GAME DEVELOPMENT

<u>CERTIFICATE</u>

This is to certify that the project report on the topic of **PYTHON GAME DEVELOPMENT** has been successfully completed by **C. Tejesh, V. Venkatesh, R. Yognikhil** of **CLASS XII A** under the guidance of **Mrs. Ajitha madam [PGT computer science]** and **Mr. Aman Jain sir [PGT computer science]** in particular fulfilment of central board of secondary education **[CBSE]** leading to the award of annual examination of the year 2023-2024.

Internal examiner

External examiner Principal

ACKNOWLEDGEMENT

We would like to express our sincere gratitude to our guide **Mrs. Ajitha madam (PGT Computer Science)** for providing their invaluable guidance, comments and suggestions throughout the period of our work. We would like to specially thank **Mr. Aman Jain sir (PGT Computer Science)** for guiding us and helping us with the procedure to be followed for the fulfilment of our project.

We would also like to thank our principal **Smt. P Anuradha** forgiving us the opportunity to do this project and including it as a part of curriculum.

TABLE OF CONTENTS

INTRODUCTION

What is gaming?

Gaming is playing an electronic video game, which is often done on a dedicated gaming console, PC or smartphone. People who often play video games are called gamers.

A recent article in The Computer Games Journal made a remarkable claim:

> *"During the last five decades, videogames evolved into a major component of popular culture as well as a multi-billion-dollar industry. The medium diversified tremendously, currently encompassing simple implementations of numeric games on the screen of a cell phone as well as vast, persistent online worlds on last generation consoles and PCs. In spite of its cultural and economic relevance, few attempts have been made to define what a videogame exactly is. (Emphasis added)"*

Fifty Years On discusses videogames. As the author notes, there are alternative terms that could be used, like computer game or console game. Everyone will mostly use the word 'game'; this should be read to include what Fifty Years On discusses as videogames. This is not because I think that videogames and games are the same thing but because the process of defining both is similar and benefits from the same considerations.

Games and videogames may not be the same thing, but the definitions of both are definitions, and therefore understanding definitions helps in making definitions for both. The problems in the definition literature of both also tend to be similar, so discussing them together is useful in clarifying issues that appear in relation to both. Regardless of which one you are trying to define, this paper points out potential pitfalls and suggests solutions.

How popular is gaming?

Gaming is extremely popular activity worldwide. The number of people who play games is estimated to be over 3 billion. Some estimates put the worldwide gaming market value between $180 billion and $220 billion in 2022. This is about double the market for the global film industry. The majority of the gaming market is people playing games on their smartphones.

Introduction to gaming:

A project is a time-bound exercise undertaken by on organisation to obtain a product, service or result. “Project management” is defined by the PMBoK (Project Management Body of Knowledge) as “a temporary endeavour to create a unique product, service or result”. The concept of project management has evolved through several schools of thought, and uses a range of theories originating from mathematics, computer science, economics, and other related fields in its modelling and analysis. Sensible decision-making is critical for the success of project. Every project begins with a decision: the decision to invest.

Often, an investor has several investment options, and each option will result in a different project, and thus one of the investment options has to be chosen before the project charter can be produced. Similarly, any large project involving subcontractors, for instance, a construction project, has a complex interplay between the main contractor (the project manager) and subcontractors, or among the subcontractors themselves, which typically has several decision points. For example, if there is an ambiguity in the contract between the contractor and subcontractor, each must decide how hard to push their case without jeopardising the whole project, and thus their own stake in it.

Similarly, when projects from competing organisations are launched, the marketing personnel have to decide what is the best timing and strategy to market the project, or its resultant product or service, so that it can gain maximum traction in the face of competition. In each of these scenarios described above, the required decisions depend on the decisions of others who, in some way, have competing interests to the interests of the decision-maker.

Project management uses a range of concepts and tools in decision-making. These include investment analysis methods such as force field analysis, the life cycle cost method, internal rate of return etc., and other tools such as utility theory, prospect theory, Net Present Value (NPV) method, Monte Carlo analysis, linear programming, queueing theory and so on. Recently, game theory has been gaining prominence as a tool useful in decision-making in project management scenarios.

Compared to the other tools mentioned above, game theory is particularly useful in scenarios where a number of entities are trying to achieve the same outcome (either in competition with each other, or in cooperation with each other), but have independent and rational decision-making abilities. Thus, it is especially useful in decision-making in the face of competition, particularly in scenarios such as those described above.

Game theory also offers a rigorous mathematical framework, and has been successfully used in fields such as economics, social science, biology and computer science, presenting many precedents and examples for project management researchers to follow. Thus, papers analysing project management scenarios have arisen in the past decade, to use game theory in their modelling. It is, however, a nascent field, and the purpose of this review is to advocate for the widespread use of game theory as a modelling and analysis tool in project management, by summarising the state-of-the-art in this niche, classifying the existing works and highlighting gaps in the literature and opportunities for future research.

There is a considerable (but not overwhelming) body of work which has attempted to use game theory in modelling project management scenarios. There is even a hypothesis (the "Bilton and Cummings" hypothesis) that states, "the use of game theory makes it possible to understand the needs and interests of the involved persons in a better way and to finalize the project successfully".

Papers which use game theory in project management relate to construction projects, information and communications technology (ICT) projects and projects from other domains. They use various types of games, and model a diverse range of players such as governments, project managers/contractors, subcontractors and clients. Therefore, a well-structured review of the field becomes necessary to comprehend the state-of-the-art, classify existing studies and identify gaps in the literature.

To our knowledge, such a review spanning several application domains yet focusing exclusively on project management and games does not exist. Indeed, there are works such as Kaplinski and Tamosaitiene, which focus on reviewing the work of individual authors or research groups or reviewing aspects of the use of game theory in operations research, which has some relevance to project management; still, a structured review focusing on the use of game theory in project management in all its domains and applications is lacking. Therefore, we present such a review here.

We select papers that use game theory in project management scenarios from the Scopus database using a rigorous selection process, and review and analyse these papers in great detail. We also analyse the relative impact of each paper to the particular niche of "game theory in project management", and to project management research is general. We introduce and use a complex multidimensional, yet principled, classification scheme that helps to highlight the areas where most effort has been exerted to date and areas where there are gaps in the literature.

We also consider citation networks of the papers in the niche, and show how these networks can be related to and explained by the classification scheme that we present. Our review and analysis highlight why game theory is a very useful tool to model project management scenarios, and in what further ways it could be applied in project management contexts in the future. In summary, the significance of this review is attested by the following aspects.

- We follow a comprehensive and principled method for searching and filtering relevant papers.

- We review papers across several disciplines, such as construction, ICT and education, and highlight the similarities and differences between them in their application of game theory in project management.

- We present a detailed multidimensional classification of the papers that we have reviewed.

• We present and analyse the citation network of the papers we have reviewed, highlighting their interdependency and relative impact.

• We identify gaps in the literature that point to potentially fruitful future research directions.

Game Theory :

Game theory, which is the study of strategic decision making, was first developed as a branch of microeconomics. However, later, it has been adopted in diverse fields of study, such as evolutionary biology, sociology, psychology, political science and computer science.

Game theory is used to study many phenomena and behavioural patterns in human societies and socio-economical systems, such as the emergence and sustaining of cooperation in communities and organisations, modelling of unethical or criminal behaviour or the decision-making processes involved in vaccination against epidemics.

Game theory has gained such wide applicability due to the prevalence of strategic decision-making scenarios across different disciplines. A typical game defined in game theory has two or more players, a set of strategies available to these players, and a corresponding set of pay-off values (sometimes called utility values) for each player (which are, in the case of two-player games, often presented as a pay-off-matrix).Game theory can be classified into two broad domains: non-cooperative game theory and cooperative game theory.

Non-Cooperative Games and Cooperative Games:

Typically, games are played for the self-interest of the players, even when the players cooperate; cooperation is the best strategy under the circumstances to maximise the individual pay-offs of the players. In such games, the cooperative behaviour, if it emerges, is driven by selfish goals and is transient.

These games can be termed “non-cooperative games”. Non-cooperative game theory is the branch of game theory that analyses such games. On the other hand, in a cooperative game, or coalitional game, players form coalitions, or groups, often due to external enforcement of cooperative behaviour, and competition is between these coalitions.

Cooperative games are analysed using cooperative game theory, which predicts the coalitions that will form and the pay-offs of these coalitions. Cooperative game theory focuses on surplus or profit sharing among the coalition members, where the coalition is guaranteed a certain amount of pay-off by virtue of the coalition being formed.

Often, the outcome of a cooperative game played in a system is equivalent to the result of a constrained optimisation process, and therefore many of the papers we review use a linear programming framework to solve the cooperative games they model.

Objectives of the Gaming System:

The game is developed for full-time entertainment and enthusiasms. It teaches the Gamer to be alert at every situation he/she faces, because if the Gamer is not fully alert and notice the saucer fire he/she must be hit by the saucer-bombs.

Though the proposed game is an action game, it doesn't involve direct violence. No zombie killing, animal killing, or human killing is performed in the game. So it can also be viewed as a nonviolence game.

Kids can also play this game, because the design of the game is very simple, controlling the game is very easy – pressing some neighbouring keys of the keyboard.

Python:

The game is developed using Python. Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built-in data structures, combined with dynamic typing and dynamic binding. Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms and can be freely distributed.

Pycharm:

PyCharm is a dedicated Python Integrated Development Environment (IDE) providing a wide range of essential tools for Python developers, tightly integrated to create a convenient environment for productive Python, web, and data science development.

Pygame:

Game programming is very rewarding nowadays and it can also be used in advertising and as a teaching tool too. Game development includes mathematics, logic, physics, Al, and much more and it can be amazingly fun. In python, game programming is done in pygame and it is one of the best modules for doing so.

DESCRIPTION OF GAME

Space invaders game:

Space Invaders is a fixed shooter in which the player moves a laser cannon horizontally across the bottom of the screen and fires at aliens overhead. The aliens begin as five rows of eleven that move left and right as a group, shifting downward each time they reach a screen edge.

Logic of game:

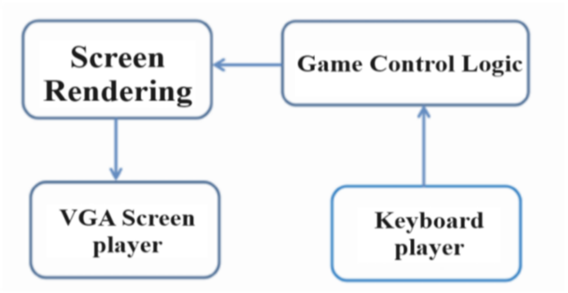
The current the space invaders game is a single player. The game interference is built in 2D format. The game is displayed in the pygame window. In the static shooting Space Invaders game, the player moves a laser weapon horizontally across the bottom of the screen to fire at the aliens above them. The aliens move in a group left and right, shifting lower as they get closer to the edge of the screen. To kill every alien by gunfire is the goal. In other words, avoid being killed in order to launch a counterattack on the invaders. If the invaders (aliens) reach the bottom of the screen the game ends immediately, while the player still has three lives.

Game analysis:

During the course of completion of this project work, the complete analysis of proposed system was done. In the analysis task, a complete care about the feasibility of the proposed system was taken. The IDLE which is editor which comes along with python during installation can be used for development. PyCharm is an IDE which is quite suitable for developing python games. Using of PyCharm made our code beautiful and more attractive for the users.

- **Software flow:**

  In terms of software flow, the space invader game can be shown as:



- **Hardware and software interface:**

  The basic hardware modules are keyboard and VGA screen.

  For installing Python

  Processor:- Single core or multiple core processor
  RAM:- 1 Gb or more64 Mb graphics
  ROM:- 40 Gb (for OS and its requirements)
  Operating System:- Windows ,Mac OS and Linux

  For installing PyCharm

  Windows:

  - Microsoft Windows 10/8/7/Vista/2003/XP (incl.64-bit)
  - 1 GB RAM minimum
  - 2 GB RAM recommended
  - 1024x768 minimum screen resolution
  - Python 2.4 or higher, Jython, PyPy or IronPython

  Mac:

  - Mac OS X 10.8 or higher
  - 1 GB RAM minimum
  - 2 GB RAM recommended
  - Python 2.4 or higher, Jython, PyPy or IronPython

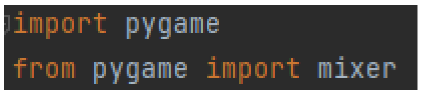
  Linux:

  - 512 MB RAM minimum
  - 1 GB RAM recommended
  - 1024x768 minimum screen resolution
  - Python 2.4 or higher, Jython, PyPy or IronPython

Modules used in game:

- **Pygame module:**

  Pygame is a cross-platform set of Python modules designed for writing video games. It includes computer graphics and sound libraries designed to be used with the Python programming language.



```
import pygame
from pygame import mixer
```

A mixer in pygame module is used for loading and playing sounds.

- **Math module:**

  The math module is used to perform various mathematical calculations, such as numeric, trigonometric, logarithmic, and exponential calculations.

```
import math
```

  Here, math module used to calculate the distance between bullet and enemy.

```
distance = math.sqrt(math.pow(enemyX - bulletX, 2) + (math.pow(enemyY - bulletY, 2)))
```

- **Random module:**

  The Python Random module is a built-in module for generating random integers in Python.

```
import random
```

- **Sys module:**

  The python sys module provides functions and variables which are used to manipulate different parts of the Python Runtime Environment.

```
import sys
```

  Here, sys module is used to increase the integer string conservation length.

```
sys.set_int_max_str_digits(30000)
```

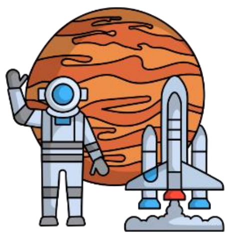
Files used in this project:

- **background.mp3:**

  Audio file used as a background sound.

- **ufo.png:**

  Icon used as a game icon and also intro screen.



- **background.png:**

  Image used as a background.

- **bullet.png:**

  Image used as a bullet.

- **enemy.png:**

  Image used as enemy.

- **explosion.wav:**

  Audio used as a blast sound when enemies were killed.

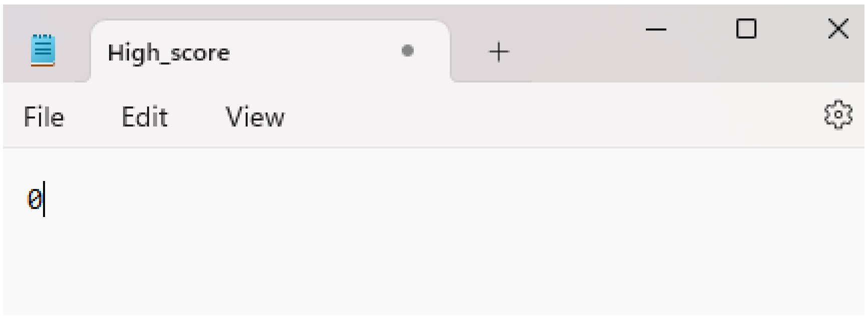
- **Player.png:**

  Image used as space ship which attacks the enemies.

- **laser.wav:**
  Audio used as a sound of the firing of bullet by spaceship.

- **VeryDamaged.ttf:**
  Font used a text in the game like score, title, high score, etc

- **High_score:**
  Text file with a text ‘0’ at the very beginning of the game. And used to store high score permanently.



0

SOURCE CODE

Main.py:

```
import pygame
from pygame import mixer
import math
import random
import sys

# Intialize the pygame
pygame.init()

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

# Background
background = pygame.image.load('background.png')
background_X = 0
background_Y = 1200

# Sound
mixer.music.load("background.mp3")
mixer.music.play(-1)

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

# Player
playerImg = pygame.image.load('player.png')
playerX = 370
playerY = 480
playerX_change = 0

# 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(30)

# bullet
bulletImg = pygame.image.load('bullet.png')
bulletX = 0
bulletY = 480
bulletX_change = 0
bulletY_change = 10
bullet_state = "ready"

# Score
score_value = 0
verydamaged_font = pygame.font.Font('veryDamaged.ttf', 32)
textX = 10
testY = 10

# High score
high_score_file = open('High_score','r')
high_score_value_str = high_score_file.readline()
sys.set_int_max_str_digits(30000)
high_score_value = int(high_score_value_str)

# Game Over
over_font = pygame.font.Font('VeryDamaged.ttf', 130)

# Intro
intro_img = pygame.image.load('ufo.png')
intro_img_X = 150
intro_img_Y = 50
skip_text = pygame.font.Font('freesansbold.ttf', 25)
title_font = pygame.font.Font('VeryDamaged.ttf', 100)
title_font_X = 1000
title_font_Y = 200
```

```
def show_intro(x, y):
    screen.blit(intro_img, (x, y))


def show_skip():
    skip_show = skip_text.render("##PRESS 'SPACE' TO SKIP##", True, (0,
250, 0))
    screen.blit(skip_show, (225, 550))

def show_title(x, y):
    title_text = title_font.render('SPACE INVADERS', True, (236, 21, 21))
    screen.blit(title_text, (x,y))

def show_score(x, y):
    score = verydamaged_font.render("Score : " + str(score_value), True, (0,
255, 0))
    screen.blit(score, (x, y))

def show_high_score(high_score_value):
    high_score = verydamaged_font.render("High Score : " +
str(high_score_value), True, (0, 255, 0))
    screen.blit(high_score, (550, 10))

def game_over_text():
    over_text = over_font.render("GAME OVER", True, (21,255, 0))
    screen.blit(over_text, (100, 200))


def player(x, y):
    screen.blit(playerImg, (x, y))


def enemy(x, y, i):
    screen.blit(enemyImg[i], (x, y))


def fire_bullet(x, y):
    global bullet_state
    bullet_state = "fire"
    screen.blit(bulletImg, (x + 16, y + 10))


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



# Intro
running1 = 2
while running1 < 3:
    # RGB = Red, Green, Blue
    screen.fill((0, 0, 0))
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            running1 = 5
        elif event.type == pygame.KEYDOWN:
            if event.key == pygame.K_SPACE:
                running1 = 7

    intro_img_Y -= 0.5

    if title_font_X < 65:
        title_font_X = 65
    else:
        title_font_X -= 0.5

    if background_Y < 0:
        background_Y = 0
        title_font_Y -= 0.1
    else:
        background_Y -= 0.5

    if title_font_Y < -100:
        running1 = 7
    screen.blit(background, (background_X, background_Y))
    show_intro(intro_img_X, intro_img_Y)
    show_title(title_font_X, title_font_Y)
    show_skip()
    pygame.display.update()

# Game Loop
running2 = True
while running2:

    if running1 == 5:
        running2 = False

    # 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:
            running2 = False

        # if keystroke is pressed check whether its right or left
        if event.type == pygame.KEYDOWN:
            if event.key == pygame.K_LEFT:
                playerX_change = -1
            if event.key == pygame.K_RIGHT:
                playerX_change = 1
            if event.key == pygame.K_SPACE:
                if bullet_state == "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

    playerX += playerX_change
```

```
if playerX <= 0:
    playerX = 0
elif playerX >= 736:
    playerX = 736

# 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] = 0.5
        enemyY[i] += enemyY_change[i]
    elif enemyX[i] >= 736:
        enemyX_change[i] = -0.5
        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)
if bulletY <= 0:
    bulletY = 480
    bullet_state = "ready"

if bullet_state == "fire":
```

```
            fire_bullet(bulletX, bulletY)
            bulletY -= bulletY_change

        if score_value >= high_score_value:
            high_score_value = score_value

        player(playerX, playerY)
        show_score(textX, testY)
        show_high_score(high_score_value)
        pygame.display.update()

    high_score_file.close()
    if score_value >= high_score_value:
        high_score_file = open('High_score','w')
        high_score_file.write(str(score_value))
```

Process:

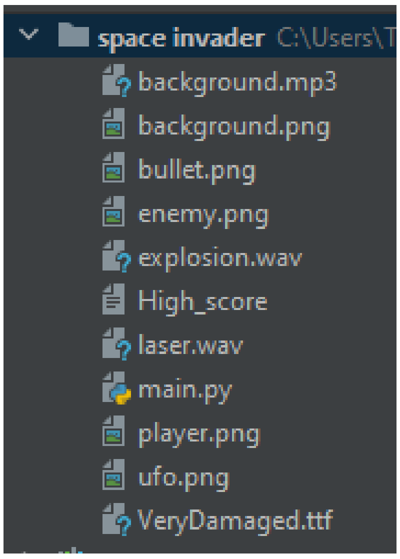
1. Install **PyCharm** and create the project as ‘**SPACE INVADERS**’. And add python file as main.py in it.

2. Load all files which we have mentioned in the ‘DESCRIPTION OF GAME’ into that project.

3. Install **pygame**.

4. Copy the source code above and paste in the **main.py** in that project.

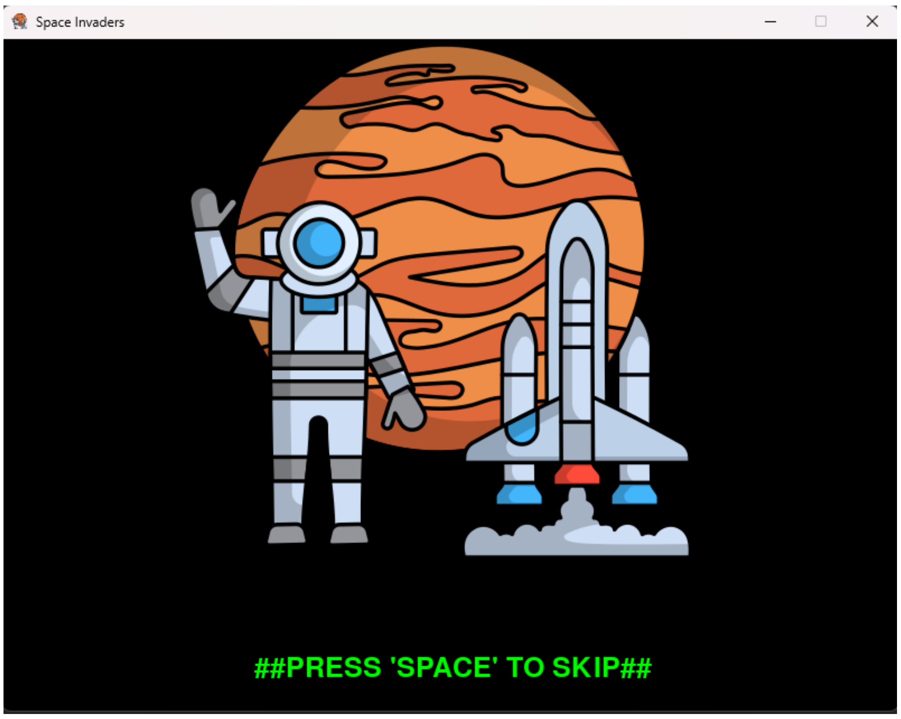
5. Make sure all the above files are named as it is to avoid error.

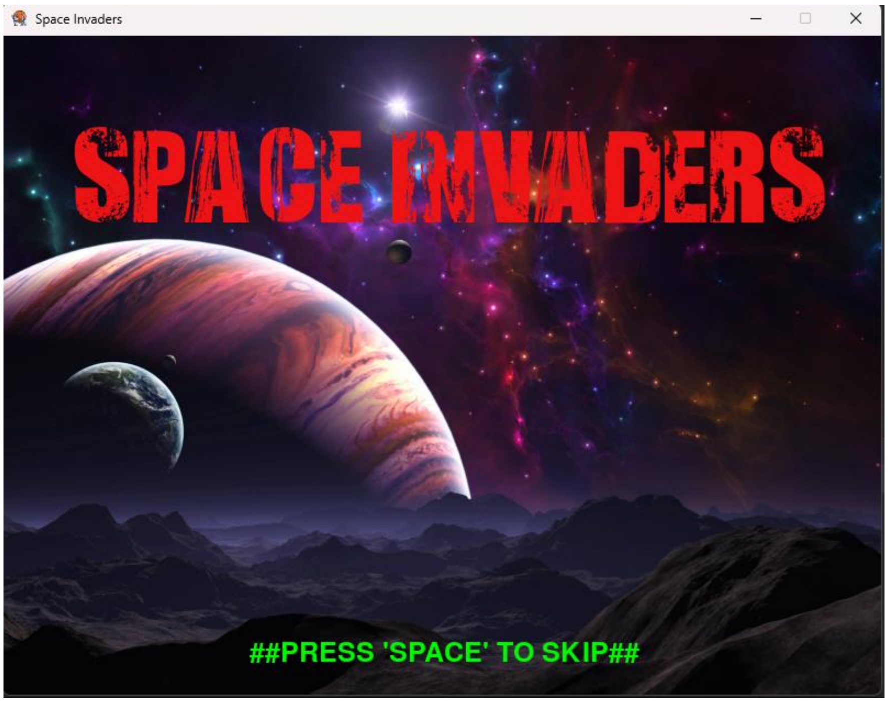


```
space invader  C:\Users\T
    background.mp3
    background.png
    bullet.png
    enemy.png
    explosion.wav
    High_score
    laser.wav
    main.py
    player.png
    ufo.png
    VeryDamaged.ttf
```

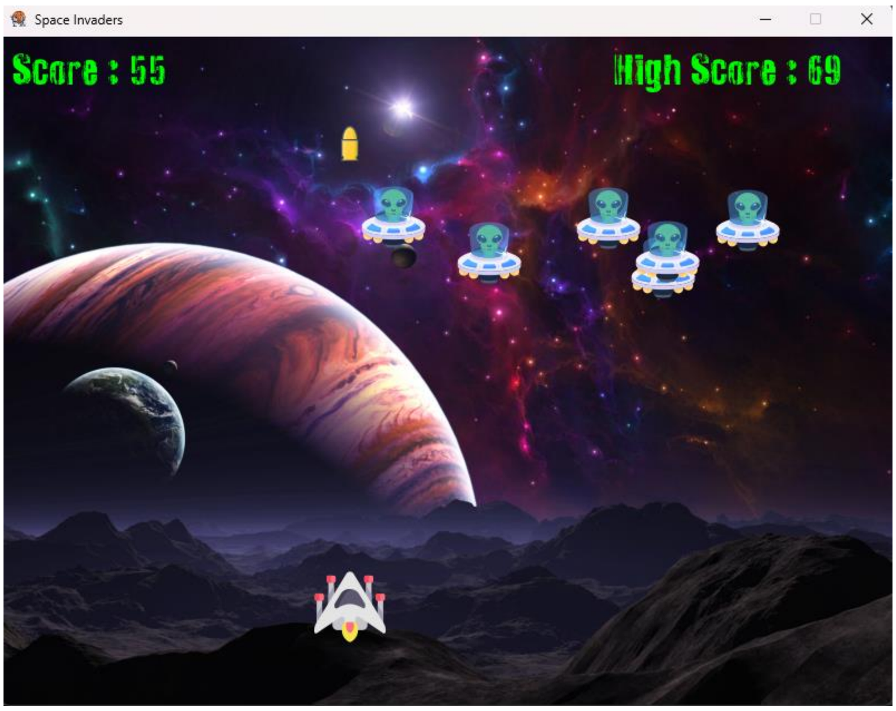
OUTPUT

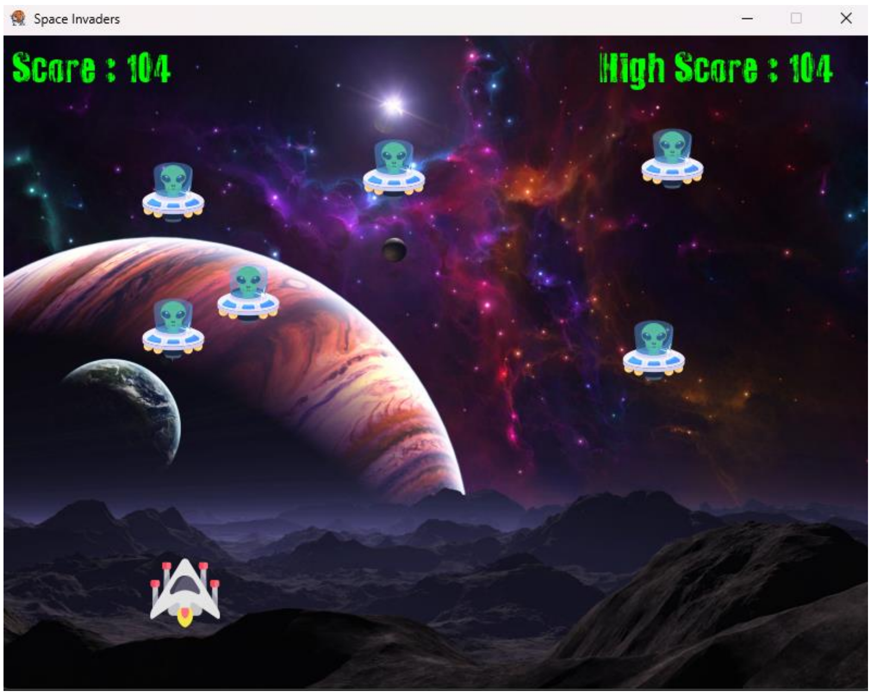
1. Screen shots of output of the introduction of the game:-



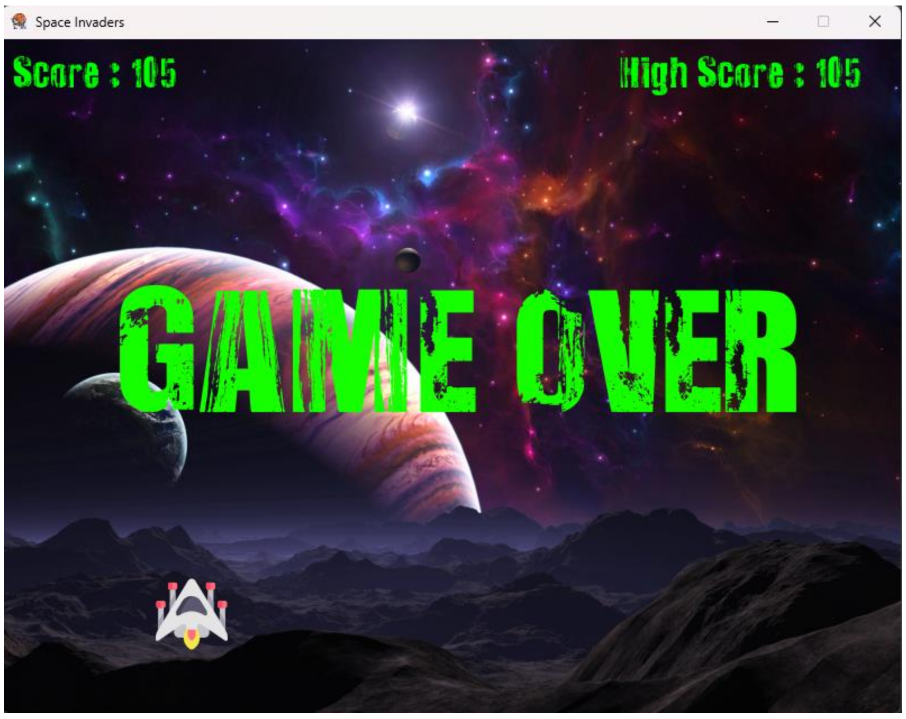


2. *Screen shots of output of the game:-*





3. Screen shot of output of game over:



CONCLUSION

Therefore by using the language python, IDE PyCharm and IDLE, the python libraries such as pygame for multimedia and sound the game has been created and is functioning to the level that it meets its objective. The Game worked on different OS and different Specification computers and delivered the same performance on all . There may be some areas of improvements and efficiency which will be carried out in future.

REFERENCES

- www.python.org
- www.jetbrains.com/pycharm/
- www.pygame.org/news
- www.flaticon.com
- https://cbsepython.in
- https://github.com/attreyabhatt/Space-Invaders-Pygame/commit/master
- www.fesliyanstudios.com
- www.bgmringtones.com

THANK YOU

TEAM MEMBERS:-

1. **C. TEJESH (TEAM LEADER)**
2. **V. VENKATESH**
3. **R. YOGINIKHIL**