COMP1021
Introduction to Computer Science

Turtle Shapes

Gibson Lam and David Rossiter

Outcomes

- After completing this presentation, you are expected to be able to:
  1. Change the shape of the turtle in turtle programming
  2. Adjust the size of the turtle
  3. Put a turtle stamp inside the window

Turtle Shapes

- There are several different shapes you can use for the turtle:
  - arrow, turtle, circle, square, triangle and classic
- You can also use any image in GIF format
- This means you can change the turtle shape according to the program you are creating
- For example, in a music program where the user see the turtle move, you can change the turtle to a musical note:

Turtle Shapes You Can Choose

- Arrow 
- Turtle 
- Circle 
- Square 
- Triangle 
- Classic 

The default shape of a turtle is "classic"

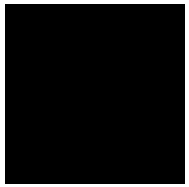
Changing the Turtle Shape

- To change the shape of the turtle you can use the following code:

  `turtle.shape(` *name of the shape* `)`

  where shape is one of the names of the shape listed in the previous slide

- For example:

  `turtle.shape("square")`

  changes the shape of the turtle to a square

Using Your Own Image

- Apart from the default turtle shapes you can also use any GIF image as your turtle shape
- For example, to use the GIF image on the right as the turtle shape you can use the following code:

ninja.gif

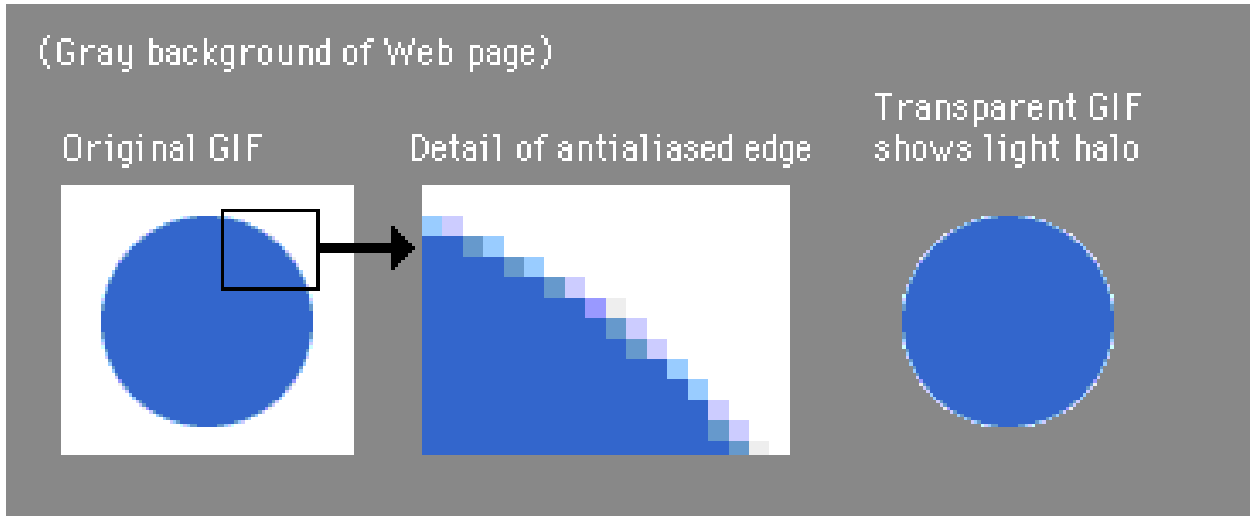
```
turtle.addshape("ninja.gif")
turtle.shape("ninja.gif")
```

Add the image so that it can be selected as a turtle shape

Use the newly added shape (the image) as the turtle shape

GIF Images

- You have to use a GIF image, not other types
- GIF images have a maximum of 256 different colours
- GIF images support simple transparency, but it's too simple for many situations i.e.



- Usually these days you would choose PNG format instead of GIF format – but PNG isn't supported

- This program shows all the possibilities, one by one

```
import turtle

def draw():
    turtle.clear()
    for _ in range(4):
        turtle.forward(100)
        turtle.left(90)

def arrow_shape():
    turtle.shape("arrow")
    draw()

def circle_shape():
    turtle.shape("circle")
    draw()
```

```
def triangle_shape():
    turtle.shape("triangle")
    draw()

def turtle_shape():
    turtle.shape("turtle")
    draw()

def square_shape():
    turtle.shape("square")
    draw()

def classic_shape():
    turtle.shape("classic")
    draw()
```

```
def gif_shape():
    turtle.addshape("ninja.gif")
    turtle.shape("ninja.gif")
    draw()
```

The GIF file needs to be in the same directory as the Python program

```
# Start of the main program
print("Repeatedly press Enter to see a new shape")

arrow_shape()
input("Press Enter")
circle_shape()
input("Press Enter")
triangle_shape()
input("Press Enter")
turtle_shape()
input("Press Enter")
square_shape()
input("Press Enter")
classic_shape()
input("Press Enter")
gif_shape()
input("Press Enter")
turtle.done()
# End of
# program
```

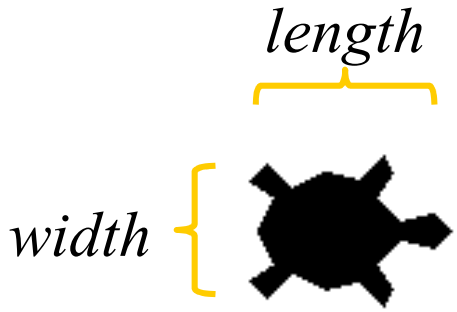
Changing the Size of Turtle Shapes

- Sometimes the turtle may look too small
- You can use `turtle.shapesize()` to make it bigger (or smaller if you like)
- For example, you can double the size of a turtle shape using this code:

```
turtle.shapesize(2, 2)
```

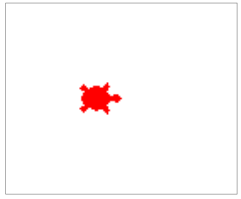
Make the width of the shape doubly bigger

Make the length of the shape doubly bigger

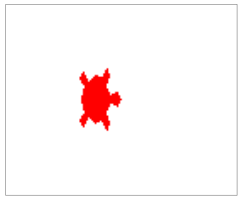


More Turtle Size Examples

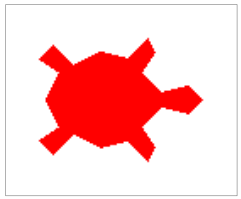
- Original turtle shape



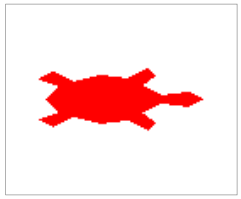
- `turtle.shapesize(2, 1)`



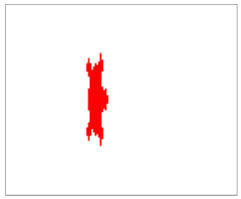
- `turtle.shapesize(4, 4)`



- `turtle.shapesize(2, 4)`



- `turtle.shapesize(3, 0.5)`

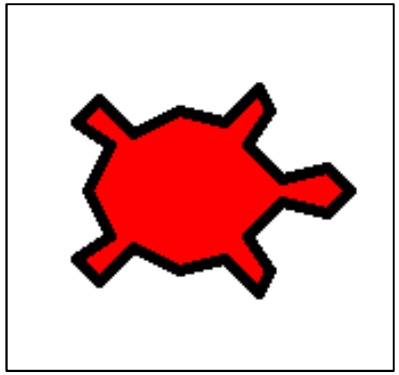


Changing the Size of Turtle Outline

- Apart from the size of the turtle you can also adjust the width of the turtle outline
- This can be done by giving a third input to the `turtle.shapesize()` function
- For example, the following line of code makes the turtle to have a thick outline:

```
turtle.shapesize(5, 5, 5)
```

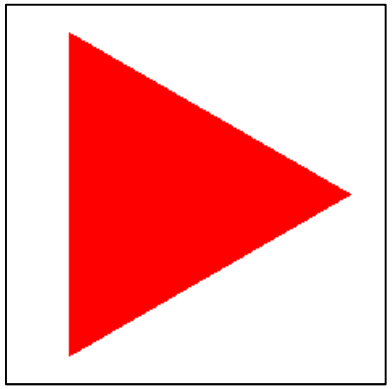
Use a thick outline for the turtle shape



Stamping the Turtle Shape

- You can ‘stamp’ the turtle shape in the window
- This is convenient when you want to draw things which resemble one of the available turtle shapes
- For example, you can quickly ‘stamp’ a big triangle in the window using this code:

```
turtle.shape("triangle")
turtle.shapesize(10, 10)
turtle.color("red")
turtle.stamp()
```



- Unfortunately, stamping does not work with your own GIF image as the turtle shape

Removing the Stamps

- After making the stamps you can remove the stamps by `turtle.clearstamps()`
  - You can remove the first *n* stamps by providing a positive number *n*, or
  - You can remove the last *n* stamps by providing a negative number -*n*, or
  - You can remove all stamps if you don't give any number at all

Turtle Stamping Example

- The following example stamps 100 turtles randomly in the window and then removes them in reverse order:

```
turtle.shape("turtle")
turtle.shapesize(3, 3, 2)
turtle.color("black", "green")
```

Use a green turtle with a black outline

```
for _ in range(100):
    turtle.goto(random.randint(-250, 250), \
                random.randint(-250, 250))
    turtle.left(random.randint(0, 359))

    turtle.stamp()
```

Randomly put turtles in the window

```
for _ in range(100):
    turtle.clearstamps(-1)
```

Remove the turtles from the last one to the first one

Turtle Stamps in the Window