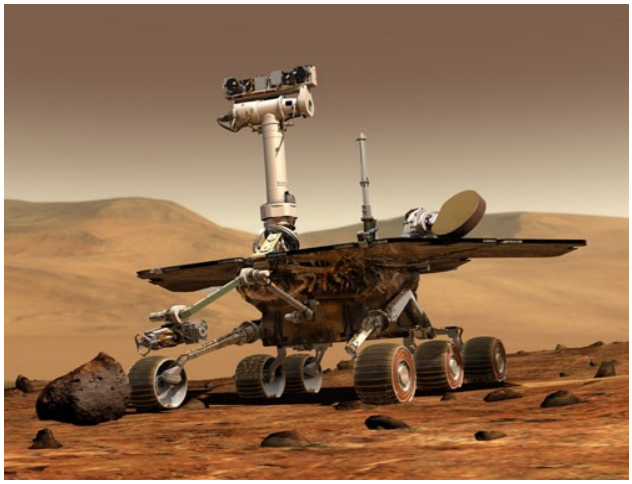
Software Requirements Specification

for

Spirit Pathfinding Engine



Document Information	
Company	Four Factor Studios © 2011
Author	Michael K. Antonelli
Revision Date	12-February-2011
Software Version Number	0.1
Document Revision Number	0

Image Courtesy NASA/JPL-Caltech.

Software Version History

Name	Date	Software Changes	Version
Michael K. Antonelli	12-February-2011	Initial Revision	0.1

Document Revision History

Name	Date	Document Changes	Revision #
Michael K. Antonelli	12-February-2011	Initial Revision	0

Table of Contents

1. Introduction

1.1 Purpose

Spirit is a pathfinding engine which will determine least resistive paths between locations. Spirit is intended to be a self-contained dynamic library which runs along side a host program. The host program will pass Spirit required pathing data and objects at initialization, and in return Spirit will allow objects to request pathing solutions.

1.2 Document Conventions

This document loosely follows IEEE 830 Software Requirements Standards. It will contain an introduction, an overall description, software functional requirements, and additional information including appendices.

1.3 Project Scope

Spirit is a self-contained software component which may be part of a bigger project. Spirit will have its own integration specifications for other software to use to communicate with it. Spirit will have the ability to read several formats of data provided the correct libraries are used. This project will include both the Spirit engine along with some included libraries for Four Factor Studios projects.

1.4 References

[IEEE 830-1998] IEEE Recommended Practice for Software Requirements Specifications

[SDL-1.2.14] Simple DirectMedia Library

2. Overall Description

2.1 Product Perspective

Spirit consists of libraries and algorithms which are used for computing paths, an engine for parallel processing of pathing, and interfaces which allow other software packages to bi-directionally communicate with Spirit. Libraries include stack and queue implementations, sorting implementations, pathing algorithms, event handling, and utility functions. The engine will include processing logic, initialization and cleanup routines, error handling, logging, parallel processing, and reporting. The interfaces will include path cost structures, object structures, suggested path structures, and exposed functions for client interactions.

==========Insert Diagrams Here==========

2.2 Product Features

Spirit provides the following functions to a host program.

2.2.1 Initialization

During initialization the host program shall send Spirit path cost data, a shared lock variable for synchronization, and the location of the objects list. Spirit can optionally take a log location, otherwise it shall output to its own log file. Spirit shall then start up its threaded process.

2.2.2 Engine

Spirit shall have an engine which controls scheduling of work. The engine is the primary processing unit in Spirit. Under reasonable conditions the engine shall not interrupt the primary game engine. The engine shall implement use of a priority system which describes actions which need to be completed immediately, and actions which are not urgent. The engine shall make use of a lock variable to prevent consuming resources of the host program.

2.2.3 Pathfinding

Spirit shall implement a pathfinding algorithm which attempts to determine low cost paths to a goal. Spirit attempt to determine usable paths immediately, and then Spirit shall use free processing time to attempt to computer improved paths. Spirit shall attempt to give the engine a usable path within one host program loop. This path may not be a solution guaranteed path. Spirit will continue to attempt to find closed solutions paths.

2.3 Operating Environment

Spirit shall be platform agnostic. Where platform specific calls must be used, macros or an abstraction library will be used in its place such that Spirit can easily become cross platform. Spirit depends on [SDL-1.2.14] or greater.

2.4 Design Constraints

Spirit is required to operate in short quantas of time. Spirit is required to return possible paths which may not have a solution yet.

Spirit is licensed under the LGPL and is subject to its limitations.

2.5 User Documentation

All code in Spirit shall have commenting containing the basics regarding the operation of the given function. Software requirements shall be updated with released program versions. Design Specifications shall be made available where possible for clients to use Spirit.

3. System Features

This section describes the design intent of Spirit's features. These features directly relate a Spirit design entity.

3.1 Initialization

Initialization shall be called upon startup. Spirit prepare its pathfinding routines and logging during this activity. Spirit shall return a status to the host program about its usability. Spirit shall not begin processing until it is issued a start or resume command.

3.2 Pause/Resume

Spirit shall be pausable. The host shall be able to issue a pause command to Spirit which will prevent Spirit from operating until a resume command is issued. When a pause is issued, Spirit may finish some of its current actions and will not yield immediately.

3.3 Logging

Spirit will have the ability to log engine events. Spirit shall log all critical events, and have the ability to log debugging events based on a specified debug level. Spirit shall be capable of accepting either a callback function to attach to a host process' logging system.

3.4 Parallel Processing

Spirit shall be capable of processing in a single thread or in a parallel environment. Spirit shall attempt to limit processing such that Spirit will not seize processing time from the host process unless commanded to.

3.5 Pathfinding Algorithm Interface

Spirit shall have a pathfinding algorithm interface. Using the interface, Spirit will be able to use dynamic, run-time, or statically linked pathfinding. Spirit shall be able to accept any pathfinding algorithms which conform to the interface. Spirit shall be able to load several algorithms and choose based on host program preferences for a per-object basis.

3.6 Pathfinding Utility Libraries

Spirit shall incorporate libraries which support pathfinding. Spirit shall also be capable of using dynamically loaded libraries from the Pathfinding Algorithm Interface. Pathfinding Utility Libraries shall include the following:

- **Queues**
- **Stacks**
- **Vector Functions**

- **Kinematic motion functions**
- **Path deviation**
- **PID motion control**

3.7 Event Handling Pipeline

Spirit shall respond to events from the host program. Spirit shall queue events and process them based on priority. Spirit shall handle high priority events before low priority events. Spirit shall use the Event Handling Pipeline to complete computations in steps. Spirit shall use a dynamic priority pipeline, capable of handling removals as well as promotions.

3.8 Cleanup

Spirit shall have a cleanup operation which releases memory, closes threads, unregisters events, and stops processing.

4. External Interface Requirements

4.1 SDL Integration

Spirit shall use SDL for its threading and event handling modes.

4.2 Pathfinding Algorithm Interface

Spirit shall implement a standard interface for dynamic or statically linked pathfinding algorithms and their utilities.

==========Insert table or description here==========

4.3 Host Program Interface

Spirit shall have a host program interface as follows.

5. Nonfunctional Requirements

5.1 Performance Requirements

Spirit shall be capable of performing pathfinding without impacting its host program. Specific requirements will be specified at a later version.

5.2 Security Requirements

Spirit shall not read or write to any system files or locations. Spirit shall not read or write files outside of its operating directories. Spirit shall not create directories. Spirit shall not read or write registry keys, environment variables, or symbol tables.