8-bit game using pygame

Last Updated : 02 Sep, 2021

Pygame is a python library that can be used specifically to design and build games. Pygame supports only 2d games that are built using different sprites. Pygame is not particularly best for designing games as it is very complex to use doesn't have a proper GUI like unity but it definitely builds logic for further complex projects.
We'll be creating a simple game with the following rules:-

- The player can only move vertically.
- Other than player block there will be two other blocks.
- One of them will be the enemy block and one of them will be score block.
- If the player collides with the enemy block then the game over screen pops up, if the player collides with the score block the score is incremented and it is compulsory to collect all score blocks.

We'll be using various techniques such as the use of functions, random variables, various pygame functions etc.

Installation
Before initializing pygame library we need to install it. To install it type the below command in the terminal.

```
pip install pygame
```

After installing the pygame library we need to write the following lines to initialize the pygame library:-


```
import pygame
pygame.init()
```

These lines are pretty self explanatory. The pygame.init() function initiates the pygame library.
Then we need to initialize the screen where we want to place our blocks. This can be done by writing following lines:-

```
res = (720, 720)
screen = pygame.display.set_mode(res)
```

The tuple res holds two values that will define the resolution of our game. Then we need to define another variable screen that will actually act as our workbench. This can be done by using pygame.display.set_mode((arg, arg)). The tuple (arg, arg) can be stored into a variable res to reduce processor load. Now we need an actual screen to pop up when we run the code this can be done by a for and while loop in following way:-

```
while True:
for ev in pygame.event.get():
if ev.type==pygame.QUIT:
pygame.quit()
```

The while loop used here run till the condition is true. We define a variable ev that in pygame.event. Now if the user clicks on the cross button of the window the condition is changed to false and the while loop ends killing the current window.
Below is the implementation.

Python3

```
# Python program to demonstrate
# 8 bit game


import pygame
import sys
import random


# initialize the constructor
pygame.init()
res = (720, 720)

# randomly assigns a value to variables
# ranging from lower limit to upper
c1 = random.randint(125, 255)
c2 = random.randint(0, 255)
c3 = random.randint(0, 255)

screen = pygame.display.set_mode(res)
clock = pygame.time.Clock()
red = (255, 0, 0)
green = (0, 255, 0)
blue = (0, 0, 255)
color_list = [red, green, blue]
colox_c1 = 0
colox_c2 = 0
colox_c3 = 254
colox_c4 = 254

# randomly assigns a colour from color_list
# to player
player_c = random.choice(color_list)

# light shade of menu buttons
startl = (169, 169, 169)

# dark shade of menu buttons
startd = (100, 100, 100)
white = (255, 255, 255)
start = (255, 255, 255)
width = screen.get_width()
height = screen.get_height()

# initial X position of player
lead_x = 40

# initial y position of player
```

```
lead_y = height / 2
x = 300
y = 290
width1 = 100
height1 = 40
enemy_size = 50

# defining a font
smallfont = pygame.font.SysFont('Corbel', 35)

# texts to be rendered on screen
text = smallfont.render('Start', True, white)
text1 = smallfont.render('Options', True, white)
exit1 = smallfont.render('Exit', True, white)

# game title
colox = smallfont.render('Colox', True, (c3, c2, c1))
x1 = random.randint(width / 2, width)
y1 = random.randint(100, height / 2)
x2 = 40
y2 = 40
speed = 15

# score of the player
count = 0
rgb = random.choice(color_list)

# enemy position
e_p = [width, random.randint(50, height - 50)]
e1_p = [random.randint(width, width + 100), random.randint(50, height
        - 100)]


# function for game_over
def game_over():

    while True:

        # if the player clicks the cross
        # button
        for ev in pygame.event.get():
            if ev.type == pygame.QUIT:
                pygame.quit()

            if ev.type == pygame.MOUSEBUTTONDOWN:
                if 100 < mouse1[0] < 140 and height - 100 < mouse1[1] \
                    < height - 80:
                    pygame.quit()
```

```
                if ev.type == pygame.MOUSEBUTTONDOWN:
                    if width - 180 < mouse1[0] < width - 100 and height \
                        - 100 < mouse1[1] < height - 80:

                        # calling function game
                        game(lead_x, lead_y, speed, count)

            # fills the screen with specified colour
            screen.fill((65, 25, 64))
            smallfont = pygame.font.SysFont('Corbel', 60)
            smallfont1 = pygame.font.SysFont('Corbel', 25)
            game_over = smallfont.render('GAME OVER', True, white)
            game_exit = smallfont1.render('exit', True, white)
            restart = smallfont1.render('restart', True, white)
            mouse1 = pygame.mouse.get_pos()

            # exit
            if 100 < mouse1[0] < 140 and height - 100 < mouse1[1] < height - 80:
                pygame.draw.rect(screen, startl, [100, height - 100, 40,20])
            else:
                pygame.draw.rect(screen, startd, [100, height - 100, 40,20])

            # restart
            if width - 180 < mouse1[0] < width - 100 and height - 100 < mouse1[1] <
                pygame.draw.rect(screen, startl, [width - 180, height- 100, 80, 20])
            else:
                pygame.draw.rect(screen, startd, [width - 180, height- 100, 80, 20])

            screen.blit(game_exit, (100, height - 100))

            # superimposes one object on other
            screen.blit(restart, (width - 180, height - 100))
            screen.blit(game_over, (width / 2 - 150, 295))

            # updates frames of the game
            pygame.display.update()


    pygame.draw.rect(screen, startd, [100, height - 100, 40, 20])
    pygame.draw.rect(screen, startd, [width - 180, height - 100, 40, 50])



    # function for body of the game
    def game(
        lead_y,
        lead_X,
        speed,
        count,
```

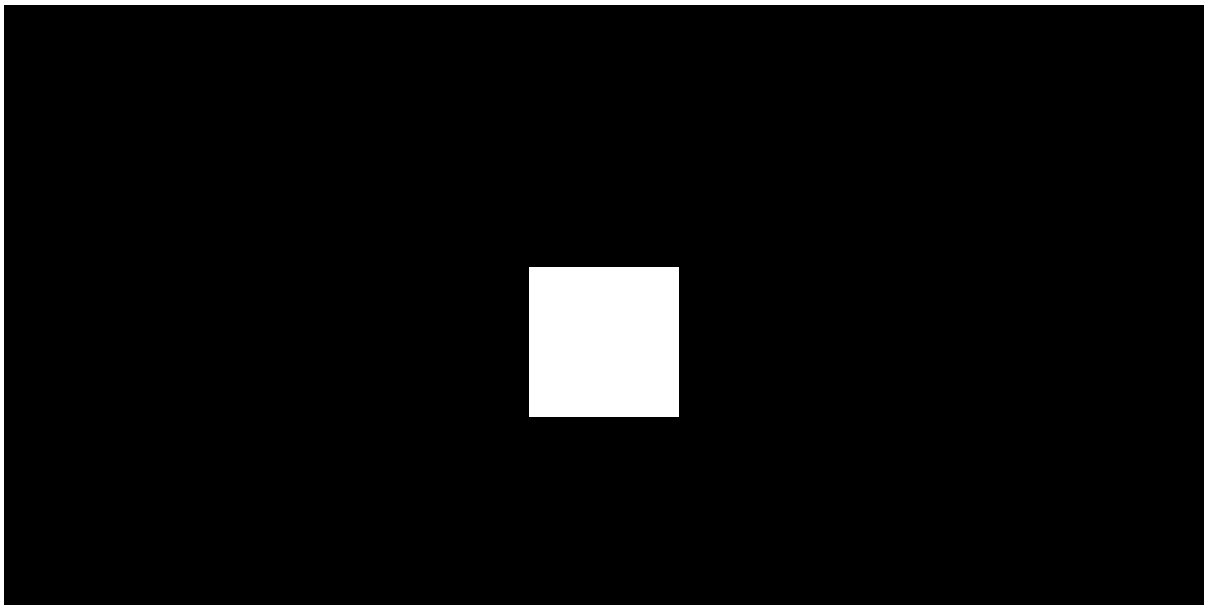
```
):

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


    # player control
    # keeps track of the key pressed
    keys = pygame.key.get_pressed()
    if keys[pygame.K_UP]:

        # if up key is pressed then the players
        # y pos will decrement by 10
        lead_y -= 10
    if keys[pygame.K_DOWN]:

        # if down key is pressed then the y pos
        # of the player is incremented by 10
        lead_y += 10
    screen.fill((65, 25, 64))
    clock.tick(speed)

    # draws a rectangle on the screen
    rect = pygame.draw.rect(screen, player_c, [lead_x, lead_y, 40,40])
    pygame.draw.rect(screen, (c1, c2, c3), [0, 0, width, 40])
    pygame.draw.rect(screen, (c3, c2, c1), [0, 680, width, 40])
    pygame.draw.rect(screen, startd, [width - 100, 0, 100, 40])
    smallfont = pygame.font.SysFont('Corbel', 35)
    exit2 = smallfont.render('Exit', True, white)

    # exit
    # gets the X and y position of mouse
    # pointer and stores them as a tuple
    mouse = pygame.mouse.get_pos()
    if width - 100 < mouse[0] < width and 0 < mouse[1] < 40:
        pygame.draw.rect(screen, startl, [width - 100, 0, 100, 40])
    else:
        pygame.draw.rect(screen, startd, [width - 100, 0, 100, 40])
    if width - 100 < mouse[0] < width and 0 < mouse[1] < 40:
        if ev.type == pygame.MOUSEBUTTONDOWN:
            pygame.quit()

    # enemy position
    if e_p[0] > 0 and e_p[0] <= width:

        # if the enemy block's X coordinate is between 0 and
        # the width of the screen the X value gets
```

```
        # decremented by 10
        e_p[0] -= 10
    else:
        if e_p[1] <= 40 or e_p[1] >= height - 40:
            e_p[1] = height / 2
        if e1_p[1] <= 40 or e1_p[1] >= height - 40:
            e1_p[1] = random.randint(40, height - 40)
        e_p[1] = random.randint(enemy_size, height - enemy_size)
        e_p[0] = width


    # game over
    # collision detection
    if lead_x <= e_p[0] <= lead_x + 40 and lead_y >= e_p[1] >= lead_y - 40:
        game_over()

    # checks if the player block has collided with the enemy block
    if lead_y <= e_p[1] + enemy_size <= lead_y + 40 and lead_x <= e_p[0] <=
        game_over()

    pygame.draw.rect(screen, red, [e_p[0], e_p[1], enemy_size,enemy_size])
    if e1_p[0] > 0 and e1_p[0] <= width + 100:
        e1_p[0] -= 10
    else:
        if e1_p[1] <= 40 or e1_p[1] >= height - 40:
            e1_p[1] = height / 2
        e1_p[1] = random.randint(enemy_size, height - 40)
        e1_p[0] = width + 100

    if lead_x <= e1_p[0] <= lead_x + 40 and lead_y >= e1_p[1] >= lead_y - 40
        e1_p[0] = width + 100
        e1_p[1] = random.randint(40, height - 40)
        count += 1
        speed += 1
    if lead_y <= e1_p[1] + enemy_size <= lead_y + 40 and lead_x <= e1_p[0] <
        e1_p[0] = width + 100
        e1_p[1] = random.randint(40, height - 40)

        # increases the score when blue box is hit
        count += 1

        # increases the speed as score increases
        speed += 1

        if count >= 45:

            # freezes the game FPS to 60 if
            # score reaches 45 or more
            speed = 60
```

```
        if lead_y <= 38 or lead_y >= height - 38:
            game_over()
        if e1_p[0] <= 0:
            game_over()

        pygame.draw.rect(screen, blue, [e1_p[0], e1_p[1], enemy_size,
                         enemy_size])
        score1 = smallfont.render('Score:', True, white)
        screen.blit(score1, (width - 120, height - 40))
        screen.blit(exit2, (width - 80, 0))
        pygame.display.update()


# intro
def intro(
    colox_c1,
    colox_c2,
    colox,
    exit1,
    text1,
    text,
    ):
    intro = True
    while intro:
        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                pygame.quit()
        screen.fill((65, 25, 64))
        mouse = pygame.mouse.get_pos()

        # start screen
        if x < mouse[0] < x + width1 and y < mouse[1] < y + height1:

            # if mouse is hovered on a button
            # its colour shade becomes lighter
            pygame.draw.rect(screen, startl, [x, y, width1, height1])
        else:
            if x < mouse[0] < x + width1 + 40 and y + 70 < mouse[1] < y \
                + 70 + height1:
                pygame.draw.rect(screen, startl, [x, y + 70, width1+40,height1])
            else:

                if x < mouse[0] < width1 + x and y + 140 < mouse[1] < y + 140 +
                    pygame.draw.rect(screen, startl, [x, y + 140,width1,height1])
                else:
                    pygame.draw.rect(screen, startd, [x, y, width1,height1])
                    pygame.draw.rect(screen, startd, [x, y + 70, width1
                            + 40, height1])
```

```
                    pygame.draw.rect(screen, startd, [x, y + 140,width1, height1])

            # start button
            if event.type == pygame.MOUSEBUTTONDOWN:
                if x < mouse[0] < x + width1 and y < mouse[1] < y + height1:
                    #music()
                    game(lead_y, lead_x, speed, count)

            if event.type == pygame.MOUSEBUTTONDOWN:
                if x < mouse[0] < width1 + x and y + 140 < mouse[1] < y + 140 + heig
                    pygame.quit()

            # this handles the colour breezing effect
            if 0 <= colox_c1 <= 254 or 0 <= colox_c2 <= 254:
                colox_c1 += 1
                colox_c2 += 1
            if colox_c1 >= 254 or colox_c2 >= 254:
                colox_c1 = c3
                colox_c2 = c3

            pygame.draw.rect(screen, (c2, colox_c1, colox_c2), [0, 0, 40,
                             height])
            pygame.draw.rect(screen, (c2, colox_c1, colox_c2), [width - 40,
                             0, 40, height])
            smallfont = pygame.font.SysFont('Corbel', 35)
            sig = smallfont.render('Designed by :- Antriksh', True, white)
            text = smallfont.render('Start', True, white)
            text1 = smallfont.render('Options', True, white)
            exit1 = smallfont.render('Exit', True, white)
            colox = smallfont.render('Colox', True, (c1, colox_c1,
                                     colox_c2))
            screen.blit(colox, (312, 50))
            screen.blit(text, (312, 295))
            screen.blit(text1, (312, 365))
            screen.blit(exit1, (312, 435))
            screen.blit(sig, (320, height - 50))
            clock.tick(60)
            pygame.display.update()


    intro(
        colox_c1,
        colox_c2,
        colox,
        exit1,
        text1,
        text,
        )
```

Output:



Looking to dive into the world of programming or sharpen your Python skills? Our **Master Python: Complete Beginner to Advanced Course** is your ultimate guide to becoming proficient in Python. This course covers everything you need to build a solid foundation from fundamental programming concepts to advanced techniques. With **hands-on projects**, real-world examples, and expert guidance, you'll gain the confidence to tackle complex **coding challenges**. Whether you're starting from scratch or aiming to enhance your skills, this course is the perfect fit. Enroll now and master Python, the language of the future!

6

Previous Article

Tic Tac Toe GUI In Python using PyGame

Next Article

Bubble sort visualizer using PyGame

Similar Reads

Adding Collisions Using pygame.Rect.colliderect in Pygame

Prerequisite: Drawing shapes in Pygame, Introduction to pygame In this article, we are going to use pygame.Rect.colliderect for adding collision in a shape usin...

3 min read

How to create Buttons in a game using PyGame?

Pygame is a Python library that can be used specifically to design and build games. Pygame only supports 2D games that are build using different...

3 min read

Building and visualizing Sudoku Game Using Pygame

Sudoku is a logic-based, combinatorial number-placement puzzle. The objective is to fill a 9×9 grid with digits so that each column, each row, and each of the nin...

7 min read

Spiral Sprint Game in Python Using Pygame

In this article, we will see how to create a spiral sprint game in Python using Pygame. In this game, we will have functionality like difficulty modes, obstacle...

15+ min read

Stimulate bouncing game using Pygame

In this article, we will learn to make a bouncing game using Pygame. Pygame is a set of Python modules designed for writing video games. It adds functionality o...

4 min read

Brick Breaker Game In Python using Pygame

Brick Breaker is a 2D arcade video game developed in the 1990s. The game consists of a paddle/striker located at the bottom end of the screen, a ball, and...

14 min read

Angry Bird Game Using PyGame

Let's discuss the angry bird game using Pygame, I hope this article will be very interesting as it provides a holistic gaming experience at the end. We begin by...

14 min read

Balloon Archer game in Python using the Pygame module

Balloon Archer is a 2D game where the player uses a bow and arrow to pop balloons that float across the screen. The main theme of this game is to shoot...

15+ min read

Snake Game in Python - Using Pygame module

Snake game is one of the most popular arcade games of all time. In this game, the main objective of the player is to catch the maximum number of fruits without...

15+ min read

Dodger Game using Pygame in Python

A Dodger game is a type of game in which a player has access to control an object or any character to dodge (to avoid) upcoming obstacles and earn points...

5 min read

Article Tags : Python Python Programs Python-PyGame

Practice Tags : python

OOAD
System Design Bootcamp
Interview Questions
Puzzles

School Subjects

Mathematics
Physics
Chemistry
Biology
Social Science
English Grammar
Commerce
World GK

GeeksforGeeks Videos

DSA
Python
Java
C++
Web Development
Data Science
CS Subjects