`

**SPACE INVADERS GAME**

SYNOPSIS

SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENT FOR THE

AWARD OF THE DEGREE OF

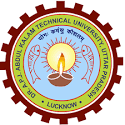
BACHELOR OF TECHNOLOGY

(CSE)

SUBMITTED BY

PRATHAM KUMAR GUPTA

(2003490100022)



15 SEPTEMBER 2022

Dr. APJ Abdul Kalam Technical University, Uttar Pradesh

LUCKNOW, INDIA

# OBJECTIVE:-

The aim is to destroy the rows of aliens as they move horizontally across the screen ,moving faster and faster as you pass

levels. Once you destroy one wave of aliens, get ready to face another faster and more difficult wave of aliens ,If you allow

any space invaders to reach the bottom of the screen or enemies bullets consume your lives , the alien invasion has been successful

and the games ends.

# INTRODUCTION:-

Space Invaders is a stationary shooter in which the player fires at aliens overhead by moving a laser weapon horizontally across the bottom of the screen. As a group, the aliens travel left and right, shifting downward as they approach the screen’s edge. The objective is to shoot all of the aliens to death. The game finishes quickly if the invaders reach the bottom of the screen while the player has three lives.

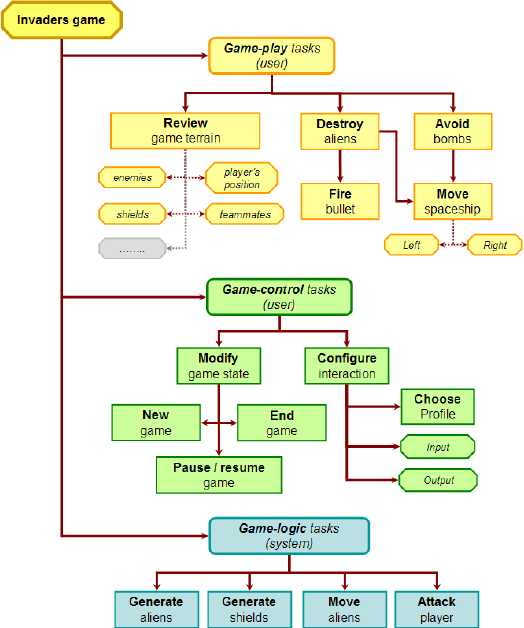
# SOFTWARE AND HARDWARE REQUIREMENTS:-

**SOFTWARE REQUIREMENT**

* Python
* Pygame
* OS-Windows
* Pycharm

**HARDWARE REQUIREMENT**

* with 8gb ram
* 512gb SSD
* Intel core i5 10th Generation
* Intel UHD Graphics



# ER DIAGRAM

# Technology used :-

# Python:

# Python is a computer programming language often used to build websites and

# software, automate tasks, and conduct data analysis. Python is a general-purpose

# language, meaning it can be used to create a variety of different programs and isn&#39;t

# specialized for any specific problems.

**Approach:-**

* Import the required module.
* Initialize the pygame.
* Create three functions:
  + **isCollision():** Which tells us whether the collision has occurred or not?
  + **game\_over():** Which returns True or False on the basis of which the code decided if the game has ended.
  + **show\_score(x, y):** This shows the score on the screen
* Create an infinite loop to execute the code continuously.

**isCollision():-**

It’s very simple actually. Before explaining this, we want you to take a look at the collision portion of the code inside the game loop first below.

The criteria for collision set inside the function is the simplest thing as the distance between the bullet and the invader (our enemy).

As you can see the formula used for calculating distance is something that every student study in their high school mathematics class. It’s the formula of the distance between two points having coordinates (x1, y1) and (x2, y2) which are being passed as parameters of the isCollision() function.

So, whenever the position of the bullet and the invader changes then the isCollision() function checks if a collision has occurred or not. That is the reason why it is being called inside the game loop.

game\_over():-

Which returns True or False on the basis of which the code decided if the game has ended. For understanding the game\_over()

function, let’s take a look at the below snippet of code which is present inside the game loop

Before getting into the explanation of code, it is recommended to know about the coordinate system followed in pygame. Take a

look at the image below:

So, the criteria for game over is also collision. When the y-coordinate of the invader is greater than the spaceship i.e., 450 (y- coordinate of the spaceship), and the distance between the invader and the spaceship is less than 80 then a collision occurs and the game\_over() function is called followed by the explosion sound.

show\_score(x, y):

The only thing show\_score() function is doing is showing the score on the screen in a proper font selected by the user.

Every time a collision between the bullet and the invaders is happening a variable “**score\_val**” is being incremented. This variable

is then being displayed on the screen by the show\_score() function as can be seen in the above code snippet.

Import pygame Import random Import math

# initializing pygame pygame.init()

# creating screen

screen\_width = 800

screen\_height = 600

screen = pygame.display.set\_mode((screen\_width, screen\_height)) # caption and icon

pygame.display.set\_caption("Welcome to Space\Invaders Game by:- styles") # Score

score\_val = 0

scoreX = 5

scoreY = 5

font = pygame.font.Font('freesansbold.ttf', 20)

# Game Over

game\_over\_font = pygame.font.Font('freesansbold.ttf', 64)

def show\_score(x, y):

score = font.render("Points: " + str(score\_val), True, (255,255,255)) screen.blit(score, (x , y ))

def game\_over():

game\_over\_text = game\_over\_font.render("GAME OVER", True, (255,255,255)) screen.blit(game\_over\_text, (190, 250))

# Background Sound

mixer.music.load('data/background.wav’) mixer.music.play(-1)

# player

playerImage = pygame.image.load('data/spaceship.png’) player\_X = 370

player\_Y = 523

player\_Xchange = 0

# InvaderinvaderImage = []invader\_X = [] invader\_Y = [] invader\_Xchange = [] invader\_Ychange = [] no\_of\_invaders = 8

for num in range(no\_of\_invaders): invaderImage.append(pygame.image.load('data/alien.png'))

invader\_X.append(random.randint(64, 737))

invader\_Y.append(random.randint(30, 180)) invader\_Xchange.append(1.2) invader\_Ychange.append(50)

# Bullet

# rest - bullet is not moving # fire - bullet is moving

bulletImage = pygame.image.load('data/bullet.png’) bullet\_X = 0

bullet\_Y = 500

bullet\_Xchange = 0

bullet\_Ychange = 3 bullet\_state = "rest“

# Collision Concept

def isCollision(x1, x2, y1, y2): distance = math.sqrt((math.pow(x1 - x2,2)) + (math.pow(y1 - y2,2)))

if distance <= 50: return True

else:

return False

def player(x, y): screen.blit(playerImage, (x - 16, y + 10)) def invader(x, y, i): screen.blit(invaderImage[i], (x, y))

def bullet(x, y):

global bullet\_state screen.blit(bulletImage, (x, y)) bullet\_state = "fire"

# game loop

running = True while running:

# RGB screen.fill((0, 0, 0))

for event in pygame.event.get(): if event.type == pygame.QUIT: running = False

# Controlling the player movement # from the arrow keys

if event.type == pygame.KEYDOWN: if event.key == pygame.K\_LEFT: player\_Xchange = -1.7

if event.key == pygame.K\_RIGHT:

player\_Xchange = 1.7

if event.key == pygame.K\_SPACE: # Fixing the change of direction of bullet

if bullet\_state is "rest": bullet\_X = player\_X bullet(bullet\_X, bullet\_Y)

bullet\_sound = mixer.Sound('data/bullet.wav’)

bullet\_sound.play()

if event.type == pygame.KEYUP:

player\_Xchange = 0

# adding the change in the player position player\_X += player\_Xchange for i in range(no\_of\_invaders):

invader\_X[i] += invader\_Xchange[i] # bullet movement

if bullet\_Y <= 0:

bullet\_Y = 600 bullet\_state = "rest“

if bullet\_state is "fire":

bullet(bullet\_X, bullet\_Y) bullet\_Y -= bullet\_Ychange

# movement of the invader

for i in range(no\_of\_invaders):

if invader\_Y[i] >= 450:

if abs(player\_X-invader\_X[i]) < 80:

for j in range(no\_of\_invaders): invader\_Y[j] = 2000

explosion\_sound = mixer.Sound('data/explosion.wav’)

explosion\_sound.play()

game\_over() break

if invader\_X[i] >= 735 or invader\_X[i] <= 0: invader\_Xchange[i] \*= -1

invader\_Y[i] += invader\_Ychange[i] # Collision

collision = isCollision(bullet\_X, invader\_X[i], bullet\_Y, invader\_Y[i]) if collision:

score\_val += 1

bullet\_Y = 600 bullet\_state = "rest"

invader\_X[i] = random.randint(64, 736) invader\_Y[i] = random.randint(30, 200) invader\_Xchange[i] \*= -1 invader(invader\_X[i], invader\_Y[i], i)

# restricting the spaceship so that # it doesn't go out of screen

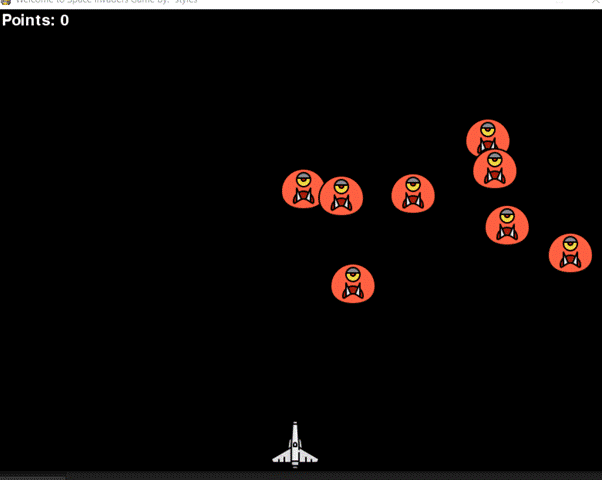
if player\_X <= 16: player\_X = 16;

elif player\_X >= 750:

player\_X = 750

player(player\_X, player\_Y) show\_score(scoreX, scoreY) pygame.display.update()

# RESULT INTERFACE:-



**CONCLUSION:-**

This Online Space invaders game provides facility to paly space invaders game anywhere and anytime. It save time since user does not need to wait for result So all user get extra knowledge and skills.

This project has helped us in getting a dearer understanding of real word application development.