Software Requirements Specification

for

Space mavERICK

Version 1.0 approved

Prepared by Erick Herrera

CMPS3350

3/4/2017
# Table of Contents

Introduction 3
- Purpose 3
- Document Conventions 3
- Intended Audience 3
- Product Scope 4
- References 4

Overall Description 5
- Product Perspective 5
- Product Functions 5
- User Classes and Characteristics 6
- Operating Environment 6
- Design and Implementation Constraints 6
- User Documentation 6
- Assumptions and Dependencies 6

External Interface Requirements 7
- User Interfaces 7
- Software Interfaces 7
- Communications Interfaces 7

Misc. 7
1. Introduction

1.1 Purpose

The purpose of this software is to modify a given framework to incorporate functions from different members of the team. These functions implemented are to change the game entire to create a different game from the one started off with. The game “Space mavERICK” is a space game based off an “Asteroids” framework that was inspired by Top Gun.

1.2 Document Conventions

The main documentation of this project comes from laying out what is expected of the game in the final stage. Important assets and functions will be recorded as we seem necessary. The assets and functions that are mentioned are likely of great importance such as sound integration and image incorporation. Other smaller functions such as player movement or player health will be briefly listed as bullet points.

1.3 Intended Audience

This document is intended for students/programmers that are starting out with OpenGL. Combined with C++, students are able to build a game from the ground up or with the assistance of a framework. Figure 1 demonstrates how easily a cube can be rendered using the OpenGL library.
1.4 Product Scope

The main purpose of the project is to create various types of functions from different files that will change the main game. These changes have to be noticeable so that every group member has to include their own share of the project. Further, assets such as pictures and music are also going to be incorporated into the game to further knowledge of OpenGL and how to include outside assets into the project. Figure 2 demonstrates an example of project assets neatly organized.

```c
{
    int i;

    for (i = 0; i < 6; i++) {
        glBegin(GL_QUADS);
        glNormal3fv(&n[i][0]);
        glVertex3fv(&v[faces[i][0][0]]);
        glVertex3fv(&v[faces[i][1][0]]);
        glVertex3fv(&v[faces[i][2][0]]);
        glVertex3fv(&v[faces[i][3][0]]);
        glEnd();
    }
}
```

This OpenGL code specifies a set of commands that describe a cube. Each of these commands executes a drawing action.

The OpenGL code has provided a set of instructions for exchanging the 3D data for this cube between the application and the graphics card, allowing the cube to be rendered to the screen. Figure 1
1.5 References

While many references will be taken from various OpenGL and C++ tutorial sites, one of the consistent references to the project will be the inclusion of the K&R coding style. Figure 3 demonstrates an example of the K&R coding style.

```java
if (hours < 24 && minutes < 60 && seconds < 60) {
    return true;
} else {
    return false;
}
```
2. Overall Description

2.1 Product Perspective

The product/program is a follow up on an “Asteroids” framework given to us by our software engineering professor. This frame will be modified and added upon with functions and added assets that will contribute into making this game into a brand new game itself. Inspirations for the game come from the movie “Top Gun” with the product’s name emulating that reference in Figure 4.

2.2 Product Functions

- Player Health
- Player Lives
- Player Menu
- Start Menu
- Game Over Menu
- Help Screen
- Background changes
- Ship Changes
- Power Ups
- Music added at key points
- Miscellaneous (smaller, less relevant add-ons).

Copyright © 1999 by Karl E. Wiegers. Permission is granted to use, modify, and distribute this document.
2.3 User Classes and Characteristics

Having four members in the project team, classes and characteristics will vary on what each member will contribute to the main game itself. In essence, user classes and functions will be inherited from four different .cpp files that will have individual member functions stored in. The main.cpp file will then grab the needed functions as needed to move forward as seen in Figure 5.

```c
// function prototypes
extern void help(int);
extern void menu(char[], int);
extern void Maverick();
extern void Maverick2(int);
```

Figure 5

2.4 Operating Environment

The current operating system that the product uses is Linux. The team members all work and recreate the game environment under Linux. It is unknown if the product will reach any other operating systems or go mobile.

2.5 Design and Implementation Constraints

The one constraint that may come across the team is the lack of knowledge using OpenGL. This may result in many trial-and-error runs that attempt to create a simple object or function. Further will be needed to fully grasp the ability and strengths of OpenGL.

2.6 User Documentation

No documentation will be issued at this current time.

2.7 Assumptions and Dependencies

The assumption is that the game will be fully functional and running by the time the project is needed to be presented. However, the dependencies come to how committed and willing the team members are to contribute their share of the work.
3. External Interface Requirements

3.1 User Interfaces

Figure 6 shows an example of how the menu and user UI that may be recreated to suit the game’s needs.

![Figure 6](image)

3.2 Software Interfaces

There a few specifications that will be needed in order to properly run Space mavERICK. For starters, users will need to be running on Linux in order for the game to be ran. Second, C++ and the OpenGL libraries will need to be installed to properly display the game and its assets.

3.3 Communications Interfaces

The game may need to be ran off of Sleipnir to properly communicate with the needed libraries in the framework.


One of the biggest challenges is trying to create a game rendered in 3-D. It is a goal for the team to convert this framework into a 3D working version. However, the bar may be too hard for the team seeing as no one is experienced with OpenGL. This is why it is not
included in the main section of the document. It is a could or could not expectation at this point.