Index

Description	*Page no.*
Abstract, Introduction	*02*
Project architecture	*03*
Project DFD	*04*
Project Life Cycle	*05*
System Requirements, Features	*06*
System Description, Working of the Project	*07*
Project testing, Functional testing	*08*
How the game runs, Problem Faced	*09*
Limitations, Advantages, Future Works, Conclusion	*10*
Project outputs	*11*

Space Shooter Game with Python

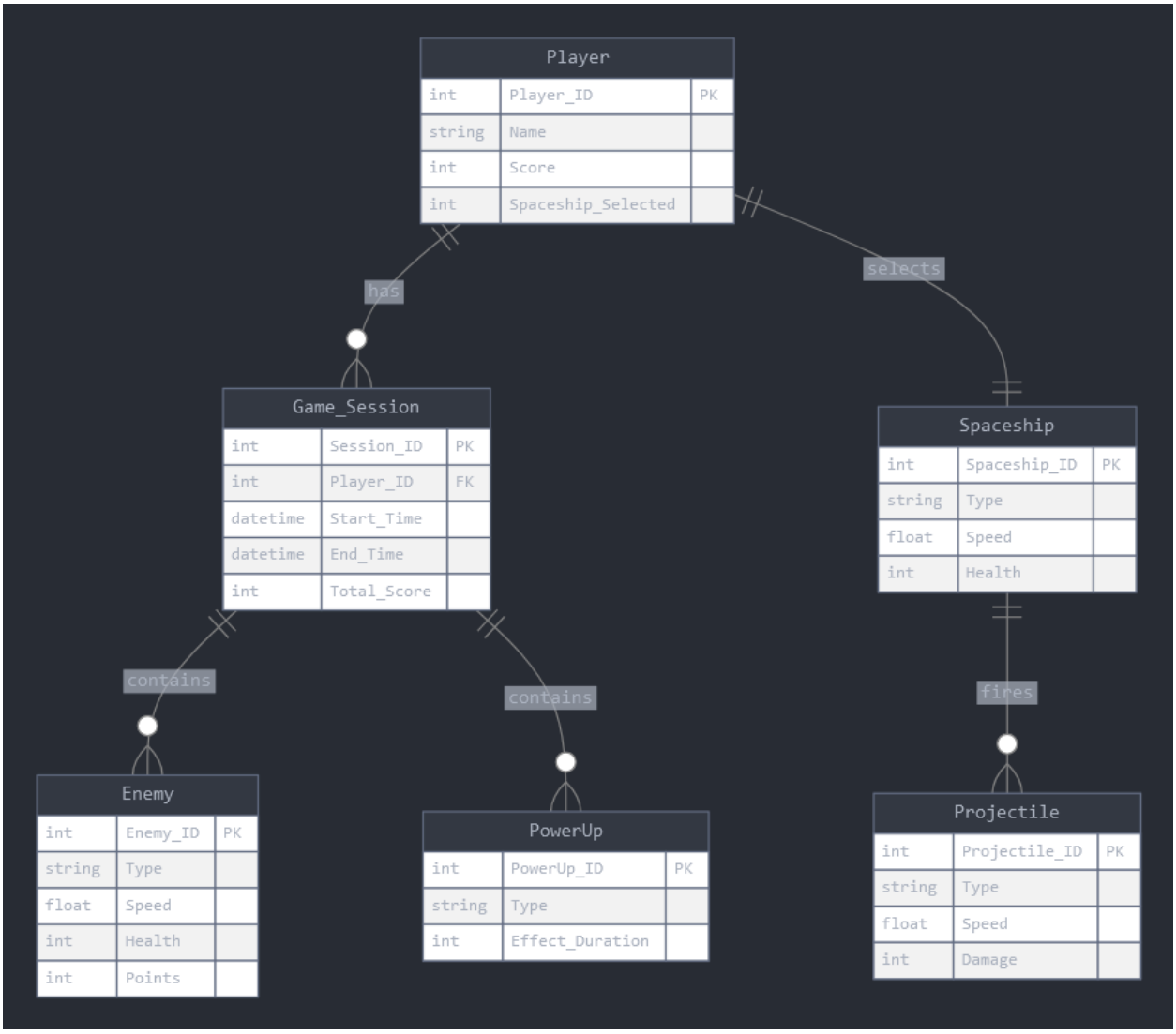
Abstract

Reminiscent of many "shoot 'em up" video games from the early eighties- Space Shooter is an arcade game developed using Python as a programming language. Although built to be simplistic, the mechanics of this game are pretty dynamic. Moreover, the game also has engaging gameplay. This project presents the development of a 2D space shooter game built using Python and Pygame. The game engages players in an immersive experience where they control a spaceship to defeat incoming waves of enemies and avoid obstacles. To enhance gameplay, players can choose between three unique spaceship designs, each selectable through simple key inputs. The game features dynamic mechanics, such as power-ups, varied enemy behaviors, and a scoring system to encourage competitive play. Additional functionalities include responsive controls, multiple weapon types, and realistic audio-visual effects. The project emphasizes modular programming for maintainability and scalability, integrating reusable components like a spaceship selector and asset manager. This report outlines the game design, implementation challenges, and key features, providing insight into the creative and technical processes involved. The resulting game is both a fun user experience and a testament to the versatility of Pygame as a game development framework.

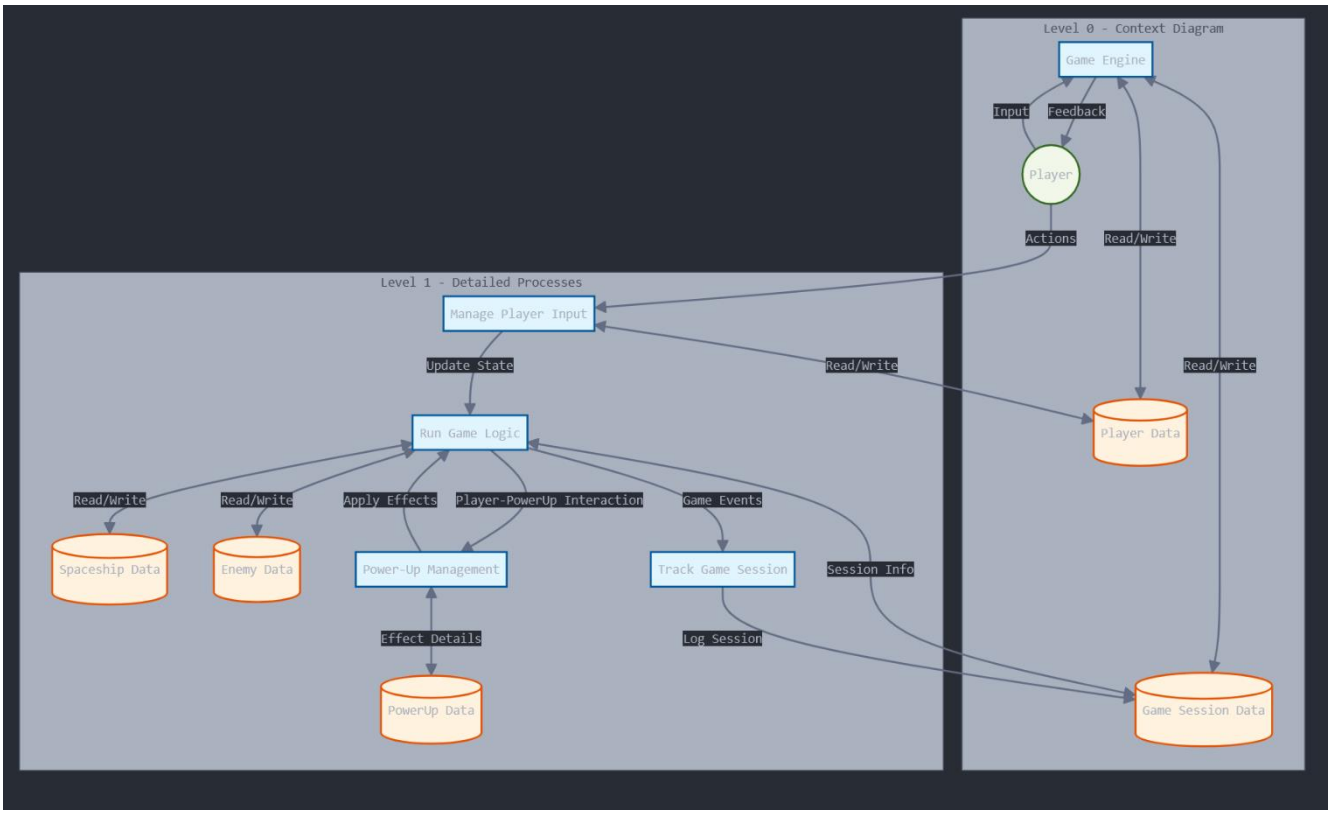
Introduction

Arcade games like space shooters remain popular for their simplicity and fast-paced action. This project recreates that experience by developing a 2D space shooter game using Python and Pygame. Players control a spaceship to defeat enemies, avoid obstacles, and collect power-ups, with the added feature of selecting between three unique spaceship designs.
The game incorporates dynamic mechanics such as enemy waves, responsive controls, and a scoring system to enhance player engagement. With modular programming and reusable components, the project demonstrates the potential of Pygame as a versatile framework for game development.

Project architecture:

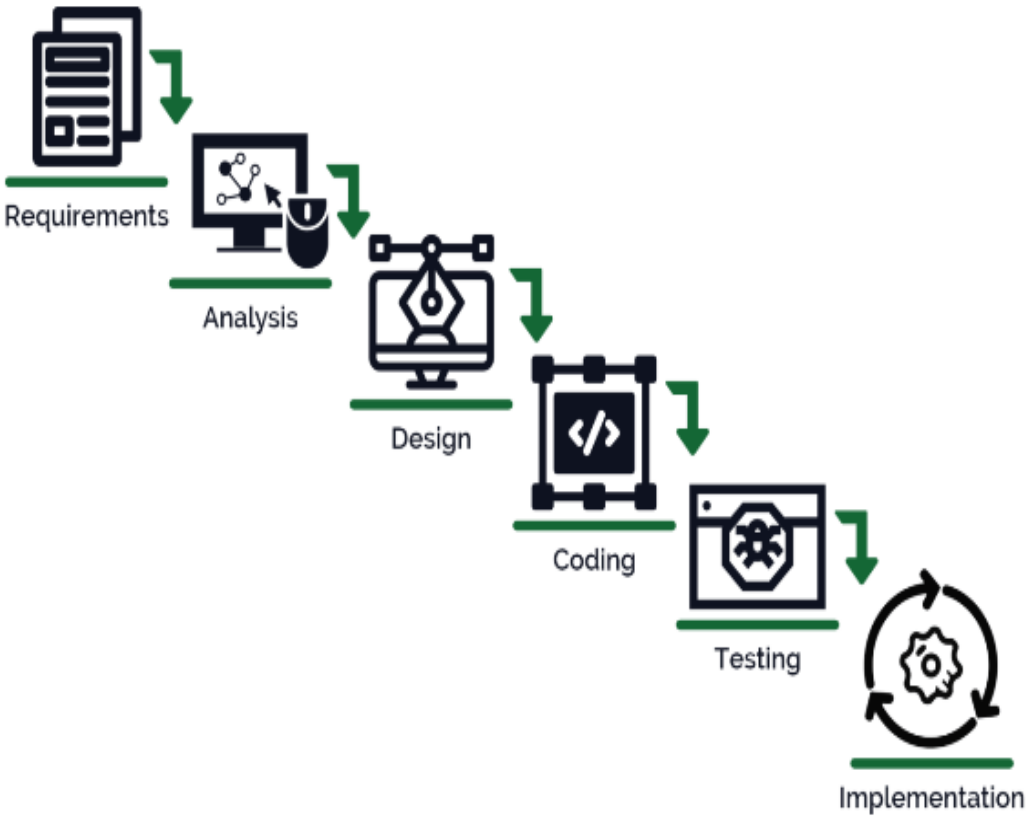


Project DFD


Level 0 - Context Diagram
Game Engine
Input
Feedback
Player
Actions
Read/Write
Read/Write
Player Data
Level 1 - Detailed Processes
Manage Player Input
Update State
Read/Write
Run Game Logic
Read/Write
Read/Write
Apply Effects
Player-PowerUp Interaction
Game Events
Spaceship Data
Enemy Data
Power-Up Management
Track Game Session
Session Info
Effect Details
Log Session
PowerUp Data
Game Session Data

Project Life Cycle

The waterfall model is a classical model used in system development life cycle to create a system with a linear and sequential approach. It is termed as waterfall because the model develops systematically from one phase to another in downward fashion. The waterfall approach does not define the process to go back to the previous phase to handle changes in requirement. The waterfall approach is the earliest approach that was used for software development.



System Requirement

I. Hardware Requirement

i. *Laptop or PC*

- *I5 processor system or higher*
- *4 GB RAM or higher*
- *100 GB ROM or higher*

II. Software Requirement

ii. *Laptop or PC*

- *Windows 10 or higher*
- *Vs code*
- *Python 3.11*
- *Pygame library*

Features

a) *Dynamic Gameplay:*
As players high score increases, the difficulty of the game gradually gets harder.

b) *Power-ups:*
While one type of power-up replenishes the player's health, the other increases their firepower. Thereby, ensuring that the game remains interesting.

c) *Complex Mechanics:*
Small asteroids have a lesser toll on the player's health bar while larger asteroids deal comparatively heavier damage.

System Description

The system comprises of 1 major module with their sub-modules as follows:

User:

- ***Main menu screen***
  - *Introductory screen*
  - *Logo of the app*
  - *Start//load/quit options*

- ***Gameplay screen***
  - *Playing the game*

- ***Pause menu screen***
  You can quit the game or go to the menu screen

Working of the Project

Pygame is a cross-platform set of Python modules which is used to create video games.
It consists of computer graphics and sound libraries designed to be used with the Python programming language.
Pete Shinners officially wrote Pygame to replace PySDL.
Pygame is suitable for creating client-side applications that can be potentially wrapped in a standalone executable.

The project uses the pygame module. We use images and animate the images using pygame module. We use pygame module to do button mapping and several other display function. We use classes to call functions after every iteration of the game level

Project testing

For your space shooter game, we combined of both white-box and black-box testing, as each serves a distinct purpose during the testing process

- *Start with white-box testing to ensure all the core logic (movement, collision, scoring) works.*
- *Follow up with black-box testing to simulate real-world gameplay scenarios and identify issues from the player's perspective.*

Functionality testing

- ***Functional Testing:** Verify that all game functionalities work as intended.*
- *Usability Testing: Ensure the game provides an enjoyable and user-friendly experience.*
- ***Performance Testing**: Ensure the game performs well under different conditions.*
- ***Regression Testing**: Ensure new features or updates don't break existing functionality.*
- ***Stress Testing**: Test the game's behavior under extreme conditions.*
- *Compatibility Testing: Ensure the game runs on different platforms and configurations.*
- ***Unit Testing:** Test individual game components in isolation.*
- *Integration Testing: Verify that different game components work together seamlessly.*
- ***Playtesting:** Collect feedback from actual players.*

How the Game runs:

- *Press shift + R + R.*
- *Press Enter.*
- *Press enter and write your name.*
- *Press "Enter".*
- *Press the space key to shoot and the left and right arrow to move.*
- *2 types of power-ups are included (shield, and update gun power).*
- *4 sizes of Asteroid and 3-speed categories.*

Problem Faced:

- *Couldn't add video background. (unsolved)*
- *After adding "choose spaceship" the game stopped running. (unsolved)*
- *Managing sound effects. (Solved)*
- *Spaceship movements in the Y axis. (unsolved)*
- *Unable to implement full Data connection according to DFD. (unsolved)*

Limitations

- *Screen is not expandable*
- *Can't record score in history*
- *Spaceship choosing option is not available*

Advantages

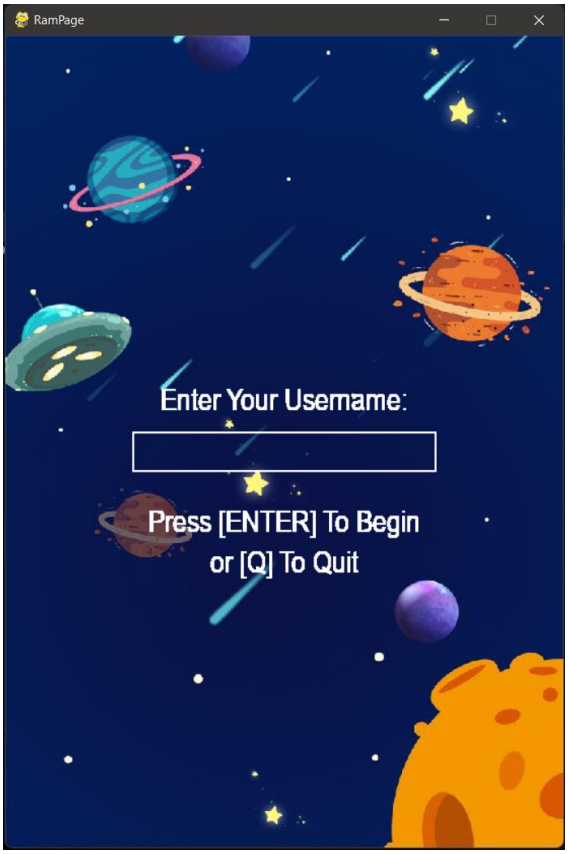
- *Runs fast.*
- *Easy to play.*
- *Increase user focus.*
- *Easy access using a keyboard only.*

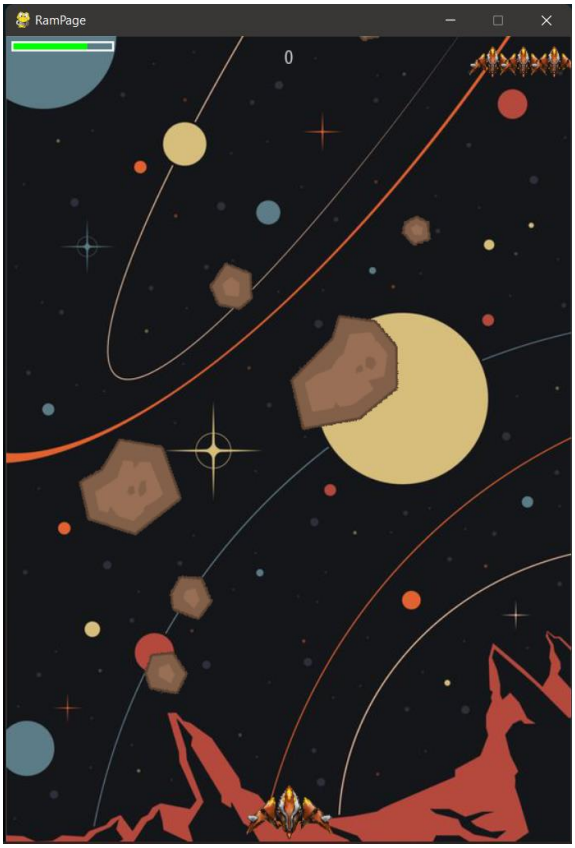
Future work

- *Add the spaceship choosing option.*
- *Big gaming screen.*
- *Add more obstacles.*
- *More attractive background.*
- *Add layers of enemies.*
- *Add game levels.*
- *Enable cloud gaming.*

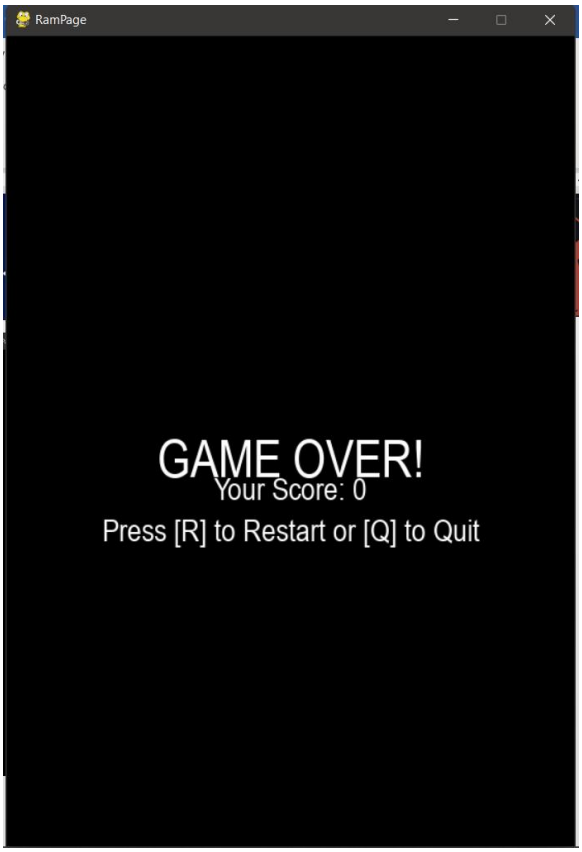
Conclusion

The space shooter game with Python is now in the basic stage. It is simple to run and play. Overcoming the limitations and problems will allow the game to become more official. Adding more features will increase user interactions and make this game more attractive.


RamPage
Enter Your Username:
Press [ENTER] To Begin
or [Q] To Quit


RamPage
0


RamPage
GET READY, ht!


RamPage
GAME OVER!
Your Score: 0
Press [R] to Restart or [Q] to Quit