Rovers

Design and Development

# Overview

This document is a brief review of the design and development process for the *Rovers* project. Discussed is the motivation and goals, a review of the requirements, a project plan with rationale and the design of the final solution.

# Motivation

The Rovers project aims to develop a solution for the problem of exploring a plateau on Mars with a squad of robotic rovers. A detailed description can be found in *problem\_description.txt*. However, a second and arguably more important goal is to demonstrate Daniel’s proficiency as a professional software engineer. This includes demonstrating a disciplined decision making process, an understanding of the complete SDLC and utilization of effective software engineering practices and tools.

# Requirements

The problem description specifies most of the system adequately, thus in the interest of brevity only ambiguous requirements, assumptions and the resolutions are listed here. The requirements won’t be formally tested themselves, but will drive the development and unit tests.

1. The plateau is a finite, uniform 2d grid.
2. All grid coordinates, dimensions and rover (X, Y) positions are positive integers.
3. A rover's facing is given as N, E, S or W, matching North, East, South and West respectively.
4. A rover will halt and ignore future commands if the next command is invalid.
5. A rover command is invalid when it results in moving off the plateau edge, crashing into another rover or is not a character in the set {'L', 'M', 'R'}.
6. A spin command, {'L', 'R'}, is relative to the rover's facing.
7. A move command, 'M' moves in the direction of the rover's facing.
8. A rover perfectly receives all sent commands.
9. A rover’s actions are independent of other rovers.
10. A rover's position is always denoted by an X Y F tuple, matching x-coordinate, y-coordinate and facing respectively.
11. A rover that is executing spin commands is classed as 'moving', implying the next rover must wait.
12. A single (X, Y) position corresponds to a maximum of one rover. For multiple rovers, this implies they'll start and end in non-overlapping positions.
13. Multiple rovers starting in the same (X, Y) position aborts the simulation (physically impossible universe!).
14. Input and output commands are ASCII encoded
15. Successive rovers will being exploring immediately after the previous rover finishes.
16. TBD – input and output, using files or std in/out ?

Requirement 4 is chosen to mimic and handle uncertainty in the environment – the robot will initially attempt to complete its mission, but if it encounters an impossible scenario, it’ll stop and await new commands. Requirement 5 is a reasonable choice to prevent rover losses.

# The Plan

*TODO – briefly discuss the agile process, task breakdown, time management, minimalist completion and supporting tech.*

*TODO – Test Plan, verification (unit tests, integration, blab la)*

# The platform

*TODO – discuss tools and technical framework decided upon in our journal*

# the design

*TODO – suppose we could whack together a brief UML or some such. Driving motivations, ie,*

* *keep the solution a single module, command line app*
* *reuse existing stl, boost components over adhoc*
* *keep the implementation unit-testable*