

Assignment 2 – An Augmented Adventure

Due: 5pm on Friday 6th October, 2006

* Forest Path

You are on a narrow path leading through the forest. It's autumn and the path is covered in golden leaves which crunch underfoot. A letterbox sits atop a post on the edge of the path, and you see a small white house to the south. It smells like it's going to rain.

> look path

The dirt path looks like it has been worn into the forest floor by many generations of trampling computer science students.

> open letterbox

You look inside the letterbox and discover a key.

Outline

Your aim in this assignment is to first understand, and then modify a program which implements a simple text adventure game. In a text adventure the player uses text commands to interact with a simulated environment. The computer responds by describing the environment and the result of the player's actions upon it.

The first text adventures appeared in the late 1970's with the release of *Colossal Cave Adventure* in 1976 and the *Zork* series in 1979. Commercial interest in text adventures reached its peak in the mid 1980's when companies such as *Infocom* and *Legend Entertainment* developed a large number of games for the home computer market.

Although text adventures are no longer viable as a commercial product, modern games such as *Myst* and *World of Warcraft* still use the same style of gameplay as the original adventures — as well as ideas from the multi-user dungeon (MUD) games that developed along with the growth of the Internet.

More background information can be obtained via the Wikipedia at:

http://en.wikipedia.org/wiki/Interactive_fiction
<http://en.wikipedia.org/wiki/Zork>
http://en.wikipedia.org/wiki/Multi-User_Dungeon

Required Task 1: Implement the `get` verb

In the provided code, support for the `get` verb is incomplete and the player is unable to pick up items from the world. Finish implementing the `get` verb. The player should only be able to pick up items in the current room.

Required Task 2: Add new rooms under the basement

In the basement of the house is an iron grate. Add a series of rooms underneath the basement which can be reached by first opening, and then passing down through this grate. The exact number, layout and description of the new rooms is up to you. Add two or three with a variety of exits between them.

Required Task 3: Make the grue happy

Looking at the grue reveals that it wants an item from the world. Modify the game so that once the player gives the grue what it desires, it appears happy instead of anxious.

Required Task 4: Something in return

Modify the game so that when the player gives the grue the item that it wants, the grue gives an item back to the player in gratitude. This item should be required to perform some other action in the game.

Free Choice Tasks

Successful completion of the *Required Tasks* above is sufficient to achieve a *Pass* grade on this assignment. For marks better than a *Pass*, you will need to extend the game with your own ideas. Here are some ideas for extra interactions, arranged approximately from easiest to hardest. You could implement some of these ideas, or use your own:

- Require the player to be *carrying the rope* to be able to descend through the grate in the basement.
- Add a *light switch* to the kitchen to turn the basement light on and off. If the player enters the basement with the light off then they shouldn't be able to see any descriptions or pick up any items.
- If the player lingers in the basement for too long with the light off (perhaps 5 moves) then they should be *eaten by the grue* and become non-alive. Warn the player that they are in danger before this happens.
- Add a *trophy case* somewhere in the house. Allow items to be placed in the trophy case. Allow the player to calculate a game score based on the selection of items in the case.

- Add a source for the *humming sound* in one of the rooms underneath the basement. Perhaps the humming sound emanates from a piece of machinery. Allow the player to interact with the machinery and turn the humming sound on and off. This should be reflected in the descriptions for the kitchen and basement as well.
- Add a *pet*, (cat, dog, bird, lizard etc) which walks around in the up-stairs house independently of the player. The pet should change rooms after every few player moves. The pet should remain in the house and not go outside or into the basement. Ensure that the player can only interact with the pet when it is in the same room as them.
- If the pet is in the same room as the player when they enter the basement, the *pet should follow the player* into the basement as well. Once the pet is in the basement it should interact with the grue in some way.
- Add some way for the player to obtain food for the pet. Allow the player to *train the pet* by giving it this food. Once the pet is trained it should follow the player wherever they go. Require this interaction to have succeeded before the player may enter a certain room — perhaps there is another creature blocking the entrance which the pet must scare away.
- **For thrill seekers:** Use the example code from the lecture on *Files and Networking* to modify the game so it can be played over the network via telnet.

Augmenting the adventure

The example code can be obtained from `/dept/dcs/comp1100/public/AAdvent` in the lab environment, or as a `.tgz` archive file from the course website. As per the previous assignment, you could copy these files to your own home directory and then work on this copy.

The example code uses *pattern-guards* which are a non-standard extension to Haskell. To run the game, change to the `AAdvent` directory and use the command:

```
ghci -fglasgow-exts -isrc src/Main.hs then evaluate main to start the game.
```

Alternatively, you can compile the code into a binary executable file:

```
ghc -fglasgow-exts -isrc --make src/Main.hs -o bin/aadvent
```

Submission

You should submit a single archive file containing all the files relevant to this assignment. You can submit either `.tgz` or `.zip` archive files.

Your archive file should contain all the source files required by your modified adventure game, *as well as a log file of the game being played* which demonstrates the interactions you have implemented. This log file should have the filename `LOGFILE` and be located in the main `AAdvent` directory.

On the lab machines: To create the logfile, open a new konsole and use ghci to start playing the game. After you have finished demonstrating your interactions, select `edit` ⇒ `save History` as and save your file to the appropriate place.

To create a `.tgz` file, change to the directory which holds your `AAAdvent` sub-directory and execute the following command:

```
tar zcf Asst2-uXXXXXXX.tgz AAAdvent
```

Replace the `XXXXXXX` in the file name with your own student number. Executing this command will create a new file `Asst2-uXXXXXXX.tgz` which contains all of the files in the `AAAdvent` sub-directory in archived form. To test that this has worked, copy `Asst2-uXXXXXXX.tgz` to a new directory and execute the following command:

```
tar zxf Asst2-uXXXXXXX.tgz
```

This will unpack the `Asst2-uXXXXXXX.tgz` archive file and reconstruct a new copy of all the files that were originally in `AAAdvent`.

On windows machines: `.zip` files can be created by using a free utility such as WinZip – available from <http://www.winzip.com>.

If you copy a `.zip` file from a windows machine to the lab environment you can use the `unzip` command on the lab machine to unpack it. Use the command

```
man unzip
```

on the lab machine to get the manual page for `unzip`.

Electronic Submission: You should submit your archive file by the published deadline. Late submissions will be penalised 10% for each day (or part thereof) past the deadline. Assignment submission will be electronic. Submit your archive file with the command:

```
submit comp1100 Asst2 <file name>
```

You must do this via the lab environment. Make sure you practice submitting your files well before the deadline. It is possible and perfectly acceptable to submit updated versions of your files. Our system will collect them all and your most recent submission will be the one assessed.

You should check that your submissions have been successful by clicking the **View Marks** button in **StReaMS** (<http://cs.anu.edu.au/streams/>). You should see a list of files together with the time and date of submission.

Assessment

This assignment forms 15% of your total assessment for COMP1100.

As this is a course on programming and not game design, you should focus on extending the game to support interesting interactions with the world, instead of simply adding more rooms and items. Detailed english descriptions of rooms and items are nice, but they are not required.

After completing the required tasks your submission will be assessed on the number and complexity of the interactions that the player can perform with the game world — along with programming style, comments, layout, technique etc. Implementing two or three of the ideas mentioned in the Free Tasks section would be more than sufficient to achieve full marks for this assignment.