In this project, we will see how to create a racing car game in Python using Pygame. In this game, we will have functionality like driving, obstacle crashing, speed increment when levels are passed, pause, countdown, scoreboard, and Instruction manual screen.

Required Modules

Here we are going to import Pygame module, Time module and Random module.We are also going to see what this modules are and what properties do they have.

PYGAME MODULE-Pygame includes a higher level sprite module to help organize games. The sprite module includes several classes that help manage details found in almost all games types. The Sprite classes are a bit more advanced than the regular pygame modules, and need more understanding to be properly used.

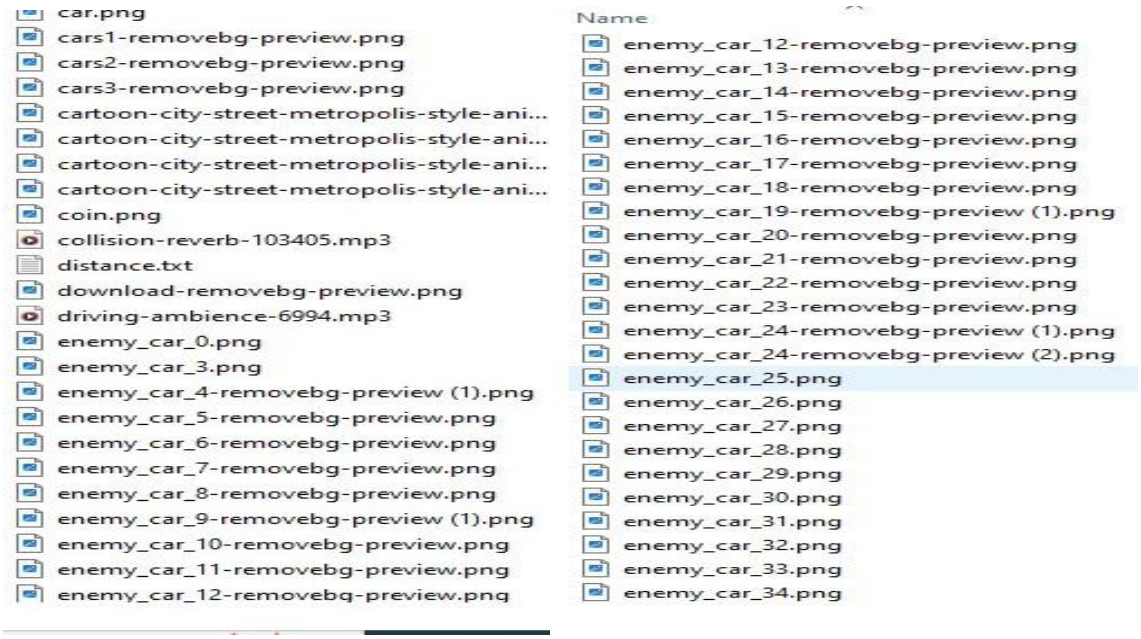
TIME MODULE-The time module is a standard Python module that offers functions for accessing and converting time. To use the module's functionalities, we must first import the Python time module. The beginning of time is measured from 1 January, 12:00 a.m., 1970, and this time is referred to as the "epoch" in Python.

RANDOM MODULE-The Python Random module is a built-in module for generating random integers in Python. These numbers occur randomly and does not follow any rules or instructions. We can therefore use this module to generate random numbers, display a random item for a list or string, and so on.

We will install necessary packages by running the following lines in the command prompt and import them to our game.py file:

```
pip install pygame
pip install time
pip install random
```

Before going any further you will need some assets to make the User Interface.



STEP-1: IMPORT ALL THE MODULES

```
#Importing all Modules

import pygame
import time
import random
import pygame.freetype
```

STEP-2:INITIALIZE PYGAME AND SPECIFY ALL THE COLOUR CODES

After importing the Pygame, now we need to initialize the Pygame so as to use all of the cool methods and all of the other stuff that is inside the Pygame. To initialize the Pygame type the following code and set all color values.

```
# Initializing Pygame Module

pygame.init()

#Colour Codes

black=(0,0,0)
bright_black=(70,70,70)
white=(255,255,255)
red=(255,50,50)
green=(0,200,0)
blue = (0, 0, 200)
bright_red = (255, 0, 0)
bright_green = (0, 255, 0)
bright_blue = (0, 0, 255)
yellow_ochre=(200, 174, 114)
yellow=(255,255,0)
dark_green=(80,125,42)
skyblue=(100, 100, 255)
```

This will initialize and import all modules from Pygame. This line is very important so whenever from now if you are going to create a game with Pygame. Make sure you have added this line otherwise Pygame code is not going to work. Thus, if you want that your Pygame should work effectively without giving any error so you should never forget these lines.

STEP-3:SETTING THE SCREEN

Now, after initializing the Pygame, let's move out to make our Game window where we can play our game. Pygame makes it really, really very easy. It's just one line of code. For creating the Game window just type the following line after the previous code.

```
#Setting the screen

display_width=800
display_hieght=600
screen=pygame.display.set_mode((display_width,display_hieght))
```

What we have done is we created a screen variable in which we accessed the pygame's display.set_mode() function. That's why we typed pygame.

STEP-4:SETTING THE TITLE

In the previous section, we have created our game window, but it was not personalized because you will notice that it says Pygame window and a peculiar logo. So, in this section, we are going to learn how to change the Title logo and background color of our game window.Changing the title in Pygame is very easy, you just need to type the following lines of code.

```
# Setting a Title for the window

pygame.display.set_caption("Racing Beast!!!")
```

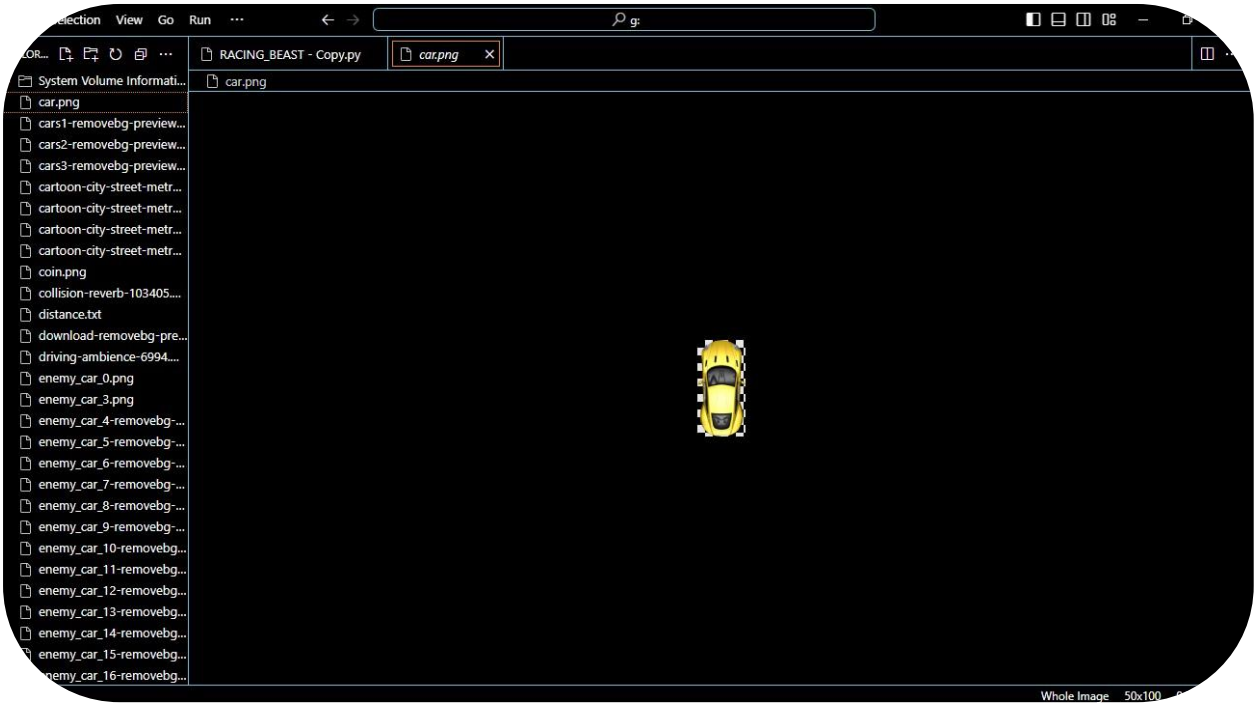
What the above code will do is it will change the title of our game window from 'Pygame window' to 'Racing Beast' obviously you can use your desired name but I used Racing Beast. Since we are dealing with our display that's why we used 'pygame.display' and 'set_caption' is a method to change the caption or the title of our window.

STEP-5:SETTING THE LOGO

Now we will change that weird logo to something related to the car. As I have said in the starting to collect all the stuff related to the game in one folder so it would be easy for you.For changing the logo we will first create a variable logo in which we will load our image and then with the help of the 'set_icon' method we'll change the logo. So our code will look like this.

```
#Setting a Logo for the window

logo = pygame.image.load('car.png')
pygame.display.set_icon(logo)
```

What we did is we created a variable logo in which we loaded the image that we are using for the icon for our game through the 'pygame.image.load()' method it is used to load image from file or file-like object. Then with the help of the 'pygame.display.set_icon()' function, we wrote a variable logo in which our image is loaded. Now run the code and check if everything is okay. See the image below.



STEP-6:ADDING IMAGES

Till now we have customized our game window now it's time to add images or sprites to our game. Pygame makes it very easy we'll load our image as previously we have done it with 'pygame.image.load()'.

After loading the sprites or images we have to display them on our game window for this we'll use the 'blit function' – this is used to display images on our window.But you might have one question in your mind that where will my image will display on our game window. So it's our choice where we want our images to be displayed. We'll display images with their x and y coordinates.

If you have studied 'coordinate geometry' in your school then you must be familiar with the x and y coordinates. If you don't know I'll give a short idea about it.Every point has some X and Y coordinates which we represent with the number and the origin is (0,0). Here first 0 is the value of X coordinate and the second 0 is the value of Y coordinate.

How these values are changing :-

You will notice that when we go from left to right our X increases and when we go from up to bottom our Y increases.So, we conclude that our X increases when we go from left to right and decreases from when we go from right to left.our Y increases when we go from up to bottom and decreases from when we go from bottom to up.X and Y can also be negative. So, let's use this knowledge in our game. I hope you understood this clearly. Now let's code to give our background an image.

```
#Loading all the image assets

carimg=pygame.image.load("car.png")
background=pygame.image.load("Screenshot (143) - Copy - Copy.png")
intro_background=pygame.image.load("intro_background2.jpg")
instruction_background=pygame.image.load("cartoon-city-street-metropolis-style-animation-3d-pour-enfants_720722-6853.jpg")
paused_background=pygame.image.load("cartoon-city-street-metropolis-style-animation-3d-pour-enfants_720722-6896.jpg")
Crash=pygame.image.load("Screenshot_142_-removebg-preview.png")
```

I have said earlier that whatever we want to use till the end of the game we have to do it in a while loop that's why we used the 'screen.blit' method. In the while loop, 'blit' is used to display the image and 'screen' our pygame surface or window.

'screen.blit()' takes two things as a parameter which comes inside the parenthesis. First is the object we want to display and second (x, y)

coordinates means where we want to display our image in the game window.

So here we have 'screen.blit(background,(0,0))' background which is our background image, and (0, 0) is a place where we want to place our image. Why we used (0, 0) as coordinates because our background image is of (800,600) pixels which will cover our whole game window.

STEP-8:SETTING UP A CLOCK

Now using time module we can set up a clock. clock() method of time module in Python is used to get the current processor time as a floating point number expressed in seconds. As, Most of the functions defined in time module call corresponding C library function. time. clock() method also call C library function of the same name to get the result.

```
# Setting a Clock

clock=pygame.time.Clock()
```

STEP-9:LOADING ALL THE SONGS

We can load song audio using another pygame feature pygame.mixer.music.load( )

```
# Loading all the songs

intro_song=pygame.mixer.music.load("Grand-Theft-Auto-San-Andreas-Theme-Song.mp3")
pygame.mixer.music.play()
Sound_crash=pygame.mixer.Sound("Hammer Smashed Face")
sound_crash=pygame.mixer.Sound("collision-reverb-103405.mp3")
countdown_beep=pygame.mixer.Sound("robotic-countdown-43935.mp3")
mouseclick_beep=pygame.mixer.Sound("interface-124464.mp3")
```

This will load a music filename/file object and prepare it for playback. If a music stream is already playing it will be stopped. This does not start the music playing. If you are loading from a file object, the namehint parameter can be used to specify the type of music data in the object.

STEP-10:THE TIMER FUNCTION

The function will display the elapsed time on the screen and update it every second. The user can exit the stopwatch by closing the window. This function demonstrates how to create a stopwatch in Python using Pygame library.

```
def timer():

    font=pygame.freetype.SysFont(None, 34)
    ticks=pygame.time.get_ticks()-timer_break_point
    millis=ticks%1000
    seconds=int(ticks/1000 % 60)
    minutes=int(ticks/60000 % 24)
    out='{minutes:02d}:{seconds:02d}:{millis}'.format(minutes=minutes, millis=millis, seconds=seconds)
    font.render_to(screen, (30,520), out, pygame.Color('dodgerblue'))

    pygame.display.flip()
```

It will start when the game starts and can reset and pause also.We are taking a variable time_break_point which will be zero initially.

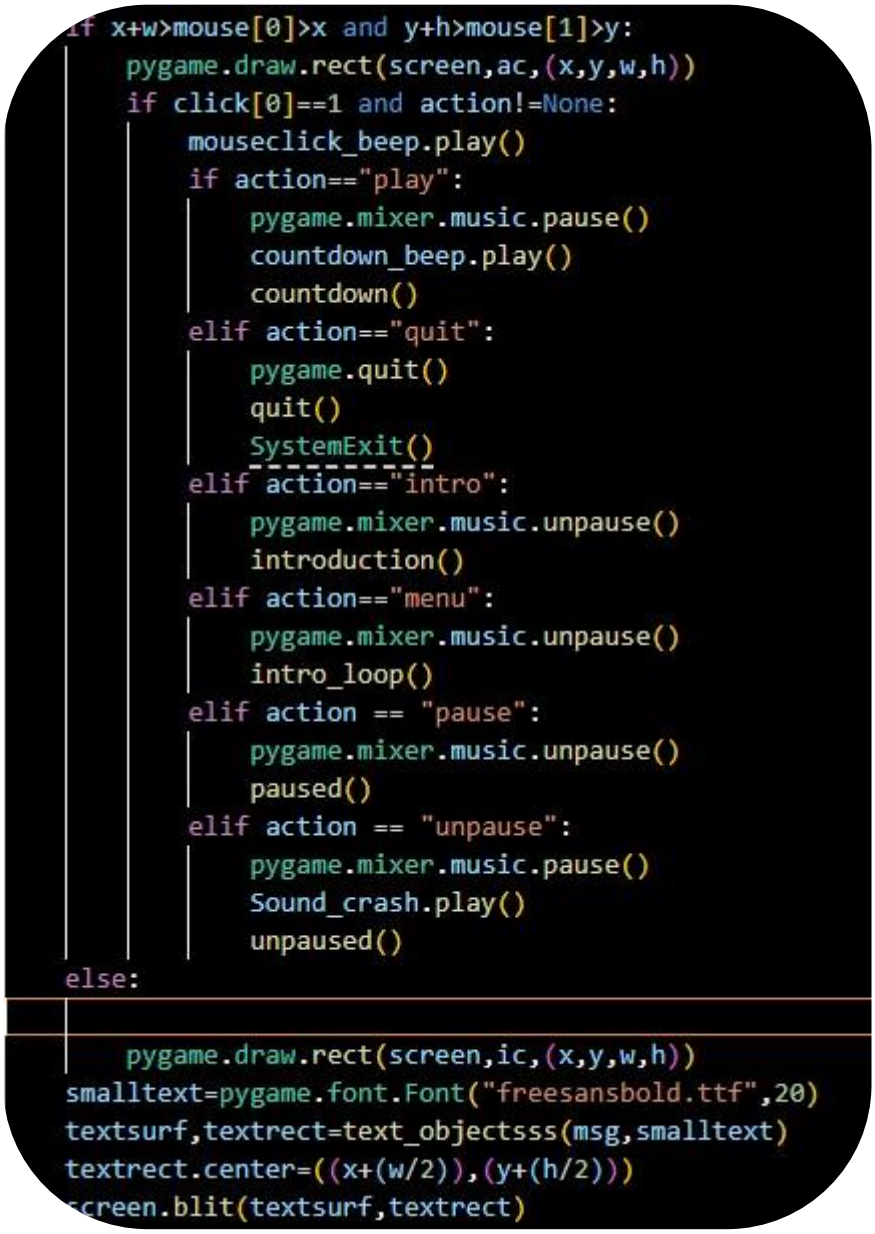
STEP-11:THE INTRO LOOP

In the intro screen, there will be three buttons named START, QUIT, and INSTRUCTION. The START button will initialize the game, the QUIT button will exit the window, INSTRUCTION button will show the player controls.Now if the current screen is the intro screen then a boolean named intro will be set to True.Draw rectangles using .rect(). Pass length, breadth, width & height to make the rectangle. blit() will take that rectangular Surface and put it on top of the screen.

```
aking the Intro Screen

def intro_loop():

    Sound_crash.stop()

    intro=True

    while intro:
        for event in pygame.event.get():
            if event.type==pygame.QUIT:
                pygame.quit()
                quit()
                SystemExit()

        screen.blit(intro_background,(0,0))

        largetext=pygame.font.Font("freesansbold.ttf",100)
        TextSurf,TextRect=text_object("CAR GAME",largetext)
        TextRect.center=(400,100)
        screen.blit(TextSurf,TextRect)
        button("START",150,520,100,50,dark_green,green,"play")
        button("QUIT",550,520,100,50,bright_red,red,"quit")
        button("INSTRUCTIONS",300,520,200,50,blue,skyblue,"intro")
        pygame.display.update()
        clock.tick(50)
```

STEP-12:THE BUTTON FUNCTION

Now let's make those fancy-looking buttons into reactive buttons. Think of scenarios like what should be the ability of the button that you have created i.e The PLAY button should trigger the main game loop with a counter which counts the number of seconds a player has played game The QUIT button should exit the window INTRO button should give an instruction manual of the game.The PAUSE button should freeze the current flow of the game and show a new screen with CONTINUE, RESTART, and MAIN MENU buttons .



```
if x+w>mouse[0]>x and y+h>mouse[1]>y:
    pygame.draw.rect(screen,ac,(x,y,w,h))
    if click[0]==1 and action!=None:
        mouseclick_beep.play()
        if action=="play":
            pygame.mixer.music.pause()
            countdown_beep.play()
            countdown()
        elif action=="quit":
            pygame.quit()
            quit()
            SystemExit()
        elif action=="intro":
            pygame.mixer.music.unpause()
            introduction()
        elif action=="menu":
            pygame.mixer.music.unpause()
            intro_loop()
        elif action == "pause":
            pygame.mixer.music.unpause()
            paused()
        elif action == "unpause":
            pygame.mixer.music.pause()
            Sound_crash.play()
            unpaused()
else:

    pygame.draw.rect(screen,ic,(x,y,w,h))
smalltext=pygame.font.Font("freesansbold.ttf",20)
textsurf,textrect=text_objectsss(msg,smalltext)
textrect.center=((x+(w/2)),(y+(h/2)))
screen.blit(textsurf,textrect)
```

```
# Function to create a button with specified parameters
# msg: The text to be displayed on the button
# x, y: The coordinates of the top-left corner of the button
# w, h: The width and height of the button
# ic: The color of the button when inactive
# ac: The color of the button when active (hovered over)
# action: The action to be performed when the button is clicked
```

STEP-13:THE INSTRUCTION SCREEN

Since the current screen is the instruction screen we will set the instruction boolean to true. Now in an Instruction manual, the most important things are the controls of the player and what the game is about. So we will organize those details in the instructions screen.

STEP-14:PAUSE AND UNPAUSE

Since the pause function will be there when you are playing the game it should have the capability to freeze every global action except its own function. So to make sure this happens we use a global declaration in the Pause button with the value set as True. When Pause is pressed it shows a new screen with CONTINUE, RESTART, and MAIN MENU buttons. When unpaused we simply reset the global boolean pause to False.

```
aused():

    Sound_crash.stop()
    pygame.mixer.music.unpause()
    global Pause
    paused_start_time = pygame.time.get_ticks()
    while Pause:
        global timer_break_point

        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                pygame.quit()
                quit()
                SystemExit()

        screen.blit(paused_background, (0, 0))

        largetext = pygame.font.Font('freesansbold.ttf', 115)
        TextSurf, TextRect = text_object("PAUSED", largetext)
        TextRect.center =((display_width/2),(display_hieght/2))
        screen.blit(TextSurf, TextRect)

        button("CONTINUE", 150, 450,150, 50, green,bright_green, "unpause")
        button("RESTART", 350, 450, 150,50, blue, skyblue,"play")
        button("MAIN MENU", 550, 450,200, 50,red, bright_red,"menu")
        pygame.display.update()
    paused_end_time = pygame.time.get_ticks()
    global timer_break_point
    timer_break_point = timer_break_point + (paused_end_time - paused_start_time)

    pygame.display.update()
    clock.tick(30)
```

```
# Making the Unpause Function

def unpaused():

    pygame.mixer.music.pause()
    Sound_crash.play()

    global Pause

    Pause = False
```

STEP-15:COUNTDOWN SYSTEM OF GAME

Let us make a scoreboard where the score and dodged cars will be accounted for. Apart from this let us also put the pause button in the countdown_background().

```
def countdown_background():

    pygame.mixer.music.pause()

    font = pygame.font.SysFont(None, 30)
    x = (display_width*0.45)
    y = (display_hieght*0.8)
    screen.blit(background,(220,0))
    screen.blit(carimg,(x,y))

    text=font.render("Passed : ",True,green)
    score=font.render("Score : ",True,red)
    high_score = font.render("High Score : ",True,white)
    distance = font.render("Distance : ",True,yellow)
    screen.blit(text,(20,60))
    screen.blit(score,(20,30))
    screen.blit(high_score,(20,90))
    screen.blit(distance,(20,120))

    button("PAUSE", 640,30, 150, 50, blue, bright_blue, "pause")
    button("QUIT",660,540,100,50,bright_red,red,"quit")
```

STEP-16:IMPLEMENTING THE COUNRDOWN FUNCTION

Whenever the game is on our countdown will be on so let's set the boolean countdown as true. Now loop until the countdown is true and in the loop call countdown_background. Also during the countdown, the clock should be ticking every 1sec so set the clock. tick(1). After every second we will update the display. This loop continues till the countdown boolean becomes false.

STEP-17:DEFINING OBSTACLES IN GAME

In a car race there will be some obstacles like opponent cars so let us define those obstacles. For obstacles to be made, we need their x,y, and obs id. Once we get this we can easily load those obstacles in the game window.

STEP-18:IMPLEMENTING SCORE SYSTEM

The score_system() function is used for showing the score and is used in the game_loop function which will be discussed below. text_objects() function is used for text rendering on screen.message_display() is used for displaying text we just need to pass the text that we want to show on the game screen.

```
# Displaying a Text

def message_display(text):

    largetext=pygame.font.Font("freesansbold.ttf",70)
    textsurf,textrect=text_objects(text,largetext)
    textrect.center=((display_width/2),(display_hieght/2))
    screen.blit(textsurf,textrect)

    pygame.display.update()
    time.sleep(3)
    game_loop()
```

```
Making the Scoring Process

def score_system(passed,score,high_score,distance):

    font=pygame.font.SysFont(None,30)
    text=font.render("Passed : "+str(passed)+" Cars",True,green)
    score=font.render("Score : "+str(score),True,red)
    high_score = font.render("High Score : "+str(high_score),True,white)
    distance = font.render("Distance : "+str(distance)+ " m",True,yellow)

    screen.blit(text,(20,60))
    screen.blit(score,(20,30))
    screen.blit(high_score,(20,90))
    screen.blit(distance,(20,120))
```

STEP-19:FALLBACK LOGIC

Whatever a developer builds the developer should always think about some fallback mechanism i.e if something goes wrong and the program does not work then the user should know what has happened. In this case, we will be displaying the message "YOU CRASHED" on the main screen.

```
def crash():

    Sound_crash.stop()
    pygame.mixer.music.pause()
    sound_crash=pygame.mixer.Sound("collision-reverb-103405.mp3")
    sound_crash.play()

    CRASH=True

    while CRASH:

        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                pygame.quit()
                quit()

        screen.blit(background,(220,0))
        screen.blit(Crash,(220,340))
        largetext = pygame.font.Font('freesansbold.ttf', 70)
        TextSurf, TextRect = text_objects("YOU HAVE CRASHED", largetext)
        TextRect.center =((display_width/2),(display_hieght/3))
        screen.blit(TextSurf, TextRect)

        button("RESTART", 150, 450, 150,50, blue, skyblue,"play")
        button("MAIN MENU", 450, 450,200, 50,red, bright_red,"menu")

        pygame.display.update()
        clock.tick(30)
```

STEP-20:WORKING ON GAME

As soon as game_loop() is initialized we check whether the boolean pause if it's false then we go further. We will set the obstacle_speed to 9 you can change the speed according to your convenience. Now we need to pop obstacles in random co-ords of the road so for that use random.randrange() for setting random x-coord.To check if the player has bumped into a car we will set initially a boolean bumped to false. Apart from this, In the game_loop function, we will configure controls for a player, a speed increase of obstacles after every level, and Incrementing scoreboard.

Now let's initiate the game by calling important functions i.e.

- intro_loop(): PLAY, QUIT, and INSTRUCTION button screen.
- game_loop() : Logic of the game.quit(): For exiting the game window

```
# Calling all the Functions

intro_loop()
game_loop()
pygame.quit()
quit()
```