# Space Escape Postmortem:

## Overview:

Space Escape is a retro style, pixel art, action game I developed. The game places the user in space on an alien planet after their ship has crashed. The core gameplay surrounds the player fighting against the aliens, collecting ship parts to repair their ship, and jumping between planets with rift vials, all while surviving with limited lives and health.

The game targets sci-fi enthusiasts and action game lovers. The permadeath mechanic and limited lives system create tension and meaningful player choices throughout the experience.

The game was built scene by scene, but in no particular order. The scenes were worked on by priority, which was allocated based on the game’s main needs, e.g. level was created first as this was the scene the entire game was based around. Other scenes like the instructions or game over screens were created last as those weren’t as crucial. The scene system kept everything organised, even with development being flexible and random. This development approach ensured that the most critical components were addressed first.

This project initially had a larger scope with about 20 different levels and 5 different planets, as well as a larger boss fight at the end. However, as I was developing this game solo, I had to make a decision about scope reduction while ensuring the core gameplay experience and quality stayed.

The technical design of Space Escape centres around a tile-based system that supports easy generation and a state machine for handling the game flow. These systems provide the foundation for the games randomised levels, enemy behaviours, scene playthrough, and distinct planet effects that create a unique experience with each playthrough.

## What Went Right:

Tile Parsing:

The tile parser I developed for level creation works well throughout the project. This system efficiently translates level data from a text file into the game, allowing for quick iteration on level designs and environmental features. The parsers reliability meant I could focuses on creative aspects rather than fighting with technical implementations when designing new environments.

Drop System:

The visual implementation of item drops became one of the games aesthetic highlights. The animation of the item dropping on the tiles ensured players could easily recognise it was something to obtain. This enhanced both the functional gameplay of gathering the resources and contributed to the overall game experience.

Player Movement:

The player sprites multi-directional movement animations transitioned smoothly, making navigation feel intuitive and precise. This core element of the gameplay provided a solid foundation that enhanced other aspects of the game, especially during combat.

Level Generation:

The completely randomised level generation system significantly enhanced the replay value without requiring manual design of several distinct levels. The algorithm made ensured that each level had its own map, effect, difficulty, and style.

Weapon Progression:

The weapon progression system functions as intended, creating a power curve throughout the game. Players could visually see the difference between the weapons and the boss drop mechanics rewarded skilled play with upgrades. The variety of weapons maintains player interest while also preserving combat balance.

## What Went Wrong:

Initial Scope:

The original scope of the game was far too large for an individual developer working within the specified timeframe. I had to make the decision to reduce the number of features to ensure the game could be completed to a high-quality standard while also remaining compelling to play. While this was the right decision, it required reworking a couple systems that I had already mocked up, which consumed valuable development time.

Texture Rendering:

A persistent issue throughout development was the improper loading sequence of image textures and created text textures. Textures that loaded after the text rendering loaded as lines and with coloured stripes instead of the proper image texture. This was a large issue as text was used in almost every scene, so this caused majority of the image textures to load with issues. To fix this, I changed how the texture class rendered textures. I added a reset to the glPixelStorei to ensure that textures were loaded with the appropriate pixels. This solved the problem and ensured that textures initialised after text rendered with the proper texture.

Item Drops:

The original design for weapon and ship part drops needed some revision as they did not function as they were supposed to. I split both the weapon and ship parts into separate systems for the drop and hud to ensure that the items were dropped and shown correctly on the hud. This refactoring took time away from other planned features.

Boss Sprite:

The initial boss sprite had a problematic hitbox due to a large amount of white space above the sprite (added to create a jumping effect). This created confusing scenarios where the player would hit what looked like empty space and still damage or be damaged by the boss. Fixing this required me to change the sprite asset to include less whitespace.

Controller Input:

After Implementing Xbox controller input, many input issues started. On one of the scenes – the instructions scene – there are arrows that allow the user to switch through the different play instructions. Once the Xbox controller input was implemented, the instructions started to flick through each one without any input. To fix this, I added a check before the Xbox input to ensure that a controller was connected to the device.

Boundary Checking:

With my initial collision detection code, players would frequently get stuck on the level boundaries. To fix this, I added a push back system that pushed the player away from the boundary to ensure they didn’t get stuck. This also allowed the player to slightly slide along the boundary when walking against it.

## Lessons Learnt:

Realistic scope:

The main lesson learnt is to ensure that the scope is realistic from the beginning. Starting with a focused core feature set and expanding only after the crucial features are complete is more efficient than cutting features later due to time. For future projects, I’ll ensure that the scope is realistic and use a better priority system to be more disciplined about adhering to the core functionality before adding extra features.

Cross Platform Input:

The challenges with the Xbox controller implementation taught me to integrate all planned input methods from the beginning rather than adding them later. Future projects will include testing for all input types early on to test how they inputs interact with the UI elements and menus.

Collision Testing:

The boundary collision issues have shown me that what works in theory doesn’t always work in practice. I’ve learned to implement more comprehensive testing, focusing particularly on edge cases where players might get stuck. Implementing safety systems like the push back system early on can help to reduce reworking later.

Sprite Designs:

The issue with the boss hitbox has shown me that visual design and technical implementation need to be considered together. In future projects, I’ll ensure that white space is considered in all sprite creations to reduce issues with the hit box which will ensure that visual and functional gameplay will align from the start.

## Conclusion:

Developing Space Escape has been both challenging and rewarding. While the final product successfully captures the retro, pixel art feel with sci-fi elements I imagined, the development process revealed important areas for improvement in my approach to game development.

The biggest insight gained was how important it is to balance ideas with practical constraints, especially when developing solo. The unique planet effects and weapon progression scheme shows that well-executed mechanics can create compelling gameplay without requiring a huge scope.

The technical implementation of Space Escape taught me numerous lessons about game architecture. The tile parsing system proved to be a robust foundation that supported the entire game, demonstrating that spending time on a core systems pays off during the development.

If I started over, I would maintain the core concept but implement a more structured development process with feedback from people outside of the development, stricter scope control, and more extensive prototyping of key functions before implementing them into the product. The technical challenges I encountered have expanded my problem solving knowledge, particularly around testing and debugging complex game systems. Establishing proper collision detection, managing input across multiple platforms, and creating consistent visual systems would all receive more attention from the beginning.

Despite the challenges faced, the lessons learnt throughout this project have provided good experience that will improve my future development in games. Working within constraints ultimately led to creative solutions that enhanced the final game.