COMP1021
Introduction to Computer Science

Creating Turtles

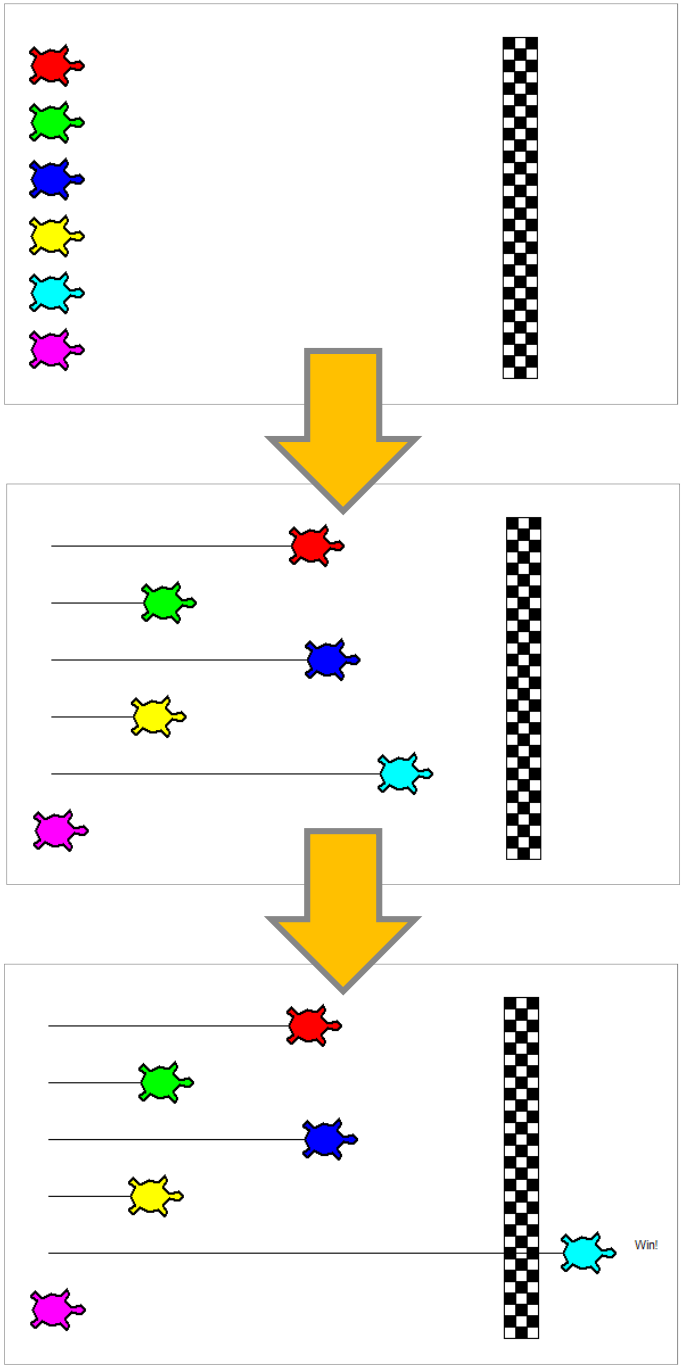
Gibson Lam and David Rossiter

Outcomes

- After completing this presentation, you are expected to be able to:
  1. Create additional turtles in a turtle program
  2. Reading and setting information about each individual turtle

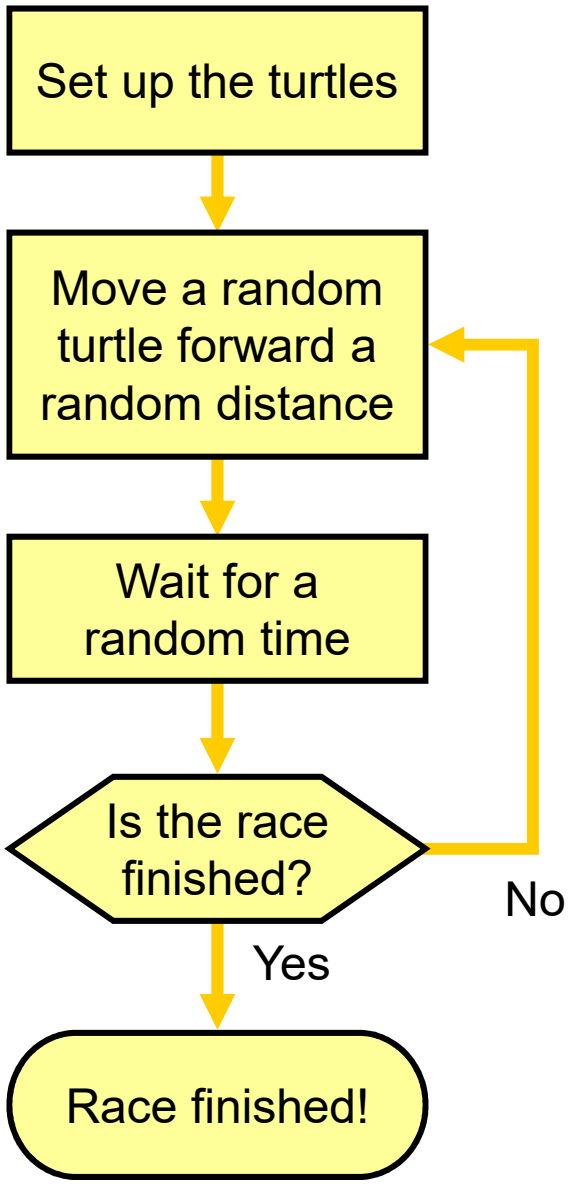
Using Multiple Turtles

- So far on the course we have only used one turtle
- Now we will learn how to create as many turtles as we want, and how to handle them
- In this presentation we will make a racing program
- When we run the program we watch as multiple turtles race to the finishing line



Flowchart of The Race

- The race is very simple
- We set up the race by creating six turtles
- A loop is used to repeatedly move one of the turtles a random distance towards the finishing line
- After the turtle is moved we wait for a random time before repeat the process again
- Whoever reaches the finishing line first is the winner of the race

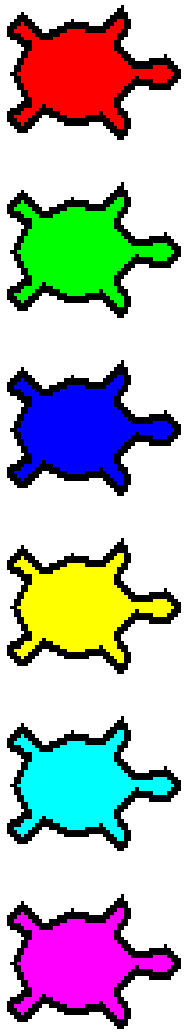


Creating a New Turtle

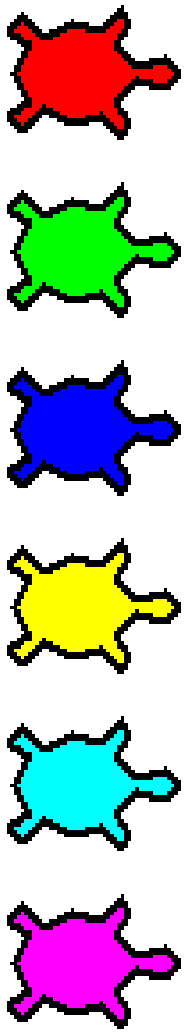
- This is how you create a new turtle:

```
another_turtle = turtle.Turtle()
```

- Now `another_turtle` is a new turtle
- After you create the new turtle, you can use all the techniques we know about e.g. `another_turtle.forward(100)`, `another_turtle.left(90)`, `another_turtle.color("red")`, and so on



Handling Multiple Turtles



- We use six turtles in our racing program
- For a nicer effect, we change the shape of the turtles to the “turtle” shape (instead of the default triangle shape) and use different colours
- To better manage the turtles we store them in a list

Adding a New Turtle to a List

- We start with an empty list:

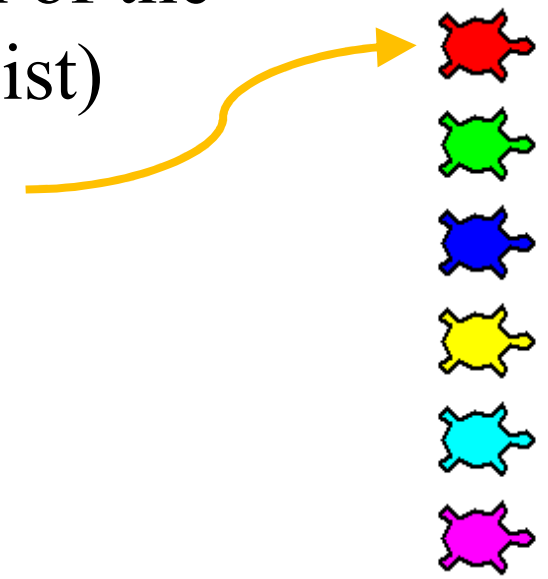
```
turtles = []
```

- Then, after we create a new turtle, we add it to the list of turtles using the `append()` function:

```
turtle_racer = turtle.Turtle()
turtles.append(turtle_racer)
```

Accessing Turtles in a List

- As you know, we can retrieve anything from a list by using the appropriate index e.g. `list[2]`
- This is true whatever is in the list, even a turtle
- In any list, the index is in the range 0 (for the first item in the list) to the length of the list minus 1 (for the last item in the list)
- For example, to access the first item in the list we can use `turtles[0]`



Creating the Turtle Racers

```
def createturtle(color):
    t = turtle.Turtle()
    t.shape("turtle")
    t.fillcolor(color)
    . . .
    turtles.append(t)

. . .
# A list to store the turtle racers
turtles = [] # Start with empty list

# Create the racers
createturtle("red")
createturtle("green")
createturtle("blue")
. . .
```

- The turtle racers are created and then stored in a list so they can be accessed later

Create a new turtle and change the shape and colour of it, then add it to the list

Create some turtle racers with different colours

Moving a Turtle

- We move a random turtle in some random distance in a function using a loop

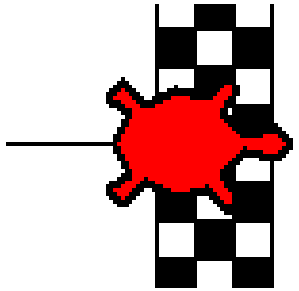
```
def moveturtle():
    . . .

    index = random.randint(0, 5)

    turtles[index].forward(random.randint(50, 200))

    . . .

while not finished:
    moveturtle()
    time.sleep(random.randint(100, 1000) / 1000)
```

Pick a turtle to move randomly

Move the turtle in some random distance

Wait for a random time before moving another turtle

Checking Turtle Position



- To check who has won the race we can check the x position of any turtle like this:

```
x = turtles[index].xcor()
if x > 200:
```

show that the turtle has won

- The `turtle.xcor` command returns the current x position of a particular turtle
- There are many more commands you can use to get information from a turtle, as shown in the next slide

Getting and Setting Turtle Information

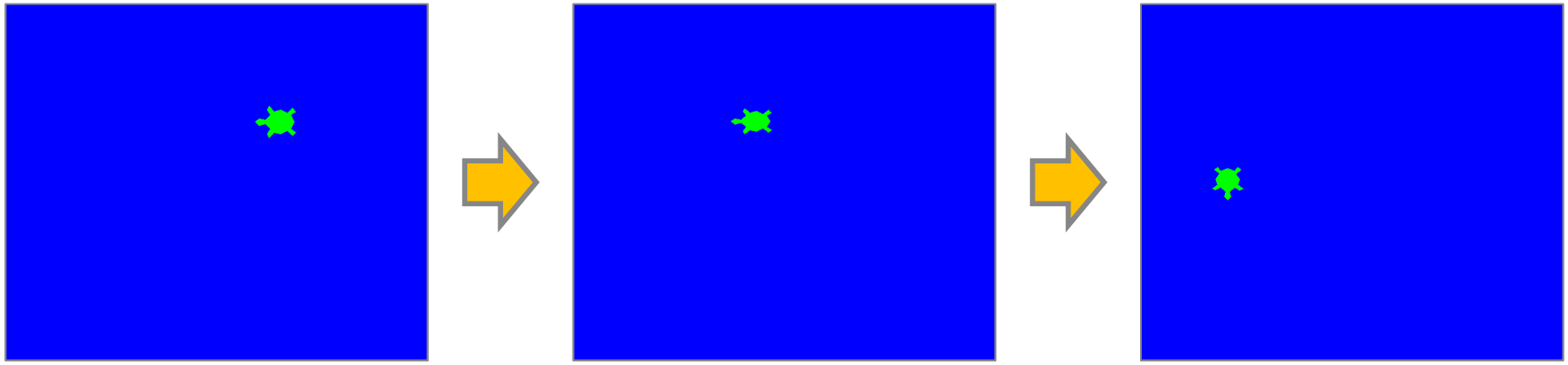
- If your turtle is stored in a variable `myturtle`, you can get its information like this:

`myturtle.xcor()`	= the x position
`myturtle.ycor()`	= the y position
`myturtle.position()`	= both x and y together
`myturtle.heading()`	= the heading angle
`myturtle.fillcolor()`	= the fill color
`myturtle.speed()`	= the speed
`myturtle.shape()`	= the shape

- You can also set its information using commands such as `myturtle.setx()`, `myturtle.sety()` and `myturtle.setheading()`

A Swimming Turtle Example

- In this example, the turtle ‘swims’ along a rectangular area of the turtle window
- While swimming the turtle keeps on changing its size based on its position



The Main Loop of the Example

```
while True:
    x, y = turtle.position()
    heading = turtle.heading()

    turtle.forward(5)
    if heading == 0   and x > 200  or \
       heading == 90  and y > 200  or \
       heading == 180 and x < -200 or \
       heading == 270 and y < -200:
        turtle.setheading(heading + 90)

    turtle.shapesize((turtle.xcor() % 60) / 60 + 2, \
                     (turtle.ycor() % 60) / 60 + 2)
```

Get the current position and heading of the turtle

Move the turtle and turn the turtle if it is at the boundary of the area

Set the exact angle

Change the size of the turtle shape based on the x and y positions of the turtle