## Session 8

## Game Development with OOPs Concepts

## Pygame with OOPs: BULLET GUN GAME USING PYGAME

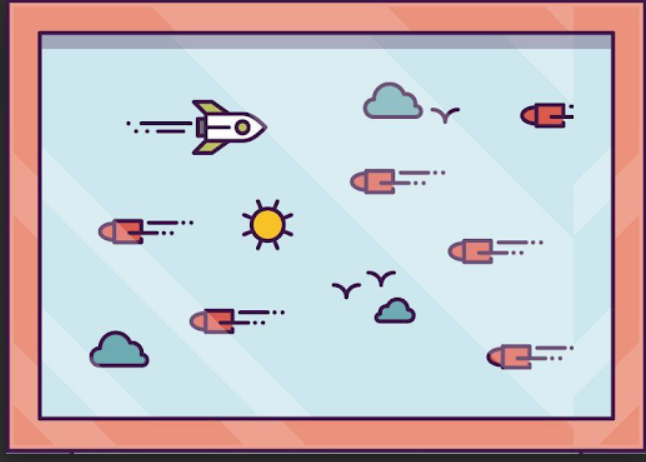
The beauty of Pygame OOP lies in how it makes complex programming tasks simpler and more manageable. By organizing code into objects, you create reusable components that can interact with each other – a must-have in the world of game development!

In this session, we are going to work on optimizing our code to make it more readable and scalable. To do this we are going to use Object Orientated Programming.

## Initializing Pygame and defining colors:

The code starts by importing the pygame module and initializing it with pygame. init(). It then defines some color constants that will be used later in the code.

import pygame  
import random  
  
*# Define some colors*  
BLACK = ( 0, 0, 0)  
WHITE = ( 255, 255, 255)  
RED = ( 255, 0, 0)  
BLUE = ( 0, 0, 255)



## Create the block class:

A class to represent the block as a to-fill color

*# --- Classes*  
  
**class** Block(pygame.sprite.Sprite):  
 *""" This class represents the block. """*  
 **def** \_\_init\_\_(self, color):  
 *# Call the parent class (Sprite) constructor*  
 super().\_\_init\_\_()  
  
 self.image = pygame.Surface([20, 15])  
 self. image.fill(color)  
  
 self.rect = self.image.get\_rect()

## Create the Player class:

A class to represent the player as a red block on the screen.

**class** Player(pygame.sprite.Sprite):  
 *""" This class represents the Player. """*  
  
 **def** \_\_init\_\_(self):  
 *""" Set up the player on creation. """*  
 *# Call the parent class (Sprite) constructor*  
 super().\_\_init\_\_()  
  
 self.image = pygame.Surface([20, 20])  
 self. image.fill(RED)  
  
 self.rect = self.image.get\_rect()  
  
 **def** update(self):  
 *""" Update the player's position. """*  
 *# Get the current mouse position. This returns the position*  
 *# as a list of two numbers.*  
 pos = pygame.mouse.get\_pos()  
  
 *# Set the player x position to the mouse x position*  
 self.rect.x = pos[0]

## Create the Bullet class:

A class to represent the bullets as black lines on the screen.

**class** Bullet(pygame.sprite.Sprite):  
 *""" This class represents the bullet. """*  
 **def** \_\_init\_\_(self):  
 *# Call the parent class (Sprite) constructor*  
 super().\_\_init\_\_()  
  
 self.image = pygame.Surface([4, 10])  
 self. image.fill(BLACK)  
  
 self.rect = self.image.get\_rect()  
  
 **def** update(self):  
 *""" Move the bullet. """*  
 self. rect.y -= 3

## Create a Pygame window:

A window is created with a width of 700 and a height of 400 pixels.

*# --- Create the window*  
  
*# Initialize Pygame*  
pygame.init()  
  
*# Set the height and width of the screen*  
screen\_width = 700  
screen\_height = 400  
screen = pygame. display.set\_mode([screen\_width, screen\_height])

## Create Sprite lists:

The all\_sprites\_list, block\_list, and bullet\_list are created to keep track of all the sprites in the game, blocks in the game, and bullets in the game respectively.

*# --- Sprite lists*  
  
*# This is a list of every sprite. All blocks and the player block as well.*  
all\_sprites\_list = pygame.sprite.Group()  
  
*# List of each block in the game*  
block\_list = pygame.sprite.Group()  
  
*# List of each bullet*  
bullet\_list = pygame. sprite.Group()

## Create the sprites:

50 blocks are created with random positions, and the player is created at the bottom center of the screen.  
*# --- Create the sprites*  
  
**for** i **in** range(50):  
 *# This represents a block*  
 block = Block(BLUE)  
  
 *# Set a random location for the block*  
 block.rect.x = random.randrange(screen\_width)  
 block.rect.y = random.randrange(350)  
  
 *# Add the block to the list of objects*  
 block\_list.add(block)  
 all\_sprites\_list.add(block)  
  
*# Create a red player block*  
player = Player()  
all\_sprites\_list.add(player)  
  
*#Loop until the user clicks the close button.*  
done = False  
  
*# Used to manage how fast the screen updates*  
clock = pygame.time.Clock()  
  
score = 0  
player. rect.y = 370

## Event processing:

The game checks for the QUIT event (the user closes the window) and the MOUSEBUTTONDOWN event (the user clicks the mouse button).

*# -------- Main Program Loop -----------*  
**while** **not** done:  
 *# --- Event Processing*  
 **for** event **in** pygame. event.get():  
 **if** event.type == pygame.QUIT:  
 done = True  
  
 **elif** event.type == pygame.MOUSEBUTTONDOWN:  
 *# Fire a bullet if the user clicks the mouse button*  
 bullet = Bullet()  
 *# Set the bullet so it is where the player is*  
 bullet.rect.x = player.rect.x  
 bullet.rect.y = player.rect.y  
 *# Add the bullet to the lists*  
 all\_sprites\_list.add(bullet)  
 bullet\_list.add(bullet)

## Game logic:

The bullets are updated by moving them upwards and checking if they hit a block. If they hit a block, they are removed from the bullet\_list and all\_sprites\_list, and the score is incremented. If they fly off the screen, they are removed from the lists.

*# --- Game logic*  
  
 *# Call the update() method on all the sprites*  
all\_sprites\_list.update()  
  
 *# Calculate mechanics for each bullet*  
**for** bullet **in** bullet\_list:  
  
 *# See if it hit a block*  
 block\_hit\_list = pygame.sprite.spritecollide(bullet, block\_list, True)  
  
 *# For each block hit, remove the bullet and add to the score*  
 **for** block **in** block\_hit\_list:  
 bullet\_list.remove(bullet)  
 all\_sprites\_list.remove(bullet)  
 score += 1  
 print(score)  
  
 *# Remove the bullet if it flies up off the screen*  
 **if** bullet. rect.y < -10:  
 bullet\_list.remove(bullet)  
 all\_sprites\_list.remove(bullet)

## Draw the screen:

The screen is cleared, all sprites are drawn, and the screen is updated. Limit the frames per second to 60.

*# --- Draw a frame*  
  
 *# Clear the screen*  
screen.fill(WHITE)  
  
 *# Draw the score on the screen*  
font = pygame.font.Font(None, 36)  
score\_surface = font.render("Score: {}".format(score), True, BLACK)  
screen.blit(score\_surface, (10, 10))  
  
  
 *# Draw all the spites*  
all\_sprites\_list.draw(screen)  
  
 *# Go ahead and update the screen with what we've drawn.*  
pygame. display.flip()  
  
 *# --- Limit to 20 frames per second*  
clock.tick(60)  
  
pygame.quit()

## Final Complete Code:

import pygame  
import random  
  
*# Define some colors*  
BLACK = ( 0, 0, 0)  
WHITE = ( 255, 255, 255)  
RED = ( 255, 0, 0)  
BLUE = ( 0, 0, 255)

*# --- Classes*  
  
**class** Block(pygame.sprite.Sprite):  
 *""" This class represents the block. """*  
 **def** \_\_init\_\_(self, color):  
 *# Call the parent class (Sprite) constructor*  
 super().\_\_init\_\_()  
  
 self.image = pygame.Surface([20, 15])  
 self. image.fill(color)  
  
 self.rect = self.image.get\_rect()

**class** Player(pygame.sprite.Sprite):  
 *""" This class represents the Player. """*  
  
 **def** \_\_init\_\_(self):  
 *""" Set up the player on creation. """*  
 *# Call the parent class (Sprite) constructor*  
 super().\_\_init\_\_()  
  
 self.image = pygame.Surface([20, 20])  
 self. image.fill(RED)  
  
 self.rect = self.image.get\_rect()  
 **def** update(self):  
 *""" Update the player's position. """*  
 *# Get the current mouse position. This returns the position*  
 *# as a list of two numbers.*  
 pos = pygame.mouse.get\_pos()  
  
 *# Set the player x position to the mouse x position*  
 self.rect.x = pos[0]

**class** Bullet(pygame.sprite.Sprite):  
 *""" This class represents the bullet. """*  
 **def** \_\_init\_\_(self):  
 *# Call the parent class (Sprite) constructor*  
 super().\_\_init\_\_()  
  
 self.image = pygame.Surface([4, 10])  
 self. image.fill(BLACK)  
  
 self.rect = self.image.get\_rect()  
  
 **def** update(self):  
 *""" Move the bullet. """*  
 self. rect.y -= 3

*# --- Create the window*  
  
*# Initialize Pygame*  
pygame.init()  
  
*# Set the height and width of the screen*  
screen\_width = 700  
screen\_height = 400  
screen = pygame. display.set\_mode([screen\_width, screen\_height])

*# --- Sprite lists*  
  
*# This is a list of every sprite. All blocks and the player block as well.*  
all\_sprites\_list = pygame.sprite.Group()  
  
*# List of each block in the game*  
block\_list = pygame.sprite.Group()  
  
*# List of each bullet*  
bullet\_list = pygame. sprite.Group()

*# --- Create the sprites*  
  
**for** i **in** range(50):  
 *# This represents a block*  
 block = Block(BLUE)  
  
 *# Set a random location for the block*  
 block.rect.x = random.randrange(screen\_width)  
 block.rect.y = random.randrange(350)  
  
 *# Add the block to the list of objects*  
 block\_list.add(block)  
 all\_sprites\_list.add(block)  
  
*# Create a red player block*  
player = Player()  
all\_sprites\_list.add(player)  
  
*#Loop until the user clicks the close button.*  
done = False  
  
*# Used to manage how fast the screen updates*  
clock = pygame.time.Clock()  
  
score = 0  
player. rect.y = 370

*# --- Event Handling*  
*# -------- Main Program Loop -----------*  
**while** **not** done:  
 *# --- Event Processing*  
 **for** event **in** pygame. event.get():  
 **if** event.type == pygame.QUIT:  
 done = True  
  
 **elif** event.type == pygame.MOUSEBUTTONDOWN:  
 *# Fire a bullet if the user clicks the mouse button*  
 bullet = Bullet()  
 *# Set the bullet so it is where the player is*  
 bullet.rect.x = player.rect.x  
 bullet.rect.y = player.rect.y  
 *# Add the bullet to the lists*  
 all\_sprites\_list.add(bullet)  
 bullet\_list.add(bullet)

*# --- Game logic*  
  
 *# Call the update() method on all the sprites*  
all\_sprites\_list.update()  
  
 *# Calculate mechanics for each bullet*  
**for** bullet **in** bullet\_list:  
  
 *# See if it hit a block*  
 block\_hit\_list = pygame.sprite.spritecollide(bullet, block\_list, True)  
  
 *# For each block hit, remove the bullet and add to the score*  
 **for** block **in** block\_hit\_list:  
 bullet\_list.remove(bullet)  
 all\_sprites\_list.remove(bullet)  
 score += 1  
 print(score)  
  
 *# Remove the bullet if it flies up off the screen*  
 **if** bullet. rect.y < -10:  
 bullet\_list.remove(bullet)  
 all\_sprites\_list.remove(bullet)

*# --- Draw a frame*  
  
 *# Clear the screen*  
screen.fill(WHITE)  
  
 *# Draw the score on the screen*  
font = pygame.font.Font(None, 36)  
score\_surface = font.render("Score: {}".format(score), True, BLACK)  
screen.blit(score\_surface, (10, 10))  
  
  
 *# Draw all the spites*  
all\_sprites\_list.draw(screen)  
  
 *# Go ahead and update the screen with what we've drawn.*  
pygame. display.flip()  
  
 *# --- Limit to 20 frames per second*  
clock.tick(60)  
  
pygame.quit()