CSC241 Project Description

Daniel R. Schlegel
Department of Computer Science
SUNY Oswego

January 14, 2018

This project involves constructing a virtual world simulation which the user will interact with via a string-based command line interface.[1]

Premise

In the simulation you will create, the player acts as a ghoul who is haunting a house. The goal is to scare everyone else out of the house before a timer expires. Being a ghoul, the player is invisible to the house occupants, but may interact with items in rooms – shaking, throwing, and possessing them. These actions cause other characters in the room to get scared and either run out of the room or out of the house. The player continues this process until the house is empty or the timer expires.

Description of the World

The virtual world consists of a series of rooms which are joined together with doors forming a house. A room can have at most 4 doors, one to each of the north, east, south, and west. The player may move from their current room to an adjacent room using the `move` command. A room may contain items and characters. Items always remain in the same room, and rooms and doors always stay in the same place. Characters, on the other hand, can move between rooms (for example, enter a room or leave it). A character leaving a room always means that the character is entering another room – there are no hallways, and a character can only be in one room at a time. There are three types of characters: the player character (PC), and two types of non-player character (NPCs) – adults and children. There is only one player character. There can be multiple NPCs.

The house will contain at most 5 NPCs, therefore a room may contain at most 6 characters – all of the NPCs plus the PC. Rooms, items, and characters all have unique names, as well as text descriptions. Each character also has access to a `look` command, which lets it see the name and description of the room it is in, as well as the names and descriptions of the characters and items in the room. Note that the player is invisible to NPCs, so while it can see itself, it cannot be seen by the NPCs.

Each item in the player's current room may be manipulated in one or more ways by the player. The three ways to manipulate an item are: shake the item; posses the item causing it to float around the room; throw the item across the room, breaking it. Not all items can have all manipulations done to them. Items are manipulated using `shake`, `possess`, and `throw` commands. Once an item is broken it can no longer be manipulated and is noted as such in the output of the `look` command.

Each NPC has a "scare level" parameter, initially at 0, indicating the NPC is not scared. When the NPC's scare level grows past 50, the NPC runs to another room; when it passes 100 the NPC runs out of the house. The NPCs scare level is increased based on how the player haunts the items *in the NPCs current room*. When a character looks around they room, the message printed should include the scare level of the occupants.

Manipulation of items by the player increases the scare level of each adult NPC in the room as follows: shaking adds a random value between 5 and 15; possessing adds a random value between 10 and 25; throwing adds a random value between 20 and 40. Children have their values increased at 1.5x the rate of adults.

Each time an NPC becomes more scared it shrieks, printing a message to the screen. When an NPC leaves a room because its scare level has passed 50, it shrieks three times. If the NPC enters a room with broken items it

[1] This project is loosely based on both the Sega Genesis game Haunting Starring Polterguy and previous CSC241 projects written by Alex Pantaleev and Lynna Cekova.

cleans them up (removing them from the item list). When the NPC leaves the house because it is scared, it gives one final (extra loud) shriek.

The player must complete scaring all of the NPCs from the house before the play timer reaches zero. The timer begins at 1 minute. If the timer expires with NPCs still in the house the game ends – a message is printed to the screen and the program exits. Every time an NPC is forced to leave the house the remaining time is increases by 30 seconds.

Input, Output, and Game Interface

The primary game interface is a command line, where the user can input String commands, and where text messages are shown to the user. Graphical components possibly will be added for portions of the interface.

Upon program startup, the user should be asked to provide an input file with information about the world (rooms, characters, and items). Your program should parse this file, and, based on the information in this file, create the rooms and the characters that are in them (including the player). A sample input file will be provided to you, together with the description of the file format. You can use this file in writing and testing your project (or you can make your own input files if you so wish), but keep in mind that other input files will be used upon evaluation of the projects. (This means you do need to parse the file based on the file format; hard-coding input values is not acceptable.)

Next, the user should be presented with a prompt to issue her/his String commands. When read by your program, these String commands should be parsed and used to make the player character perform actions or order other characters to perform actions. Output will be returned to the user (telling what is happening as a result of the actions), followed by another prompt for a user command. The user will be presented with prompts for commands until the user types `exit`, or until the game ends. If the game ends, the user receives a message. (Sample messages are "Congratulations, you cleared the house and won the game!" or "Shame on you! You lose.")

A Note on the Wording of Messages and Commands

The wording of the messages themselves is of no importance, as long as all the information of what is happening is presented. In fact, you are encouraged to make adjustments and personalizations as this often makes the development process more fun. The wording of the commands is important because they have to be processed computationally, but again feel free to customize as long as you are consistent and the functionality remains. It's worth taking the time to decide in advance what your set of commands will be. Define the format of your own String commands and show them to the user, together with a description of what they do, any time the user types `help` at the prompt.