Wentworth Institute Of Technology

College of Engineering & Technology
Department Of Computer Science & Networking
Software Design and Development – COMP 566

1. INTRODUCTION

Fructus Victus is a procedurally generated first person exploration game, written in Java using LibGDX. Fructus Victus makes use of existing ray-casting rendering techniques in order to emulate the visuals seen in Wolfenstein 3D and Doom. The goal of Fructus Victus is exploration and survival through a labyrinth. What you discover and accomplish within Fructus Victus is up to the user, and the aim is to make an experience that will be unique on each playthrough.

Upon pressing “New Game”, the user then finds themselves inside of a cavern with one instruction: "Find the source of the mutation!". On the wall in front of the user, instructions for movement and interaction are listed. As the user moves into subsequent rooms, avoiding traps and searching for clues, the situation becomes clearer and clearer. The world's fruit has begun to mutate, and its up to you to search for the answer at the source: inside the fruit itself!

1.1 Purpose of this Document

The purpose of creating this Software Requirements Specification (SRS) document is to describe key components and layout the functionalities of Fructus Victus. The four key components of Fructus Victus are the ray-casting game engine, map generator, input handler, and game menu. The ray-casting game engine will be the component responsible for calculating and rendering the pixels on the screen. The map generator will be responsible for creating randomized maps based on commonly used algorithms. The input handler is simply a control manager that takes in commands from the user and translates the commands to movements on the screen. The game menu will allow the user to begin or continue a game, as well as to adjust options.

1.2 Scope of the Development Project

Fructus Victus is a procedurally generated game which utilizes ray-casting to render the maze that users explore. One drawback to ray-casting is that the game must be played from the first-person perspective, and although the user appears to be in a 3D environment, the world itself is entirely flat. However, this drawback can also be seen as an advantage; by having a flat world, random levels can be generated inexpensively. Another advantage is the speed of ray-casting, which will allow our game to be played by users with a range of system specs, on a wide range of devices.

1.3 Definitions, Acronyms, and Abbreviations

LibGDX - A Java game library.

Ray-casting - A rendering technique which performs mathematical computations whenever the user's origin coordinates or view direction changes. For each vertical line displayed on the screen, a ray is casted from the users position until it collides with a wall. Once the collision has occurred, basic trigonometry is performed to determine the height of the wall.

1.4 References

Chollak, Devin. "Libgdx/libgdx." GitHub. N.p., n.d. Web. 25 Jan. 2015.

Loftis, Hunter. "PlayfulJS." Ray Casting Engine. N.p., n.d. Web. 25 Jan. 2015.

Permadi, F. "Ray-Casting Tutorial For Game Development And Other Purposes." Ray Casting Programming Tutorial For Game Development. N.p., n.d. Web. 25 Jan. 2015.

"Procedural Content Generation Wiki." Map Generation -. N.p., n.d. Web. 25 Jan. 2015.

1.5 Overview of Document

2. General Description
    1. User Characteristics
    2. Product Perspective
    3. Overview of Functional Requirements
    4. Overview of Data Requirements
    5. General Constraints, Assumptions, Dependencies, Guidelines
    6. User view of Product Use
3. Specific Requirements
    1. External Interface Requirements
    2. Detailed Description of Functional Requirements
        1. Template for describing functional requirements
        2. Through 3.2.x - The description of each functional requirement.
    3. Performance Requirements
    4. Quality Attributes
4. Other Requirements
5. Charts/Diagrams

2. GENERAL DESCRIPTION

Fructus Victus is a useful project since it brings a level of nostalgia to the user base. There have been few games recently that use ray-casting so there is untapped potential for a popular new game. By creating unique textures, and by using procedural levels, each world will look unique on every playthrough

2.1 User Characteristics

This game currently only requires the use of the keyboard and mouse for input. Since ray-casting is an extremely inexpensive rendering technique, Fructus Victus should be playable on a wide range of hardware.

2.2 Product Perspective

Fructus Victus is a standalone product, with little to no competitors. Since ray-casting is an outdated technique which has fallen out of favor now that 3D modelling has taken off, there are very few games being developed that still use it. Of these games, many are never completed, or are simply created as cheap prototypes. Fructus Victus will be the first game which is both ray-casted and utilizes procedural world generation.

2.3 Overview of Functional Requirements

N/A

2.4 Overview of Data Requirements

The game will log the users progress using states, and can be paused and exited at any time without losing any data. Map files will have potential to be saved to the disk as well.

2.5 General Constraints, Assumptions, Dependencies, Guidelines

Since Fructus Victus is written in Java, it will be playable on all devices, such as Windows, Mac, Linux, Android, iOS, or over the Web. Minor changes would be required for an iOS or Android version of Fructus Victus.

2.6 User View of Product Use

The users first interaction with the game after launching it is a view of the menu screen. This menu will contain options to start a new game, continue the previous game, change the game settings, or quit the game. If the user decides to start a new game they are placed in a random, procedurally generated world in which they are free to move around using the keyboard to unravel parts of the story. If they pause the game they will have the option to save their position in the game or to quit from this menu which brings them back to their desktop. If the user decides to continue the previous game, they will continue from wherever they made the most recent save. The settings menu allows the user to change system settings such as key bindings and screen resolution. Quitting the game brings the user back to their desktop.

3. SPECIFIC REQUIREMENTS

3.1 External Interface Requirements

N/A

3.2 Detailed Description of Functional Requirements

N/A

3.2.1 Template for describing functional requirements

N/A

3.2.2 through 3.2.x - The description of each functional requirement.

N/A

3.3 Performance Requirements

N/A

3.4 Quality Attributes

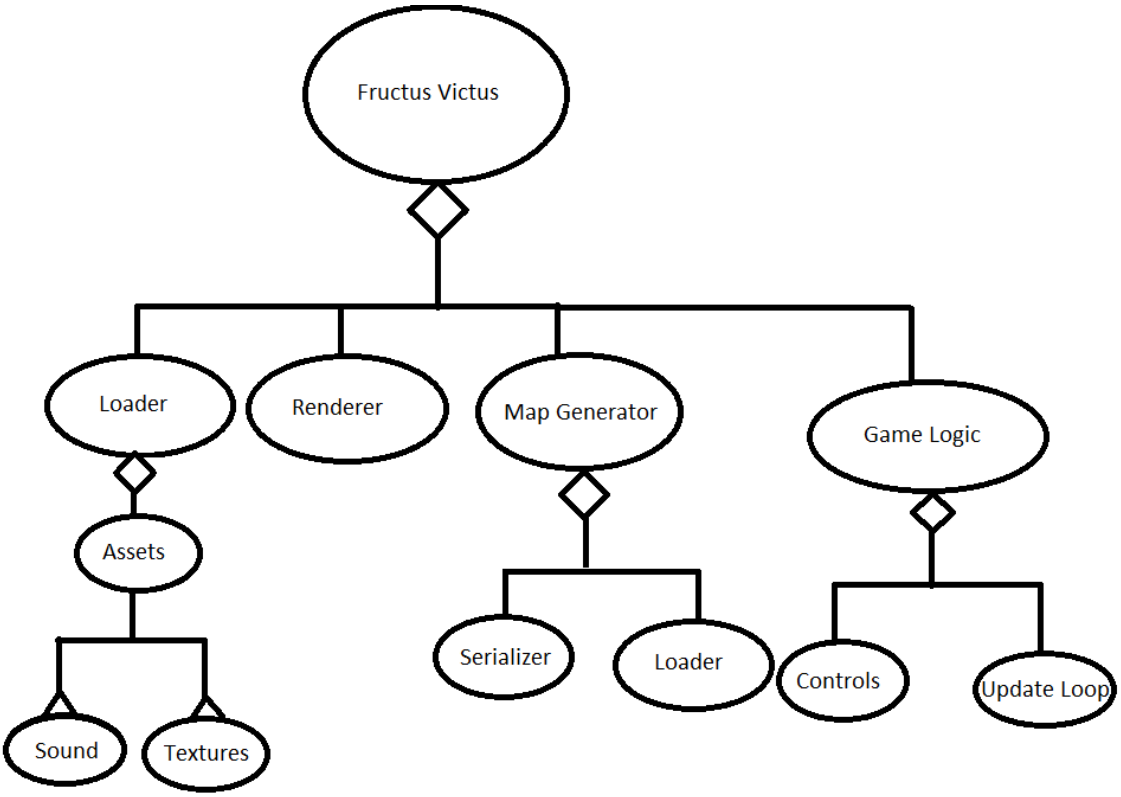
N/A

4. Other requirements

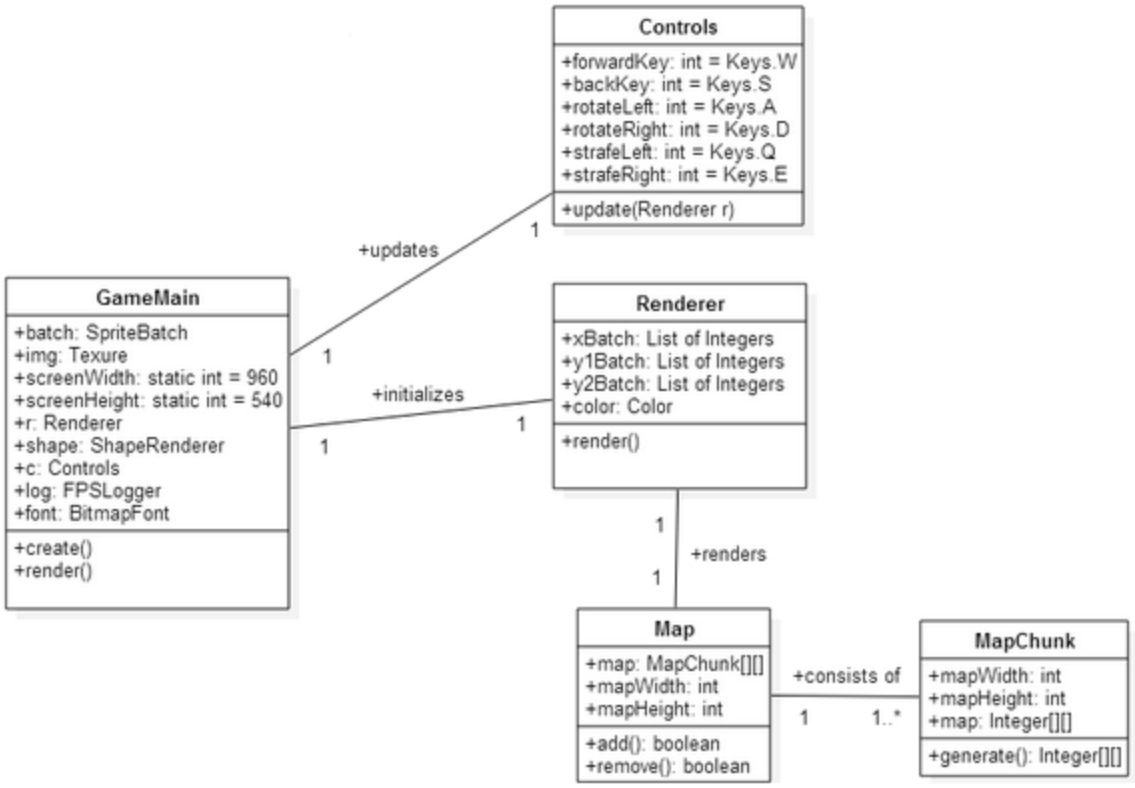
N/A

5. Diagrams

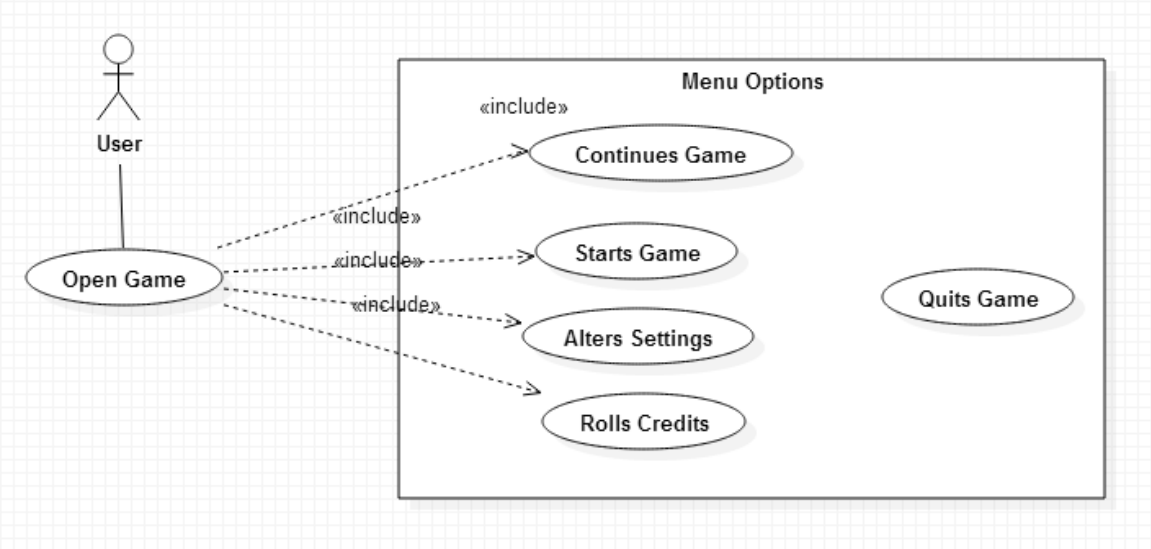
Hierarchy Diagram



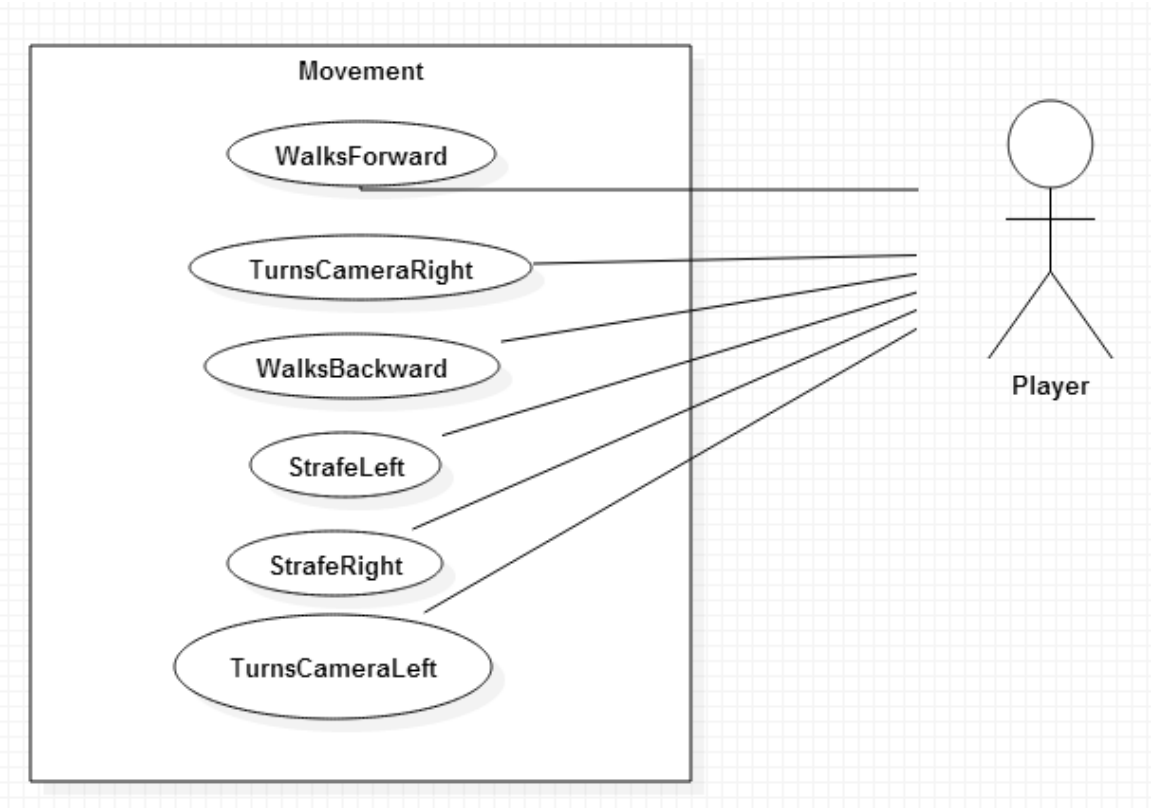
Class Diagram:



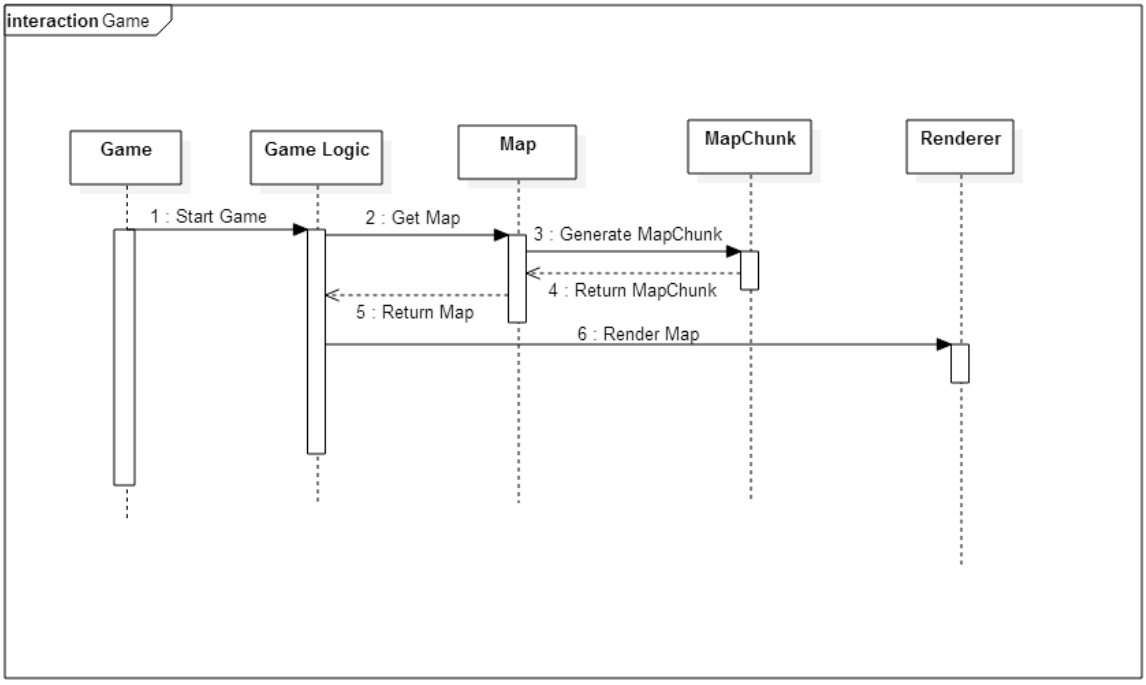
Use Case Diagram 1:



Use Case Diagram 2:



Sequence Diagram #1: Game



Sequence Diagram #2: