Deliverable 4 – Kayla Cheung

**Retrospective Report – Space Shooter Game**

Building my **2D Space Shooter Game** with Python and Pygame has been one of the most rewarding and educational experiences I've had in game development. The goal was to create a fast-paced arcade-style shooter where the player controls a spaceship, fights off waves of enemies, and progresses through increasingly difficult levels. The final game includes **4 challenging levels, boss battles, score tracking, and a dynamic scrolling background**, all of which came together through a lot of planning, coding, testing, and iteration!

**What Went Well**

One of the biggest successes was implementing a clean, modular structure using object-oriented programming (OOP). I was able to separate different responsibilities/ functionalities into different classes, like Spaceship, Alien, Boss, Projectile, and various screen components, which made the code easier to manage and scale.

Using the arrow keys and spacebar for movement and shooting also gave the game an intuitive feel. The scrolling background added an extra layer of polish and made the gameplay feel more dynamic and alive.

Level transitions and the congratulatory screen added a real sense of progression and accomplishment. The health and lives system worked well to provide challenge without being too punishing, and I was pleased with how the scoring system rewarded player effort.

**Challenges Faced**

Despite the successes, I definitely faced a few challenges along the way. One of the hardest parts was balancing enemy wave spawning and boss fights without breaking the game's pacing. I also ran into issues managing state transitions between levels, especially making sure each boss was properly handled, and new waves didn’t interfere with boss logic.

Implementing the boss blood attack and making sure it triggered on a timer while allowing multiple bosses to function independently took some trial and error. Another hurdle was ensuring collisions were detected and handled consistently, especially with multiple enemies, projectiles, and blood attacks flying across the screen.

Testing specific levels (like jumping straight to level 3 or 4) without playing through earlier ones also required modifying the game logic temporarily, which helped but showed me the importance of adding a debug or level select feature early on in development.

**Lessons Learned**

This project taught me the importance of **designing around modularity and scalability from the beginning**. By creating reusable components, I was able to expand the game (adding new enemies, levels, and features) without completely rewriting core systems.

I also learned that separating the user interface (menus, transitions, game over screens) into its own set of classes helps keep game logic clean and manageable. More importantly, I gained a deeper appreciation for testing and debugging iteratively (not everything will work on the first try but it will work eventually!!!)

If I were to start again or continue development, I’d focus on building out more special mechanisms, like sound effects and power-ups, to increase player engagement. I’d also consider adding a high score system and more diverse attack patterns to enrich gameplay.