Eric Roberts
CSCI 121

Handout #52
November 16, 2018

Adventure Contest

Due date: Wednesday, December 5, 11:59 P.M.

Project #5 requires you to build a minimal Adventure program. In terms of features, it has about as little as you can get away with and still have a chance to construct data files that allow it to play even mildly interesting games. For the Adventure contest, your goal is to extend your Project #5 submission to be the most interesting Adventure game you can create. To create your new game, you will have to change both the program and the data files. You are free to incorporate parts of the Crowther cave, but you will be judged only on the material you add. The most exciting and playable submission wins.

For the purposes of the contest, it is acceptable to abandon the principle that programs of this sort should be entirely data-driven. In Will Crother's original Adventure game, the FORTRAN program included highly specific code to implement most of the operations. In particular, each of the action verbs was associated with a section of code that explicitly took care of each possible situation. If the player typed in **`WAVE`**, for example, the program would explicitly check to see what object was being waved and whether the player was standing in a particular room. If the right conditions applied, the program would take whatever actions were necessary to update the state. For your contest entry, you should feel free to do the same thing.

Possible extensions

The following extensions would make the Adventure program much more powerful and would allow the construction of more interesting games:

- *Active objects.* The biggest weakness in the current game is that the objects are entirely passive. All you can do with an object is to pick it up or drop it. Moreover, the only way in which the objects enter into the play of the game is in the specification of locked passages in the room data file: if you're carrying an object, some passage is open that would otherwise be closed. It would be wonderful if it were possible to type **`WAVE WAND`** or **`UNLOCK GRATE`** and have the appropriate thing happen. Moreover, being able to **`READ`** or **`EXAMINE`** an object adds a lot of interest to the game.
- *Permanent objects.* Many objects such as the metal grate are in fixed locations. In the Project #5 version of the game, these features are part of the room description and not part of the object or vocabulary files. This situation leads to silly exchanges like

```
You are in a 25-foot depression floored with bare dirt.
Set into the dirt is a strong steel grate mounted in
concrete.  A dry streambed leads into the depression from
the north.
> TAKE GRATE
I don't know what you want me to take.
>
```

It would make far more sense if the grate were an object that could not be taken.

- *Object state.* In the original version of Adventure, objects can have different states. For example, the grate at the entrance to the cave can be either locked or unlocked; similarly, the snake in the Hall of the Mountain King can be blocking your path or driven away. You might add some way to allow the program to keep track of the state of each object and then make it possible for the motion rules to indicate that a particular passage can only be taken if an object is in a certain state: you can go through the grate only if it is unlocked.
- *Containment.* In Don Woods's extension to Adventure, some objects can contain other objects. Putting this concept into the game adds dimensionality to puzzle construction, but also requires implementing prepositional phrases in the parser so that the program can parse such constructions as

  ```
  > PUT NUGGET IN CHEST
  ```

- *Filler words.* The current parser limits the player to using commands that consist of one or two words. Saying

  ```
  > TAKE THE KEYS
  ```

  causes an error because the program doesn't know the word **THE**; if the parser ignored articles and other filler words, the program would seem more conversational.
- *Adjectives.* A similar extension to the parser is the introduction of adjectives that allow the player to issue commands like

  ```
  > TAKE BLACK ROD
  ```

  In the classic Adventure game, adjectives are associated uniquely with the noun to which they refer. In Zork, on the other hand, adjectives were used to differentiate many different objects of the same time, so that there could be both a black rod and a green rod in the same game. In fact, completing the Zork game required the player at one point to issue the five-word command

  ```
  > TELL ROBOT PRESS TRIANGULAR BUTTON
  ```

- *Convenient shorthands for "all" and "it".* When you're in a room with many objects, it is extremely useful to be able to type

  ```
  > TAKE ALL
  ```

  to take all the objects at the location. Similarly, the conversation flows more smoothly if you can refer to the last mentioned object as **IT**.
- *Random passages.* There are several rooms in the original adventure game at which the motion through a passage is probabilistic. You could implement this sort of feature by specifying a percentage chance on a locked passage rather than an object. Thus, if the data for a room specified the connection entries

  ```
  SOUTH      RoomA/30
  SOUTH      RoomB
  ```

moving south would go to room `RoomA` thirty percent of the time and to room `RoomB` the rest. (The program can differentiate this specification syntax from traditional locked passages because the percentage chance begins with a digit.)

- *Graphics.* Given the success of games like Doom and Myst, the feature that most people would like to add to this game is graphics of some kind. You are free to add graphics to the game, although doing so places you in a different judging category, as outlined later in this handout.

The real test of an adventure game, however, is not how many features the driver program contains but rather the quality of the puzzles you have designed. As you write your contest entry, you should focus on implementing those features that allow you to design a better and more playable game.

Submitting entries

To enter the contest, you must submit a zip file containing the code and data files that implement your adventure game. If your extensions increase the range of what the player is allowed to do, you should also describe those extensions as part of the text generated by the `HELP` command.

Judging categories

Past experience has shown that it is impossible to compare entries that use graphics against purely text-based adventure games. Each style has its own characteristics and requires different kinds of creativity. The programming necessary to add graphics to the game is not really that difficult, but it takes a lot of time to collect the images you use in the game. A text-based adventure often requires more programming—often in the form of adding sophisticated features such as containment or object state to the game—to achieve the same levels of playability. Thus, there are two separate categories of prizes for this game:

1. Purely text-based adventures that make no use of graphics.
2. Graphical adventures that make use of graphics in any way.

The entries will be judged separately, and we will award a grand prize in each category.

Prizes

The first prize in the Adventure contest is that you (or both members of your team if you are working in pairs) will be able to replace the lowest individual score—assignment, midterm, or final—with a 100%.

Instead of awarding runner-up prizes, each extension you make to the game will be treated as an extension to Project #5 and bring you closer to a + or ++ score.

Contest rules

1. Only students registered in CSCI 121 are eligible to submit entries.
2. If you are working as a two-person team on Project #5, you must submit your entry as that team. Both members of a team will receive the prize.

3. Your entry is due by 11:59 P.M. on Wednesday, December 5. Thus, only assignments that are submitted on time are eligible for the contest. No late days may be used on the contest.
4. Contest entries should be sensitive to Reed's individual and cultural diversity. Programs or narratives that have the effect of perpetuating negative stereotypes will not be accepted.