Hour of Code

TouchDevelop

Student handbook

Name: _____

While you wait…

1. Join the WIFI
2. Go to
   https://www.touchdevelop.com/app
3. Sign In…

TURTLE WORKSHOP PART 1

Go to http://aka.ms/appdayturtle and follow the tutorial!

You can also follow the instructions below. You will need to follow the steps below and add the code in the **`main action`**.

```
action main ()
end action
```

0

The **turtle** can move and turn and its colored pen leaves a trail that creates the drawings. The code defines what the turtle does.
Let's add a **line of code** to tell the **turtle** to move **forward** by **200** steps.

```
♻turtle → forward(200)
```

Run your program: the turtle is moving forward 200 steps

1

The computer runs the code line by line. We told the turtle to move forward and so it did! Since **turtle** is facing up, we need to add the code to **turn** it to the **left** before moving again.
Let's add the code to make the **turtle turn left 90** degrees.

```
  ♻turtle → left turn(90)
```

Run your program: the turtle moves forward, then turns left

2

Let's add the code to **set** the **pen color**.

```
  ♻turtle → set pen color(colors → random)
```

3

The turtle is facing left, it is ready to move forward to draw the next side of the square.
Let's add the code to move **forward 200** again.

```
  ♻ turtle → forward(200)
```

Run your program: the turtle is halfway through the square

4

The turtle is facing left. It needs to turn before moving again.
Let's add the code to **turn left 90** degrees.

```
♻ turtle → left turn(90)
```

Run your program!

5

The turtle is facing down. It is ready to move forward to draw the next side of the square.
Let's add the code to move **forward 200**.

```
♻ turtle → forward(200)
```

Run your program: ready to turn!

6

Let's add the code to **turn left 90** degrees again.

```
♻ turtle → left turn(90)
```

Run your program: almost done!

7

Let's add the code to move **forward 200**.

```
♻ turtle → forward(200)
```

Run your program: nice square!

8

Did you notice that we repeated 4 times **forward** and **left turn**? Repeating code is very common in apps.
Let's **draw another square** where we will use a **for loop**. A **for loop** allows to repeat code efficiently.
Let's add a **for loop** to repeat code **4 times**.

```
for 0 ≤ i < 4 do
end for
```

9

The code nested under the **for** gets repeated 4 times. We need to move and turn on each iteration to draw the square.
Let's add the code to move the **turtle forward 200**.

```
for 0 ≤ i < 4 do
  ♻ turtle → forward(200)
end for
```

10

Let's add the code to **turn right** the **turtle 90** degrees.

```
for 0 ≤ i < 4 do
  ♻ turtle → forward(200)
  ♻ turtle → right turn(90)
end for
```

Run your program: tadaaa! another square!

11

Using **for loop** is quite powerful and can lead to really drawings. Let's try to draw a **spiral** using another **for loop**.
Let's first add the code to **speed** up the turtle.

```
♻ turtle → set speed(1000)
```

12

A **spiral** is usually built by repeating many times a move and a turn.
Let's add a **for loop** that repeats **100** times.

```
for 0 ≤ j < 100 do
end for
```

13

j is a variable that contains a number value. The variable **j** starts at 0 and grows by 1 on each iteration until it reaches 99 and the loop stops because **j < 100** is not true anymore..
If we move forward by **10 + j * 6**, the turtle will first move by 10, then by 16, then 22` and so forth.

```
for 0 ≤ j < 100 do
  ♻ turtle → forward(10 + j * 6)
end for
```

14

We are going to make the turtle turn **91 degrees instead of 90**. A single degree is a small change but repeated 100 times it will have great results!

```
for 0 ≤ j < 100 do
  ♻ turtle → forward(10 + j * 6)
  ♻ turtle → left turn(91)
end for
```

Run your program: tadaaa! a spiral!

15

It would be great if we could have our spiral go through the colors of a rainbow. We can do that by varying the **hue** of a color, similarly to turning a color wheel.
The **hue** needs to be a number between 0 and 1. We can do that by dividing **j** by **99** and **store** it in a **variable**.

```
for 0 ≤ j < 100 do
  ♻ turtle → forward(10 + j * 6)
  ♻ turtle → left turn(91)
  var hue := j / 99
end for
```

16

Let's add the code to pick a color from the **color** wheel using the **hue**.

```
for 0 ≤ j < 100 do
  ♻ turtle → forward(10 + j * 6)
  ♻ turtle → left turn(91)
  var hue := j / 99
  var rainbow := colors → wheel(hue)
end for
```

17

Let's add the code to **set** the **pen color**.

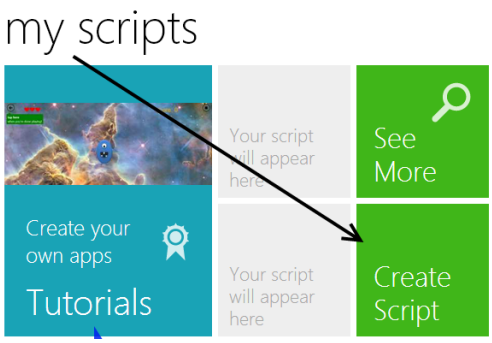
```
for 0 ≤ j < 100 do
  ♻ turtle → forward(10 + j * 6)
  ♻ turtle → left turn(91)
  var hue := j / 99
  var rainbow := colors → wheel(hue)
  ♻ turtle → set pen color(rainbow)
end for
```

Run your program: rainbow colors!

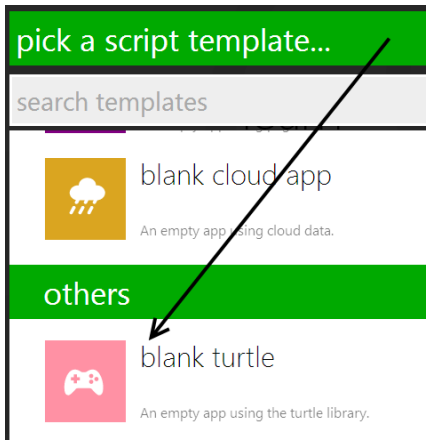
Awesome work! Let’s try the next workshop!

TURTLE WORKSHOP PART 2

- go to the TouchDevelop hub at https://www.touchdevelop.com/app
- tap the **Create Script** button



- scroll down and tap **blank turtle**



- pick a cool name and tap **Create**

You can also follow the instructions below. You will need to follow the steps below and add the code in the **`main action`**.

```
action main
end action
```

0

The code is entered line by line in TouchDevelop.

```
♻ turtle → forward(100)
```

Run your program

1

You can also command the turtle to turn left or right. All angles are in degrees.
Let's add the code to **turn right 120** degrees.

```
♻ turtle → right turn(120)
```

Run your program

2

You can change the pen color or size between any ♻turtle move. This allows to create cool designs.
Let's add the code set a random color and thicker size to the pen.

```
♻ turtle → set pen color(colors → random)
♻ turtle → set pen size(8)
♻ turtle → forward(100)
```

Run your program

3

Let's add the rest of the code to draw a triangle.

```
♻ turtle → right turn(120)
♻ turtle → forward(100)
♻ turtle → right turn(120)
```

Run your program

4

Next, we will use a for loop to repeat code more succinctly. The for loop is very similar to the Java or Snap! for. The index starts at **0** and increases by **1** on each iteration.
Let's add the code to create a for loop that repeats code **4** times.

```
for 0 ≤ i < 4 do
end for
```

5

The code nested between **for** and **end for** repeats 4 times.
Let's add a **forward** move and **right turn** of **90** degrees.

```
for 0 ≤ i < 4 do
  ♻ turtle → forward(100)
  ♻ turtle → right turn(90)
end for
```

Run your program

6

Excellent! Before we go on, let's add code to clear the screen and speed up the turtle.

```
action main
  ♻ turtle → clear screen
  ♻ turtle → set speed(1000)
  ...
```

7

for loops are very powerful. You can combine them to create cool shapes. For example, let's draw an hexagon 3 times and turn a little bit between each one. We can do that by **nesting** for loops.
Let's add the code that repeats **3** times and turn **120** degrees.

```
for 0 ≤ j < 3 do
```

```
   ♻ turtle → left turn(120)
end for
```

8

Let's add the nested **for** loop that draws a hexagon. This is the same idea as the square.

```
for 0 ≤ j < 3 do
   ♻ turtle → left turn(120)
   for 0 ≤ k < 6 do
      ♻ turtle → forward(100)
      ♻ turtle → right turn(60)
   end for
end for
```

Run your program

9

Let's add the code to change the pen color and draw a circle on each move to make it look better.

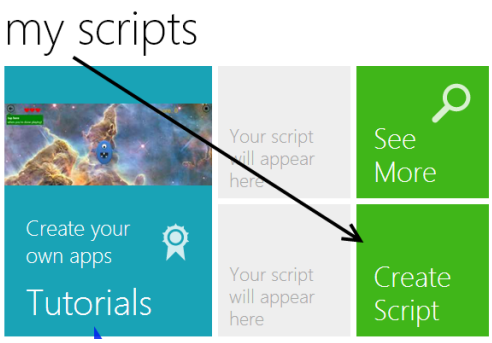
```
for 0 ≤ j < 3 do
   ♻ turtle → left turn(120)
   for 0 ≤ k < 6 do
      ♻ turtle → forward(100)
      ♻ turtle → right turn(60)
      ♻ turtle → set pen color(colors → random)
      ♻ turtle → circle(50)
   end for
end for
```

Run your program

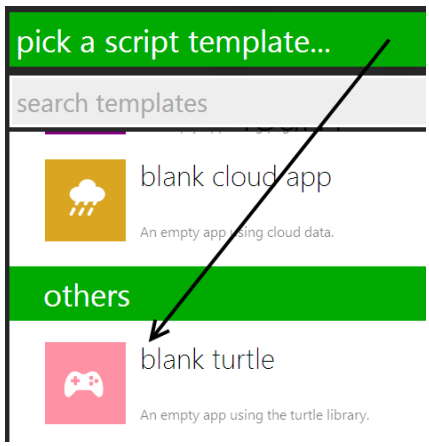
Yay! You're ready to draw your own shapes and drawings. Let's start started with it!

TURTLE WORKSHOP PART 3 (OPTIONAL)

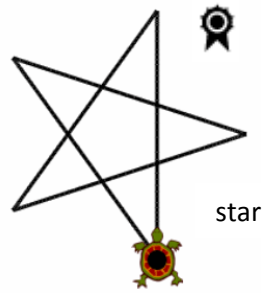
- go to the TouchDevelop hub at **https://www.touchdevelop.com/app**
- tap the **Create Script** button



- scroll down and tap **blank turtle**



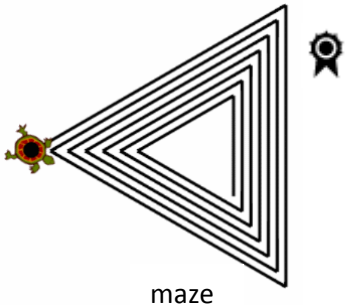
- pick a cool name and tap **Create**
- create some of these drawings… or come up with your own. **Earn 1 ticket for each completed**



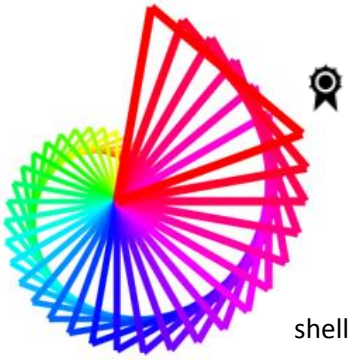
star

wheel

snowflake



maze



shell

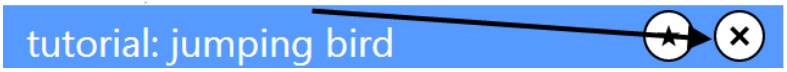
JUMPING BIRD WORKSHOP

Go to http://aka.ms/appdaybird

and follow the tutorial!

After finishing the guided tutorial, continue your jumping bird game. **If you've lost your script, go back to the TouchDevelop hub and you will find it under "my scripts".**

close the tutorial mode: in the editor, tap on the [x] button in the blue status bar at the bottom.



remove the debug mode: not needed anymore

```
~~board->set debug mode(false)~~
```

- run the code; the grid should be gone

move the bird to the left: the bird starts in the middle of the screen, let's position it at 1/5 of the screen horizontally to give the user more time to avoid the pipes. **Add this code at the bottom of main.**

```
sprite->x := board->width / 5
```

- run the code; your bird should be on the left of the screen

make the bird fall faster: the bird is falling but not quite fast enough. **Replace `400` with `500` when setting the acceleration y.**

```
sprite->acceleration y := 500
```

- run the code; your bird is falling faster. Change the gravity to your taste!

tilt the bird: make the bird tilt up and down based on his vertical speed. Add the **underlined** code below in the `sprite->on every frame` handler.

```
sprite->on every frame(perform)
    where perform() is
        sprite->set angle(sprite->speed y / 10)
        ...
```

- run your code; your bird is facing up while moving up and facing down while moving down

too many lives! By default, the game engine starts with 3 lives but you should only have 1 in this game! **Add this code at the bottom of main.**

```
♻game->set life(1)
```

move the pipes up and down: move the pipes up or down when they reappear to make the game harder. Add the **underlined** code below in the **if** statement that tests if the pipe left the screen.

```
for 0 ≤ ...
    ...
    pipe->on every frame(perform2)
        when perform2() is
        if pipe->x < 0 then
            ...
            pipe->set y(math->random range(board->height*3/4, board->height*5/4))
```

- run the code; the pipes should pop up at different vertical positions

add the upper pipes: the idea is to create a second sprite for the upper pipe in the **for** loop; then synchronize the x and y-coordinates with the bottom pipe (they y-coordinate will be shifted to the top). Add each **underlined** line of code to create this effect.

```
for 0 ≤ ...
    ...
    var pipe up := board->create rectangle(80, board->height)
    pipe up->color := pipe->color
    pipe->on every frame(perform2)
        when perform2() is
            pipe up->set x(pipe->x)
            pipe up->set y(pipe->y – board->height * 4/3)
            if pipe->x < 0 then ...
```

- run your code; you should have upper pipes!

detect collision with the upper pipe The bird can fly through the upper pipes unharmed (try it out). That is because we forgot to update the code that detects collision. Add the **underlined** code below.

```
for 0 ≤ ...
    ...
    pipe->on every frame(perform2)
        when perform2() is
            ...
            if pipe->overlaps with(sprite) or pipe up->overlaps with(sprite) then
                ...
```

- run your code and send the bird in the upper pipe. The game should end.

give the bird a heartbeat! Add a little 'beat' animation on the bird to make it look like it is moving. Add the **underlined** code under the line that creates the bird sprite. 0.3 is the duration is seconds, -1 is the number of repetition (-1 for infinity), 1.1 is the scaling.

```
var sprite := board->create picture(…)
sprite->create animation->beat(0.3, -1, 1.1)
```

- Run your code; the bird should be 'beating'. Change the duration and scaling values to your taste.

scroll the background: create a scrolling background by replacing wall->set background picture(...). The game engine will tile the background on the screen. Tip: select a picture that can be repeated without detecting the borders.

```
~~board->set background picture(✿background)~~
board->background scene->create layer(0, ✿background)
board->add on every frame(perform)
    where perform() is
        board->background scene->view x := board->background scene->view x + 1
end
```

- run your code; the background should be scrolling left.

use a picture for the pipes! Replace the code that creates rectangle sprites and use pictures instead.

play a sound when the screen is tapped. Add a new line of code in **sprite->on tapped**.

```
sprite->on tapped(perform)
    when tapped(x : Number, y : Number)
        ...
        ✿sound->play
end
```

- run your code; you should hear your sound at each tap.

change the rules!!! Make it your own, add your own twist to the flappy bird mania!

Other Cool Things to Try

- **Export your game to your phone**
- **Interested about building an informational app about your school, city or favorite band? Check out http://appstudio.windows.com/**