Ex. No. : 1 Date: 09/11/2025

Register No.: 241701023 Name: KAVIYA S

Title : Space Shooter Game using Python Tkinter

Aim:

To design and develop a 2D Space Shooter game using Python Tkinter, where the player controls a spaceship to shoot falling enemies, gain score, and advance through levels.

Algorithm / Procedure (Short Version):

1. **Start the program.**
2. **Import necessary modules: tkinter, random, and time.**
3. **Create the Tkinter window and a canvas for the game area.**
4. **Draw a spaceship (player) at the bottom of the screen.**
5. **Allow movement of the spaceship using Left and Right arrow keys.**
6. **Shoot bullets with the Space bar and store active bullets in a list.**
7. **Generate enemies randomly at the top and move them downward continuously.**
8. **Check collisions between bullets and enemies:**
    - **On hit → destroy enemy and increase score.**
9. **Increase difficulty and level as score increases.**
10. **Detect collisions between enemies and player or bottom → game over.**
11. **Display final score and option to restart (press R).**

PROGRAME :

```
# Space Shooter Game using Python Tkinter
# Author: (Your Name)
# Controls: Arrow Keys to Move | Space to Shoot | R to Restart

import tkinter as tk
import random
import time

# ----------------------
# Game configuration
# ----------------------
WINDOW_W = 500
WINDOW_H = 700
PLAYER_W = 50
PLAYER_H = 30
PLAYER_SPEED = 20
BULLET_SPEED = 15
ENEMY_SPEED = 3
ENEMY_SPAWN_DELAY = 1200
LEVEL_UP_SCORE = 50
```

```
class SpaceShooter:
    def __init__(self, root):
        self.root = root
        self.root.title("🚀 Space Shooter Game")
        self.canvas = tk.Canvas(root, width=WINDOW_W, height=WINDOW_H,
bg="#0b0f1a")
        self.canvas.pack()

        # Initial values
        self.score = 0
        self.level = 1
        self.game_over = False

        # Create player
        self.player = self.canvas.create_rectangle(WINDOW_W//2-25,
WINDOW_H-80,
                                                   WINDOW_W//2+25, WINDOW_H-50,
                                                   fill="#00d4ff")

        # Lists
        self.bullets = []
        self.enemies = []
```

```
        # HUD
        self.score_text = self.canvas.create_text(10, 10, anchor="nw",
text="Score: 0",
                                                  fill="white", font=("Consolas", 14, "bold"))
        self.level_text = self.canvas.create_text(WINDOW_W-10, 10,
anchor="ne", text="Level: 1",
                                                  fill="white", font=("Consolas", 14, "bold"))

        # Bind controls
        root.bind("<Left>", self.move_left)
        root.bind("<Right>", self.move_right)
        root.bind("<space>", self.shoot)
        root.bind("r", self.restart)

        # Start game
        self.update()
        self.spawn_enemy()

    def move_left(self, e=None):
        if not self.game_over:
            self.canvas.move(self.player, -PLAYER_SPEED, 0)

    def move_right(self, e=None):
        if not self.game_over:
```

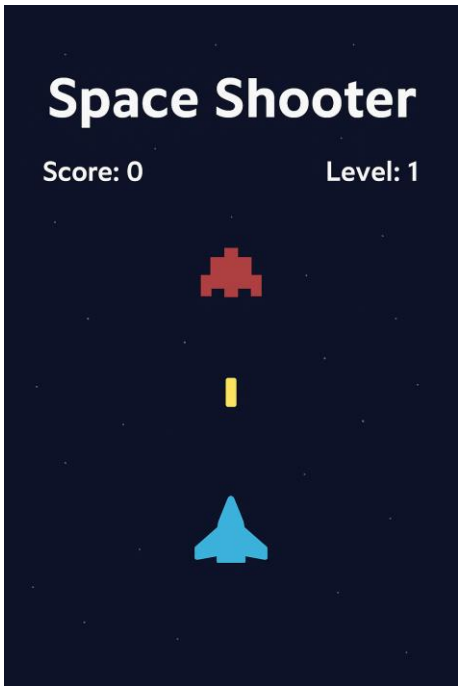
```
        self.canvas.move(self.player, PLAYER_SPEED, 0)


    def shoot(self, e=None):
        if not self.game_over:
            x1, y1, x2, y2 = self.canvas.coords(self.player)
            bullet = self.canvas.create_rectangle((x1+x2)//2-2, y1-10,
(x1+x2)//2+2, y1,
                                                  fill="yellow")
            self.bullets.append(bullet)


    def spawn_enemy(self):
        if self.game_over: return
        x = random.randint(20, WINDOW_W-40)
        enemy = self.canvas.create_rectangle(x, 0, x+40, 30, fill="red")
        self.enemies.append(enemy)
        self.root.after(max(300, ENEMY_SPAWN_DELAY - self.level*100),
self.spawn_enemy)


    def update(self):
        if self.game_over:
            return


        # Move bullets
        for b in list(self.bullets):
```

```
        self.canvas.move(b, 0, -BULLET_SPEED)
        if self.canvas.coords(b)[1] < 0:
            self.canvas.delete(b)
            self.bullets.remove(b)

    # Move enemies
    for e in list(self.enemies):
        self.canvas.move(e, 0, ENEMY_SPEED + self.level)
        x1, y1, x2, y2 = self.canvas.coords(e)
        if y2 > WINDOW_H:
            self.end_game()
        # Check collision with bullets
        for b in list(self.bullets):
            bx1, by1, bx2, by2 = self.canvas.coords(b)
            if bx1 < x2 and bx2 > x1 and by1 < y2 and by2 > y1:
                self.canvas.delete(b)
                self.canvas.delete(e)
                if b in self.bullets: self.bullets.remove(b)
                if e in self.enemies: self.enemies.remove(e)
                self.score += 10
                if self.score % LEVEL_UP_SCORE == 0:
                    self.level += 1
                break
```

```
        # Check collision with player
        px1, py1, px2, py2 = self.canvas.coords(self.player)
        if px1 < x2 and px2 > x1 and py1 < y2 and py2 > y1:
            self.end_game()

        # Update HUD
        self.canvas.itemconfig(self.score_text, text=f"Score: {self.score}")
        self.canvas.itemconfig(self.level_text, text=f"Level: {self.level}")

        self.root.after(50, self.update)

    def end_game(self):
        self.game_over = True
        self.canvas.create_text(WINDOW_W//2, WINDOW_H//2, text=f"Game
Over!\nFinal Score: {self.score}\nPress R to Restart",
                                fill="white", font=("Consolas", 18, "bold"))

    def restart(self, e=None):
        self.canvas.delete("all")
        self.__init__(self.root)

if __name__ == "__main__":
    root = tk.Tk()
```

```
game = SpaceShooter(root)
root.mainloop()
```

SCREENSHOTS:



RESULT :

Thus, a **2D Space Shooter Game** was successfully developed using **Python Tkinter**. The game integrates motion, keyboard events, collision detection, scoring, and level progression.