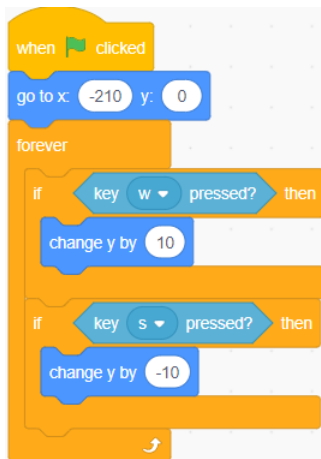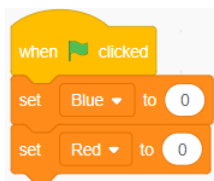
## Backdrop

```
when 🏳 clicked
set Blue ▾ to 0
set Red ▾ to 0
```

## Ball

```
when 🏳 clicked
point in direction 90
go to x: 0 y: 0
forever
    move 10 steps
    if on edge, bounce
    if   touching Blue Player ▾ ?  or  touching Red Player ▾ ?  then
        turn ↻ pick random 170 to 190 degrees
    if   x position ▾ of Ball ▾ < -215 then
        change Red ▾ by 1
        go to x: 0 y: 0
        point in direction pick random 80 to 100
        wait 1 seconds
    if   x position ▾ of Ball ▾ > 215 then
        change Blue ▾ by 1
        go to x: 0 y: 0
        point in direction pick random -80 to -100
        wait 1 seconds
    if   Blue = 5 then
        say Blue wins for 2 seconds
        stop all ▾
    if   Red = 5 then
        say Red wins for 2 seconds
        stop all ▾
```