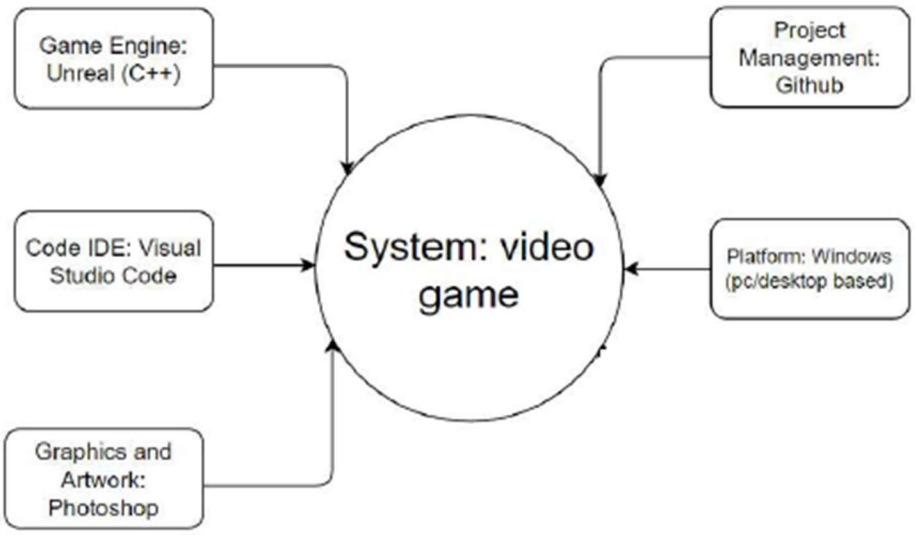
Software Requirements Specification

2/c Fulton, 2/c Taha

Project Description/Proposal

System Scope:



Stakeholders:

Cadets: Those who have access to the game launcher will be able to play the game. Their role is to play and enjoy the videogame. In addition, they will provide feedback on the playability, functionality, and overall game design so that we as developers can make improvements.

Instructors: The role of the instructor is to evaluate the final product and develop a quantitative evaluation based on the specified criteria. The instructor will also provide feedback on how to improve in the future.

Process Description:

Once the game is developed, users will have a fully built level-based game that utilizes strategy, logic and problem solving to discover clues and unlock the next level. The method to deliver this game will be through an elevator system which brings the user to each floor (level) as they are completed. The goal is for the user to unlock all the doors and make it to the final floor.

User Requirements

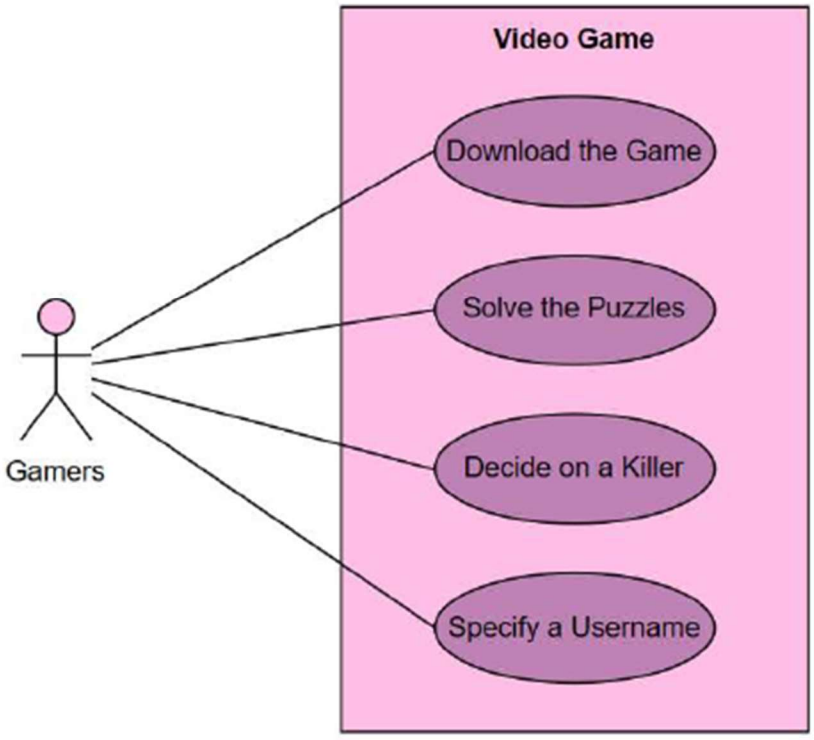
1. The system shall have a user-friendly interface and that is easy to navigate.
2. The system should have adequate graphics and sound effects that enhance the player's experience.
3. The system shall provide clear instructions on how to play and what the player's objectives are.
4. The system should be accessible to players of all ages and skill levels.
5. The system shall have clear instructions and optional hints on how to solve puzzles, when needed.
6. The user should be able to save and return to a level.

System Requirements

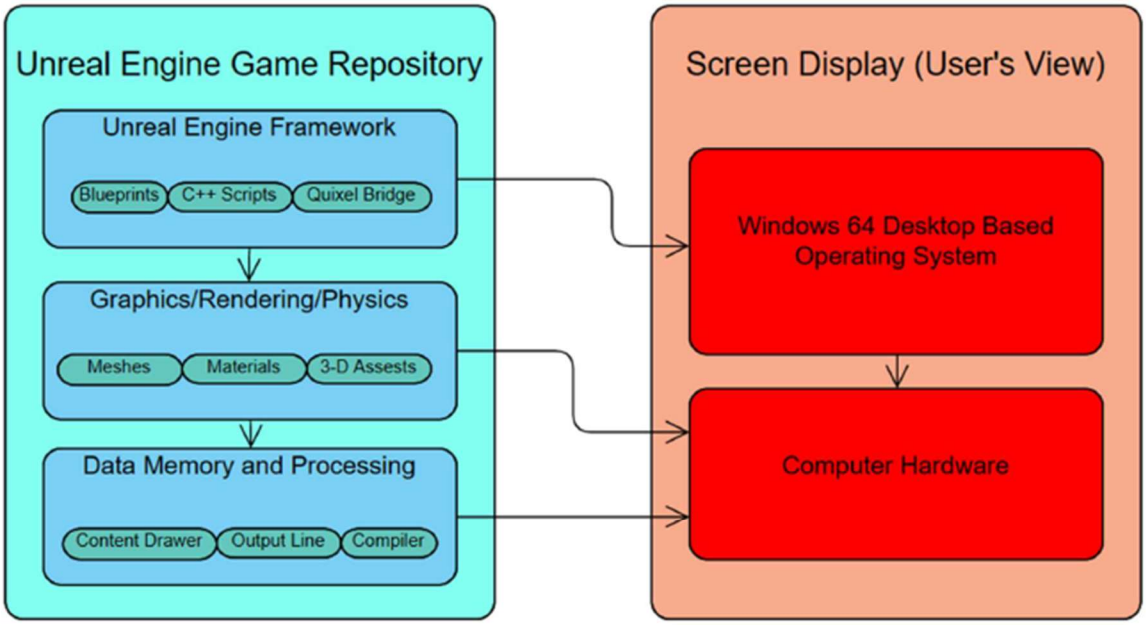
1. The system shall be compatible with Windows operating systems.
2. The system should not require an internet connection to play.
3. The system should be executable from a desktop device.
4. The system code shall integrate both C++ and Blueprint.
5. The system should be designed to minimize graphics usage/load.

Use Cases

1. Users download the game.
2. Users make a username.
3. Users solve the puzzles.
4. Users decide on a killer.



Architecture Diagram



Unreal Engine Game Repository: The repository is hosted by Unreal Engine and is broken down into three sections. The first is the framework of the engine. The repository is supplied with blueprints, C++ (VS Code) scripts, and the Quixel Bridge. Blueprints and C++ code are the scripting systems integrated in the Unreal Engine that are used to develop programming assets that can be stored in the repository and utilized for the game itself. The Quixel Bridge is an engine plug-in that supplies the 3-D assets, Materials, and other objects used for the visual design aspect of the videogame. The next section is the graphics and physics logic. The repository will store actors and pawns (game assets) acquired from the Quixel Bridge into a content drawer to be used by the game. These assets consist of meshes, sprites, and even complete maps (game levels). The last section is the data memory and processing. The Unreal Engine Repository uses an output line, compiler, and content drawer to store and process the scripts/assets used to program the videogame.

Screen Display (User's View): The screen monitor, which will display the videogame, is broken down into its operating system and computer hardware. The operating system for this game will be Windows OS running on a 64-bit system. The host for this operating system is a desktop computer or PC. The computer hardware consists of the GPU, CPU, RAM, monitor system, and many other working parts. The computer hardware has a direct impact on the level of memory and processing that will affect the display of the videogame itself.

The Unreal Engine Framework is associated with the operating system as the repository is designed/generated specifically for a windows 64 system. The graphics, data memory, and processing systems are linked to the computer hardware which will directly impact the level of computing power used to run the videogame.

The system architecture used for our videogame is a blend of both layered and pipe-and-filter patterns. The layered architecture shows the hierarchy within both the repository and the user's display and how those components interact with one another. This approach also supports the incremental development of our videogame as certain parts of the overall system will be available as the software is developed. The pipe-and-filter architecture shows the flow of data from the game engine to the user's display. Components within the repository directly interact with the operating system and hardware of the computer to receive the input information and output a fully functioning videogame to the user's display.