Anything But The Machine

A surrealist text based adventure game

Jacob Willard

For my project I will be creating a text-based adventure game titled *Anything But The Machine*. The game will be of the puzzle/adventure genre and will employ elements of surrealist media and art in its story and world design.

Structure:

In order to keep things neat and tidy, the structure of the game will exist as a package of python files I will create that will contain all of the essential classes and functions that the core game file will reference. In the main file, I will program the main menu and NES-style password progress system. Every time you quit the game, you are given a password that will let you return to the level you were on. In the game package there will be a file containing all of the classes, a file keeping all of the scripted levels as functions to be called by the main file, and a file containing all of the large texts strings (exposition, long dialogue, etc).

Classes:

- item() -- For objects that will exist within an “inventory” library. This class has a method that allows it to interact with objects in the “world” library.
- prop() -- For objects that will exist within a “world” library. You can interact with these using items.
- npc() -- “Non-playable-characters”. Exist within the “world” library. These have a talk() method that you can call to start a conversation sequence.
- body() -- There will be times when you can interact with things directly, so a level might include your hand or your feet as a useable object. I don’t want these to show up in your inventory so they get their own class, but this will just be a subclass of the item class.
- space() -- This class automatically organizes all items, props, and npcs into their own respective libraries. This streamlines a lot of code. This is where “world” and “inventory” libraries are put together.
- link() -- This class is meant to establish connections between two objects in the game. When you interact with something using an item, the code checks if there exists a link between those two objects.
- game_loop() -- This class streamlines the actual mechanics of the game. Every level will exist as a sequence of while loops, each having their own set of conditions before the loop breaks and the code moves on to the next one. Each loop will just be a separate instance of the game_loop.loop() method. The text-based control mechanic will also exist as a method of this class and will be called at the beginning of every run through the loop.

Control:

This game will be text-based, which means that everything you know about what is happening in the game is given to you via printed text and all the control you have over what happens in the game is controlled via text input. For each level, the game will exist within a while loop where you will be prompted to give commands. Your command options are “look”, “inspect”, “interact”, “take”, “drop”, and “talk”. When you input any of these commands, the game will then prompt you to specify an object if the command is object specific. The code will then check if there exists a link between those two objects and then returns a string that may or may not affect the level.

Objectives/Level Design:

The world that the player will be interacting with in this game will be of a dream-like/surrealist nature. Because of this, the objective of each level may not be explicitly stated or even make a lot of logical sense, but the pieces ought to fit together after the player does a little investigating and thinks about it a little. What I have in mind is similar to Alice in Wonderland, where you go from weird situation to weird situation and being able to progress through will take a lot of thinking outside of the box. I figure given the time we have to finish this project, the best way to make a genuinely interesting and challenging puzzle game would be to dive down the Surrealism rabbit hole. This means making puzzles that are a bit different than what

you would get from a logic game like Myst. Essentially, I will be creating “illogic” puzzles rather than logic puzzles. A good example of where this concept works well is in the game Antichamber. This is a 3D game that forces you to use illogical spacial reasoning to solve puzzles. Having played the game, I can confirm that puzzles that rely on “illogic” rather than logic can be just as satisfying to solve as logic puzzles, and sometimes they are actually more satisfying. They also have the convenient quality of being easier to design than they are to solve. It will be challenging enough for me to figure out how to balance “challenging and interesting” with “frustrating and stupid”, but I think I can actually accomplish that goal whereas a real logic puzzle game would be something that I probably couldn’t finish in time.