**Major Project Report**

**ON**

**SPACE SHOOTER (GAME)**

**SUBMITTED FOR PARTIAL FULFILMENT OF DIPLOMA IN**

**COMPUTER ENGINEERING**

**(Batch 2021-2024)**

**BY :-**

**RITIK SINGH**

**(10621415)**

**Under the guidance of**

**Dr. Kusan Biswas**

** *Assistant professor***

Delhi Skill and Entrepreneurship University

**DSEU Rajokri Campus Delhi-110038**

## CERTIFICATE FROM GUIDE

It is to certify that the project entitled “**SPACE SHOOTER GAME**”, submitted by

Mr. **RITIK SINGH** , roll no **10621415** to the **DSEU Rajokri Campus***, Delhi,* has been completed under my supervision and the work is carried out and presented in a manner required for its acceptance to *Diploma in CE*.

**Project Guide:**

Signature: **……………………….**

Name: **………………………...**

Date: **……………………….**

**ACKNOWLEDGEMENT**

First of all, we thank our God for giving me the spirit to complete the project. And Secondly Our Family. Without their support this Game will not going to be developed

I also thank my friends for supporting me

Special thanks to our supervisor, Dr. kusan Biswas for his support and patience throughout this project

………………………………………………………………………………………………………………

Space Shooter game

*Developed on Python using Turtle graphics*

**DECLARATION**

I **RITIK SINGH** (10621415), student of 6th semester Diploma in Computer engineering, **DSEU** **RAJOKARI CAMPUS**, Delhi, hereby declare that the project work entitled “**SPACE SHOOTER GAME”** submitted to Rajokri. during the academic year 2021-24 is a record of work done by me under the guidance of **Dr. kusan Biswas,** Lecturer, **Rajokri**. This project work is submitted in partial fulfilment of the requirements for the award of the Diploma in Computer Engineering.

Date: **RITIK SINGH**

Place: New Delhi

**ABSTRACT**

Author RITIK SINGH

Title SPACE SHOOTER GAME

Year 2021

Language English

Name of supervisor Dr. kusan Biswas

……………………………………………………………………

In Recent times, Game have become an important part of student’s life, So by considering the likes of students, We decided to Make a game. After that we do a survey with our friends, family members, neighbours and the most common preference we got was a racing game. So we decided to develop a racing game.

As we are doing our internship on python, we have two choices either we can make the game using tkinter module or turtle module. We choose turtle module for the development of the game by also including some of the tkinter function, The reason behind choosing the turtle module is that because it is easy to use and support various functions which is sufficient to develop a racing game.

During the development of the game our supervisor advised us to add more features in the game to make it more attractive, So After a deep discussion between us (RITIK SINGH) we came to the conclusion to add a betting feature in the game. And after so many hardworking sessions we developed the game successfully!

CONTENTS

*Acknowledgement*

*Abstract*

1. Introduction .............................................................................09
   1. Score ..........................................................................09
   2. Lives ..........................................................................09
   3. Invaders .....................................................................09
   4. Missile .......................................................................10
   5. Controls Of The Game ..............................................10
   6. Technologies& Modules ...........................................11
   7. Python Programming Language ................................11-12
   8. Python Turtle Graphics ..............................................12-13
   9. Python Os And Random Module ...............................14
2. Feasibility Study ......................................................................15

2.1 Technical Study .........................................................15

2.2 Economic Study ........................................................16

2.3 Operational Study ......................................................16

2.4 Schedule Study ..........................................................16-17

1. Gui Design ................................................................................17
   1. Background .................................................................17-19
   2. Border ........................................................................ 20
   3. Enemy And Alley .......................................................21
   4. Create Enemy ............................................................21
   5. Create An Ally ...........................................................22
2. Implementation Of Game: ...........................................................23
   1. Player Class ....................................................................23
   2. Create Missile ................................................................24
3. Score And Lives Indicator ...........................................................25
   1. Player And Invaders .......................................................26-27
   2. Player Name ...................................................................28
4. Missile Collision With Alley Or Enemy ......................................29

7.1 Missile Collision With Enemy ...............................................29

7.2 Missile Collision With Ally ...................................................29

7.3 Keys Binding & Function ......................................................30-31

7.4 Border ....................................................................................31

7.5 No Life Message Box ............................................................31-32

1. Main Gameloop .......................................................................32
2. Help Menu ..............................................................................33-34
3. Reference .....................................................................................35
4. Conclusion ...............................................................................36

**LIST OF FIGURES, TABLES AND CODE SNIPPETS**

Figure 1.Level 1 ..........................................................................18

Figure 2. Level 2 .........................................................................19

Figure 3. Level 3..........................................................................19

Figure 4. Border Or Game Display .............................................20

Figure 5. Devil Icon …………………........................................21

Figure 6. Angel Icon ....................................................................22

Figure 7. Score And Lives Indicator.............................................26

Figure 8. Player Name Input Box ……………............................28

Figure 9. Player Name Displaying…............................................28

Figure 10. No Lives Message Box ...............................................32

Figure 11. Help Menu...................................................................33

Table 1. Various Testing Operations On Game............................34

Code Snippet 1. Turtle Motion.....................................................13-14

Code Snippet 2. Implementing Pictures…....................................17-18

Code Snippet 3. Draw Border………………………...................20

Code Snippet 4.Create Enemy....................................................21

Code Snippet 5.Create Ally........................................................22

Code Snippet 6. Create Player.......................................................26-24

Code Snippet 7. Create Missile……………………….................24-25

Code Snippet 8. Score And Lives Indicator. ................................25

Code Snippet 9. Player And Invaders............................................26-27

Code Snippet 10. Missile Collision With Enemy.........................29

Code Snippet 11. Missile Collision With Ally.............................29-30

Code Snippet 12. Key Binding.....................................................30-31

Code Snippet 13. Border..............................................................31

Code Snippet 14. Main Game Loop.............................................32

1. INTRODUCTION**:**

SPACE SHOOTER GAME! Is a Space combat game in which our player (Turtle) will hit the beam to kill the invaders and earning the points This game is mainly developed for the Kids.

In this game

* 1. **SCORE:**

Whenever an invader is hit by the beam releases by the player, the player will be awarded with 10 points. If the player is collapse the invader, the invader randomly

Moves to the other places. If the player hit the enemies realised by the beam then

Score will be reducing 5 points.

* 1. **LIVES:**

We have assign three lives two the players, when the player beam is hits to the red invader the one life of the player will be reduced.

When the player lives becomes equal to zero, A message will be appear saying you have no life to play the game and the game will end Here.

* 1. **INVADERS:**

There are two Invaders playing in our game, these two invaders has been given name as Enemy and Alley, Enemy is the Red Invader and Ally is the green invader.

When our missile hit the Enemy our one life will be decrease

When our missile hit the Ally 10 points will be added in the Final Score.

* 1. **MISSILE:**

When we press the Space key of the keyboard, our player will release the missile, when this missile hit an Alley it will destroy them but when the missile hit the Enemy it won’t affect them but will decrease the life of the player.

* 1. **CONTROLS OF THE GAME:**

To Control the turtle player of the game the user has to use the Up, Down, Left and Right key of the PC, The function of these keys in the game is as follows:

**Up Key-** It will increase the speed of the player

**Down Key-** It will decrease the speed of the player

**Left Key-** It will move the player on some degree on left side.

**Right Key-** It will move the player on some degree on right side.

**Space Key-** Release the missile

* 1. **TECHNOLOGIES& MODULES:**

This section showing the technologies, Language and it’s modules which are used to develop the “SPACEWAR” game and the technologies are python language, python turtle graphics, python OS module, Python random module etc.

* 1. **PYTHON PROGRAMMING LANGUAGE:**

Python is an integrated, high-level, general-purpose programming language. Created **by Guido van Rossum** and first released in **1991**, python has a design philosophy that emphasizes code readability, notably using significant whitespace.

It provides constructs that enable clear programming on both small and large scales. Van Rossum led the language community until stepping down as leader in **July 2018**.

Features and philosophy

Python is a multi-paradigm programming language. Object-oriented programming and structured programming are fully supported, and many of its features support functional programming and aspect-oriented programming (including by metaprogramming and metaobjects (magic methods)). Many other paradigms are supported via extensions, including design by contract and logic programming.

Python uses dynamic typing, and a combination of reference counting and a cycle-detecting garbage collector for memory management. It also features dynamic name resolution(late binding), which binds method and variable names during program execution.

Python's design offers some support for functional programming in the Lisp tradition. It has filter, map, and reduce functions; list comprehensions, dictionaries, sets and generator expressions.The standard library has two modules (intercools and functools) that implement functional tools borrowed from Haskell and Standard ML.

The language's core philosophy is summarized in the document The Zen of Python (PEP 20), which includes aphorisms such as:

• Beautiful is better than ugly

• Explicit is better than implicit

• Simple is better than complex

• Complex is better than complicated

• Readability counts

Users and admirers of Python, especially those considered knowledgeable or experienced, are often referred to as Pythonists, Pythonistas, and Pythoneers.

* 1. **PYTHON TURTLE GRAPHICS:**

Turtle graphics is a popular way for introducing programming to kids. It was part of the original logo programming language.

The **turtle** module is an extended reimplementation of the same-named module from the python standard distribution up to version python 2.5

The object – oriented interface uses essentially two classes:

1. The Turtle Screen class defines graphics windows as a playground for the drawing turtles. Its constructor needs a tkinter. Canvas or a Scrolled Canvas as argument. It should be used when turtle is used as part of some application.

Raw Turtle defines Turtle objects which draw on a turtle screen. Its constructor needs canvas, scrolled canvas or Turtle Screen as argument, so the Raw Turtle objects know where to draw.The turtle has three attributes: a location, an

1. A full turtle graphics system requires control flow, procedures, and recursion: many turtle drawing programs fall short. From these building blocks one can build more complex shapes like squares, triangles, circles and other composite figures. The idea of turtle graphics, for example is useful in a Lindenmayer system for generating fractals orientation (or direction), and a pen. The pen, too, has attributes: color, width, and on/off state.
2. The turtle moves with commands that are relative to its own position, such as "move forward 10 spaces" and "turn left 90 degrees". The pen carried by the turtle can also be controlled, by enabling it, setting its color, or setting its width. A student could understand (and predict and reason about) the turtle's motion by imagining what they would do if they were the turtle. Seymour Papert called this "body syntonic" reasoning.
3. .
4. Turtle geometry is also sometimes used in graphics environments as an alternative to a strictly coordinate-addressed graphics system.
5. Turtle motion

Move and draw

**Code snippet 1**

#import the Turtle module

import turtle

turtle.fd(0)

#set the animations speed to the maximum

turtle.speed(0)

#change the background color

turtle.bgcolor("black")

#change the window title

turtle.title("Space War")

#hide the default turtle

turtle.ht()

#this save memory

turtle.setundobuffer(1)

#this speeds up drawing

turtle.tracer(0)

class Sprite(turtle.Turtle):

def \_\_init\_\_(self,spriteshape, color, startx, starty):

turtle.Turtle.\_\_init\_\_(self, shape = spriteshape)

self.speed(0)

self.penup()

self.color(color)

self.fd(0)

self.goto(startx, starty)

self.speed =1

* 1. **PYTHON OS AND RANDOM MODULE:**

The OS module in python provides functions for interacting with the operating system. OS comes under Python’s standard utility modules. This module provides a portable way of using operating system dependent functionality.

The RANDOM module implements pseudo-random numbers generators for various distributions.

For Integers, Uniform selection from a range.

For sequences, uniform selection of a random Number, A function to generate a random permutation of a list in-place, and a function for random sampling without replacement.

**2 FEASIBILITY STUDY:**

A feasibility study is a study, usually done by engineers, that establishes whether conditions are right to implement a particular project.

A feasibility study is done as part of a systems development lifecycle, in order to drive precision for the implementation of technologies. It includes the following components:

* Technical
* Economic
* Operational
* Schedule

**2.1 TECHNICAL STUDY:**

Technical feasibility assesses the current resources (such as hardware and software) and technology, which are required to accomplish the user requirements in the software within the allocated time and budget.

According to our technical feasibility study, a user requires the following items install on his/her computer to run our Turtle betting race game successfully on his system without any problem-

* Python 3 should be installed on his system because if any lower version of python is installed then user must face some issues in the program functionality as we have uses some new function in our code which is introduced in Python 3 and not available in lower versions of python.
* For proper viewing of the application output user screen must be set into 1920x1020 resolution.
* The User must have basic knowledge of English as our all dialogs appear in English language.
* The User must have basic knowledge of Computer for properly playing the Game.

**2.2 ECONOMIC STUDY:**

Economic feasibility determines whether the required software is capable of generating financial gains for an organization. It involves the cost incurred on the software development team, estimated cost of hardware and software, cost of performing feasibility study, and so on.

As per the Economic study of our game, This game is fee and there is no money spent to develop this game as this game is of organic type (in which the system size consist of 2-50 KLOC) hence the working environment is friendly and developed on the single system at home.

Takes no expenses except time of coders.

This aspect is important for big projects not for small projects like that are of organic type

**2.3 OPERATIONAL STUDY:**

Operational feasibility assesses the extent to which the required software performs a series of steps to solve business problems and user requirements. This feasibility is dependent on human resources (software development team) and involves visualizing whether the software will operate after it is developed and be operative once it is installed.

According to operational feasibility of the game, This will fulfil the wish of user who wants to do betting in racing. The operational functions of this game is very simple and user will not face any kind of problem using it.

Any problem generate during the program execution will halt the game or cause malfunctioning of the game, at this Stage the user requires to press the “ESC” key which will cause taking exit from the game.

**2.4 SCHEDULE STUDY:**

Schedule feasibility is the degree to which a deadline for a strategy, plan, project or process is realistic and achievable.

During schedule study the allotted time for the project is decided to be of 3 months (Same time as of our internship period).

At the end, we successfully created our project in the decided time period. In Schedule study we divided the whole game into different task as follows:

* Creating day background
* Creating night background
* GUI of the game
* Internal functioning of the game (Making logic of running and putting bet on the players)
* Creating Money game
* Creating Internal functioning of the money game
* Money game GUI work
* Exception handling of the game

Etc

And also assign time to accomplish this task.

1. **GUI DESIGN:**

The GUI design of this Space war game is done by using the **turtle** module of python programming language.

Basic functions of the turtle module is used to create the basic GUI, An Image is used to give a nice night background to the game.

* 1. **BACKGROUND:**

An image is used for the background so the background will look very similar to a Space environment .We have inserted 3 background in this game the background will change by entering in new level . The screenshot of the background used in the game is shown below:

**Code snippet 2**

#Implementing pictures

screen=turtle.Screen()

img3="C:\Users\ritik\Desktop\my game/bg1.gif"

img4="C:/Users/ritik/Desktop/my game/bg2.gif"

img5="C:/Users/ritik/Desktop/my game/bg3.gif"

bg=turtle.Turtle()

screen.addshape(img3)

screen.addshape(img4)

screen.addshape(img5)

bg.shape(img3)

bg.goto(0,0)



Figure 1 :LEVEL 1



Figure 2 :LEVEL 2



Figure 3 :LEVEL 3

**3.2 BORDER:**

To create the border we simply moves a turtle across while the screen this turtle when moves from any place creates a line there as a result a square was formed.

This square is our border inside which the player and invader will move.

**Code snippet 3**

def draw\_border(self):

#draw border

self.pen.speed(0)

self.pen.color("white")

self.pen.pensize(3)

self.pen.penup()

self.pen.goto(-300,300)

self.pen.pendown()

for side in range(4):

self.pen.fd(600)

self.pen.rt(90)

self.pen.penup()

self.pen.ht()

self.pen.pendown()



Figure 4: BORDER OR GAME DISPLAY

**3.3 ENEMY AND ALLEY:**

The enemy and alley is actually the object of turtle, but in game they were shown as some shape this shapes are actually the gif images.

This enemy is part of the gaming. It is created using the class Enemy

Whenever the player is realised to the beam to the enemy then player

Lives will be reducing.

**3.4 CREATE ENEMY:**

Class Enemy is used to create the enemies which is generated

Randomly. To create the enemies it is used to list in which we

Can created many more enemies

**Code snippet 4**

class Enemy(Sprite):

def \_\_init\_\_(self,spriteshape, color, startx, starty):

Sprite.\_\_init\_\_(self,spriteshape, color, startx, starty)

self.speed = 4

self.setheading(random.randint(0,360))

enemies = []

for i in range(8):

enemies.append(Enemy(img2, "red",-100, 0))

**C:\Users\sahil\Desktop\my game\devils.gif**

Figure 5 : DEVIL

**3.5 CREATE AN ALLY:**

Inside the game it used the class Ally which is used to create

The allies. Inside the game when the player released the beam

By the player and beam hit to the ally then that time score will

Be Increment by 10 points.

**Code snippet 5**

class Ally(Sprite):

def \_\_init\_\_(self,spriteshape, color, startx, starty):

Sprite.\_\_init\_\_(self,spriteshape, color, startx, starty)

self.speed = 8

self.setheading(random.randint(0,360))

allies = []

for i in range(6):

allies.append(Ally(img1, "blue", 0, 0))

****

Figure 6 :ANGLE

1. **IMPLEMENTATION OF GAME:**
   1. **Player Class**

We have create a Sprite Class that is have a child of a Turtle

Class which is making a game objects. In this Player Class used

A Sprite Class which will inherits all the Turtle abilities and

Also create a Player Class which is inherits Sprite Class. Attributes inside Sprite Class same having the Player Class.

Inside the Sprite Class which is also define the move () function

By which move the player.

**Code snippet 6**

class Sprite(turtle.Turtle):

def \_\_init\_\_(self,spriteshape, color, startx, starty):

turtle.Turtle.\_\_init\_\_(self, shape = spriteshape)

self.speed(0)

self.penup()

self.color(color)

self.fd(0)

self.goto(startx, starty)

self.speed = 1

def move(self):

self.fd(self.speed)

class Player(Sprite):

def \_\_init\_\_(self,spriteshape, color, startx, starty):

Sprite.\_\_init\_\_(self,spriteshape, color, startx, starty)

self.shapesize(stretch\_wid = 1, stretch\_len = 1.5, outline = None)

self.speed = 1

self.lives = 3

def turn\_left(self):

self.lt(45)

def turn\_right(self):

self.rt(45)

* 1. **Create Missile**

To creating a missile have to create a Missile Class and

It is child of Sprite Class. And it is inherits all the attributes

From the Sprite Class. Inside Missile Class define the fire

Method (). If the fire become ready than status will be “firing”

**Code snippet 7**

class Missile(Sprite):

def \_\_init\_\_(self,spriteshape, color, startx, starty):

Sprite.\_\_init\_\_(self,spriteshape, color, startx, starty)

self.shapesize(stretch\_wid = 0.3, stretch\_len = 0.4, outline=None)

self.speed = 20

self.status = "ready"

self.goto(-1000, 1000)

def fire(self):

if self.status == "ready":

self.goto(player.xcor(), player.ycor())

self.setheading(player.heading())

self.status = "firing"

def move(self):

if self.status == "ready":

self.goto(-1000, 1000)

if self.status == "firing":

self.fd(self.speed)

#boder check

if self.xcor() < -290 or self.xcor() > 290 or \

self.ycor() < -290 or self.ycor() > 290:

self.goto(-1000, 1000)

self.status = "ready"

1. **SCORE AND LIVES INDICATOR:**

The Score and Lives are shown on top of the screen window, whenever player missile hit alley the score will increase and in case of Enemy live will be decrease.

The Coding used to implement them is shown below; the show status function is used to show the Status of the Score and Lives

**Code snippet 8**

**def**show\_status**(**self**):**

self**.**pen**.**undo **()**

self**.**l**.**undo **()**

self**.**l**.**color **(**"white"**)**

self**.**l**.**hideturtle **()**

msg **=**"SCORE: %s"**%(**self**.**score**)**

msg2**=**"LIVES: %s"**%(**self**.**lives**)**

self**.**pen**.**penup**()**

self**.**pen**.**goto**(-**300**,**310**)**

self**.**pen**.**write**(**msg**,** font **=(**"Arial"**,**16**,**"normal"**))**

self**.**l**.**penup**()**

self**.**l**.**goto**(**200**,**310**)**

self**.**l**.**write**(**msg2**,** font **=(**"Arial"**,**16**,**"normal"**))**

**if** game**.**lives**==**0**:**

messagebox**.**showinfo**(**"BETTER LUCK NEXT TIME"," GAME OVER "**)**

turtle**.**bye**()**

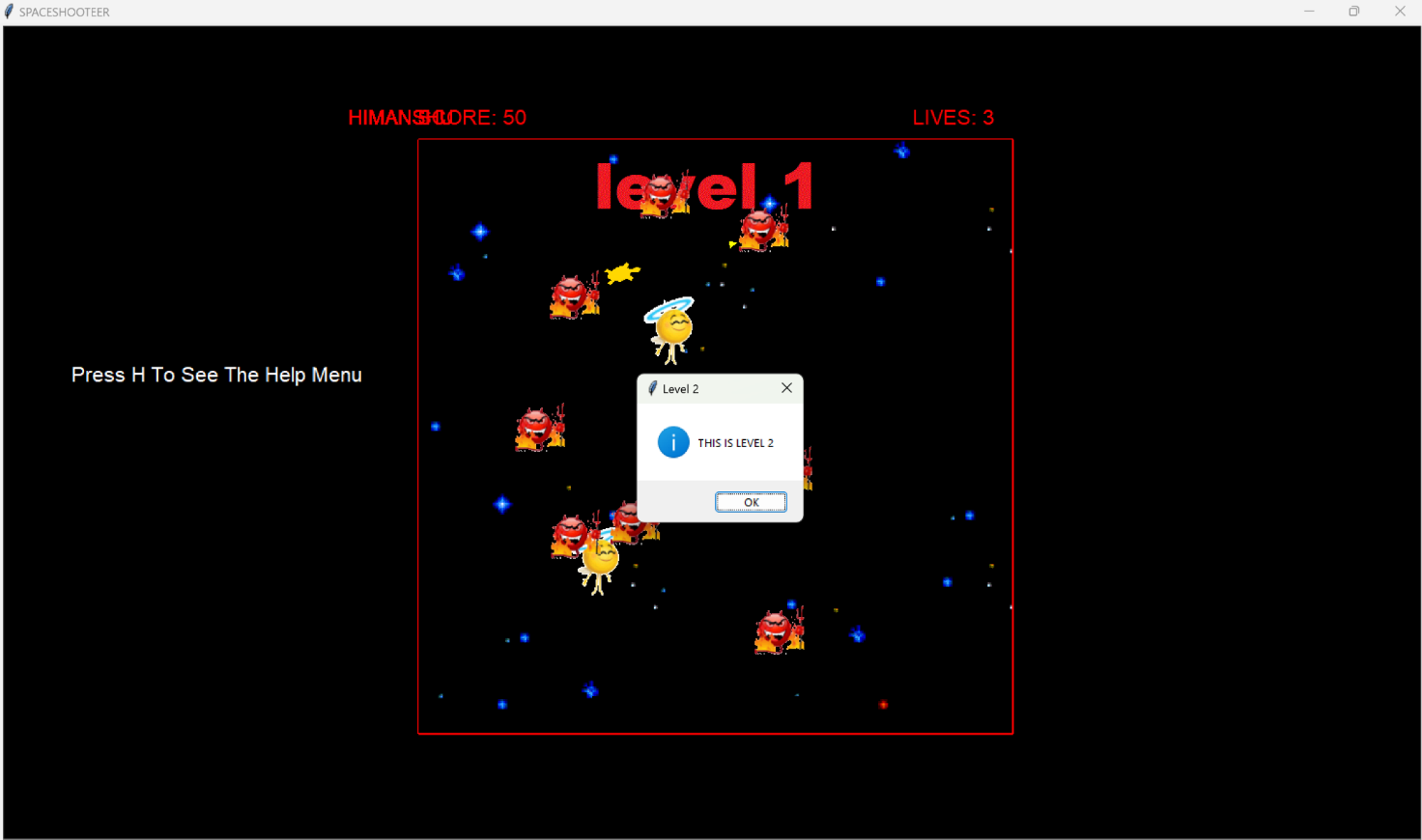


Figure 7 : SCORE AND LIVES INDICATOR

**5.1 PLAYER AND INVADERS:**

The Player is assign with shape turtle and invaders are given the gif image, these gif images are first loaded into the turtle module shapes and then assign these pictures to the player and invader.

**Code snippet 9**

#Implementing pictures

screen**=**turtle**.**Screen**()**

#img**ae**

img1**=**"C:/Users/ritik/Desktop/my game/my angel.gif1` `"

img2**=**"C:/Users/ritik/Desktop/my game/devils.gif"

screen**.**addshape**(**img**)**#img=Figher

screen**.**addshape**(**img1**)**#img1=Red Invader

screen**.**addshape**(**img2**)**#img2=Green Invader

#create my sprites

player **=** Player**(**"turtle"**,**"white"**,**0**,**0**)**

#enemy = Enemy("circle", "red",-100, 0)

missile **=** Missile**(**"triangle"**,**"yellow"**,**0**,**0**)**

#ally = Ally("square", "blue", 0, 0)

enemies **=[]**

**for** i **in** range**(**8**):**

enemies**.**append**(**Enemy**(**img2**,**"red"**,-**100**,**0**))**

allies **=[]**

**for** i **in** range**(**3**):**

allies**.**append**(**Ally**(**img1**,**"blue"**,**0**,**0**))**

* 1. **PLAYER NAME**

we have provided the option of entering the player name in our game in which player have to insert their name in game before playing.

After inserting the name, the game will start and player name will show at top of the screen at left side of score.

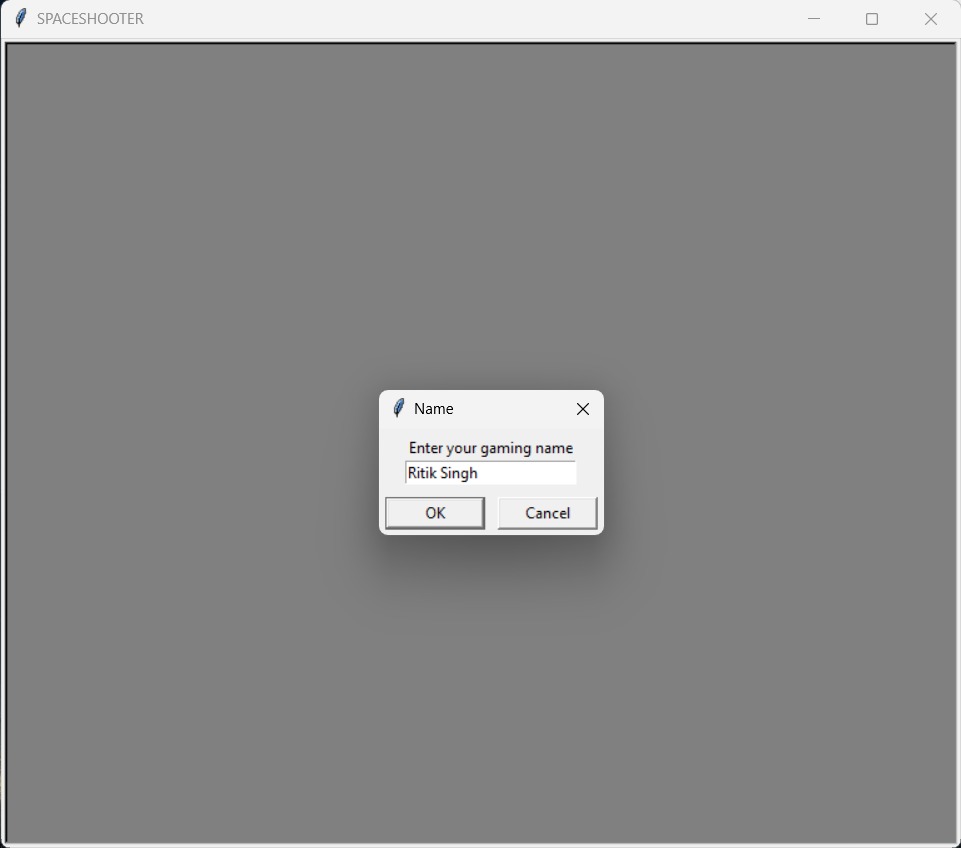


Figure 8 PLAYER NAME INPUT BOX

****

Figure 9; PLAYER NAME DISPLAYING

1. **MISSILE COLLISION WITH ALLEY OR ENEMY:**

When missile collide with any one Enemy or Alley the score or live will increase or decrease, the coding for detecting the collision between missile and Alley or Enemy is given below:

**6.1 MISSILE COLLISION WITH ENEMY:**

Inside the game when the missile collision with the enemy

Then that time score will be increase by plus 10 points.

**Code snippet 10**

#check for a collision between the missile and the enemy

**if** missile**.**is\_collision**(**enemy**):**

x **=** random**.**randint**(-**250**,**250**)**

y **=** random**.**randint**(-**250**,**250**)**

enemy**.**goto**(**x**,**y**)**

missile**.**status **=**"ready"

#increase the score

game**.**score **+=**10

game**.**show\_status**()**

**6.2 MISSILE COLLISION WITH ALLY:**

When the player released the missile and enemy hit by the

Missile that time the player will by decrees by 1 lives .when

The lives become zero than a status show you have no lives.

**Code snippet 11**

#check for a collision between the missile and the ally

**if** missile**.**is\_collision**(**ally**):**

x **=** random**.**randint**(-**250**,**250**)**

y **=** random**.**randint**(-**250**,**250**)**

enemy**.**goto**(**x**,**y**)**

missile**.**status **=**"ready"

#increase the score

game**.**lives **-=**1

game**.**show\_status**()**

* 1. **KEYS BINDING& FUNCTION:**

As we discuss in the section 1, the keys for controlling the players are: Up, Down, Left, Right and Space Key.

The binding of these key with the program are done by using the following code:

The Functions turn left, turn right, accelerate, deaccelerate and fire will do their given task.

**Code snippet 12**

#keyboard binding

turtle**.**onkey**(**player**.**turn\_left**,**"Left"**)**

turtle**.**onkey**(**player**.**turn\_right**,**"Right"**)**

turtle**.**onkey**(**player**.**accelerate**,**"Up"**)**

turtle**.**onkey**(**player**.**decelerate**,**"Down"**)**

turtle**.**onkey**(**missile**.**fire**,**"space"**)**

turtle**.**listen**()**

**def**turn\_left**(**self**):**

self**.**lt**(**45**)**

**def**turn\_right**(**self**):**

self**.**rt**(**45**)**

**def**accelerate**(**self**):**

self**.**speed **+=**1

**def**decelerate**(**self**):**

self**.**speed **-=**1

**def**fire**(**self**):**

**if** self**.**status **==**"ready"**:**

self**.**goto**(**player**.**xcor**(),** player**.**ycor**())**

self**.**setheading**(**player**.**heading**())**

self**.**status **=**"firing"

* 1. **BORDER:**

The borders inside which the whole player, Enemy and Alley are moving are draw by using the following code.

The logic is simple; we just move a turtle across whole screen by setting his colour to white.

**Code snippet 13**

**def**draw\_border**(**self**):**

#draw border

self**.**pen**.**speed**(**0**)**

self**.**pen**.**color**(**"white"**)**

self**.**pen**.**pensize**(**3**)**

self**.**pen**.**penup**()**

self**.**pen**.**goto**(-**300**,**300**)**

self**.**pen**.**pendown**()**

**for** side **in** range**(**4**):**

self**.**pen**.**fd**(**600**)**

self**.**pen**.**rt**(**90**)**

self**.**pen**.**penup**()**

self**.**pen**.**ht**()**

self**.**pen**.**pendown**()**

* 1. **NO LIFE MESSAGE BOX:**

The player has assign 3 lives by us and whenever one missile hits the Enemy the one life of the player will be reduced and when the lives was equal to zero, A message will appear saying “You have No life to play the Game”.

For displaying this message box the message box function of **tkinter module** was used, the coding was shown below:

**Code snippet 14**

**if** game**.**lives**==**0**:**

messagebox**.**showinfo**(**"BETTER LUCK NEXT TIME"," GAME OVER :) .YOU HAVE NO LIFE TO PLAY THE GAME"**)**

turtle**.**bye**()**

Figure 10; NO LIVES MESSAGE BOX

1. **MAIN “GAMELOOP”**

This is the important part of the main game loop structure.

In this section have to call some function which is used during

The playing game. This some function call many times till the

Game does not end. Some function used inside the main game

Loop like – move () function, update () function and sleep ()

Functions etc.

#main game loop

while True:

turtle.update()

time.sleep(0.03)

player.move()

missile.move()

1. **HELP MENU**

we have provided help menu to the user to guide the user how to play the game and also provide the information about the game rules

to open the help menu user have to press the H button after pressing the H button the help menu will be open on the screen while playing game .the example to help menu is in the fig.below.

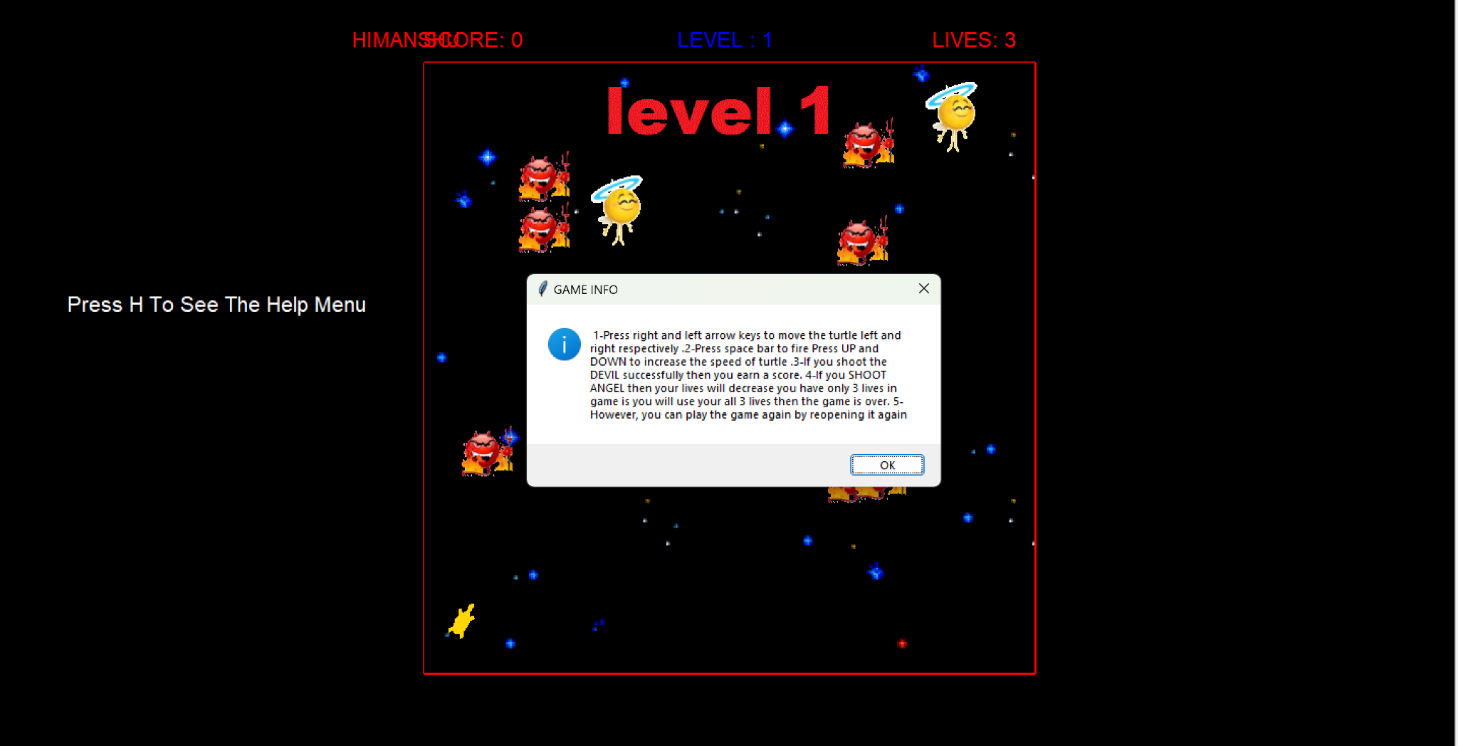


Figure 11; HELP MENU

**description of help menu:**

1-Press right and left arrow keys to move the turtle left and right respectively

2-Press space bar to fire ,

3-Press UP and DOWN to increase the speed of turtle.

if you shoot the DEVIL successfully then you earn a score.

If you SHOOT ANGEL then your lives will decrease .

you have only 3 lives in game is you will use your all 3 lives then the game is over.

However, you can play the game again by reopening it again.

1. **REFERENCE:**

* PYTHON:

<https://www.w3schools.com/python/>

* TURTLE MODULE:

<https://docs.python.org/2/library/turtle.html>

* TKINTER MODULE

<https://docs.python.org/3/library/tk.html>

* RANDOM MODULE

<https://docs.python.org/3/library/random.html>

* MATH MODULE

<https://docs.python.org/3/library/math.html>

# **Python Tutorials** <https://www.youtube.com/watch?v=uyjd3pBPiU8>

1. **CONCLUSION:**

The Main Purpose of developing this application to create a “SPACE SHOOTER GAME” Game which will be playing by the kids. Many kids enjoy the playing games. Now a day’s games become popular in the Worlds. No one has not to play the games. Today’s in the world games becomes a big market place in worlds. When some games prove the money when the people play the game and because of this offers people taking more interest play the games.

We encounter several problems while developing it like the logic development of collision between missile and enemy or alley, moving the players, binding keys and making them work in an appropriate manner etc.

At last the game was finally developed and working successfully.