Open-World Adventure Game

MSAI 371 Knowledge Representation and Reasoning Project Proposal

Liqian Ma, Qingwei Lan, Wentao Yao

February 8, 2022

1 Introduction

We are trying to build an open-world adventure game where a role-played hero can navigate and explore. This game is loosely inspired by the existing game Dungeons and Dragons.

The world we build will be a rule-based world and will try to include complex systems such as weather, physics, day-night cycles, etc. The hero travels through the world by choosing its next move and the world's reasoning engine will determine whether the move is a valid one or not. We will also build in randomization by using a dice to determine the choices of the hero's next move.

Knowledge Representation and Reasoning

The rules and facts will need to be engineered as knowledge and added to our knowledge base. This part requires detailed knowledge representation.

The inference engine will determine, based on the rules and facts, what moves are permitted and what will happen once the hero makes the move. It will use the move that the hero makes (newly added fact) to infer a new world state that sets the hero up for the next move.

2 Requirements and Questions

The game will be built in the logic programming language Prolog [1]. This language contains an inference engine that will allow us to derive rules based on existing rules and facts and is best suited for our open-world game.

The following lists some requirements and questions:

- How do we build a dice in Prolog?
- How do we model random world moves based on dice output, i.e. how do we decide the hero's actions based on the dice's outcome.
- Can we build random worlds based on a certain set of rules?
- How complex can our rules be? In other words, how large can the world be?
- Should the world contain a map, or can the hero move anyway it wants?
- Does the game have an end state? How will we determine the end state?

3 Detailed Game Design

3.1 Character

- A character has Hit Points (HP) and Action Points (AP). The character will die when HP goes to zero. The character will need to have a rest when AP goes to zero.
- A character has attributes like Strength, Dexterity, Constitution, Intelligence, Wisdom, Charisma, etc. These attributes will impact the power of the character's skill and the action sequence. Some skills may require a certain level of attributes.

3.2 World

- Consists of a bounded map of custom length and width.
- The map is generated randomly when starting the game.
- The goal is to find the required object (and not die during the process). Once the required object is found, the game ends.

3.3 Gameplay

- Clock-based game.
  - Hero and NPC's moves are based on the clock. The character's move has a cost of ticks. For example, moving forward cost one tick, but a powerful spell may cost three ticks.
  - The day-night cycles are based on the clock. For example, a day has 24 ticks and 12 ticks in the daytime.
  - Weather also changes on ticks, but the following weather and how many ticks it holds depends on the dice. For example, the current weather has six ticks when we get six by dice.

4 Plan and Milestones

4.1 High Level Requirements

- Design the rules of the world (i.e. the world itself)
- Design the hero (characteristics, powers, etc.)
- Design the gameplay structure
- Design how the game ends
- Prolog system design: how do we model the dice and the moves of a player?

4.2 Detailed Weekly Plan

Week 1:

- Design the basic rules and systems of the world
  - Rules for moving
  - Day-night cycle system?
- Design how the game ends (game objectives)

Week 2:

- Build basic static world rules in Prolog
- Allow basic hero to move around within the static world

Week 3:

- Ensure the game ends properly when objective is reached

Week 4:

- Construct demo and write report paper
- Build a more complex world
  - More complex rules
  - Hero capabilities and abilities
  - Subgoals within the adventure
  - Make the world change dynamically

5 Risks

The biggest risk of this project is that we are not familiar with the Prolog language and are not familiar with its capabilities. Fortunately, the manual is rather descriptive and the language is open-source and can be engineered to our own purposes.

Furthermore, modeling an open-world is not an easy task. We may need to build a simple physical reasoning engine, day-night cycles, weather, etc. to make the gameplay more interesting and more "life-like". We will need to scope this down properly.

Designing rules that work together and interact with one another requires very careful design. This is an interesting yet daunting task that we plan to tackle early on before we start building. We will start out with simple rules and add more complex ones later into the project.

6 First Actions

Our first actions are two-fold:

- Evaluate the technical feasibility of this project and scope the project accordingly. For example, if modeling a dice is not possible, then we can make moves deterministic and focus more on designing the open-world rules.
- Designing the open-world rules. We need to figure out how complicated it should be and what components (physical reasoning engine, day-night cycles, weather, etc) we should include.
- Building a basic world and hero that allows moving throughout.

References

[1] Jan Wielemaker, Tom Schrijvers, Markus Triska, and Torbjörn Lager. Swi-prolog, 2010.