**PROJECT REPORT**

**ON**

**SPACE INVADERS IN PYTHON USING PYGAME**

****

REPORT SUBMITTED

TO

VISHWAKARMA INSTITUTE OF INFORMATION TECHNOLOGY, PUNE

FOR THE PBL PYTHON FOR ENGINEERING

IN

**ENGINEERING AND APPLIED SCIENCE DEPARTMENT**

BY

**Vipin Thombare -** 9035/ 22010580

**Prasanna Eklahare -** 9036 / 22010596

**Tanmay Deshpande -** 9037 / 22010612

**Tejas Fuke -** 9038 / 22010628

**Yatik Khairnar -** 9039 / 22010644

**Sangam Mandwade -** 9040 / 22010660

**Class: FE Division: I Batch: I2**

**Batch Teacher**

**Mrs. Aparna Barbadekar**

**INDEX**

|  |  |  |
| --- | --- | --- |
| **Sr. No.** | **Contents** | **Page No.** |
| **1** | **ABSTRACT** | **1** |
| **2** | **Introduction and Theory** | **2** |
| **3** | **FLOW of Program** | **3** |
| **4** | **source code of the program** | **8** |
| **5** | **output** | **16** |
| **6** | **LIST OF THE TOPICS/ CONCEPT WHICH ARE COVERED FROM THE SYLLABUS** | **17** |
| **7** | **LIST OF THE TOPICS/ CONCEPT WHICH ARE OUT OF SYLLABUS** | **18** |
| **8** | **CONCLUSION** | **19** |
| **9** | **REFERENCES** | **20** |

**ABSTRACT:**

The aim of this project is to build a 2D arcade like game from starting in Python. The core of the game, is PyGame, a set of Python modules designed for game development, which was used for tasks such bliting images on the screen or moving the said images.

The goal of the player being to destroy all the enemy spaceships. The following report will provide an indepth understanding of the project.

The report starts by creating a context for this work and will continue guiding the reader through the journey of its implementation.

The second part of the document will give information about the development of the game and the flow of program. And Which topics are used on the project from syllabus and which topics are learned which were out of syllabus.

Finally, the report concludes with a reflection of the outcome of the project.

**Introduction and Theory:**

Game development is a field dominated by programming languages such as c / c++ . We aimed to create a game purely from python changes

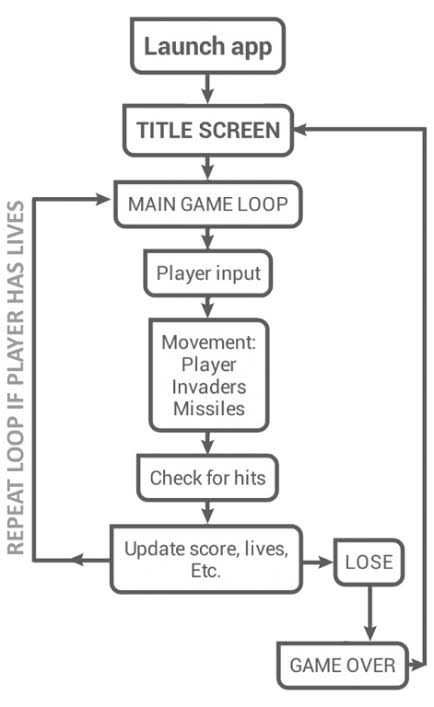
The game, called space defender, is a 2d arcade­like space shooter built with python and elements of pygame. The main idea of the game is for the player to destroy all the enemy spaceships before advancing to the last level of the game where intelligent bad guys will do their absolute best to defeat the good guy.

The player spaceship is controlled keyboard and the enemy spaceships have a predefined movement.

The game itself has a number of graphical items, such as the player spaceship, the enemy space ships, bullets, that interact with each other maximising the gameplay experience. All these entities are loaded as PNGs for easier manipulation.

The Enemies were defined as at the starting of the game only one enemy is present on the screen. As the first enemy is hitted, number of enemies also increased and speed of enemies also changes increases. And As the score of the player increases so does the difficulty of game , after reaching a certain score the number of enemies increases and also the speeds of enemy ships increase . This will go on until the enemy spaceships collide with the player or reach the bottom of the screen after which the game will be over and final score will be displayed.

**FLOW OF PROGRAM:**

****

1. **Start**

The Start Window is made using tkinter module. It has two buttons one for Starting the game and other exit. It also has game logo representing Spaceship.

1. **Game window appears**

The starting window has an amazing background image. At the starting of the game player spaceship and only one enemy spaceship is present.

1. **Player and Its movement**

The player's spaceship can move in horizontal and vertical directions upon pressing certain keys on keyboard assigned for that purpose , the player's spaceship can only move within the constraints of the boundary of the screen .

1. **Bullets are fired**

The bullets are fired when space key is pressed.

1. **Score increases**

As the player kills the enemies, he gain some score and Using render function, live score is displayed on screen.

1. **enEMIES AND collision**

Different types of enemies appeared on screen having different images. Their movement is defined with time. The enemies move sideways rebounding in opposite direction when colliding with the boundary of the screen, and they also move towards the bottom of the screen after each rebound. When an enemy is killed there is explosion and after that enemy get reappears.

1. **Multiple ENEMIES AND Difficiulty levels**

With increasing score the difficulty level also increase and with increase in difficulty level the number of enemy also increase. We have created multiple enemies using list and we have different types of enemies.

1. **Music**

We have used different types of music in our project. We have inserted background music and music for collision. We have inserted music using the mixer module from pygame.

1. **Game over**

When the game reaches some certain score, game gets over and game over window pops up.

**SOURCE CODE OF THE PROGRAM:**

**Main Screen File:**

**(Main\_screen.py))**

from tkinter import \*  
  
root = Tk()  
root.geometry("350x450") # width x height  
root.title("SPACE INVADERS")  
root.configure(bg="dark blue")  
  
text = Label(root**,** text="𝐒𝐏𝐀𝐂𝐄 𝐈𝐍𝐕𝐀𝐃𝐄𝐑"**,** padx=**11,** pady=**11,** bg="cyan"**,** fg="black"**,** font="helvetica,10,bold"**,** borderwidth=**2,** relief=GROOVE)  
text.pack()  
  
photo = PhotoImage(file="project.png")  
labelphoto = Label(root**,** image=photo)  
labelphoto.pack()  
  
  
def command():  
 import Game\_loop  
  
  
  
  
btn = Button(root**,** text="𝙉𝙀𝙒 𝙂𝘼𝙈𝙀"**,** bg="orange"**,** padx=**48,** pady=**12,** font="Gotham,10,bold"**,** command=command)  
btn.pack()  
btn1 = Button(root**,** text="𝙀𝙓𝙄𝙏"**,** bg="orange"**,** padx=**80,** pady=**12,** font="Gotham,10,bold"**,** command=quit)  
btn1.pack()  
  
root.mainloop()

**Game loop:**

**(Game\_loop.py)**

# importing modules  
  
import pygame  
import math  
import random  
from Exit\_screen import game\_over  
from pygame import mixer  
  
# initialise pygame  
pygame.init()  
  
#adding background music  
  
mixer.init()  
mixer.music.load("Space\_Invaders\_Music.ogg")  
mixer.music.play(-**1**)  
  
#adding screen and background image  
screen = pygame.display.set\_mode((**800, 600**))  
  
background = pygame.image.load('back.png')  
  
  
# Adding caption to screen  
pygame.display.set\_caption("space invaders")  
icon = pygame.image.load(r'spaceship.png')  
pygame.display.set\_icon(icon)  
  
# initial score  
score = **0**# player constants and image  
  
playerimg = pygame.image.load(r'playerimg.png')  
playerx = **370**playery = **480**player\_hzt = **0**player\_vert = **0**# enemiy empty lists and constants  
  
q=**1**x = " "  
enemyimg = []  
alienx = []  
alieny = []  
vel\_x = []  
vel\_y = []  
x\_speed = **0.9**y\_speed = **5**# bullets image and constants  
  
bulletimg = pygame.image.load(r'bullet1.png')  
bulletx = **0**bullety = **480**bulletx\_change = **0**bullety\_change = **10** # speed at which bullets y coordinate changes i.e speed of bullet  
bullet\_state = "ready"  
  
# bringing player to screen  
def player(x**,** y):  
 screen.blit(playerimg**,** (playerx**,** playery)) # blit function used to draw on screen  
  
# bringing player to screen  
def enemy(x**,** y**,** i):  
 screen.blit(enemyimg[i] **,**(x**,** y) )  
  
# bringing bullets to screen  
def fire\_bullet(x**,** y):  
 global bullet\_state  
 bullet\_state = "fire"  
 screen.blit(bulletimg**,** (x + **16,** y + **10**))  
  
  
# collision logic  
def is\_collision(bulletx**,** bullety**,** alienx**,** alieny):  
 distance = math.sqrt((math.pow(bulletx - alienx**, 2**)) + (math.pow(bullety - alieny**, 2**)))  
 if distance < **28**:  
 return True  
 else:  
 return False  
  
  
# enemy collision image  
v = **1**m = **0**n = **0**# GAME LOOP : every in game related action should be in the game loop  
running = True  
  
while running:  
  
 screen.fill((**0, 0, 0**))  
  
 screen.blit(background**,** (**0, 0**))  
 # screen.blit(text, text\_rect)  
  
  
  
 # CREATING ENEMIES  
 num\_of\_enemies = q  
 colm = **5** for i in range(num\_of\_enemies):  
 for j in range(colm):  
 if j == **0**:  
 enemyimg.append(pygame.image.load('enemyimg.png'))  
 if j == **1**:  
 enemyimg.append(pygame.image.load('en1\_2.png'))  
 if j == **2**:  
 enemyimg.append(pygame.image.load('en2\_2.png'))  
 else:  
 enemyimg.append(pygame.image.load('en3\_2.png'))  
 alienx.append(**140** + j \* **90**)  
 alieny.append(**100** + i \* **70**)  
 vel\_x.append(**2**)  
 vel\_y.append(**40**)  
  
  
  
 # CODE FOR EXITTING WINDOW  
  
 for event in pygame.event.get():  
 if event.type == pygame.QUIT:  
 running = False  
  
 # Making buttons for player movement  
  
 if event.type == pygame.KEYDOWN :  
 if event.key == pygame.K\_UP or event.key == pygame.K\_w :  
 player\_vert = -**3** if event.key == pygame.K\_DOWN or event.key == pygame.K\_s:  
 player\_vert = **3** if event.key == pygame.K\_LEFT or event.key == pygame.K\_a:  
 player\_hzt = -**3** if event.key == pygame.K\_RIGHT or event.key == pygame.K\_d:  
 player\_hzt = **3** if event.key == pygame.K\_SPACE:  
 if bullet\_state == "ready":  
 bulletx = playerx  
 bullety = playery  
 fire\_bullet(bulletx**,** bullety)  
  
 if event.type == pygame.KEYUP:  
 if event.key == pygame.K\_LEFT or event.key == pygame.K\_RIGHT or event.key == pygame.K\_a or event.key == pygame.K\_d:  
 player\_hzt = **0** if event.key == pygame.K\_UP or event.key == pygame.K\_DOWN or event.key == pygame.K\_w or event.key == pygame.K\_s:  
 player\_vert = **0** # Ensuring enemy inside screen  
  
 playerx = playerx + player\_hzt  
 if playerx <= **0**:  
 playerx = **0** elif playerx >= **736**:  
 playerx = **736** playery = playery + player\_vert  
 if playery <= **0**:  
 playery = **0** elif playery >= **536**:  
 playery = **536** # enemy movements  
  
 for i in range(num\_of\_enemies):  
 alienx[i] = alienx[i] + vel\_x[i]  
 if alienx[i] <= **0**:  
 vel\_x[i] = x\_speed  
 alieny[i] = vel\_y[i] + alieny[i]  
 elif alienx[i] >= **736**:  
 vel\_x[i] = -x\_speed  
 alieny[i] = vel\_y[i] + alieny[i]  
  
  
 # Collision of enemy and bullet  
  
 collision = is\_collision(bulletx**,** bullety**,** alienx[i]**,** alieny[i])  
 if collision:  
 bullety = **480** bullet\_state = "ready"  
  
 v = **3** m = alienx[i]  
 n = alieny[i]  
  
 # sound for collision  
 mixer.init()  
 mixer.music.load("coll.wav")  
 mixer.music.play(**1**)  
  
 # score calculation  
 score = score + **1** x = str(int(score))  
  
  
 print(x)  
 # global y  
 y = int(x)  
 alienx[i] = random.randint(**0, 730**)  
 alieny[i] = random.randint(**50, 150**)  
 if alieny == **0**:  
 running = False  
 enemy(alienx[i]**,** alieny[i]**,** i)  
  
 # enemy image collision  
  
 if v > **2** and v < **20**:  
 colli\_image = pygame.image.load("explosionpurple.png")  
 screen.blit(colli\_image**,** (m**,** n))  
 v += **1** # bullet movement  
  
 if bullety <= **0**:  
 bullety = **480** bullet\_state = "ready"  
 if bullet\_state == "fire":  
 fire\_bullet(bulletx**,** bullety)  
 bullety = bullety - bullety\_change  
  
  
 # SCORE AND ENEMY level DEFINING  
  
 if score > **1**:  
 x\_speed = **1** q =**2** if score > **3**:  
 q = **4** if score > **9**:  
 x\_speed = **1.5** q = **6** if score > **12**:  
 q =**10** if score > **15**:  
 x\_speed = **1.7** if score > **18**:  
 x\_speed = **3** # game over window  
  
 if score == **20**:  
 game\_over()  
   
   
  
 font = pygame.font.SysFont("Arial"**, 30**)  
 text = font.render("score:" + x**,** True**,** (**240,240,240**) ) # add parameteres , requires at least 3 parameters  
 text\_rect = text.get\_rect()  
 text\_rect.center = (**60, 30**)  
 screen.blit(text**,** text\_rect )  
  
  
 # updating screen  
  
 player(playerx**,** playery)  
 pygame.display.update()

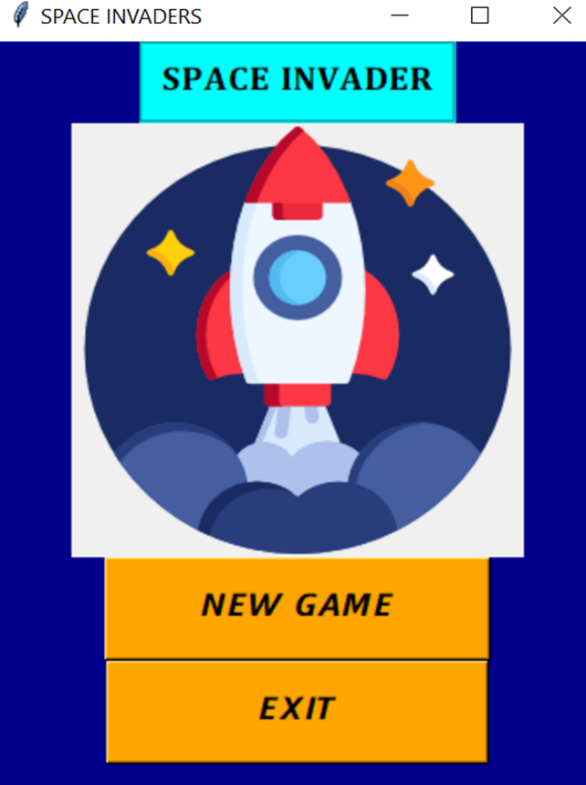
**Exit\_screen:**

(Exit\_screen.py)

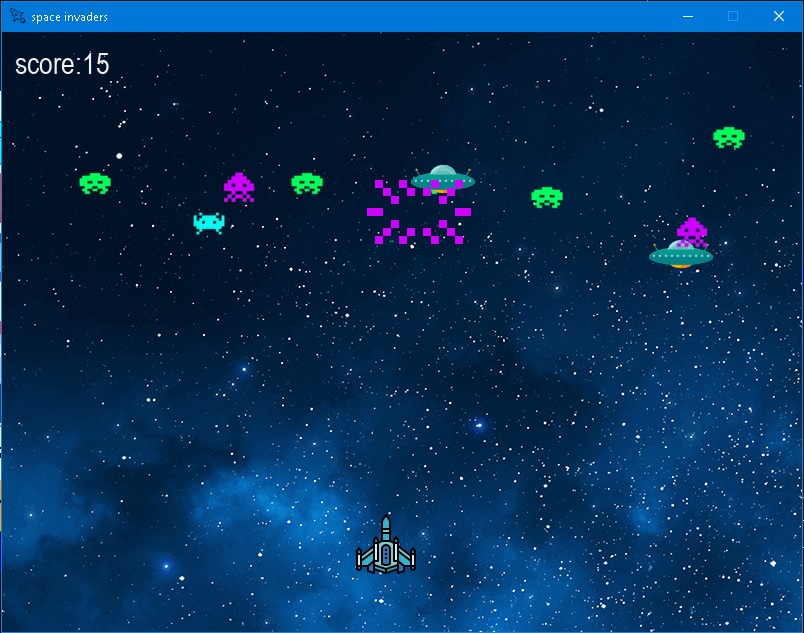
from tkinter import \*  
import Game\_loop  
  
def play\_again():  
 import Game\_loop  
  
def game\_over():  
 rt = Tk()  
 rt.geometry("350x450")  
 rt.configure(bg="orange")  
  
  
 label1 = Label(rt**,** text="GAME OVER"**,** padx=**25,** pady=**15,** bg="orange"**,** fg="black"**,** font=("Arial Bold"**, 20**))  
 label1.pack()  
  
 label1.place(relx=**0.2,** rely=**0.4**)  
  
 but1 = Button(rt**,** text="exit"**,** padx=**30,** pady=**12,** command=quit**,** font=("helvetica bold"**, 12**)**,** bg="purple"**,** fg="white")  
 but1.pack()  
 but1.place(relx=**0.4,** rely=**0.7**)  
  
 but2 = Button(rt**,** text=" play again"**,** padx=**30,** pady=**12,** font=("Arial bold"**, 12**)**,** bg="purple"**,** fg="white"**,** command=play\_again)  
 but2.pack()  
 but2.place(relx=**0.3,** rely=**0.2**)  
  
 rt.mainloop()

**output:**

**Start Window Exit window**

** **

**Game Window**

****

**LIST OF THE TOPICS/ CONCEPT WHICH ARE COVERED FROM THE SYLLABUS:**

1. Importing various modules such as pygame, math , random.
2. Function declaration.
3. Declaring function parameters.
4. Function calling.
5. Use of while loop.
6. Use of for loop.
7. If else loop.
8. Use of Nested loops.
9. Basic functions and methods like Append , print .
10. Use of logical & arithmetic operators and lists.

**LIST OF THE TOPICS/ CONCEPT WHICH ARE OUT OF SYLLABUS:**

Various functions related to **pygame module** used to draw on the screen , to load the images , to define event types and their outcomes .

1. Pygame.display
   1. Pygame.display.update
   2. Pygame.display.set\_mode
   3. Pygame.display.set\_caption
2. Pygame.image
   1. Pygame.image.load
3. Pygame.screen.blit

blit function used to draw on screen

1. Pygame.event.type

Use of math module for measuring the distance between two points .

Use of random module for giving random coordinates to the enemy spaceships

1.Tkinter module

1. Root function
2. Creation of widgates

**CONCLUSION:**

All in all, building a game for our first year project was much more fun and even challenging than building other projects with databases.

We began the project with absolutely no idea on how graphics work, how a collision detection system works or even how to display an image on the screen and, by the end of the project, we got to a point where we could create our own or modify existing game specific algorithms to suit my needs. We learn many different module of python and also learned how to make a project in team.

In the end, we can conclude that, even though Python is not highly recommended programming language for game development, but from this project we can conclude that it can be comparatively useful in its own way due to availability of various build-in modules such as pygame and tkinter (for graphics).

**REFERENCES:**

* Making Games with Python & Pygame

1. By Al Sweigart
2. ISBN (978-1469901732) 1st Edition

* Python Basics - Youtube Channel
  1. <https://youtube.com/c/PythonBasics>
  2. Used for pygame learning.