**Post-Mortem Analysis: Angry Birds Remake Project**

**Introduction**

In this project I aimed to recreate Angry Birds using Box2D and SFML, with features such as different levels and projectile types. The main objectives of the assignment were to demonstrate proficiency in using a physics engine, and develop a functional game with multiple levels and interactive elements that utilize said physics engine.

**Implementation Challenges**

One of the main challenges I faced was memory management and managing project files. As discussed with Alexa, I had an issue early on where the release build wouldn't work, so by the final day, I had everything finished only in debug mode. This was because I based the project on a previous Box2D project from last year, which also had issues with its release build. To overcome this, I ended up needing to switch my project to the example project, which had working release and debug modes. However, this solution resulted in broken collision detection, causing crashes with invalid memory pointers when killing the final enemy in level one.

Another significant challenge was fixing these collision errors. I addressed this in many ways over two days, including attempting to switch to purely SFML (which felt like not doing the assignment properly). The final solution was removing my boxes and enemies’ collision functions and only having the projectile collision functions, as well as changing it to only check collision between Object B => Object A, not both ways.

**Software Engineering Approach**

My development process utilized many iterations on the levels and collisions to eventually find both levels I'm happy with and a working collision system. Key software engineering principles I applied included modular design, separation of concerns, (being able to easily separate out the concern I had was made easy through the use of github, where each feature I had was its own branch, so I could easily revert back or separate just that issue) and iterative development. This approach benefited the project by allowing me to isolate and fix issues more effectively, as well as making it easier to adjust and balance gameplay elements.

**Known Issues in Release Build**

One known issue in the release build is the need for a buffer between switching levels. This 1-second buffer ensures that all physics bodies and sprites are properly deleted between switching levels, preventing immediate switches that could cause crashes. Another issue is that the explosive projectile type stopped working through all the changes I had to make to collision detection to get it to work without crashing the project.

Another area for improvement is definitely memory and collision management. The rest of the project was relatively straightforward, but getting stuck on these issues for multiple days caused significant problems.

**Lessons Learned**

The most valuable lesson I learned is the importance of starting with a robust collision system and an even more robust memory management system. These are crucial for preventing crashes, improving performance, and making debugging easier. If I were to approach this project again, I would definitely start by implementing and thoroughly testing these systems with basic elements like a slingshot and boxes before adding more complex features. This is what I plan to do for future projects, properly test these fundamental systems before expanding the game's functionality.

**Conclusion**

Overall, the Angry Birds remake project was a challenging memory management experience. Despite the difficulties faced with collision detection and memory management, I was able to create a functional game with multiple levels and projectile types. The experience gained from this project will be invaluable in future game development (I’ll have to figure out how to do proper memory management in game engine), particularly in emphasizing the importance of solid foundational systems and thorough testing throughout the development process.