Progress Report: Shmup Game

Ashita, Rivva, Radha

Shmupters

July 18, 2024

Objective

Our aim is to explore the intersection of game development and programming by creating a shmup game using C# and Unity, fostering hands-on learning, and developing problem-solving skills. We delve into game design principles, C# scripting, and Unity's features to create an engaging gaming experience.
After trying to create a Space Invaders game in Unity, we decided to further enhance our learning by recreating the game using Pygame, a popular game development library in Python. This exercise helps in understanding different game development frameworks and reinforces the core concepts of game programming and OOP.

Key Differences Between Unity and Pygame

- **Language:** Unity uses C# while Pygame uses Python.
- **Environment:** Unity offers a robust integrated development environment (IDE) with visual tools, whereas Pygame is more code-centric with less graphical interface.
- **Complexity:** Unity provides more built-in functionalities and tools for complex game development, while Pygame requires more manual handling of game components.

What Have We Done So Far in Unity?

- **Familiarized with the Unity Interface and Core Concepts:**
  - Scene View: Navigated and manipulated objects, used transformation tools.
  - Game View: Tested gameplay in real-time, switched between Scene and Game views.
  - GameObjects: Created and manipulated player, bullet, and other objects.
  - Components: Added behaviors like SpriteRenderer, Rigidbody2D, and BoxCollider2D.
  - Prefabs: Created reusable templates for bullets, maintaining consistency.
- **Overcame Challenges through Iterative Testing and Debugging:**
  - Challenge 1: Understanding Unity's coordinate system.
  - Challenge 2: Synchronizing player and bullet mechanics.

What Have We Done So Far in Pygame?

- **Pygame Basics:**
  - Installed and set up Pygame.
  - Understood the Pygame event loop and handled events.
- **Game Development Fundamentals:**
  - Created a game loop with initialize, update, render, and handle events.
  - Controlled the game's frame rate using FPS.
- **Graphics and Rendering:**
  - Loaded and displayed images using Pygame's image module.
  - Understood sprite coordinates, sizes, and positions.
- **Game Logic and AI:**
  - Created enemy ships with basic AI.
  - Implemented a scoring system.
- **Object-Oriented Programming (OOP) Concepts:**
  - Defined classes and objects using Python.
  - Used inheritance to create a Sprite class.

Summary of Progress

- Game Setup: Initialized the Pygame environment and set up the main game window.
- Background: Successfully added a background image to the game.
- Enemy Sprites: Implemented and displayed enemy sprites on the screen.
- Audio effects: Added sound effects to complete the game.
- Bullets : Different bullets with different aliens ans ufo's .
- Display: Added UI text and score details