CS Survivor Systems Analysis

Teal Team 6

Brief Introduction

Our game is called CS Survivor.

- Like a zombie survival, wave clearing game, but instead of defending yourself against zombies, you're making your way through the perilous path that is a computer science degree.

The goal of CS Survivor is to purchase, build, and upgrade weapons and abilities to conquer the battlefield of the computer science degree.

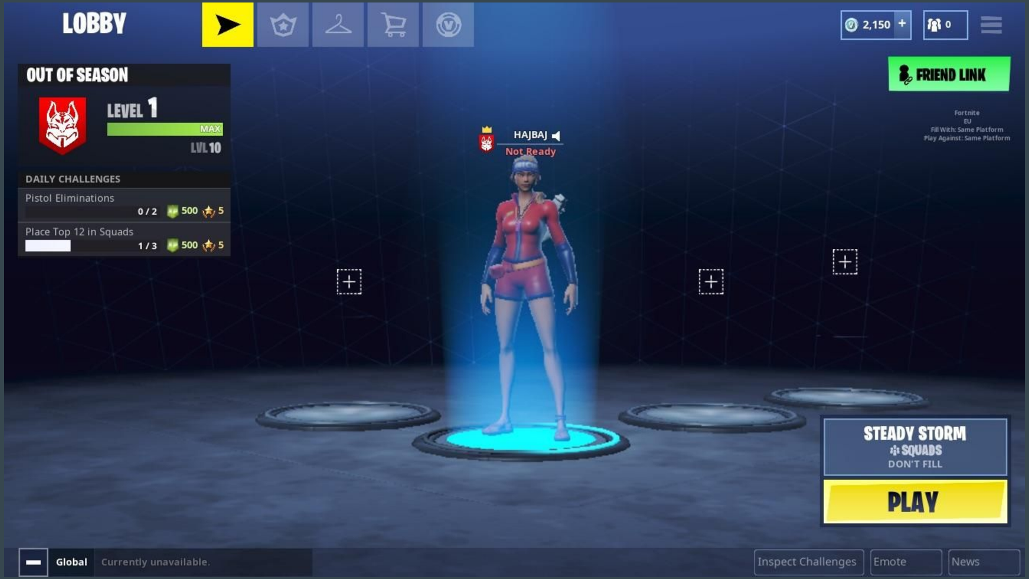
Brief Introduction

The player begins at home base with an introduction to get started.

- Home base is where you can upgrade abilities and regroup to prepare for the next level of the computer science battlefield.

Once you are ready, you are transported to the battlefield where you must defend yourself from computer science concepts using weapons you’ve obtained from computer science professors of the University of Idaho.

Storyboard



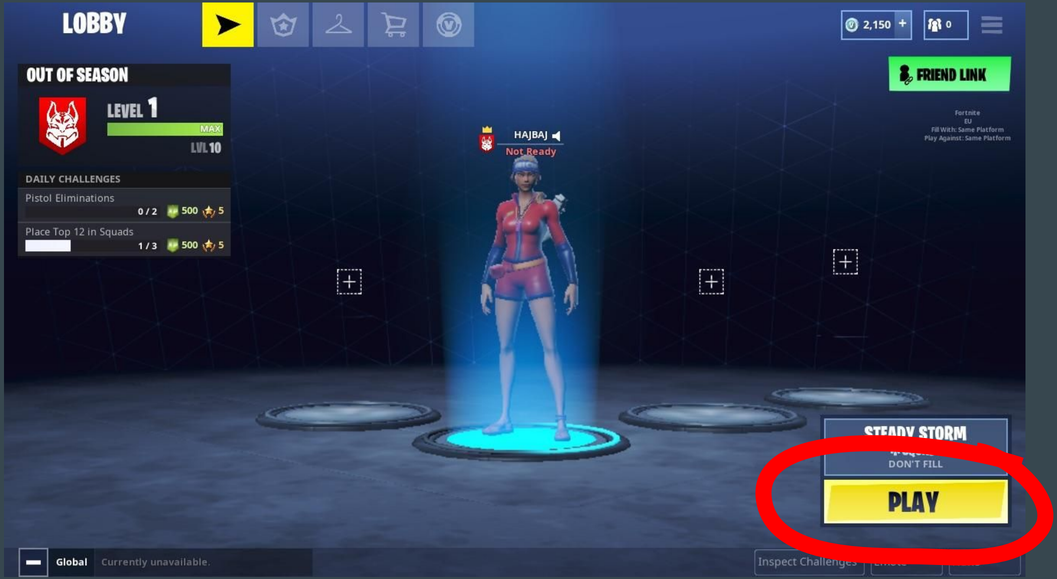
Fortnite
EU
Fill With: Same Platform
Play Against: Same Platform

Notes: Lobby screen example

Action: Player starts the game and begins in the lobby.

Dialogue: This is the lobby. You can see your level, alter your skill tree, and click 'play' to get into a match.

Storyboard



Notes: Lobby screen example

Action: You click 'play' to get into your first match.

Storyboard

PATREON: SACMAPS

Notes: Battlefield example

Dialogue: Fight waves of computer science concepts to complete the stage. Buy different weapons from the CS teachers to ward off these monsters.

Storyboard

Notes: Battlefield example

Action: You defend yourself against waves of enemies by using the weapons and utilities you start with.

Action: The enemies you kill drop gold for the shops and exp for the skill tree.

Storyboard

What do you need?
PATREON: S

Notes: Battlefield example

Action: You interact with the shop vendors (CS teachers).

Dialogue: (Bolden) What do you need?

Action: You purchase weapons, weapon upgrades, utilities, etc. from the vendor to get through the waves of enemies.

Storyboard

YOU DIED
PATREON: SACMAPS

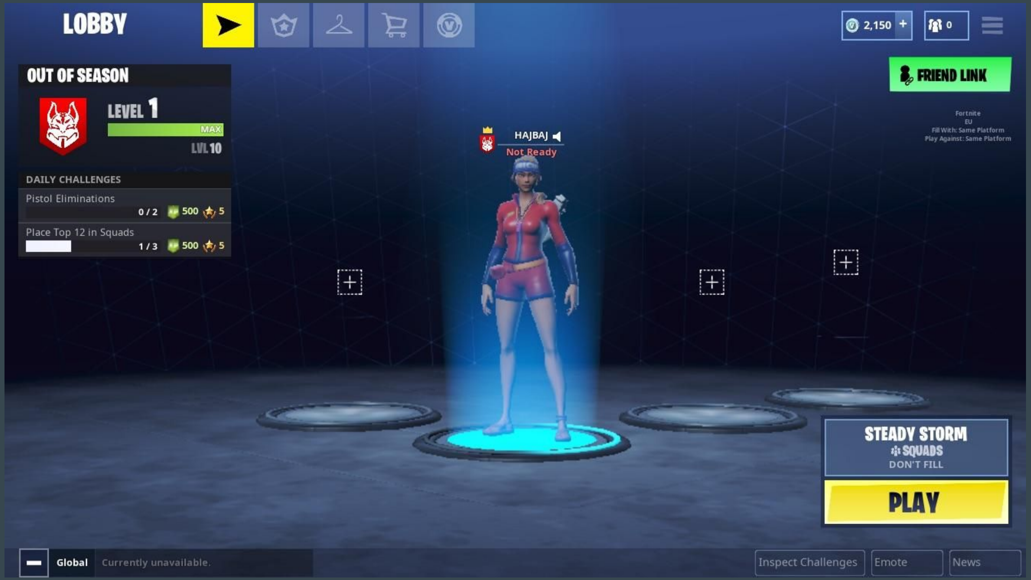
Notes: Battlefield example

Notes: The new weapon seen was obtained by visiting the shop vendors.

Action: You get overwhelmed with the waves of enemies and die.

Action: You are returned to the lobby.

Storyboard



Notes: Lobby screen example

Action: You open the skill tree and spend your skill points to upgrade abilities that make clearing waves easier.

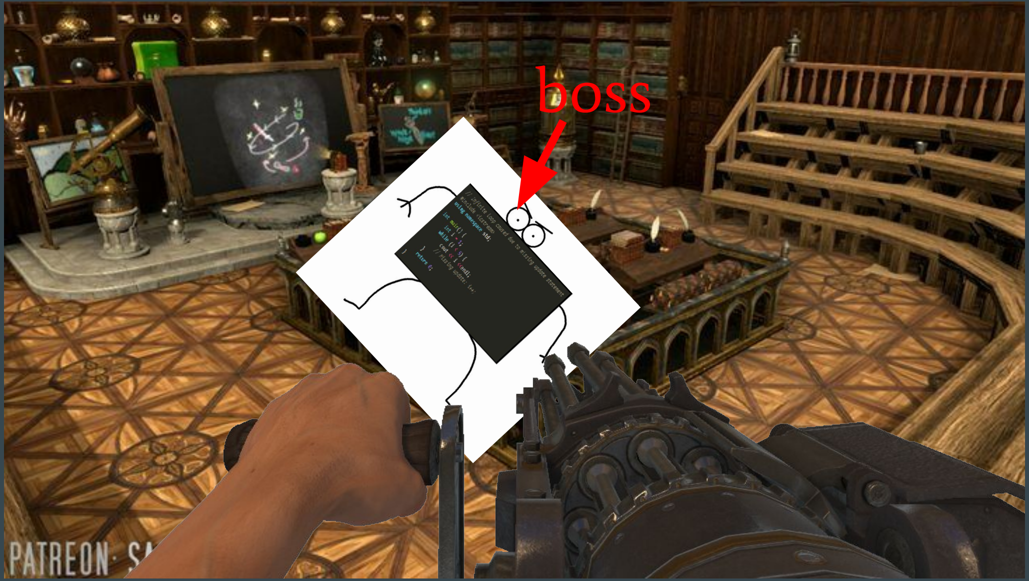
Action: You click 'play' to try again.

Storyboard

Notes: Battlefield example

Action: You defeat the waves of enemies, purchase weapons from the shops, defeat more waves, etc.

Storyboard



Notes: Battlefield example

Action: The final boss spawns.

Action: You kill the boss.

Storyboard

Dialogue: You’ve defeated the final boss! Congrats on getting through the computer science degree!

Context Diagram

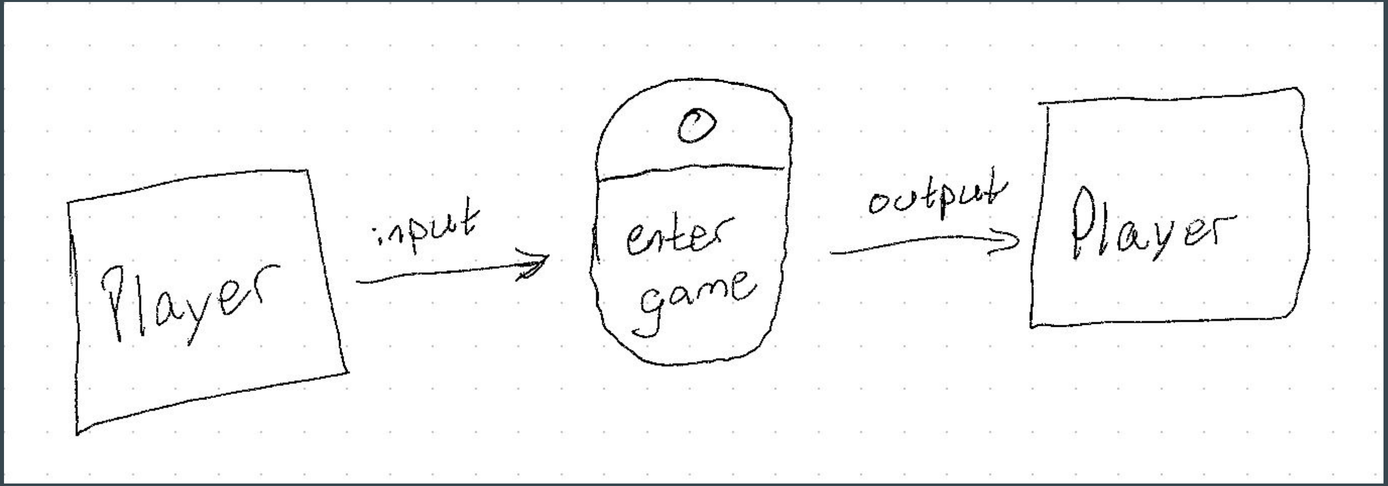
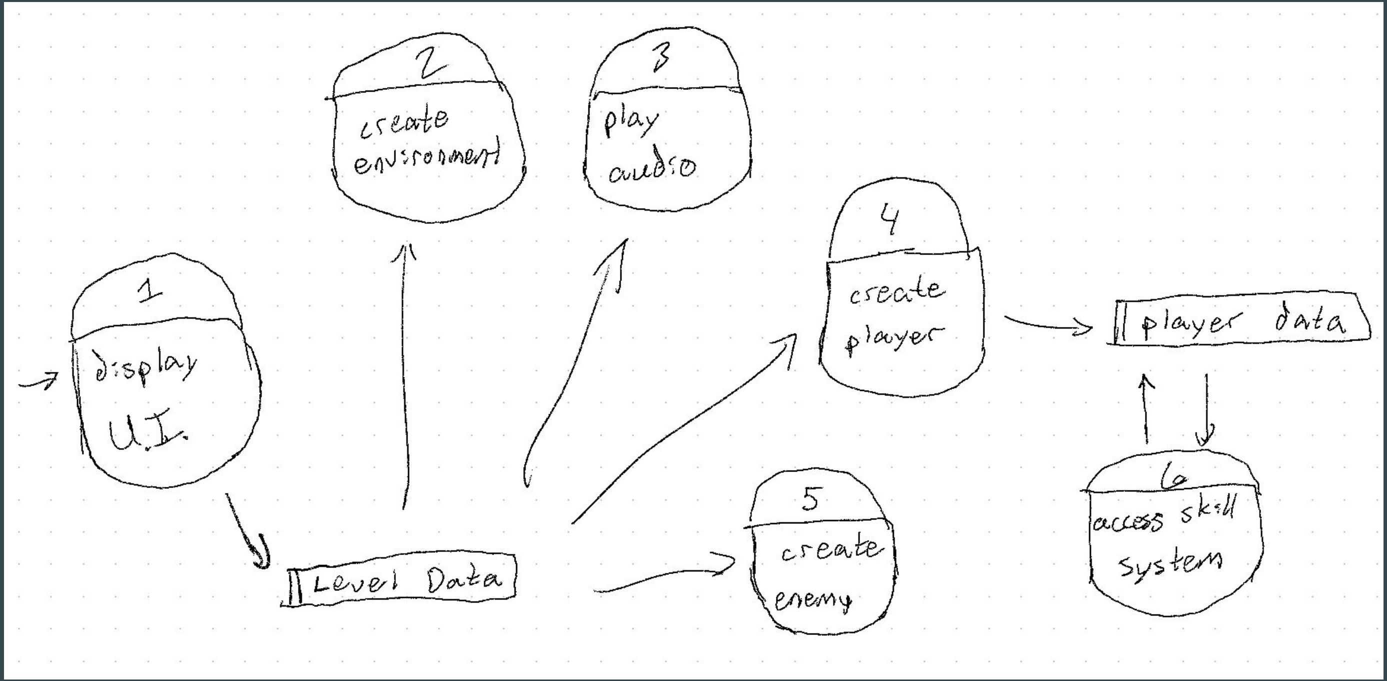
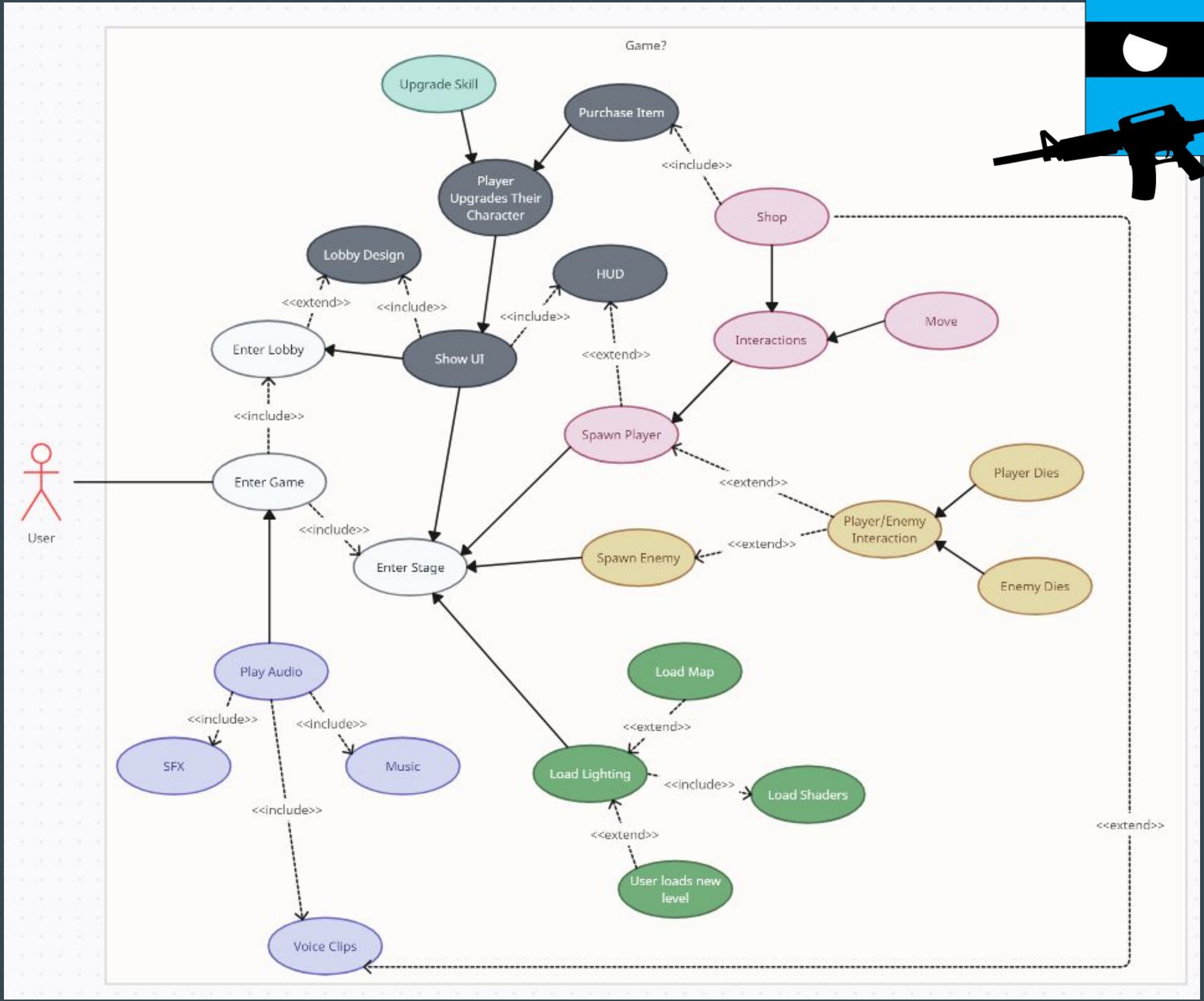


Diagram 0

1. Daniel
2. Sam
3. Ben
4. Devon
5. Zac
6. Ashton



Global Use Case



Skill System - Ashton Pomerantz

Scales the player so that you can compete with higher and harder stages of the game

The process:

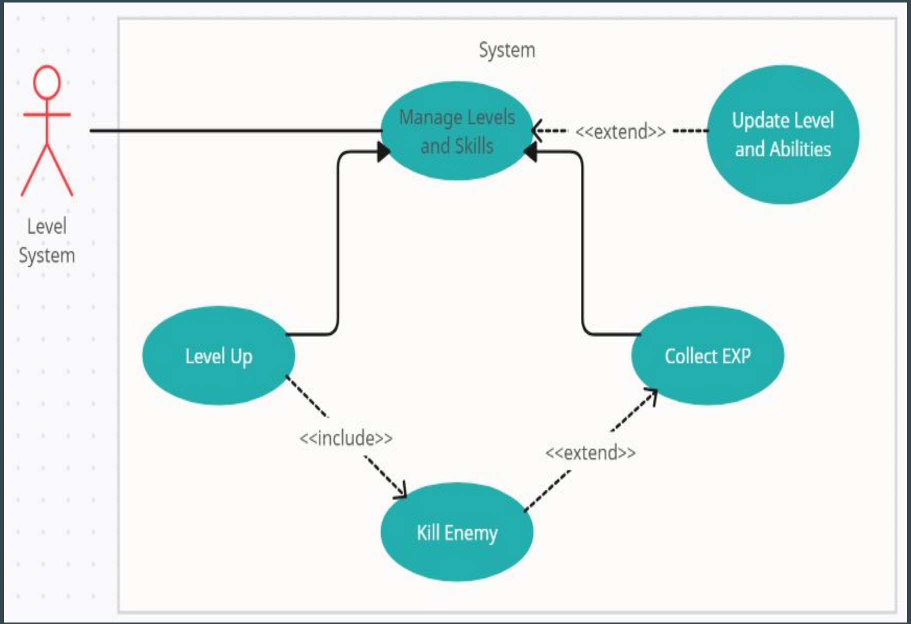
- Gain exp points by defeating enemies
- Exp levels up your character
- Gain skill points when you level up
- Use skill points to upgrade skills and abilities

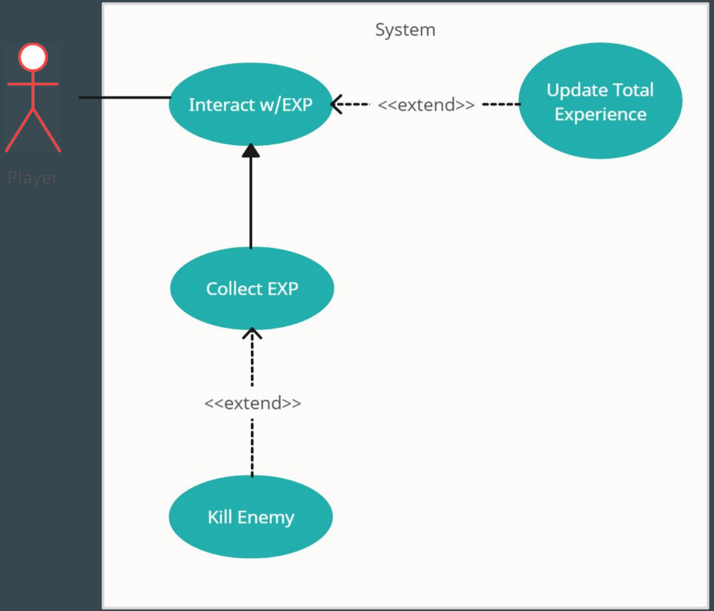
Similar to Diablo

Priority: 2 - Not necessary for main game loop, but required to make a more complete, balanced game

Skill System - Ashton Pomerantz

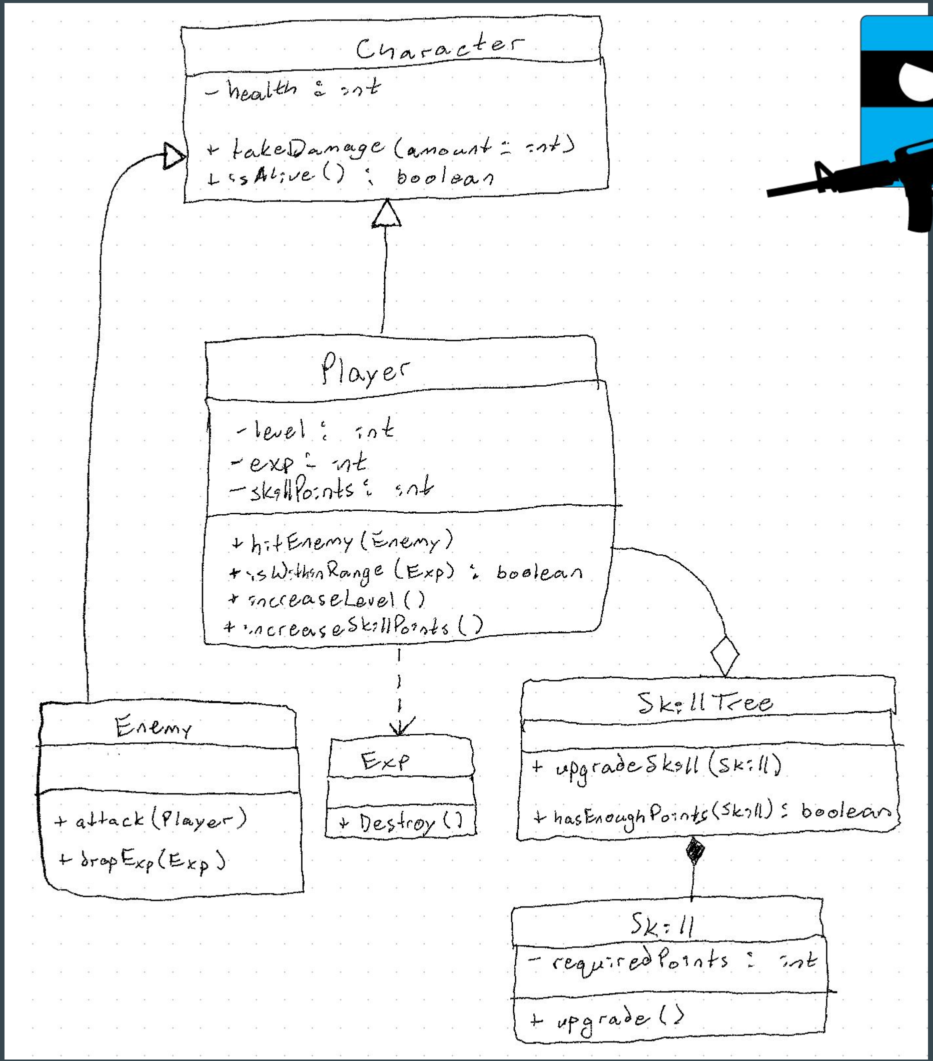
Case diagrams for skill system as well as exp interaction:





Skill System - Ashton Pomerantz

Class Diagram!



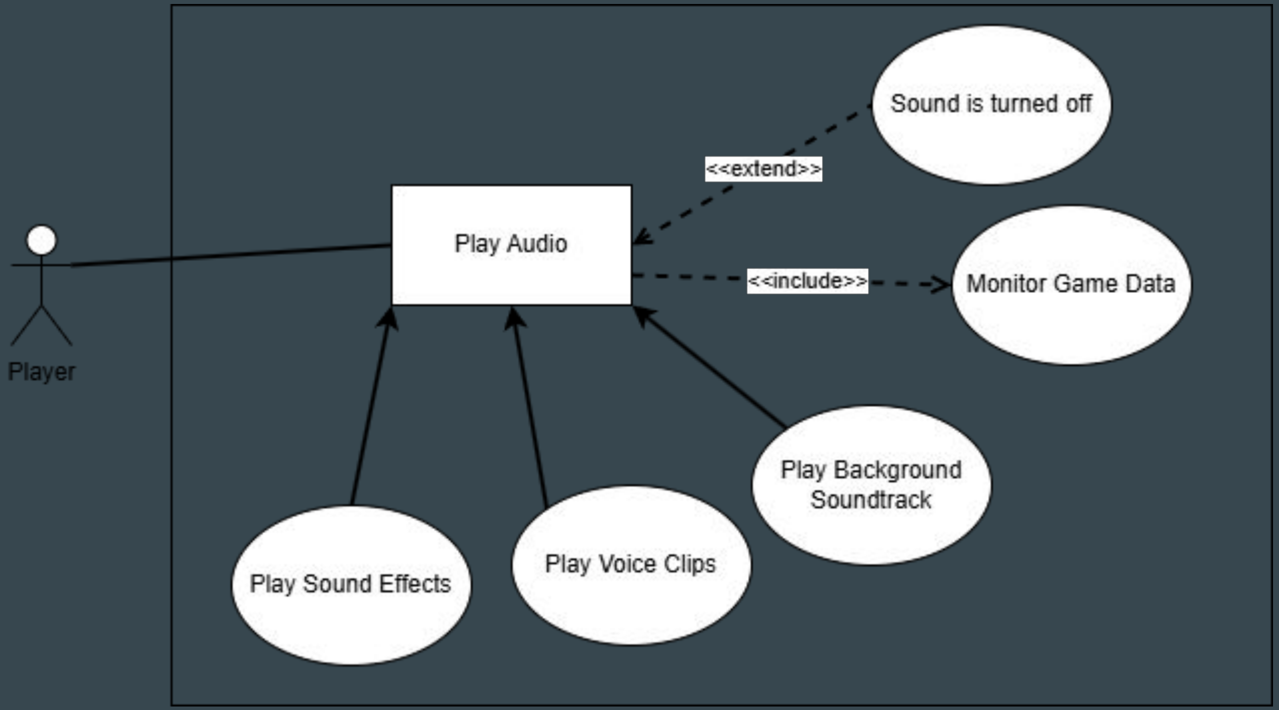
Audio System - Ben Randolph

The audio system manages the start and stop of playing audio including voice clips, sound effects, and background music.

Priority of 2 - not necessary to play the game, but sound effects can contain vital information the player needs to respond to enemies, and get context clues for various things happening that they may not be able to see.

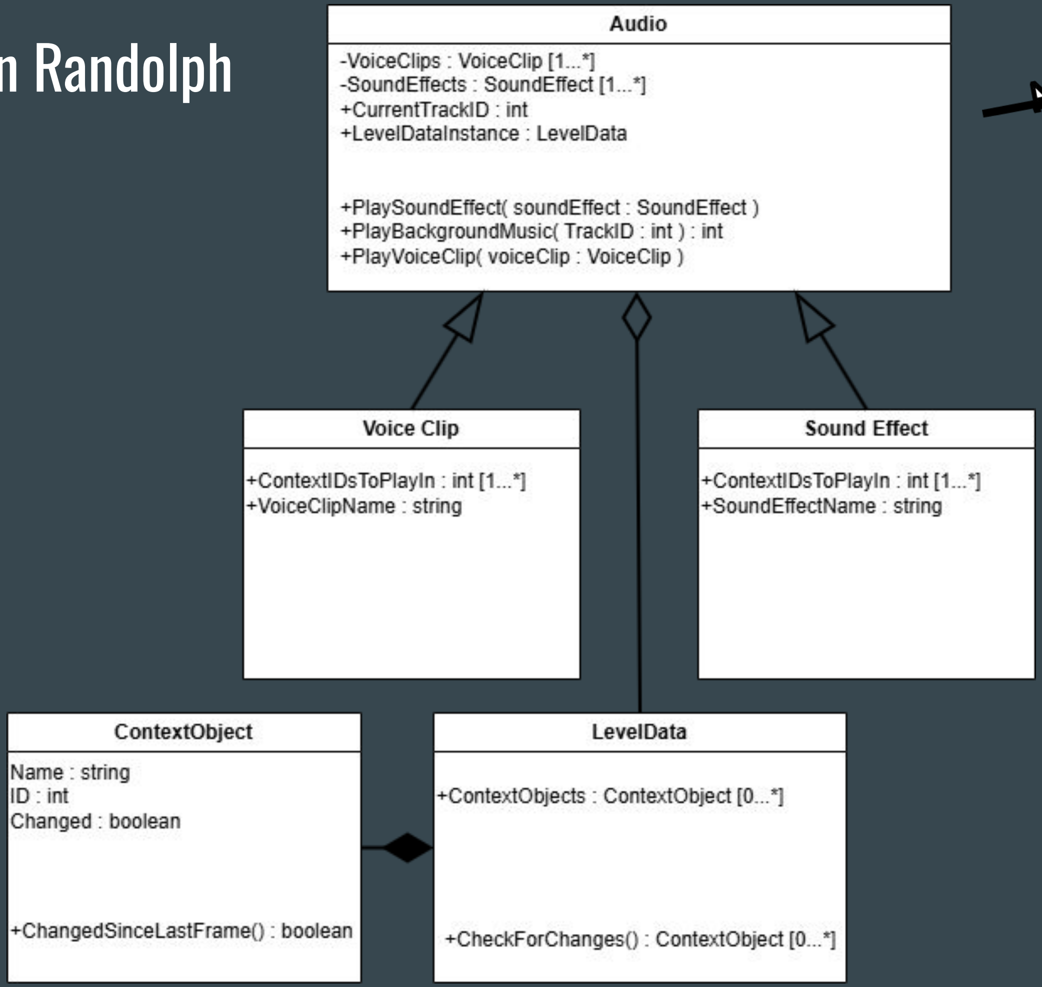
Audio System - Ben Randolph

Use case diagram for the audio system:

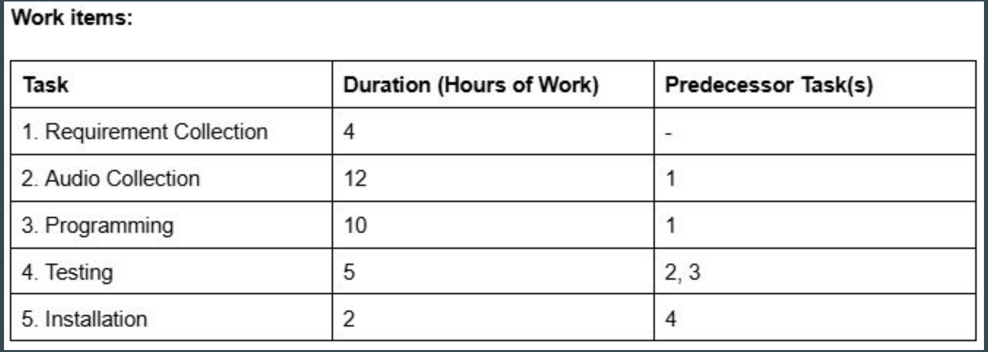


Audio System - Ben Randolph

Class diagram:

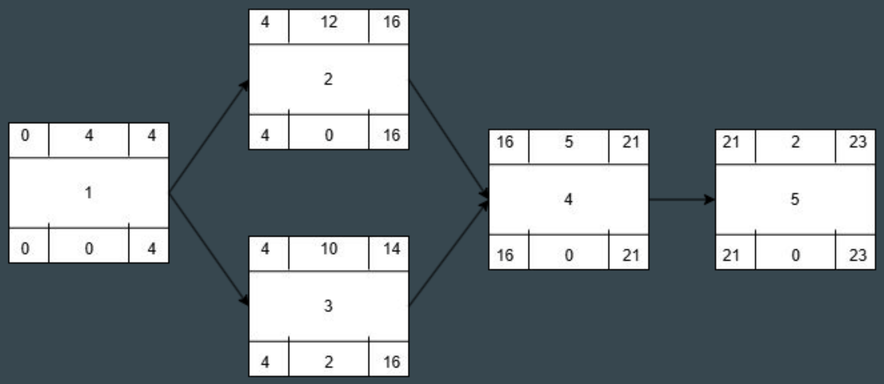

Audio
-VoiceClips : VoiceClip [1...*]
-SoundEffects : SoundEffect [1...*]
+CurrentTrackID : int
+LevelDataInstance : LevelData
+PlaySoundEffect(soundEffect : SoundEffect)
+PlayBackgroundMusic(TrackID : int) : int
+PlayVoiceClip(voiceClip : VoiceClip)
Voice Clip
+ContextIDsToPlayIn : int [1...*]
+VoiceClipName : string
Sound Effect
+ContextIDsToPlayIn : int [1...*]
+SoundEffectName : string
ContextObject
Name : string
ID : int
Changed : boolean
+ChangedSinceLastFrame() : boolean
LevelData
+ContextObjects : ContextObject [0...*]
+CheckForChanges() : ContextObject [0...*]

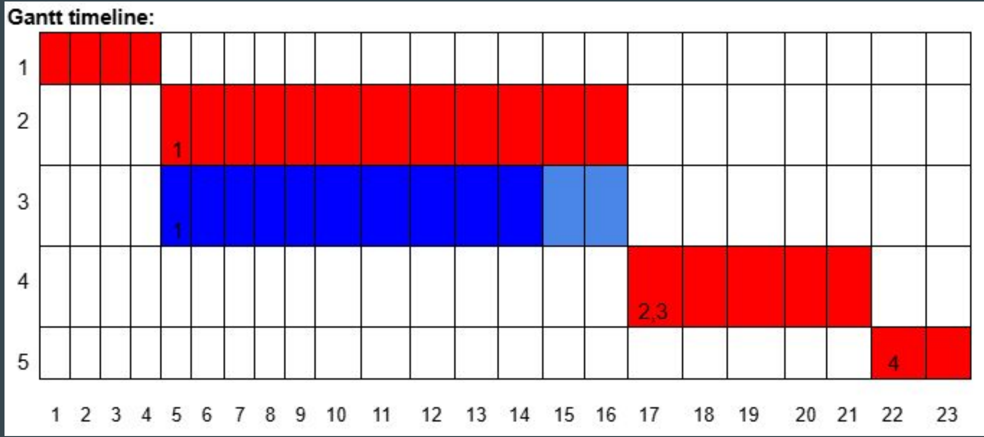
Audio System - Ben Randolph



Work items:

Task	Duration (Hours of Work)	Predecessor Task(s)
1. Requirement Collection	4	-
2. Audio Collection	12	1
3. Programming	10	1
4. Testing	5	2, 3
5. Installation	2	4





Samuel Beal: Environment

World map with lighting and shaders.

The player loads a level, then we load the map and send visuals back to the player.

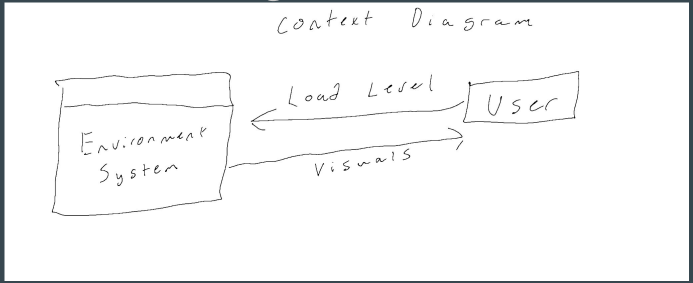
-Priority - 1 for World Map. Absolutely necessary to have a world to play in.

-Priority - 2 for Lighting. Makes game more playable and not in darkness/solid colors.

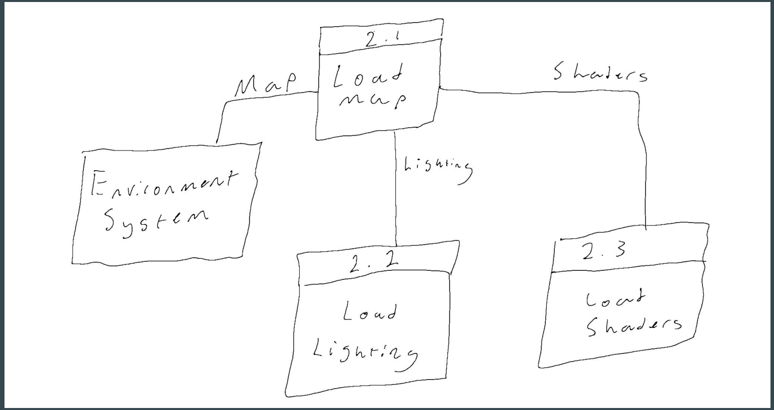
-Priority - 3 for Shaders. Not necessary but makes game look much cleaner.

Samuel Beal: Environment

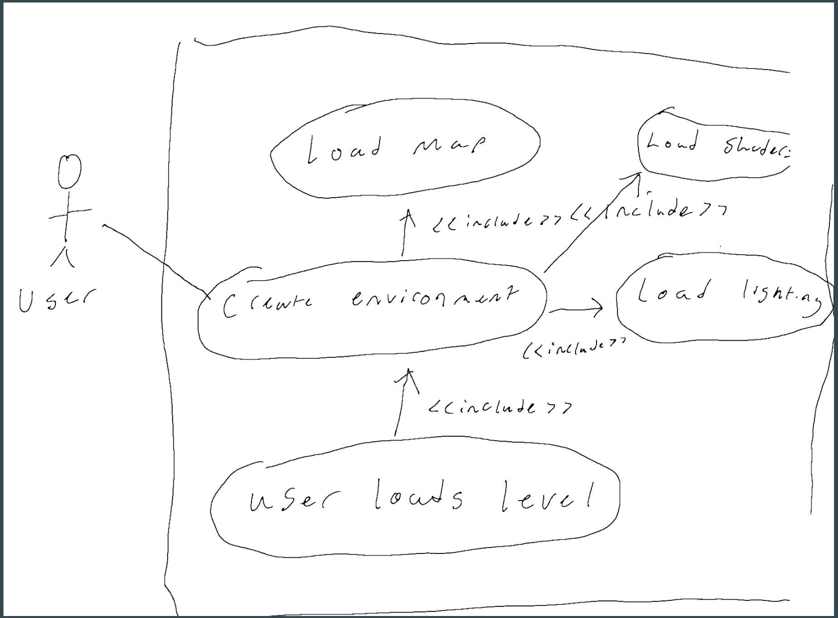
1) Context Diagram



2) Diagram 1

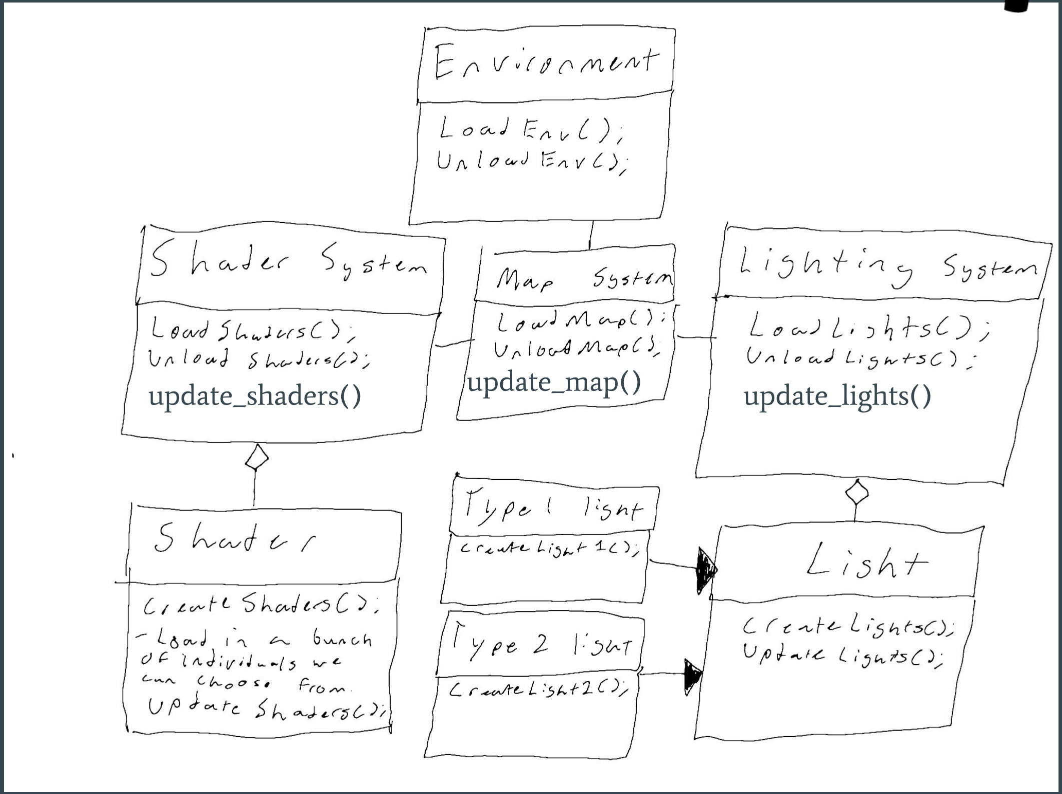


3) Use cases



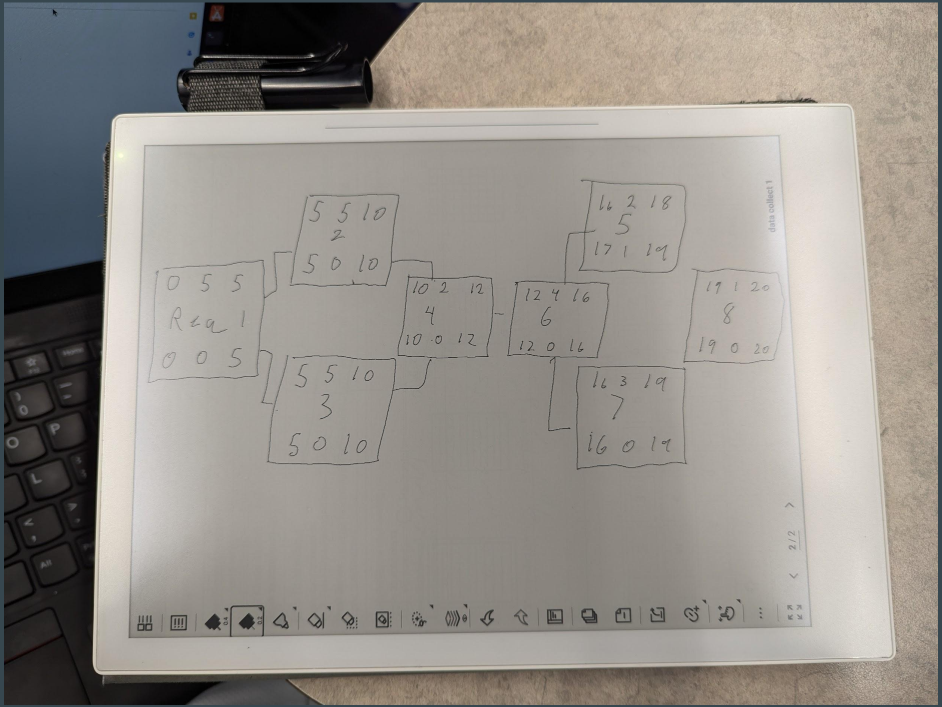
Samuel Beal: Environment (cont.)

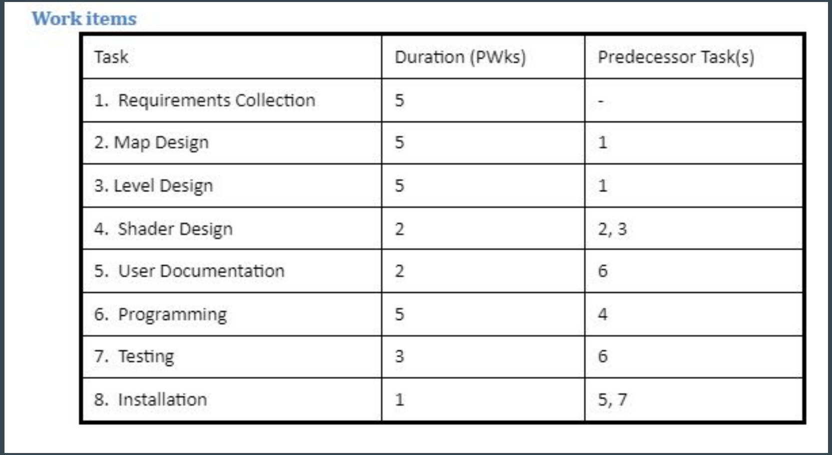
4) Class Diagram



Samuel Beal: Environment (cont.)

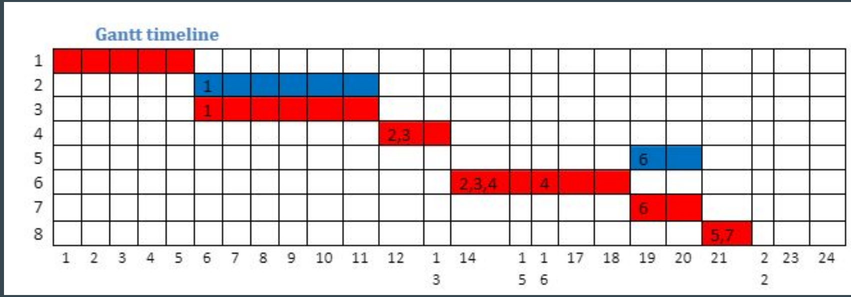
Pert Diagram





Work items

Task	Duration (PWks)	Predecessor Task(s)
1. Requirements Collection	5	-
2. Map Design	5	1
3. Level Design	5	1
4. Shader Design	2	2, 3
5. User Documentation	2	6
6. Programming	5	4
7. Testing	3	6
8. Installation	1	5, 7



Daniel Franks: UI

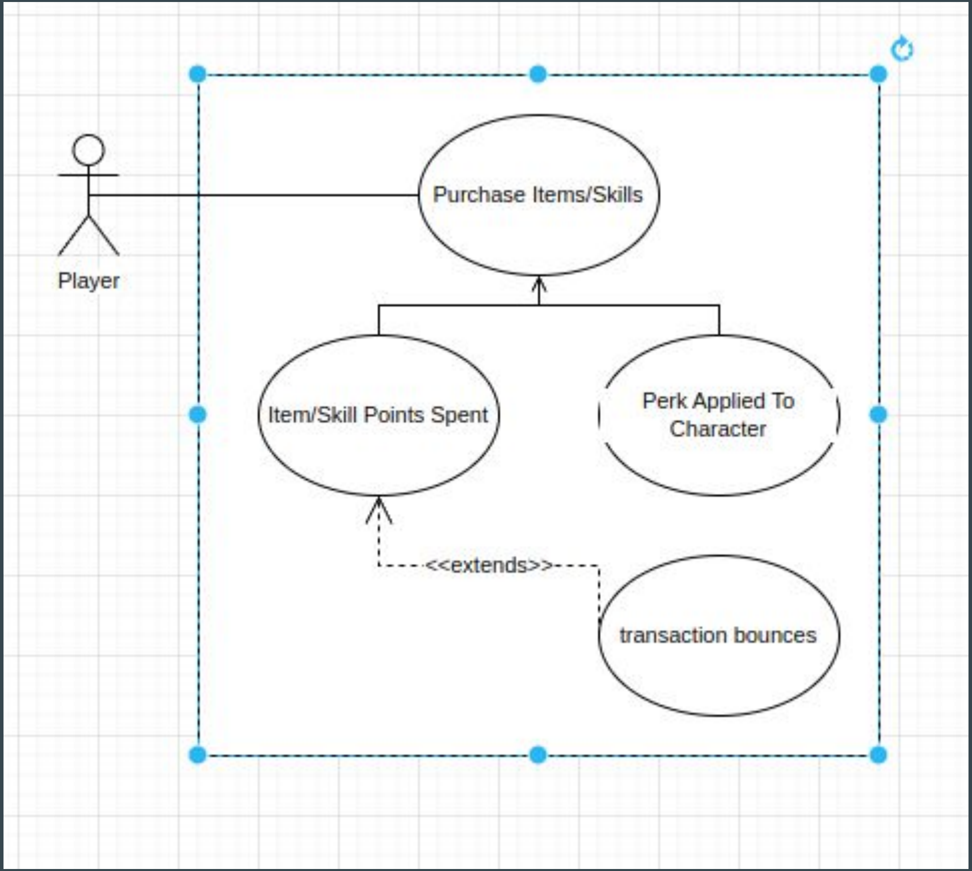
I’m creating the game UI, including:

- The skill tree menu
- The shop interfaces
- The pause menu
- Anything else we wind up with bandwidth for

Priority: 1
-> there are major parts of the game that don’t exist without a menu to get them

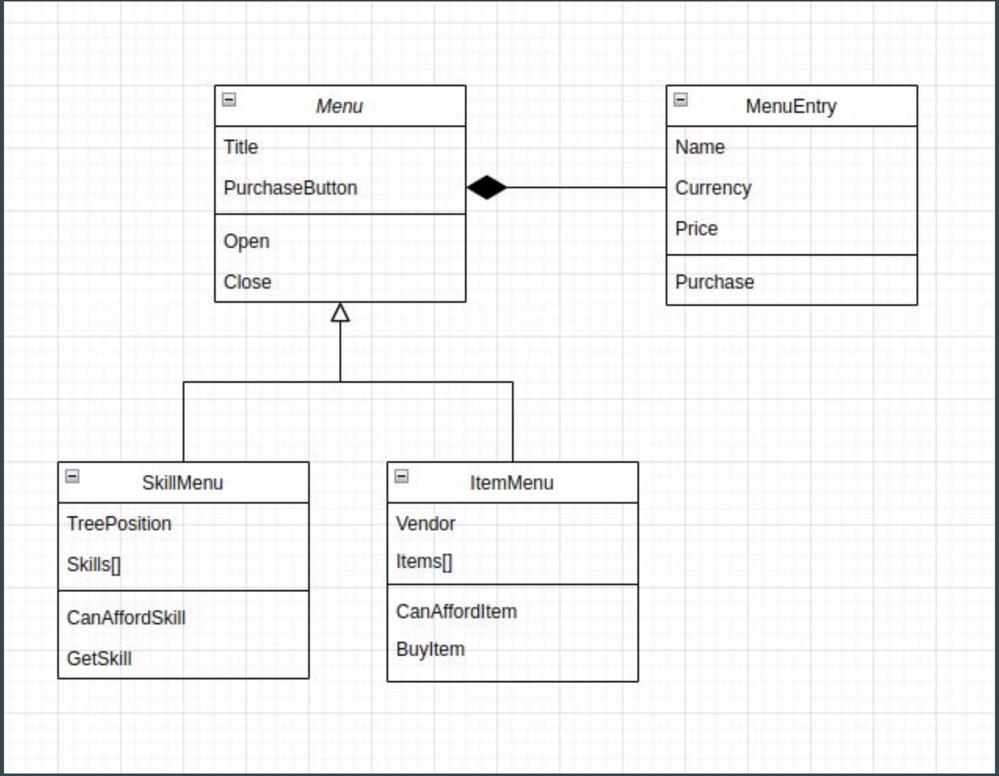
Daniel Franks: UI

Use case diagram:



Daniel Franks: UI

Class diagram:



Zac Squires: Enemies/AI

Items:

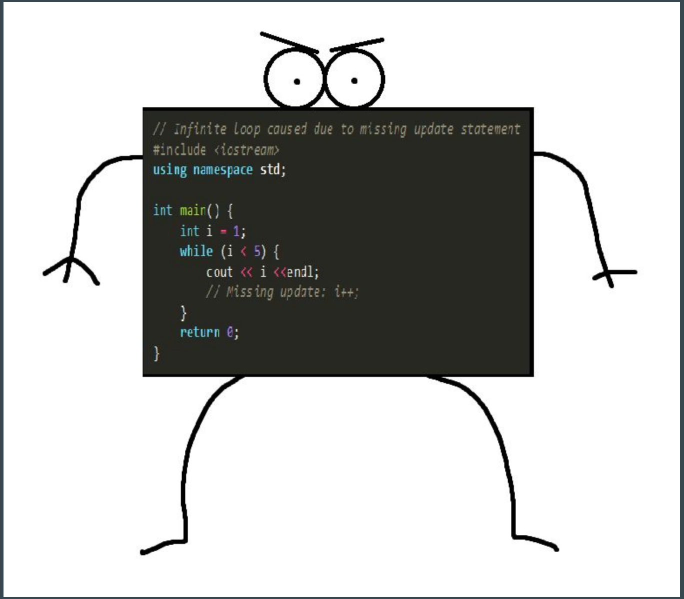
- Behavior
- Attributes
- Types?
- Manager
- Pathing

Priority:

- Significant

Estimated Complexity:

- Moderate

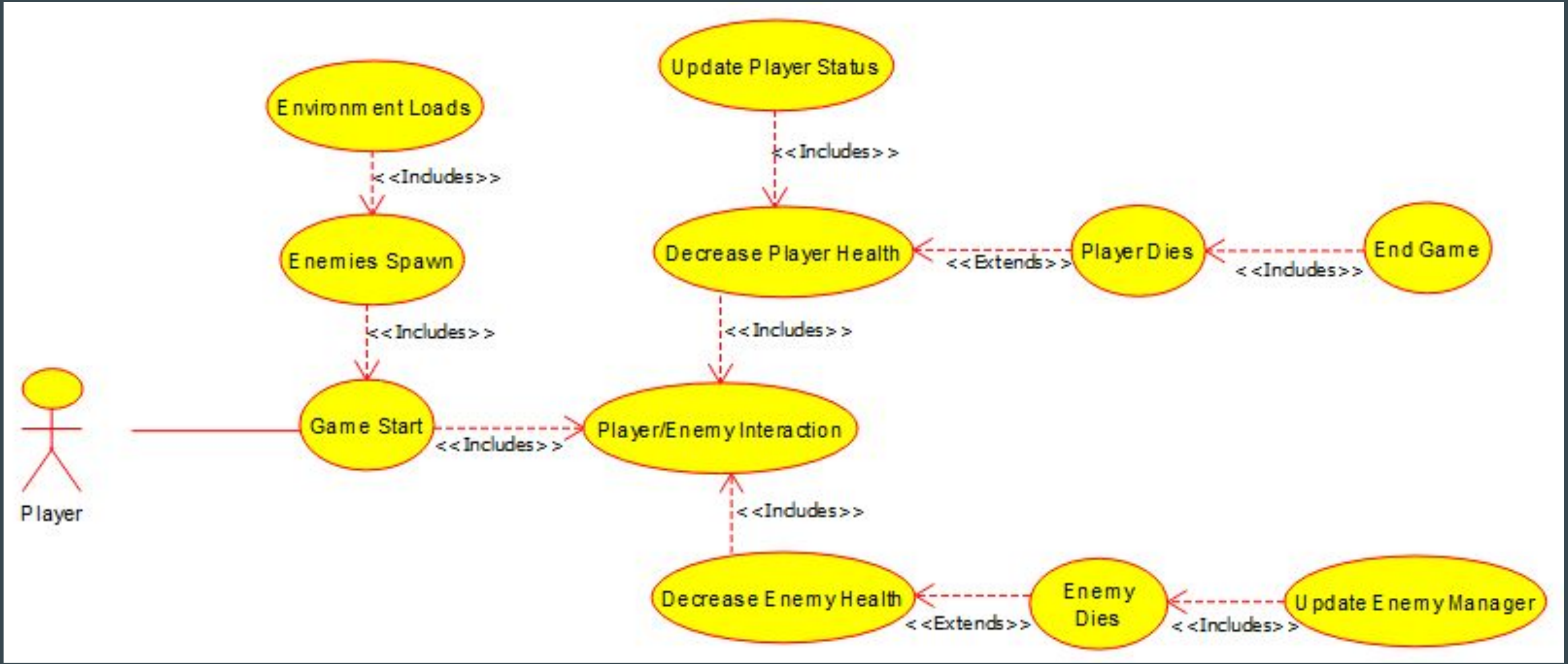


```
// Infinite loop caused due to missing update statement
#include <iostream>
using namespace std;

int main() {
    int i = 1;
    while (i < 5) {
        cout << i <<endl;
        // Missing update: i++;
    }
    return 0;
}
```

Concept Art

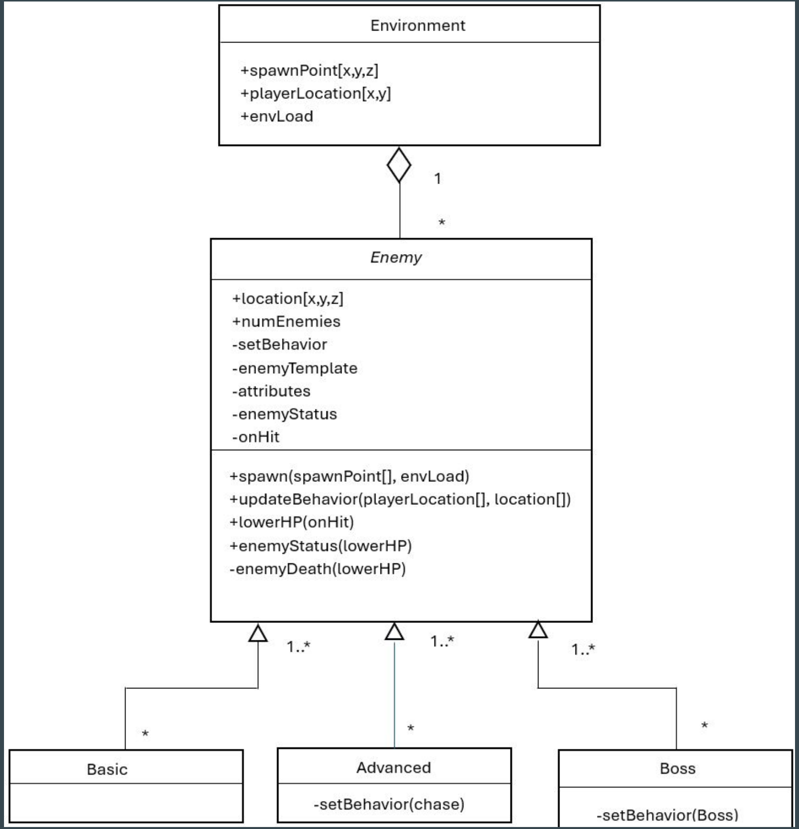
Zac Squires: Enemies/AI (cont.)

Enemy Use Case:



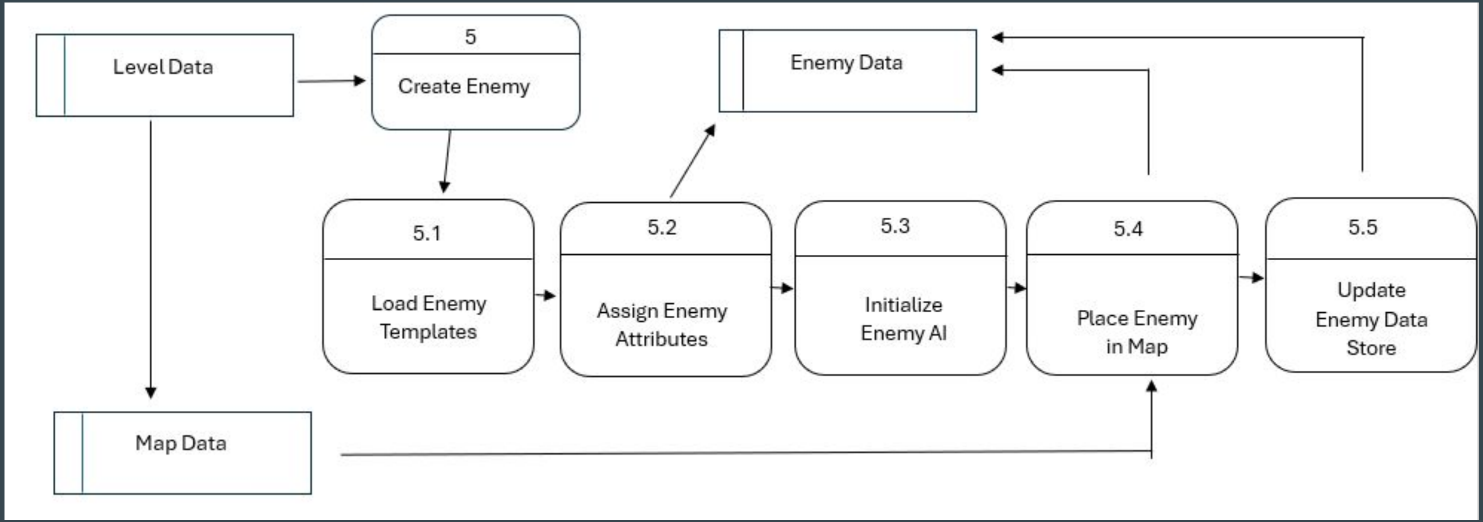
Zac Squires: Enemies/AI (cont.)

Enemy Class Diagram:



Zac Squires: Enemies/AI (cont.)

Diagram 1:



Devon Bryce: Movement

Level 0

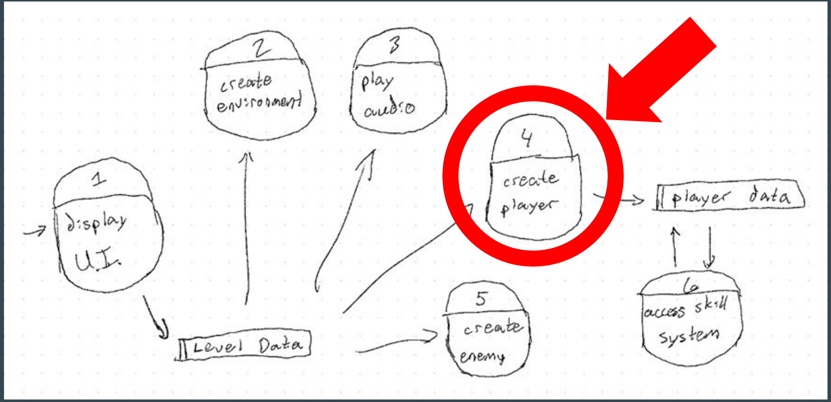
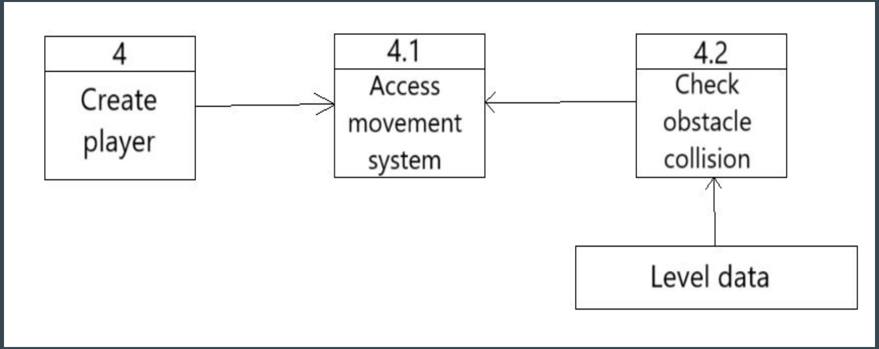
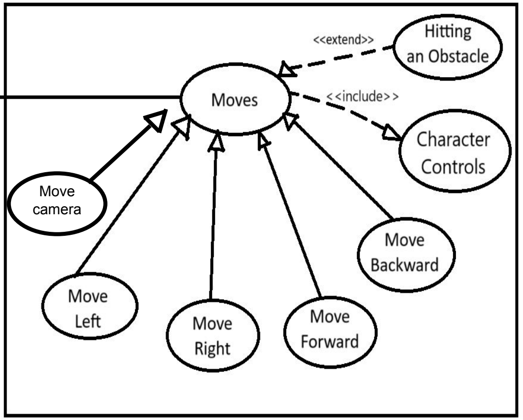


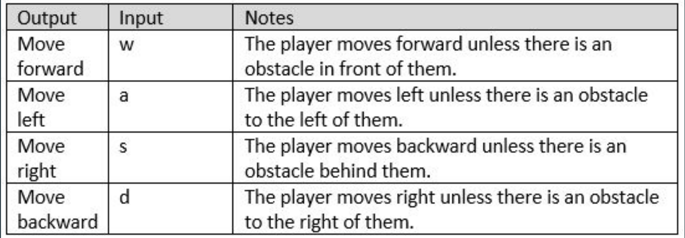
Diagram 1



Devon Bryce: Movement

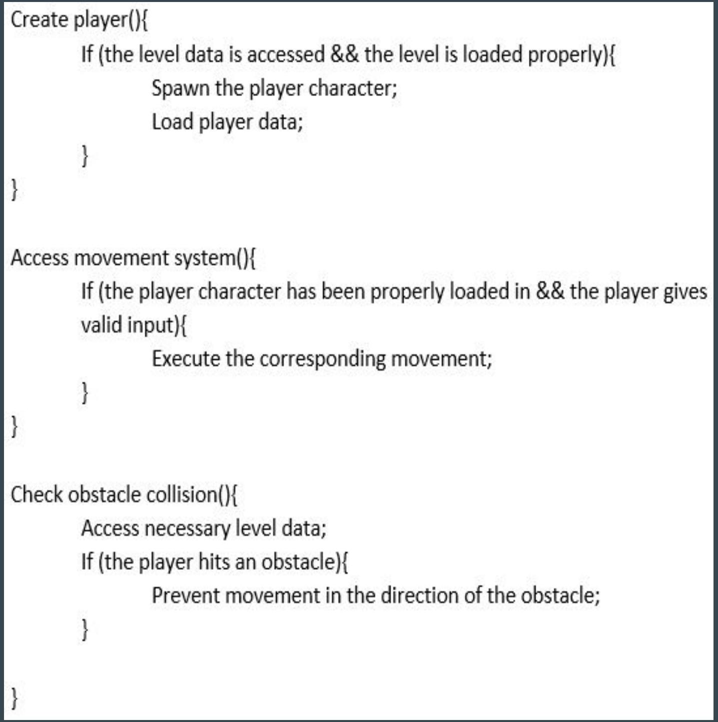
Uses cases





Output	Input	Notes
Move forward	w	The player moves forward unless there is an obstacle in front of them.
Move left	a	The player moves left unless there is an obstacle to the left of them.
Move right	s	The player moves backward unless there is an obstacle behind them.
Move backward	d	The player moves right unless there is an obstacle to the right of them.
Move camera	Mouse	The camera tracks the movement of the mouse so that the player can look around.

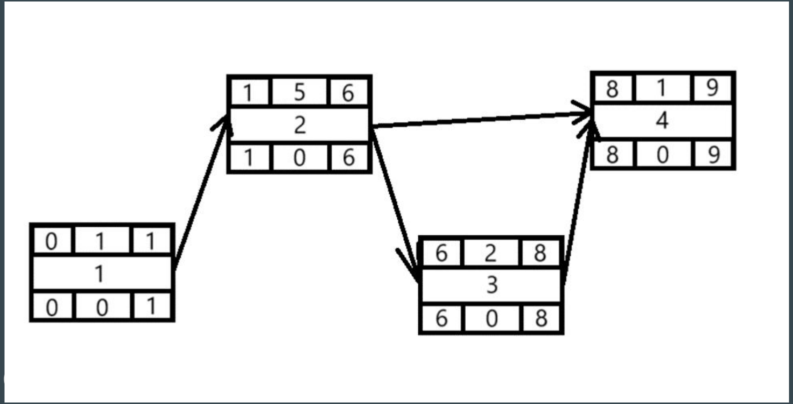
Process descriptions



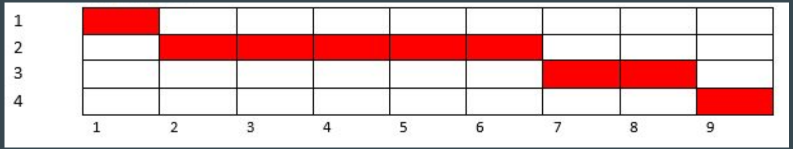
```
Create player(){
        If (the level data is accessed && the level is loaded properly){
                Spawn the player character;
                Load player data;
        }
}

Access movement system(){
        If (the player character has been properly loaded in && the player gives
        valid input){
                Execute the corresponding movement;
        }
}

Check obstacle collision(){
        Access necessary level data;
        If (the player hits an obstacle){
                Prevent movement in the direction of the obstacle;
        }

}
```

Devon Bryce: Movement

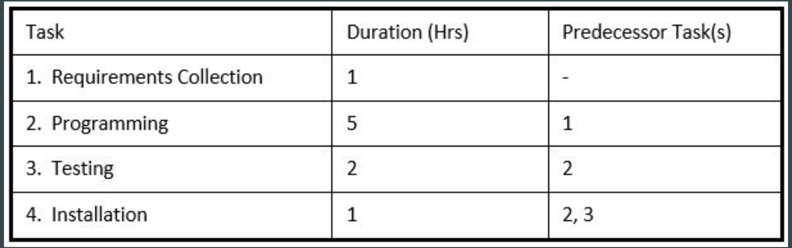
Pert diagram:



Gantt timeline:



Work items:



Task	Duration (Hrs)	Predecessor Task(s)
1. Requirements Collection	1	-
2. Programming	5	1
3. Testing	2	2
4. Installation	1	2, 3

Priority: Must have - If the player can't move or face towards enemies, they can't defeat them and progress the game (assuming skill/abilities don't help bypass this problem).

Questions?