Flappy Bird Tutorial (Beginner)

Table of Contents

Overview

The Flappy Bird project is a great segway from beginner projects to intermediate ones; it utilizes many different skills and types of blocks. This project can be difficult for those who are still developing their sequencing skills, but serves as a fun and recognisable project students will be proud to say they've made.

Flappy Bird is a game where the player controls a character who can jump whenever they press the spacebar, but the character is always falling so they must keep pressing space to stay in the air. In addition, there are huge obstacles in their way and they have to jump to avoid them. The goal is to see how long they can stay alive and how many points they're able to get.

Platform and languages

Platform	Optional Platform	Language
Scratch	Piskel	Blocks

Project length

Two to three hours.

Skills

The main skills this project trains are:

- Physics
- Collisions
- Cloning
- Integers
- If statements

Customization ideas

This tutorial uses the typical bird and pipe sprites; encourage students to pick their own theme and sprites for their game. As long as their main character isn't too long or wide and what they use for the pipes has a similar shape, the game will function correctly.

This game isn't as complex as the original Flappy Bird; it doesn't increase in difficulty as it progresses. If students are interested, work with them to modify this version of the game to include more features, potentially some that weren't in the original Flappy Bird game like power ups or other types of obstacles.

Project

This project is separated into three segments.

Part 1: The bird

This section focuses on creating the main character and getting them to move.

Skills:

- Gravity
- Input
- Comparing values

Sprite setup

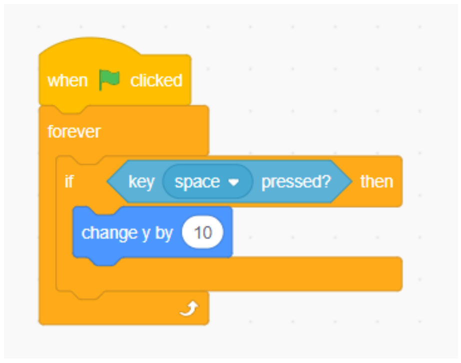
Select the sprite for your main character and background. Make sure the character isn't too irregularly shaped and is sized similarly to the sprite in the following image:

Basic setup

Jumping

Make the character jump with **Jump code**. This moves the character's y position up whenever the player presses space.

Jump code



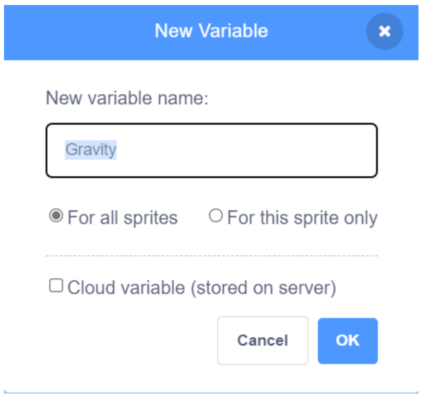
```
when 🏳 clicked
forever
    if <key (space ▾) pressed?> then
        change y by (10)
```

Note:Let the student notice how it isn't falling, ask them what makes objects fall in real life and see if they answer, gravity.

Gravity

Create a variable to keep track of gravity called **Gravity**.

Gravity variable



New Variable

New variable name:

Gravity

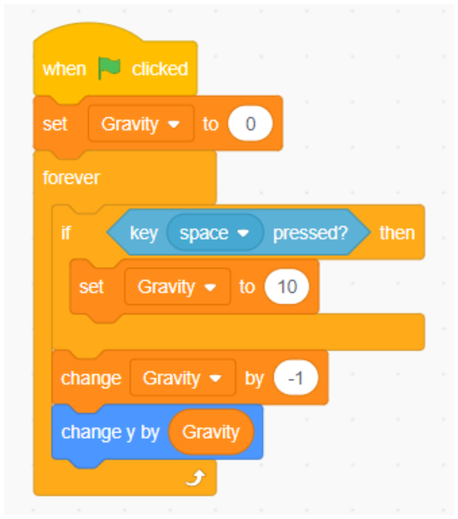
[✓ For all sprites] For this sprite only

☐ Cloud variable (stored on server)

Modify the **Jump code** to become **Gravity code**. The sprite's y position will change based on gravity. Gravity is always being decreased except for when the player presses the spacebar, allowing them to flap up a little.

Note: Let the student figure out the code. Try not to tell them what to do but encourage them as best you can and allow them to try out different numbers for the variables. If their numbers don't work, fixing them is another challenge for them to do.

Gravity code

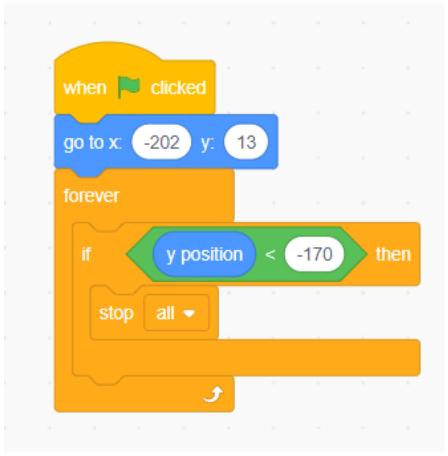


```
when flag clicked
set Gravity to 0
forever
    if key space pressed? then
        set Gravity to 10
    change Gravity by -1
    change y by Gravity
```

Splat

Now to end the game when the bird hits the bottom of the screen. Add this block of code to the main character. It makes the player start from the same spot at the start of the game, then checks to see if they've hit the ground; if they have, then the game ends.

Splat code

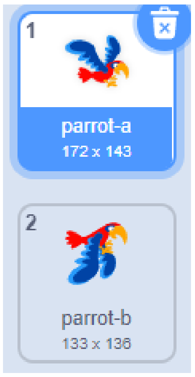


```
when flag clicked
go to x: -202 y: 13
forever
    if y position < -170 then
        stop all
```

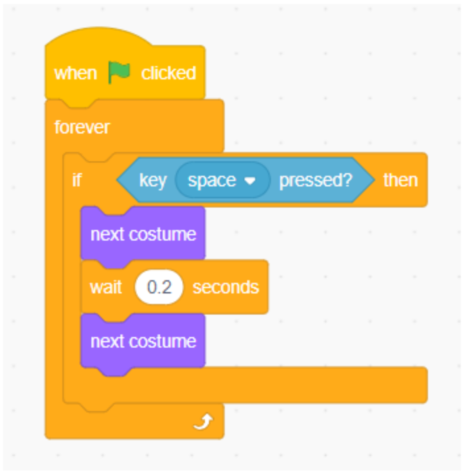
Flapping (optional)

If your sprite has two images, one with its wings up and the other with them down like the parrot, you can add a flapping animation:

Parrot flap



Add this block of code to switch costumes rapidly whenever the player presses space and make it look like the parrot is flapping:



```
when flag clicked
forever
    if key space pressed? then
        next costume
        wait 0.2 seconds
        next costume
```

Part 2: The pipes

This section focuses on creating the obstacles and getting them to move.

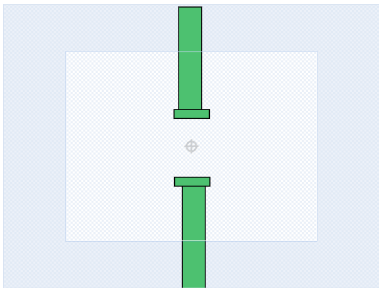
Skills:

- Cloning
- X-Y movement
- Bools

Setup

It's important the sprites you use for your obstacles are tall, otherwise you'll have to adjust many of the variables in this section. You can stretch them out under costumes but it's recommended you create your own so they're easier to fill the size requirements. They should be similarly sized as this image in the Scratch costume editor:

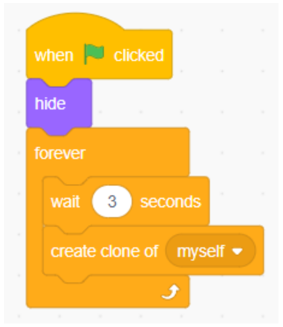
Length



Cloning

Cloning is an easy way to make the pipes appear every few seconds. To set it up, you can create a **forever loop** with a built in **wait** statement to stagger it. Include the **hide** at the top so the original pipe won't be seen.

Clones

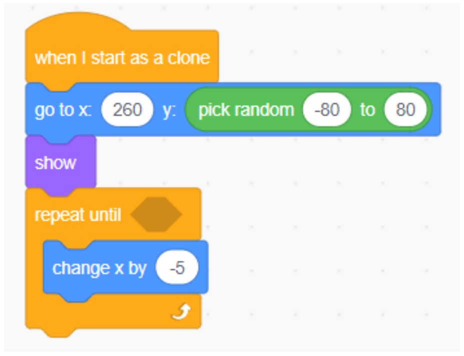


```
when 🏳 clicked
hide
forever
    wait (3) seconds
    create clone of (myself ▾)
```

Movement

This section needs to be under a **When I start as a clone** block to ensure it will work for all the clones that are made. It starts by sending the pipe to the right side of the screen; **pick random** adjusts the height on the y axis. It then moves it to the left of the screen.

Move

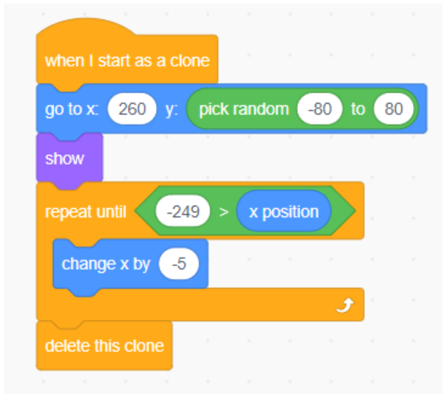


```
when I start as a clone
go to x: (260) y: (pick random (-80) to (80))
show
repeat until < >
    change x by (-5)
```

Deleting clones

Obstacles need to be deleted when they reach the left side of the screen. Add a clause to the **repeat until** loop in the **Move** block to delete the clones.

Delete clones



```
when I start as a clone
go to x: 260 y: pick random -80 to 80
show
repeat until -249 > x position
    change x by -5
delete this clone
```

Part 3: Collisions

This section focuses on setting up collisions between objects and allowing the player to lose.

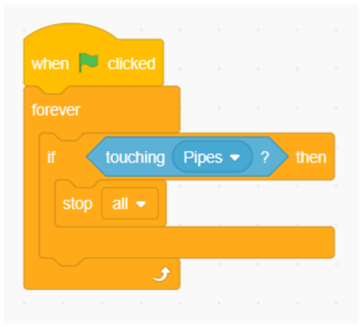
Skills:

- Collision code
- Triggering events
- Integers

Hitting a wall

The game should end when the player hits the wall just like when they hit the ground; this code is similar to that code except it checks for the collision with the **touching** block.

If touching



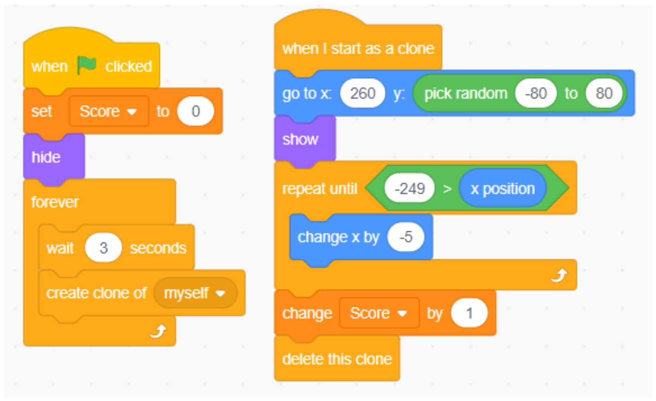
```
when 🏁 clicked
forever
    if touching Pipes ? then
        stop all
```

Getting points

Whenever the player passes an obstacle they should get a point. Create a variable named **Score**.

Modify these blocks in the obstacle by setting **Score** to zero at the start of each game and increasing it whenever a pipe is deleted.

Score block



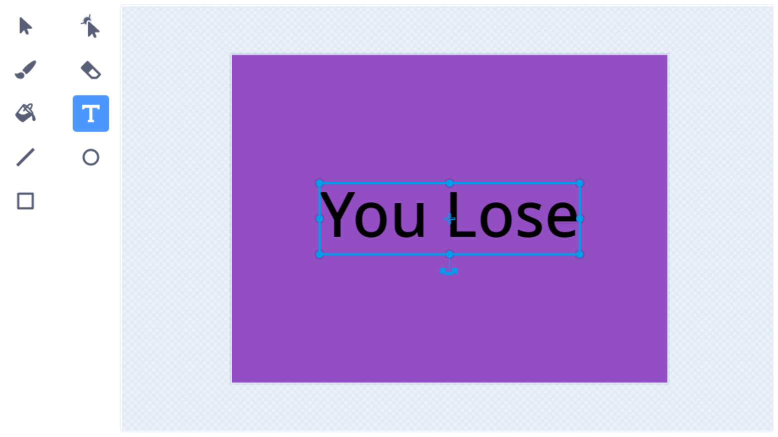
```
when [flag] clicked
set Score to 0
hide
forever
    wait 3 seconds
    create clone of myself

when I start as a clone
go to x: 260 y: pick random -80 to 80
show
repeat until -249 > x position
    change x by -5
change Score by 1
delete this clone
```

Losing

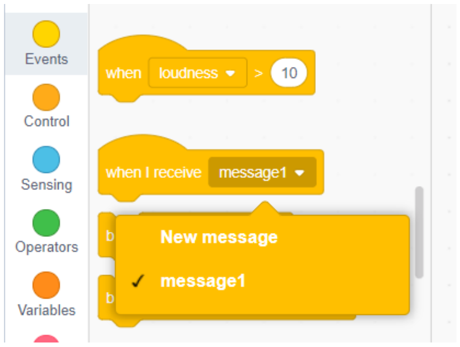
The game needs a you lost/try again screen so the player will know when they failed. Design your own using the text tool in Scratch or find one online.

Lose screen



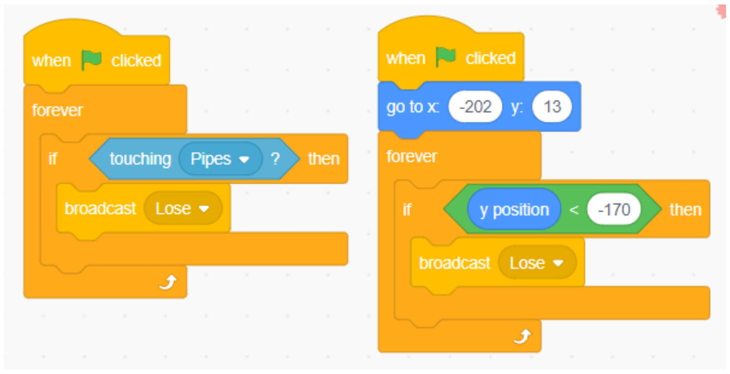
To make it appear, you need to broadcast a message to signal when it's time for it to show. Create a new message and call it **Lose**.

Lose message



Replace the **stop all** blocks with **broadcast Lose** blocks in these two code blocks inside of the player:

Broadcast Lose

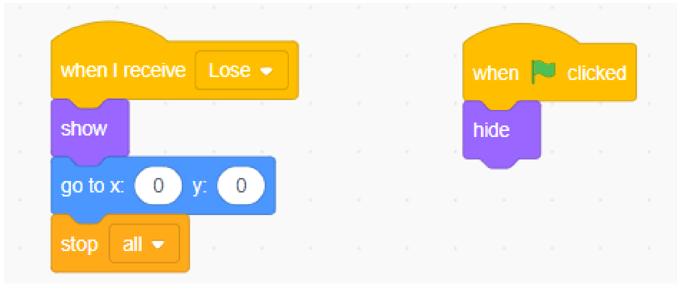


```
when 🏴 clicked
forever
    if touching Pipes ? then
        broadcast Lose
```

```
when 🏴 clicked
go to x: -202 y: 13
forever
    if y position < -170 then
        broadcast Lose
```

Add th;is code to the lose screen sprite. They make it disappear and reappear whenever the **Lose** message is broadcasted.

Game over



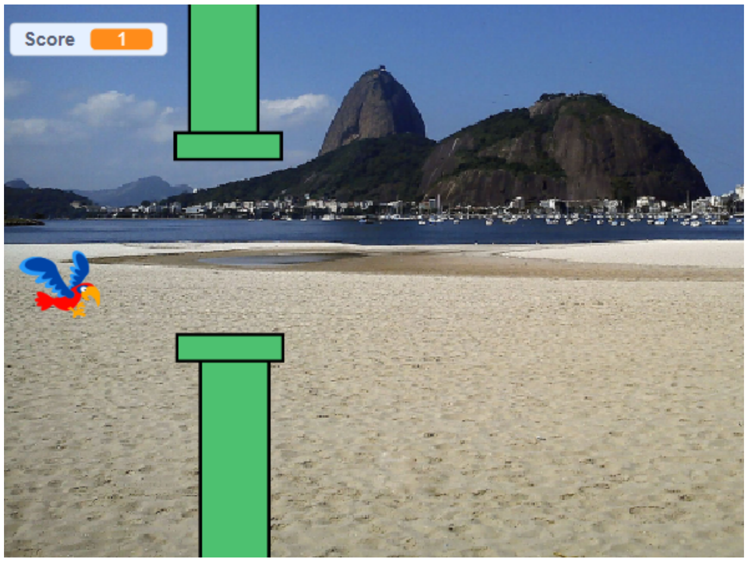
```
when I receive Lose
show
go to x: 0 y: 0
stop all
```

```
when 🏴 clicked
hide
```

Conclusion

The end product should resemble something like this:

Finished



Final notes

- Encourage your students to make their game balanced if they find it's too hard or easy.

- Add new features if your student wants to continue with this project.
- Find out what your students like about the project and use that to pick the next project to start with them.

Version	**Author**	**Reviewed by**	**Date**
0.1	Zacarias Milton		9/30/21