***SOFTWARE REQUIREMENTS SPECIFICATION***

**1.0 Introduction**

This section provides an overview of the entire requirement document. This document describes all data, functional and behavioral requirements for software.

**1.1 Goals and objectives**

Overall goals and software objectives are described.

**1.2 Statement of scope**

A description of the software is presented. Major inputs, processing functionality and outputs are described without regard to implementation detail.

**1.3 Software context**

The software is placed in a business or product line context. Strategic issues relevant to context are discussed. The intent is for the reader to understand the 'big picture'.

**1.4 Major constraints**

Any business or product line constraints that will impact the manner in which the software is to be specified, designed, implemented or tested are noted here.

**2.0 Usage scenario**

This section provides a usage scenario for the software. It organized information collected during requirements elicitation into use-cases.

**2.1 User profiles**

The profiles of all user categories are described here.

**2.2 Use-cases**

All use-cases for the software are presented.

**2.3 Special usage considerations**

Special requirements associated with the use of the software are presented.

**3.0 Data Model and Description**

This section describes information domain for the software

**3.1 Data Description**

Data objects that will be managed/manipulated by the software are described in this section.

**3.1.1 Data objects**

Data objects and their major attributes are described.

**3.1.2 Relationships**

Relationships among data objects are described using an ERD- like form. No attempt is made to provide detail at this stage.

**3.1.3 Complete data model**

An ERD for the software is developed

**3.1.4 Data dictionary**

A reference to the data dictionary is provided. The dictionary is maintained in electronic form.

**4.0 Functional Model and Description**

A description of each major software function, along with data flow or class hierarchy (OO) is presented.

Major Software Components used:

C++ programming language – main component of development

OpenGL – basic graphical rendering libraries

glut – advanced graphical rendering libraries

SDL – audio input and output libraries that work well with OpenGL/glut

SMB.cpp

**4.1 Description for Function “initSounds”**

initSounds – initialize SDL sound files

**4.1.1 Processing narrative (PSPEC) for function**

Function will initialize the SDL sound functions by importing sound files from the source folder

**4.1.2 Function interface description**

Function will take .wav files as input and allow for sound output

**4.1.3 Performance Issues**

There is a slight delay in sound output. We are not sure if this is an issue with SDL or with the use of chunk sounds.

**4.1.4 Design Constraints**

Only a finite number of sounds can be implemented (limited to the number of MIX chunks allowed by the system

**4.1 Description for Function “drawScene”**

drawScene – draw the scene

**4.1.1 Processing narrative (PSPEC) for function**

Function will render every stationary object in the scene, which includes the background, ground, platforms, and other various objects.

**4.1.2 Function interface description**

This function takes no parameters and outputs graphical renderings.

**4.1.3 Performance Issues**

We must determine whether we want to either move the objects in the scene along with the character when they reach the end of the screen (scrolling scene) or render very long stretches of terrain and move the camera along with the character when they reach the end of the screen (scrolling camera).

**4.1.4 Design Constraints**

Entire game will be rendered in 2 dimensions, but we must make sure the background is rendered behind all other objects in the scene.

**4.1 Description for Function “drawPlayer”**

drawPlayer – render the player in the scene

**4.1.1 Processing narrative (PSPEC) for function n**

Function will render the player in the game as they move throughout the scene.

**4.1.2 Function interface description**

This function takes no parameters and outputs the character rendering.

**4.1.3 Performance Issues**

We must decide at what point we will scroll either the scene or the camera along with the player (ex. when the character is touching the edge of the screen or when the player is approaching the edge of the screen but not touching it yet).

**4.1.4 Design Constraints**

Player must be rendered in front of the background and its movement must be constrained by boundaryTests

**4.1 Description for Function “AI”**

AI – artificial intelligence for enemy movement and actions

**4.1.1 Processing narrative (PSPEC) for function**

Function will dictate how the enemies in the scene act and interact with the player. These enemies will be controlled by algorithms and will not be controlled by players. Their movement will either be a simple back-and-forth movement or they will move towards the player’s position.

**4.1.2 Function interface description**

This function takes a time variable as a parameter and changes enemy movement with respect to time.

**4.1.3 Performance Issues**

Special performance required for the subsystem is specified.

**4.1.4 Design Constraints**

The computer characters will be restricted to movement in certain areas and will move at preset velocities.

**4.1 Description for Function “boundaryTests”**

boundaryTests – testing boundaries for movement

**4.1.1 Processing narrative (PSPEC) for function**

Function will set up all boundary testing for both character movement and AI enemy movement with objects in the scene and each other.

**4.1.2 Function interface description**

Function takes no parameters and simply bounds movement variables like character and AI enemy positions. It does not have any true output.

**4.1.3 Performance Issues**

We must keep track of all enemy and character position variables as well as the locations of all objects in the scene to properly test boundaries. This will be done with global variables to reduce the amount of data passed between functions since these functions are called constantly.

**4.1.4 Design Constraints**

The boundary numbers will be ballpark numbers which will be tested and perfected to make the smoothest collisions possible.

**4.1 Description for Function “printToScreen”**

printToScreen – print bitmap text to the screen

calls drawText to be able to render

**4.1.1 Processing narrative (PSPEC) for function**

Function defines what text will be printed to the screen. Most likely some form of directions will be printed at the start of the game, and the player’s score and number of lives will be displayed in the top right corner throughout the duration of the game.

**4.1.2 Function interface description**

Function takes no parameters and outputs text to the scene in locations determined by parameters inside this function.

**4.1.3 Performance Issues**

We must standardize the screen resolution in which the game will be displayed to ensure the text will appear the same size on every computer.

**4.1.4 Design Constraints**

Since we have elected to use bitmap fonts, there are only a select number of font options available in OpenGL/glut. We must be sure to print the text inside the viewing volume of the scene at all times.

**4.1 Description for Function “smoothMoves”**

smoothMoves – smooth keyboard-based movement

**4.1.1 Processing narrative (PSPEC) for function**

Function will ensure the player’s smooth movement via keyboard input which is slightly different than simply reading if a key has been pressed. We must check to see if the key is being held down or not as well.

**4.1.3 Function interface description**

This function takes a time variable as a parameter and changes the character’s position variables smoothly with respect to time and which button on the keyboard is pressed down.

**4.1.5 Performance Issues**

Only one button can be detected at a time for smooth movement, which we cannot change.

**4.1.6 Design Constraints**

Since only one button at a time can be detected for smooth movement, we will use simple if-else statements to determine what type of variable updating to perform.

**4.1 Description for Function “jump”**

jump – make the player jump

would modify variables in drawPlayer

**4.1.1 Processing narrative (PSPEC) for function**

Function is called when player presses spacebar. Upon call, player’s vertical position will be moved along a cosine curve (up and back down, simulating gravity) until they land back on a surface (either the ground or a platform).

**4.1.2 Function interface description**

The function will take a time variable as input and change the player’s vertical position along the cosine curve with respect to time.

**4.1.3 Performance Issues**

We will have to fine tune the mechanics of this function to figure out boundary testing for both landing on platforms and running into the bottom/side of platforms while jumping, which we expect will be quite complicated.

**4.1.4 Design Constraints**

Once the function is implemented, we will have to see how left/right movement works while jumping and possibly restrict the amount of left/right movement while in the air.

**4.1 Description for Function “display”**

display – render everything in the game

calls all rendering functions as well as AI and boundaryTests

**4.1.1 Processing narrative (PSPEC) for function**

Function will both call all rendering functions such as drawScene and drawPlayer as well as call the movement, AI and jump functions. In the OpenGL programming structure, the display method is constantly called. Because of this, it is ideal to make calls to movement functions and other variable-updating functions that must constantly be checked from display.

**4.1.2 Function interface description**

The function takes no parameters and outputs nothing directly but constantly calls all rendering and movement functions.

**4.1.3 Performance Issues**

Since this function does nothing but call other functions, no performance issues arise.

**4.1.4 Design Constraints**

None.

**4.1 Description for Function “idle”**

idle – various items to process while game is idle (time variables)

interacts with smoothMoves for time-based movement

**4.1.1 Processing narrative (PSPEC) for function**

Function updates all time variables while the game is idle. These variables are called from other functions for uniform movement speed.

**4.1.2 Function interface description**

The function takes no parameters and updates global time variables which are used in other functions to generate uniform movement speeds. The smoothMoves function is also called from here, passing a specific frame rate-related time variable to it.

**4.1.3 Performance Issues**

The glutGet(GLUT\_TIME\_ELAPSED) function returns units in milliseconds, so to get the time elapsed in seconds we must multiply the input time by (1/1000).

**4.1.4 Design Constraints**

To capture the frame rate and allow for uniform movement speeds on all computers running the game, we have set up a delta\_seconds variable which will determine the change in seconds between each frame of the game and make the game run the same speed on all computers.

**4.1 Description for Function “drawText”**

drawText – allow text to be rendered and drawn to the screen

**4.1.1 Processing narrative (PSPEC) for function n**

A processing narrative for function n is presented.

**4.1.2 Function n flow diagram**

A diagram showing the flow of information through the function and the transformation it undergoes is presented.

**4.1.3 Function n interface description**

A detailed description of the input and output interfaces for the function is presented.

**4.1.4 Function n transforms**

A detailed description for each transform (subfunction) for function n is presented. Section 4.1.4 is repeated for each of k transforms.

**4.1.4.1 Transform k description (processing narrative, PSPEC)**

**4.1.4.2 Transform k interface description**

**4.1.4.3 Transform k lower level flow diagrams**

**4.1.4.4 Transform k interface description**

**4.1.5 Performance Issues**

Special performance required for the subsystem is specified.

**4.1.6 Design Constraints**

Any design constraints that will impact the subsystem are noted.

**4.1 Description for Function “special\_down”**

special\_down – keep track of when special buttons are held down

special buttons are arrow keys, function keys, etc. (no ASCII value)

**4.1.1 Processing narrative (PSPEC) for function n**

A processing narrative for function n is presented.

**4.1.2 Function n flow diagram**

A diagram showing the flow of information through the function and the transformation it undergoes is presented.

**4.1.3 Function n interface description**

A detailed description of the input and output interfaces for the function is presented.

**4.1.4 Function n transforms**

A detailed description for each transform (subfunction) for function n is presented. Section 4.1.4 is repeated for each of k transforms.

**4.1.4.1 Transform k description (processing narrative, PSPEC)**

**4.1.4.2 Transform k interface description**

**4.1.4.3 Transform k lower level flow diagrams**

**4.1.4.4 Transform k interface description**

**4.1.5 Performance Issues**

Special performance required for the subsystem is specified.

**4.1.6 Design Constraints**

Any design constraints that will impact the subsystem are noted.

**4.1 Description for Function “special\_up”**

special\_up – keep track of when special buttons are let go

special buttons are arrow keys, function keys, etc. (no ASCII value)

**4.1.1 Processing narrative (PSPEC) for function n**

A processing narrative for function n is presented.

**4.1.2 Function n flow diagram**

A diagram showing the flow of information through the function and the transformation it undergoes is presented.

**4.1.3 Function n interface description**

A detailed description of the input and output interfaces for the function is presented.

**4.1.4 Function n transforms**

A detailed description for each transform (subfunction) for function n is presented. Section 4.1.4 is repeated for each of k transforms.

**4.1.4.1 Transform k description (processing narrative, PSPEC)**

**4.1.4.2 Transform k interface description**

**4.1.4.3 Transform k lower level flow diagrams**

**4.1.4.4 Transform k interface description**

**4.1.5 Performance Issues**

Special performance required for the subsystem is specified.

**4.1.6 Design Constraints**

Any design constraints that will impact the subsystem are noted.

**4.1 Description for Function “keyboardUp”**

keyboardUp – keep track of when keyboard buttons are let go

buttons with ASCII values (letters, numbers, etc.)

**4.1.1 Processing narrative (PSPEC) for function n**

A processing narrative for function n is presented.

**4.1.2 Function n flow diagram**

A diagram showing the flow of information through the function and the transformation it undergoes is presented.

**4.1.3 Function n interface description**

A detailed description of the input and output interfaces for the function is presented.

**4.1.4 Function n transforms**

A detailed description for each transform (subfunction) for function n is presented. Section 4.1.4 is repeated for each of k transforms.

**4.1.4.1 Transform k description (processing narrative, PSPEC)**

**4.1.4.2 Transform k interface description**

**4.1.4.3 Transform k lower level flow diagrams**

**4.1.4.4 Transform k interface description**

**4.1.5 Performance Issues**

Special performance required for the subsystem is specified.

**4.1.6 Design Constraints**

Any design constraints that will impact the subsystem are noted.

**4.1 Description for Function “keyboard”**

keyboard – keep track of when keyboard buttons are held down

buttons with ASCII values (letters, numbers, etc.)

changes variables in drawPlayer

**4.1.1 Processing narrative (PSPEC) for function n**

A processing narrative for function n is presented.

**4.1.2 Function n flow diagram**

A diagram showing the flow of information through the function and the transformation it undergoes is presented.

**4.1.3 Function n interface description**

A detailed description of the input and output interfaces for the function is presented.

**4.1.4 Function n transforms**

A detailed description for each transform (subfunction) for function n is presented. Section 4.1.4 is repeated for each of k transforms.

**4.1.4.1 Transform k description (processing narrative, PSPEC)**

**4.1.4.2 Transform k interface description**

**4.1.4.3 Transform k lower level flow diagrams**

**4.1.4.4 Transform k interface description**

**4.1.5 Performance Issues**

Special performance required for the subsystem is specified.

**4.1.6 Design Constraints**

Any design constraints that will impact the subsystem are noted.

**4.1 Description for Function “CreateGlutWindow”**

CreateGlutWindow – create a full screen game window

**4.1.1 Processing narrative (PSPEC) for function n**

A processing narrative for function n is presented.

**4.1.2 Function n flow diagram**

A diagram showing the flow of information through the function and the transformation it undergoes is presented.

**4.1.3 Function n interface description**

A detailed description of the input and output interfaces for the function is presented.

**4.1.4 Function n transforms**

A detailed description for each transform (subfunction) for function n is presented. Section 4.1.4 is repeated for each of k transforms.

**4.1.4.1 Transform k description (processing narrative, PSPEC)**

**4.1.4.2 Transform k interface description**

**4.1.4.3 Transform k lower level flow diagrams**

**4.1.4.4 Transform k interface description**

**4.1.5 Performance Issues**

Special performance required for the subsystem is specified.

**4.1.6 Design Constraints**

Any design constraints that will impact the subsystem are noted.

**4.1 Description for Function “CreateGlutCallbacks”**

CreateGlutCallbacks – call glut functions constantly for checks

functions called: keyboard, keyboardUp, special\_down, special\_up, idle, display

also allocate memory to store which key is held down

**4.1.1 Processing narrative (PSPEC) for function n**

A processing narrative for function n is presented.

**4.1.2 Function n flow diagram**

A diagram showing the flow of information through the function and the transformation it undergoes is presented.

**4.1.3 Function n interface description**

A detailed description of the input and output interfaces for the function is presented.

**4.1.4 Function n transforms**

A detailed description for each transform (subfunction) for function n is presented. Section 4.1.4 is repeated for each of k transforms.

**4.1.4.1 Transform k description (processing narrative, PSPEC)**

**4.1.4.2 Transform k interface description**

**4.1.4.3 Transform k lower level flow diagrams**

**4.1.4.4 Transform k interface description**

**4.1.5 Performance Issues**

Special performance required for the subsystem is specified.

**4.1.6 Design Constraints**

Any design constraints that will impact the subsystem are noted.

**4.1 Description for Function “InitOpenGL”**

InitOpenGL – clear the background color to white

**4.1.1 Processing narrative (PSPEC) for function n**

A processing narrative for function n is presented.

**4.1.2 Function n flow diagram**

A diagram showing the flow of information through the function and the transformation it undergoes is presented.

**4.1.3 Function n interface description**

A detailed description of the input and output interfaces for the function is presented.

**4.1.4 Function n transforms**

A detailed description for each transform (subfunction) for function n is presented. Section 4.1.4 is repeated for each of k transforms.

**4.1.4.1 Transform k description (processing narrative, PSPEC)**

**4.1.4.2 Transform k interface description**

**4.1.4.3 Transform k lower level flow diagrams**

**4.1.4.4 Transform k interface description**

**4.1.5 Performance Issues**

Special performance required for the subsystem is specified.

**4.1.6 Design Constraints**

Any design constraints that will impact the subsystem are noted.

**4.1 Description for Function “main”**

main – main loop where functions are constantly called for rendering

functions called: CreateGlutWindow, CreateGlutCallbacks, InitOpenGL, initSounds

**4.1.1 Processing narrative (PSPEC) for function n**

A processing narrative for function n is presented.

**4.1.2 Function n flow diagram**

A diagram showing the flow of information through the function and the transformation it undergoes is presented.

**4.1.3 Function n interface description**

A detailed description of the input and output interfaces for the function is presented.

**4.1.4 Function n transforms**

A detailed description for each transform (subfunction) for function n is presented. Section 4.1.4 is repeated for each of k transforms.

**4.1.4.1 Transform k description (processing narrative, PSPEC)**

**4.1.4.2 Transform k interface description**

**4.1.4.3 Transform k lower level flow diagrams**

**4.1.4.4 Transform k interface description**

**4.1.5 Performance Issues**

Special performance required for the subsystem is specified.

**4.1.6 Design Constraints**

Any design constraints that will impact the subsystem are noted.

**4.2 Software Interface Description**

The software interface(s)to the outside world is(are) described.

**4.2.1 External machine interfaces**

Interfaces to other machines (computers or devices) are described.

**4.2.2 External system interfaces**

Interfaces to other systems, products or networks are described.

**4.2.3 Human interface**

An overview of any human interfaces to be designed for the software is presented.

**4.3 Control flow description**

The control flow for the system is presented with reference to Section 5.0 of this document.

**5.0 Behavioral Model and Description**

A description of the behavior of the software is presented.

**5.1 Description for software behavior**

A detailed description of major events and states is presented in this section.

**5.1.1 Events**

A listing of events (control, items) that will cause behavioral change within the system is presented.

Menu – This will display a graphical menu to the user, and will display all of the options contained in the system.

New Game – This option will be selected from the menu, and will create a new game session for the user with all values set at default. The default value for extra lives will be 2.

Pause Game – This event will be initiated by pushing the p key on the computer’s keyboard, and will allow the user to pause their current game session. As there will be no data persistence, the game will only be paused for as long as the user’s computer is running.

Resume Game – This event will be initiated by pushing the ENTER key on the computer’s keyboard, and will allow the user to resume a previously paused game with game settings as they were at the time the pause occurred.

Quit Game – This option will be selected from the menu, and will allow the user to exit from the game. This will cause all game-related processes running on the computer to terminate.

Sound On/Off – This option will be selected from the menu, and will allow the user to turn the game sounds on or off. The game will not be sound dependant, so this will not affect the playability of the game.

Select Character – This option will allow the user to select either the male Mountaineer character or the female Mountaineer character to use in gameplay.

Jump – This event will be initiated by pushing the up arrow on the computer’s keyboard. Jump will cause the playable game character to jump in the 2d space of the game. Jumping is the main way of attacking enemy characters. To kill them, the user must skillfully jump on the heads of the enemy characters.

Move Right – This event will be initiated by pushing and holding the right arrow on the computer’s keyboard. Move Right will cause the playable game character to move to the right in the 2d space of the game for the duration of the key being held down by the user.

Move Left – This event will be initiated by pushing and holding the left arrow on the computer’s keyboard. Move Left will cause the playable game character to move to the left in the 2d space of the game for the duration of the key being held down by the user.

Duck – This event will be initiated by pushing and holding the down arrow on the computer’s keyboard. Duck will cause the playable game character to crouch, or duck, under other objects in game play. The character will remained crouched for as long as the button is pressed by the user.

Go to Menu – This event will be initiated by pushing the Esc key on the computer’s keyboard. Go to Menu will cause the user to be directed to the games main menu, where they may choose what they wish to do from the available choices on the menu.

Throw Item – This event will be initiated by pushing the spacebar on the computer’s keyboard. If the playable game character is in possession of throw-able items, pressing the space bar will throw the items when the space bar is pressed for the duration of the possession of said items.

Touch Enemy Character – Coming into contact with an enemy character other than when jumping on top of them will result in a loss of health and items from the playable character. Depending on the health status of the character, the contact may cause the playable character to die.

Fall Off World – Falling off of the world happens when the playable character falls down a hole in the 2d world. This results in the death of the playable character.

Collect Item – Items will be collected throughout game play. There will be a variety of different items, which will cause different events to happen. Those items are:

* Football Helmet - These items will cause the playable game character to grow in size, increase health, and will allow special items to be collected and utilized.
* Beer Cans – These items will be collected for points as well as extra lives. Collecting 100 of them will grant the playable character an extra life.
* Musket Balls – Collecting this item will enable the playable character to shoot musket balls at enemy characters.
* First Aid Kit – Collecting this item will grant the playable character an extra life.

**5.1.2 States**

A listing of states (modes of behavior) that will result as a consequence of events is presented.

This section describes the main states of behavior that will result from an action being presented.

Main Menu – Allows the user to choose what they would like to do. The options in the main menu are: New Game, Quit Game, and Sound On/Off.

New Game – The action of the user selecting the new game button from the main menu will result in a new game session being created for the user. All game values will be set to default in this state, and the playable character will be positioned at the start point in the game.

Pause Game – The action of selecting the pause game button from the main menu will cause the application to temporarily stop running, and all aspects of the game will be retained until the user resumes the game. A constraint of this state is that the computer must be kept on for the game data to be retained at the paused position. There is no way to save a game session in memory.

Resume Game – The action of selecting the resume game button from the main menu will allow the user to resume playing a previously paused game session. Game play will resume precisely where it left off when the game was paused. Again, the computer must remain on for all time between pausing and resuming the game session.

Game Over – This state occurs when the playable character runs out of lives in the game. The user may choose to start a new game, or exit the game from this state.

Game in Session – This state occurs when the user is currently engaged in a game session. The user may perform all in-game actions in this state, and can press the Esc key to switch to the main menu of the system.

Quit Game - Selecting the Quit Game option from the Menu allows the user to exit from the game system. The user will be prompted with a question to make sure they wish to exit, if they confirm that they wish to exit, then the system will be exited.

Sound On/Off - Selecting the Sound On/Off option from the Menu allows the user to turn the sound on or off in their current game session. The default value for sound will be on.

**5.2 State Transition Diagrams**

State Transitions of the Super Mountaineer Brothers system.

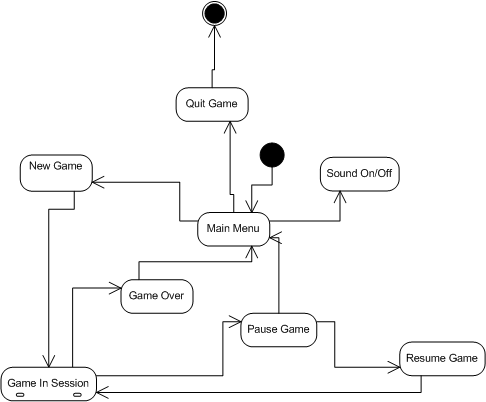


Figure 5-1 State Transition Diagram

Game in Session State Transition Diagram

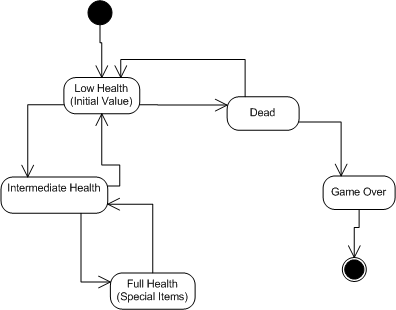


Figure 5-2 Game in Session State Transition Diagram

**5.3 Control specification (CSPEC)**

The first method where the software manages control is through opening the game application. Once the user is in the game environment, they will be directed to the main menu of the game. They will be presented with the options of “New Game”, “Quit Game”, and “Sound On/Off”. If the user selects “new Game”, the system will generate a new game session for the user to play. While in game play, the user may press “P” to pause the game. This will cause the system to pause game play. When in the paused state, the user may press “ENTER” to resume game play. At any point during game play or game paused, the user may press “Esc” to be directed to the main menu.

If the user selects “Quit Game” from the main menu, a prompt will be displayed to verify if they wish to quit. If they answer yes, the game application will be exited. If the user selects “Sound On/Off” from the main menu, they will be given the option to turn the game sounds on or off. By default, sound is turned on.

**6.0 Restrictions, Limitations, and Constraints**

The game application will run on a personal computer, and as such it will be restricted to the limits of that machine. The game will require that the personal computer be running a Windows operating system, XP or later. The user must have sound drivers installed on the computer in order for the game sound to be played. Also, the user must have a keyboard to play the game.

**7.0 Validation Criteria**

The validation process is intended to be a way of testing the game application to ensure that all possible errors are caught and handled properly and the system does not crash. The possible error scenarios are listed with their appropriate response in the following section.

**7.1 Classes of tests and Responses**

**7.1.1 Incorrect keyboard input**

If the user presses incorrect keys in an attempt to control the game, the system will ignore these keystrokes and wait for a correct key to be pushed.

**7.2 Performance bounds**

Due to the fact that this application is a 2d graphical game, graphical rendering will be limited by the graphics card contained in the computer the game is being played on. The game sounds depend on sound drivers and some sort of speaker being installed on the local machine as well.

**8.0 Appendices**

Presents information that supplements the Requirements Specification

**8.1 System traceability matrix**

A matrix that traces stated software requirements back to the system specification.

**8.2 Product Strategies**

If the specification is developed for a product, a description of relevant product strategy is presented here.

**8.3 Analysis metrics to be used**

A description of all analysis metrics to be used during the analysis activity is noted here.

**8.4 Supplementary information (as required)**