## Session 6

## Creation of Images and Input Text Box in PyGame

## THE PYGAME-BASED APPROACH RELATED TO OOPS

## IMAGES AND INPUT TEXT BOX

1. **HOW TO ROTATE AND SCALE IMAGES USING PYGAME**

## 1. The Pygame-based approach relates to OOP's.

**By adding the attributes associated with an object into the class we have effectively achieved encapsulation** This solves one of the major issues that we faced in the practice problem which is to keep track of all the variables associated with individual circles and draw the circles themselves.

import pygame  
from random import randint  
  
pygame.init()  
screen = pygame.display.set\_mode([500, 500]) *# creates a screen with the said size*  
clock = pygame.time.Clock()  
*# Creating the circle class*  
**class** Circle():  
 **def** \_\_init\_\_(self):  
 self.x = randint(-50,500)   
 self.y = randint(0,500)  
 self.r = randint(10,50)  
 self.color = (randint(0,255),randint(0,255),randint(0,255)) *# (randint(0,255),randint(0,255),b)*  
   
 **def** draw(self):  
 pygame.draw.circle(screen, self.color, (self.x, self.y), self.r)  
   
*# Creating circle objects using the Circle class*  
c1 = Circle()  
c2 = Circle()  
c3 = Circle()  
c4 = Circle()  
c5 = Circle()  
c6 = Circle()  
c7 = Circle()  
c8 = Circle()  
c9 = Circle()  
c10 = Circle()  
  
run = True  
**while** run:  
 **for** event **in** pygame. event.get():  
 **if** event.type == pygame.QUIT:  
 run = False  
   
 screen.fill((255,255,255))  
   
 *# Creating individual circle objects using the draw method*  
 c1.draw()  
 c2.draw()  
 c3.draw()  
 c4.draw()  
 c5.draw()  
 c6.draw()  
 c7.draw()  
 c8.draw()  
 c9.draw()  
 c10.draw()  
 pygame. display.flip()  
   
 clock.tick(30)  
   
   
pygame.quit()

The next step is to identify the methods associated with the object. Methods are functionalities that are performed by the object itself. In the case of the example that we are doing, the method that a circle object can have is to move around. So calling the move methods will make the ball move in a certain direction with a certain speed. We will have to define the x and y speed attributes which can be used by the move() method to move the ball around by manipulating the x and y coordinate.

**Attributes to be defined**

1. x speed
2. y speed **Methods to be defined**
3. move()

import pygame  
from random import randint  
  
pygame.init()  
  
screen = pygame.display.set\_mode([500, 500]) *# creates a screen with the said size*  
  
clock = pygame.time.Clock()  
  
  
*# Creating the circle class*  
**class** Circle():  
 **def** \_\_init\_\_(self):  
 self.x = randint(0,500)  
 self.y = randint(0,500)  
 self.r = randint(10,50)  
 self.color = (randint(0,255),randint(0,255),randint(0,255))  
 self.x\_speed = randint(-2,2)  
 self.y\_speed = randint(-2,2)  
   
 **def** draw(self):  
 pygame.draw.circle(screen, self.color, (self.x, self.y), self.r)  
   
 **def** move(self):  
 self.x = self.x+self.x\_speed  
 self.y = self.y+self.y\_speed   
   
   
*# Creating circle objects using the Circle class*  
c1 = Circle()  
c2 = Circle()  
c3 = Circle()  
c4 = Circle()  
c5 = Circle()  
  
  
run = True  
**while** run:  
 **for** event **in** pygame. event.get():  
 **if** event.type == pygame.QUIT:  
 run = False  
   
 screen.fill((255,255,255))  
   
   
   
 *# Creating individual circle objects using the draw method*  
 c1.draw()  
 c2.draw()  
 c3.draw()  
 c4.draw()  
 c5.draw()  
   
  
   
 pygame. display.flip()  
   
 clock.tick(30)  
   
   
pygame.quit()

**Running the above code won't make the ball move around as we have not called the move method on individual circle objects**

import pygame  
from random import randint  
  
pygame.init()  
  
screen = pygame.display.set\_mode([500, 500]) *# creates a screen with the said size*  
  
clock = pygame.time.Clock()  
  
  
*# Creating the circle class*  
**class** Circle():  
 **def** \_\_init\_\_(self):  
 self.x = randint(0,500)  
 self.y = randint(0,500)  
 self.r = randint(10,50)  
 self.color = (randint(0,255),randint(0,255),randint(0,255))  
 self.x\_speed = randint(-2,2)  
 self.y\_speed = randint(-2,2)  
   
 **def** draw(self):  
 pygame.draw.circle(screen, self.color, (self.x, self.y), self.r)  
   
 **def** move(self):  
 self.x = self.x+self.x\_speed  
 self.y = self.y+self.y\_speed   
   
   
*# Creating circle objects using the Circle class*  
c1 = Circle()  
c2 = Circle()  
c3 = Circle()  
c4 = Circle()  
c5 = Circle()  
  
  
run = True  
**while** run:  
 **for** event **in** pygame. event.get():  
 **if** event.type == pygame.QUIT:  
 run = False  
   
 screen.fill((255,255,255))  
   
   
   
 *# Creating individual circle objects using the draw method*  
 c1.draw()  
 c2.draw()  
 c3.draw()  
 c4.draw()  
 c5.draw()  
   
 *# Moving the circle objects*  
 c1.move()  
 c2.move()  
 c3.move()  
 c4.move()  
 c5.move()  
   
   
   
 pygame. display.flip()  
   
 clock.tick(30)  
   
   
pygame.quit()

**2. Images And Input Text Box**

How to create a text input box with Pygame?

Pygame can be used to create a text input box which will be explained step by step further in this article.

## Function used

import pygame  
import sys  
   
   
*# pygame.init() will initialize all*  
*# imported module*  
pygame.init()  
   
clock = pygame.time.Clock()  
   
*# It will display on the screen*  
screen = pygame.display.set\_mode([600, 500])  
   
*# Basic font for user-typed*  
base\_font = pygame.font.Font(None, 32)  
user\_text = ''  
   
*# create rectangle*  
input\_rect = pygame.Rect(200, 200, 140, 32)  
   
*# color\_active stores color(lightskyblue3) which*  
*# gets active when the input box is clicked by the user*  
color\_active = pygame.Color('lightskyblue3')  
   
*# color\_passive store color(chartreuse4) which is*  
*# color of the input box.*  
color\_passive = pygame.Color('chartreuse4')  
color = color\_passive  
   
active = False  
   
**while** True:  
 **for** the event **in** pygame. event.get():  
   
 *# If the user types QUIT then the screen will close*  
 **if** event.type == pygame.QUIT:  
 pygame.quit()  
 sys.exit()  
   
 **if** event.type == pygame.MOUSEBUTTONDOWN:  
 **if** input\_rect.collidepoint(event.pos):  
 active = True  
 **else**:  
 active = False  
   
 **if** event.type == pygame.KEYDOWN:  
   
 *# Check for backspace*  
 **if** event.key == pygame.K\_BACKSPACE:  
   
 *# get text input from 0 to -1 i.e. end.*  
 user\_text = user\_text[:-1]  
   
 *# Unicode standard is used for string*  
 *# formation*  
 **else**:  
 user\_text += event.unicode  
   
 *# It will set the background color of the screen*  
 screen.fill((255, 255, 255))  
   
 **if** active:  
 color = color\_active  
 **else**:  
 color = color\_passive  
   
 *# draw a rectangle and argument passed which should*  
 *# be on screen*  
 pygame.draw.rect(screen, color, input\_rect)  
   
 text\_surface = base\_font.render(user\_text, True, (255, 255, 255))  
   
 *# render at position stated in arguments*  
 screen.blit(text\_surface, (input\_rect.x+5, input\_rect.y+5))  
   
 *# Set the width of a text field so that text cannot get*  
 *# outside of the user's text input*  
 input\_rect.w = max(100, text\_surface.get\_width()+10)  
   
 *# display.flip() will update only a portion of the*  
 *# screen to updated, not full area*  
 pygame. display.flip()  
   
 *# clock.tick(60) means that for every second at most*  
 *# 60 frames should be passed.*  
 clock.tick(60)

## 3. How to display images with PyGame?

Here we are first importing the required library and then setting the width and height of the image then creating the display surface for that size then giving the path of the required image in the image.load() function and then finally iterate over the list of event objects.

*# Importing required library*  
import pygame  
  
*# activate the pygame library.*  
pygame.init()  
X = 600  
Y = 600  
  
*# create the display surface object*  
*# of specific dimension..e(X, Y).*  
scrn = pygame.display.set\_mode((X, Y))  
  
*# Set the pygame window name*  
pygame.display.set\_caption('image')  
  
*# create a surface object, image is drawn on it.*  
imp = pygame.image.load("PATH\\robo1.jpg").convert()  
  
*# Using blit to copy content from one surface to another*  
scrn.blit(imp, (0, 0))  
  
*# Paint screen one time*  
pygame. display.flip()  
status = True  
**while** (status):  
  
*# iterate over the list of Event objects*  
*# that was returned by the pygame. event.get() method.*  
**for** i **in** pygame. event.get():  
  
*# If the event object type is QUIT*  
*# then quitting the pygame*  
*# and program both.*  
**if** i.type == pygame.QUIT:  
status = False  
  
*# deactivates the pygame library*  
pygame.quit()

## How to Rotate and Scale images using PyGame?

## Rotating image

To rotate the image we use the pygame. transform. rotate(image, degree) method where we pass the image that we are going to rotate and the degree by which rotation is to be done.

*# Import pygame*  
import pygame  
  
*# Initialise pygame*  
pygame.init()  
  
*# Set window size*  
size = width,height = 600, 600  
screen = pygame.display.set\_mode(size)  
  
*# Clock*  
clock = pygame.time.Clock()  
  
*# Load image*  
image = pygame.image.load("PATH\\robo1.jpg")  
  
*# Set the size for the image*  
DEFAULT\_IMAGE\_SIZE = (200, 200)  
  
*# Rotate the image by any degree*  
image = pygame.transform.rotate(image, 180)  
  
*# Set a default position*  
DEFAULT\_IMAGE\_POSITION = (200,200)  
  
*# Prepare loop condition*  
running = False  
  
*# Event loop*  
**while** **not** running:  
  
*# Close window event*  
**for** the event **in** pygame. event.get():  
**if** event.type == pygame.QUIT:  
running = True  
  
*# Background Color*  
screen.fill((0, 0, 0))  
  
*# Show the image*  
screen.blit(image, DEFAULT\_IMAGE\_POSITION)  
  
*# Part of the event loop*  
pygame. display.flip()  
clock.tick(30)

## Rotating and Scaling the Image

Let us see how to perform the Scaling and Rotation of an image given. We will set the default image size that is agreeable and the default image position where we want to see our image on the window screen. The same methods that are explained above will be used for scaling and rotation of the image.

import pygame

import sys

pygame.init()

clock = pygame. time.Clock()

screen = pygame. display.set\_mode([600, 500])

base\_font = pygame. font.Font(None, 32)

user\_text = ''

input\_rect = pygame.Rect(200, 200, 200, 32)

ACTIVE\_COLOR = pygame.Color('lightskyblue3')

PASSIVE\_COLOR = pygame.Color('chartreuse4')

color = PASSIVE\_COLOR

active = False

while True:

for event in pygame. event.get():

if event.type == pygame.QUIT:

pygame.quit()

sys.exit()

if event.type == pygame.MOUSEBUTTONDOWN:

if input\_rect.collidepoint(event.pos):

active = not active

else:

active = False

if event.type == pygame.KEYDOWN:

if event.key == pygame.K\_BACKSPACE:

user\_text = user\_text[:-1]

else:

user\_text += event.unicode

screen.fill((255, 255, 255))

# Draw input field border

pygame. draw.rect(screen, color, input\_rect, 2)

if active:

color = ACTIVE\_COLOR

else:

color = PASSIVE\_COLOR

# Render text once outside the loop

text\_surface = base\_font.render(user\_text, True, (255, 255, 255))

# Handle longer text

if text\_surface.get\_width() > input\_rect.width - 10:

offset = text\_surface.get\_width() - (input\_rect.width - 10)

else:

offset = 0

# Render the clipped text

screen.blit(text\_surface, (input\_rect.x + 5 - offset, input\_rect.y + 5))

pygame. display.flip()

clock.tick(60)