VISVESVARAYA TECHNOLOGICAL UNIVERSITY

BELAGAVI-590 018, KARNATAKA.

MINI PROJECT REPORT

ON

"SPACE INVADERS USING PYGAME"

PROJECT ASSOCIATES

Arpitha Prakash Hegde	**4BD17CS018**
Sonika Prakash	**4BD17CS104**
Roja Devina	**4BD17CS094**
Teju G M	**4BD17CS114**
Bhavana V	**4BD17CS024**

PROJECT GUIDES:

Prof. Naseer R

Asst. Professor

Department of CS&E

B.I.E.T., Davanagere

2019-2020

Department of Computer Science and Engineering

Bapuji Institute of Engineering & Technology

Davangere- 577004

Bapuji Institute of Engineering and Technology

Davangere -577004

Department of Computer Science and Engineering

CERTIFICATE

This is to certify that **ARPITHA PRAKASH HEGDE, SONIKA PRAKASH, ROJA DEVINA, TEJU G M** and **BHAVANA V** bearing **USN 4BD17CS018, 4BD17CS104, 4BD17CS094, 4BD17CS114** and **4BD17CS024** respectively of **Computer Science and Engineering** department have satisfactorily submitted the mini project report entitled **"SPACE INVADERS USING PYGAME"**. The report of the project has been approved as it satisfies the academic requirements in respect of project work prescribed for the academic year 2020.

Project Guide

_____

Prof. Naseer R
Asst. Professor
Department of CS&E
B.I.E.T., Davangere.

Head of Department

_____

Dr. Nirmala C.R Ph.D., M. Tech., M.I.S.T.E
Prof &Head, Department of CS&E
B.I.E.T., Davangere

Date:

Place:

Signature of Examiners:

(1) _____

(2) _____

ACKNOWLEDGEMENT

Salutations to our beloved and highly esteemed institute, **"BAPUJI INSTITUTE OF ENGINEERING AND TECHNOLOGY"** for having well qualified staff and labs furnished with necessary equipments.

We express our sincere thanks to our guide **Prof ANU C.S & Prof NAVEEN H.M** for giving us constant encouragement, support and valuable guidance throughout the course of the project without whose stable guidance this project would not have been achieved.

We express whole hearted gratitude to **Dr NIRMALA C.R** who is our respectable HOD of Computer Science & Engineering Department. We wish to acknowledge her help who made our task easy by providing with his valuable help and encouragement.

We also express our whole hearted gratitude to our principal, **Dr H.B ARAVINDA ,** for his moral support and encouragement.

We would like to extend my gratitude to all staff of Department of Computer science and engineering for the help and support rendered to me. I have benefited a lot from the feedback, suggestions given by them.

We would like to extend my gratitude to all my family members and friends especially for their advice and moral support.

ARPITHA PRAKASH HEGDE(4BD17CS018)

SONIKA PRAKASH(4BD17CS104)

ROJA DEVINA(4BD17CS094)

TEJU G M(4BD17CS114)

BHAVANA V(4BD17CS024)

Bapuji Educational Association (Regd.)

Bapuji Institute of Engineering and Technology, Davangere-577004

Department of Computer Science and Engineering

Vision and Mission of the Department

VISION

To be a center of excellence in imparting state-of-the –art technology in Computer Science and Engineering education enabling the students to become professionally sound and ethically strong.

MISSION

M1: Adapting best in class teaching and learning methodology to mould the students to

become industry ready.

M2: Creating conducive environment for imparting quality education to facilitate research and Innovation

M3: Establishing industry institute relationship to bridge the skill gap.

M4: Educating the students to be successful lifelong learners by inculcating ethical values and social responsibilities.

CONTENTS

ABSTRACT

Space Invaders is a two-dimensional fixed shooter game in which the player controls a spaceship with keyboard arrows by moving it horizontally across the bottom of the screen and firing at descending aliens with bullet. The aim is to defeat the aliens before they come too close to the spaceship. The player defeats an alien, and earns points, by shooting it with the bullet.

CHAPTER 1

INTRODUCTION

1.1 INTRODUCTION TO PYGAME

Pygame is a set of Python modules designed for writing video games. Pygame adds functionality on top of the excellent SDL library. This allows us to create fully featured games and multimedia programs in the python language.

1.2 HISTORY

Pygame was originally written by Pete Shinners to replace PySDL after its development stalled. It has been a community project since 2000 and is released under the open source free software GNU Lesser General Public License.

1.3 INSTALLING PYGAME

Pygame requires Python. We python 3.6.1 or greater, because it is much friendlier to newbies, and additionally runs faster.

The best way to install pygame is with the pip tool (which python uses to install packages). We use the --user flag to tell it to install into the home directory, rather than globally.

python3 -m pip install -U pygame --user

1.4 FEATURES OF PYGAME

- **Multi core CPUs can be used easily**
  With dual core CPUs common and 8 core CPUs cheaply available on desktop systems, making use of multi core CPUs allows you to do more in your game. Selected pygame functions release the dreaded python GIL, which is something you can do from C code.
- **Uses optimized C and Assembly code for core functions**

C code is often 10-20 times faster than python code, and assembly code can easily be 100x or more times faster than python code.

- **Truly portable**

  Supports Linux (pygame comes with most main stream linux distributions), Windows (95, 98, ME, 2000, XP, Vista, 64-bit Windows, etc), Windows CE, BeOS, MacOS, Mac OS X, FreeBSD, NetBSD, OpenBSD, BSD/OS, Solaris, IRIX, and QNX. The code contains support for AmigaOS, Dreamcast, Atari, AIX, OSF/Tru64, RISC OS, SymbianOS and OS/2, but these are not officially supported. You can use it on hand held devices, game consoles and the One Laptop Per Child (OLPC) computer.

- **It's Simple and easy to use**

  Kids and adults make shooter games with pygame. Pygame is used in the OLPC project and has been taught in essay courses to young kids and college students. It's also used by people who first programmed in z80 assembler or c64 basic.

- **You control your main loop**

  You call pygame functions; they don't call your functions. This gives you greater control when using other libraries, and for different types of programs.

- **Fast response to reported bugs**

  Some bugs are patched within an hour of being reported. Sometimes we suck at bug fixes, but mostly we're pretty good bug fixers. Bug reports are quite rare these days, since a lot of them have been fixed already.

- **Small amount of code**

  It does not have hundreds of thousands of lines of code for things you won't use anyway. The core is kept simple and extra things like GUI libraries, and effects are developed separately outside of pygame.

CHAPTER 2

SYSTEM REQUIREMENTS

2.1 Software Requirements

1. Operating System : Microsoft Windows 10
2. Python module used: Pygame
3. Pygame version: 1.9.6
4. Language used: Python 3

2.2 Hardware Requirements

1. Main Processor: PENTIUM III
2. Processor Speed: 800 MHz
3. RAM Size: 128MB DDR

CHAPTER 3

IMPLEMENTATION

3.1 BASIC SETTINGS

First we initialize pygame instance and create a game window. Then we declare all the images and audios used in the game.

```
import math
import random

import pygame
from pygame import mixer

# Intialize the pygame
pygame.init()

# create the screen
screen = pygame.display.set_mode((800, 600))

# Background
background = pygame.image.load('background.png')

# Sound
mixer.music.load("background.wav")
mixer.music.play(-1)

# Caption and Icon
pygame.display.set_caption("Space Invader")
icon = pygame.image.load('ufo.png')
pygame.display.set_icon(icon)
```

Then we set a game loop that keeps displaying the background image and takes input from the user. The spaceship movement is controlled by the keyboard arrows.

```
# Game Loop
running = True
while running:

    # RGB = Red, Green, Blue
    screen.fill((0, 0, 0))
    # Background Image
    screen.blit(background, (0, 0))
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            running = False

        # if keystroke is pressed check whether its right or left
        if event.type == pygame.KEYDOWN:
            if event.key == pygame.K_LEFT:
                playerX_change = -5
            if event.key == pygame.K_RIGHT:
                playerX_change = 5
            if event.key == pygame.K_SPACE:
                if bullet_state is "ready":
                    bulletSound = mixer.Sound("laser.wav")
                    bulletSound.play()
                    # Get the current x cordinate of the spaceship
                    bulletX = playerX
                    fire_bullet(bulletX, bulletY)

        if event.type == pygame.KEYUP:
            if event.key == pygame.K_LEFT or event.key == pygame.K_RIGHT:
                playerX_change = 0
```

3.2 ENEMIES

Then we create a list that holds the images for the enemies and two more lists for the enemy's coordinates. We have created six enemies in our game.

```

# Enemy
enemyImg = []
enemyX = []
enemyY = []
enemyX_change = []
enemyY_change = []
num_of_enemies = 6

for i in range(num_of_enemies):
    enemyImg.append(pygame.image.load('enemy.png'))
    enemyX.append(random.randint(0, 736))
    enemyY.append(random.randint(50, 150))
    enemyX_change.append(4)
    enemyY_change.append(40)
```

3.3 SPACESHIP

The spaceship, controlled by the player, is very similar to enemy. He's different in size, position and he's placed in the screen center at the bottom of the screen.

```
25  # Player
26  playerImg = pygame.image.load('player.png')
27  playerX = 370
28  playerY = 480
29  playerX_change = 0
```

3.4 BULLET

The bullet used to defeat enemies is the initialized. Bullet's image and its coordinates are initialized.

```
46  # Bullet
47
48  # Ready - You can't see the bullet on the screen
49  # Fire - The bullet is currently moving
50
51  bulletImg = pygame.image.load('bullet.png')
52  bulletX = 0
53  bulletY = 480
54  bulletX_change = 0
55  bulletY_change = 10
56  bullet_state = "ready"
```

3.5 SCORE AND GAME OVER

The initial score is set to zero. The coordinates where it is to be displayed is then set. The game over text is also initialized.

```
58  # Score
59
60  score_value = 0
61  font = pygame.font.Font('freesansbold.ttf', 32)
62
63  textX = 10
64  testY = 10
65
66  # Game Over
67  over_font = pygame.font.Font('freesansbold.ttf', 64)
```

3.6 DISPLAYING IMAGES AND TEXT

Finally all the loaded images and texts are displayed in the game window using screen.blit() method.

```
70  def show_score(x, y):
71      score = font.render("Score : " + str(score_value), True, (255, 255, 255))
72      screen.blit(score, (x, y))
73
74
75  def game_over_text():
76      over_text = over_font.render("GAME OVER", True, (255, 255, 255))
77      screen.blit(over_text, (200, 250))
78
79
80  def player(x, y):
81      screen.blit(playerImg, (x, y))
82
83
84  def enemy(x, y, i):
85      screen.blit(enemyImg[i], (x, y))
86
87
88  def fire_bullet(x, y):
89      global bullet_state
90      bullet_state = "fire"
91      screen.blit(bulletImg, (x + 16, y + 10))
```

3.7 COLLISION

A collision between a bullet and an enemy occurs if the distance between them is less than 27 pixels.

```
def isCollision(enemyX, enemyY, bulletX, bulletY):
    distance = math.sqrt(math.pow(enemyX - bulletX, 2) + (math.pow(enemyY - bulletY, 2)))
    if distance < 27:
        return True
    else:
        return False
```

3.8 ENEMY MOVEMENT

The enemy movement is done by changing their images' position every by small amounts so that they appear to be in a continuous motion.

If a collision occurs then an explosion sound is played and the enemy is are relocated at the top of the screen.

```
    # Enemy Movement
    for i in range(num_of_enemies):

        # Game Over
        if enemyY[i] > 440:
            for j in range(num_of_enemies):
                enemyY[j] = 2000
            game_over_text()
            break

        enemyX[i] += enemyX_change[i]
        if enemyX[i] <= 0:
            enemyX_change[i] = 4
            enemyY[i] += enemyY_change[i]
        elif enemyX[i] >= 736:
            enemyX_change[i] = -4
            enemyY[i] += enemyY_change[i]

        # Collision
        collision = isCollision(enemyX[i], enemyY[i], bulletX, bulletY)
        if collision:
            explosionSound = mixer.Sound("explosion.wav")
            explosionSound.play()
            bulletY = 480
            bullet_state = "ready"
            score_value += 1
            enemyX[i] = random.randint(0, 736)
            enemyY[i] = random.randint(50, 150)

        enemy(enemyX[i], enemyY[i], i)
```

CHAPTER 4

SNAPSHOTS

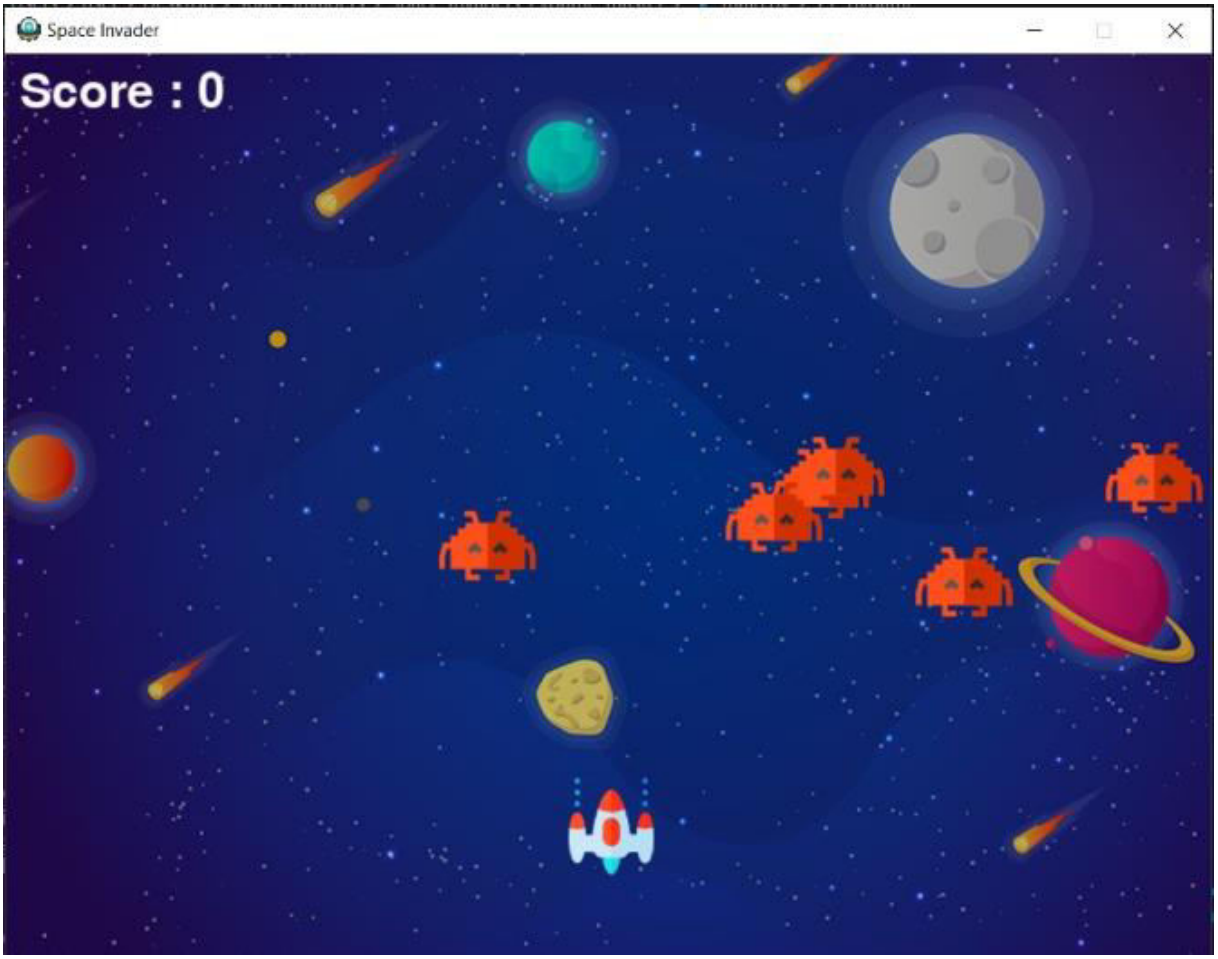

Fig 4.1: Game play

This figure illustrates the spaceship shooting bullets at the enemies to defeat them.

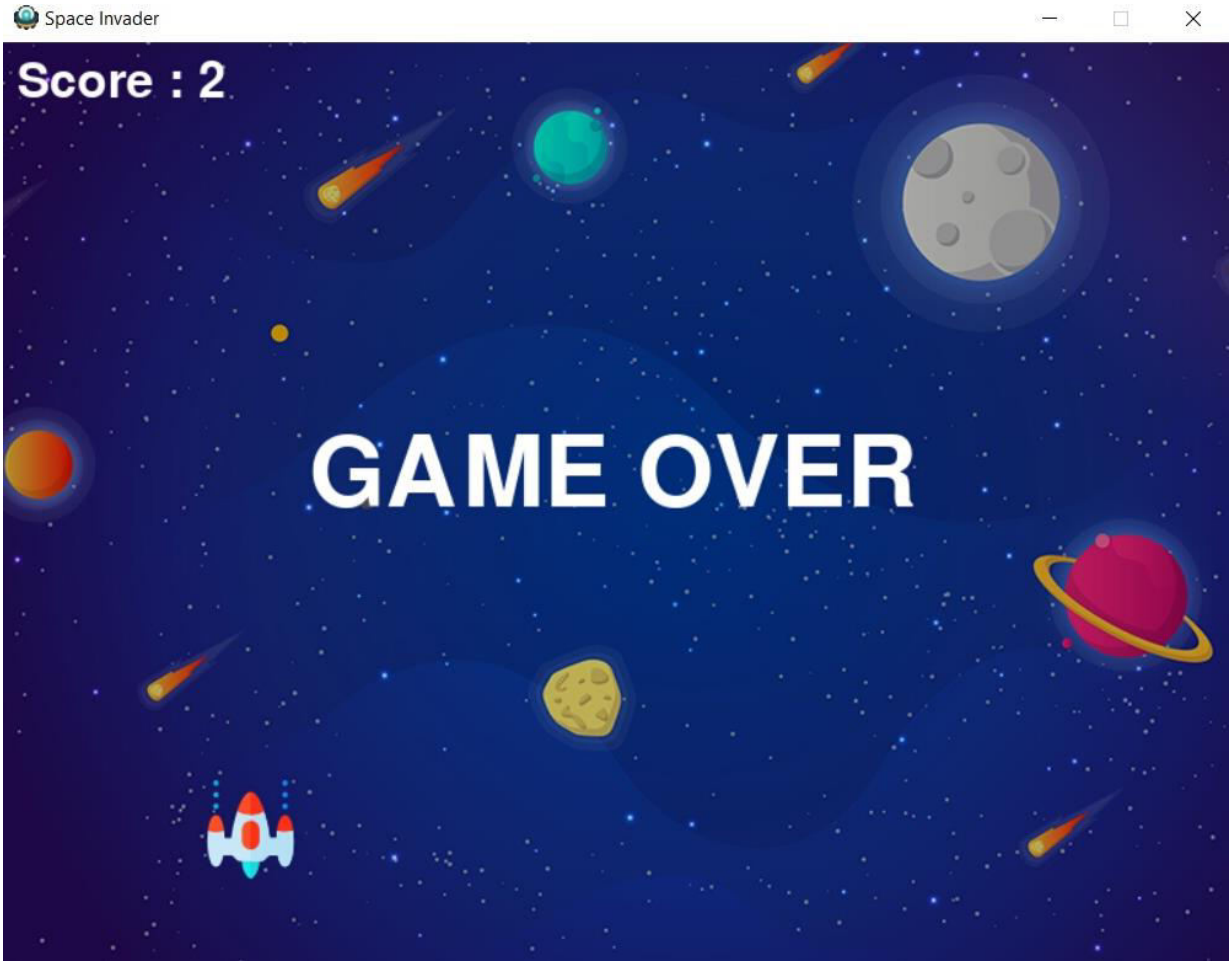

Fig 4.2: Game over

This figure illustrates that the game is over and the final score is displayed.

CONCLUSION

This Python mini project was built completely using Pygame module. The fact that new users can get up to speed so easily makes Pygame a good place to start for making games. Python is also a pretty easy language to write in for experienced coders, so making a quick prototype of a game in Pygame is easy and fun. During the development of this project, efforts lead to understanding the basics of Pygame module. Various functions and operations of the pygame library provided the learning platform to get the maximum performance of the Pygame functions. On conclusion this mini project is implemented successfully with the Pygame module.

BIBLIOGRAPHY

Websites

[1] https://itnext.io

[2] https://github.com

[3] https://youtube.com