# chizuru4rogue: Deep Reinforcement Learning for Dungeon Crawling

Project Proposal

Dylan G. Drescher

November 2022

## 1 Problem Description

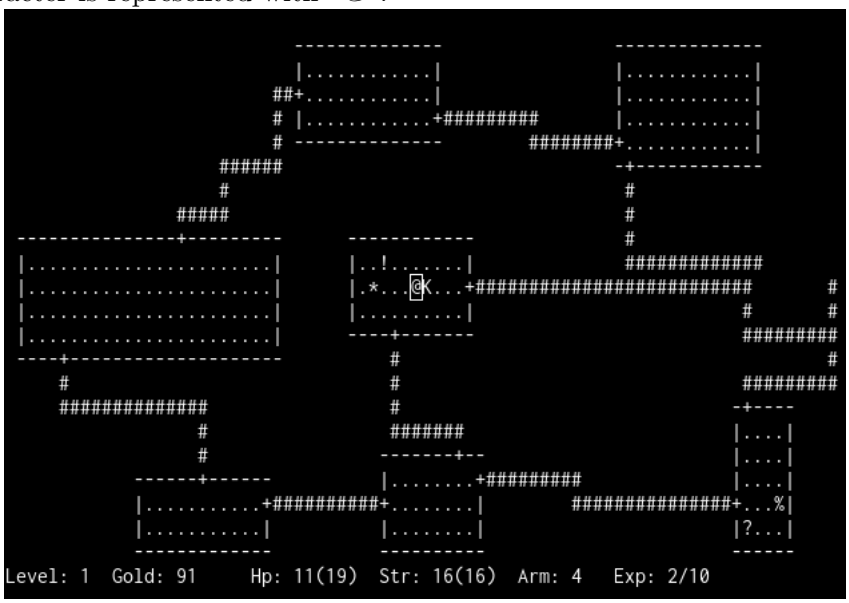
### 1.1 Introduction

Rogue is a computer game that belongs to a genre called "roguelikes". Roguelikes are a genre of computer game characterised by challenging, turn based dungeon crawling gameplay, procedurally generated levels and permanent character death. They are a fascinating domain to apply reinforcement learning methods to due to the amount of strategies and gameplay styles that roguelike games allow. Additionally, an agent playing a roguelike game would only need to perform one action per turn, and often times turns in these games are resolved near-instantly, allowing for efficient simulation and training.

The aim of chizuru4rogue is to apply deep reinforcement learning methods for an agent to learn, survive and thrive within Rogue. Reinforcement learning methods are suitable for this problem domain because they work well with games, due to the nature of games granting rewards for skilled play. The hope is that by the end of project development, chizuru4rogue will be able to play the game to a level where it can win the game, however further investigation into which methods are most suitable to accomplish this task will be performed during research and design of the agent, starting with researching methods used in other agents that play games such as AlphaGo.

Should the project prove successful, it can be used as a foundation for creating agents to play more complex roguelike games, such as NetHack or Cogmind, improving upon the program to allow the agent to adapt to these more complex environments.

Figure 1: A screenshot of an example Rogue game. Items, monsters and other map information are represented with a unique letter. Here, gold is represented with "*", a kestrel enemy is represented with "K" and the player character is represented with "@".




## 1.2 Problem Environment

The environment the agent will interact with is the computer game Rogue, a 1980 terminal-based dungeon crawler and the namesake of the "roguelike" genre, inspiring games similar to Rogue such as NetHack and Angband. I am using Rogue as an environment as Rogue is a simpler game than its successors, while still being challenging enough to be an interesting problem to work with.

In Rogue, the player's objective is to navigate their way down numerous levels of a procedurally generated dungeon in order to reach the 26th level and collect the "Amulet of Yendor".

The dungeon's layout is initially unknown, being revealed to the player as they explore. As the player explores the dungeon, the player will find items to loot and monsters to fight. If the player is killed, the game ends and the player must start from dungeon level 1. The player is given a score once the game ends.

## 1.3 Existing Systems

There have been several attempts at creating an agent to play Rogue. One of the most prolific is Rog-O-Matic, an expert system developed in 1983 in Carnegie Mellon University that was observed to play as well as human

players (Mauldin et al. 1983, p.8). This system makes no use of any reinforcement learning methods, but provides a good yardstick for measuring how the agent should perform once it has been trained to play Rogue effectively.

Another, more recent attempt is Rogueinabox, a framework for Rogue agents where a Deep Q-Learning neural network was applied (Asperti et al. 2017, p.149). Rogueinabox in their experiment lacked several core gameplay features that their agent could have considered such as combat and item usage, instead focusing on exploration without fighting enemies or collecting items.

While this is fine for testing an agent interacting within Rogue, the agent would need to be able to interact with the environment fully if the agent is to win the game, since if the agent never engages in combat with enemies, or collects better equipment, the agent will not get stronger, and will be easily killed by the later levels' enemies.

# 2 Objectives

1. The first objective is to create a module that acts as an interface between Rogue and chizuru4rogue. It will take actions that the agent wishes to perform, runs that action in Rogue, takes the resulting consequence and encapsulate it in a way that chizuru4rogue can interpret.

2. The second objective is to develop chizuru4rogue proper. During development, bug fixing will be common. The end goal of this stage is to get the agent created, running and learning.

3. Once chizuru4rogue is in a working state, the third objective is to run chizuru4rogue over many episodes and measure its performance as it plays in terms of its final score.

4. The fourth objective is to evaluate the data generated by the agent, summarise what has happened within the report, consider any improvements that could have been made during development and present other improvements for related future projects.

# 3 Requirements Specification

## 3.1 Interface Requirements

1. The interface must allow the agent to send an input to Rogue.

2. The interface must be able to interpret Rogue's output.

3. The interface must encapsulate Rogue's output so that it can be interpreted by the agent.

4. The interface must tell the agent what actions it can currently take.

### 3.2 Agent Requirements

1. The agent must learn from its gameplay experiences.
2. The agent must be able to interact with the game through the interface.
3. The agent must perform some action during its turn.
4. The agent must perform an action out of the action set that the interface gives.
5. The agent must take the Rogue output and use it to learn from what has happened.
6. The agent should improve its skill over many gameplay episodes.
7. The agent could utilise LSTM

### 3.3 Project Requirements

1. The entire project must be complete and in a working state by the 1st May 2023.
2. The project must have a Literature, Technology and Data Survey submitted by the 2nd December 2022.
3. The project must have a Demonstration of Progress submitted by the 20th February 2023.
4. The project should have a minimum viable product ready by the 20th February 2023.

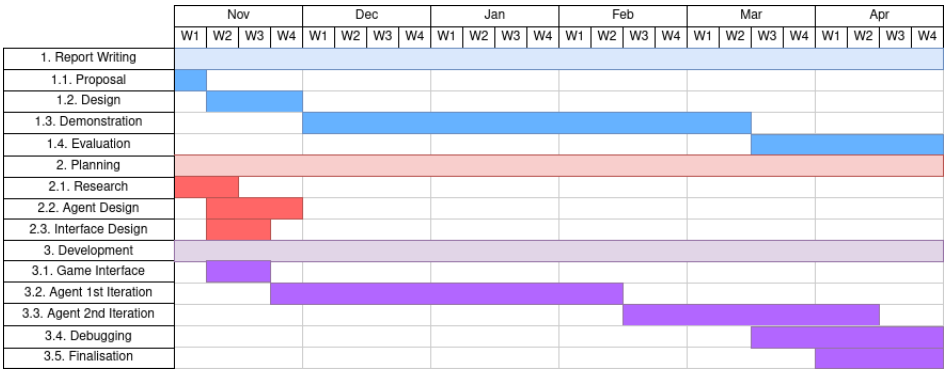
## 4 System Plan

### 4.1 Plan Overview

While Rogue is written in C, chizuru4rogue will be written in Python. The agent will not need to interact with the code of the game, it will only be able to use the same information that the game would give a human player. Python was chosen due to the large amount of machine-learning based libraries available for the language, such as TensorFlow. An interface for Rogue will be created before work on the actual agent begins, also written in Python. This interface will allow chizuru4rogue to interact with Rogue, and it will send information back to the agent about the current state of the game, as well as any results due to the agent's actions.

The entire project's code will be managed on a Git repository, hosted on the online service Sourcehut.

### 4.2 Gantt Chart




### 4.3 Required Resources

Many reinforcement learning-based articles and books are available online, either for free or accessible through the university library, therefore no special provisions must be met in order to accomplish research and design.

Running reinforcement learning programs require lots of compute resources. To accomplish this project, I will need to contact the HPC departmental champion, James Davenport, to be granted access the University's Balena supercomputing environments.

## References

Asperti, A., De Pieri, C. & Pedrini, G. (2017), 'Rogueinabox: An environment for roguelike learning', *International Journal of Computers* **2**, 146–154.

Mauldin, M., Jacobson, G., Appel, A. & Hamey, L. (1983), 'Rog-o-matic: A belligerent expert system'.