QUEST - GAMING

GAMING MODULES

- Pygame
- Pyglet

Python Pygame

- Game Development Library
- Cross - Platform
- Consists of inbuilt Computer Graphics and Sound Libraries
- Written by - Pete Shinners
- Used to create client-side applications that can be wrapped in a standalone executable game.

Python Pyglet

- Visually rich GUI applications such as games, multimedia, etc.
- Features - Windowing, user interface event handling, Joysticks, OpenGL graphics, loading images, and videos, and playing sounds and music.
- Pyglet is provided under the BSD open-source license, allows you to use it for both commercial and other open-source projects.
- No external dependencies or installation requirements.
- Take advantage of multiple windows and multi-monitor desktops.

INSTALLATION

Pygame Installation

- ➢ Installation of Python IDE such as IDLE, Pycharm, Spyder, etc with the version 3.6.1+.
- ➢ Installation through pip using CMD -
- ➢ Windows : *py -m pip install -U pygame --user*
- ➢ MAC OS X Latest Version : *python3 -m pip install -U pygame --user*
- ➢ Installation through Pycharm IDE - *File > Settings > Project Interpreter > Pygame > Install Package*

INSTALLATION

Pyglet Installation

- ➢ Installation of Python IDE such as IDLE, Pycharm, Spyder, etc with the version 3.6.1+.
- ➢ Installation through CMD - *pip install pyglet*
- ➢ Installation through IDE - *import pyglet*

Pygame Module

- **import pygame** - imports all the functions of pygame and provides access to it's framework
- **pygame.init()** - initialise all the required module of pygame
- **pygame.display.set_mode((width, height)) -** this is used to display a window of the desired size. The return value is a Surface object which is the object where we will perform graphical operations.
- **pygame.event.get()-** Collects the event that are done by the user in the form of surface objects and empty the event queue. If piled up in the event queue, the game will become unresponsive according to the OS.
- **pygame.QUIT -** This is used to terminate the event when we click on the close button at the corner of the window. (X)
- **pygame.display.flip() -** Pygame is double-buffered, so this shifts the buffers. It is essential to call this function in order to make any updates that you make on the game screen to make visible.

Pygame Module

Pygame Surface :

The pygame Surface is used to display any image. The Surface has a pre-defined resolution and pixel format. The Surface color is by default black. Its size is defined by passing the size argument. Surfaces can have the number of extra attributes like **alpha planes, color keys, etc**.

Pygame Blit :

The pygame blit is the process to render the game object onto the surface, and this process is called **blitting**. If we don't render the game objects and run the program, then it will give the black window as an output. Blitting is one of the slowest operations in any game so, we need to be careful to not to blit much onto the screen in every frame.

Syntax : `blit(source,dest,area=None,special_flags=0)`

This function is used to draw one image into another. The draw can be placed with the dest argument. The dest argument can either be a pair of coordinates representing the upper left corner of the source.

Pygame Module

Pygame Clock :

Represented in milliseconds (1/1000 second). Used to track the time. Essential to create motion, sound, or reaction to any game.The clock also provides various functions to help in controlling the game's frame rate.

tick() - tick(framerate = 0) : updates the clock.This method should be called once per frame. It will calculate how many milliseconds have passed since the previous call. The **framerate** argument is optional to pass in the function, and if it is passed as an argument then the function will delay to keep the game running slower than the given ticks per second.

tick_busy_loop()

The tick_busy_loop() is same as the tick(). By calling the **Clock.tick_busy_loop(20)** once per frame, the program will never run at more than 20 frames per second.

get_time()

The get_time() is used to get the previous tick. The number of a millisecond that is passed between the last two calls in Clock.tick()

Pygame Module

Adding Image -

1. Add a blank surface = pygame.Surface((width,height)) //24 bit RGB image with default Black Colour
2. For the transparent initialization of Surface, pass the SRCALPHA argument : pygame.Surface((width,height), pygame.SRCALPHA)
3. Add image to your Window = pygame.image.load(‘directory’)
4. Blit image to your screen = screenName.blit(image,(x,y))

Draw a Rectangle -

1. Pygame uses Rect objects to store and manipulate rectangular areas. A Rect can be formed from a combination of left, top, width, and height values.The **rect()** function is used to perform changes in the position or size of a rectangle. It returns the new copy of the Rect with the affected changes. Original stays the same.
2. The Rect object has various virtual attributes which can be used to move and align the Rect such as x, y, top, left, right, bottom, topleft, bottomleft, topright, bottomright, midtop, midleft, midbottom, midright, center, center_x, center_y, size, width, height, w, h.
3. pygame.draw.rect(screen, (R, G, B), pygame.Rect(x, y, width, height))

Pygame Module

Pygame Keys -

Pygame KEYDOWN and KEYUP detect the event if a key is physically pressed and released. **KEYDOWN** detects the key press and, **KEYUP** detects the key release. Both events (Key press and Key release) have two attributes which are the following:

- **key:** Key is an integer id which represents every key on the keyword.
- **mod:** This is a bitmask of all the modifier keys that were in the pressed state when the event occurred.

```
if event.type in (pygame.KEYDOWN, pygame.KEYUP):

     key_name = pygame.key.name(event.key)

     key_name = key_name.upper()

      if event.type == pygame.KEYDOWN:

          print(u'"{}" key pressed'.format(key_name))

      elif event.type == pygame.KEYUP:

            print(u'"{}" key released'.format(key_name))
```

Pygame Module

Pygame Draw -

Pygame provides geometry functions to draw simple shapes to the surface.Most of the functions accept a width argument to signify the size of the thickness around the edge of the shape. If the width is passed 0, then the shape will be solid(filled).

All the drawing function takes the color argument that can be one of the following formats:

- A pygame.Color objects
- An (RGB) triplet(tuple/list)
- An (RGBA) quadruplet(tuple/list)
- An integer value that has been mapped to the surface's pixel format

Draw a rectangle -

```
pygame.draw.rect(surface, color, rect,width = 0)  // width < 0 : will draw nothing and width > 0 : will draw lines with that thickness
```

Pygame Module

Pygame Draw -

Draw a polygon- Syntax :

```
pygame.draw.polygon(surface, color, points, width=0)
```

Draw an ellipse- Syntax :

```
pygame.draw.ellipse(surface, color, rect, width=0)
```

Draw a straight line- Syntax :

```
pygame.draw.line(surface,color,start_pos,end_pos,width=1)
```

Pygame Module

Pygame Draw -

Draw a Circle- Syntax :

```
pygame.draw.circle(surface, color, center, radius, width=0)
```

Draw an elliptical arc- Syntax :

```
pygame.draw.arc(surface, color, rect, start_angle, stop_angle, width=1)
```

There are three conditions for start_angle and stop_angle parameter:

a. If start_angle < stop_angle then the arc will be drawn in a counter-clock direction from the start_angle to end_angle.
b. If start_angle>stop_angle then tau(tau=2*pi) will be added to the stop angle.
c. If start_angle==stop_angle, nothing will be drawn.

Pygame Module

Pygame Text and Font -

- pygame.font.SysFont() - Load Fonts from System
- pygame.font.Font() - Font Objects are created which render the text to the Screen.
- render(text, antialias(Smoothness), color, background=None) - Draw text on new Surface. Note that it doesn’t render the text on the existing surface.
- set_bold(bool) - True if you want your text to be Bold in Layout.
- font = pygame.font.SysFont("Times new Roman", font_size)
- text = font.render("Hello, Pygame", True, (R, G, B))
- screenName.blit(text,(ScreenWidth - text.get_width()
- all_font = pygame.font.get_fonts() - Returns all the fonts of Pygame
- font = pygame.font.Font(None,size) - Default system Font

Pygame Module

Pygame Sprite and Collision detection

A pygame sprite is a two-dimensional image that is part of the large graphical scene. Usually, a sprite will be some object in the scene.

One of the most advantages of working with sprites is the ability to work with them in groups. We can easily move and draw all the sprites with the one command if they are in the group.

The Sprite module contains the various simple classes to be used within the games. It is optional to use Sprite classes and different group classes when using pygame.

Pygame provides sprites and sprite groups that help for collision detection. Collision detection is the process when two objects on the screen collide each other. For example, if a player is hit by the enemy's bullet, then it may lose a life or, the program need to know when the player touches a coin so that they automatically picked up.

Pygame Sprite and Collision detection

```
import pygame

import sys

class Sprite(pygame.sprite.Sprite):

        def __init__(self, pos):

                 pygame.sprite.Sprite.__init__(self)

                self.image = pygame.Surface([20, 20])

                self.image.fill((255, 0, 255))

                self.rect = self.image.get_rect()

                self.rect.center = pos
```

Pygame Sprite and Collision detection

```
def main():
    pygame.init()
    clock = pygame.time.Clock()
    fps = 50
    bg = [0, 0, 0]
    size =[300, 300]
    screen = pygame.display.set_mode(size)
    player = Sprite([40, 50])
    player.move = [pygame.K_LEFT, pygame.K_RIGHT, pygame.K_UP, pygame.K_DOWN]
    player.vx = 5
    player.vy = 5
    wall = Sprite([100, 60])
    wall_group = pygame.sprite.Group()
    wall_group.add(wall)
    player_group = pygame.sprite.Group()
    player_group.add(player)
```

Pygame Sprite and Collision detection

```
while True:
        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                return False
        key = pygame.key.get_pressed()
        for i in range(2):
            if key[player.move[i]]:
                player.rect.x += player.vx * [-1, 1][i]
        for i in range(2):
             if key[player.move[2:4][i]]:
                player.rect.y += player.vy * [-1, 1][i]
         screen.fill(bg)
          hit = pygame.sprite.spritecollide(player, wall_group, True)
```

Pygame Sprite and Collision detection

```
//Inside while
        if hit:
            # if collision is detected call a function to destroy
              # rect
              player.image.fill((255, 255, 255))
          player_group.draw(screen)
          wall_group.draw(screen)
          pygame.display.update()
          clock.tick(fps)
  pygame.quit()
  sys.exit
if __name__ == '__main__':
        main()
```

PRACTICE HOUR

Share a short essay of your favourite game. This includes -

1. **Why is it your favourite game?**
2. **What is your favourite feature about it?**
3. **The year it’s beta version was released.**
4. **Was this game that drove you to this Game Developing track?**

PRACTICE HOUR

Try -

Collision of 2 Ellipses Sprite with texts on them when they collide as :

Text on the Ellipse :- Shoot - Collided !

Pyglet Module

- Import your gaming module to your source code - import pyglet
- Create a window, gaming space for your game :

  screenName = pyglet.window.Window()
- To begin with, we’ll print a simple text to our Screen - SAMPLE WINDOW

  Text = pyglet.text.Label (“SAMPLE WINDOW”, font_name = ‘xyz’, font_size = xyz, x_coordinate, y_coordinate, anchor_x = ‘alignment’, anchor_y =’alignment’)
- Draw Screen Method :

```
@screenName.event
def on_draw() :
        screenName.clear()
        Text.draw()
```

- Executes the Program - pyglet.app.run()

Pyglet Module

CREATE YOUR GAMING WINDOW

```
import pyglet

gscreen = pyglet.window.Window()

text = pyglet.text.Label(“SAMPLE WINDOW”, font_name = ‘Arial’, font_size = 18, x =
gscreen.width//2, y = gscreen.height//2, anchor_x = ‘centre’, anchor_y = ‘centre’ )

@gscreen.event
def on_draw() :
        gscreen.clear()
        text.draw()

pyglet.app.run()
```

Quest - Gaming

#wepressplay