SCIENCE NORTH SCIENCE NORD
SUDBURY, ONTARIO, CANADA

Fractions and Flight	Grade 5 and 6

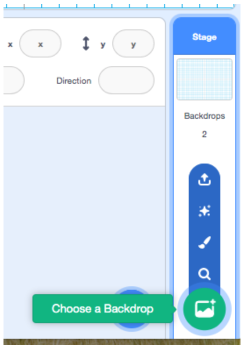
Flight Coding Guide

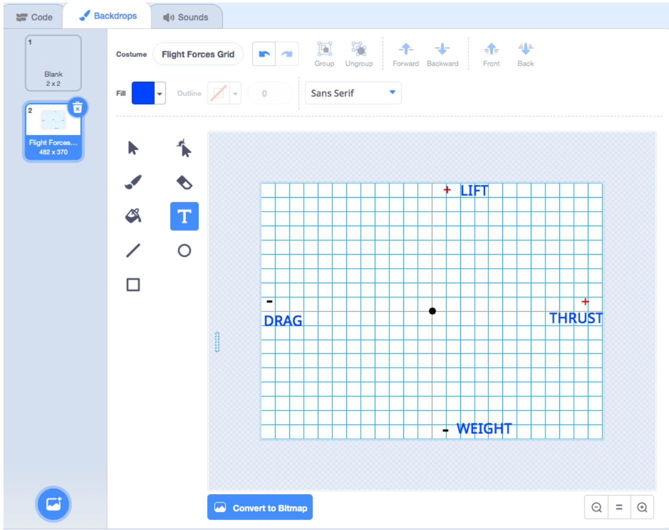
We will be looking at a Flappy Bird style game in Scratch. We will use code to mimic the forces of flight within the game.

We often say that in flight, we can group the forces into four. We have the WEIGHT of the aircraft pulling downwards, so we need to create LIFT to go in the opposite direction. We need a forward force that we call THRUST and the force of the air against the forward motion, we call DRAG.

In our code, we'll consider WEIGHT, LIFT and THRUST. We'll ignore DRAG for now.

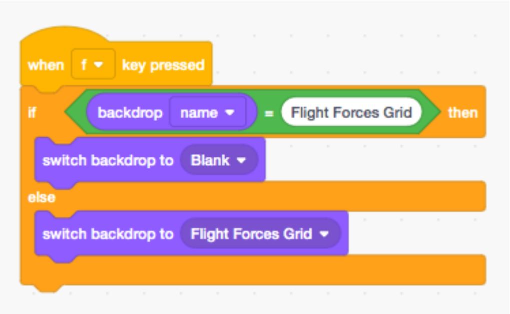
Let's create a backdrop that can help us remember the directions of these forces. From the Choose a Backdrop menu, find an XY grid.





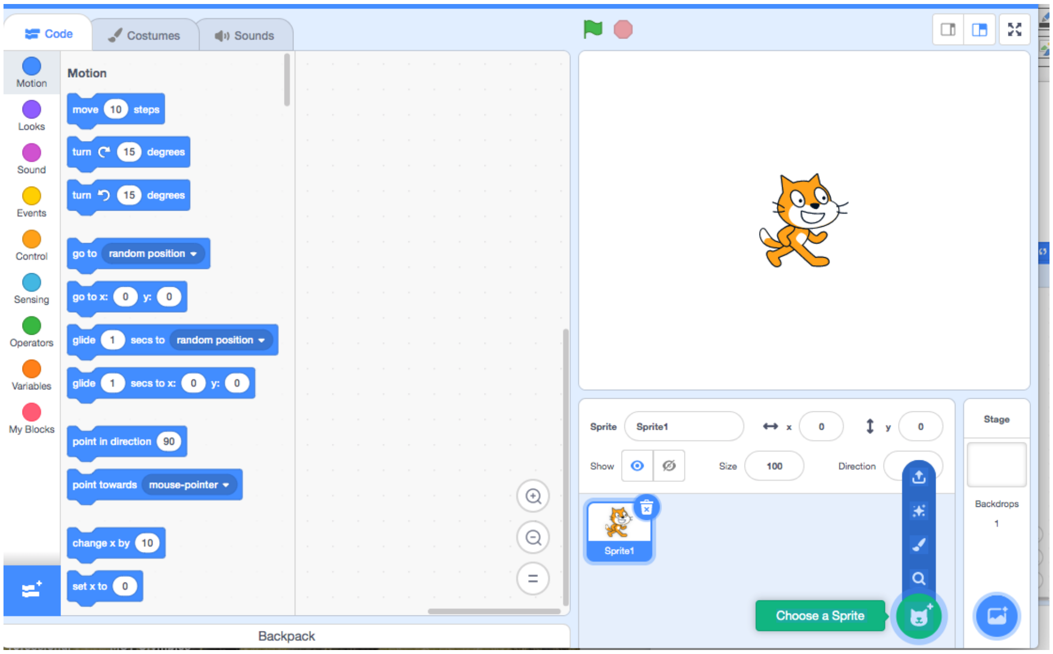
You can use the Xy-grid-20px. Select the Stage and then the Backdrops Paint Editor. You can add a small circle at the origin and the positive and negative signs on the x and y axes. The last part is the force labels and a name change to Flight Forces Grid. We won't want this backdrop all the time so let's add some code so we can toggle the grid on/off using the F key (for forces).

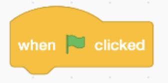
We'll use a When key pressed block from the Events menu and an If/Then/Else conditional statement from the Control menu. We'll need an Equals block from the Operators menu and some switch backdrop blocks from the Looks menu. If we are on the grid, we want to change to the Blank backdrop. If we are on the blank, we want to change to the grid. Let's set up the blocks to accomplish this.



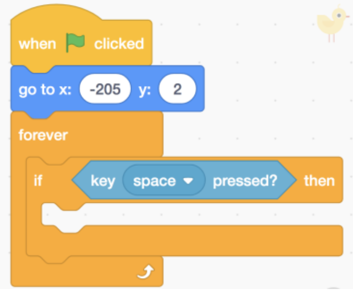
```
when f ▾ key pressed
if < backdrop name ▾ = Flight Forces Grid > then
    switch backdrop to Blank ▾
else
    switch backdrop to Flight Forces Grid ▾
```

Now we’ll find a bird sprite from the Choose a Sprite menu or draw one in the Sprite Paint Editor.



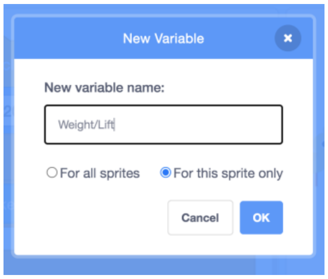


After you have chosen your sprite, start your code with a block from the Events Menu. We’ll want the bird to start at the left part of the screen. We can drag the bird to the position we want and then select the Go To block from the Motion menu. The numbers adjust to the position we have chosen so this will place the bird at the start position we have selected. From the Control Menu, we’ll need a Forever loop and an If/then conditional statement. Since we want our bird sprite to respond to the Space Bar, we’ll choose that block from the Sensing Menu.



```
when clicked
go to x: -205 y: 2
forever
    if key space pressed? then
```

When we begin the program, it will start to check if the space is pressed for as long as the program runs. To complete the conditional, we must include the ‘Then’ instruction. In order to do this, we will need to create a variable. Let’s call this variable Weight/Lift. We will use this variable to code a downward accelerating movement due to gravity and an upward lift movement when the Space is pressed.

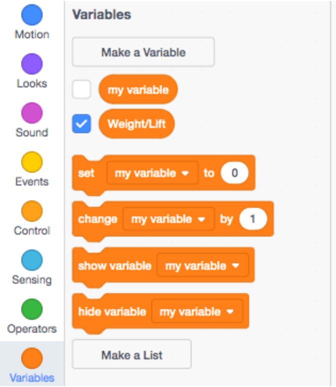


New Variable ×

New variable name: Weight/Lift

○ For all sprites ● For this sprite only

Cancel OK



Variables

Make a Variable

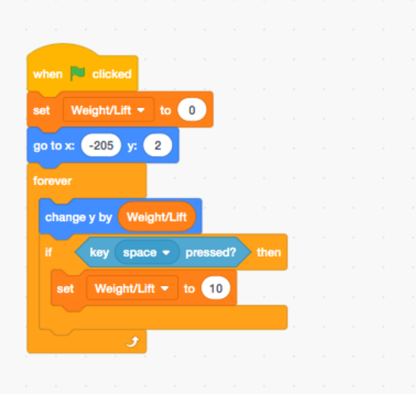
□ my variable

☑ Weight/Lift

```
set my variable to 0
change my variable by 1
show variable my variable
hide variable my variable
```

Make a List

In order to incorporate a movement that acts as the Weight, we need to have the bird changing in the Y direction. From the variable menu, we'll choose 2 Set variable blocks. We'll start the code by setting Weight/Lift to 0. We'll next set the Weight/Lift to 10 if the Space is pressed. We'll also add a Change Y by block and add the Weight/Lift variable there as well.

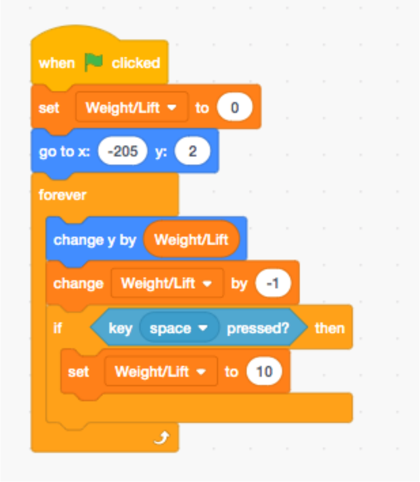


```
when 🏴 clicked
set Weight/Lift to 0
go to x: -205 y: 2
forever
    change y by Weight/Lift
    if key space pressed? then
        set Weight/Lift to 10
```

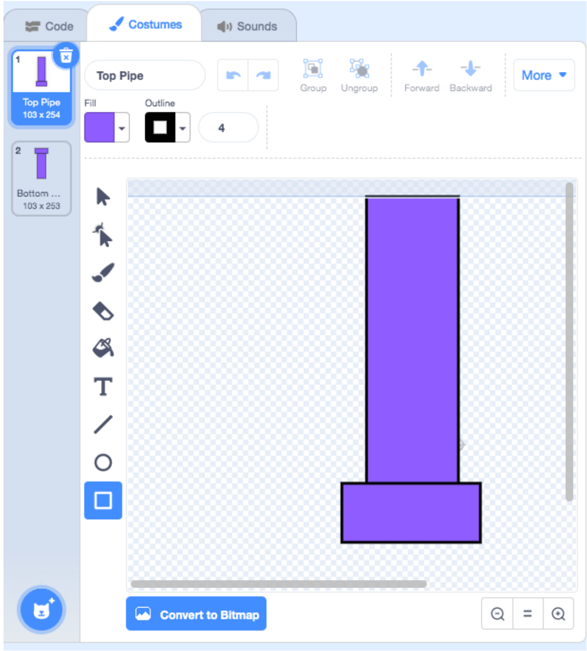
If we run the code right now, the bird will continue to go upwards since we haven't accounted for the weight yet, only the lift. In order to set the weight, we will add a Change block into the Forever Loop. This is so that the weight acts forever. The lift will only act when the space is pressed.

We want the change to be a negative number so that the bird goes downward. Let's choose -1.

If we run the code now, we'll see that the bird is always going down until we press the Space at which time, it goes up for a bit and then back down again. We have a jumping bird!



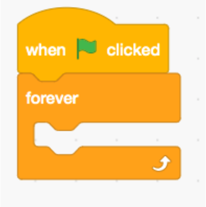
```
when 🏴 clicked
set Weight/Lift to 0
go to x: -205 y: 2
forever
    change y by Weight/Lift
    change Weight/Lift by -1
    if key space pressed? then
        set Weight/Lift to 10
```



In a Flappy Bird game, the bird flies between pipes. We'll work on coding them next but first we will need to create the sprites. Choose Paint from the Sprite menu. You can use the rectangle tool to draw simple pipes and select a colour. You'll need to create 2 costumes for the pipe sprites. One Top pipe and one Bottom pipe. Once you have created the first one, you simply need to copy it and use the Flip Vertical button in the paint Menu.

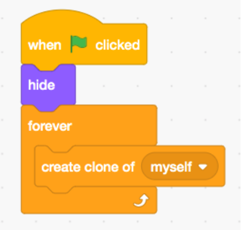
Now let's look at the code for the pipe sprites. We will look at how to use the cloning function to reproduce copies of the pipes in the game window.

We'll start with a When Green Flag Clicked and a Forever loop.



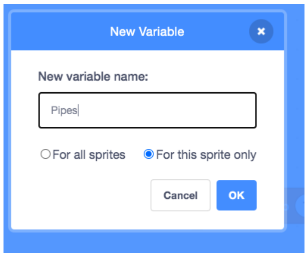
```
when ⚑ clicked
forever
```

We will use a Clone function to create copies of the pipes. Since there can be many copies of the pipe on screen at a time that leads to confusion, we'll also drag a Hide block from the Looks Menu.



```
when ⚑ clicked
hide
forever
    create clone of myself ▾
```

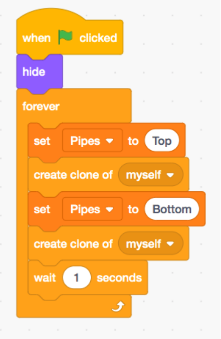
We want to create both bottom and top pipes though so we'll need create a new variable called Pipes. This variable will help us set whether we want a bottom or top pipe. When creating the variable, make sure to click for this sprite only.



New Variable

New variable name: Pipes

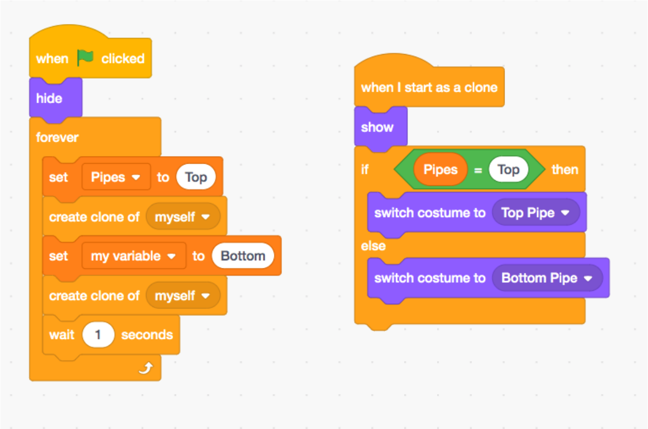
○ For all sprites ● For this sprite only



```
when ⚑ clicked
hide
forever
    set Pipes ▾ to Top
    create clone of myself ▾
    set Pipes ▾ to Bottom
    create clone of myself ▾
    wait 1 seconds
```

Now that we have the Pipes variable, we can set it to Top then create a clone and also set it to Bottom and create a clone. We'll also add a Wait block from the Control menu. This will give a second between clones.

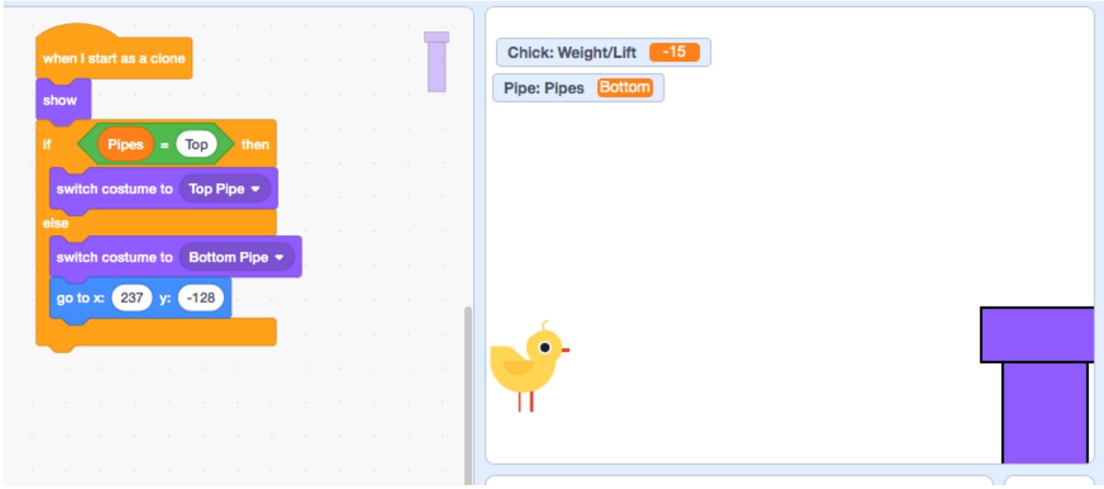
Now we want to change the Costume of the pipes so that the image reflects what we have Set. We'll add a conditional statement from the Control menu of the form If/Then/Else. We'll also use a new Control block called When I start as a clone and a new Looks block called Switch Costume. We'll need a Math Operator to complete the If part of the statement.



```
when [flag] clicked
hide
forever
    set Pipes to Top
    create clone of myself
    set my variable to Bottom
    create clone of myself
    wait 1 seconds
```

```
when I start as a clone
show
if Pipes = Top then
    switch costume to Top Pipe
else
    switch costume to Bottom Pipe
```

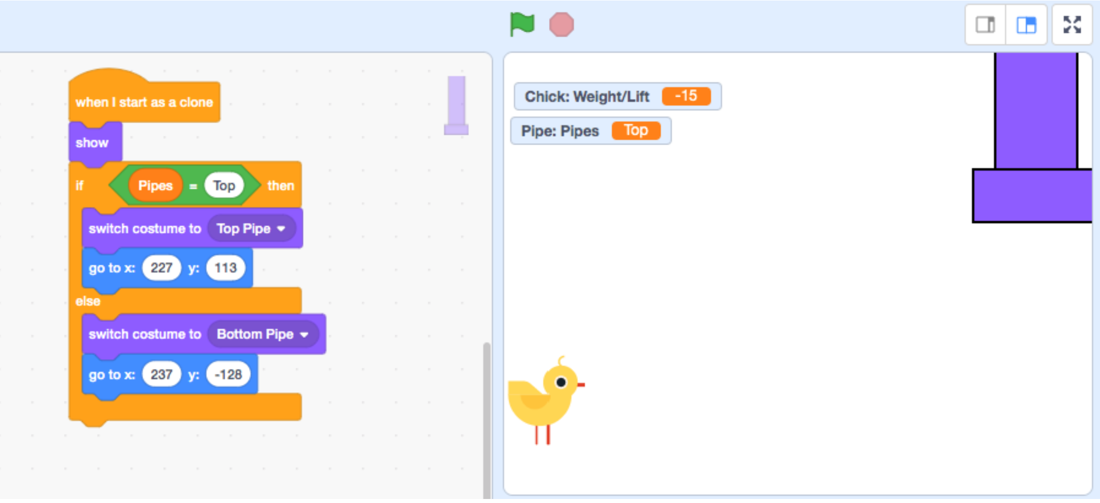
If we test our pipe code so far, we'll see that the new pipes appear in the bottom left corner of the stage. We'll want to move them so they appear from the far right of the stage instead. If we click the When I start as a clone block, a pipe should appear. You can drag it over to the left and place it where you think it should go. As you do this, the Go To block in the Motion menu will show the x and y coordinates for where you are placing the pipe. You can then place that block with that position into the code. After you do one pipe, do the other.



```
when I start as a clone
show
if Pipes = Top then
    switch costume to Top Pipe
else
    switch costume to Bottom Pipe
    go to x: 237 y: -128
```

Chick: Weight/Lift -15

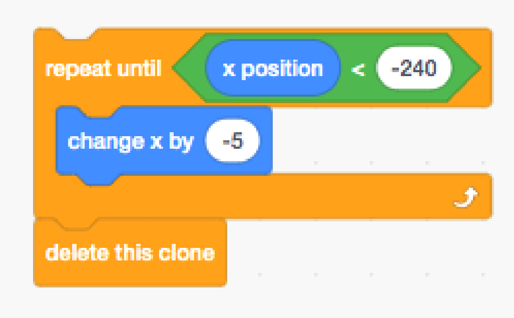
Pipe: Pipes Bottom



```
when I start as a clone
show
if <(Pipes) = (Top)> then
    switch costume to (Top Pipe ▾)
    go to x: (227) y: (113)
else
    switch costume to (Bottom Pipe ▾)
    go to x: (237) y: (-128)
```

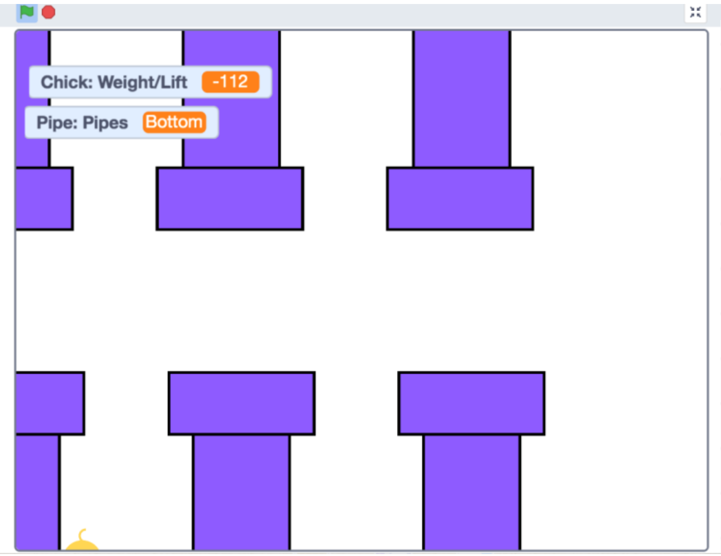
Chick: Weight/Lift -15
Pipe: Pipes Top

The next part that we will have to code is the pipe movement. The pipe movement is what makes it look like the bird is going forward. It is essentially the thrust force. We want the pipe to keep moving towards the left until it gets to the end of the screen. We'll use a Repeat Until block from the Control menu. We'll need to add a statement that we can test for to see if the Until is reached. We'll use a Less Than block from the Operators menu. After the x position is less than a number of our choice, then the pipe should disappear. You can take the pipe from the stage and bring it to the left to test what the x position should be when we want it to disappear. If we change x by a negative number, the sprite will travel to the left.

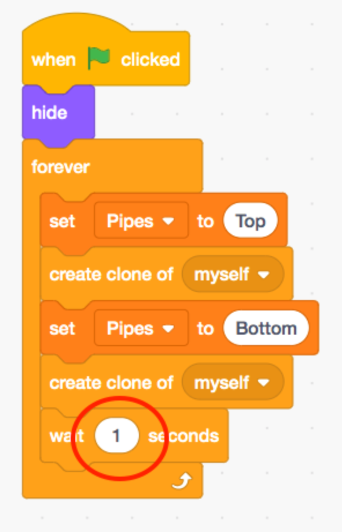


```
repeat until <(x position) < (-240)>
    change x by (-5)
delete this clone
```

If we test the code at this point, our pipes should spawn on the right and travel to the left and disappear. If we want to play with the rate at which the pipes appear, we need to change the time we indicated in the Wait block.

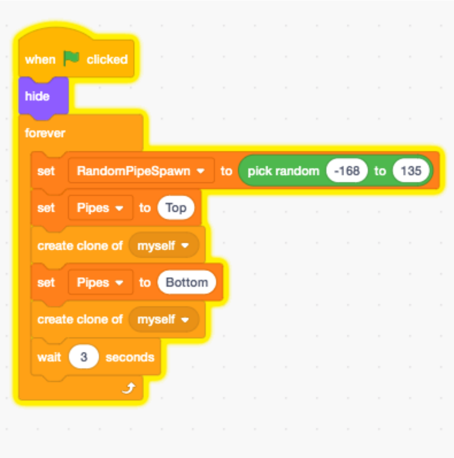


Chick: Weight/Lift -112
Pipe: Pipes Bottom

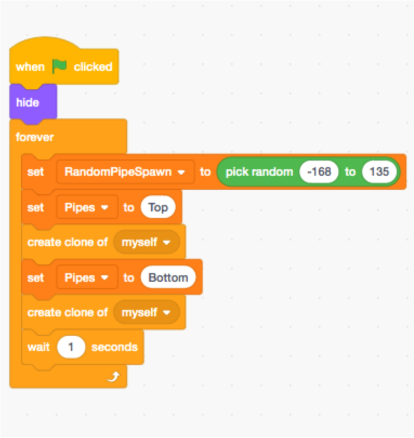


```
when ⚑ clicked
hide
forever
    set (Pipes ▾) to (Top)
    create clone of (myself ▾)
    set (Pipes ▾) to (Bottom)
    create clone of (myself ▾)
    wait (1) seconds
```

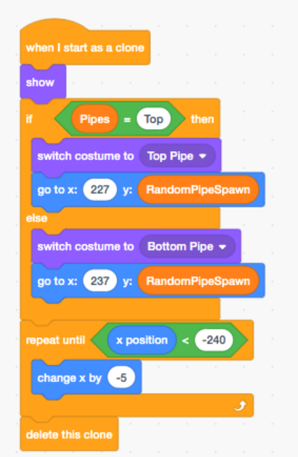
Since the code for the x position works, we want to move onto coding a change in y for the pipes. We want them to spawn at different heights. We'll need to create a new variable for this sprite only. We will call it RandomPipeSpawn. We'll want to set this variable in our Forever loop before we set the Pipes variable. Since we want it to be at random, we will choose the Pick Random block from the Operators menu. We'll need to determine what range we want the pipes to spawn into. We can move the pipes in the Stage to test out the topmost and bottommost positions. Move both pipes around, make note of their y position and enter the range into the Pick Random block.



```
when 🏴 clicked
hide
forever
    set RandomPipeSpawn ▾ to pick random -168 to 135
    set Pipes ▾ to Top
    create clone of myself ▾
    set Pipes ▾ to Bottom
    create clone of myself ▾
    wait 3 seconds
```



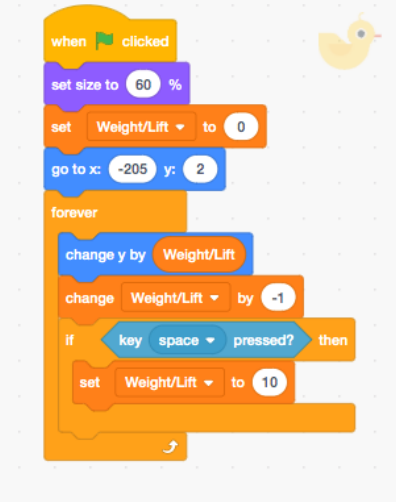
```
when 🏴 clicked
hide
forever
    set RandomPipeSpawn ▾ to pick random -168 to 135
    set Pipes ▾ to Top
    create clone of myself ▾
    set Pipes ▾ to Bottom
    create clone of myself ▾
    wait 1 seconds
```



```
when I start as a clone
show
if Pipes = Top then
    switch costume to Top Pipe ▾
    go to x: 227 y: RandomPipeSpawn
else
    switch costume to Bottom Pipe ▾
    go to x: 237 y: RandomPipeSpawn
repeat until x position < -240
    change x by -5
delete this clone
```

After we have decided the range, we can set our y position in the Start Clone code to the RandomPipeSpawn variable.

If we run the code now, we will see the pipes randomly spawning in the y position moving to the left and disappearing as if the bird was going forward(thrust). We see the bird goes up when we press Space(lift) and the bird falls elsewise(weight). In order to make this a playable game though, the size of the bird and the space between the pipes need to be maneuverable.

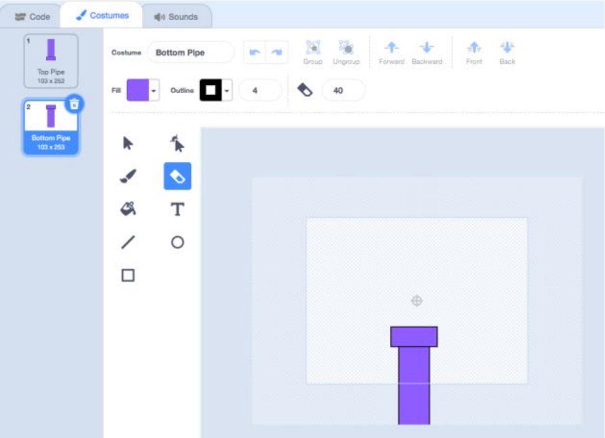


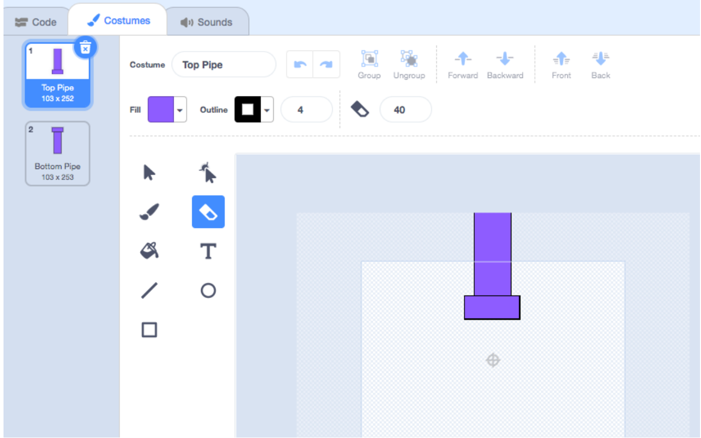
```
when 🏴 clicked
set size to 60 %
set Weight/Lift ▾ to 0
go to x: -205 y: 2
forever
    change y by Weight/Lift
    change Weight/Lift ▾ by -1
    if key space ▾ pressed? then
        set Weight/Lift ▾ to 10
```

In the Paint editor for the chick costume, we can remove the legs so it takes up less space. In the chick code, we can use the Set Size block from the Looks menu to adjust it. Let's try 60%.

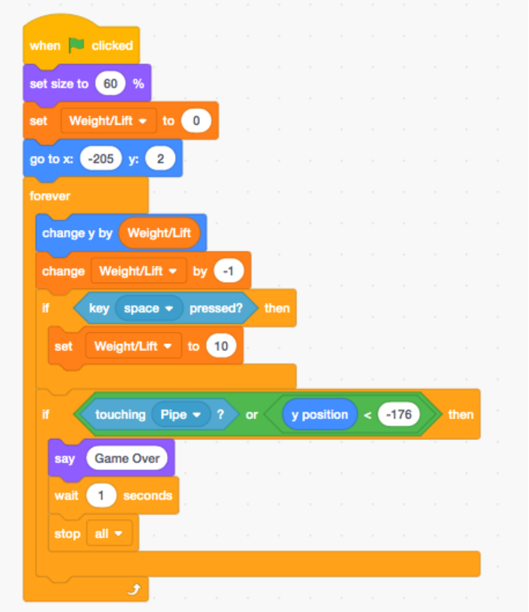
In order to reduce the frequency of the pipes spawning, we can adjust the Wait time in the pipes code. We can try 3 seconds between spawns instead of 1.

If we want to change the spacing between the pipes, we can do it in the Costumes Paint editor as well. We want them to be positioned directly above each other with the spacing we want to try. We can always make adjustments later if we don't like the spacing..



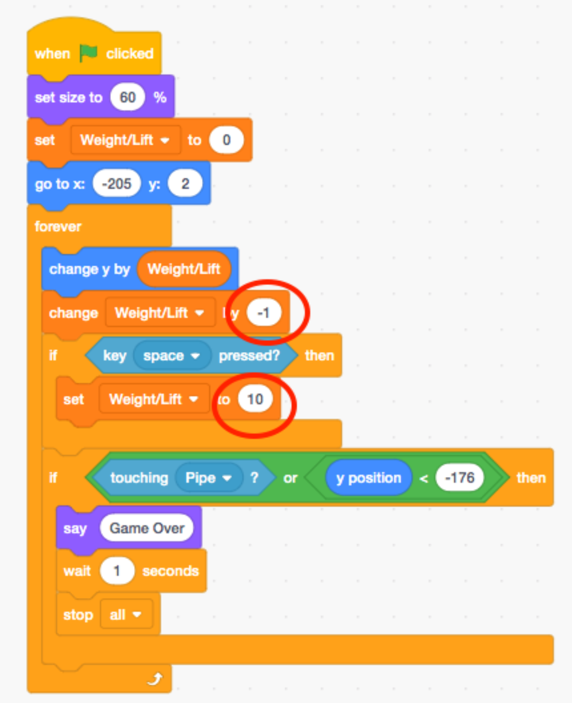


We can now add a Lose or End Game condition. When the chick touches a pipe or the bottom of the Stage, the game should stop. We'll use a conditional if/then statement and a Stop block from the Control menu. From the Operators menu, we'll select an Or and a Less Than. From the Sensing menu, we'll use a Touching block. From the Looks menu we can add a Say block and a Wait Block from the Control menu. We'll add these into the Forever loop for the bird. We will have to move the bird to the bottom of the Stage to determine the y position for the Less Than block.



```
when clicked
set size to 60 %
set Weight/Lift to 0
go to x: -205 y: 2
forever
    change y by Weight/Lift
    change Weight/Lift by -1
    if key space pressed? then
        set Weight/Lift to 10
    if touching Pipe ? or y position < -176 then
        say Game Over
        wait 1 seconds
        stop all
```

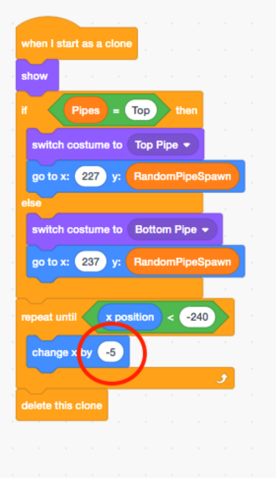
Here you can continue to adjust and modify certain parts of the code to affect playability like the spacing between the pipes and the size of the bird.



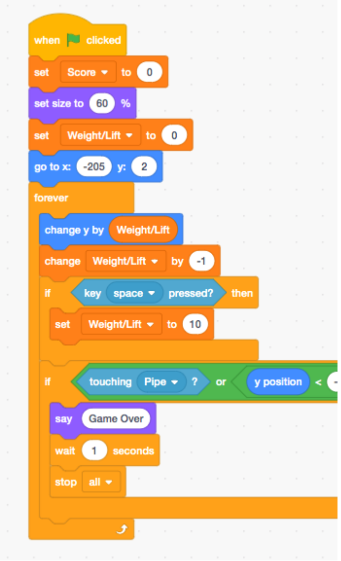
```
when 🏴 clicked
set size to 60 %
set Weight/Lift ▾ to 0
go to x: -205 y: 2
forever
    change y by Weight/Lift
    change Weight/Lift ▾ by -1
    if key space ▾ pressed? then
        set Weight/Lift ▾ to 10
    if touching Pipe ▾ ? or y position < -176 then
        say Game Over
        wait 1 seconds
        stop all ▾
```

We can also change the code in relation to the forces of flight. If the bird is on a different planet with a stronger gravitational field, how would we adjust the Weight? If we want the bird to get more lift on each Space press, where do we adjust?

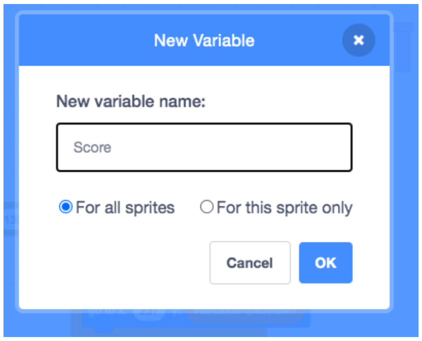
If we want to increase the thrust on the bird (speed of the pipes), what will we alter?



```
when I start as a clone
show
if Pipes = Top then
    switch costume to Top Pipe ▾
    go to x: 227 y: RandomPipeSpawn
else
    switch costume to Bottom Pipe ▾
    go to x: 237 y: RandomPipeSpawn
repeat until x position < -240
    change x by -5
delete this clone
```



```
when 🏴 clicked
set Score ▾ to 0
set size to 60 %
set Weight/Lift ▾ to 0
go to x: -205 y: 2
forever
    change y by Weight/Lift
    change Weight/Lift ▾ by -1
    if key space ▾ pressed? then
        set Weight/Lift ▾ to 10
    if touching Pipe ▾ ? or y position < -
        say Game Over
        wait 1 seconds
        stop all ▾
```



New Variable

New variable name:

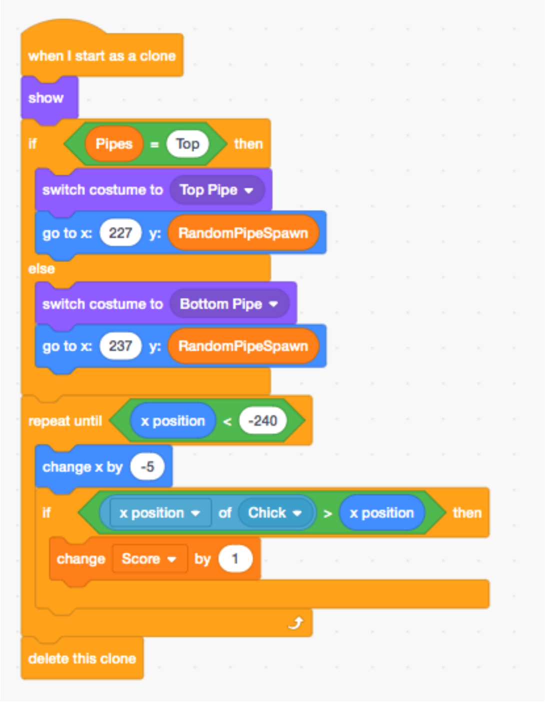
Score

● For all sprites ○ For this sprite only

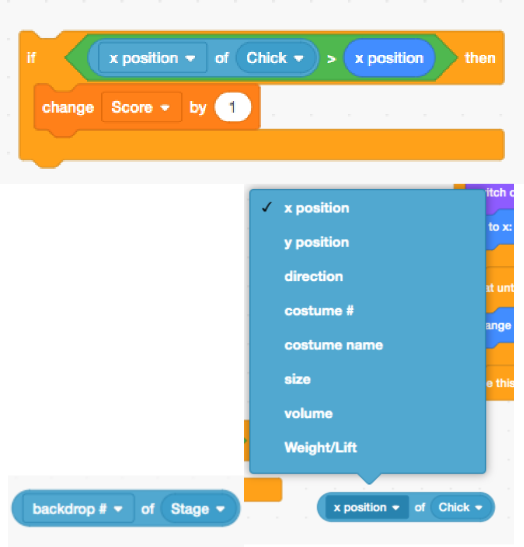
Extension: Now we have a playable game but it would be more fun if we could keep score! Let's add a score counter. We'll start by creating another variable called Score. This time we will set it For all sprites.

We'll want it to restart at 0 every time we play the game so let's set it to 0 When the Green Flag is clicked in the bird sprite code.

We want to gain points for every time the bird passes through a set of pipes. We will work within the code for the pipe Sprite. If the x position of the bird is greater than the the x position of the pipes then, add to the score. How will we write this with code blocks? We'll need an If/Then from the Control menu and a Greater Than from the Operator menu. From the Sensing menu, we can select the block that says Backdrop# of Stage. We'll use the drop-down menus within this block to make it say x position of Chick. Lastly, we'll need a Change variable block from the Variables menu.



```
when I start as a clone
show
if Pipes = Top then
    switch costume to Top Pipe
    go to x: 227 y: RandomPipeSpawn
else
    switch costume to Bottom Pipe
    go to x: 237 y: RandomPipeSpawn
repeat until x position < -240
    change x by -5
    if x position of Chick > x position then
        change Score by 1
delete this clone
```



```
if x position of Chick > x position then
    change Score by 1
```

We'll want to set the Score as the variable to change. We'll place it in the Repeat Until loop in the When I start as a clone pipe code.

Now when we test out the code, the score will increase for each set of pipes that the bird flies through.

Can you beat your own score in your own game?

Source code: https://scratch.mit.edu/projects/494730499/