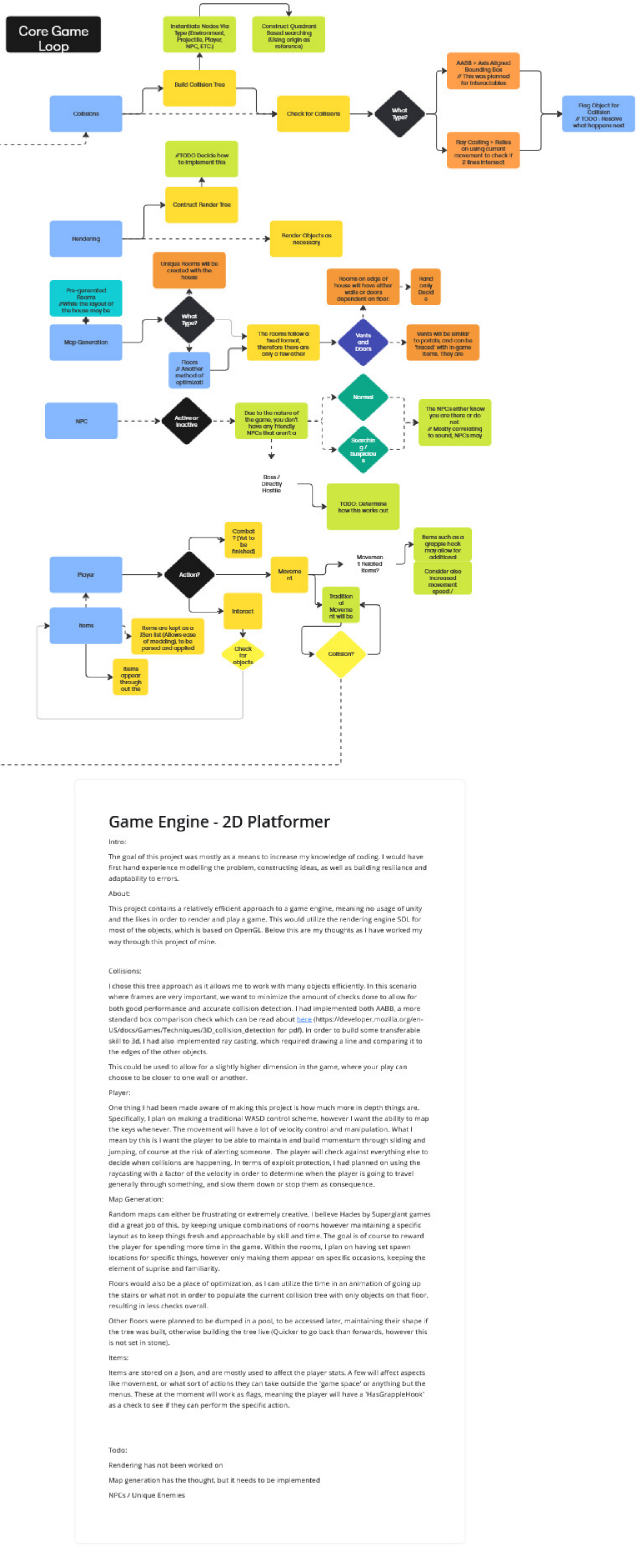

Game Engine - 2D Platformer

Intro:

The goal of this project was mostly as a means to increase my knowledge of coding. I would have first hand experience modelling the problem, constructing ideas, as well as building resiliance and adaptability to errors.

About:

This project contains a relatively efficient approach to a game engine, meaning no usage of unity and the likes in order to render and play a game. This would utilize the rendering engine SDL for most of the objects, which is based on OpenGL. Below this are my thoughts as I have worked my way through this project of mine.

Collisions:

I chose this tree approach as it allows me to work with many objects efficiently. In this scenario where frames are very important, we want to minimize the amount of checks done to allow for both good performance and accurate collision detection. I had implemented both AABB, a more standard box comparison check which can be read about here (https://developer.mozilla.org/en-US/docs/Games/Techniques/3D_collision_detection for pdf). In order to build some transferable skill to 3d, I had also implemented ray casting, which required drawing a line and comparing it to the edges of the other objects.

This could be used to allow for a slightly higher dimension in the game, where your play can choose to be closer to one wall or another.

Player:

One thing I had been made aware of making this project is how much more in depth things are. Specifically, I plan on making a traditional WASD control scheme, however I want the ability to map the keys whenever. The movement will have a lot of velocity control and manipulation. What I mean by this is I want the player to be able to maintain and build momentum through sliding and jumping, of course at the risk of alerting someone. The player will check against everything else to decide when collisions are happening. In terms of exploit protection, I had planned on using the raycasting with a factor of the velocity in order to determine when the player is going to travel generally through something, and slow them down or stop them as consequence.

Map Generation:

Random maps can either be frustrating or extremely creative. I believe Hades by Supergiant games did a great job of this, by keeping unique combinations of rooms however maintaining a specific layout as to keep things fresh and approachable by skill and time. The goal is of course to reward the player for spending more time in the game. Within the rooms, I plan on having set spawn locations for specific things, however only making them appear on specific occasions, keeping the element of suprise and familiarity.

Floors would also be a place of optimization, as I can utilize the time in an animation of going up the stairs or what not in order to populate the current collision tree with only objects on that floor, resulting in less checks overall.

Other floors were planned to be dumped in a pool, to be accessed later, maintaining their shape if the tree was built, otherwise building the tree live (Quicker to go back than forwards, however this is not set in stone).

Items:

Items are stored on a Json, and are mostly used to affect the player stats. A few will affect aspects like movement, or what sort of actions they can take outside the 'game space' or anything but the menus. These at the moment will work as flags, meaning the player will have a 'HasGrappleHook' as a check to see if they can perform the specific action.

Todo:

Rendering has not been worked on

Map generation has the thought, but it needs to be implemented

NPCs / Unique Enemies