The Gridworld Language and its Applications

By Andrew Phan, Kevin Weng, Loren Weng, and Zikai Lin
Uni: ap3243, kw2538, lw2504, zl2442

September 30, 2015

Professor Edwards
PLT Proposal
Columbia University

1 Introduction

Gridworld is a language designed for those who want to create their own RPG (role-playing games). It will try to provide simple, readable functions and syntax to build the game so that users with no prior knowledge of programming can easily learn how to design and implement an RPG.

The gridworld language can help users to design two significant parts of RPG, both the story and combat. Using the template code, users just need to input the characters name, the dialog and the narrations for the main story. After that, some options can be set between the different stage of the story for the players to choose the storyline. More importantly, users can design some combats for some special plots. The combat design is different from the story design and Gridworld will provide enough elements for the users to make their own combat system.

Not only does Gridworld focus on teaching the basic principles of game design, coding fundamentals, but it is also an open sandbox, giving freedom to the content creator in order to seamlessly unleash their creativity and logical thinking into a RPG.

2 Applications

The main motivation of this program is to allow an easier way to construct the world of an RPG or other similar games. The language would facilitate users to define the rules of their world and to reinforce these rules as the game progresses and the state of the world changes.

Additionally, the language would simplify the process of keeping track of the current state of the world in a grid-based format and to be able to update the state as needed. This is also true of the objects in the world as the language provides means to attach attributes to the objects and define their interaction with the world. Also, as randomness is prevalent in many RPGs, the language will offer functions to facilitate this with less difficulty. These functionalities allow for users to create their own worlds and characters in a straightforward manner.

3 Language Syntax Overview

The idea of Gridworld is to allow the user to easily create a working RPG game environment, in which, the designer is able to model a world to their own specifications. Not only can they apply our built-in tools but they can also incorporate their own user-defined functions. Our language is able to provide a simple and effective way of combining the elements of game design, story-telling, and object oriented programming. In the following sections below are what we as a group believe are the crucial elements in developing a free-form RPG.

3.1 Data Types

Table 1: Various data types with their in-game applications

Data Types		
Name	Description	Examples
Boolean	True/False	If object isLiving=True, then it will have hitpoints, damage, image, and location.
Integers	Numerical.	Move 2 units to the right on map or how many players in RPG?
Strings	Prints text output.	Can hold name of characters, monsters, and print storyline.
Floats	Numerical.	Used to calculate attacks and hitpoints.
Object	Contains location, image, attribute list.	Player starts at default location, image=F for fighter, with Godlike attribute list.
Image on Map	Represents 1 character on a map.	Easily distinguish and track what is what on the game map.
Attribute	User-defined value attached to an object.	Set Attribute=Godlike, if object is a character, then it will not lose any hitpoints.

3.2 Functions

Table 2: Important RPG Functions

Game Functions		
Name	Action	Example
random()	Randomly generate environment.	Makes the virtual world with random number of monsters, trees, and rocks spawn in various locations
attack()	Attack a character/monster.	Attack(player1, player2) - Player1 attacks Player2.
move()	Move to up, down, left, or right on the map.	Move(left, player1) - Player1 moves to the left on the map.
checkHealth()	Determines how much health a player has.	checkHealth(player1) - first verify whether or not it is a living object, then store and print health after each attack.
save()	Saves the state of the entire game world.	save(rpgland1.sav) - saves the file to rpgland1.sav so that the game can be continued later.
load()	Loads the state of the entire game world.	load(rpgland1.sav) - loads the file rpgland1.sav and checks if .sav file exists. If it exists, load the game from where it left off.

3.3 Loops and Conditionals

Table 3: Loops and Conditionals with Examples

Loops and Conditionals		
Name	Action	Example
for loop	Repeat code for a known amount of times.	int numberofPlayers=10 for i=1 to numberofPlayers print (checkHitpoints(numberofPlayers[i])) Can print the health of all players in the game.
while loop	Continue to repeat code for an unknown amount of time.	while (!players) print(there are no players initialized in RPG game) promptPlayers() While there are no players in the game, it can ask if you would like to add a number of players.

if/else	Conditional statement that executes code if something is true, else if something is false.	attack(player2, player1)
		if (Player 1(hitpoints) == 0)
		print(cannot kill Player 1 because Player 1 is dead)
		else
		subtractHitpoints()
		printHitpoints()
		// Statement checks to see if a player is alive or not and then subtracts the number of hitpoints from an attack. It then prints the new hitpoints.

3.4 Operators

Table 4: Useful Operators to Implement, [Wikipedia, 2005].

Operators	
Name	Action
+, -, *, /	Add, Subtract, Multiply, and Divide
!=, ==, <, <=, >, >=	Not equal to, Equal to, less than, less than or equal to, greater than and greater than or equal to
=, +=, -=	Assignment, Addition Assignment, and Subtraction Assignment
&&, ‖, !	Logical AND, logical OR, and logical negation NOT.

3.5 Sample Code / Pseudocode

```
1 main
2 create world size 10x10
3
4 new object rock has attributes (image =    X    , noStack)
5 loop i from 0 to 10{
6   loop j from 0 to 10
7     if (i==0 OR i==10 OR j==0 OR j==10)
8       create rock r at i,j
```

```
}
new object player has attributes (canmove,hp = 10, image =    0    ,speed
    =10)
create player p at 1,1
new object fist has attributes (inanimate,weapon,damage =1d4)
new object sword has attributes (inanimate, weapon, damage = 2d4)
p.give fist,sword
new object dog has attribute (cannotmove,hp=1, image =    D   )
create dog d1 at 4,4
create dog d2 at 4,5
dog makePrompt bark(say    B a r k   )
start();
//creates ascii map of world and populates with movable characters

function attack(object)
if object.attribute.contains (damage){
  roll(object.damage) //roll should be a built in function that parses and
      rolls x amount of y-faced diced, denoted by xdy
}
```

Listing 1: samplecode.txt

References

Wikipedia. List of logic symbols, August 2005. URL https://en.wikipedia.org/wiki/List_of_logic_symbols.