Table 1: Revision History

Date	**Developer(s)**	**Change**
09/23/2016	Or	Create outline .tex
09/25/2016	Ian	Add Meeting Plan, Team Roles, and Proof of Concept Plan
09/25/2016	Mikhail	Added Communication Plan and Technology
09/26/2016	Mikhail	Added Git Workflow Plan
...	...	...

SE 3XA3: Development Plan
Rogue Reborn

Group #6, Team Rogue++

Ian Prins	prinsij
Mikhail Andrenkov	andrem5
Or Almog	almogo

Friday, September 30, 2016

Rogue is a classic video game, originally developed in 1980. This project aims to rewrite Rogue using modern software development techniques and a modern language. This document outlines the development plan for this project. Incuded below are plans for how team members will coordinate, and divide work; what technology will be used and the project code style; and the project schedule, with details for the requirements for the Proof of Concept deadline.

1 Team Meeting Plan

Meetings will be held once weekly, in Thode library at 3:30pm on Wednesday. These weekly meetings will be chaired by the team leader. The team leader will be responsible for developing a rough meeting agenda and ensuring the meeting follows the agenda. This agenda will no be posted but will be briefly outlined to the meeting participants. Full minutes will not be recorded, but the meeting scribe will record the outcomes of the meeting. Any such outcomes will be posted to the team git repository. If a team member cannot make the meeting, a brief summary of the meeting and links to its outcomes will be posted in the team slack channel by the meeting scribe. Any changes to the meeting format, location, or time will be posted to the team slack channel by the meeting chair.

2 Team Communication Plan

Team communications will be distributed across several social platforms. This way, conversations can be grouped together according to the degree of formality and desired visibility. Specifically, Slack will be used to exchange informal messages for purposes such as conveying meeting times or briefly discussing topics. Team members will be required to check this service at least once

per day in order to facilitate quicker response times and encourage more open communication. Next, the GitLab ITS will serve as an official means of tracking bugs, reporting issues, and announcing any other code correction request. All other communications will be performed in person during the lab sessions as well as the weekly group meeting. Currently, there is no plan to include any other communication streams as this will most likely dilute the conversations to a point where accessibility and traceability is sacrificed.

3 Team Member Roles

Mikhail will be the team leader. The team leader is responsible for chairing the meetings, allocating work to the team members, and ensuring that all team members are up-to-date about the project status and deadlines. The other team members will alternate fulfilling the role of meeting scribe. Outside of meetings, various team members will fill the role of experts in the project technology. Mikhail will be the git and LaTeX expert, Ori the testing expert, and Ian the C++/libtcod expert. Expert roles don't constitute work allocations, but rather an indication of who should make sure to be up-to-date on portions of the project and who questions should be directed to.

4 Git Workflow Plan

To minimize the overhead associated with creating, updating, and managing project files, the Rogue++ team will collaborate using several Git integration flows. In general, all project source files and documents associated with the main (stable) development branch will be hosted in a central GitLab repository. Any developers (or distinguished stakeholders) may clone, view, and modify this code, given that their changes do not compromise any tested functionality. Note that it is still acceptable to commit a change to this branch that is not fully integrated; however, the application must be able to successfully compile and run. In the event that a prototype demonstration is required, a new branch will be created to host any temporary changes that will not be merged back into the main branch. This way, developers are free to adapt existing source code in any manner they choose in order to showcase their progress to a stakeholder without violating the integrity of the stable branch.

To create a semantic history of the development process, labels will be used extensively throughout the course of the Rogue Reborn project. This will enable stakeholders to oversee the progress of the project more clearly and developers to gauge their own productivity with greater ease. Milestones will also be incorporated as a means of measuring the progress of the application against the goals of the stakeholders. These milestones will also be used internally within the Rogue++ team to coordinate feature implementation or debug deadlines. If done correctly, this system will allow for more efficient communication between

the involved parties and improve the visibility of the entire development cycle.

5 Proof of Concept Demonstration Plan

To demonstrate the feasibity of the project, a proof of concept will be developed. The PoC will demonstrate the following features:

- Basic dungeon generation, including rooms, corridors, and placement of gold, items, monsters, and traps
- Line of sight and pathfinding implementation
- Non-functional items and traps implemented
- Minimum viable monster AI
- Basic movement and very simple environmental interaction (picking up items, basic combat)

If there are no issues implementing the above features, it is assumed there are no fundamental flaws with the requirements or architectural design of the project. There are major features of the project that are touched by the above (advanced item manipulation, traps, hidden passageways, and complex monsters) but these have been left out of the PoC as too ambitious. So long as the underlying code is well architectured, these more complex features should flow out of the PoC foundation.

It is unlikely that a straight-forward implementation of the PoC features will prove unusually difficult. It may however be somewhat difficult to implement these features in sufficiently extensible manner that they can be reused without readjustment. The reverse-engineering of the algorithms for various features from the original source is likely to prove difficult. In addition integrating tests into the implementation may prove difficult as no team member has much experience with testing or testing frameworks. At this time the project is planned to only be developed for Linux systems, so portability is not expected to be an issue. The required dependencies have already been installed, with a test application running, so that should remain a non-issue going forward.

6 Technology

The technology behind the Rogue Reborn project was selected to facilitate a productive development process and a powerful user experience. At its very core, C++ will serve as the primary programming language for this application. This decision was heavily influenced by the superior performance benefits and community support behind the language, not to mention its prevalence in the professional game developer industry. Another factor that motivated the use

of C++ was its compatibility with *libtcod*: a lightweight graphics library that offers a simple interface to draw ASCII-style art and collect user input. For these reasons, the development team believes that developing a C++ project would yield the best experience from both a technical and practical perspective.

With respect to the environment of the technology, the Rogue++ team agreed to use Linux as the primary development platform (attempting to achieve cross-platform portability from the start could significantly hinder progress). Otherwise, every team member is free to use a text editor of their choice: imposing a constraint on an IDE could result in unnecessary complications and may interfere with productivity during the early coding stages. To gain confidence in the correctness and versatility of the code, the Boost Test framework will be heavily utilized to perform unit testing across the project. This library was selected on the merits of its superb documentation, simple approach to test creation, and robust assertion support. On a final note, the Rogue Reborn documentation will be generated using two tools. Specifically, all design documentation will be generated using LaTeX, while all source code documentation will be delegated to the Doxygen tool.

7 Coding Style

In any large-scale project, it is important that all members are on the same page. This is especially true for a software project, where every team member must be able to read and understand the code that another wrote. The Rogue++ project will be utilizing a modified version of Google's C++ Style Guide to format and organize the code. Google's style guides are a professional standard, utilized in every corner of the software industry. The team has decided to implement some changes to it, however, primarily due to the nature of the project and past proramming experience. As with any style guide, what is most important is that the code is persistent, not that it conforms to one system or another. Having said that, the team shall differ on two matters from Google's style guide:

- Opening and closing curly braces ({}) shall be on the same column. The reason for this is simple: curly braces are used to separate code blocks and to designate to whom or what a piece of code belongs. In a trivial case it makes no difference, but with multiple layers of nested for-loops, if-statements, and function definitions, opening and closing curly braces on new lines will yield code that is easier to read and easier to debug.

- Google's C++ Style Guide calls for inline comments describing classes, functions, methods, and file contents. While this is a noble goal, and is most definietly necessary, this job has been delegated elsewhere. As mentioned before, the team will be using the Doxygen tool for documenting code structure. Doxygen will encapsulate the design-documentation sphere, stepping into a fair portion of Google's style guide. The team will

still, of course, leave in-line comments explaining the functional aspects of the code.

8 Project Schedule

Over the course of a one-hour meeting, the Rogue++ team decided on an overarching timeline for the development of the project. The timeline consists of many parts, some overlapping with others. This timeline exists as a way of benchmarking the project's completeness; a guideline, not a rulebook. It will be wise to refer back to this schedule at least once a week, to update the project status and assess any complications that could arise. The schedule is of the form of a Gantt Chart, soon to be made available. For now, listed below are the deadlines set forth:

Require Docs	Sep 26	Oct 7
Milestone 1 (Rev -1)	Oct 10	Oct 14 (Break)
Explore Tech	Oct 17	Oct 21
BOOST + Test Plan	Oct 24	Oct 28
Design Doc	Oct 31	Nov 11
Rev 0 (Milestone 2)	Oct 21	Nov 19
Rev 1 fixes (Milestone 3)	Nov 18	Nov 30
Design Doc Final	Nov 26	Dec 7

9 Project Review