

**PROJECT ON COMPUTER GRAPHICS**

**NAME OF PROJECT:** Slither Gotham.

**SUBMITTED BY:** 16531

**SUBMITTED TO:** Naveen Chauhan Sir.

**DESCRIPTION:** The given game is a single player game. The game is written in a python script using the *pygame* module in python. The game makes use of event handling ability of the python which makes it more efficiently for *graphic based programming.*

**PLOT:** In this game the player has to wear the mantle of Batman ride the batmoblie to save the city of Gotham by collecting as many as bombs possible.

**Instruction:** Use the arrow keys to move your batmobile and the length of your batmobile will keep on increasing avoid collision with the walls and yourself.

***Use of Computer Graphics:*** The game is written in python with importing the pygame module which give a advantage of controlling the game in basic KEYUP and KEYDOWN *events .*

**Co-ordinate System:** The randomly appearing bombs use a *randrange* function which picks the random co-ordinates and when the co-ordinates of batmobile and the bomb match we replace the bombs co-ordinates and increment the score. Also on each direction keydown we increment the respective co-ordinates to make it work.

*Usage of co-ordinate system is shown below:*

for event in pygame.event.get():

gameDisplay.fill(dark)

#gameDisplay.blit(boundry, [0, 0])

print(event)

if(event.type==pygame.QUIT):

gameExit= True

gameOver = False

intro = False

if (event.type==pygame.KEYDOWN):

if(event.key==pygame.K\_LEFT):

leadXMove = -blockSize

leadYMove = 0

direction= 'left'

elif(event.key==pygame.K\_RIGHT):

leadXMove =blockSize

leadYMove =0

direction ='right'

elif(event.key==pygame.K\_UP):

leadYMove = -blockSize

leadXMove =0

direction='up'

elif(event.key==pygame.K\_DOWN):

leadYMove = blockSize

leadXMove =0

direction ='down'

**Frames Per Second:** In order to make the movement of our batmobile visible we have to fix the frame per second in my project it is set to 15 . Which is multipled with 1.5 on each level update.

**Pixel-Phasing –** The collision between the batmobile and the bomb had to be a perfect collision under a given range of pixel for that we used the pixel phasing via rounding-off the approximate values of in the multiples of 10 as the batmobile is always incremented in 10 units.

**Image Bliting:** The images such as in the starting menu screen is blited with the help of pre defined function such as *pygame.display.blit(“imag.png”,[x\_co-ordinate ,y\_-co-ordinate]).*

def menuScreen():

intro = True

while intro:

gameDisplay.blit(quoteImage,[0,0])

messageOnScreen("Slither Gotham",purple,100,180,40)

messageOnScreen("DESCRIPTION : ",red,100,230,20)

messageOnScreen(" Help The Batman save Gotham.",discriptionColor, 100,250,15 )

messageOnScreen(" Ride the Batmobile & collect as many as bombs you can in order to ",discriptionColor,100,270,15)

messageOnScreen(" save the Gotham City. ", discriptionColor, 100,290,15)

messageOnScreen("press any key to start or I for info",green,100, 350 , 20)

pygame.display.update()

for event in pygame.event.get():

if event.type == pygame.KEYDOWN and event.key != pygame.K\_i:

time.sleep(1)

intro = False

elif event.type == pygame.KEYDOWN and event.key == pygame.K\_i :

information()

intro = False

**Rotation**: On every keydown event the head of our batmobile id rotated to 90 degree clock wise and anti-clock wise .

def batMobile ( bombList ,blockSize,direction):

if direction == 'right':

head = pygame.transform.rotate(batImage,270)

if direction == 'left':

head = pygame.transform.rotate(batImage,90)

if direction == 'up':

head = pygame.transform.rotate(batImage, 0)

if direction == 'down':

head = pygame.transform.rotate(batImage, 180)

**Core Logic**: The core of the game revolves around the co-ordinate matching and the increment in length of the batmobile is stored as a list of tuple of co-ordinates. Which are checked every time and upon coincidence the game shuts down.

batHead =[]

batHead.append(leadX)

batHead.append(leadY)

bombList.append(batHead)

if(len(bombList)> bombSize):

del bombList[0]

gameDisplay.blit(head,(bombList[-1][0],bombList[-1][1]))

for xny in bombList[:-1]:

pygame.draw.rect(gameDisplay,backofBatmobile , [xny[0] ,xny[1],blockSize,blockSize])