Group 4: “Scripting Language Model for Designing Worlds” Project Specs

Objective:

Create a language that will generate a simple world

Key Features:

Scripting Language:

- Top of the file will have a “World” function containing a list of World commands
    - World commands will be similar to Minecraft commands so that they are easy to understand for new programmers
- The “Agents” function will define all the agents in the world. Each agent needs at least a name and position.
- World() will be called first, then Agents() will be called. Finally, the world will be generated.

Sample syntax:

```
fn fillWithGrass(x1, y1, x2, y2):
    fill(x1, y1, x2, y2, material:grass)

fn World():
    # World commands go here
    fill(10, 10, 20, 100, material:stone)
    winFlag(60, 120)

    # defining variables
    grassStartX = 50
    grassStartY = 50

    # math operations are as you expect
    grassEndX = grassStartX + 50
    grassEndY = grassStartY * 2

    # calling a function
    fillWithGrass(grassStartX, grassEndX, grassStartY, grassEndY)

```

```
fn Agents():
    # Agents & associated data goes here
    player = {
        name = "Jayson",
        pos=[25, 125], # spaces are optional after equals sign
        health = 100
        age = 22,
        wallet = 100.55, # trailing comma is allowed
    },
    enemy = {
        name = "Goblin",
        pos = [50, 150],
        health = 10,
    }
```

World Commands section

- Allow users to create / modify / remove the world and its objects
- Commands will be self-named
- Ex: fill(x1, y1, x2, y2, material:materialName)
    - Fills in the grid cells from (x1, y1) to (x2, y2) with material materialName
- winFlag(x, y)
    - Sets the position of the “win condition” flag

Agents section

- Syntax: agentName = { attribute1 = value1, …, attributeN = valueN }
    - Attribute ex: {health = 100, wallet = $2.97}
- Agents & attributes are comma separated
- Positions will have the syntax: pos = [x, y]

World:

Background

- Will store the background image of the world (if needed)

Cells

- The “things” in the world
- Properties:
    - Name, image, size, location, interactable
    - Properties will be defined by the scripting language
- Will have the ability to store custom images
- Cells will be intractable with agents

Weather

- Allows / prevents actions taken by agents based on the weather
- Ex. When raining, allows user to dig items up from the ground
- Ex. When raining, prevents agents from having a fire

Actions

- How the agent moves / interacts with the world
- Up, down, left, right
- Push, pull
- Jump, duck, dialogue, reading
- Picking up / dropping inventory

World changes

- Defined by the scripting language
- Ex. If an enemy is defeated, then the dungeon exit opens

Other functionality

- Usage information Help functions (i.e. how many agents are currently in the grid)
- Saving and loading via streams
- Unit tests for functions and classes

Features that other modules will need to provide:

- Interface modules will need to provide and display custom images for cells
- Agent modules will need to tell us what action to take

Why this project is challenging:

- We will need to create our own language that is versatile and easy to understand
- We will need to figure out a way to save and load world objects if multiple agents can play at the same time
- Objects that are picked up by a player will still be available for other players
- Dynamic interactions between the environment and objects
- Communicating with other teams about what we want to do