Erin Lewis

Professor Ranjodh Singh Dhaliwal

Critical Digital Studies

9 December 2021

Final Project Reflection

For my final project, I decided to create a text adventure called *Awaken*. Playing this game requires four files: adventure.R, which provides the code for the game; actions.md, a markdown file containing the descriptions of rooms and actions; map.csv, a matrix allowing for the structuring of different rooms; and, finally, objects.csv, a second matrix of the status of objects and characters within the game. All these files should be contained in a folder titled “adventure” on your Desktop.

To the best of my knowledge, the code must be run on R Studio. I apologize for this extra hurdle, on top of everything else. It is free to download [here](https://www.rstudio.com/products/rstudio/download/).

Code

One of the foremost challenges for me involved finding a way to actually create a working game. I’m not familiar with any of the coding languages typically used in the game creation. C++, Java Script, Python…all these languages seemed too complex for me to learn in such a short time. Even worse, when I downloaded Inform—the same, free software that allowed for the creation of Galatea—it was, for whatever reason, unable to run on my computer. For a while, this dilemma stopped me from even thinking about attempting this project.

Then, I had an epiphany. This semester, I have been taking a course called Biocomputing with Professor Stuart Jones. Part of the curriculum involves learning R, a coding language typically used for data analytics and statistical computing. The question was this: would it be possible to hijack R Studio’s functions to allow for a text-based adventure game? The answer to this question, thankfully for me, is yes. It was still a larger task than I could handle alone, which prompted me to re-purpose code from another game created by Peter Provos called *The Secret of Landusia*. If you glance through the assets, certain remnants from the previous game remain; there is a random crow in the objects.csv that never once appears in the game, for example. The rooms in map.csv are labelled incorrectly—you will unfortunately never visit the ramparts of a castle during your time in *Awaken*, nor will you prance through a meadow. There is a section of code titled “screwdriver in a tree”. Something else used to be in that tree. There also used to be a tree. None of these quirks hampered the function of the game, and so I let them be. I feel they are charming, hidden nods to the code’s origins. Almost like the easter eggs you might find in other video games.

Anyway.

In the spirit of Rita Raley, I feel the need to at least touch on the way in which the code’s depth and surface intermingle: To begin the game, you open the adventures.R file in R Studio. In the upper right corner of the screen, you’ll see a blue arrow pointing to the right along with the word “Source”. Clicking that button begins the initialization process, and the code reads the map, objects, and descriptions into memory (you’ll likely see the global environment in the upper right filling with values and functions) before it sets the starting variables for the player. In addition, it sets up many of the functions the game will be running in the background, unbeknownst to the player, such as the prose function, which allows for text to be output for the player to read, and the inventory function, which lists the objects that the player carries. From then on, the game is reactive to the player. The readline function will prompt the player for a typed input, which the program will then need to interpret. The state of the game (player, map, objects, and characters) is changed depending on said input, resulting in different text being output onto the console for the player to read.

Thus, the game loop begins. And it will continue to begin over and over again. Every time you die, the game resets back to the opening, and you’re to play through once more. To exit the game loop prematurely, you may also type “quit” into the readline, and the cycle will break.

Story

Now that I had figured out how exactly my game was to be created (the answer: in a very convoluted fashion), it was time to pin down what exactly my game would be about. Eventually, I decided to begin with a bit of a cliché: the player character wakes in a strange facility, with no idea how or why they are there. The only instructions the player is given on what to do come from a disembodied voice attempting to guide the player through the facility to an objective. Along the way, the player must decide whether they will follow the voice’s instructions.

In the game itself, I don’t ever plainly and directly explain to the player what exactly is going on. The game hints at certain things, but, ultimately, I wanted to give the player room to consider the text and come to their own conclusions. Half the pleasure of a text adventure game is the contemplation, the puzzling things through, the uncertainty. Looking at the finished product, though, I hope that the narrative isn’t too fractured, and that the clues aren’t too subtle. So that you might at least retroactively understand (I have no idea if you mean to play the game, and then read the reflection, or vice versa), I’ll explain some things here.

One of the things I try to hint at, but never say: the player character is not human. They are an android, blue blood runs through their veins. This is how they can die multiple times, but always come back to life: because their consciousness (though, not their full memory) is transferred into a new android body each loop. Another piece of the puzzle: the android that you can choose to either kill or spare is one of your bodies that went insane and began killing the scientists within the facility indiscriminately.

The rest of the story is intentionally malleable. Did the player character initially volunteer for these experiments? Is the voice real or just inside her head? You can determine the likely answers for yourself.

Mechanics

The player character can move in the compass directions (north, south, east west) and up and down. They can also: look, take or put down objects, use objects, and kill, according to user input.

The hardest mechanic to make operational was ironically the respawn function itself. The challenge lied in the fact that each time the player character goes through a cycle of the game, they will irreversibly change the conditions of the virtual world. The variables attached to the player character will change; for example, the moment you disobey the voice, your obedience variable drops to zero. Or, the character will interact with an object, taking it into their inventory, moving it, etc. As a result, simply sending the character back to the beginning of the loop is ineffective. They will wake in the next loop with the sword already in hand, or they will show up in Ms. Ulrich’s office, and the computer code will refuse to spawn into the player’s inventory. Some bugs of this nature may remain in the code, but I tried very hard to remove these defects. The respawn function is an important mechanics when it comes to this game’s narrative, after all.

Another tricky aspect to conquer was coding in multiple ways to end a loop. Eventually, I figured it out, using a dizzying number of if-else statements. The first way to end a loop is to lower your health stat to zero; this causes your character to die temporarily, before awakening once more at the beginning of a loop. The second way to end a loop is to follow what the voice says to the letter, even when it tells you to kill the android. After that, the voice decides you lack initiative, and takes over the game, leading your player character to an autopsy table to conduct further research. The loop resets. Closer to the end of the game, you finally escape the facility and make it outside. One last time, the voice tries to convince you to stay. In the third ending, you listen to the voice and go back inside the facility, after which the loop resets. The last ending involves ignoring the voice and leaving the facility (and eternal life) behind. With that, the loops can come to an end.

Aesthetics and Allegory

As discussed in *Respawn*, one of the elements that is most interesting about *Cave Adventures* was the intentional way in which it commented on its own digitality. I was particularly fascinated by the way that the game shifted to become, as Milburn writes, “a representation of its own computational structure” (61). My attempt to mimic this effect was not as dramatic as literally appearing inside the computer’s repository; I tried to focus on the aesthetics of the virtual world to mimic this effect. The facility is composed of whites and greys, much like the R Studio interface that the player will be interacting with for the duration of the game. The player character’s blood is blue, much like the readline function’s font color. This is meant to suggest a connection between the input and the characters themselves; the player’s input is what animates these characters—it is their very lifeblood. Another nod to the hardware and software of the game is found in the name Ms. Christina P Ulrich—or rather, Ms. CPU. Players find an essential access code within her office.

Another layer to *Cave Adventure* lies in allegory. The twisting caves within *Adventure* also represented the technical networks the late twentieth century, hidden from sight, inaccessible to those who stay on the surface. Similarly, Octodad used it’s fumblecore mechanics to impart a broader message; the intentionally difficult gameplay is meant to comment on the difficulties that come with being a closeted LGBTQ+ individual within a straight relationship. I hoped that my video game could contain room for a broader interpretation and application as well. Particularly, I feel that *Awaken* has interesting things to say about conformity and obedience.

When you obey, you’re guaranteed safety (and in this case, even eternal life). There’s no need to question what your purpose is, or where you’re meant to go, or what you’re meant to do. You follow the voice’s instructions. There is a degree of simplicity to it. On the other hand, you must kill parts of yourself to achieve this peace of mind. When your choices are not your own, it’s almost as if you are allowing someone else to live your life for you (see: ending three). On the other hand, freedom brings uncertainty and danger. There is a risk for destruction and death. You’re not always sure where to go, or what to do. But where there is greater risk, there is also greater reward: there is a possibility for life and joy and fun.