CECS 424 Assignment 1 - Specialty Languages with Inform 7

Due Date: 15 FEB 2019

20 points

Assignment Description. The purpose of this assignment is to familiarize you with the concept of specialty programming languages and expose you to some of the unique grammars and structures that programming languages can offer.

Part One. The first part of the assignment is to familiarize yourself with how works of interactive fiction are played by the end user. Interactive fiction is, above all else, an early example of computer video games. Unlike the early graphical games, like those in the Atari 2600 and early arcade games, early PC games catered to a different kind of audience; people who owned PCs in the early 1980s were typically more educated and more often worked professionally in computing (i.e. computer programmers). Because of this, the computer operating systems and built-in programming languages were often intertwined together and the expectation was that an owner of a PC was also a programmer to some degree.

In this first part, you will play the game Planetfall, written by Steve Meretzky. *Planetfall* is considered by many to be one of the best entries of interactive fiction and it is one of my personal favorites. I have included all of the necessary game files for you to download on Beachboard, including all of the relevant game manuals. You will also need an interpreter in order to play these games.

- **Windows**: I have included WinFrotz on Beachboard for your use. The program has long been abandoned by its creator, but it's still one of the best Inform interpreters around.
- **MacOS**: Download the program Spatterlight from Github.
- **Linux**: Install the program *frotz* in apt-get or similar repository. It is a command-line IF interpreter and it is extremely robust/considered the standard for IF interpreters today.

I know that it is not common practice in computer games today to read manuals before diving into games (that's what tutorials are for, right?) but *Planetfall* harks back to a time when disk space was extremely limited (in the kilobyte range!) and online documentation was non-existent. As such, **I highly recommend reading all of the included documentation before attempting to play the game**. Not only will the documentation help you understand how the parser will interpret your input, but the manuals (mostly) are written to help you as the player fully immerse yourself in this world you are about to enter and give you necessary background information. They are also, in my opinion, some of the most quirky and humorous game materials that I have ever read and serve well to involve you in the game from the start.

Planetfall is a text game which is puzzle-based and the game will throw various puzzles at you to solve in order to progress the storyline. **Please avoid the temptation to Google a walkthrough for the game**. The game should be played and enjoyed; I do not expect you to be able to finish the entire game (as people often played these games for weeks before finishing), but attempt to get as far in the game as you can. The game also contains many "in" jokes intended for the computer-literate crowd which were atypical for the era (which I also hope that you will appreciate). Don't be afraid to experiment within the game and test the limits of the parser—you might be surprised at how well the parser will understand you. Part of the enjoyment of these games is trying new things out and attempting to solve puzzles in unexpected ways.

Planetfall also exists on several sites which allow you to play in a web browser, but they typically lack the save/restore features which are important to actually finishing the game. You will also need to know how to use an interpreter to play the game you will eventually create in Part Two.

No deliverables are required for Part One. Simply use this time to play and enjoy the game and understand how interactive fiction works.

Part Two. In Part Two, you will create a game using the Inform programming language. The use of the included IDE for Inform is optional, but recommended.

Your game should contain one (or more, if you like) of each of the following:

- **Locked door**. In its most boring form, you must find a key and use it to unlock a door, thus giving you access to one or more additional rooms. With a little more imagination: You aren't admitted without a badge. You need to buy a ticket. You must give the troll a gold piece before you can cross the bridge. Waving the magic wand causes the rainbow bridge to appear. Et cetera. Any sort of locked door puzzle will do.
- **Hidden object**. Boring form: You open a box and find something inside. More interesting: You break open a treasure chest. You use the combination to a safe. You peer into the crystal ball. You buy the candy bar from the vending machine. You disassemble the robot to get some part out of it.
- **Incomplete object**. Your flashlight needs batteries. Your gun needs bullets. Your car needs gas. Your bicycle has a flat tire. You need a computer to get at the information on a floppy disk. You are a zombie and need a brain.
- **Limited resource.** You have a limited amount of time (to find the bomb before it goes off) or money (to buy the things you need), or food, drink, or sleep (so you don't collapse), or some other resource. Maybe you can find more resources in the game, maybe you can't.

Review the documentation for Inform 7 on the language's official website. The site includes the language documentation as well as example source code files in the form of complete, compliable games.

Deliverables. Submit your source code **and** the compiled Inform file (in .z8 format) through Beachboard Dropbox. I should be able to play your game by downloading the compiled file and running it in my interpreter of choice (all compiled .z* files should work in any compatible interpreter). Make sure that you have adequately commented your original source code, else I will not grade it and you will receive zero points for the assignment!