Project Proposal

Samuel Xiang

0.1 Introduction

The language I would like to design would solve the ”problem” of designing turn-based RPG games on a computer. This language would be able to generate a playable game with an interactable world and combat between characters, while giving the programmer plenty of flexibility to add to the existing structure as they please. In essence, it would provide the programmer with an outlet that allows them to express complex creative ideas in simple ways.

There are a vast selection of unique turn-based RPG's in the real world, but the genre has rules and restrictions of form that are almost impossible to break. As such, expressing the basic tenets of any turn-based RPG can be boiled down to several patterns, which mean that a formal grammar is perfectly capable of encapsulating these tenets. However, building these core ideas over and over again is redundant and time consuming, so a programming language should be a fun way to create these games with minimal labor.

0.2 Design Principles

This language should express lengthy ideas and systems of turn-based RPG games in simple English. From the technical perspective, this program should generate a playable game to the users specifications within the scope of the language. By playable, I mean a game that an outsider user interact with and complete without knowledge of the inner workings - like an actual video game. Because these ideas are expressable in simple English, I would like to create a readable program without lots of weird characters that almost anyone could access and easily intuit. Because the creation of a game involves the creation of a fictional world, the program should almost read as a story or encyclopedia of this world.

0.3 Examples

0.4 Language Concepts

The language as I envision it right now doesn't actually have many moving parts - the set of primitives I want combine nicely, and the most of the functionality/computation is hidden from the programmer. As such, the programmer simply needs ideas to fill out the game. Creating the map, executing combat, etc. are all things that the programmer doesn't consider - rather, they create the guidelines that the functionality of the language operates within. The most important things the programmer has to understand are how rooms are connected and how stats interact in and outside of combat.

0.5 Syntax/Grammar

```
<paragraph> ::= <sentence>
              | <sentence><paragraph>
<sentence>  ::= <character>
<character> ::= There is a <type> named <name> with stats: <stats> and abilities:
    <abilities>.
<type>      ::= playable
              | nonplayable
<name>      ::= <string>
<string>    ::= *a string in F#*
<stats>     ::= hp = <int>, mp = <int>, atk = <int>, def = <int>, matk = <int>,
    mdef = <int>, spd = <int>
<int>       ::= *an int in F#*
<abilities> ::= <ability>
              | <ability>, <abilities>
<ability>   ::= <name> has effect <effect>
```

```
<effect>    ::= <string>
```

Syntax	Abstract Syntax	Type	Prec/Assoc	Meaning
<Sentence>(one or more)	Paragraph of Sentence list	Sentence list	n/a	Paragraph is a list of one or more sentences.
There is <type>character named <name>with stats: <stats>and abilities: <abilities>.	Character of record	Sentence	n/a	Character of record takes in four types and combines them in a record. The primitive types are Type, Name, Stats, Abilities. If these types are not present, an error will be thrown.
<type>	Type of string	string	n/a	<type>is a primitive, represented by a standard F# string.
<name>	Name of string	string	n/a	<name>is a primitive, represented by a standard F# string.
hp = <int>, mp = <int>, atk = <int>, def = <int>, matk = <int>, mdef = <int>, spd = <int>	Stats of record	record	n/a	<stats>is a primitive, represented by a record of ints.
<Ability>(z eroor more)	Abilities of Ability list	Ability list	n/a	Abilities is a list of zero or more sentences.
ability <name>has effect <effect>	Ability of record	record	n/a	Ability is a record of Name and Effect. If these types are not present, an error will be thrown.
<effect>	Effect of string	string	n/a	<effect>is a primitive, represented by a standard F# string.

0.6 Semantics