

# GEMpire Manual

Chris Marshall

8/14/2001

## Contents

|           |                                      |          |
|-----------|--------------------------------------|----------|
| <b>1</b>  | <b>Introduction</b>                  | <b>1</b> |
| 1.1       | Board and Turns . . . . .            | 2        |
| 1.2       | Combat . . . . .                     | 2        |
| 1.3       | Cities . . . . .                     | 2        |
| 1.4       | Exploration . . . . .                | 2        |
| 1.5       | Visibility . . . . .                 | 3        |
| 1.6       | Basic Strategy . . . . .             | 3        |
| 1.7       | Programs . . . . .                   | 4        |
| 1.7.1     | Map Editor . . . . .                 | 5        |
| 1.7.2     | Server and Client Programs . . . . . | 7        |
| 1.7.2.1   | Ending Turns . . . . .               | 8        |
| 1.7.2.2   | Setting Passwords . . . . .          | 8        |
| 1.7.2.3   | Saving Games on the Server . . . . . | 9        |
| 1.7.2.4   | Other Client Commands . . . . .      | 9        |
| 1.7.2.4.1 | Unload . . . . .                     | 9        |
| 1.7.2.4.2 | Build Piece . . . . .                | 9        |
| 1.7.2.4.3 | Get Map . . . . .                    | 9        |
| 1.7.2.4.4 | Review Turn . . . . .                | 9        |
| 1.7.2.4.5 | Production Select Mode . . . . .     | 9        |
| 1.7.2.4.6 | Piece Select Mode . . . . .          | 9        |
| 1.7.2.4.7 | Prompt Next Piece . . . . .          | 9        |

## 1 Introduction

GEMpire is a simple turn based multiplayer strategy game, written in java, designed to be played over the internet in client-server fashion. It is a generalization of the commercial game Empire Deluxe for the PC. It consists mainly of a server, a client, and a map editor.

## 1.1 Board and Turns

The game is played on a board which is a hexagonal grid. Each hex has a terrain type which can be land, sea, or several other species of land types. For the moment, we will consider the simplest configuration of the game. Each player, on any given turn, will have some number of pieces (which can be armies, fighters, bombers, destroyers, or several other types of military units) on the board. Each hex usually has at most one piece on it. The first player carries out his turn by moving some of his pieces around. Each piece has its limits (how many hexes it can move in one turn). After the first player has moved each of his pieces up to their limit, or at any point before, he uses the "end-turn" menu item to finish his turn. At this point, control of the board passes to player 2 and she can move each of her pieces up to their limit if she so desires, and when she ends her turn, control passes to player 3. Once each player has taken their turn, turn 1 is over and turn 2 begins the cycle again.

## 1.2 Combat

If player 1 tries to move one of his pieces (a fighter, say) on top of one of player 2's pieces (an army, say), combat ensues and if the army loses, it is removed from play and the fighter now occupies the army's former hex. If the fighter loses, it is removed from play and the army remains where it was.

## 1.3 Cities

Certain hexes contain cities. Cities have the ability to manufacture pieces (although land bound cities cannot manufacture ships<sup>1</sup>). Each piece type takes a certain number of turns to produce. All cities on the map start out either neutral, or owned by one of the players. A common scenario starts each player out with one city and no pieces (one of the reasons I wrote GEMpire was that I wanted to explore variations on this and many other aspects of the game). Each player, on their first move, chooses which piece type they want to manufacture (since they don't have any pieces yet, there is nothing else for them to do). As pieces become available, players move them out of the cities they were produced in and start exploring the map.

## 1.4 Exploration

Which brings me to the next point: exploration. The map is initially unknown to each player (all of the hexes are black except for the initial city and its neighbors). As a player moves her first piece out of her first city, the hexes around the piece are filled in with an appropriate images depicting the terrain type. As pieces move around, more and more of the map is filled in. Eventually, two player's pieces move next to each other and first contact is established.

---

<sup>1</sup>Acutally, they can, but the ship wouldn't be able to go anywhere once produced.

## 1.5 Visibility

Which brings me to the next point: limited visibility. Any given player can only see what is in the hexes adjacent to one of their pieces. Players have no idea what other players are doing except when their pieces rub up against one another. Thus, fighters, which can move over any terrain type at a rate of five spaces per turn, are the ultimate reconnaissance units, and typically lead the exploration of new land masses.

## 1.6 Basic Strategy

Let's explore the initial stages of the classic each-player-starts-with-one-city-and-no-pieces scenario some more. Player 1 has a city on the interior of a landmass and decides to produce an army first. 6 turns later, the city produces an army and player 1 starts marching it around. Before long, the army discovers another city (neutral) and attacks (by attempting to move the army into the city). The attack is successful and player 1 now owns two cities, and sets the second city to producing armies. Turns whizz by and player 1 is well on her way to owning her initial land mass (say she owns five cities now and there are a total of 8 cities on the mass). There has been no sign of enemy pieces yet. Now is the time to start producing a troop transport with one city and fighters with another. Fighters take 12 turns to produce and transports 30. Once fighters are available, player 1 can begin to see what lies beyond the shores of her initial land mass (within limits: a fighter can only fly so far before having to return to a friendly city or to an aircraft carrier to refuel; failure to do so will cause the fighter to run out of fuel and crash), and once a troop transport is available armies can be loaded onto it and the transport can carry them to another land mass to start taking cities. In the best of circumstances, by the time the transport is ready, the fighters will have located a city on another land mass so the troop transport can head directly for it. Second best would be if the fighters had located another land mass but not found a city yet. This would at least allow the transport to head for land, unload its armies, and head back to the original land mass to gather more armies. In the worst case, the transport heads off into the great unknown in search of other land masses at the rate of 2 sea hexes per turn. Soon after the first transport's production is under way is the time to start thinking about producing a destroyer (24 turns to produce). Destroyers, with their movement rate of three sea hexes per turn, are the fastest ships of all. Troop transports (unescorted) are easy prey for them. There is nothing quite like the satisfaction of sinking an enemy transport with 6 armies loaded on it (the maximum capacity), and nothing quite like the devastating loss it represents when you own the transport.

For these reasons (to sink enemy transports, to guard your own transports, and to explore the sea lanes of the map), destroyers are a must early in the game. As your military machine fully ramps up, it is time to think about producing capital ships, which have a very long lead time, and without which you will be at a severe disadvantage: cruisers, battleships, and air craft carriers. Cruisers

take 36 turns, Battleships 60 and aircraft carriers 48. Crusiers and battleships move at two spaces per turn and have the ability to bombard armies on land from sea hexes. A bombardment occurs when a cruiser is in a sea hex, an army is in an adjacent land hex, and the cruiser attempts to move onto the land hex. If the cruiser wins (it would have to be pretty heavily damaged to lose, although it might take 1 or 2 points of damage in the effort), the army is removed from play and the cruiser stays in its sea hex.

## 1.7 Programs

O.K., enough theory. How do you run the various programs? What do they do?

Here's what the command lines look like for the 3 main applications: the map editor, the client, and the server:

1. `java GEMpire.MapEditor`
2. `java GEMpire.EmpireClient`
3. `java GEMpire.EmpireServer <port> someMap.map`
4. `java GEMpire.EmpireServer <port> someMap.map GUI`

Before these commands will do anything, you must have unzipped the distribution file somewhere, you must have some version of the Java Developers Kit installed, and you have to set your classpath properly. If you are running under linux and your user account is fred, and you unzipped the distribution in `/home/fred/gempire-0.1`, you would want to set your classpath in a bash shell like this:

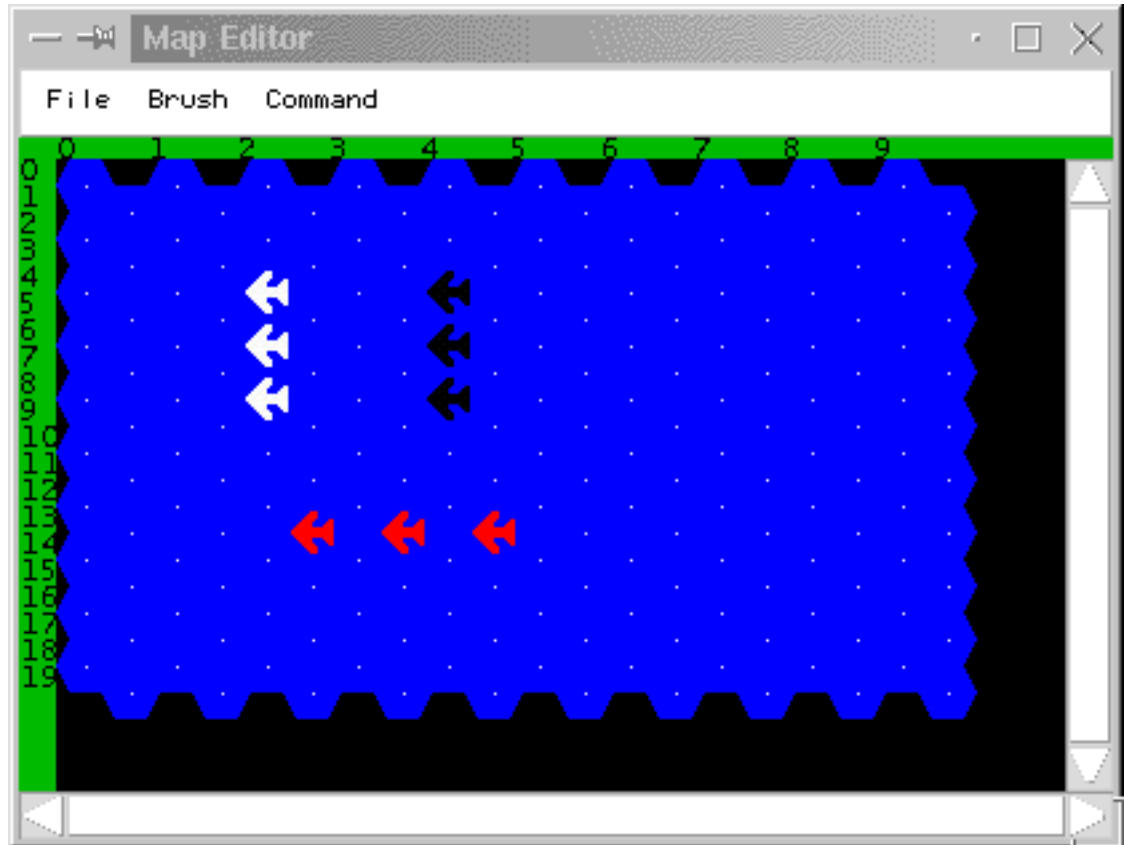
```
export CLASSPATH=/home/fred/gempire-0.1/GEMpire
```

Also, you will want to invoke these commands from `/home/fred/gempire-0.1/GEMpire`, so that GEMpire can find the `pieceinfo.txt` configuration file located there and the icons in the `images` directory there.

Line 1 starts a MapEditor, Line 2 starts a client, and line 3 starts a server that listens at the given local port and instructs it to load the given map or game file. Line 4 is similar to 3 but also brings up a client-like GUI without the visibility restrictions of a client (in other words, the whole map is visible, and you can watch the movements of every piece on the board as they happen in real time). Starting a server without a GUI is a useful thing to do when you want to run a server where you don't want or can't have X windows running at the same time. Say you have access to a shell account on a server somewhere, and you want to run an empire server on it. If the empire server always insisted on bringing up a GUI, you'd be out of luck. It is also possible to use a client to see the same view as the GUI from the server by logging in as player 0, which is a special player with god-like powers. This allows a game referee to remotely administer and keep tabs on a game.

### 1.7.1 Map Editor

In the top level GEMpire directory, there should be a test map called, test.map. First, try loading it into the map editor (choose “file->open”, and select the file “test.map” in the file dialog). You should see that it is a small map entirely composed of sea hexes with 9 fighters, 3 from each of 3 teams, that looks something like this:



The map editor has a "file," "brush," and "command" menu. If you choose “Brush->Piece Palette,” a control that looks like this should pop up:



The Piece Palette has one row of terrain images at the top with a grid of piece images below that. The grid has one row for each player and one column for each piece type. From left to right, the terrain types are sea, land, unknown<sup>2</sup>, mountain, forest, rough, river, and channel. Again from left to right, the piece types are city, infantry, armor, fighter, bomber, air base, troop transport, destroyer, submarine, aircraft carrier, cruiser, and battleship.

When you start a new map (by selecting “file->new”), it defaults to an all sea world. You can alter the terrain of a hex by clicking on the terrain type you want in the piece palette, then clicking on the hex. You add pieces to a map by clicking on the piece type for the player you want and then clicking the hex on the map where you want to deposit the piece. You can remove a piece by clicking on the hex again. Finally, you can enlarge or reduce the map (I mean the number of rows and columns in the map, not its display size) by choosing

<sup>2</sup>O.K. I this requires a little explanation. This is not any type of terrain, really, but an artifact of the exploration mechanism of GEMpire. Until a player has first visited a hex, she doesn't know what terrain it is. The EmpireClient uses the same Map class the EmpireServer does to represent the world, except the client starts out not knowing what most of the terrain is. The map has to assign a terrain type to each hex to draw it to the screen, hence the 'unknown' type. Perhaps it would have been better named the 'unexplored' type.

“command->resize map” from the menus. The number of rows and columns are required to always be even because of how the map is drawn.

The first row of pieces in the palette are for player 0, which is a special player as noted above. The only piece you would normally put on a map for player 0 would be a city. Player 0’s cities are “neutral” cities waiting to be conquered by the normal players (players 1 and above). Although it is possible to place neutral units of any type on the map, they are not intended for that purpose and I have not taken any pains to make sure the server and client programs handle them correctly. For all I know, they could cause a crash in certain circumstances. I suppose the proper thing to do would be to code the piece palette to not show those pieces as options, but I oppose that solution on principle, the principle being that I prize flexibility over idiot proofing. Idiot proofing drastically increases the complexity of code and decreases its flexibility. It seems a poor trade-off to me. For all I know, neutral pieces might turn out to have their uses later on, at which point, more testing could be done to make sure they work correctly. There seems to be something about GUIs that encourages idiot proofing at the expense of flexibility and this has been the cause of endless woe in software development.

### 1.7.2 Server and Client Programs

Line 3 is how you would typically start a server after you have created a world map. Again, for our example, you would want to type that command from `/home/fred/gempire/GEMpire`.

Line 2 shows how to start the client program. Once started, you use the “file->connect” menu item to connect to a server. If you had started the server with

```
java GEMpire.EmpireServer 8000 test.map
```

then you could connect to it on the same machine by entering “localhost” in the host field and “8000” in the port field of the connect dialog that pops up when you select “file->connect.” After connecting, you will get a login dialog, asking for a player number and password. By default, the server starts a new game by assigning the password “pass” to all players, including player 0, the referee.

If you look down to the command shell you started the client from, you should see some messages from the server, informing you of what the valid player numbers are.

Try logging in as player 0 by entering “0” in the player field, and “pass” in the password field, then clicking “ok” in the login dialog. You the client should retrieve a map of the world from the server and display it. You probably want to stretch the client window to see the whole map. You should be staring and the same map you looked at in the MapEditor earlier.

Now, from a different command shell, start another client and log in as player 1 this time. The player 1 client will have most of the map colored in black and only show you the 3 pieces owned by player 1 along with the cells surrounding them.

Now, from yet another command shell, start yet another client and log in as player 2. You should see a map, again mostly black, that shows you only player 2's pieces and their vicinity.

Because of the order in which login's occurred, player 1 currently controls the board. Player 2 can see what's going on but can not move any pieces. In the player 1 client, click on one of his pieces (the piece should start blinking, indicating that it is the "active piece") and start moving it toward one of player 2's pieces by using the arrow keys on your keyboard. Make sure you have the numlock off on the numeric keypad so you can use the PgUp, PgDn, Home, and End keys to move in diagonal directions.

You should see the piece movements in both the player 0 client and the player 1 client. The player 2 client won't see anything until player 1's piece is adjacent to one of her pieces. If you attempt to move player 1's piece on top of player 2's piece, combat will ensue and one of the pieces will be removed from play. From the player 0 client, you can rig the combat, if you wish, by using the "Server Command" menu so that either the attacker always wins or the defender always wins. If you try to do this from the player 1 or 2 client, the server will send you a message that permission was denied.

**1.7.2.1 Ending Turns** If you exit from the player 1 client, or you select the "Command->End Turn" menu item, control of the board will pass from player 1 to player2, who can now move her pieces. The normal order of play is for each player to log in, move their pieces, then select "end turn." Once each player that still has pieces on the board selects "end turn" the server will declare the turn over and start a new turn. It is possible for a player to log in, move some pieces, log out without ending their turn, log in again, move the rest of their pieces, and then select end turn. Notice how this simplifies the logistics over running play by mail games where turns have to be made in a fixed order of players. Although I haven't implemented this yet, I will add a timeout option to the server where players who don't make their turn in a given length of time (say, 24 hours) have their turn automatically ended for them. I have seen many multiplayer play by mail games run over the internet stall because one player stops making turns and refuses to divulge their password. With the server client architecture, you only need to trust one person, the referee. Cheating becomes \*much\* more difficult as the server does not send any more information to the client than it has a right to see. Writing a custom client will not allow someone to cheat.

**1.7.2.2 Setting Passwords** At any time, you can use the "file->set password" menu item to change the password for your player. The game referee should, in fact, change player 0's password before advertising the host address and port number his server is listening on, lest a sneaky player login as player 0 to cheat before the other players log on. Actually, the best protocol is for the referee to make up passwords, set them for each player, then email the players their passwords separately.



**1.7.2.3 Saving Games on the Server** From a client logged in as player 0, you can cause the server to save the current game state by using the “Server Commands->Save Game” menu item, which will then prompt you for a filename. If you go to the directory where the server is running after doing this, you should see a file with the name you specified. If you then kill the server and wish to restart it from the saved game file, assuming you had saved the game to the file “test.gam”, you would simply start it with:

```
java GEMpire.EmpireServer 8000 test.gam
```

#### **1.7.2.4 Other Client Commands**

**1.7.2.4.1 Unload** Once you have loaded infantry or armor pieces on transports, or planes on carriers, you use this command to unload them. You select the transport or carrier by clicking on it and, once it is blinking, choose this menu item, at which point the client will prompt you for which piece to unload. You can also access this command by typing ‘u’ after you have selected a transport or a carrier.

**1.7.2.4.2 Build Piece** This command is used for infantry and armor to build air bases.

**1.7.2.4.3 Get Map** This command retrieves the latest map. You normally wouldn’t need to use it as the map should be kept up to date automatically.

**1.7.2.4.4 Review Turn** This command lets you review what happened on a given turn. The Server keeps a log of all of the messages it sent to the client for each turn and will simply squirt them back at the client when requested to. This mechanism needs some work, though, to be more usable.

**1.7.2.4.5 Production Select Mode** You use this to set city production. You first select this item, then when you click on a city, its production menu comes up, allowing you to determine which pieces the city should produce. This should be the first thing you do after you capture a city. If you don’t, the city will not produce anything.

**1.7.2.4.6 Piece Select Mode** You would typically choose this to exit the Production Select Mode so you can select pieces for movement again.

**1.7.2.4.7 Prompt Next Piece** This hasn’t been implemented as a menu item yet, but exists as a keyboard shortcut. If you type ‘p’, the client will randomly select a piece that still has movement left and activate it for you, causing it to blink. Once you have a good number of pieces on the map, it can be very hard to remember which ones you have moved already and which you haven’t. The game would hardly be playable without this command.