SUN INTERNATIONAL SCHOOL

INFORMATICS PRACTICES PROJECT

SPACE INVADERS

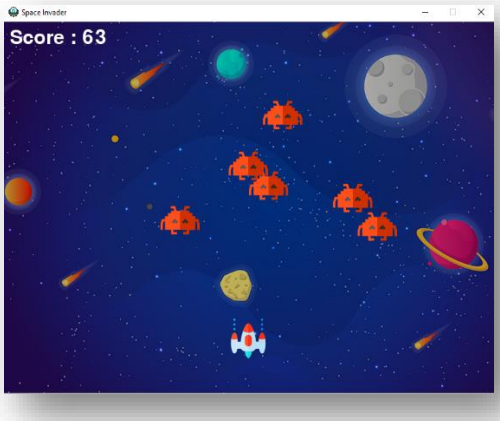
Submitted to-

Nishtha Chaubey Vyas Ma'am

Submitted by-

Siddhant Nagaria – XII-A

Priyanshu Agarwal – XII-C



PREFACE

With the advancement of technology, rapid globalization, diversified and multifaceted stimulants, children and young people are commonly intelligent and quick-minded.

As a part of the course curriculum and to deepen and widen the practical knowledge in the concept of "Python coding", we have made a project on the same.

Working on this project, we came to know and understand various aspects of the topic "Game - Space Invaders", at the same time the very importance of creativity and knowledge in student's life.

We have expressed our experiences in our own simple way. We hope who goes through it will find it interesting and worth reading. All constructive criticism and feedback is cordially invited.

Siddhant Nagaria - XII-Science

Priyanshu Agarwal - XII - Commerce

ACKNOWLEDGEMENT

We are very thankful to everyone who all supported me, for we have completed our project effectively and moreover on time. We are overwhelmed in all humbleness and gratefulness to acknowledge our depth to all those who helped us to put these ideas well. We are equally grateful to our computer teacher - Mrs. Nishtha Mam. They gave us moral support and guided us in different matters regarding our Project. With the help of their valuable suggestions, guidance and encouragement, we were able to perform this project work. Last but not the least, we would like to thank our parents who helped us a lot in gathering different information, collecting data and guiding us from time to time. Despite of their busy schedules, they gave us their precious time in making this project unique.

Siddhant Nagaria - XII-Science

Priyanshu Agarwal - XII - Commerce

INDEX

S.No.	Particulars	Page no.
1	Cover Page	
2	Preface	1
3	Acknowledgement	2
4	Index	3
5	Introduction	4
6	Python Basics	5
7	Functions/Modules	8
8	Special module	9
9	Code	10
10	Output	17
11	Future Scope	21
12	References	22

INTRODUCTION

Hello everyone..!!

We have made a fabulous interesting game - Space Invaders. It is a spaceship shooting game.

What is Space Invaders ?

Space Invaders is considered one of the most influential video games of all time. It helped expand the video game industry from a novelty to a global industry, and ushered in the golden age of arcade video games. It was the inspiration for numerous video games and game designers across different genres, and has been ported and re-released in various forms.

What is the uses of Space Invaders?

The objective of Space Invaders, which was one of the earliest video games released, is to pan across a screen and shoot descending swarms of aliens, preventing them from reaching the bottom of the screen. It is viewed as a pioneer of modern gaming.

Space Invaders is a very simple game by modern standards, but it was a technological marvel in its time. Typical levels consist of a player piloting a laser cannon to battle columns of descending aliens while using shields to block alien fire. The speed of the alien approach increases as the game progresses, adding to the tension.

PYTHON BASICS

What is Python ?

Python is an interpreted, high-level and general-purpose programming language. Created by Guido van Rossum and first released in 1991, Python's design philosophy emphasizes code readability with its notable use of significant whitespace. Its language constructs and object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects.

Python is dynamically typed and garbage-collected. It supports multiple programming paradigms, including structured (particularly, procedural), object-oriented, and functional programming. Python is often described as a "batteries included" language due to its comprehensive standard library.

Python was created in the late 1980s as a successor to the ABC language. Python 2.0, released in 2000, introduced features like list comprehensions and a garbage collection system with reference counting.

Basic Structure of Python Code –

In General Python Program Consists of so many text files, which contains python statements. Program is designed as single main, high file with one or more supplement files

In python high level file has important path of control of your Program the file, you can start your application. The library tools are also known as Module files. These tools are implemented for making collection of top - level files. High level files use tools which are defined in Module files. And module files will implement files which are defined in other Modules. Coming to our point in python a file takes a module to get access to the tools it defines. And the tools made by a module type. The final thing is we take Modules and access attributes to their tools. In like manner this shows Programming structure of Python

Attributes and Imports:

The struct**ure Python Program consists of** three files such as : a.py,b.py and c.py. The file model a.py **is chosen for high level file,** it is known as a simple text file of statements. And it can be executed from bottom to top when it is launched. Files b.py and c.py are modules. They are calculated as better text files of statements as well. But **they are generally not started d**irectly. Identically this attributes define Programming structure of Python.

Functions:

For example b.py defines a function called spam. For external use. B.py has python def statement to start the function. Later operated by passing one or more values like the below.

Def spam(text): print text, 'spam'

If a.py wants to use spam, it has python statements like below

Import b b.spam ('gumby')

Statements:

Python import statement gives file a.py access to file b.py. it shows that "load fileb.py" and gives access to all its attributes by name b" import statements will execute and implement other file for at run-time. In python cross file module is not updated until import statements are executed.

The next part is statements will call the function spam. module b used by object attribute notation. B.spam means get value of name spam within object b. And we can implement a string in parenthesis if these files run by a.py.

In regular if we see object. Attribute in total python scripts. Many objects have attributes traced by "python operators".

The process of Importing considered as general in total python. Any sort of file can get tools from any file. Getting of chains can be go as deep as you can. By this Instance you will get it notified module a can import b and b can Import c, and c

again Imports b. correspondingly this statements include Programming structure of Python

Modules:

If we take this as a part, python serves as biggest company structure. Modules are having top end of code. By coding components in module files used in any program files. If we take an example function b.spam is regular purpose tool. We can again implement that in a different program. This is simply known as b.py from any other program files.

Standard library files:

Python has large collection of modules known as standard library.it contains 200 modules at last count. It is platform independent common programming works. Such as GUI Design, Internet and network scripting. Text design matching, operating system Interfaces. So, comparatively all the Above will explain Programming structure of Python.

Prerequisites:

There are nothing additional prerequisites required in order to Learn Python course. So, It's better to have a basic knowledge of programming languages like C (or) Java.

FUNCTIONS / MODULES USED IN OUR PROJECT

Functions used are -

1. Import
2. Range
3. Def
4. While
5. If
6. Elif

Modules used are –

1. Math
2. Random
3. Pygame

SPECIAL MODULE – PYGAME

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.

Pygame uses the Simple Direct Media Layer (SDL) library, with the intention of allowing real-time computer game development without the low-level mechanics of the C programming language and its derivatives. This is based on the assumption that the most expensive functions inside games can be abstracted from the game logic, making it possible to use a high-level programming language, such as Python, to structure the game.

Other features that SDL doesn't have include vector math, collision detection, 2d sprite scene graph management, MIDI support, camera, pixel-array manipulation, transformations, filtering, advanced free type font support, and drawing.

Applications using pygame can run on Android phones and tablets with the use of pygame Subset for Android (pgs4a). Sound, vibration, keyboard, and accelerometer are supported on Android

.

CODING

The code of our game – Space Invaders is –

```
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)
```

```
24.
25.     # Player
26.     playerImg = pygame.image.load('player.png')
27.     playerX = 370
28.     playerY = 480
29.     playerX_change = 0
30.
31.     # Enemy
32.     enemyImg = []
33.     enemyX = []
34.     enemyY = []
35.     enemyX_change = []
36.     enemyY_change = []
37.     num_of_enemies = 6
38.
39.     for i in range(num_of_enemies):
40.         enemyImg.append(pygame.image.load('enemy.png'))
41.         enemyX.append(random.randint(0, 736))
42.         enemyY.append(random.randint(50, 150))
43.         enemyX_change.append(4)
44.         enemyY_change.append(40)
45.
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"
57.
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)
68.
69.
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))
92.
93.
94.     def isCollision(enemyX, enemyY, bulletX, bulletY):
95.         distance = math.sqrt(math.pow(enemyX - bulletX, 2) +
    (math.pow(enemyY - bulletY, 2)))
96.         if distance < 27:
97.             return True
98.         else:
99.             return False
100.
101.
102.    # Game Loop
103.    running = True
104.    while running:
105.
106.        # RGB = Red, Green, Blue
107.        screen.fill((0, 0, 0))
108.        # Background Image
109.        screen.blit(background, (0, 0))
110.        for event in pygame.event.get():
```

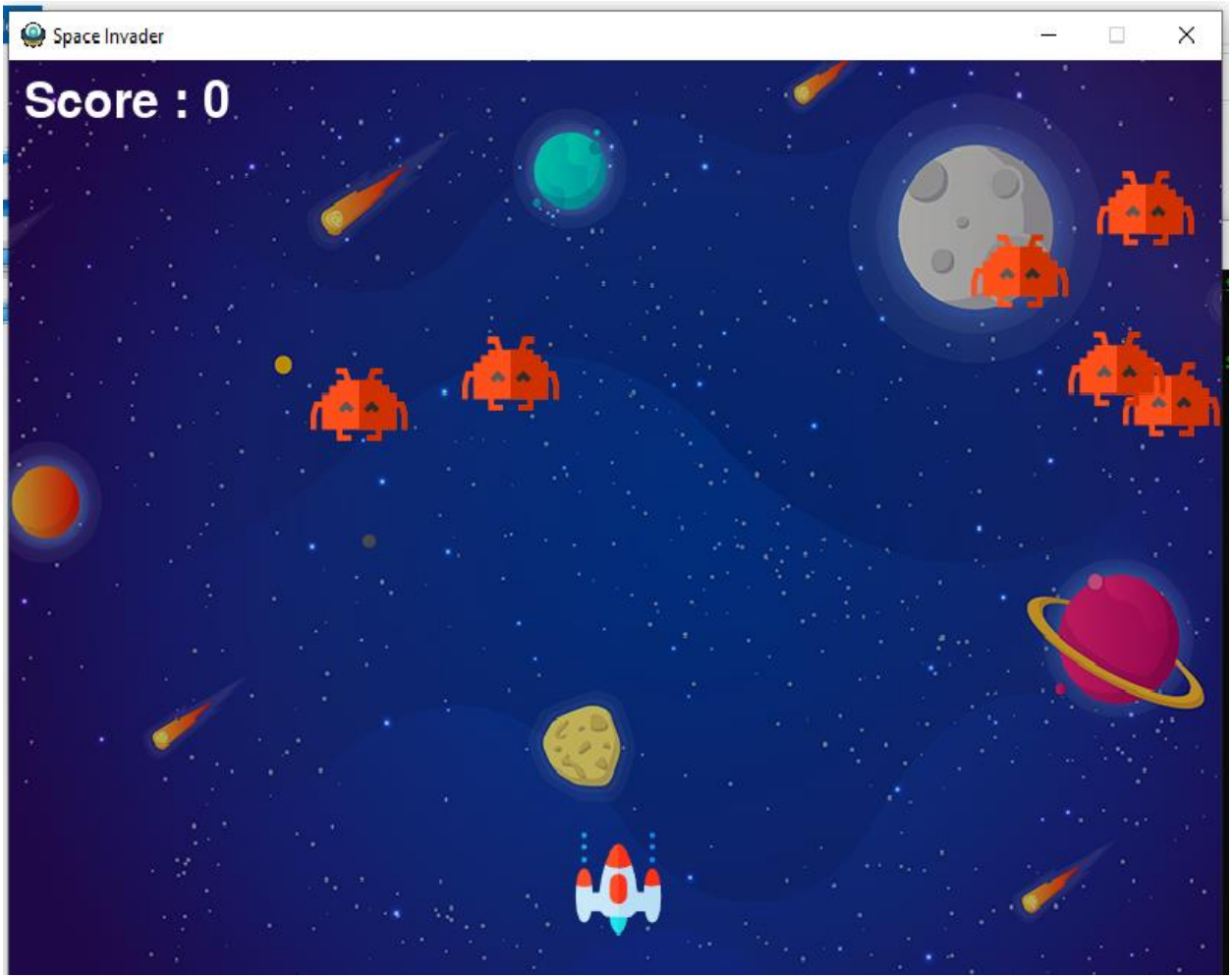
```
        if event.type == pygame.QUIT:
            running = False

        # 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

    # 5 = 5 + -0.1 -> 5 = 5 - 0.1
    # 5 = 5 + 0.1

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

```
140.
141.        # Enemy Movement
142.        for i in range(num_of_enemies):
143.
144.            # Game Over
145.            if enemyY[i] > 440:
146.                for j in range(num_of_enemies):
147.                    enemyY[j] = 2000
148.                game_over_text()
149.                break
150.
151.            enemyX[i] += enemyX_change[i]
152.            if enemyX[i] <= 0:
153.                enemyX_change[i] = 4
154.                enemyY[i] += enemyY_change[i]
155.            elif enemyX[i] >= 736:
156.                enemyX_change[i] = -4
157.                enemyY[i] += enemyY_change[i]
158.
159.            # Collision
160.            collision = isCollision(enemyX[i], enemyY[i], bulletX,
    bulletY)
161.            if collision:
162.                explosionSound = mixer.Sound("explosion.wav")
163.                explosionSound.play()
164.                bulletY = 480
165.                bullet_state = "ready"
166.                score_value += 1
167.                enemyX[i] = random.randint(0, 736)
168.                enemyY[i] = random.randint(50, 150)
```

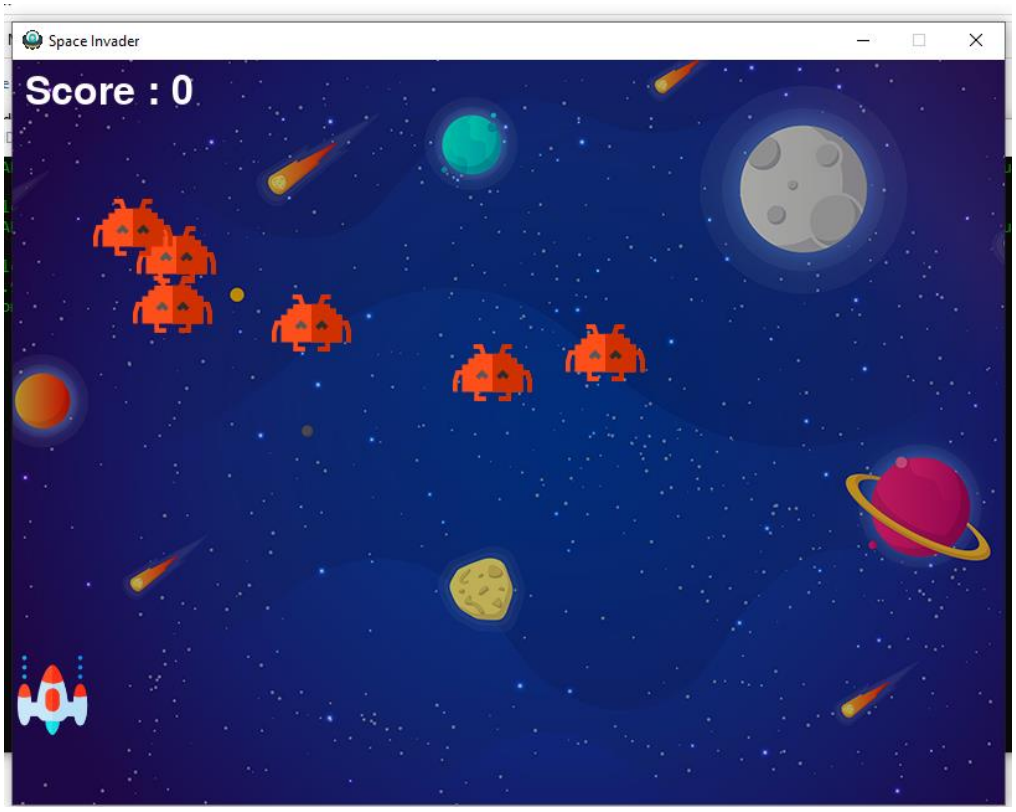
```
169.
170.            enemy(enemyX[i], enemyY[i], i)
171.
172.        # Bullet Movement
173.        if bulletY <= 0:
174.            bulletY = 480
175.            bullet_state = "ready"
176.
177.        if bullet_state is "fire":
178.            fire_bullet(bulletX, bulletY)
179.            bulletY -= bulletY_change
180.
181.        player(playerX, playerY)
182.        show_score(textX, testY)
183.        pygame.display.update()
```

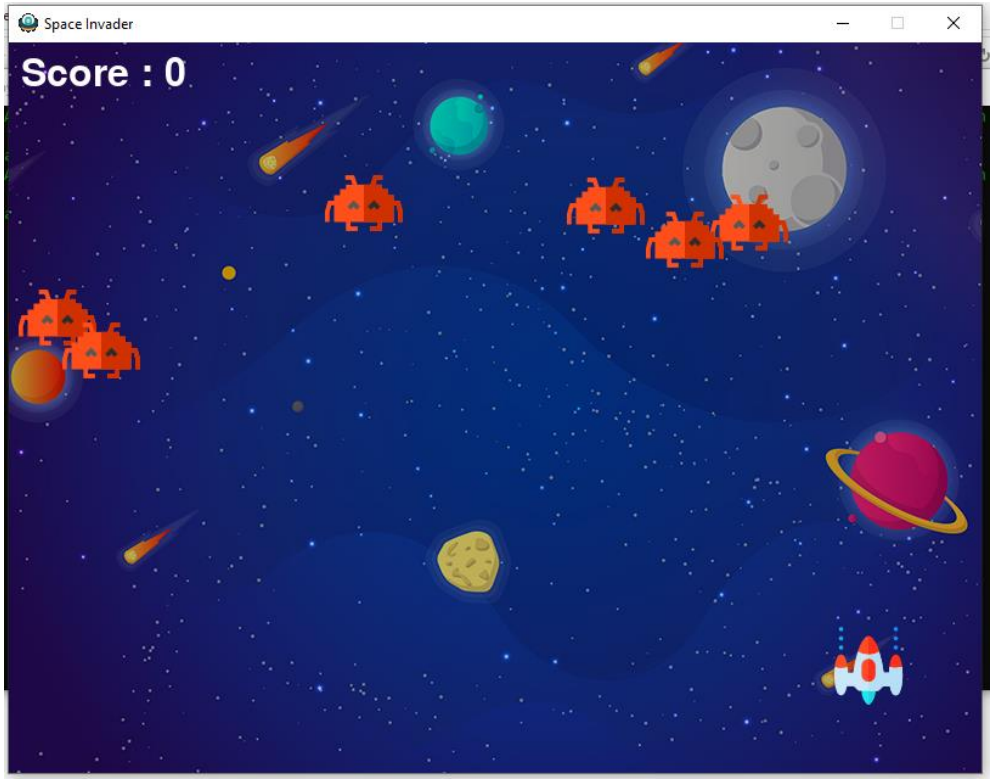
OUTPUT

When game starts and player is in center -

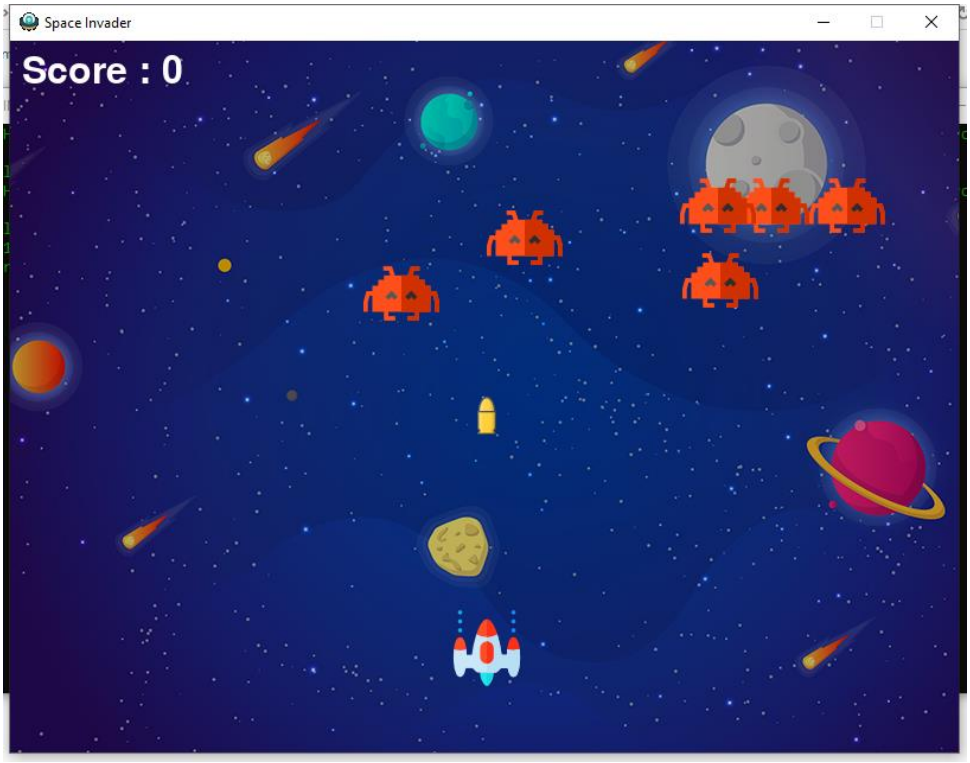


When player moves in left and right direction –

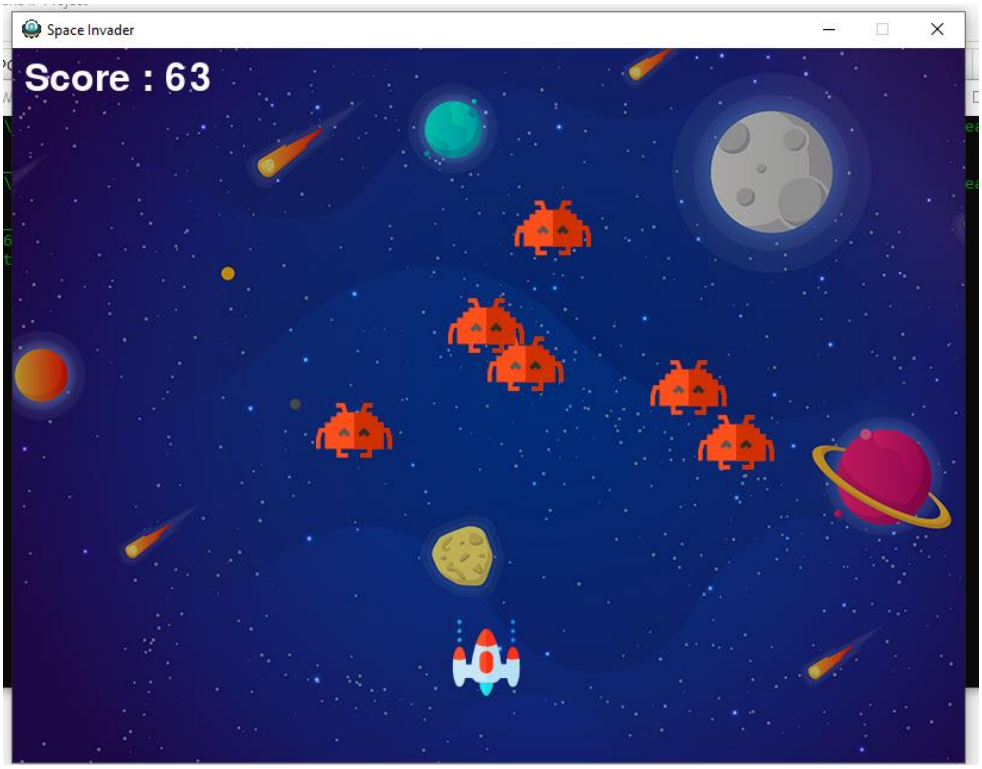




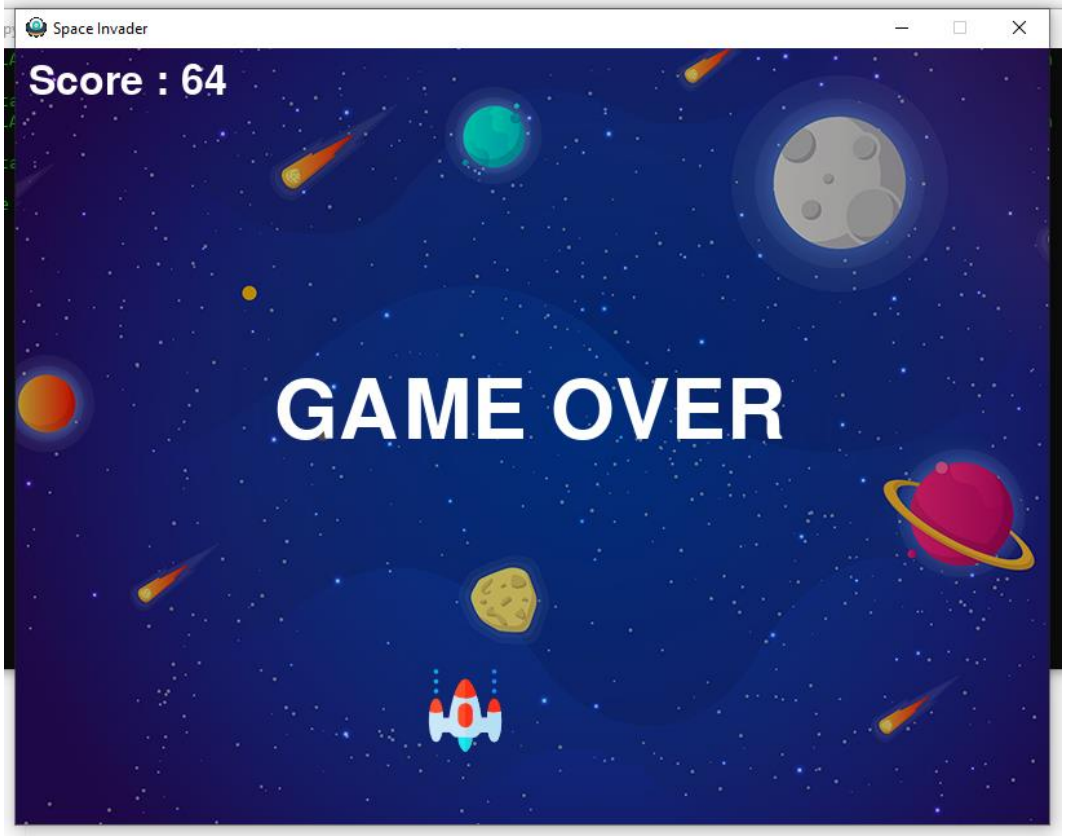
When player fires bullet then it is like –



When score increases, then scorecard is like –



When game is over, then game window is like –



FUTURE SCOPE

Space Invaders is a well – known fascinating game. After its first launch, in Japan. It got a huge success from 1980s, by releasing its new features, updates, improvements, in graphics, sound, etc.

Earlier, people considered it a very easy game. But, later on, after its many updated versions, it started consisting of difficulties, hard levels and much more.

Today, it still had a great involvement of players and especially teenagers.

It has more scope in future as it is upgrading time by time.

REFERENCES

Books used –

1. Informatics Practices – Class XI – Sumita Arora
2. Informatics Practices – Class XII – Sumita Arora

Websites –

1. www.google.com
2. www.wikipedia.org
3. https://realpython.com/pygame-a-primer/

THANK YOU ☺

Regards,

Siddhant Nagaria - XII-Science

Priyanshu Agarwal - XII - Commerce