John Romero Programming Proverbs

- 4. ‘‘Great tools help make great games. Spend as much time on tools as possible.’’

- John Romero, ‘‘The Early Days of Id Software - John Romero @ WeAreDevelopers Conference 2017’’

PyGame

- PyGame documentation ⟨`http://www.pygame.org/docs`⟩
- can set the window size by:
- ```
  import pygame

  pygame.init ()
  screen = pygame.display.set_mode([width, height])
  ...
  ```
- notice the parameter is a list (or tuple) of two components (x, y)
  ```

PyGame

- you can put this into full screen mode via:

```
from pygame.locals import *

screen = pygame.display.set_mode([width, height], FULLSCREEN)
...
```

- you can toggle this in game via:

```
pygame.display.toggle_fullscreen()
```

Defining colours

- colours can be defined using rgb triples

```
black = (0, 0, 0)
white = (255, 255, 255)
brightred = (255, 0, 0)
brightgreen = (0, 255, 0)
brightblue = (0, 0, 255)
```

- it is worth spending a little time adjusting the colours to match your aesthetic aspirations

Defining colours

```
wood_light = (166, 124, 54)
wood_dark = (76, 47, 0)
blue = (0, 100, 255)

dark_red = (166, 25, 50)
dark_green = (25, 100, 50)
dark_blue = (25, 50, 150)
```

Drawing objects: circle

```
#!/usr/bin/python

import pygame, sys
from pygame.locals import *

dark_blue = (25, 50, 150)
black = (0, 0, 0)

height = 300
width = 400
```

Drawing objects: circle

```
pygame.init()
screen = pygame.display.set_mode([width, height])
pygame.draw.circle(screen, dark_blue, (50, 90), 30, 0)
pygame.display.flip()

while True:
    for event in pygame.event.get ():
        if event.type == pygame.QUIT:
            sys.exit(0)
        if event.type == KEYDOWN:
            if event.key == K_ESCAPE:
                sys.exit(0)
```

Notes for the circle example

- the parameters for `circle` are as follows:
  - `screen` the surface on which to draw the circle
  - `dark_blue` colour of the circle
  - `(50, 90)` x, y coordinate, although remember 0, 0 is top left
  - `30` the radius of the circle
  - `0` thickness of the circle, 0 means fill it completely

- please refer to the PyGame documentation ⟨`http://www.pygame.org/docs/ref/draw.html`⟩ for more examples and details

Keyboard input

- in the last example the code at the end introduced minimal keyboard handling

-
```
while True:
    for event in pygame.event.get ():
        if event.type == pygame.QUIT:
            sys.exit(0)
        if event.type == KEYDOWN:
            if event.key == K_ESCAPE:
                sys.exit(0)
```

Keyboard input

- can test key goes down or up and which key etc
- the function `pygame.event.get()` will return a list of all pending events seen by pygame
- modify the code to print out the event
- what happens if you click the mouse button?

Drawing objects: rectangle

```
#!/usr/bin/python

import pygame, sys
from pygame.locals import *

dark_blue = (25, 50, 150)
black = (0, 0, 0)

height = 300
width = 400
```

Drawing objects: rectangle

```
pygame.init ()
screen = pygame.display.set_mode([width, height])
pygame.draw.rect (screen, dark_blue, (50, 50, 60, 60), 0)
pygame.display.flip()

while True:
    for event in pygame.event.get ():
        if event.type == pygame.QUIT:
            sys.exit(0)
        if event.type == KEYDOWN:
            if event.key == K_ESCAPE:
                sys.exit (0)
```

Drawing objects: rectangle

- the line

- ```
  pygame.draw.rect (screen, dark_blue, (50, 50, 60, 60), 0)
  ```

- means:
  - call the `rect` method and draw on `screen` a rectangle of colour `dark_blue`
  - which has a top left corner of `50, 50`
  - and a bottom right corner of `60, 60`
  - this rectangle will be completely filled (boarder size of `0`)
  ```

Drawing objects: rectangle

- notice that in both the circle and rectangle examples nothing is displayed until you flip the buffer

- ```
  pygame.display.flip()
  ```

- using a common technique of double buffering
  - your application draws everything off screen and then it is flipped onto the screen, giving the illusion everything is drawn at once
  ```

Drawing objects: polygon

```
#!/usr/bin/python

import pygame, sys
from pygame.locals import *

dark_blue = (25, 50, 150)
black = (0, 0, 0)

height = 300
width = 400
```

Drawing objects: polygon

```
pygame.init ()
screen = pygame.display.set_mode([width, height])
pygame.draw.polygon (screen, dark_blue, [[50, 50], [100, 100], [50, 100]], 0)
pygame.display.flip()

while True:
    for event in pygame.event.get ():
        if event.type == pygame.QUIT:
            sys.exit(0)
        if event.type == KEYDOWN:
            if event.key == K_ESCAPE:
                sys.exit (0)
```

Drawing objects: polygon

- the line

```
pygame.draw.polygon (screen, dark_blue, [[50, 50],
                                         [100, 100],
                                         [50, 100]], 0)
```

- means:
  - call the `polygon` method and draw on `screen` a polygon of colour `dark_blue`
  - which has a top left corner of `50, 50`
  - a corner at `100, 100`
  - and a final corner at `50, 100`
  - this polygon will be completely filled (boarder size of `0`)

Drawing objects: polygon

- the list of corners is often called a list of vertices as it also describes the lines (verticies of the polygon)

Tutorial

- write a pygame program to place a circle at any mouse position on the screen
  - left click will drop a circle at the current mouse position