COMPSCI 270 - Numeric Artificial Intelligence
Project I
Two Player Games

Due date: October 29

1 Description

In this project you will design and implement a general game player, and apply it to a role playing game that Monika has devised for the class. The specific details of the role playing game will be provided in the readme file on the class web page.

The preferable implementation language is some dialect of C, but other languages can be used as well, as long as you can establish communication with the game server that we will provide and so long as you provide adequate comments in your code.

The project is designed to be quite open-ended and you are encouraged to have some fun exploring different methods of improving your player. We want to encourage some competition, but not make things so competitive that we take the fun out of it. You will receive a score of 70 percent for a successful implementation of alpha beta search that beats Monika's greedy player. You will receive an additional 25 percent for implementation, and discussion of an advanced method (more below), and the remaining 5 percent will be determined by your participation in a tournament. The best player in the tournament will receive extra credit.

2 The Game

We won't go into great detail on the game in this handout, but the basic idea is that you have a band of characters who are engaged in combat with another band of characters. At each turn, you select from one of four characters and pick one of three possible attacks against the opposing set of characters. This gives a nominal branching factor of 48, but in practice there will typically be fewer options than this since characters must rest between turns and, in the end game, some characters will get killed off.

The object of the game is to kill off all of your opponent's characters before he or she kills off your characters.

3 Tasks to Do

Here is list of tasks that you are expected to accomplish for this assignment:

- Implement minimax search. For this part you should use the evaluation function that simply looks at the difference between the health of your characters and your opponent's characters. You will also have to consider what sort of data structures you will need in order to create a game tree.

- Construct an improved evaluation function and show that it at least beats the simple one described above.
- Make the search more efficient by implementing alpha-beta pruning. Make sure that you can turn alpha-beta pruning on and off as you wish.
- Implement at least one improvement to basic alpha-beta pruning. Some acceptable improvements would be: techniques for improving pruning by carefully choosing which nodes are expanded first, the use of reinforcement learning to improve your evaluation function, or other advanced techniques described in Chapter 6. *Before deciding on one of these techniques, you should check with me to make sure that you are doing something reasonable.* **Do this by October 19.** Your writeup should clearly describe the technique you have tried and should show a thorough evaluation to measure the merit of this technique. Your technique does not need to be completely successful in gameplay for you to get a good grade on this part, but you should demonstrate that you have done a thorough evaluation.

4 Competition

When all of the projects have been turned in, we will have a competition to determine who has made the best player. The competition will involve two limits: A limit on the number of nodes expanded per turn, and a limit on the number of CPU seconds per turn. It will be up to you to enforce the former (but we'll check your code to make sure) and the game server will enforce the latter.

The actual time and node limits will be announced before the competition.

5 Measuring Time in C

Since each move in the tournament will have a time limit, you will probably need to measure time in your program. You may use any method you want, but here we will present a simple way to get a reliable measure of the total CPU time used by your process. First you need to include the file `time.h`. Now you can use the procedure `clock()` that returns a value of type `clock_t` (which is actually `long int`). This is the number of clock ticks used by your process since the first time `clock()` was called, which we do in the initialization of the client. In Unix, one clock tick is a micro-second; a safer way of converting clock ticks to seconds is to divide by the constant `CLOCKS_PER_SEC`. You should make sure that this total number of seconds never exceeds the time limit! For more information on this function, you can read the Unix man page on `clock`.

If you want to perform a loop for no more than `t` seconds, then (assuming each iteration does not take more than `epsilon` seconds) you can write:

```
clock_t start;
start = clock();
while (((clock() - start)/CLOCKS_PER_SEC) / 1e9 < (t - epsilon)) {
  /* Do something in loop */
}
```

If `epsilon` is too big for effective measurements, you can put some test points inside the loop, based on the time left.

6 What is a Role Playing Game?

There is long history of role playing games, both on paper and on the computer. Some of you may be familiar with games such as *Dungeons and Dragons* in which groups of poorly socialized teenagers

explored imaginary dungeons and fought imaginary enemies, all within the confines of their parents' basements. The game and the genre continue today in many forms and it has even become socially acceptable to play these games in certain contexts.

Some of the earliest computer games of this type were *Rogue*, *Larn*, or *Hack*. In the first home computer craze of the 1980's these games were given primitive graphics in, for example, the *Wizardry* or *Ultima* Series. Modern games such as the *Final Fantasy* series continue this legacy.

7 Where did the characters come from?

The characters in the game Monika has created are based upon an animated series called *South Park*, a cartoon for adults that combines coarse, shock humor with social commentary.

You do not need to know anything about the show to play the game. Moreover, if you find the show offensive, you can easily change the names of the characters and their attacks to something that is more acceptable to you.

If you are curious about the show, let us know and we will arrange for an airing of some episodes.