COMP1021
Introduction to Computer Science

Using Screen Events

Gibson Lam, David Rossiter and Leo Tsui

Outcomes

- After completing this presentation, you are expected to be able to:
    1. Write code to handle mouse click events on the turtle window (not the turtle)
    2. Write code to handle key press events

Events We Have Looked At

- So far, we have looked at the following events:
  - Click (clicking on a turtle)
    e.g. `turtle.onclick(drawcircle)`
  - Drag (dragging a turtle)
    e.g. `turtle.ondrag(turtle.goto)`
  - Timer
    e.g. `turtle.ontimer(draw, 2000)`
- Now let's look at using these screen events:
  - Clicking on the turtle window (not the turtle)
  - Pressing a key

Clicking on the Turtle Window

- `onscreenclick()` is used for when you click on the turtle window (the event does not occur if you click on a turtle)
- For example:

```
def myfunction(x, y):
    . . .
```

x and y give the location where the click occurred, they are automatically given to the function

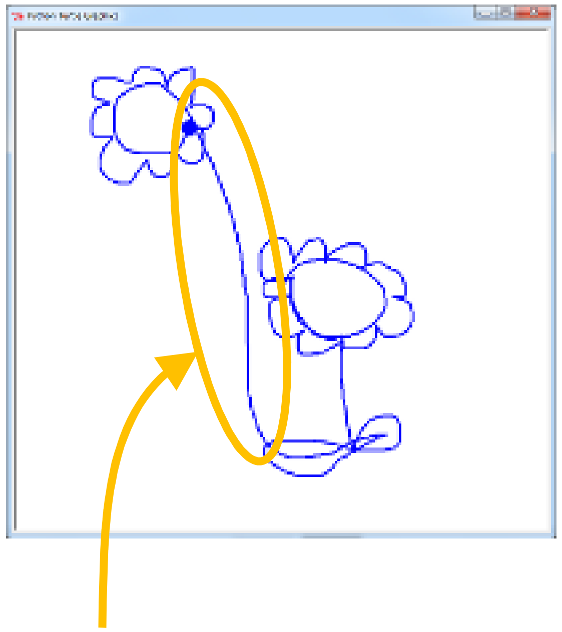
```
turtle.onscreenclick(myfunction)
```

The mouse click event is applied to the turtle window

When the user clicks somewhere on the turtle window (but not on a turtle) the `myfunction` *function will be executed*

Improving the Previous Drawing Program

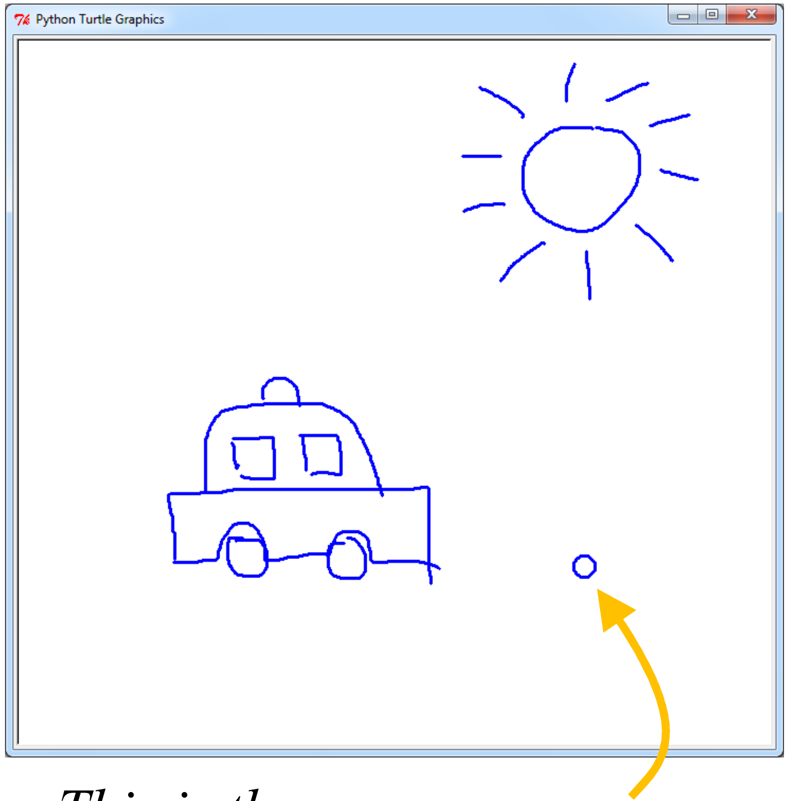
- In previous discussions, we showed a ‘drawing program’ which used the mouse drag event
- A problem with that program is that the resulting lines have to be connected
- We can improve that drawing program by also using the *screenclick* event to jump to a new place



When the previous program is used an unwanted line connects everything

Improving the Drawing Program

- Using the screenclick event the turtle can ‘jump’ to a new position – without drawing any line from the old position
- An example picture drawn using the improved drawing program is shown on the right
- That means pictures can be created which are not made from a single long line



This is the appearance of the turtle in the improved drawing example

Improved Drawing Program

```
import turtle
def jump(x, y):
    turtle.up()
    turtle.goto(x, y)
    turtle.down()
```

This function moves the turtle to a new position (x, y) without drawing a line to that position

```
turtle.ondrag(turtle.goto)
```

The turtle goes where it is dragged; the `goto` *function is automatically given the x and y values*

```
turtle.onscreenclick(jump)
```

The turtle jumps to a new position when the user clicks on the window; the jump function is automatically given the x and y value

```
turtle.done()
```

Wait forever for any event to occur; run the appropriate event handler function

Making the Turtle Better

- The code on the previous slide gives the most important code in the program
  (i.e. the code which handles the event)
- However, this code is also included in the program to make the turtle easier to see and drag around:

```
turtle.shape("circle")     # Looks better than a triangle
turtle.fillcolor("")       # Make the circle hollow
turtle.shapesize(1, 1, 3) # Make the outline thicker
turtle.pencolor("blue")    # Looks nicer than black
turtle.pensize(3)          # Make the drawn lines thicker
turtle.speed(0)            # Make the turtle move quickly
```

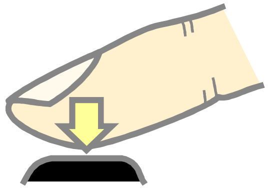
Pressing a Key

- Let’s look at another type of screen events, handling keys
- There are two kinds of action you can do on a key: pressing (push down) a key and releasing a key
- In this presentation, we focus on handling the pressing of a key, which is typically more useful than the releasing of a key

The Key

- You have to state the name of a specific key when you set up a keyboard event
  - For example, you can use ‘a’, ‘b’, … ‘z’ or ‘0’ … ‘9’
- It can also be a special key, such as:
  - ‘Return’ – Enter key
  - ‘Escape’ – Esc key
  - ‘Up’ – up arrow key
  - ‘Down’ – down arrow key

The Key Press Event



- The `onkeypress()` function assigns an event handling function for handling the key press event of a particular key
- For example:

```
def mykeyfunc():
    . . .
```

Whenever the user presses 'a' this function will be executed

```
turtle.onkeypress(mykeyfunc, 'a')
```

The key press event is applied to the turtle window

The `mykeyfunc` *function is assigned to the key press event*

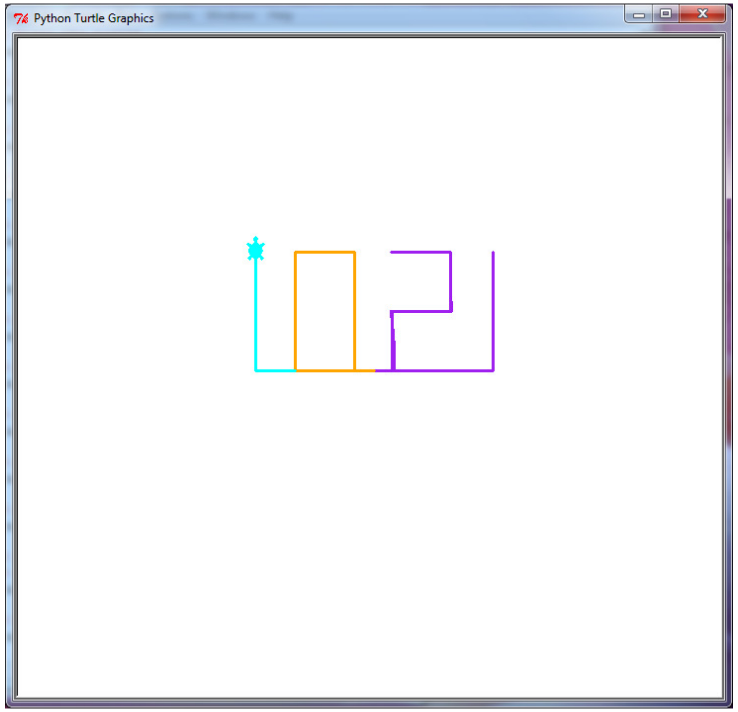
A key ('a' in this example) that is handled by the event handler

Listening for Keyboard Events

- For keyboard events, simply setting up the event handling functions is not enough
- To be able to receive key events a special function `turtle.listen()` has to be called
- This function tells the turtle window to listen for any keyboard events that occur in the window
- You need to have `turtle.done()` at the end of the program, like usual

Key Events Example

- This example uses keys to control the movement of the turtle:
  - Up key – move forward
  - Down key – move backward
  - Left key – rotate left
  - Right key – rotate right
- It also allows colour change:
  - ‘o’ key – orange
  - ‘p’ key – purple
  - ‘c’ key – cyan



Key Events Example 1/3 – Event Handlers for Turtle Movement

```
pixels_for_one_step = 4
angle_for_rotation = 5
```

```
def moveforward():
    turtle.forward(pixels_for_one_step)

def movebackward():
    turtle.backward(pixels_for_one_step)
```

These event handler functions move the turtle forward (up arrow key) or backward (down arrow key)

```
def rotateleft():
    turtle.left(angle_for_rotation)

def rotateright():
    turtle.right(angle_for_rotation)
```

These event handler functions rotate the turtle to the left (left arrow key) or right (right arrow key)

Key Events Example 2/3 – Event Handlers for Changing Colour

```
def orange():
    # Change the pen color and
    # the turtle to orange
    turtle.color("orange")
```

For the 'o' key

```
def purple():
    # Change the pen color and
    # the turtle to purple
    turtle.color("purple")
```

For the 'p' key

```
def cyan():
    # Change the pen color and
    # the turtle to cyan
    turtle.color("cyan")
```

For the 'c' key

Key Events Example 3/3 – Main Program

```
turtle.shape("turtle")
turtle.speed(0)
turtle.color("purple")
turtle.width(3)

turtle.onkeypress(moveforward, "Up")
turtle.onkeypress(movebackward, "Down")
turtle.onkeypress(rotateleft, "Left")
turtle.onkeypress(rotateright, "Right")
```

Assign the up, down, left and right keys for moving the turtle

```
turtle.onkeypress(orange, "o")
turtle.onkeypress(purple, "p")
turtle.onkeypress(cyan, "c")
```

Assign the 'o', 'p' and 'c' keys for the colour change functions

```
turtle.listen()
```

Ask Python to listen for keyboard events

```
turtle.done()
```

Must have this at the end