Astro

Software Requirements & Specification

Prepared by **Team Astro:**
Aaron, Ryan,
Shane, Garret,
Shane, Meghan,
Wesley, Mitchell

Version 1.5 approved
1/20/2014

Revision History 4

1. Introduction 5

1.1 Purpose 5

1.2 Document Conventions 5

1.3 Intended Audience and Reading Suggestions 5

1.4 Document Sections 5

1.5 Project Scope 6

1.6 References 6

2. Overall Description 7

2.1 Product Perspective 7

2.2 Product Features 7

2.3 User Classes and Characteristics 7

2.4 Operating Environment 7

2.5 Design and Implementation Constraints 7

2.6 User Documentation 8

2.7 Assumptions and Dependencies 8

3. System Features 8

3.1 Keyboard input 8

3.2 Game Display 9

3.3 Character Progression System 9

3.4 Save System 12

3.5 Combat System/Movement 12

3.6 Dungeon Generation 14

3.7 User Help System 14

3.8 Interactive environment 15

4. External Interface Requirements 18

4.1 User Interfaces 18

4.2 Hardware Interfaces 18

4.3 Software Interfaces 19

4.4 Communications Interfaces 19

5. Other Non-functional Requirements 20

5.1 Performance Requirements 20

5.2 Safety Requirements 20

5.3 Security Requirements 21

5.4 Software Quality Attributes 21

5.5 Business Rules 22

6. Key Milestones 23

7. Key Resource Requirements 24

Revision History

Name Date Reason For Changes Version

1. Introduction

1.1 Purpose

The purpose of this document is to provide comprehensive requirements and goals for the rogue-like[1] video game, Astro. This document helps the reader to visualize the intent of the developers and how they expect the program to run and react to input from the player. Astro is intended to provide entertainment to players through strategic game-play involving intense combat and limited resources.

1.2 Document Conventions

This document will be written in Arial font. The priority of certain features and/or requirements will be identified using tags in the beginning of the feature/requirement's description. While some features may be listed earlier than others, this is merely the nature of the document and as such there is no implicit emphasis or priority in the order presented. Any priority a feature may be assigned by the developers shall be made explicit in the feature description.

This document may make references to a "project," "software package," or "game." These and any similar phrases shall be made in reference to Astro, the project which this document describes. Any terms the reader may not be familiar with may be defined with footnotes on the page that they first appear.

1.3 Intended Audience and Reading Suggestions

This document is intended for all those involved in the development of the Astro project as well as anyone who is interested in the developmental features involved with the Astro project.

1.4 Document Sections

- **Section 2:** Provides a summary of the project, focusing on a list of features and the way the game shall operate. This provides a high-level description of the project.
- **Section 3:** Provides an in-depth view of each feature of the project, giving more detail to each topic covered in "Product Features" in Section 2.
- **Section 4:** This section covers the External Interface Requirements to fully utilize this game, such as a monitor, keyboard, or perhaps other system hardware.
- **Section 5:** This section describes the Nonfunctional Requirements that this game shall strive to meet, such as performance, security, safety, and general software quality.

[1] Rogue-like refers to a genre of role-playing games with tile-based graphics, perma-death, and random level generation.

- **Section 6:** Key Milestones section covers the dates of major events that will influence the development process of Astro. These are also dates that the product will be presented and the product will be under evaluation at that time.
- **Section 7:** Key Resource Requirements section lists the different skills and abilities needed to complete the project, such as which types of languages will need to be known to develop the product and those in our group who can accomplish each task.

1.5 Project Scope

This software will be a stand-alone desktop video game that will be used for entertainment purposes. The goal of this project is to provide a user with a leisurely but challenging gaming experience that will grab and hold their attention and interest as well as intellectually stimulate them. The game will be two-dimensional with a simplistic graphics engine and feature a robust turn-based combat system.
This software is intended to be delivered as a freeware, open-source product to interested players, with the intent that savvy users could modify the source code to suit their own needs, so long as they provide attribution to the original creators, the authors of this document.

1.6 References

- The C++ Resources Network: http://www.cplusplus.com/
- Libtcod Documentation: http://doryen.eptalys.net/data/libtcod/doc/1.5.1/index2.html
- Libtcod Developer's Hompage, maintained by Jice: http://doryen.eptalys.net/libtcod/

Libtcod C++ Tutorial (for those interested in the library): http://codeumbra.eu/complete-rogue-like-tutorial-using-c-and-libtcod-part-1-setting-up

2. Overall Description

2.1 Product Perspective

Our software package will be a stand-alone videogame in the rogue-like genre set in a futuristic sci-fi environment. It will feature a story summarized thusly: A classified military space station has gone dark and the mission is to retrieve information, including the ship's black box as well as other info, about why and what they were doing.

2.2 Product Features

In addition to the story outlined in the previous section it will have:

- Graphics
- Class and Character Progression System
- Procedurally Generated Environments
- Perma-death
- Saving
- Tutorial and Help System
- Monster Combat System
- Item and Inventory System

2.3 User Classes and Characteristics

Expected users are anyone with a computer with a desire to play video games and more specifically those who enjoy rogue like video games. Our game will require some general knowledge about rogue-likes but will include a tutorial to make it accessible to most users easily.

2.4 Operating Environment

Our game will be developed on and for Windows machines in C++. It's native OS will be Windows 7, but should be playable on all Windows machines and also any machine that can run C++. Our software requires the use of a keyboard and optionally a mouse.

2.5 Design and Implementation Constraints

The game will be written in C++ with the libtcod library included and it will also use a rendering engine to model the graphics. The computer's main memory will be utilized to store saved files so the user can play across time periods. We will design it so that the game will utilize minimal memory so that it can run at a high fps.

2.6 User Documentation

There will be an optional in-game tutorial that the user can opt into that will explain the basics of the game's mechanics in a safe area before they begin the game. In addition to this there will be a game manual included with the .exe files and a "help" option in the toolbar to assist the player mid game that will explain the game's specific mechanics.

2.7 Assumptions and Dependencies

We assume that the user has the ability to easily run C++ and the libtcod library, in addition we assume that the user's computer can run and render basic .jpeg, .gif, and .png files dynamically to facilitate the playing of our game. Also the ability of the user's OS to extract and automatically run a .exe file is assumed to be standard.

It is also assumed that the user understands English since all text contained in or concerning Astro will be written in this language.

3. System Features

3.1 Keyboard input

3.1.1 Description and Priority - Many keys, including the arrow keys and number pad, will be utilized to control the interaction between the player and the game. This is of extremely high priority.

3.1.2 Stimulus/Response Sequences - When the user presses a key, such as the 'g' key (which corresponds to item pick-up), a corresponding action will be undertaken by the player's character. The keys and the actions which have been mapped to them shall be explained to the player in a game manual, and many of the basic functions may be covered in a tutorial level for the player.

3.1.3 Functional Requirements

- [Req] - When the player presses an action key, the game shall consistently carry out the appropriate action.
- [Req] - The game shall provide the player with the opportunity to learn the action associated with each key.

3.2 Game Display

3.2.1 Description/Priority

Below is the game display model. It has a high priority.

3.2.2 Stimulus Response System

The display will appear when a game is initiated.

3.2.3 Functional Requirements

- [Req] - The game shall display player stats, inventory, console, and tile information during game play.

3.3 Character Progression System

3.3.1 Character Statistics

3.3.1.1 Strength - Base melee weapon damage and carrying capacity. Governs axes, swords, and maces. A large enough value may be required for some weapons and armor.

3.3.1.2 Dexterity - Base ranged weapon damage and combat mobility. Governs laser weapons. Increases accuracy with all lasers and melee weapon types. Increases chance of dodging enemy attacks. A large enough value may be required for some weapons.

3.3.1.3 Intelligence - Base tech usefulness and damage. Governs cryo-weapons, flamethrowers, telekinesis, and explosives. Gives bonus damage against tech-based enemies. Required for certain powerful weapons and tech.

3.3.1.4 Vitality - Base health. Potency of food. Resistance to astrocephaly[2].

[2] Astrocephaly refers to the infection found on the ship.

3.3.1.5 Priority

3.3.1.5.1 High Priority - Strength and melee damage and carrying capacity, dexterity and ranged damage, Intelligence and grenade damage.

3.3.1.5.2 Medium Priority - Intelligence and flamethrower damage.

3.3.1.5.3 Low Priority - Hit and miss chance, intelligence and bonus damage versus tech enemies

3.3.1.6 System Stimulus Response Sequences - The player may increase these statistics by completing appropriate actions in the game, such as defeating enough enemies to level up or collecting items that provide bonuses. By viewing the character information window, the player may learn the numerical values associated to their various statistics.

3.3.1.7 Functional Requirements

- [Req] - The game shall adjust the player's abilities (or lack thereof) according to the player character's statistics, through various mathematical formulae.
- [Req] - The game shall provide the player with the means to increase these statistics by collecting items and completing other such relevant tasks.
- [Req] - The game shall allow the player to view these statistics in the character information window.

3.3.2 Health mechanics

3.3.2.1 Natural regeneration - Characters passively regenerate health every turn capped at 50% maximum health. Armor may modify the rate and cap.

3.3.2.2 Health Decreases - When the player gets hit by an enemy, he or she loses health, represented by a red bar. Astrocephaly will also decrease health over time.

3.3.3 Races & Classes (Descriptions, Priorities, System Stimulus Response Sequences, Functional Requirements)

3.3.3.1 Races

- **Human -** Base class. Flat starting stats with no innate specialization. Adequate at everything. Ability: Greed: doubles the chance of money drops for 10 turns. Requires many kills to refresh.
- **Robots -** High starting vitality, strength, and intelligence. Cannot wear armor or eat food. Ability: Consume Armor: destroys armor for small health regeneration.
- **Aliens -** High base dexterity and intelligence. Low base vitality. Ability: Consume Corpse: eliminates corpses for small health regeneration.

3.3.3.2 Classes & Subclasses

- **Marine -** Combat-focused specialties. Starts with gun with cheap parts. Ability: Barricade: drops a small wall to block a hallway that lasts two turns. Requires many kills to refresh.
    - **Infantry -** 10% more laser weapon damage.
    - **Medic -** Intelligence increases chance of keeping a health pack after using it.
    - **Quartermaster -** Starts game with 3 blank key cards. These can be used to open any locks. Can hold 3 additional items at the start of the game.

- **Explorer -** Utility-based specialties. Starts with a combat knife. Have more starting vision distance. Ability: Flare: Shoots a pen flare, giving vision surrounding a line until it hits a wall or enemy. Requires a few kills to refresh.
    - **Survivalist -** Takes 1 less damage from all sources.
    - **Pirate -** Corpses have a higher chance of dropping loot.
    - **Merchant -** Takes 1 more damage from all sources. Starts with one grenade, one health pack, and one food.
- **Mercenary-** Narrowly-focused classes. Start with 3 grenades. Ability: Dash: Gives a free move without using a turn. Requires many kills to refresh.
    - **Assassin -** Has half health (calculated after bonuses). Deals double damage. Starts with a sword and 5 throwing knives.
    - **Brute -** Has extra starting vitality. Starts with 6 bonus grenades and a mace.
    - **Hacker -** Can hack vending machines for lower prices. Increased damage against robots. Has bonus intelligence at start of game. Starts with taser.

3.3.3.3 Priority of Races and Classes

- **High Priority -** Human race, Marine class, and Infantry subclass.
- **Medium Priority -** Explorer class, Survivalist subclass, Mercenary class, Assassin subclass.
- **Low Priority -** all other races, classes, and subclasses.

3.3.3.4 System Stimulus Response Sequences - The player, upon starting a new game, can navigate a series of character creation menus to create a character of a selected race and class. The game shall also allow the player to select skills from a menu that they wish to increase upon leveling up. This shall be facilitated by using the keyboard arrow keys to navigate the menus and the “Enter” key to make menu selections.

3.3.3.5 Functional Requirements

- [Req] - The game shall allow the player to choose a race and class combination from the available options.
- [Req] - Likewise, the game shall allow the palyer to choose skills into which they can allocate points upon leveling up.
- [Req] - The game shall provide consequences, both positive and negative, for choosing a particular combination of race, class, and skills over another.

3.4 Save System

3.4.1 Saving

3.4.1.1 Description/Priority - The save system will feature both user created save file and autosave files which store character progression. The game will autosave when entering a new level/deck and when exiting the game. Users will also be able to save at any point during the game. User save files will also be deleted upon death. The priority for this feature is high.

3.4.1.2 Stimulus/Response Sequence - Upon launching the game, users will have the option of starting a new game or loading any existing save files from the main menu. Users will be also be able to load their save file via the in-game menu. Users can create/overwrite their save files via the in-game menu.

3.4.1.3 Functional Requirements

- [Req] - The game will feature a save system to store all user progression, including level, inventory, and location.
- [Req] - The save system will allow users to create their own saves at any point in the game
- [Req] - The save system will have an autosave feature, which saves when entering a new level/deck and when exiting the game.
- [Req] - The game will feature Perma-Death, which will delete the user's save file upon death.

3.5 Combat System/Movement

3.5.1 Melee combat

3.5.1.1 Description/Priority - The melee combat system will consist of the typical rogue-like combat system; being adjacent to an enemy and pressing the appropriate direction toward them to initiate an attack. This is of high priority as it is crucial to basic gameplay.

3.5.1.2 Stimulus Response Sequences - The melee combat will be instigated when an enemy is next to the player and either the player or the enemy initiates the button sequence to attack one another. This will then, upon a hit landing appropriately onto the recipient of the attack, reduce the recipient's health; the factors that reduce the health are the strength of the weapon, the strength of the user and the recipient's defense.

3.5.1.3 Functional Requirements

- [Req] - The game shall allow movement thus facilitating attack.
- [Req] - The game shall allow items to be used as weapons in melee combat.
- [Req] - The game shall allow enemies to move and attack as well.
- [Req] - The game shall allow the comparison of character statistics.
- [Req] - The game shall allow the removal of health and death.

3.5.2 Ranged combat

3.5.2.1 Description/Priority - The ranged combat system will consist of the standard rogue-like ranged combat engine which is described as such: highlighting a tile to attack and drawing a vectored line to cover the damage area. This is a high priority feature, although it is less critical to the project's completion than melee combat.

3.5.2.2 Stimulus Response Sequences - The ranged combat will be instigated when the player presses the "fire" button and begins the firing sequence. This will display a range of certain tiles that can be selected for attack. The player selects one tile to attack and then a vectored line will display between the center of the player's tile and the center of the designated tile, any tile covered by this line will take the damage indicated by the weapon being used minus the defense of the recipient.

3.5.2.3 Functional Requirements

- [Req] - The game shall allow for a button to be pressed to allow for ranged attack.
- [Req] - The game shall support the ranged attack range to be displayed.
- [Req] - The game shall allow the player to select a tile to attack.
- [Req] - The game shall allow a vector to be drawn between points.
- [Req] - The game shall allow the player to deal damage and read statistics.

3.5.3 Enemy AI

3.5.3.1 Description/Priority - The enemy AI will respond to the player's movement and attempt to attack the player, work with the player and barter with the player based on their disposition. This is of high priority because it is necessary to play an entertaining game.

3.5.3.2 Stimulus Response sequences - During the game the player will move or commit some actions. While committing this, the game will decide where the AI controlled enemies should move or if they should perform any actions. The game will take in many factors in this such as current position of player and AI enemy, health, statistics, etc. Upon correct implementation it will seem like a fluid system where the player is fighting against actual opponents.

3.5.3.3 Functional Requirements

- [Req] - The game shall have actors[3] with which to move.
- [Req] - The game shall have an AI system to decide how to move the actors.
- [Req] - The game shall have a way to move actors independent of the player.
- [Req] - The game shall have a way to take in all data about the game-state.

[3] Actors refer to any object in the game which can be represented with a tile on the map

3.5.4 Movement and Turn Initiative

3.5.4.1 Description/Priority - The turn initiative system will determine the order of which enemies are able to act. Each action holds a cost. An actor capable of smaller actions will have a higher frequency of turns. This system has a very high priority because it is necessary for combat for game-play.

3.5.4.2 Stimulus Response Sequences - After a player presses any command key, the player character will immediately preform the entered command. The player's next turn will not occur until the delay from the cost of the command is completed. Meanwhile, every actor with a action cost less than the player's will be able to perform their actions. Every actor with larger cost actions will have their current action cost reduced by the player cost amount.

3.5.4.3 Functional Requirements

- [Req] - The game shall have actors with which to move.
- [Req] - The game shall have actions with costs.
- [Req] - Actions will determine the turn order of actors.

3.6 Dungeon Generation

3.6.1 Description/Priority - The layout of each level will be randomly generated from an algorithm. Dungeon creation is high priority due to the game requiring a dungeon to function.

3.6.2 Stimulus Response Sequences - When a level is accessed for the first time, rooms will be created with a chance of being chosen from a pre-made set. Corridors and hallways will attach the rooms. Limits will be placed in the generation to prevent inaccessible and overlapping rooms.

3.6.3 Functional Requirements

- [Req] - The game shall generate levels when they are first accessed.
- [Req] - The dungeon generation will have parameters.

3.7 User Help System

3.7.1 In-Game Tutorial

3.7.1.1 Description/Priority - The in-game tutorial will be an option tutorial that will be comprised of a small in-game level (most likely a small space ship) that will teach the player the basics of the inventory system, combat system, and movement system. The priority of this tutorial would be below the basic features of the game as they would have to be implemented prior the tutorial.

3.7.1.2 Stimulus Response Sequences -The player will be able to access tutorial via the main menu.

3.7.1.3 Functional Requirements

- [Req] - The tutorial shall let a player exit it at any point.

3.7.2 User Manual

3.7.2.1 Description/Priority - This will be a textual manual that explains the mechanics of the game and it will be accessible as both a document included in the game files as well as an in-game help menu. This will also explain key-binding. This has a priority just below that of the in-game tutorial.

3.7.2.2 Stimulus Response Sequences - The in-game help menu shall be activated during a game via the menu.

3.7.2.3 Functional Requirements

- [Req] - The game shall provide a textual manual, which the player can navigate, explaining core mechanics of the game as well as which actions have been bound to which action keys.
- [Req] - The game shall also be distributed with a text file that contains the text found within the in-game help menu.

3.7.3 In-Game Tile Info Tool

3.7.3.1 Description/Priority - This window will display relevant tile information to the player. Items and enemies located on the selected tile, as well as possibly the infection level and oxygen level of the selected tile.

3.7.3.2 Stimulus Response Sequences - The player shall indicate a tile, either by moving onto it or choosing the tile with the look function, and the game shall display the relevant info in the appropriate area in the game display.

3.7.3.3 Functional Requirements

- [Req] - The game shall store relevant tile information to be retrieved at the player's will.
- [Req] - The game shall display this information in a manner that is understandable by the player on the game display.
- [Req] - The game shall restrict the player's access to tile information concerning tiles that they have yet to explore.

3.8 Interactive environment

3.8.1 Inventory System

3.8.1.1 Weapons/ammunition

3.8.1.1.1 Description/Priority - The weapons in the game are summarized in two sections, melee (requiring adjacency to attack) and ranged (which can be fired from a distance and hurt multiple enemies.) Ranged weapons will have an attachment system that allows the user to select different stats for the weapon or different scatter patterns. This is of high priority because without it the player cannot attack.

3.8.1.1.2 Stimulus Response sequences - The player will find weapons and ammunition lying around the map and will be able to pick them up via the inventory and equip using the equipment screen.

3.8.1.1.3 Functional Requirements

- [Req] - The game shall have a pickup system.
- [Req] - The game shall have an inventory system.
- [Req] - The game shall have an equipping system.

3.8.1.2 Armor

3.8.1.2.1 Description/Priority - The armor in game will be equippable items that allow the user different levels of protection by way of armor class and stat manipulation. This is of medium priority as it allows for more in-depth gameplay.

3.8.1.2.2 Stimulus Response sequences - Much like with weapons, the user will find armor lying around and equip it via the equipment screen.

3.8.1.2.3 Functional Requirements

- [Req] - The game shall have a pickup system.
- [Req] - The game shall have an inventory system.
- [Req] - The game shall have an equipping system.

3.8.1.3 Tech

3.8.1.3.1 Description/Priority - Tech is any item that does not fit into a weapon/armor category and is not used for health regeneration. This includes items such as pen flares, grenades, etc. This is of medium priority because it allows in-depth play but is not critical.

3.8.1.3.2 Stimulus Response sequences - Much like the weapons, the user will find tech lying around and equip it via the equipment screen.

3.8.1.3.3 Functional Requirements

- [Req] - The game shall have a pickup system.
- [Req] - The game shall have an inventory system.
- [Req] - The game shall have an equipping system.

3.8.1.4 Health items

3.8.1.4.1 Description/Priority - The health items will have two categories, health-kits and food items. The health-kits will instantly restore a portion of the player's health upon use while food will regenerate health over a certain amount of time at a certain rate. Implementing health items is of medium priority.

3.8.1.4.2 Stimulus Response sequences - Much like with weapons, the user will find health items lying around and equip it via the equipment screen.

3.8.1.4.3 Functional Requirements

- [Req] - The game shall have a pickup system.
- [Req] - The game shall have an inventory system.
- [Req] - The game shall have an equipping system.

3.8.1.5 General Inventory Mechanic

3.8.1.5.1 Description/Priority - The inventory system will be a sidebar on the screen that will list the items that the user has access to. This is a high priority.

3.8.1.5.2 Stimulus Response sequences - The user can open the inventory with the "i" key or clicking the sidebar. The user may then select the item he/she wants from the menu with the letter corresponding to it.

3.8.1.5.3 Functional Requirements

- [Req] - The game shall have an inventory system.
- [Req] - The game shall have an equipping system.
- [Req] - The game shall have a sidebar screen.
- [Req] - The game shall have the ability to accept user input from keyboard.

3.8.2 Keycard Progression

3.8.2.1 Description/Priority - To get into some specialized rooms, such as infirmaries, armories, or some special decks, the player will need to find certain keycards from dead bodies to advance, i.e. to get into the infirmary the user will need to find a doctor's keycard. This is of low priority.

3.8.2.2 Stimulus Response sequences - The user must find and kill the appropriate character that has the keycard and then put it in his inventory and then use it to access the door it goes to.

3.8.2.3 Functional Requirements

- [Req] - The game shall have a keycard system.
- [Req] - The game shall have an inventory system.
- [Req] - The game shall have a premade room editor.

4. External Interface Requirements

4.1 User Interfaces

The user will play the game via their monitor and the standard “windowed” screen display, an example of the Astro main menu is displayed in Figure 4.1 below. In addition they will need to have peripheral devices and software packages as indicated in the section “Hardware Interfaces.”

- [Req] - The game shall be able to port to the monitor.
- [Req] - The game shall be able to display a windowed display in the OS’s native screen.

Figure 4.1: Main Menu

4.2 Hardware Interfaces

The keyboard’s arrow keys or ‘w’, ’a’, ’s’, and ’d’, or the number pad keys will move the character, menu selector, and in-game menus. The mouse selection will also be available as a default, but opt out, option to the player to navigate with more ease. The mouse will be used to aim projectiles select inventory, investigate tiles, and other miscellaneous features in game. Specific keys may be bound to abilities in game at the user’s discretion. The ‘Esc’ key will produce a pause menu. Errors will be printed to console output. The user is required to use a keyboard and mouse to effectively use the software.

- [Req] - The game shall take input from the keyboard.
- [Req] - The game shall interpret input from the keyboard.

4.3 Software Interfaces

This product uses the Libtcod library. It will save to a binary file. Items and progress will be saved, along with character progression. The user must have an OS that supports the unzipping of files and the running of .exe's to allow seamless download of files and playing. The code for the game will be written in C++.

- [Req] - The game shall store its current state in a save file.
- [Req] - The game shall load a save file to restore a previous state.

4.4 Communications Interfaces

This product, once installed, requires no active Internet connections, or inter-device communication.

5. Other Non-functional Requirements

5.1 Performance Requirements

5.1.1. Frame rate:

Requirement: Astro shall execute graphics at least 20 fps (frames per second).
Justification: High frame rate is not necessary for a turn-based role-playing game, however since the game is terminal-based and not graphically intensive, a higher fps is not out of the question.

5.1.2. Processing Power:

Requirement: Astro will only a require a 20 mhz processor to run smoothly.
Justification: Astro is a terminal-based game and will not require much processing power.

5.1.3. Executable Size:

Requirement: The size of Astro's executable will be less than one gigabyte.
Justification: The entire libtcod library is less than 15 megabytes and the game is programmed in C++, so this is not an unreasonable limitation.

5.1.4. Save File Size:

Requirement: Astro's individual save files will not exceed 10 megabytes.
Justification: The save files are compressed text files so it inconceivable for them to fail to meet this requirement.

5.1.5. In-game Load Times:

Requirement: Astro's load time between levels will be near instantaneous
Justification: Astro's is graphically minimalistic so loading new levels will not be time consuming.

5.1.6. Launch Time:

Requirement: The time to reach the main menu upon launching the game will not exceed 10 seconds.
Justification: Since Astro's size is required to be small and played via the terminal, launch time will be quick.

5.2 Safety Requirements

5.2.1 External System Integrity:

Requirement: The game will not be able to open, create, edit, or delete any non-game related files and will run in isolation to other unrelated processes that are operating concurrently.
Justification: This will help to prevent Astro from causing any damage to the computer or its internal components.

5.2.2 Save File Integrity:

Requirement: Upon a game crash, user's progress will be reverted to the earliest save prior the crash.
Justification: This will minimize the loss of user's data that occurs after a crash and the implementation of auto-saves additionally aid to minimize data loss.

5.3 Security Requirements

5.3.1 Online and Multi-Player Functionality

Requirement: Astro is a single-player game with no online or multi-player functionality.
Justification: There are very low to non-existent security risks while playing Astro.

5.3.2 Preventing Save File Manipulation:

Requirement: Save files used in the game will be compressed so they become illegible to users.
Justification: To essentially create a black-box which prevents users or other parties from manipulating save files to prevent possible cheating.

5.4 Software Quality Attributes

5.4.1 Save Files

Attribute: The user will be able to load their previous save files.
Reasoning: This adds replayability, as it allows the player to resume a previously played session, a feature that many players would prefer.

Attribute: The user will be able to save the game during any point of play.
Reasoning: Flexibility is added since Astro is allowing players to leave the game and come back to it at any time, which is something the player will be looking for.

Attribute: Save files for Astro are small enough to be transferred between computers that have the game installed
Reasoning: The game has portability since you can continue playing the same game on any computer that has the game installed, which is something that may be desired by the player.

5.4.2 Mapping System

Attribute: The mapping system will have both pre-made maps and randomly-generated maps.
Reasoning: This allows for adaptability of the game play and reusability since players will always be able to experience a new gaming environment.

5.4.3 Character Variety

Attribute: There are options between different choices of races, classes, and sub-classes.
Reasoning: Users will have new experiences when replaying the game with these choices, since their choices will influence their playing strategies, which is something desired by the developers.

5.4.4 Tutorial and Help System

Attribute: There shall be a tutorial/FAQ, manual/help system.
Reasoning: Usability can be seen in this since this will help teach new user the controls for Astro, what each weapons do, how the items helpful, etc. This would be most helpful to players who have never played Astro, or any game that is a rogue-like genre.

5.4.5 Keyboard Use

Attribute: Multiple keys are not pressed at a time, and time is not a factor since nothing in the game happens unless a turn is taken.
Reasoning: Usability is shown since it allows for user with limiting physical abilities, such as with difficulties with arm/hand movement or are slower in movements, to play with more ease. This is allows for an even broader base of people we can have as players.

Attribute: Key-mapping, the ability to change what a key does in the game, will be a feature.
Reasoning: This is a feature of most rogue-likes and will allow for those who prefer one set of keys to play the game to find Astro more user-friendly. This is something that will allow customizability for the user.

5.5 Business Rules

Roles will be shared among the members of the group. Specific tasks will be delegated throughout the development process. Each will have different roles and parts of the product they will have to work on and maintain. These can include

- Art and Graphics
- Map & Rendering
- Item/Inventory
- Actor Development
- User Interface
- Testing

Any major decision in terms of the game development must be discussed and decided among the members of the group. There must be a general/majority consensus for decisions that have resulted in conflict.

6. Key Milestones

#	Milestone	Target Completion Date	Comments
1.	SRS Document	01/21/14	Rough draft to be done by 01/18 to allow time for polishing
2.	SDD Document	01/27/14	Rough draft to be done by 01/25 to allow time for polishing
3.	Scrum Sprint 1 - End	02/23/14	Create presentation following 02/17 to be shown 02/24
4.	Scrum Sprint 2 - End	03/30/14	Create presentation following 03/24 to be shown 03/31
5.	Scrum Sprint 3 - End	04/27/14	Finalize software package and create presentation to be shown 05/01
6.	Software Festival	05/01/14	Project complete

7. Key Resource Requirements

Major Project Activities	Skill/Expertise Required	Internal Resource	External Resource	Issues/ Constraints
User interface design	C++ and libtcod	The group has previous experience with C++ and data structures	Libtcod documentation	N/A
Graphical tile support	Photoshop, SDL, OpenGL, libtcod, and C++	Shane E. has digital graphic design experience	SDL, OpenGL, and libtcod documentation	SDL and OpenGL are new technologies for the group.
Class and race system	C++, and libtcod	The group has previous experience with C++ and data structures.	Libtcod documentation	Game-play balance must be taken into account.
Procedural map generation	Data Structures C++ and libtcod	The group has previous experience with C++ and data structures.	Libtcod documentation	N/A
Robust save system	C++ and libtcod	The group has previous experience with C++ and data structures.	Libtcod documentation	Save file system must meet file size requirements.
Item and inventory system	C++ and libtcod	The group has previous experience with C++ and data structures.	Libtcod documentation	N/A
Non player character AI	C++ and libtcod	The group has previous experience with C++ and data structures.	Libtcod documentation	N/A
Complete combat system	C++ and libtcod	The group has previous experience with C++ and data structures.	Libtcod documentation	N/A
User help system	C++, libtcod, and writing	The group has previous experience with C++ and data structures.	Libtcod documentation	Someone new to the game is needed for feedback.