GOLD (Game Oriented Language for DnD) Language - Project Proposal COMSW4115

Language Guru: Timothy E. Chung (tec2123)
System Architect: Aidan Rivera (ar3441)
Manager: Zeke Reyna (eer2138)
Tester: Dennis Guzman (drg2156)

1. Introduction and Motivation:

Our team decided that the creation of a game language would be the most enjoyable for all involved, and that to reasonably do such, a turn-based interactive game on a 2 dimensional matrix would be doable. Essentially, our language contains common constructs in games that allow developers to quickly prototype their game's environment. This way, they can focus on assembling the business logic of the game rather than worrying about writing a backend that keeps track of the states of the game like Characters and location of the Characters.

2. Language Overview:

Primitive Data Types

Name	Description
int	Regular integer that is 64bit
bool	Boolean, true/false
string	String, “”
amount	(ie. 2gold) integer followed by string
typeStruct	(ie. Item, Character) Definition with fields and their types; all our default typeStructs are denoted by capitalized letter and camel casing like in “Item” and “Character”
null	Null is the uninitiated state

Supported Data Types

Name	Description
[]	Array of any primitive.

Basic Keywords and default typeStruct (typeStruct is capitalized while actions start with lowercase)

Name	Description
dim(x int,y int)	Action: Set the dimension of the world in the game randomized if not specified.
Item	Defines an item (refer to example code for struct fields)
Character	Declares a character (refer to example code for struct fields)
Enemy	Declare an enemy (refer to example code for struct fields)
Barrier	Barrier in map (refer to example code for struct fields)
func	func defines a function, it can be used with a name like func helloWorld() or as anonymous func() when used with **let**
let	let allows user to bind a custom func to a variable name. i.e. *let myFunc = func(arg1 int) {}*

Declaration

Name	Description
Type()	Creates an object of Type that is a typeStruct
Type{(),()}	Create an array of Type is a typeStruct

Operators

Name	Description
-,+,*,/,-=,+=	Operator and shorthand operators for multiplication, subtraction, addition, and division.
>, <, ==, !=	Boolean logical operator
=	Assignment operator

3. Examples and Sample Program

Our goal is to create a language that allows developers to quickly build a gaming backend and the game's business logic using abstractions most games have like Characters and Barriers. Our language aims to provide a convention and framework to abstract all the recurring logic and constructs in most common games while leaving space for customization.

```
let add2health = func(myCharacter typeStruct) {
    if (myCharcter.health < myCharacter.maxHealth - 2) {
        myCharacter.health += 2;
    } else {
        myCharacter.health = myCharcter.maxHealth;
    }

}

let add8health = func(myCharacter typeStruct) {
    if (myCharcter.health < myCharacter.maxHealth - 8) {
        my_character.health += 8;
    } else {
        myCharacter.health = myCharcter.maxHealth;
    }
}
// Program must have dim specified
dim(7, 7); //Back end prevents out of bounds movement, spot declaration

/* Item defined by tuple of name, type, characteristic, x-position,
y-position ...*/
Item("sword", "weapon", add2health, Character, 3, 4);
Item{("green_potion", "potion", add8health, Character, 3, 3),
     ("hummus", "potion", 2, 1, 6)};

/* Char defined by tuple of name, type, location, health, other stats... */
Character("Zekius Penius", "Warrior", 100, 0, 0);
Character{("Aidario", "Mage", 80, 0, 1),
          ("Dennis le Menace", "Rogue", 90, 1, 0)};

/*Barrier defined by tuple of type, starting spot, ending spot CANNOT BE
DIAGONAL*/
Barrier("wall", 4, 6, 7, 6);

/* Enemy defined by tuple of  name, boss?, location, health, other stats,
reward */
Enemy{("Goblin", false, 2, 4, 30, 20g),
      ("Kilgore", true, 5, 6, 200, 2green_potion)};

/* ------------ (Backend definitions: defaults, types, etc.) ----*/
```

```
Character {
	name = string;
	type = string;
	location = int;
	health = int;
	otherStat = int;
}

Enemy {
	name = string;
	isBoss = bool;
	location = int;
	health = int;
	otherStat = int;
	reward = int // this is the worth of the Enemy when captured
}
Barrier {
	type = string;
	x1 = int; /* if not specified or default, randomized */
	y1 = int;
	x2 = int;
	y2 = int;
	permeability = bool false; // this means default false
}

Item { /*can be held in inventory*/
	name = string;
	type = string;
	effect = func; // the effect func must accept a typeStruct that is
the same type as target for target will be used as argument
	target = typeStruct;
	x = int;
	y = int;
}
```