Galaxy Robot Program Competition

The Galaxy competition is all about developing a program to compete with other programs to conquer the galaxy. This particular game is easy to build a robot for in 1 to two hours for a simple robot, but for those that want a smarter one you can do a lot better with more time developing the analysis portion of the program. The galaxy game itself is fashioned after Risk and a similar named game from the 80s. You can build a robot in any language. The first thing a Robot needs to do is print to stdout it’s name. The robot below is named “MarkBot”.

Output from the robot is in red bold and is simply text sent to stdout.

Input from the galaxy server is green and comes in to the robot program from stdin.

The robot must first announce its name.

*MarkBot*

In C this would be printf(“markbot”). When the server is run it redirects stdin and stdout to itself, thus simple scanf and printf enable communication between the server and the robots.

After collecting the names the server creates a galaxy with planets. Planets have an x,y location used for calculating distance and travel time, a name which consists of a single letter, a initial quantity of ships and a production rate which generates new ships each turn. Next the server sends the list of planets to stdin of your robot. Here is a sample list….

*\*Planet Update turn 3 markbot*

*Planet A Owner Nikhil X 4 Y 3 Ships 40 Prod 10*

*Planet B Owner Daniel X 0 Y 4 Ships 40 Prod 10*

*Planet C Owner markbot X 2 Y 1 Ships 40 Prod 10*

*Planet D Owner NEUTRAL X 1 Y 1 Ships 60 Prod 5*

*Planet E Owner NEUTRAL X 9 Y 2 Ships 58 Prod 4*

*Planet F Owner NEUTRAL X 6 Y 1 Ships 58 Prod 4*

*<Planet G Owner NEUTRAL X 3 Y 1 Ships 68 Prod 9*

*Planet H Owner NEUTRAL X 0 Y 2 Ships 60 Prod 5*

*Planet I Owner NEUTRAL X 1 Y 2 Ships 60 Prod 5*

*Planet J Owner NEUTRAL X 7 Y 4 Ships 68 Prod 9*

*Planet K Owner NEUTRAL X 3 Y 0 Ships 70 Prod 10*

*Planet L Owner NEUTRAL X 4 Y 4 Ships 68 Prod 9*

*Planet M Owner NEUTRAL X 4 Y 2 Ships 60 Prod 5*

*\*Planet Update Complete*

This particular galaxy has 9 planets names A-M. There are 3 players. This particular player is “markbot” and it is turn 3. Note that the update sections have a header and footer which begin with an asterisk for easy parsing. The same galaxy data is sent to each robot with only the name changed in the first line.

The robot can issue multiple move commands to move ships from one planet to another. The robot can only move ships on planets owned by the robot. The robot does not have to issue a move command every turn.

Sent to stdout by the robot to move 30 ships from planet A to planet C:

*move A to C ships 20*

The completion of the move commands must be indicated by the robot issuing the “end turn” command to stdout.

*end turn*

The server then collects all of the move statements from all players, it then calculates the number turns to travel and schedules the arrival of the ships to their destination. The game steps are the following for each turn:

1. Planet update
2. Collect move commands from all players
3. Schedule arrivals for move commands based on distance
4. Update production for each planet
5. Simulate battles
6. Repeat until victory condition (all planets owned and no ships in flight from other players)

The server issues event results to the screen. The information is not sent to the robot, but can be inferred by the planet update since it lists the owners and ships.

Here are the three types of event updates that the server will report:

*Event: markbot has reinforced planet A with 166 ships and now has 225 ships*

*Event: markbot has attacked Terminator with 520 ships on planet A with a defense of 8 ships and has won with 518 ships left*

*Event: markbot has attacked Terminator with 10 ships on planet A with a defense of 20 ships and has lost leaving 8 ships*

Victory occurs when a player owns all of the planets and no opponent’s ships exist. The game will end.

*VICTORY has been achieved for John!!!*

Some rules:

1. All robots see the same galaxy. All information is visible as it would be for a board game.
2. Neutral planets choose to not use their production
3. There is 6 out of 10 advantage for defense

**Checking out the Galaxy Server**

Download the server from github. It can run in Ubuntu Linux or Cygwin. Use the following command:

***git clone*** [***https://github.com/martintrowland/techdaycompetition***](https://github.com/martintrowland/techdaycompetition)

**Running the Server**

***make run***

You can use “make” to build the binaries and “make clean” to remove the files.

To run the server list each robot and the name the robot should use. The format is the following

***galaxy –x <Max X of the galaxy grid> -y <Max Y of the galaxy grid> -neutrals <number of neutral planets> -bots {{<robot binary> <name or parameter>}… }***

an actual example:

***./galaxy –x 10 –y 5 –neutrals 5 –bots ./playerbot.exe MarkBot ./playerbot.exe John ./player2.exe Terminator***

Other command line options:

***-seed*** <number> :to force the same game. Good for testing.

***-prompt*** :waits for enter to be pressed between turns. Good for debugging your bot.

***-help*** :prints a usage statment

The server will create child processes for each robot executable and re-direct their input and output. It will be referee and perform status updates.

If you have any questions or see some bugs, send me an email [Mark.Rowland@intel.com](mailto:Mark.Rowland@intel.com)