PYGAME FINAL PROJECT

COMP6047001 – Algorithm and Programming

RENJIRO KUNIO PRADANA HANDOKO
STUDENT ID - 2902740591

1. Project Specification

1.1 Problem Description

Arcade style-games require real- time input, object interaction, and constant state updates, which makes them a suitable subject for applying algorithmic thinking and programming concepts. However, implementing these systems requires careful structuring of logic, data, and a modular design.

The problem addressed by this project is how the design and implementation of a simple 2D game demonstrates core programming principles like object-oriented programming, input validation, and control flow. The project focuses on creating an interactive space shooter game where the player controls a space ship and avoids/destroys enemies, and tracks their health and score throughout gameplay.

1.2 Project Objectives

The objective of this project is as follows:

- To develop a python-based application using the Pygame library
- To apply object-oriented programming concepts through the use of custom classes and methods.
- To implement real-time user input handling and game state updates.
- To utilize data structures like sprite groups and lists to manage in-game objects.
- Demonstrate modular program design by using multiple python files.
- Apply basic algorithms for collision, detection, scoring, and health management.

1.3 Project Scope

This project focuses on a singleplayer 2D space like shooter game. The scope of the project includes:

- Player movement and shooting mechanics.
- Enemy spawning and movement.
- Collision detection with the player sprite and the enemies and their projectiles.
- Health and score tracking.
- Use of graphical and audio assets.

2. Solution Design

2.1 Overall Design Approach

The program was designed as a real-time, application using the Pygame framework. The program follows a game loop, which continuously processes user input and update the game state in real-time, then it prints visual elements to the screen. This allows the game to respond immediately to player actions while maintaining smooth gameplay.

Object-oriented programming principles were applied in order to separate responsibilities between different game components. Each entity in the game, the player, the enemies, and the projectiles, is represented by a dedicated class. This design improves the code by making it modular and readable while also making it easier to add additional features into the game.

2.2 Program Structure

The program Is made modular across several Python files to ensure readability and this makes it easier to find specific code. Each file is responsible for a specific aspect of the game logic. The main.py starts the game environment, load required assets, and control the main loop.

Supporting modules support game entities and handle specific behaviors like movement, collision detection, and rendering. This structure is modular and allows individual components to be developed and tested independently, while still functioning together as a complete application.

2.3 Data Structures Used

Several data structures were used in the project to manage game data:

- Lists and Sprite Groups: Used to store and manage collections of enemies, bullets, and other game objects. These structures allow efficient iteration and collision checking.
- Classes and Objects: Custom classes represent different game entities such as the player character and enemies, encapsulating their attributes and behaviours.
- Dictionaries: used to organize configuration data or asset references.

2.4 Algorithms Implemented

The project implements several fundamental algorithms to support gameplay:

- Collision Detection: Algorithms provided by Pygame are used to detect collisions between sprites, like bullets hitting enemies or enemies colliding with player.
- Enemy Spawning Logic: Enemies are generated at controlled intervals using the random module to increase game difficulty.
- Movement Algorithm: Position update are calculated based on player input and predefined movement speeds.
- Scoring and Health Management: simple conditional logic updates the player’s score and health based on in-game events.

3. What was implemented and How it Works

3.1 Core features

The project implements a fully functional 2D space shooter game with some core gameplay features. The player controls a spaceship that can move horizontally within the game window using keyboard input. The player is also able to fire projectiles to destroy incoming enemies.

Enemies are continuous spawned during gameplay and move towards the bottom side of the screen. When projectiles hi an enemy, the enemy is removed from the game, and the player’s score is increased. If an enemy hits a player the players health is increased. The game continues until the player’s health reaches 0, when that happens the game ends.

3.2 Program Execution

The program begins by initializing the Pygame environment and loading all required assets, including images and sound effects. The main game loop then starts and runs continuously until the player exits the game or a game-over condition is reached.

Within each game loop, the following steps occur:

1. Input Handling: Keyboard input is detected and processed to control player movement and attacks.
2. Game state update: Positions of the player, enemies and projectiles are updated whenever something changes. Collision detection is performed to determine interactions between game objects.
3. Rendering: All game elements are drawn to the screen, including the background, player, enemies, and score display.
4. State validation: Health and score values are checked to determine whether the game should continue or end.

3.3 Object-oriented Programming Implementation

Object-orientated programming is a key aspect of the project design. Each game entity is implemented as its own class with its own attributes and methods. For example, the player class manages position, movement, and health, while enemy classes handle movement behaviour and collision response.

Encapsulation is used to group related data and functionality within classes, reducing code duplication and improving maintainability. This approach allows game entities to be updated independently while still interacting correctly within the game environment.

3.4 Input Validation and Error Handling

Player movement is restricted to the visible game window to prevent the player ship from moving off-screen. Conditional check is used to validate collisions and ensure that game state updates occur only when appropriate.

Basic error handling is also present to prevent runtime errors during loading and object interactions. These measures contribute to a more stable and reliable application during execution.

3.5 User of modules and External Libraries

The game utilizes the Pygame library to manage graphics, sound, input handling, and sprite interaction. Additional Python modules from the standard library are used where necessary to support timing and control.

By leveraging external libraries and organizing code into modules. The project demonstrates an understanding of how to extend Python functionality beyond the core language.

4. Evidence of Working Program

Figure 1: Game start screen showing the player spaceship and initial score.

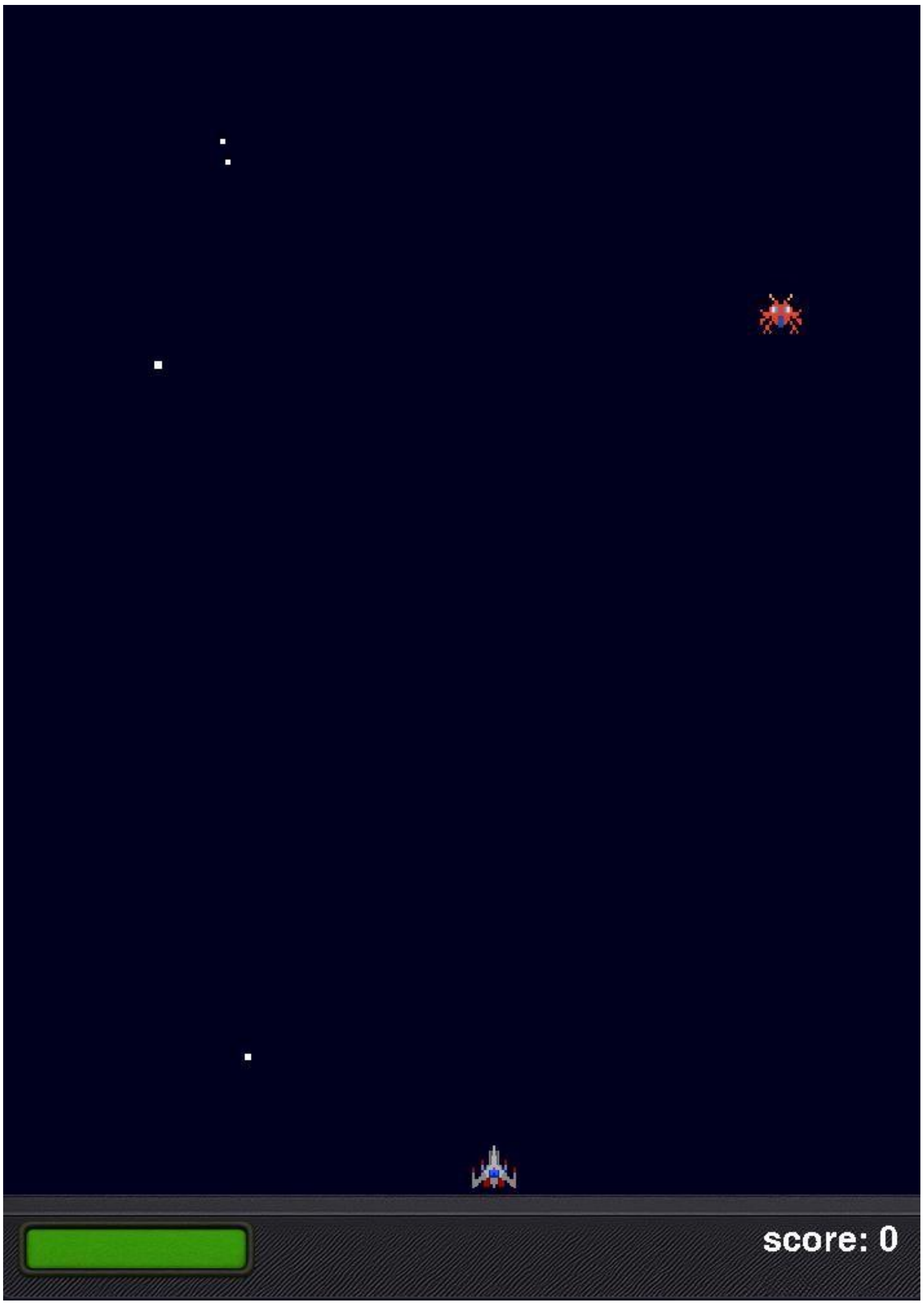


Figure 2: Gameplay in progress with multiple enemies and active projectiles

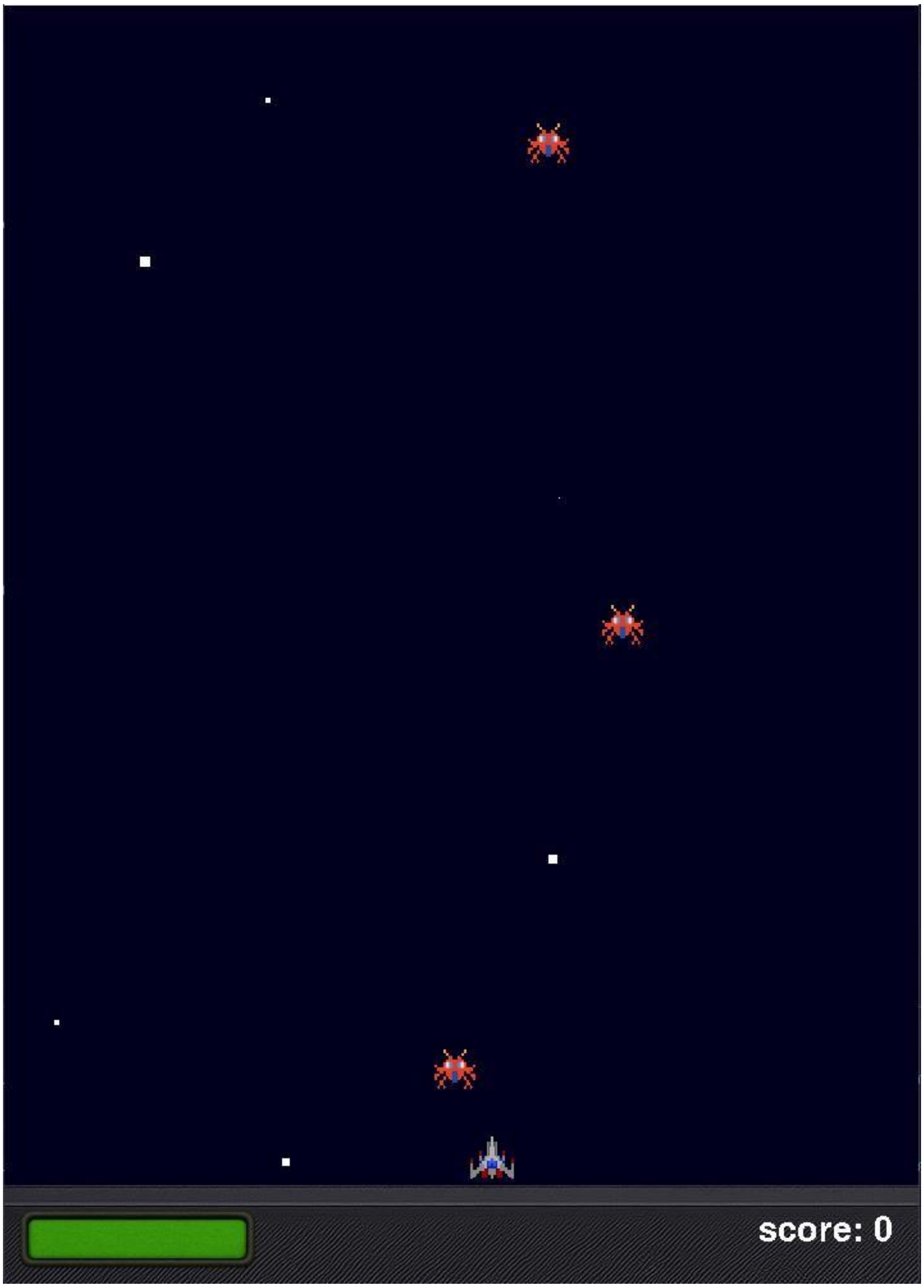
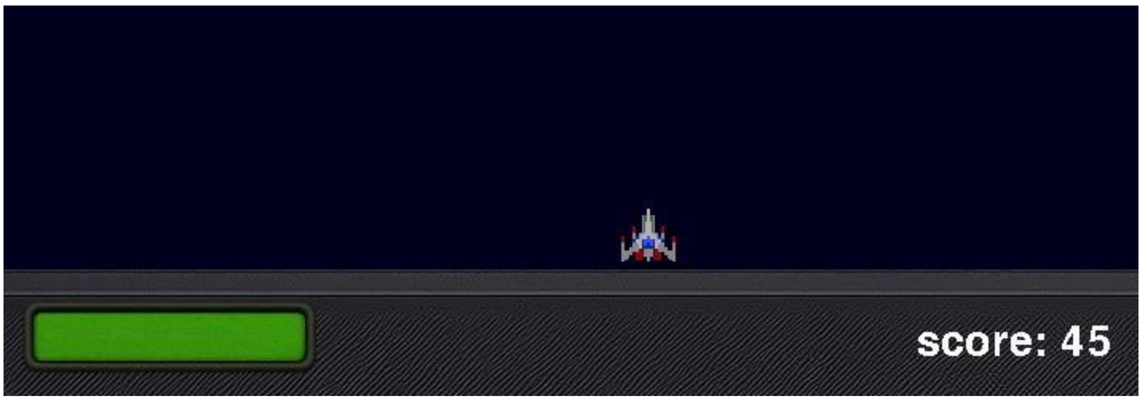


Figure 3: Collision even demonstrating score increase



5. Statement on the Use of Artificial Intelligence

AI Tool Used: ChatGPT 5.2
Purpose:

- General Outline for Final PDF report
    - Prompt Used: “Please create an Outline for a project report of the attached pygames folder”
    - The AI output was used as a guide to making each section of this document.
- Code comments
    - Prompt Used: “Please help me write comments for these blocks of code”
    - The AI output was used as a comment for code blocks in python file

Github Repository Link

https://github.com/RenjiroKunio/pygame_final_project/tree/master

Poster

Pygame Space Shooter

A Python-Based 2D Arcade Game

Renjiro Kunio Pradana Handoko
2902740591
COMP6047001 – Algorithm and Programming

Project Overview

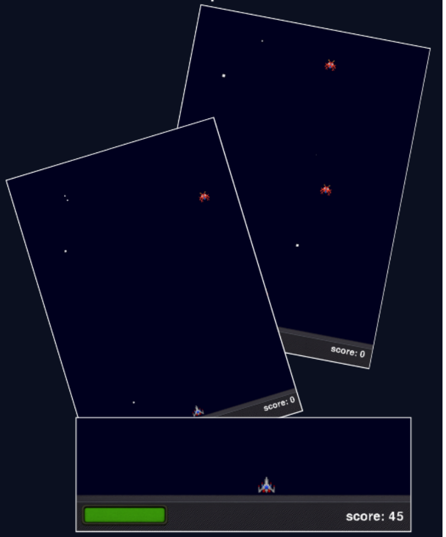
This project is a 2D arcade-style space shooter game developed using Python and the Pygame library.
The player controls a spaceship and survives waves of enemies while managing health and score.

Design & Implementation

The game uses a real-time game loop that processes input, updates game state, detects collisions, and renders graphics continuously.

Results

The completed game runs smoothly and demonstrates correct collision detection, responsive controls, and dynamic score updates.



Objectives

- Apply Python programming concepts
- Implement object-oriented design
- Use data structures for game entities
- Handle real-time input and collisions

Key Concepts

- 🧱 Object-Oriented Programming
- 📦 Data Structures
- ⚙️ Algorithms
- 🧩 Modular Code

Conclusion

This project demonstrates practical application of Python programming and problem-solving through game development.