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.

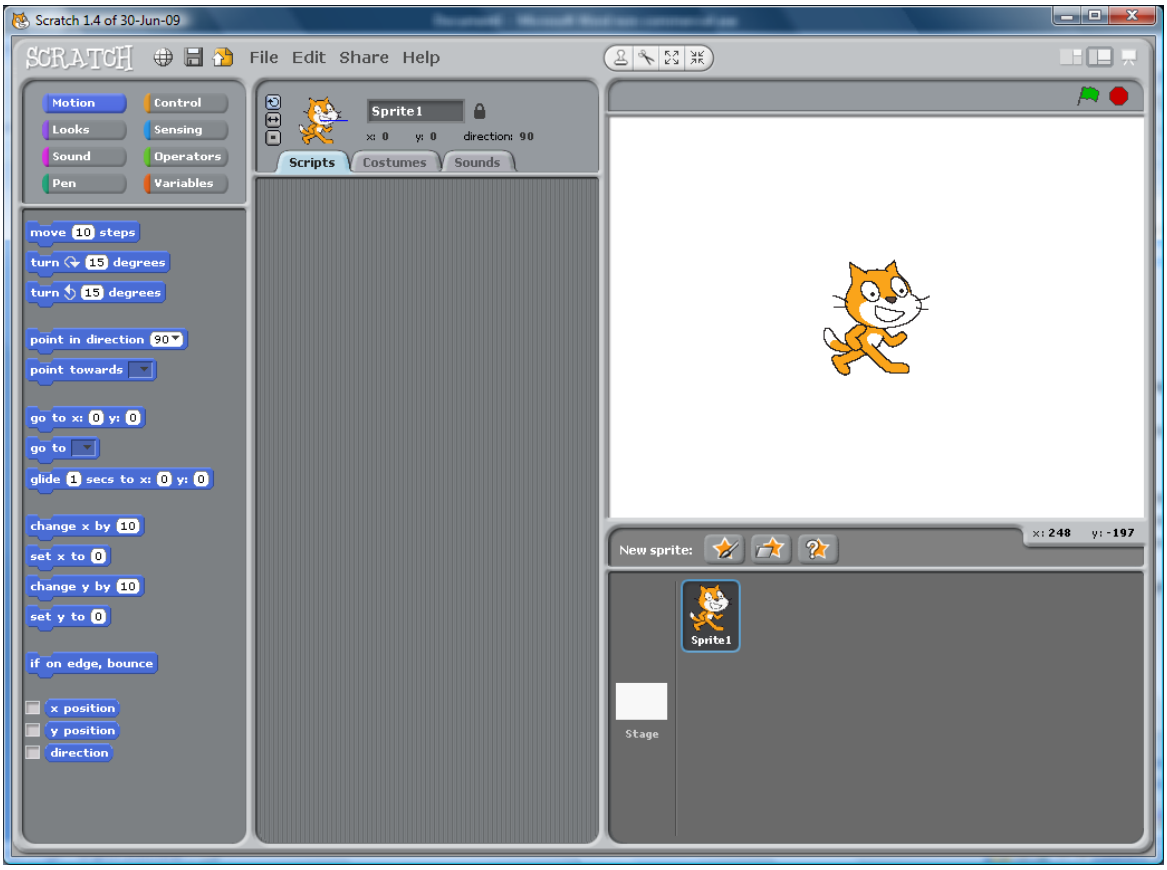
Not only is Scratch a great way to get started with basic programming, it's also **free**! You can download Scratch for Windows, Linux and Mac from http://scratch.mit.edu/download.

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 3 main columns in the screen.



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

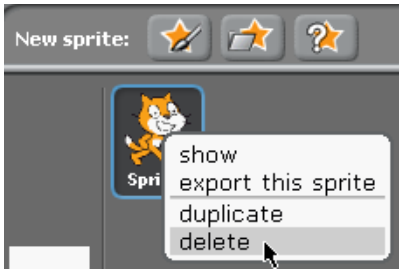
In the middle is the **Scripts Area**. This is where you piece programming blocks together.

In the right section you have the **Stage** where all the action takes place, with your **Sprite List** below that.

In the top right corner you will see icons for the **View Modes** which let you change how much room the stage takes up. Leave it on the middle option for now. 

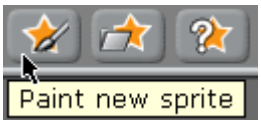
Characters and objects on your stage are known as **Sprites**. You can import sprites or draw your own sprites for use in your programs. Each time you start a new Scratch project you will begin with the Cat sprite but we won't use that one for now.

1. In the **Sprite List** in the bottom right corner, **right click** on Sprite1 (the cat) and select **Delete** to remove it.

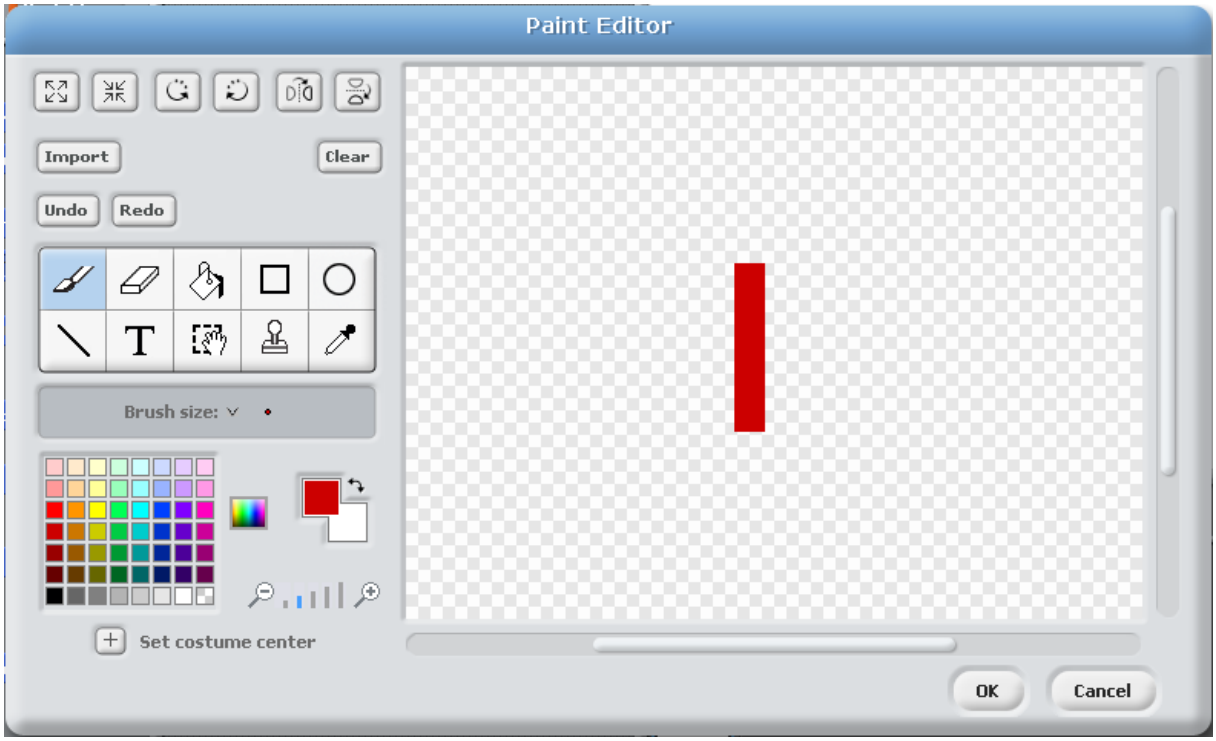


Exercise 2. Creating the Bat

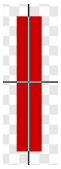
1. Click the **Paint new sprite** button.



2. Select a colour and then use the **Rectangle** drawing tool ☐ to draw a rectangle like the one below.

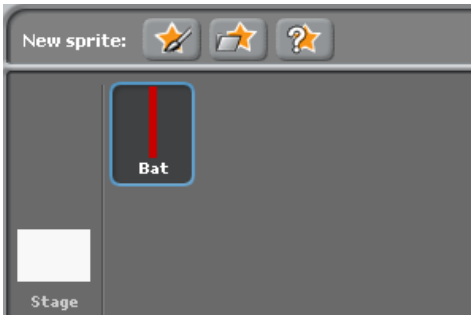


3. Click the ⊕ **Set costume center** button and then drag on the drawing area to make sure the centre is in the middle of your rectangle



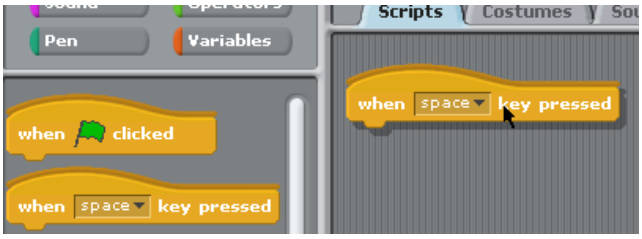
4. Click **OK** to finish editing your sprite.
5. Edit the sprite name in the top of the window and change the name to *Bat*.

The name of the sprite will now show under the sprite's icon in the Sprite List.



6. Make sure the *Bat* sprite is still select as any programming blocks we add in the **Scripts Area** will apply to the object we have selected.
7. Above the **Blocks Palette** are buttons for selecting a block category. Click on the **Control** button. This will allow us to select from blocks which are used to control what happens in your program.

8. Drag the **when space key pressed** block to the top of the **Scripts Area**. This block allows us to control what happens when a certain key is pressed.



9. Click the arrow next to space and select *up arrow*. The block should now say when up arrow is pressed. We will add a programming block that tells the program to move the Bat sprite upward when the up arrow key on the keyboard is pressed.

10. Select the **Motion** category

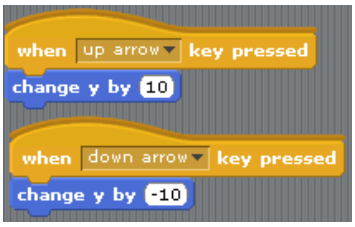
In Scratch, left and right movement is controlled by the X axis while up and down movement is controlled by the Y axis. To make our sprite move up, it needs to move on the Y axis.

11. Find the change y by 10 block. 

12. Drag it to the Script Area so that it locks on to the bottom of the **when up arrow key pressed** block. You will notice that they fit together, which is why the blocks are shaped with small tabs on them like jigsaw puzzle pieces.

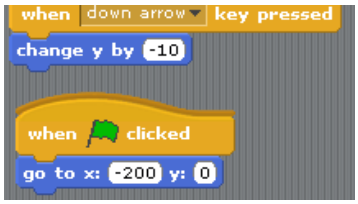
13. Press the up arrow on your keyboard. Each time you press the up arrow key, your sprite will now move upward 10 pixels. You can change the number in the block to make it move more or less each time.

14. Repeat steps 8 to 14 to add a new block for when the down arrow is pressed. Note that you will need to change the number to a negative number so that it will move down instead of up.



You can now move the sprite up and down using your arrow keys.

15. From the **Control** blocks, drag the **When**  **clicked** block in to the **Scripts** area. It doesn't matter where you place it but there should be plenty of room below the other blocks. This block allows us to provide instructions for what happens when the  button is clicked to start the program. We will use it to set the starting location of our bat.
16. From the **Motion** category, add the **go to x: y:** block to the bottom of the  block. Set the *x* number to *-200* and set the *y* number to *0*. It should look like the example below.



17. Click the green flag  in the top right corner of the stage to test the block. The bat should move to the middle left of the stage.

Tip You can test any block in the scripts area by clicking on the block. Clicking a block will run any blocks in the group.

Exercise 3. Creating the Ball

1. Click the **Paint new sprite** button. 
2. Use the drawing tools to create a small ball in your choice of colour. Make sure the costume centre is set in the middle of the ball. 
3. Change the name of the sprite to *Ball*. It should now appear in your **Sprite list** next to the *Bat* sprite.

4. Make sure the *Ball* sprite is still selected and click on the **Control** category button 

5. Drag the **When**  **clicked** block in to the **Scripts** area.
6. From the Motion category, add the **go to x: y:** block to the bottom of the  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 and then clicking the  button.

7. Make sure you are still in the **Motion** category and add the **move 10 steps** block to the bottom of the existing blocks.
8. 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.

9. 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.
10. 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.
11. 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.
12. 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.

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

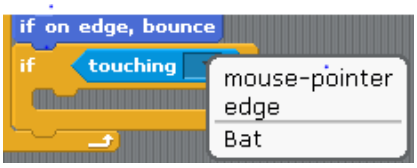
Currently the ball is ignoring the bat since we haven't yet told it what to do when it touches the bat. That's our next job.

We will add a block which contains a 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.

15. Under the Control category, find the **if** block (the one listed below **forever if**). Add this if 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.

16. Select the **Sensing** category and locate the **touching** block. It should be at the top.
17. 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.

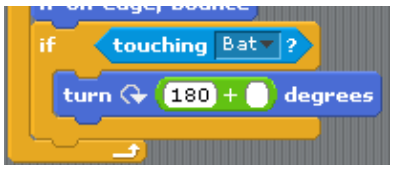


Now we can add blocks to specify what will happen when the ball touches the bat.

18. From the **Motion** category find **the turn 15 degrees** block and drag it to the middle of the **if** block. Change the number to *180*.
19. Test the program and the ball will now bounce back when it touches the bat.

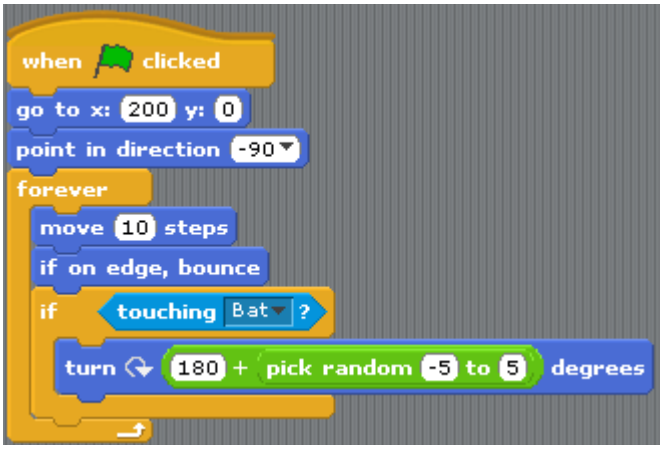
Going back in exactly the same direction is a little boring though so well make it change direction slightly each time. We can do that by making the direction of the turn slightly different each time with a random number block.

20. The **Operators** category is where we find all our number related blocks. Select the first 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 to the *180* will be randomly generated.

21. Locate the **pick random 1 to 10** block and drag it to the second space. Change the numbers to -5 and 5. The whole block now should be adding any random amount from -5 to 5 to the 180° turn (note: sometimes it can be difficult to drag a longer block like the random block in to a small number space but keep trying and you'll get the hang of where to position it).
22. Test your program. Now each time the ball touches the bat it will bounce back on a slightly different angle.



23. Save your completed program as *Bat and Ball* and then try some of the extra things on the following page.

Exercise 4. 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 

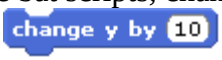
block right under the **When**  **clicked** block. Make sure the angle is set to -90 and not just 90 if you want it moving to the left.

- **Change the stage background**. Vanilla a bit boring? Click on the stage in the Sprites list. From the **Scripts area** click on the Backgrounds tab. Click the Import button and then choose a background picture. Scratch comes with many interesting backgrounds installed and you will find them under the Scratch program folder in backgrounds (normally the default location when you choose the import option). 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 import 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.

Exercise 5. 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.

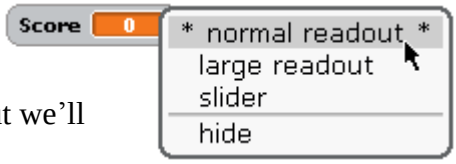
1. Select the **Stage**.
2. Select the **Variables** category 

3. Click the **Make a Variable** button. 

4. For the variable name enter **Score**.

A score block will appear on your stage. 

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



5. Select the **Ball** sprite from the sprite list at the bottom.

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.

6. In the **Variables** category, find the **change score by 1** block and add it to the **if touching bat** section you already have in your script area.

7. 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.
8. Add another **If** block underneath the current one. We will add a condition that checks to see if the x position of the ball has gone past a certain point (past where the bat is).

9. From the **Operators** category, select a **less than** block  and add it to the blank condition section of the **If** block.
10. From the **Sensing** category select a **x position of ball** block 

and add it to the first part of the **less than** block.
11. In the second part of the **less than** block, enter the number -220 (with the minus sign).
12. From the **Variables** category, find a **set score to 0 block** and place it inside the **If** block. The completed section should look like the example to the right.

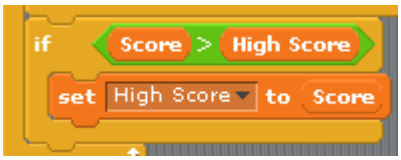
13. We will add one last block that sets the score to 0 when the green flag button is clicked. Otherwise every time you start, the score will continue from where it was before.

14. Add another **set score to 0 block** just under the **When** clicked block.
15. 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, even after the button has been clicked.

Exercise 6. Adding a High Score

No point in getting a great score if it's going to vanish the moment you miss. A high score will keep track of your best result.

1. Select the **Stage** in the sprite area.
2. Click the Make a variable button.
3. Enter **High Score** for the variable name.
4. Move it to a convenient place on your stage so it isn't overlapping the **Score** variable.
5. Select the **Ball** sprite.
6. 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.