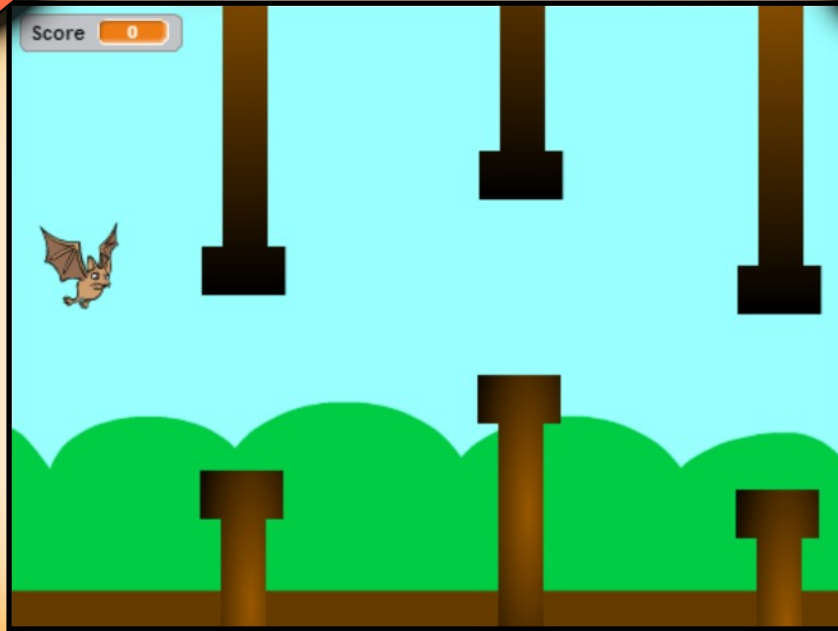


# Flappy Bat



```
when I start as a clone
  show
  go to x: 240 y: 0
  repeat until x position < -235
    change x by -3
  change Score by 1
  delete this clone
```

```
when clicked
  forever
    if key up arrow pressed?
      then
        point in direction 0
        move 5 steps
```

```
when clicked
  forever
    if timer < 5
      then
        set color effect to 20
    else
      set color effect to 0
```

Scratch 2.0



Programming for kids by





## Contents

<b>The Basic Game</b>	<b>Page</b>
All About Scratch, Using Scratch	2
Flappy Bat - Getting Started	6
Getting Flappy Flapping	11
Making Flappy Fly and Fall	13
Moving Columns and Scoring	17
Game Over	26
Final Testing	30
<b>Extension Tasks (Making the Game Better!)</b>	
Introduction (Extension Tasks)	31
Percy, the Parrot of Doom	31
Speeding Things Up	33
I'm Invincible	35
<b>Solutions</b>	
Solutions 1 to 8	38
<b>Glossary</b>	
Definitions of Programming Terms Used	51



## All About Scratch

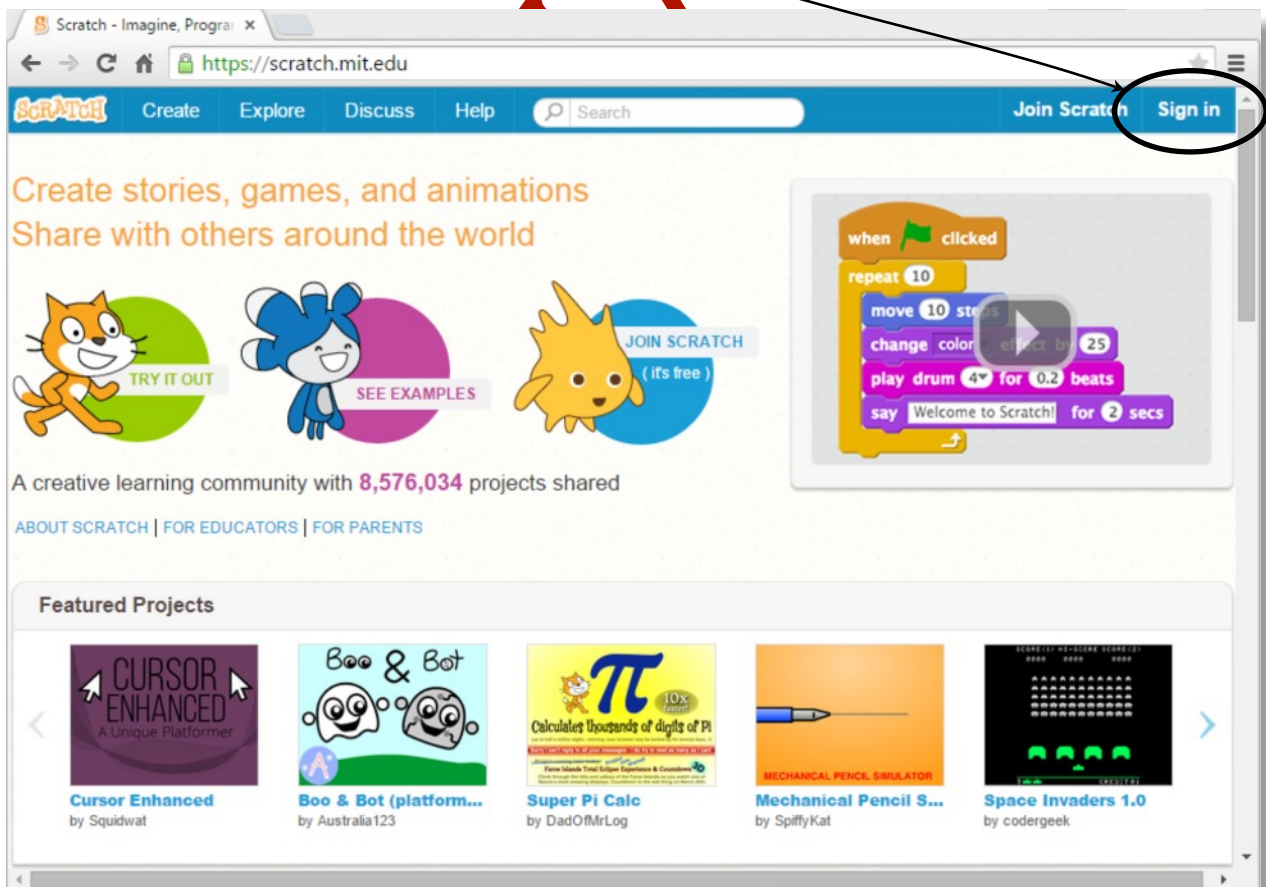
Scratch is a '[programming environment](#)' used to teach school students how to write computer [programs](#). It is simple to use, great fun and easy to learn.

## Using Scratch

Scratch can be used in two ways. You can either install it on your computer or you can use it on the Scratch website.

Before you can write a program on the website version you must first log into the website ([scratch.mit.edu](https://scratch.mit.edu)).

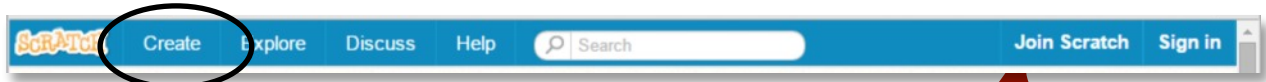
Your teacher will ask you to create a new account, or give you a username and password to sign in, if you are using the website.



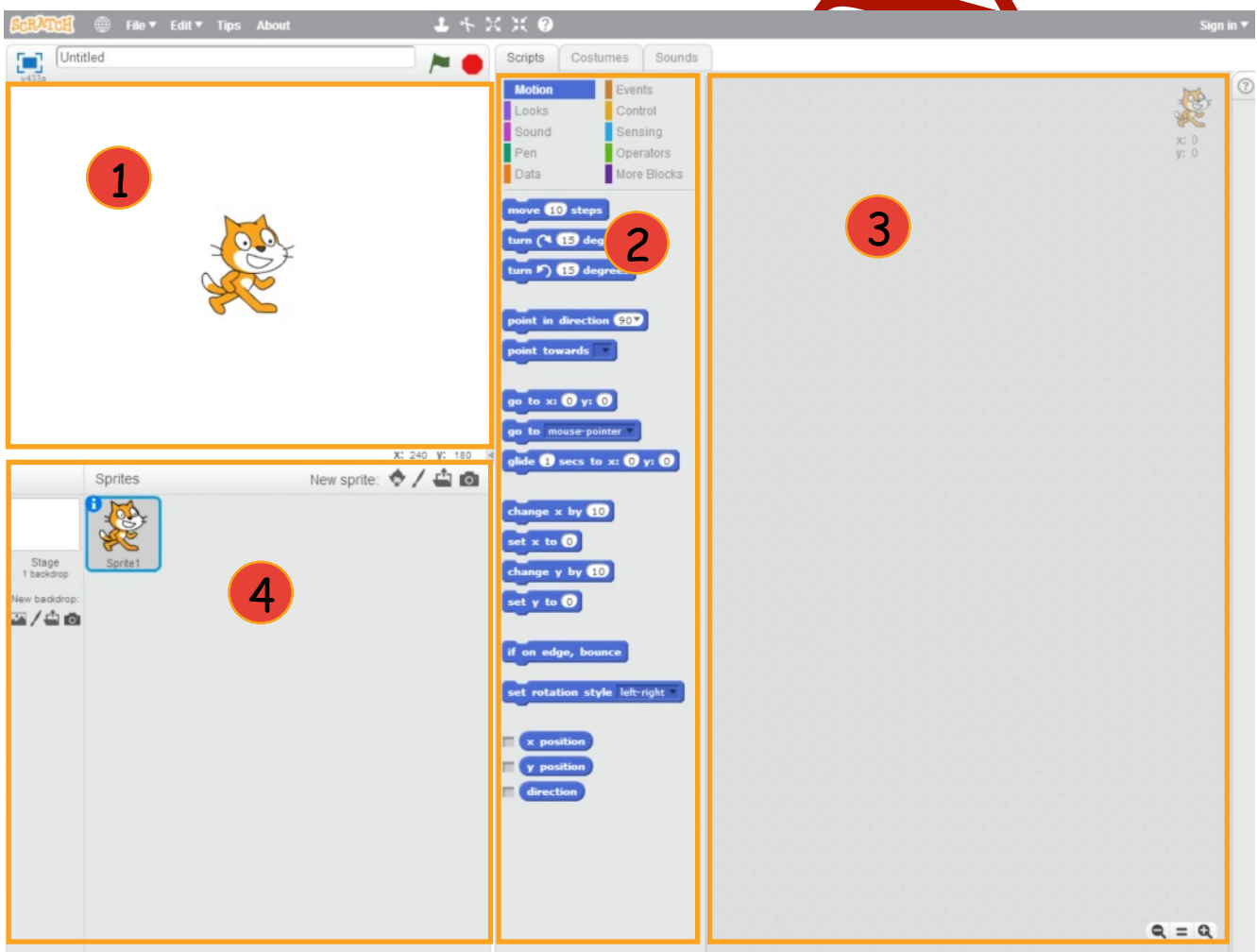


Let's look at the Scratch environment (working area).

Click on the 'Create' link to start a new project.



The Scratch environment has four different windows. These have been numbered below.

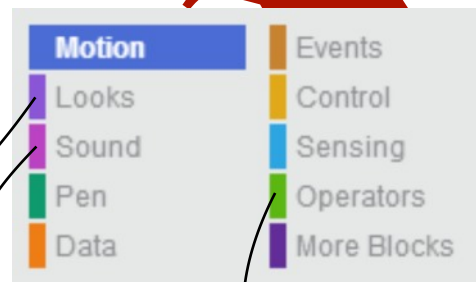


- 1 **Stage** - The place we put our characters and objects.
- 2 **Blocks Palette** - A menu of blocks used to write program [code](#).
- 3 **Scripts Area** - Where our [program](#) code goes.
- 4 **Sprite List** - A list of all the [sprites](#) in our game.



A computer [program](#) is a group of instructions that tell a computer how to carry out a task. Scratch programs are created by dragging blocks from the 'Blocks Palette' into the 'Scripts Area' on the right. The blocks fit together, like Lego bricks, to build a computer program.

There are 10 different types of block. Try clicking your mouse on each type, one after the other and you will see the list of blocks underneath changes.



### Looks

- say Hello! for 2 secs
- say Hello!
- think Hmm... for 2 secs
- think Hmm...
- show
- hide
- switch costume to costume2
- next costume
- switch backdrop to backdrop1
- change color effect by 25
- set color effect to 0
- clear graphic effects
- change size by 10
- set size to 100 %
- go to front
- go back 1 layers

### Sound

- play sound meow
- play sound meow until done
- stop all sounds
- play drum 1 for 0.25 beats
- rest for 0.25 beats
- play note 60 for 0.5 beats
- set instrument to 1
- change volume by -10
- set volume to 100 %
- volume
- change tempo by 20
- set tempo to 60 bpm
- tempo

### Operators

- +
- 
- \*
- /
- pick random 1 to 10
- <
- =
- >
- and
- or
- not
- join hello world
- letter 1 of world
- length of world
- mod
- round
- sqrt of 9



Click on **Control** and drag the block shown below into the Scripts Area.

Each block in Scratch is used for a different reason. This block makes part of your program repeat other blocks.

Some blocks have numbers that can be changed. Change the number by clicking here and typing in 500.



To become a good programmer you have to practise solving harder and harder problems. When you make the Flappy Bat game you will be asked to complete lots of challenges.

Time for your first challenge.

VIEW

## Challenge 1

Find the correct blocks and create this program [code](#).

When you've finished click the green flag to [run](#) the program.



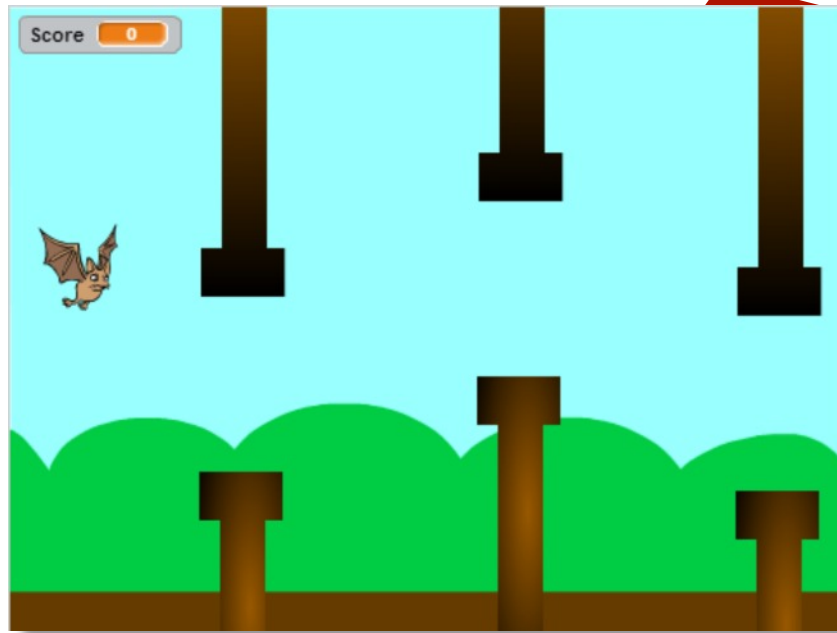
Try changing a few of the numbers (for example, change the move block to 50 steps). [Run](#) the program again and see what happens.

```
when green flag clicked
clear
set pen color to 0
go to x: 0 y: 0
pen down
repeat 240
  repeat 6
    move 80 steps
    turn 60 degrees
  turn 187 degrees
  change pen color by 5
```



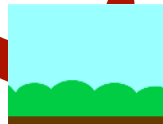
## Flappy Bat - Getting Started

The game you are going [to code](#) looks like this. Columns will move from right to left across the screen. The player moves a character called Flappy Bat up and down, trying to fly through the gaps in the moving columns. Each column the player passes adds 1 to the score.

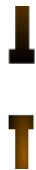


To complete the basic game you will have to learn how to:

- add a backdrop picture



- add each of the [sprites](#) (characters and objects)



- code the sprites to move



- know when one sprite is touching another



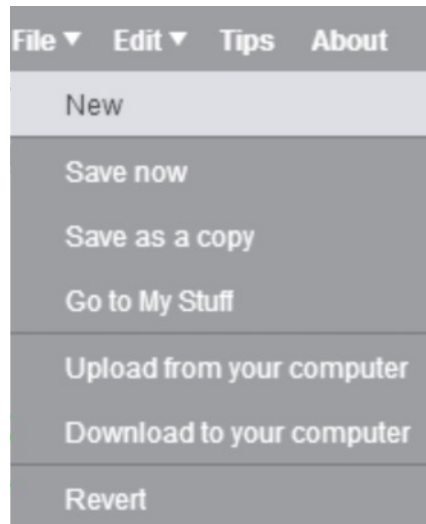
- keep a score



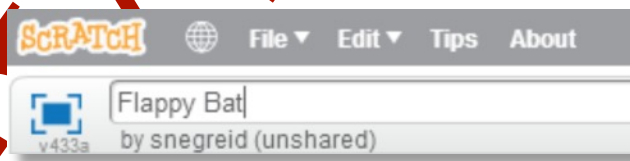


Before you can start any new Scratch program and start coding you must create a new project and give it a name.

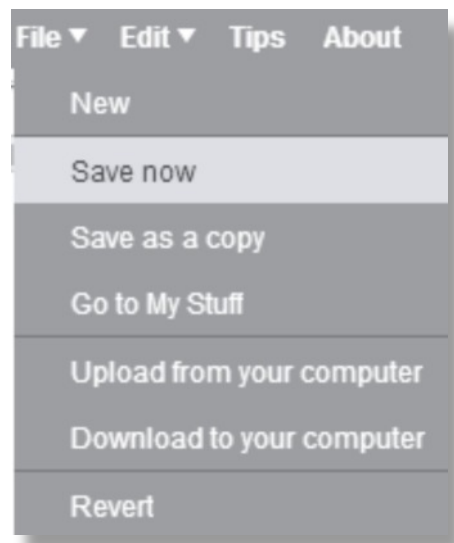
Click on the 'File' menu and select 'New'.



Give your project a name.



Now use the 'File' menu again to save your new project.

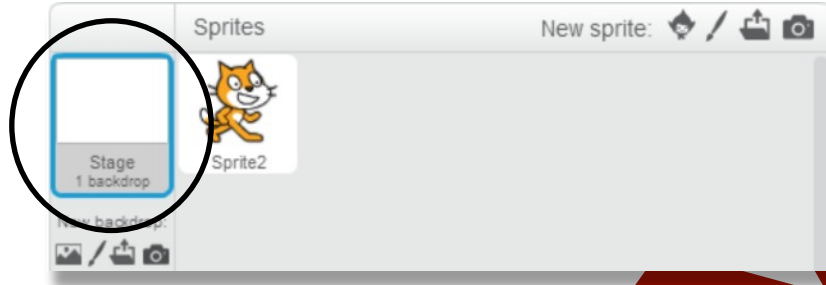




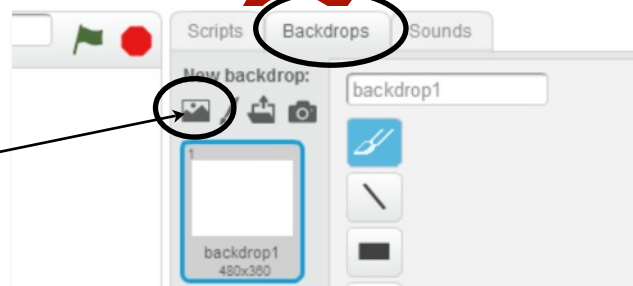


## Changing the Backdrop Picture

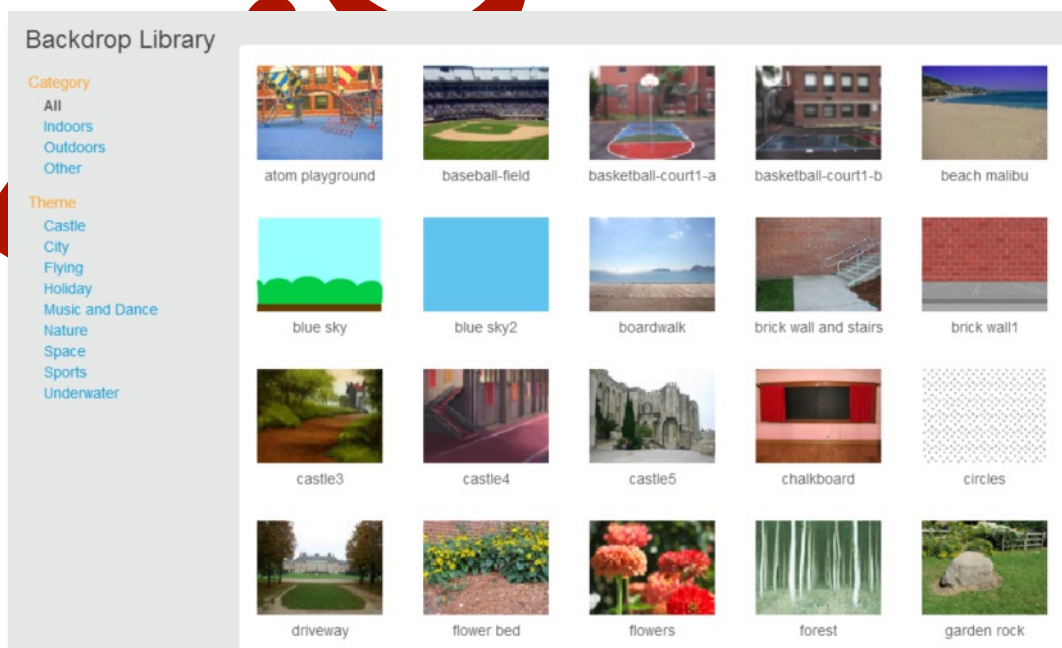
To change the backdrop picture click on the stage.



Now choose Backdrops from the top of the window and click here.



This opens the 'Backdrop Library' which has lots of different backdrops. Choose a backdrop you like.





Every Scratch project starts with the cat [sprite](#) in the middle of the stage.

You must delete the cat before you add our own bat sprite.

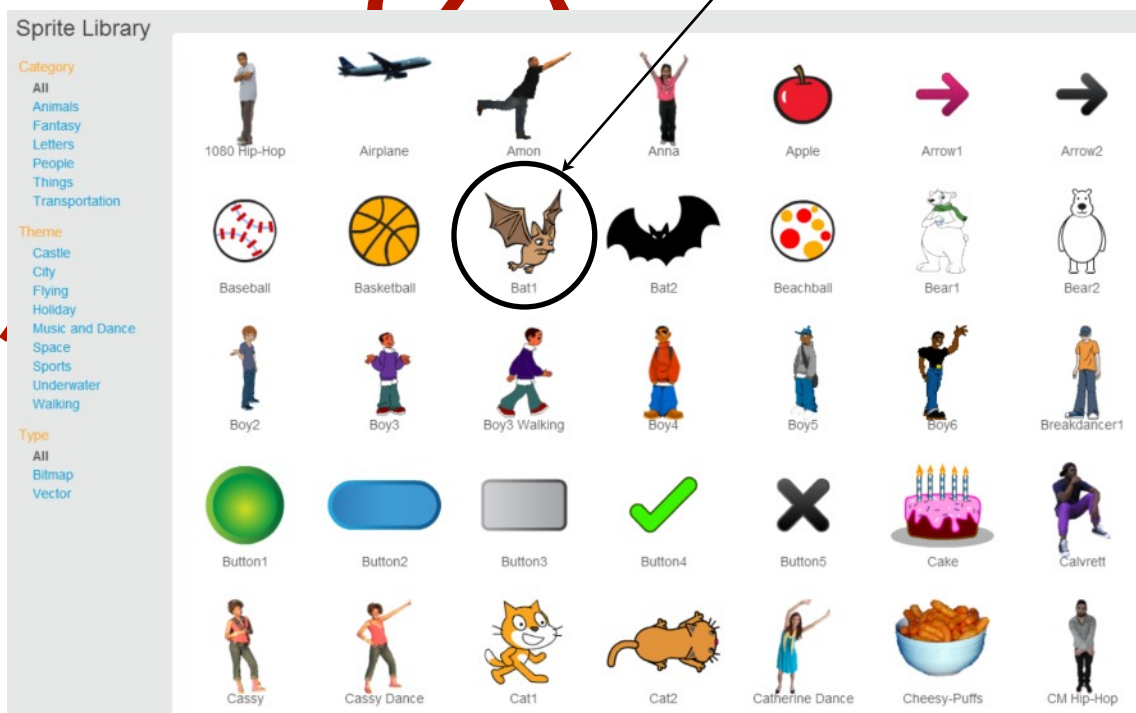
Right click on the cat and select 'delete'.



To add a new sprite (Flappy Bat) click on the option shown below.

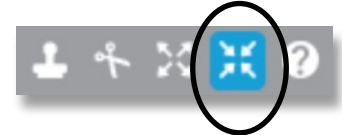


This opens the Scratch 'Sprite Library'. Double click on the bat.





To make Flappy Bat smaller, choose the shrink icon at the top of the window. Click several times on Flappy Bat. We need him to be a lot smaller.



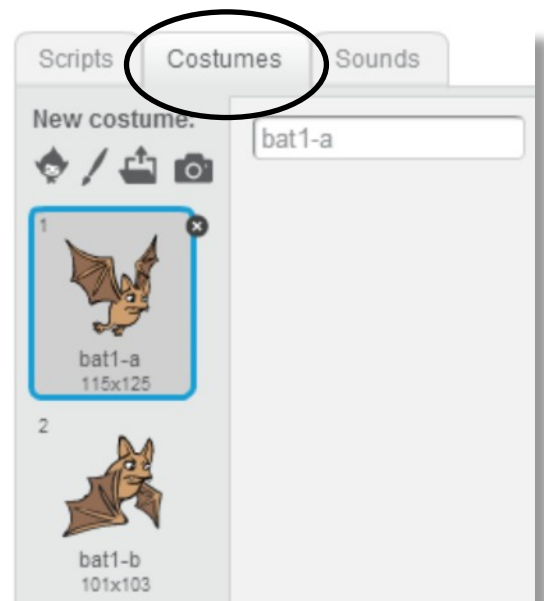
Flappy should look about this size on the stage.



One sprite can have several different looks, called costumes.

Flappy has two costumes. You can use the 'Costumes' tab to see them.

The first bit of coding you are going to do is to make Flappy change from one costume to the other and then back again.

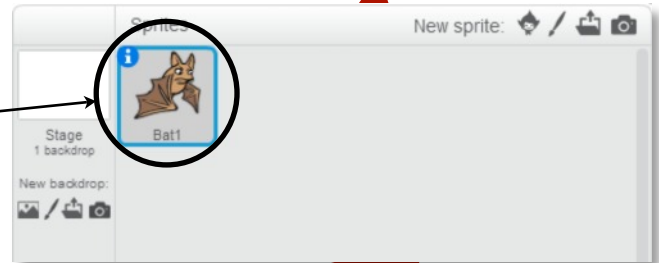




## Getting Flappy Flapping

While the game is running Flappy should keep changing from one costume to the next. This will make him look like he's flapping his wings.

Make sure Flappy is highlighted in the sprite list. (click on him if he isn't)



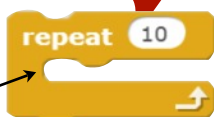
Find and add the blocks shown to the scripts area.



 Run the program a few times by clicking the green flag.

Each time you click the green flag Flappy will change to the next costume. This works but what we really want is Flappy to keep changing costume over and over without having to click the green flag all the time.

To make something happen more than once in a computer program we use a loop. There are three types of loop used in Scratch.



Loop a set number of times.



Loop all the time until the program stops running.

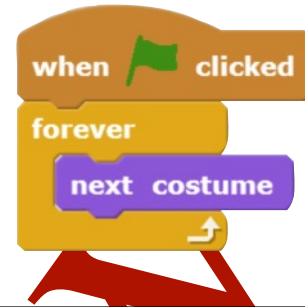


Loop until something happens in the program to stop the loop.



You will use the forever loop as you want Flappy to flap constantly for the whole game.

Add a 'forever' loop to Flappy's script and place the 'next costume' block inside the loop.



Try running the program and you'll see that Flappy bat is flapping too fast.

## Challenge 2

Look through the block palette and find a block that will make your program pause for a period of time.



Use the block to slow down Flappy's flapping. You'll have to work out where to add the block to the program.

The number you type into the new block is in seconds.

- 1 = one second
- 2 = two seconds
- 0.5 = half a second



Try several different numbers until Flappy looks like he's flapping his wings realistically.





## Making Flappy Fly and Fall

When you are coding it is a good idea to think about a problem carefully before you start creating the program code.

To make Flappy fly and fall we have two problems to solve .



 1. Going Up  
 Flappy should move up the stage when a key is pressed on the keyboard.

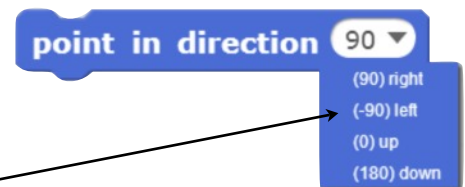


 2. Falling Down  
 Flappy should fall down the stage when a key isn't being pressed.

Good programmers can take large problems and break them down into smaller ones. This is called refining (or refinement). Let's look at different parts of the two problems and how to solve them.

### How do I make Flappy move up or down?

This can be done using the 'point in direction' block.

This block allows you to point your sprite up, down, left or right.



If you then add a move block your sprite will then move in that direction.





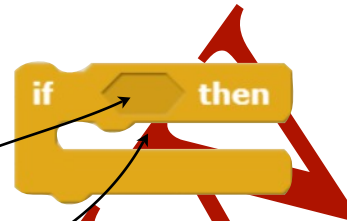
## How do I decide if something happens or not?

Decisions can be made (like "is a key being pressed?") using an 'if' block.

If blocks decide whether something is true or not. For example:



These decisions would be added here.



The blocks that are placed inside the if block only run if the decision being checked is true.

## How do know if someone has pressed a key?

The 'key pressed' block can be used tell if a key has been pressed on the keyboard.



The key can be selected here.

## Now put it all together

So, if you want to check if the up arrow has been pressed and then move Flappy in that direction you can put all these blocks together.



If the up arrow is being pressed...  
...point Flappy up...  
...and move 5 steps.



Add these blocks to Flappy Bat's scripts.

```

when clicked
  if key up arrow pressed? then
    point in direction 0
    move 5 steps
  
```



Test the program by running the program using the green flag. Press the up arrow on the keyboard to see if Flappy moves up the screen.

## Why doesn't it work!

You'll notice from your test that the new blocks don't work as you would expect them to. This is because when the program runs it only checks once to see if the key is pressed.

If you want Flappy to move upwards *every time* you press the up arrow key the program has to *check over and over again* to see if the key is being pressed.

Remember if you want something to happen more than once you need to use one of the three 'loop' blocks.

Put the blocks you already have inside a 'forever' loop.

```

when clicked
  forever
    if key up arrow pressed? then
      point in direction 0
      move 5 steps
    
```



Test your program again. It should work now.






## Falling Down

Making Flappy move down is a lot easier. If no key is being pressed Flappy should constantly fall down the screen.

Add another 'point in direction' (set to point down) and move blocks underneath the 'if' block.

```

when green flag clicked
  forever loop
    if key up arrow pressed? then
      point in direction 0
      move 5 steps
    point in direction 180
    move 5 steps
  
```

 Testing the program shows that Flappy stops falling when the up arrow is pressed but he doesn't actually go up.

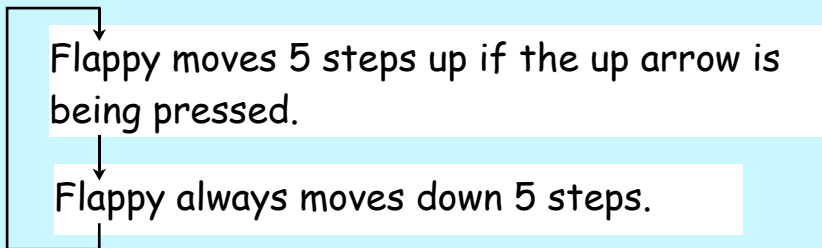
## Challenge 3

If you think carefully about what our program is doing, you can work out why Flappy doesn't go up when you press the up arrow key.

```

when green flag clicked
  forever loop
    if key up arrow pressed? then
      point in direction 0
      move 5 steps
    point in direction 180
    move 5 steps
  
```

Each time the forever loop runs.



When the up arrow is being pressed, Flappy moves 5 up and then 5 down. The two moves cancel each other out and Flappy ends up in the same place. So how would you make Flappy actually move up?

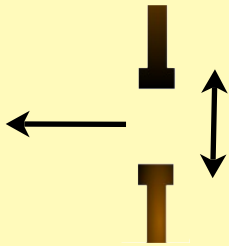
Your challenge is to change your program to make it work.

Clue - You don't need any new blocks.



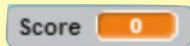
## Moving Columns and Scoring

The next part of Flappy Bat to code are the moving columns. Here are the main problems to solve.



### 1. Moving Columns

Columns will appear at the right hand edge. The columns should appear at different heights on the stage. The columns will all move at the same speed towards the left hand edge of the stage. When a column reaches the left hand edge it should disappear.



### 2. Scoring

When a column reaches the left hand edge one point should be added to the score.

## Refining the Problem

Let's refine this by taking your large problem and breaking it up into lots of smaller problems. This should make the original problem easier to understand. This is a large problem so you end up with a long list of smaller problems, that you can now solve one at a time.

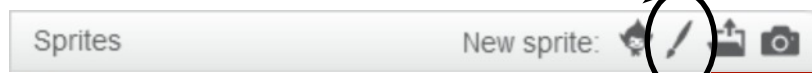
- create a column sprite
- make the column sprite appear on the right of the stage
- make the column move
- make the column disappear
- add 1 to a score
- make more than one column appear at different heights



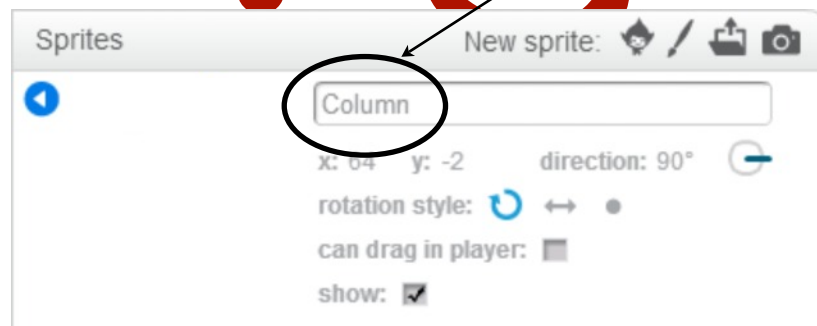
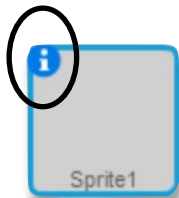
## Creating a Column Sprite

To create a column you will have to draw one.

Click on the *paint new sprite* icon.



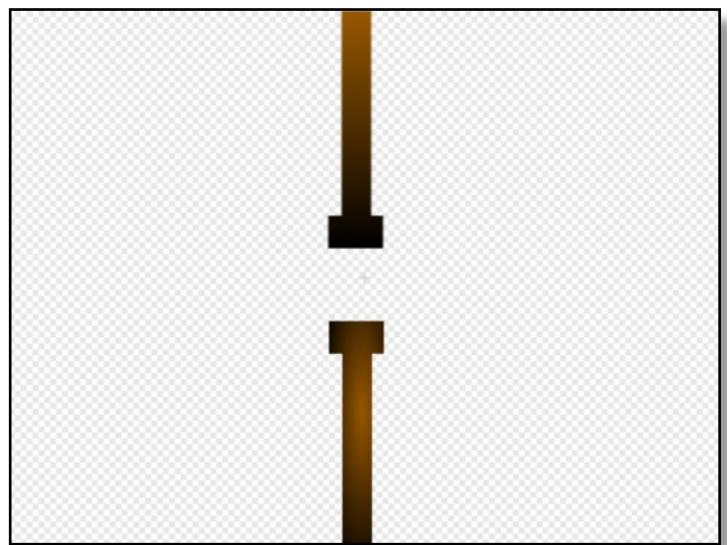
You should always make sure that you name your sprites. Click the small 'i' and change the name of the sprite to "Column".



Use the painting tools to draw your own columns.

The columns should:

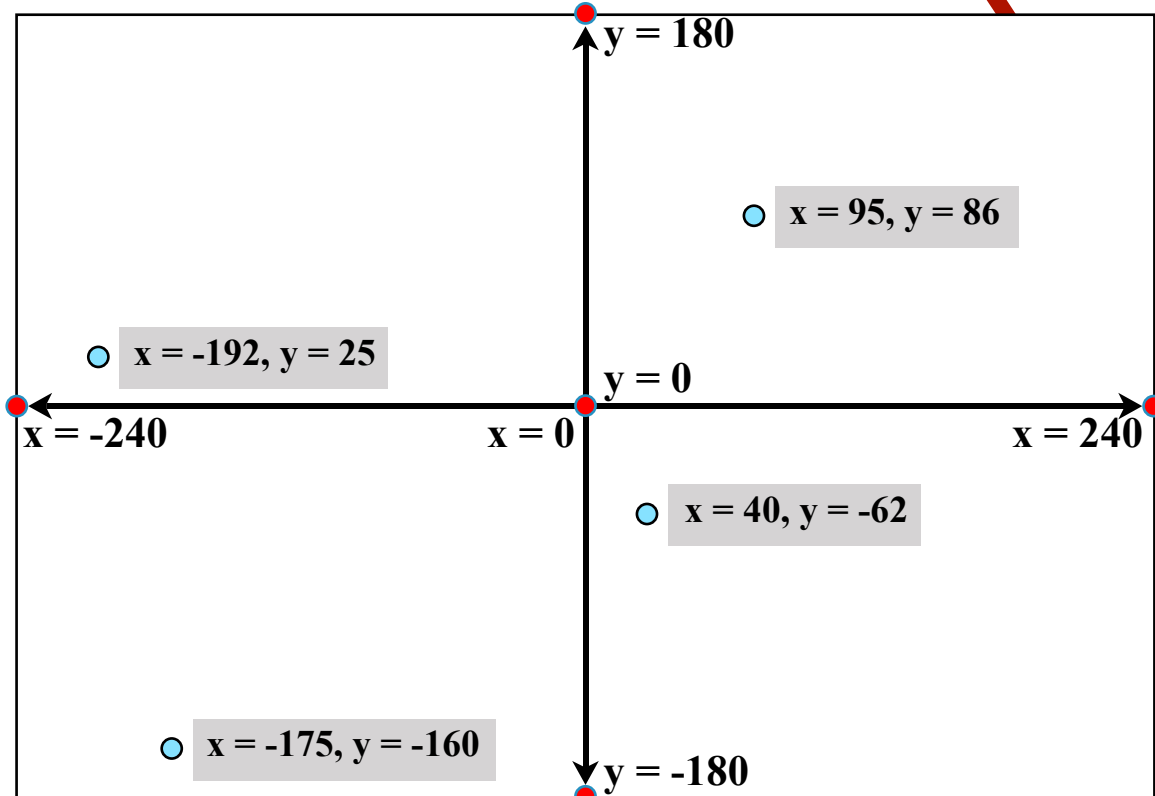
- be fairly thin
- touch the top and bottom
- have a good gap in the middle
- be a colour of your choice





## Make the Column Sprite Appear on the Right of the Stage

Every spite in a Scratch game has x and y coordinate that gives its position on the stage. All coordinates start at the middle of the stage where x and y are both 0.



The **y coordinate** is used to give a position **up and down** the stage (from -180 to 180)

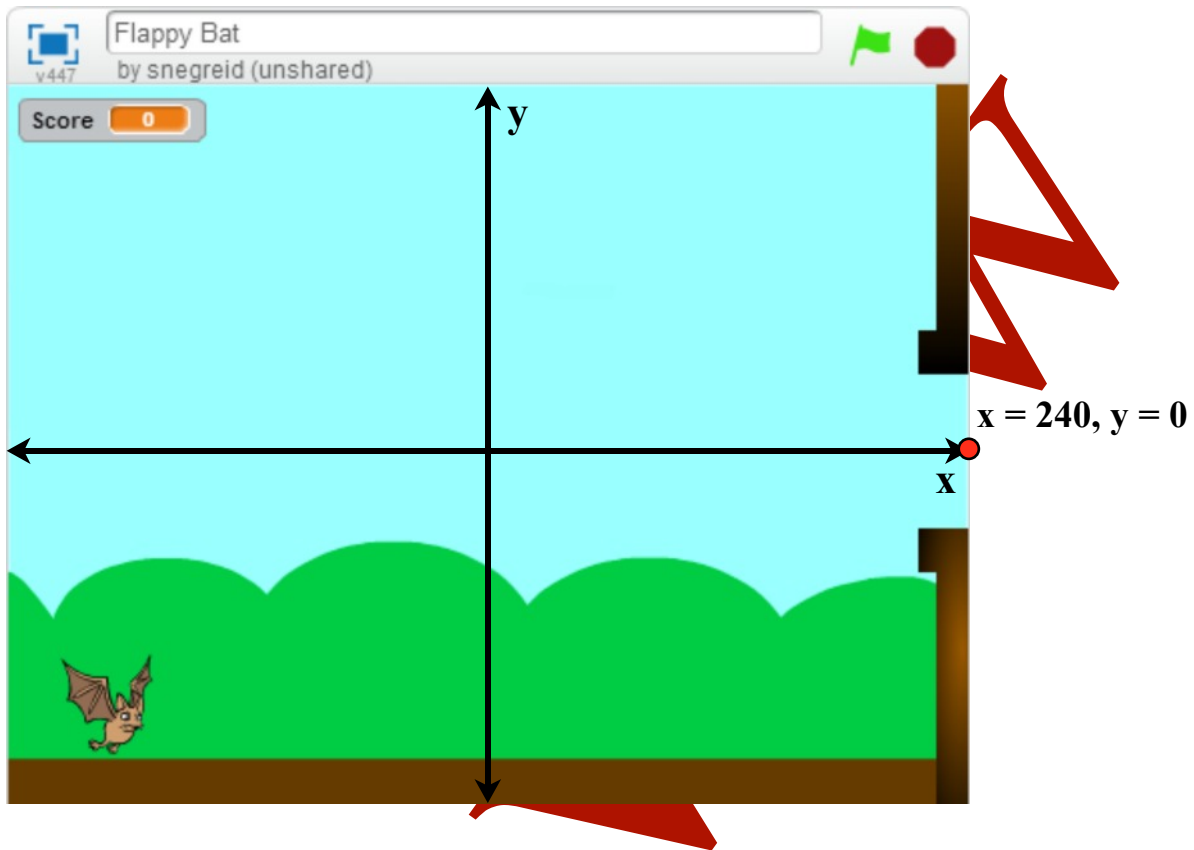
The **x coordinate** is used to give a position **left and right** on the stage (from -240 to 240)

The centre of the stage would be the coordinate  $x=0, y=0$ .

The four example dots  $\bullet$  show how the x and y coordinates are either greater than 0 or less than 0 depending which quarter of the stage they are found in.



To make the column appear on the right hand edge of the stage you need to move it to the correct coordinate ( $x = 240, y = 0$ ).



Sprites can be moved to a coordinate with a 'go to' block.

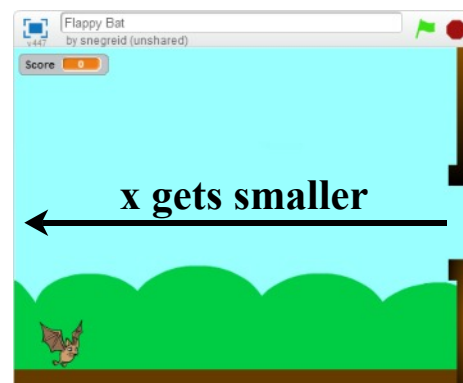


## Make the Column Move

To make the column move to the left you can change the x coordinate of the column sprite like this.



Changing the x coordinate by -3 will move the sprite to the left as the x coordinate will get smaller.





You must put the 'change x by' block inside a loop to make the sprite move -3 more than once.

This time you should use a 'repeat until' loop. This won't keep going forever but will stop when the column is near the left hand edge of the stage (when the x coordinate is less than -235).

```
repeat until x position < -235
  change x by -3
```

## Make the Column Disappear

To make sure the column sprite appears at the right and disappears when it reaches the left you can use 'show' and 'hide' blocks.

```
show
hide
```

You can put all of this together to build the code for one column.

Build this code for your own column.

```
when green flag clicked
  show
  go to x: 240 y: 0
  repeat until x position < -235
    change x by -3
  hide
```



Test your program. You should have one moving column.

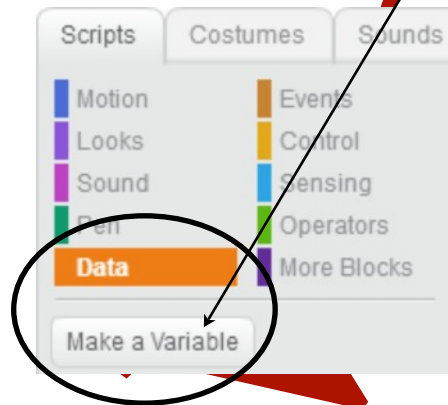


## Add 1 to the Score

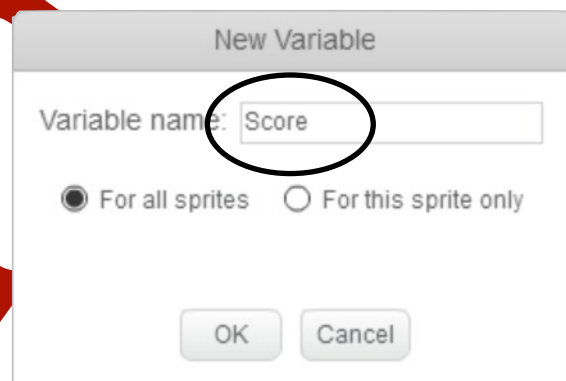
As we play Flappy Bat the score should start at 0 and then go up by 1 each time a column reaches the left hand edge of the stage.

In programming, when we want to store a number that changes like this we use [variables](#).

To create a 'Score' [variable](#) click on the button shown below.

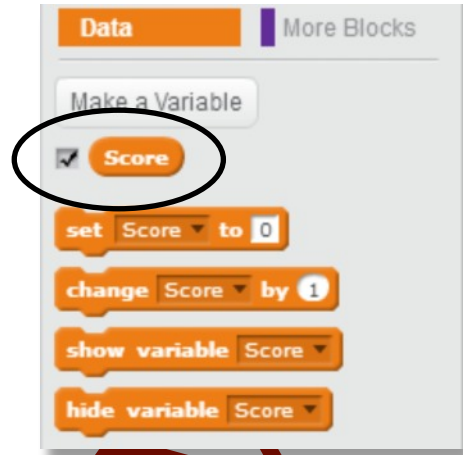


Type in the Variable name 'Score' and click the OK button.





To make the Score variable appear on the stage you must make sure that this has been ticked.

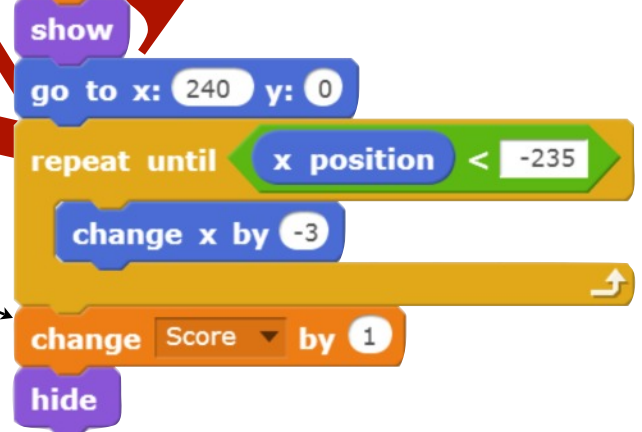


Now add the two variable blocks to your column code.

The Score variable should be set to 0 when the game starts.



1 should be added to the Score variable after the column reaches the left hand edge of the stage.



Test the program again.

Make sure 1 is added to the Score variable when the column reaches the left hand edge of the stage.





## Make More Than One Column Appear at Different Heights

To make more columns appear on the stage you need to create copies of the column. In Scratch, copies are called clones.

To create a copy (clone) of the column sprite you can use this block.

```
create clone of myself
```

If we wanted to keep creating new copies every few seconds we could use a 'forever' loop with a 'wait' block to control how often the columns appear.

Add this new code to the column sprite.

```
when clicked
  forever loop
    wait 2 secs
    create clone of myself
```

clones will be made every 2 seconds

Each new clone that is created must appear, move and disappear just the original column.

Also add this new code to the column sprite.

After the clone reaches the left hand edge you can add 1 to the score and then delete it.

```
when I start as a clone
  show
  go to x: 240 y: 0
  repeat until x position < -235
    change x by -3
  change Score by 1
  delete this clone
```



Test the program.  
Make sure the program is now creating lots of columns.

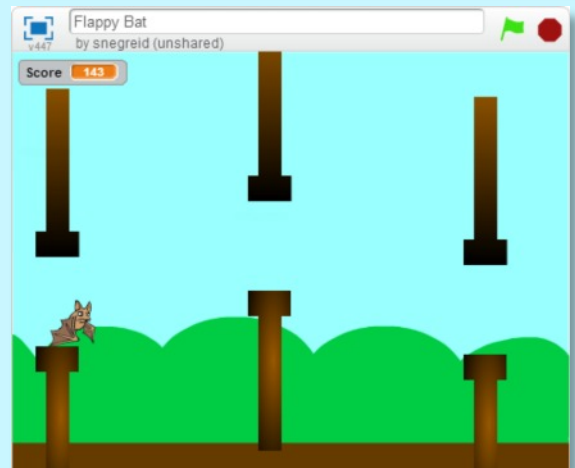
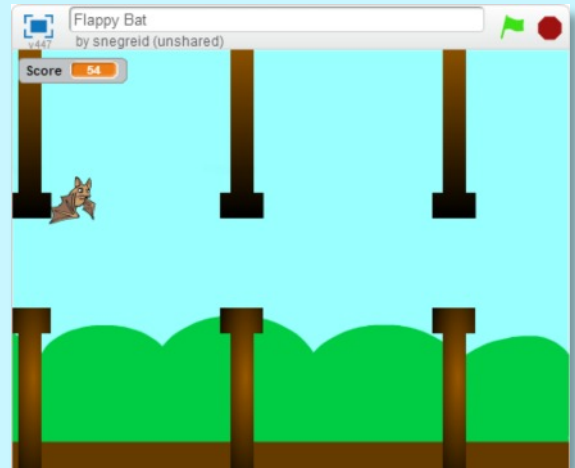


## Challenge 4

When you tested the game you will have seen that the columns all appear at the same height. This would be a very boring game to play.

Flappy Bat will be a much better game if the columns appears at different y coordinates.

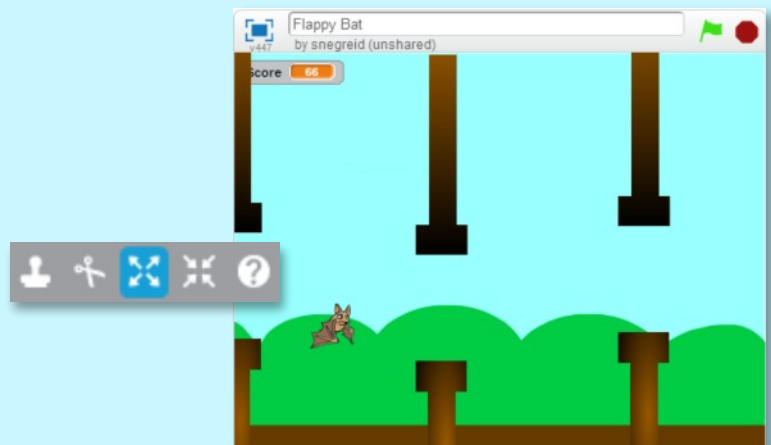
Your challenge is to replace the 0 in the y coordinate of the two 'go to' blocks with a new block that will make the y coordinate change each time a column appears.



```
go to x: 240 y: 0
```

You will have to find the correct block, change the numbers in the block and then test it.

When you get this challenge working, use the grow icon to make the columns taller than the stage. This will stop gaps appearing at the top and bottom when the columns appear.

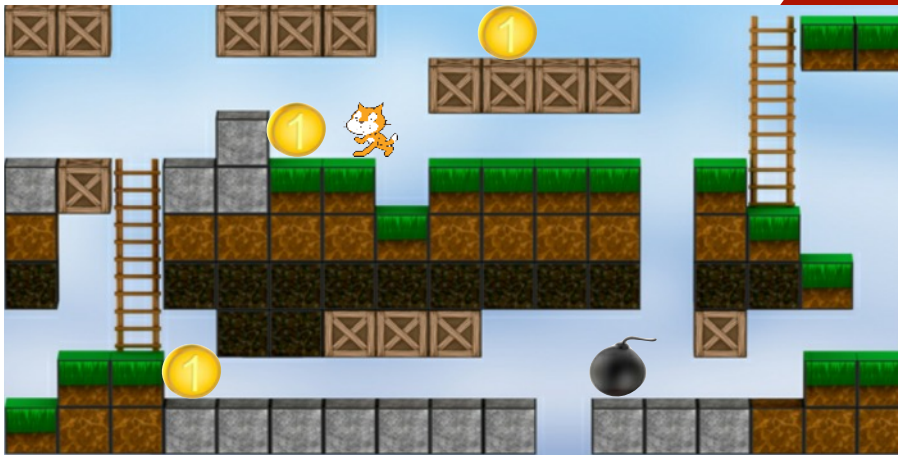




## Game Over

If you think of a few different computer games you'll know that sprites have to be able to react to other sprites:

- falling sprites may stop when they hit another sprite
- character sprites might move up a ladder sprite
- bomb sprites may destroy other sprites
- character sprites could pick up other sprites like coins



The final part of Flappy Bat is to make sure the game finishes when Flappy hits one of the moving columns.

Here are the two problems to solve.



### 1. Touching a Column

When Flappy is touching one of the columns the game should stop.

**GAME OVER**

### 2. Game Over

When the game stops a message will appear saying "Game Over".



## Touching a Column

Flappy Bat must be able to react to the bat sprite touching the column sprite.

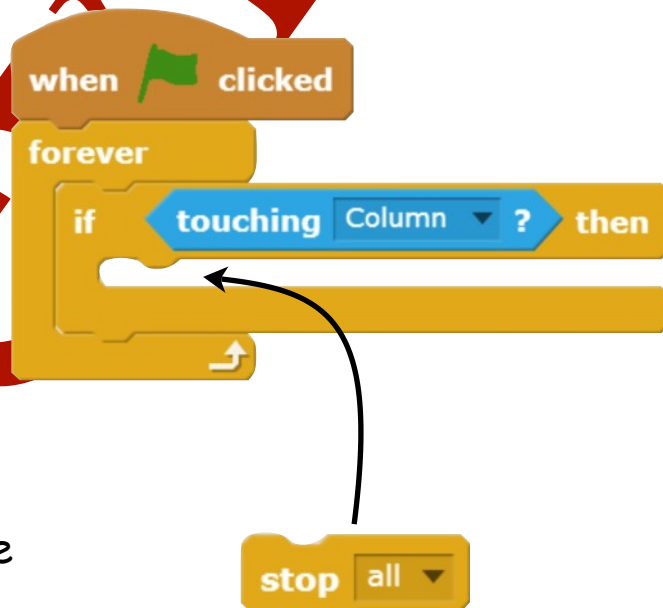
This can be done with an 'if' block and the 'touching' block.



What happens when the Column is touched will be added here.

If these blocks are added to Flappy's scripts, the code would know when Flappy was touching a column. A forever loop will keep the program checking if Flappy is touching a column.

Add this code to Flappy Bat's scripts.



To stop the game running you can use the 'stop' block. Place this inside the 'if' block.



Test the program.

Make sure the game stops when Flappy touches one of the columns.



## Game Over Message

To show that the game is over you can create and draw another sprite that will appear when the game finishes.

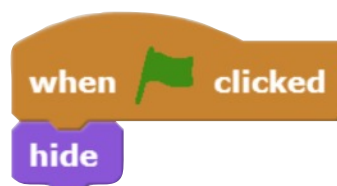


Create a new sprite.  
Change the name of the sprite to Game Over.

Use the 'Text' tool to add the words "Game Over" to the costume.



Add the code below to make sure the Game Over sprite is hidden when the game starts.





## Sending Messages

Sometimes in computer games what happens to one sprite will have an effect on another sprite. In our game, when Flappy touches a Column sprite the Game Over sprite must appear.

You can't just add a 'show' block to Flappy's code as this would make Flappy appear and not the Game Over sprite.

```

when clicked
  forever loop
    if touching Column ?
      show
      stop all
  
```

Wrong!

Instead, Flappy's code must send a message to the Game Over sprite telling it when to appear.

This can be done using the 'broadcast' block.

```

when clicked
  forever loop
    if touching Column ?
      broadcast message1
      stop all
  
```



A larger game may have lots of message so it's important that each message makes sense.

Add the 'broadcast' block.

Change the message to what you want to happen when the message is received.

New Message

Message Name:

OK Cancel



The Game Over sprite can now receive this message and use it to show the sprite.

Add these blocks to the Game Over sprite.



Test the program.

Make sure the Game Over sprite is hidden when the game starts and only appears when the game finishes.

## Final Testing

Up until now you've been testing the program, after solving each problem, to make sure new code works correctly.

When a program is finished a different type of testing is carried out to see if the working program does what it's supposed to do. For Flappy Bat that means making the game enjoyable to play.

## Challenge 5

If the game is too easy or too hard you will need to find ways of fixing the game by editing the sprites. This could mean:

- changing how the sprites move (Flappy could move up faster and down slower)
- changing the speed sprites move at (the columns could move faster or slower)
- or even redrawing the sprite's costume (the gap in the columns could be made larger or smaller)

Your final challenge, to complete the Basic Game, is to make the game playable. You decide how hard you want Flappy Bat to be.



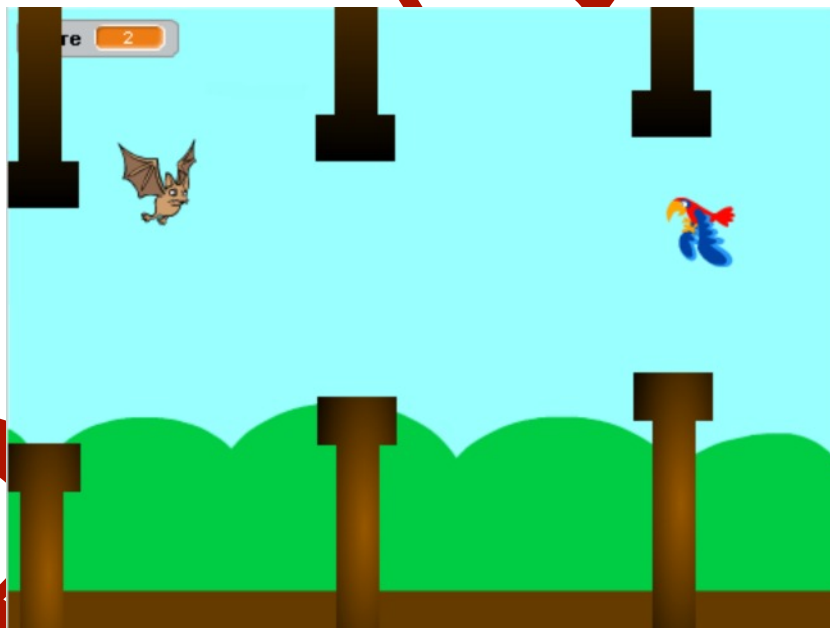
## Introduction (Extension Tasks)

To become a good programmer you must to learn to solve problems on your own. Each of the extension tasks makes our game even more fun by adding new features to the Basic Game.

Read each task carefully and then try to build the code to solve the problems.

## Percy, the Parrot of Doom

Percy the Parrot doesn't like Flappy very much and nothing makes him happier than watching Flappy fly into a column.



### Problem

Percy should appear at the right hand edge of the screen and fly in a straight line towards the left where he will disappear. If Percy hits Flappy, the game should end. Flappy should also be able to move left and right to help him avoid Percy.





To help you solve the Percy problem it has been refined into a list of smaller problems.

## Challenge 6 - Percy, the Parrot of Doom

Solve each of the refined problems below.

1. Add a new 'parrot' sprite to the program

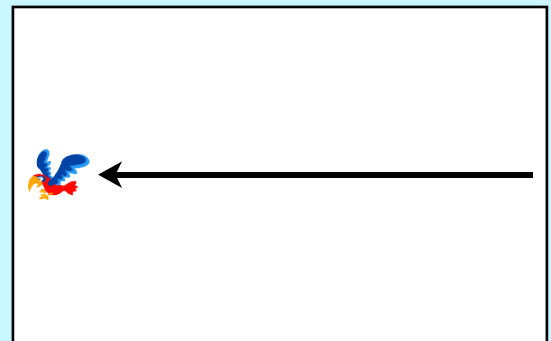


2. Percy should be hidden when the game starts and should then appear every few seconds.

3. When Percy appears he should always be at the right hand edge of the stage. He could appear anywhere between the bottom and the top of the stage.



4. Percy should fly quickly across the stage, disappearing when he reaches the left hand edge.



5. While he flies, Percy should also flap his wings (like Flappy).



6. If Flappy hits Percy or a Column the game should end.

7. To help Flappy avoid Percy the Parrot of Doom, extra code should allow him to move left and right as well.





## Speeding Things Up

Variables are used in computer programs to store values (numbers, letters and words) that will change as the program runs.

If you know that a sprite will always move 5 steps you would enter 5 into the move block in your program.

If you want the sprite to move faster or slower during a game you must use a variable instead.

These three blocks do exactly the same as `move 5 steps` because the `moveBat` variable has been set to 5. The difference is you can now change the number of steps the sprite moves later by changing the `moveBat` variable.

If you want the sprite to move more steps (faster) you can increase the number stored by the variable.

`moveBat` now equals 7

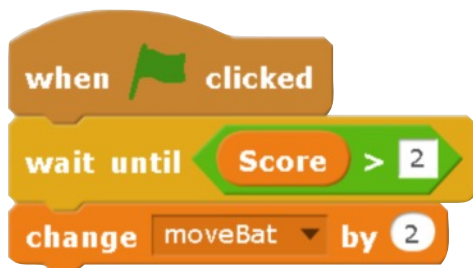
To move the sprite fewer steps (slower) you can reduce the number stored in the variable.

`moveBat` now equals 2



If you want your sprites to speed up (or slow down) at certain points in our game you must add more code to control this. There are a few ways to do this.

Here's one example. If you want to change what a variable is storing when the game reaches a certain score you could do this.



This code waits until the score is greater than 2 before it runs the next block.

The 'change' block adds 2 onto the moveBat variable (making Flappy move faster)

## Challenge 7 - Speeding Things Up

Solve each of the refined problems below.

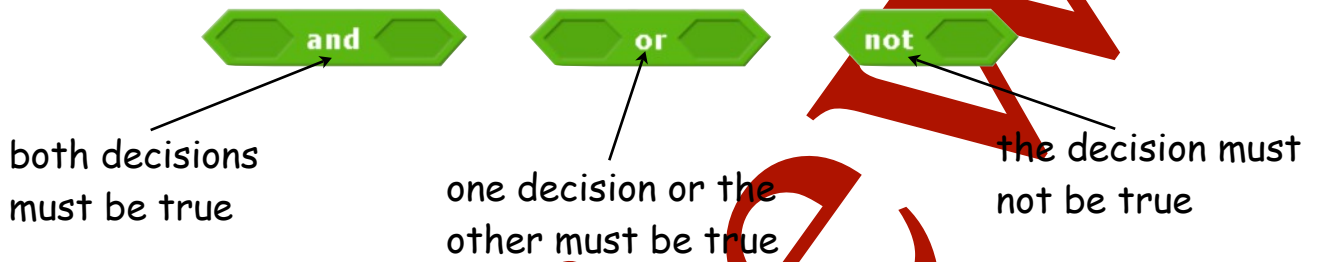
1. Using a variable, make the columns move faster when the player scores more than 5 and then faster again when the player scores more than 10.
2. Using a variable, make Flappy move faster when the player scores more than 5 and then faster again when the player scores more than 10.
3. As the columns move faster across the stage the gap between them gets bigger. When the player scores more than 5 and more than 10 change the amount of time between each column being created so that the columns appear faster (and the gap between them stays the same).



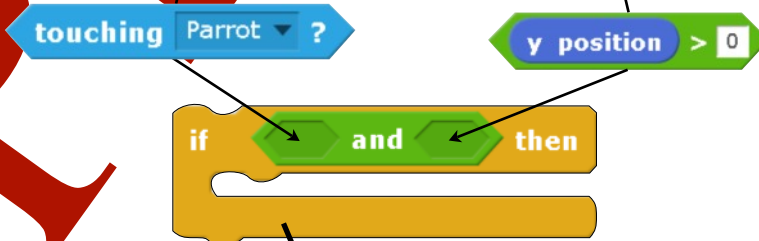
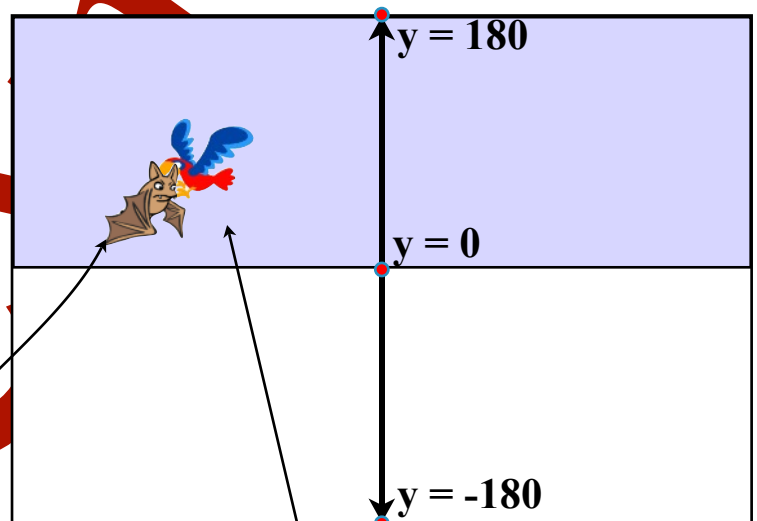
## I'm Invincible

As your code become more and more complicated you will have to make harder decisions.

If blocks can be built which combine two or more decisions using operators.



For example, if you wanted to check if one sprite was touching another but only do something if the sprite is in the top half of the stage you would need to see if both these things were true.



the finished if block would look like this





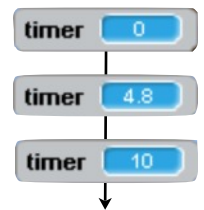
Computer games often use time to make decisions. A player may have to do a task before a clock runs out or a bonus object may appear for just a few seconds before disappearing again.

If you are going to make Flappy invincible for a short period of time you can use the Scratch 'timer' to control this.

To start the timer, use the **reset timer** block.

This sets the timer to 0 and starts the clock running.

The timer can be used like other variables to control events.



This code changes the colour of the stage (or a sprite) if the **timer** is less than 5 seconds.

If the **timer** is not less than 5 seconds then the effect is removed by setting the effect to 0.

```
when green flag clicked
  forever loop
    if timer < 5 then
      set color effect to 20
    else
      set color effect to 0
```

The timer could also be used to end a Scratch game, move a game on to a new level or control when sprites move faster (the variables could be changed every time the timer reaches 10 seconds).



This is the hardest problem you have asked to solve in this project. The harder a problem is, the more you should try to refine it. This challenge has again been refined into lots of small problems.

## Challenge 8 - I'm Invincible

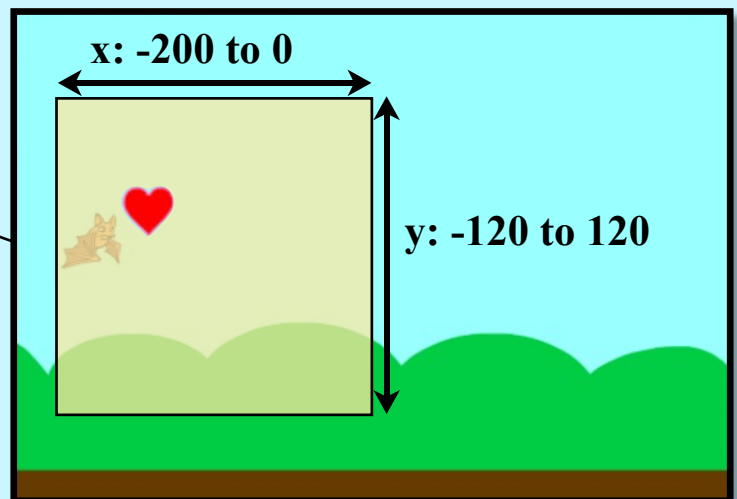
Solve each of the refined problems below.

1. Add a new sprite (you choose the picture) to the program and call it Invincible.



2. The Invincible sprite should appear every few seconds and then disappear a few seconds later.

3. The Invincible sprite should appear randomly somewhere inside the area shown on the right.



4. When the Invincible sprite touches Flappy Bat the sprite should disappear and the timer should be reset.
5. Change Flappy's code so that the game can only end if Flappy touches a Column or Percy and the timer is greater than 5.
6. Change the colour of the stage when the timer is less than 5 (you can use page 36 for this code).



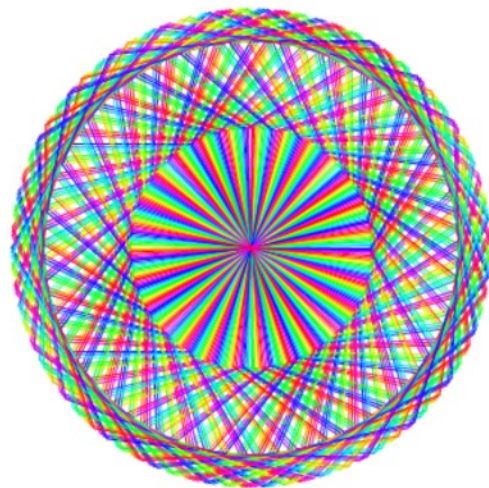
## Challenge 1

The first challenge was simply to type in the code and then make changes to it. The original code and pattern it produces are shown below.

```

when clicked
clear
set pen color to 0
go to x: 0 y: 0
pen down
repeat 240
  repeat 6
    move 80 steps
    turn 60 degrees
  turn 187 degrees
  change pen color by 5

```



If the first repeat is changed to 1 you can see the shape drawn by the code inside the second repeat.

```

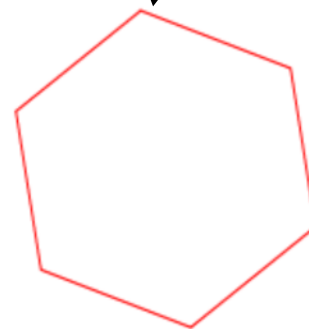
when clicked
clear
set pen color to 0
go to x: 0 y: 0
pen down
repeat 1
  repeat 6
    move 80 steps
    turn 60 degrees
  turn 187 degrees
  change pen color by 5

```

```

repeat 6
  move 80 steps
  turn 60 degrees

```





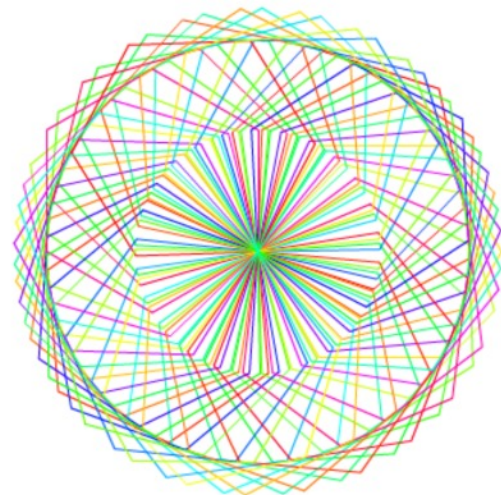
Here is the effect of some number changes.

```

when clicked
clear
set pen color to 0
go to x: 0 y: 0
pen down
repeat 60
  repeat 6
    move 80 steps
    turn 60 degrees
  turn 187 degrees
  change pen color by 5

```

Reducing the number of repeats would draw the same shape but fewer times.



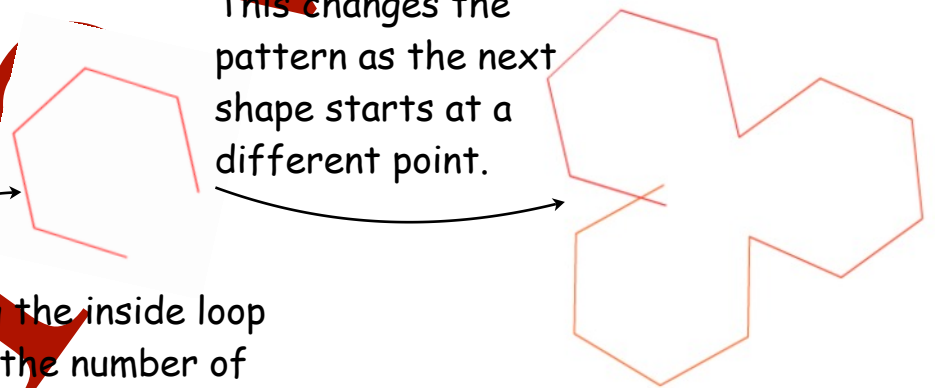
```

when clicked
clear
set pen color to 0
go to x: 0 y: 0
pen down
repeat 240
  repeat 5
    move 80 steps
    turn 60 degrees
  turn 187 degrees
  change pen color by 5

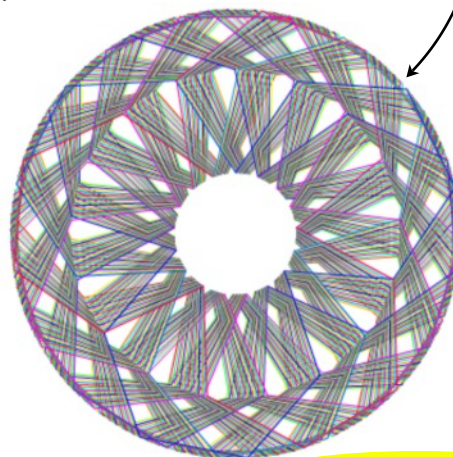
```

Changing the inside loop changes the number of sides in the shape.

This changes the pattern as the next shape starts at a different point.



This is the pattern it creates.





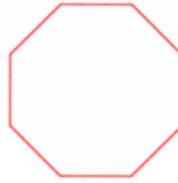


```

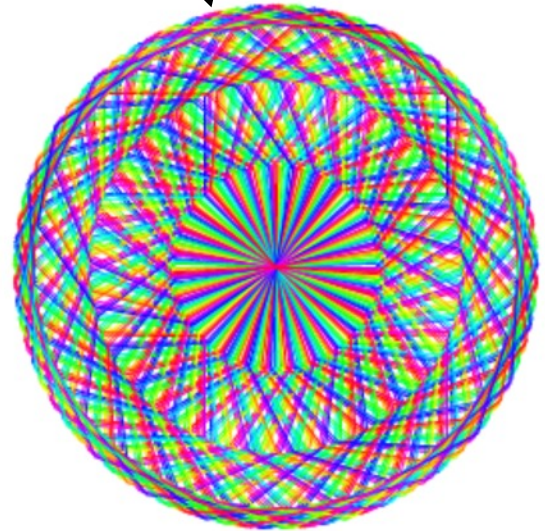
when green flag clicked
clear
set pen color to 0
go to x: 0 y: 0
pen down
repeat 240
  repeat 8
    move 50 steps
    turn 45 degrees
  turn 187 degrees
  change pen color by 5

```

The second repeat loop can be changed to draw a different shape.



Producing a slightly different pattern.



Here's a few other shapes that can be created using the second repeat loop.

Triangle

```

repeat 3
  move 100 steps
  turn 120 degrees

```



Square

```

repeat 4
  move 100 steps
  turn 90 degrees

```

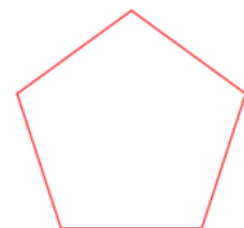


Pentagon

```

repeat 5
  move 100 steps
  turn 72 degrees

```





## Challenge 2

The second challenge was to find a way of controlling how fast Flappy's wings flap.

This can be done using the 'wait' block. The wait block stops the code running for a number of seconds. The loop then repeats changing the costume and stopping over and over.

To get a realistic flap the code should pause for just a fraction of a second as shown.

```
when green flag clicked
  forever loop
    next costume
    wait 0.2 secs
```

## Challenge 3

In the third challenge you had to solve the problem that Flappy only stays in the same place when the up key is pressed. This is because the move 5 up (when the up key is pressed) and move 5 down (each time it loops) inside the forever loop cancel each other out.

The solution is simple. Make Flappy move up further by increasing the number of steps inside the move block.

If Flappy moves 10 up and 5 down then overall he moves 5 up when the key is pressed.

```
when green flag clicked
  forever loop
    if key up arrow pressed? then
      point in direction 0
      move 10 steps
    point in direction 180
    move 5 steps
```



## Challenge 4

To make the columns move up and down we have to change the y coordinate in the move block.

```
when I start as a clone
show
go to x: 240 y: 0
repeat until x position < -235
  change x by -3
change Score by 1
delete this clone
```

To ensure that a different y coordinate is chosen each time a new clone is created we need to generate a random number.

```
pick random -45 to 45
```

The numbers should range from a negative value (bottom half of the stage) to a positive value (top half).

When this is used to replace the y coordinate the columns will appear at different heights.

```
when I start as a clone
show
go to x: 240 y: pick random -45 to 45
repeat until x position < -235
  change x by -3
change Score by 1
delete this clone
```



## Challenge 5

The changes made to make each game playable depend on each individual game.

Here's some advice on changes you could make to Flappy's code and the Column's code.



```

when clicked
  forever
    next costume
    wait 0.2 secs
  
```

```

when clicked
  forever
    if touching Column ? then
      broadcast Show Game Over
      stop all
  
```

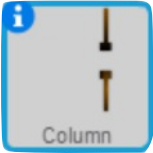
```

when clicked
  forever
    if key up arrow pressed? then
      point in direction 0
      move 10 steps
    point in direction 180
    move 5 steps
  
```

The two move blocks control Flappy's speed.

The game can be made harder by making both number bigger. Flappy would move faster up and down.

To make the game easier make both numbers smaller.



The range of random y coordinates controls the different heights the columns appear at.

If each column is closer in height to the previous one the game is easier, as Flappy does not have to move up and down so much. Decrease this range to make the game easier (for example -25 to 25).

The game can be made harder by increasing this range (for example -65 to 65).

```

when green flag clicked
  set Score to 0
  show
  go to x: 240 y: pick random -45 to 45
  repeat until x position < -235
    change x by -3
  change Score by 1
  hide

```

```

when green flag clicked
  forever
    wait 2 secs
    create clone of myself

```

```

when I start as a clone
  show
  go to x: 240 y: pick random -45 to 45
  repeat until x position < -235
    change x by -3
  change Score by 1
  delete this clone

```

The speed of the columns (and clones) is controlled by the 'change x' blocks. Increasing the negative value (-5) will make the columns move faster, making the game harder.

A lower value will make the game easier (-1)

Or will it? In this case maybe a column moving faster will make it easier to get through the gap. This one might depend on each individual player of the game.



## Challenge 6 - Percy, the Parrot of Doom (Extension Task)

To complete the Percy challenge you should add a parrot sprite and the following code.



```

when green flag clicked
  hide
  forever loop
    wait pick random 5 to 10 secs
    go to x: 220 y: pick random -120 to 120
    show
    repeat until x position < -230
      change x by -15
    hide
  
```

Percy starts hidden.

Percy waits before he appears.

He appears at the right hand edge (x) but appears at a random height (y).

Percy moves quickly (-15 each move) until he reaches the left hand edge (x 230).

Percy is hidden again. The forever loop then repeats everything above.

Percy flaps his wings while the game is running.

You also need to change Flappy's code in two places.



```

when green flag clicked
  forever loop
    if touching Column ? or touching Parrot ? then
      broadcast Show Game Over
      stop all
  
```

Flappy now has to check if he is touching a Column or Percy.



```

when green flag clicked
  go to x: -211 y: 13
  forever loop
    if key right arrow pressed? then
      point in direction 90
      move 5 steps
    if key left arrow pressed? then
      point in direction -90
      move 5 steps
    if key up arrow pressed? then
      point in direction 0
      move 10 steps
  point in direction 180
  move 5 steps

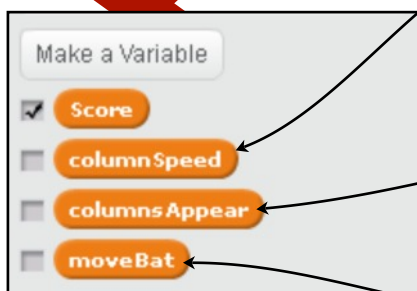
```

If Flappy can move left and right we have to make sure he starts in the correct place.

Two more if blocks are needed for the left and right keys. For each key Flappy must point in the correct direction.

## Challenge 7 - Speeding Things Up (Extension Task)

This challenge was all about variables. To complete the challenge you need to create 3 new variables. We know this because the problem describes three values that will change while the game is running.



The speed the columns will move at.

The rate at which new columns appear.

The speed Flappy moves up and down.



This variables should then be given a value when the game starts.



```

when green flag clicked
  set moveBat to 5

```



```

when green flag clicked
  set columnSpeed to -3
  set columnsAppear to 2

```

These are the values that are already used in the game.

You can now use the variables in the other parts of the program.



```

when green flag clicked
  go to x: -211 y: 13
  forever loop
    if key right arrow pressed? then
      point in direction 90
      move moveBat steps
    if key left arrow pressed? then
      point in direction -90
      move moveBat steps
    if key up arrow pressed? then
      point in direction 0
      move moveBat * 2 steps
    point in direction 180
    move moveBat steps

```

The moveBat variable will be used to replace how far Flappy moves.





```

when clicked
  set Score to 0
  show
  go to x: 240 y: pick random -45 to 45
  repeat until x position < -235
    change x by columnSpeed
  change Score by 1
  hide

```

```

when I start as a clone
  show
  go to x: 240 y: pick random -45 to 45
  repeat until x position < -235
    change x by columnSpeed
  change Score by 1
  delete this clone

```

The columnSpeed variable will be used to replace how far the column moves each time.

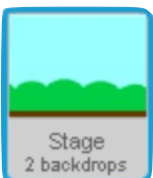
```

when clicked
  forever
    wait columnsAppear secs
    create clone of myself

```

The columnsAppear variable will be used to replace how long the program waits before it creates a new column.

Finally you need extra code to control when the variables will change. This code only changes variables so it can be added to the stage.



```

when clicked
  wait until Score > 5
  change columnSpeed by -2
  change columnsAppear by -0.5
  change moveBat by 2
  wait until Score > 10
  change columnSpeed by -2
  change columnsAppear by -0.5
  change moveBat by 2

```

The program waits until the score is greater than 5 (and then 10) and then the values stored in the three variables are changed.



## Challenge 8 - I'm Invincible (Extension Task)

To complete this task a new sprite is added.



The invincible sprite uses a forever loop to hide and show itself over and over.

```

when clicked
  forever
    hide
    wait pick random 2 to 8 secs
    go to x: pick random -200 to 0 y: pick random -120 to 120
    show
    wait pick random 2 to 4 secs
  
```

The sprite waits a random number of seconds before it appears.

The sprite appears at a random place on the left of the stage.

The sprite waits a random number of seconds before it disappears again.



```

when clicked
  forever
    if touching Bat1 ? then
      hide
      reset timer
  
```

When Flappy touches the invincible sprite the sprite disappears and the timer starts from 0.

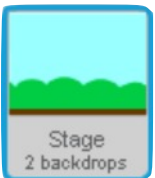


The game can only end now if Flappy is touching a Column or Percy and if the timer is over 5 seconds.

This means that while the timer is between 0 and 5 seconds Flappy can not die. He is invincible!

```

when green flag clicked
  forever loop
    if touching Column ? or touching Parrot ? and timer > 5 then
      broadcast Show Game Over
      stop all
  
```



The code to change the colour of the background was given to you before the task and simply had to be added to the stage.

```

when green flag clicked
  forever loop
    if timer < 5 then
      set color effect to 20
    else
      set color effect to 0
  
```

If Flappy is invincible (the timer is less than 5) change the colour of the stage.



## Glossary

Below is a dictionary of Computing Science words used in the Flappy Bat workbook.

### Clone

A copy of character or object in a game.

### Code

The instructions that make up a computer program. In a Scratch program, the blocks are the code.

### Loop

A computer instruction that makes other instructions repeat (run more than once). Loops may run forever, a set number of times or continue running until something happens in the program.

### Operator

AND, OR, NOT. Used to join two or more decisions together.

### Program

A program is one or more computer instructions that tell a computer how to carry out a task. Examples of programs are computer games, applications like PowerPoint or Word and mobile phone apps. Very large programs may have millions of instructions.

### Programming Environment

A program or website, like Scratch, that is used to write computer code and make your own programs.



## Refine

Refining is when a large problem is broken down into smaller problems that are then easier to solve.

## Run

A program is running when it is carrying out its instructions. When you are playing a computer game the game program is 'running'.

## Sprite

A character or an object in a computer game.

## Testing

Running a program to see if it works correctly. Testing often finds mistakes in code (called bugs) which then need to be fixed.

## To Code

A slang term meaning 'to write a computer program'.

## Variable

A stored value (number, letters or words) that can be changed by a program. In a computer game, a player's score would be stored in a variable.

PROVIEW