Gamesman How To…

About Gamesman

Gamesman is designed to allow programmers to easily program games into the system and provide a nice interface for playing them. It is a program that solves simple games and then plays them perfectly: the only way to make it lose is to play from a position that is inherently disadvantageous to it—and not make a single mistake. Gamesman learns the winning strategy of any game by generating its complete game tree. That is, starting from the initial position of the game, it makes every possible move from every possible position, examining every single sequence of moves that is possible. It then knows exactly which move(s) to make from any given position to maximize its chances of winning. This process of *solving* the game is computationally expensive, which is why creating a complete game tree in Scheme is only feasible for small games with fewer than (approximately) ten thousand positions.

To "teach" Gamesman about a new game, you need to write functions and put them into a file that we'll call a *game module*. The big required functions are (there are others you'll write):

1. **`print-position`** – Prints a pretty textual representation of the game position.
2. **`whose-move`** – Returns the name of the piece that should make the next move.
3. **`do-move`** – Returns the position that results from a given move.
4. **`generate-moves`** – Returns all of the possible moves from a given position.
5. **`primitive-position`** – Tells us if the game is over or not.
6. **`print-help`** – Prints a help message about the game.

The description of each function is vague since a better one appears in the project description document. For this project, your job will be to write a game module for any one of the four games you choose.

Today, you should get acquainted with Gamesman and the two example modules that are provided. The first module implements the *1,2,…,10* game and is called `m1210.scm`; the second module implements the *Tomorrow's Tic-Tac-Toe* game and is called `mtttt.scm`. This naming convention is important: all game modules should be similarly named `m`*x*`.scm`, where *x* is the name of your game. Since *1,2,…,10* is such a simple game, the code required to write the module is minimal; the *Tomorrow's Tic-Tac-Toe* module is a bit more complicated.

How to Run Gamesman

Gamesman runs only on the STk Scheme interpreter; it will not work in Dr Scheme or the portal's interpreter. To run Gamesman you'll need to go to your Unix prompt and go into the directory called `gamesman` in your home directory. You do this by typing `cd ~/gamesman` to the Unix prompt. Keep all game modules and related files in this directory and things will work more smoothly.[1]

[1] To use Gamesman at home on Windows, you'll need to download STk and then expand the `gamesman.zip` file to your Gamesman directory in C:\Program Files\STk\MS-Win32\gamesman. The reason for putting the `gamesman` directory there is that STk will always be able to find it since STk itself must be installed there. On a Linux/Mac OS X machine, you can unzip the `gamesman.zip` file and put the `gamesman` directory anywhere. If you are using an operating system other than Windows, Linux or Mac OSX, please ask in lab or post to the forum and we'll figure out where to put the `gamesman` directory on your home computer.

The main Gamesman page (available off the portal) has more detailed instructions for downloading the code at home if you're curious. To start STk with `gamesman.scm` loaded into it (once you're in the `gamesman` directory), type `stk -load gamesman` You will get the STk prompt back, and then you type `(gamesman)` to run the program.

How to Use Gamesman

The Gamesman interface consists of four menus.

1. The very first one you see after typing `(gamesman)` at the STk prompt is the **Load Game Menu**. It detects all the game modules that are in your `modules` subdirectory in your Gamesman directory and provides an easy way for you to load them. Modules should adhere to the naming convention `m`*x*`.scm` so that this menu can spot them among your other files. Both `mtttt.scm` and `m1210.scm` should be in the menu. Go ahead and load the one you want to play.
2. You next come to the **Game-Specific Options Menu**. Unlike the other menus, the contents of this menu depend on the game you're playing. This menu lets you set the rules of the game before solving it (if you choose to solve it). It is here that you can choose to play the *misére* version of your game, or start the game from a different initial position. Each of these options is particular to the game you're playing. For example, if you've loaded the *Tomorrow's Tic-Tac-Toe* module, you can choose to allow or disallow diagonal wins. Soon you'll be adding your own game-specific options to this menu.

   Each time you change a setting in the game-specific menu Gamesman *will need to re-solve the game* since the previous game tree is no longer valid. Gamesman may still need to re-solve the game even if you ask it to solve but you change the settings back (it's not currently smart enough to realize that you have changed an option—and then switched it back). Gamesman lets you know if it plans to re-solve the game at the top of the menu, set off by the >>> symbol.

   An alternative to solving a game is to read a game tree from a file. This is a lot quicker than generating the game tree from scratch. **For the purposes of this lab, please do not solve Tomorrow's Tic-Tac-Toe; instead, read the game tree from the .tree files that are provided or just play without solving.** You *may* solve *1,2,...,10*, however, since it is a very, very small game (just 20 possible positions). More information on saving and restoring the game tree from a file is provided below.

   If you are playing *1,2,...,10*, press `s` to solve and play the game. If you are playing *Tomorrow's Tic-Tac-Toe*, load one of the game trees – either standard game or misére game – and press `s`. Don't worry, Gamesman will not re-solve the game if the game tree has been read from a file. (Hitting `p` will always play without solving)
3. The next menu you come to is the **Play Options Menu**. This menu determines the players of the game: human vs. human, human vs. computer or computer vs. computer, as well as allows you to pick your game pieces. Additionally, this menu lets you change options that are relevant when playing the game (as opposed to solving it). For instance, you can choose to turn *predictions* on or off. *Predictions* are messages from Gamesman that inform you of how close you are to winning, tieing or losing the game. You also may turn on SIMPLE graphics. *You will be required to come up with simple graphics for your game modules.* There also is a GUI (Graphics User Interface) setting of the graphics option. Soon we'll tell you how you can use the GUI option to create more advanced graphics for your game module. When you're ready to play the game, press `p`.

4. While you're playing the game, you have access to the fourth and final Gamesman menu, the **Move Options Menu**. To view this menu as you're playing the game, type a question mark instead of a move. You can use this menu to abort your game and return to any of the previous three menus. Another cool feature you have access to when playing the game are the *safe moves* (if the game has been solved or if the tree has been read from a file) If you type `s` instead of a move, Gamesman will tell you which moves it would pick from the current position. If you are winning, the safe moves are the moves that will secure a win in the shortest time; if you are in a tie, the safe moves will keep you from losing. If, however, you are losing, then there is nothing that safe moves can do for you; every move is equally futile. Of course, you are under no obligation to follow the prediction; you can always undo a previous move, start the game over or just quit.

How to Save and Restore Game Trees

As not more than four or five people can be solving their games at the same time in the lab (that process is computationally intensive), we highly encourage you to save the solved game trees to a file. After solving the game, you can write the game tree to a file with the `W` command in the Play Options Menu. Reading the game tree, on the other hand, is done at the Game-Specific Option Menu with the `R` command. The reason that they are in different menus is that it only makes sense to save a game tree after Gamesman has solved the game. Right after solving, Gamesman will put you into the Play Options Menu, so we put the `W` command there. Reading a game tree makes sense right *before* the game is solved—or, more, accurately, it is done *instead* of solving; the menu where you will be before solving the game will be the Game-Specific Options Menu, so we put the `R` command there.

All Gamesman game trees must be named *x*`.tree`, where *x* is anything you want, and stored in the `trees` subdirectory. The game tree file will be created when in your Gamesman directory. When reading a game tree, enter the name of the file (including the .tree extension). Gamesman will look for the game tree file in the `trees` subdirectory of your Gamesman directory.

Images & Sounds

You are welcome to use images (`gif` and `ppm`) with graphics, both simple and GUI-based. All images must be stored in the `images` subdirectory of your Gamesman directory. Similarly, all sounds should be stored in the `sounds` subdirectory of your Gamesman directory. Note that *you're not required to have any images or sounds to get full credit for the project*, but they're a great way to get extra credit!

Where to Get Help

If you are having technical problems with either the `gamesman` system, your game module itself or just need some help, the best thing to do is to ask the staff in lab or post your question to the forum on bspace. Get in the habit of reading it regularly as we do this project since we will check it regularly. Sending email to Dan or your TA *is not* the way to ask questions for your project, these emails will be ignored (sorry). DO NOT post *your code* to the forum (e.g., to ask for help with a bug); it could be considered academic impropriety.

We will also be maintaining a Web page devoted to the project with answers to Frequently Asked Questions (FAQ). We will be posting there frequently. There is a link off the portal.

We certainly hope you enjoy this project as much as we do!!