Performance Considerations for Metroidvania Design
By DJ Standish

Summary

This document attempts to identify several performance issues that are likely to arise during the development of a Metroidvania. It will then identify strategies that have been implemented to deal with each issue, and then discuss the author's personal implementation of those strategies.

Concerns

Given modern hardware, optimization for 2D games can range from unnecessary to fundamental to your success. Complex lighting engines, large rooms with multiple enemies, and fine textures, all play a significant role in how well your game functions on a variety of hardware. However, some of these things may be integral to the game that you're trying to design.

Strategies

Let's take a look at each individual concern. Having a large room with multiple enemies is bound to happen at some point in any metroidvania, so we can avoid taxing our GPU by only drawing the enemy while it is in view of the character. Similarly we can avoid expensive pathfinding calculations in our CPU by adding a function that only activates the AI if the enemy is within a certain range of the player.

Light baking is a method of calculating static lighting and storing it into a lightmap, and only rendering dynamic lighting in real time. This reduces power usage and GPU resource expenditure. This can be particularly useful in 2D games, as lighting isn't often dynamic.

If you're seeing significant CPU performance issues with regard to your textures, it may be that you have too many individual objects rendering at once. Consider combining multiple static objects into a single mesh, so that they're loaded as a single unit. This won't improve GPU performance, but may significantly improve CPU performance.

Personal Strategies

Godot has a built in handler for offscreen objects, that turns off their AI and rendering if you can't see them, or rather the area that you designate around them. This allows the AI to activate while the enemy is just off-screen, creating a seamless transition for the player, while still conserving resources.

I don't personally plan to use any dynamic lighting for my game, and my textures and models are all sufficiently low-resolution, which means I should see little impact on performance.

https://code.tutsplus.com/code-optimizations-for-game-development-basic-structures-and-mindsets--cms-30760t
https://unity.com/how-to/advanced/optimize-lighting-mobile-games
https://docs.unity3d.com/2019.1/Documentation/Manual/OptimizingGraphicsPerformance.html