Liquid War - a unique multiplayer wargame

Christian Mauduit

v5.6.2

Contents

Chapter 1

Rules

1.1 The Liquid War concept

Liquid War is a wargame. But it is different from common wargames.

When playing Liquid War, one has to eat one's opponent. There can be from 2 to 6 players. There are no weapons, the only thing you have to do is to move a cursor in a 2-D battlefield. This cursor is followed by your army, which is composed by a great many little fighters. Fighters are represented by small colored squares. All the fighters who have the same color belong to the same team. One very often controls several thousands fighters at the same time. And when fighters from different teams meet, they eat each other, it is as simple as that.

1.2 How do teams react?

Teams are composed of little fighters. These fighters all act independently, so it can happen that one single fighters does something different from what all the other do.

The main goal of these fighters is to reach the cursor you control. And to do that, they are in a way quite clever, for they choose the shortest way to reach it. Check it if you want, but it is true, they *really* choose *the* shortest way to reach the cursor. That is the whole point with Liquid War.

But these fighters are not perfect, so when they choose this shortest way, they do as if they were alone on the battlefield. That's to say that if there is a fighter blocking their way, they won't have the idea to choose another way, which is free from fighters but would have been longer otherwise. So fighters can be blocked.

1.3 Who eats whom?

When two fighters from different team meet each other, they first try to avoid fighting, and they dodge. But if there is no way for them to move, they get angry and attack the guy which is blocking them. Sometimes, they attack each other and both loose health. But it can happen that a fighter is attacked by another one, which is himself not attacked at all.

Here is an example of this behaviour: A blue fighter and a red fighter both want to move to their right, for that would be the shortest way to reach their cursor if there was nobody on the battlefield. But they are blocked by other fighters. If, for instance, the red fighter is on the right and the blue fighter on the left, it is the red fighter which will be eaten.

When a fighter is attacked, he first looses health, that is to say that he gets darker. When his health reaches 0, his color changes and he becomes a member of the team by which he has been attacked. Therefore the number of fighters on the battlefield always remains the same.

When fighters of a same team get stuck together and block each other, then they regenerate, that is to say that they get brighter.

However, I think the best way for you to understand the way it works is to try the game...

1.4 Basic strategy

When I play Liquid War, I always try to surround my opponents, and it usually works.

By the way, the computer has no strategy at all, he is a poor player, and if you get beaten by him, it means you have to improve yourself a lot!

But still, the computer doesn't do one thing which I've seen many beginners doing: he never keeps his cursor motionless right in the middle of his own fighters, for this is the best way to loose.

1.5 The winner is...

The clever guy who has got the greatest number of fighters in his team at the end of the game. Or the one who exterminates all the other teams!

Chapter 2

Authors

2.1 Thom-Thom

Liquid War rules have been invented by Thomas Colcombet.

He was trying to find algorithms to find the shortest path from one point to another, and found the Liquid War algorithm. Then it came to his mind that a game could be build upon this algorithm, and Liquid War was born. He programmed the first two versions of Liquid War using Borland Pascal for DOS, and gave me some information about the algorithm so that I could re-program it.

2.2 U-Foot

I'm the guy who programmed the latest versions of Liquid War. I enhanced the algorithms, and did quite a bunch of work to have the game playable by (almost) anyone, that's to say create a correct GUI.

If you want to join me, here's all the information you'll ever need:


```
Christian Mauduit

E-mail: ufoot@ufoot.org
Web site: http://www.ufoot.org

GnuPG public key: FD409E94 - http://www.ufoot.org/gnupg.pub
GnuPG fingerprint: 4762 1EBA 5FA3 E62F 299C  B0BB DE3F 2BCD FD40 9E94

Snail mail: 32 rue Jean Moulin  95100 Argenteuil  FRANCE
```


2.3 Other contributors

As Liquid War is now free software, protected by the GPL, anyone is allowed to view, edit, modify, re-compile the source code, and distribute it, as long as Liquid War is still distributed under the GPL.

Here's a list of the contributors:

- Alstar: drew a map, which is now included in the main distribution.
- Peter Wang: ported Liquid War to GNU/Linux.
- Cort Danger Stratton : helped me setting up network support.
- Tim Chadburn : wrote midi files for the game. His contribution has been truely appreciated since it's rather hard to find GNU GPL compliant artwork. He also wrote documentation and helped with midi support in general.
- Jan Gretschuskin : contributed 11 maps, and made the German translation. So if you run Liquid War with German menus, you know who you have to thank -> Jan! 8-) Also do not forget to visit http://www.game-factor.de
- Mouse : contributed a map.
- Rene Stach : drew 3 maps.
- Roderick Schertler : implemented HTTP 1.1 support.
- Ryan D. Brown : ported Liquid War to Mac OS X.
- Eduard Bloch : maintained the Debian package, and helped with German support.
- Michael Terry : provided a .desktop file for better integration with Gnome, KDE and other UNIX desktop environments.
- Kasper Hviid : contributed many maps, with their own textures, and made the Danish translation. Great work, also available on http://levels.2v1.cz/index.html
- David Redick : wrote an external random map generator, available on http://xdavidx.sqrville.org/lwmapgen/index.html
- Alexandre Pineau : maintains the Debian package.
- Michael Wagner : translated the web site in German.
- Peter Williams : fixed the "too many opened socket" server bug.

Many other people helped me by submitting bug reports and patches, and I want to thank them for their precious help. Thanks to all the Debian people too, who nicely maintain the Liquid War .deb package.

Chapter 3

Mailing lists

3.1 liquidwar-user

3.1.1 Description

This list is for general discussions about Liquid War. Here you can make suggestions, submit bug reports, ask for help, find players, etc... Basically, any question or remark which concerns the game is welcomed on this list.

3.1.2 Practical informations

You can't send messages to the list without subscribing. The only reason for this is that it's one of the only way to block spam efficiently. I first thought it could be OK to allow anyone to post, but liquidwar-user seems to have be harvested by robots, so now I need to restrict posters. However, I insist on the fact that anyone can subscribe, and the subscription to the list is not moderated. So if you are a human being and not a stupid spam robot, you're welcome on the list 8-)

Here's a list of usefull URLs:

- To (un)subscribe: http://mail.nongnu.org/mailman/listinfo/liquidwar-user
- To consult archives: http://mail.nongnu.org/pipermail/liquidwar-user/
- To post on the list: liquidwar-user@nongnu.org

3.2 Chat and IRC

3.2.1 Web-based chat-box

I have have set up a web-based chat-box which is accessible here: http://www.ufoot.org/liquidwar/metaserver.php3

It's not as good as a good old IRC channel but not everybody can use IRC (because of firewalls and the likes), and I like the idea that people can chat and have the list of available servers in one single web page.

3.2.2 IRC channels

I personnally spend some time on irc.freenode.net so you might use it to find other players - though I'm not really an IRC addict... ...not yet at least!

Here are the channels I recommend:

- #liquidwar : Liquid War dedicated channel, to find players and chat while playing.
- #netgame_players : general channel for players who want to play Internet games - Free Software and/or Open Source games of course, we're on freenode.net 8-)

Chapter 4

Menus and hot keys

4.1 Introduction

This section describes how the GUI works. Since programming advanced GUIs with Allegro is not so easy - standard C programming definitely lacks flexibility -, and also since it's somewhat hard for me to figure out what is user-friendly and what's not, Liquid War's menus are not always self-explanatory. I'll just try and do something better next time!

4.2 Menus

4.2.1 Map menu

The map menu allows you to choose the map you are going to play on. A map is defined by 3 things:

- A frame. The frame can be chosen with the slider which is below the preview. The frames are automatically sorted by alphabetical order.
- A texture for walls.
- A texture for the zone where fighters are allowed to move.

In the middle of the screen, there is a preview of the level. In this menu, the values of the parameters can be independently changed by:

- Moving a slider.
- Clicking on a "+" or a "-" button.
- Typing a number.

On each side of the preview, sliders allow you to choose the two textures. There is also a preview of each texture. Below this preview there are 128 little buttons which allow you to choose single colored textures.

The name of the map and its resolution are displayed in the lower part of the screen.

You'll notice that on some maps the texture selection zones simply disappear. This is because these maps are associated with a texture, so choosing a different texture is often not recommended for it won't look as nice as with the right one. If you still want to override this behaviour you can click on the ”on/off” button just right to the ”Use default texture” label. This is a toggle button which will allow you to use your own textures even on maps that normally come with their own skin.

You'll also notice that a ”Random map” button is available. This button generates a new random map using an external program, ”lwmapgen”, developped by David Redick, available on http://www.cs.clemson.edu/˜dredick/lwma

This program supports many command line options, and if you want a very precise control on the generated maps, you'll need to run it separately. Looking at LW's log file you should be able to see the commands LW issues when calling this program, this can give you ideas on how to launch it manually. Alternatively using the ”–help” option (for instance ”liquidwar-mapgen –help” under UNIX) should describe how to use it.

4.2.2 Teams menu

This menu allows you to choose the teams which are going to play. There are 6 square zones in this menu. Each of them is associated to a team.

Each team can be either:

- Disabled (”Off”)
- Controlled by a player (”Human”)
- Controlled by the computer (”Cpu”)

The computer plays poorly, so remember that Liquid War is basically a multiplayer game, and that the cpu control is dedicated to beginners only.

You can also choose the color associated to each team by clicking on one of the 12 colored buttons.

Below the 12 colored buttons, there are four buttons which allow you to choose your keys. Click on one of these buttons and then press the key you want to define. Joystick movements and buttons are considered as keys. You can disable the joystick with the button which is at the bottom left of the menu. Mouse input is also possible, and mouse movements are considered as keys too. To define mouse control, click on the button associated to the direction you want to control, and then move the mouse. Then the button should display something like ”M->”. Mouse sensibility can be set with the little slider at the bottom right of the menu.

4.2.3 Graphics menu

Here you can choose the graphic options of the game.

The ”Video mode” button allows you to switch between fullscreen and windowed mode. This button is not available under DOS.

The ”Brightness” slider allows you to set the brightness of the game.

The ”Menu res” slider allows you to set the resolution used by the menus. There are currently 5 possible values, which depend on which platform you're running the game on.

I personnaly think the menus look best with the 640x480 resolution, but some may prefer higher resolutions. Lower resolutions should only be used if you have problems using SVGA video modes.

The "Game res" slider allows you to set the resolution used during the game. The allowed values are the same than those for the menus. I recommend that you don't use resolution higher than 640x480, unless you have a Pentium VIII running a 10GHz.

Page flipping can be toggled. It is up to you to decide wether you keep this option or not. The main disavantage of turning page flipping off is that the info bar and the battlefield can look rahter ugly if they overlap. But if you turn page flipping on you will not easily reach the 166 frames per second I sometimes get on small levels with my K6-225. I personnaly always turn page flipping off.

The viewport size defines how much of your screen will be used by the battlefield.

- If you set the slider on its left position, the batllefield will not be stectched at all. Or if is strechted, it will be by a x2 or a x4 factor. So this is the mode wich allows the fastest display.
- If you set the slider ont its right position, the game will run in fullscreen mode.
- With all the other positions of the slider, the battlefield will keep its general proportions but it will be stretched.

The "Waves" button allows you to toggle the wave effect. You can also do this while playing, by simply pressing F4.

4.2.4 Sound menu

This section allows you to set the sound volumes. There are 4 sliders, which are:

- "Sfx": sets the volume of all the sfx sounds, thats to say'the sounds you hear when the game starts, when you loose etc...
- "Click": sets the volume of the click, this nasty noise you hear each time your press on a button.
- "Game water": sets the volume of the blop blop blop sounds which are played continuously while you are playing.
- "Menu water": the same thing than "Game water" except that it concerns the sounds played while your are choosing options.
- "Music": general music volume.

4.2.5 Rules menu

This menu is the one where you can change the rules of the game.

The "Time" slider controls the time limit. The game will stop after this time is elapsed. You can pause the game by pressing the "F3" key.

By the way, an info bar can display the time left while you are playing. This info bar can be toggled during the game by pressing the "F1" key, and you can change its location by pressing the "F2" key. It also displays how many fighters there are in each team.

The "Army size" slider controls the amount of fighters there will be on the battlefield. The position of the slider reflects the amount of fighters of all the teams together. If there are 4 teams, then each player will have half as many fighters than if there had only been 2 teams.

The "Cursor x" slider controls the speed of your cursor.

- If it is set on the left, the cursor goes at the same speed than the fighters.
- If it is centered, the cursor goes twice faster than the fighters.
- If it is set on the right, the speed of the cursor is multiplicated by 3.

Below is a "Min 160x100" box with a slider on its right. This means that maps will automatically be magnified so that they have a size of at least 160x100. Indeed, some of the maps that come with Liquid War were designed in 1995 when 486 Intel computers were common. Therefore the maps were smalls. Today, these maps are not really fun to play on fast computers, so Thomas Harte suggested this automatic magnifying feature, and that was IMHO a smart idea. You can move the slider to the right to make maps use a higher resolution - ie magnify them.

The "Defaults" button of the "Rules" menu will reset rules to their defaults. This way you can tweak rules and then come back to the default rules whenever you want. Note that there's also a "Defaults" button in the main "Options" menu, but it will reset *all* options, including player names... The advantage of the "Defaults" button in the "Rules" menu is that it will only reset rules parameters, and keep the rest of your configuration options untouched.

4.2.6 Speeds menu

The "frames/s" slider allows you to limit the number of frames per second. If this slider is set on the left, there won't be any limit, so Liquid War will repaint your screen each time the fighters move. But this can be a weird behaviour if your machine is really fast, for no one cares about 100 fps per second, one can not even see them... So this paramters limits the refreshment rate, so that there can be several logical moves of the fichters for only one screen refreshing. If it is set on its right, the display is limite to 10 fps, so you'll have to find your setting. I personnally set it right in the middle, and get 40 fps. If you press "F5", you'll get the number of frames per second, and if you press "F6", you'll get the number of logical moves per second. You can also press "F7" or "F8", and you will get the percentage of time your computer spends on calculating or displaying the level.

The "rounds/s" slider allows you to limit the number of rounds per second. If this slider is set on the left, there won't be any limit, so Liquid War will run as fast as possible. This setting will be of no use if you use Liquid War on a slow computer or if you play with hudge maps, but sometimes, with a high-end Pentium class computer, it's simply impossible to play on small maps because things simply go too fast. So this parameter is here to help you and avoid the "10000 moves per sec" problem.

4.2.7 Waves menu

This is where the wave parameters are set. The waves are just a graphic effect, which is not really usefull. I don't often use waves, but I still think they can sometimes look nice. Change these parameters if you really mean to do it, but if you don't understand what they mean, it is really OK...

There are 4 different types of waves, each of them being defined by:

- An "Ampli" parameter, to define how big the waves have to be.

- A "Number" parameter, to define how many waves should be displayed at the same time.
- A "Speed" parameter, to define how fast the waves should move.

If you want to undestand what the "WX", "HY", "WY", and "HX" codes mean, try to pay with only one type of wave, the "Ampli" parameter of the 3 other types of wave being set to 0 (that is to say the slider is on its left position), and sea how it looks like.

The wave effects can be toggled during the game by pressing the "F4" key.

4.2.8 Advanced menu

This menu allows the user to change the behaviour of the fighters.

The "Attack" slider sets the agressivity of the fighters. If it is set on the right, fighters eat each other very fast. If it is set on the left, it takes ages to fighters to change teams.

The "Defense" slider sets the capacity that the fighters have to regenerate themselves. The more it is on the right, the faster fighters regenerate.

The "New health" slider sets the health of the fighters which have just changed teams. The more it is on the left, the weaker these fighters will be.

The "Winner help" slider controls a parameter which causes fighters to attack with various strength depending on how many fighters belong to their team. Not very clear... Let's just say that:

- If this slider is set on the right, the more fighters you have in your team, the more aggressive they will become.
- If it is centered, all the fighters of every team will always attack with the same strength.
- If it is set on the left, the less fighters you have, the stronger they will be. In this mode, games usually never end.

The "Cpu strength" parameter never makes the computer more intelligent than a monkey. But if you set it on the right, it advantages the machine outrageously and fighters controlled by the cpu will be really strong. So to get rid of them you'll definitely need to be clever. Again and again, don't forget that Liquid War was conceived as a multiplayer game and that playing against the computer is not really an interesting thing to do.

The "CPU vs human" parameter allows you to control how aggressive CPUs are towards humans.

- If set to "Always", CPUs will always attack humans and will never try to attack another CPU, unless there are no humans left. This used to be the default behavior in previous Liquid War versions, but some players remarked that it was rather unfair, so now this is an option.
- If set to "Random", CPUs won't care wether their opponents are CPUs or humans, they'll attack anybody. This is the default behavior.
- It set to "Never", CPUs will attack each other before bothering human players.

The "Allow net bots" button can be used to allow bots to connect on network games. Indeed, bots are by default disabled in network games, since in this case LW assumes that bots are useless (there are already several human players). However, turning this option on will allow you to connect bots within the game. It's important to note that this is a per client option, this means that you can't use it to forbid access to bots to a given network game. This option was simply created to avoid confusion when connecting on network games, while still allowing bots to connects if you really want them to.

The "Algorithm" parameter allows you to force the algorithm to standard C mode. There's no real good reason you would like to do this, since the C algorithm is slower than the ASM one. Moreover, the ASM algorithm is automatically disabled if you play against a computer which does not have ASM enabled. Think of this as a testing/debugging option.

4.3 Hot keys

Here's a list of keys you might use while playing:

- F1: toggles the "info" zone where the game time and the state of each team is displayed.
- F2: moves the "info" the zone arround, possible positions being top, right, bottom and left.
- F3: pauses the game. This function is disabled during network games.
- F4: toggles the "wave effect". Without this "wave effect", which is turned on ny default,the game will run faster.
- F5: displays the number of frames per second (*).
- F6: displays the number of rounds per second (*).
- F7: displays the precentage of CPU spent on the game logic, calculating where fighters must go for instance (*).
- F8: displays the precentage of CPU spent on graphics (*).
- F9: turns on/off the "capture" mode. In this mode, screenshots of each frame are taken, and written to the hard drive as bitmaps.
- F10: quits the game right away without any confirmation prompt, also known as the "my boss is coming here!" function.

(*) all these figures tend to be clearly false as computer go faster and faster. Basically, the time required for "logic" and "display" operations is getting shorter and shorter, and the tools I use to measure it are not precise enough. Therefore I get approximations which might by plainly wrong.

Chapter 5

Network game

5.1 Basics

Since release 5.4.0, Liquid War includes network support, that's to say that people can play over a LAN (Local Area Network). However, due to limitations in Liquid War's legacy code, and also because of the lack of time I have, it might be a little tricky to set up a network game at first. So please read this section carefully.

You should keep in mind that:

- DOS only releases of Liquid War do not include network support, only Windows and GNU/Linux versions will allow you to set up a network game.
- The game should run fine on any LAN, but there's no garantee the game will be playable on the Internet. Indeed if your "ping delay" is not good enough, the game will be awfully slow. Bandwidth is not an issue, since Liquid War rarely needs more than 2 Kb/sec.
- You'll need to know what an IP address is.
- You don't need to set up a network game to run a multiplayer game. Liquid War was originally a multiplayer game without network support. Network support is here only for people who don't feel comfortable when playing at 6 on the same keyboard 8-)

5.2 Getting started

5.2.1 What do you need?

You'll basically need 2 computers connected on the same LAN. We'll call them computer A and B. You might be able to play over the Internet too, but the game can be harder to set up and - which is worse - very slow.

You'll also need to know the IP address of computer A. Type "ipconfig" under Windows or "ifconfig" as root under GNU/Linux to get this information if you don't have it.

5.2.2 Starting the server

Liquid War uses a very traditionnal client/server approach. Basically, the server gets informations from all the clients and then dispatches the collected information to everybody.

So you'll need to start a server on computer A by running "liquidwar-server" on GNU/Linux or "lwwin-srv.exe" on windows. This is a console application, ie it does not set up any graphic mode.

Here's a small example of a server start on GNU/Linux:

```
$ liquidwar-server
How many teams will connect to this server?
```

At this point you must enter a number between 2 and 6, and then press "ENTER". In this example we will answer 2. The server really needs to know how many teams will be in the game: when enough teams are connected, the game starts. It can also be a good idea to answer 6 systematically and then force game start by clicking the "Start now" button within the client. It's possible to skip this question by typing "liquidwar-server -2" instead of a plain "liquidwar-server".

```
Use "-2" to get rid of this question.
Register on "www.ufoot.org/metaserver/" (y/n)?
```

Now if we answer "y", then the server will automatically contact the "meta-server" and it will be listed on http://www.ufoot.org/liquidwar/metaserver.php3

This can be convenient for people who want to find other gamers to play with on the Net. For now, let's answer "n", we'll test this meta-server stuff later 8-)

```
Use "-private" to get rid of this question.
2002-06-03 16:43:00: Listening on port 8035...
2002-06-03 16:43:00: Waiting for 2 teams...
```

Now the server is ready to accept clients. By default it listens to clients on port 8035. You could change this behavior setting by calling "liquidwar-server -port 8061" for instance, but let's use the default port to make things easier.

5.2.3 Starting the clients

Start the client on computer A normally by typing "liquidwar" on GNU/Linux or double-click "lwwin.exe" on Windows.

Go to the "Teams" menu and select 2 teams, a red human and a blue CPU for instance. If you don't know how to do this, then try and play Liquid War on a single computer first. It's important to notice that by default the "blue CPU" won't connect on the network game. It's possible to allow bots to connect on network games, but they are disabled by default.

Now come back to the main menu, and a "Net Game" button should be available. Click it. Now you should be able to:

- Start the game.
- Change the IP address of the server.

- Change the communication port.
- Set a password.
- Search for internet games automatically.

Since the server is also running on the same machine (A), you can leave the default IP address as is (127.0.0.1).

Now you are ready to start the second client on computer B. Like with computer A, you'll have to:

- Select 2 teams, green an yellow this time.
- Select "Net Game" in the main menu.

But this time you'll also need to change the server address, since the client is not running on the same computer than the server.

Now click on "Start game" on computer A. The server should play a "system beep", which is usefull if you run a server and want to be notified of client connections without watching the console all the time, and it should display messages like this:

```
2002-06-03 16:44:48: Connection from "127.0.0.1:34677"
2002-06-03 16:44:48: Team "Napoleon" on client "127.0.0.1:34677" accepted
2002-06-03 16:44:49: Client "127.0.0.1:34677" accepted
2002-06-03 16:44:49: Waiting for 1 team...
```

And on the client you should see a screen which says "Waiting for 1 team(s)" with the list of connected players below (Napoleon). You do not need to click on the "Start now" button.

Now click on "Start game" on computer B. The server should display messages like this:

```
2002-06-03 16:49:14: Connection from "192.168.1.1:1098"
2002-06-03 16:49:14: Team "Henri IV" on client "192.168.1.1:1098" accepted
2002-06-03 16:49:15: Client "192.168.1.1:1098" accepted
2002-06-03 16:49:15: Client "192.168.1.1:1098" ready
2002-06-03 16:49:15: Client "127.0.0.1:34677" ready
2002-06-03 16:49:15: Sending info to "127.0.0.1:34677"
2002-06-03 16:49:15: Sending info to "192.168.1.1:1098"
2002-06-03 16:49:16: Game start
```

And at that point, the game should start 8-)

5.2.4 Restart a new game

Once the game is over, you can start another network game on the clients without touching the server, because the server automatically restarts and waits for players to connect.

To stop the server - if you want to change its settings for instance - just go to the console where it's running and press CTRL-C.

5.3 Using the meta-server

5.3.1 Basics

The meta-server is a piece of software which is running on my web site, and allows servers to register themselves so that client can get a list of available servers.

It's written in PHP and is _very_ basic but I believe it's enough for what has to be done: maintain a list of running servers.

The source code for the meta-server is included in the source package of Liquid War, so you might run such a server yourself if you want to. However, by default, servers will register themselves on my web site, and will be listed on http://www.ufoot.org/liquidwar/metaserver.php3

5.3.2 How to register a server

Launch the server, and when you get the question:

```
Register on "www.ufoot.org/metaserver/" (y/n)?
```

answer "y".

Note that if you're behind a proxy or a firewall, the server might be unable to register itself. Clients might also have problems to connect themselves on your server if there's a machine which does NAT (Network Address Translation) between you and the meta-server.

5.3.3 How to find a server

In the main menu, click on "Net Game" and then "Search for internet games".

Now you should see a list of available servers. You can click on the items in the list to get more informations about a given server. Once you have chosen a server, click on "Join now".

Now you get on a "Waiting for teams" screen. You might be interested in using the "Start now" button. Indeed, if you are 4 players connected on a server that accepts up to 6 players, maybe you'll want to start the game right away without waiting for 2 more players. In this case, every player must click "Start now". A "*" character will replace the "-" in the players list when a player clicks on "Start now". When all the players are displayed with a "*a, the game starts.

You can also chat with other players by entering text in the area above the "Send message" button, and then click on this button. Keep in mind that this is a very primitive chat and that the best way to chat efficiently is IMHO to play in windowed mode and have an IRC client at hand.

Note that you can also get the list of available servers from http://www.ufoot.org/liquidwar/metaserver.php3 There you'll also find a little chat-box which will allow you to send exchange messages with other players.

5.4 Options

5.4.1 Server options

You can pass options to the server using the command line. The following parameters are accepted:

- "-n" where "n" is a number between 2 and 6 : with this option you can tell the server how many teams will connect to the game. Beware, there can be several teams on the same computer, so if you want to have a computer with 2 players on it and 2 other computers with a single player, then you need to use the "-4" option.

- "-lag n" where "n" is an integer : with this option, you can control the lag used at startup. Normally, Liquid War handles this parameter automatically, but you might want to force it to a given value.

- "-port n" where "n" is an integer : allows you to change the IP port used by the server to listen to the clients. if you omit this parameter, the default port is (8035) is used.

- "-log file.log" : dumps all informations in "file.log" instead of using the standard output.

- "-public" : skips the "Register on ..." question, and registers the server automatically on the meta-server, so that clients can find it easily.

- "-private" : skips the "Register on ..." question, and does not register the server at all.

- "-comment This_is_a_comment" : associates a comment to the server, which will be displayed by the meta-server. Note that the character "_" will be replaced by spaces. This makes command line parsing easier. I'm lazy 8-)

- "-password xxx" : associates a password to the server. With this option, clients will need to give the right password to be able to connect on the server.

- "-callback cmd" : with this option, the command "cmd" will be launched whenever someone connects on an empty server. For instance, if you want to run a permanent server and want to know when someone connects but do not wish to permanently keep an eye on the server's log, you might use this option and put in "cmd" a command that automatically sends you a mail. You might also use a command that fires a popup window. It's up to you. 2 sample scripts are available in the "misc" directory of the Liquid War source distribution. One is "misc/liquidwar_notify.sh" and works on UNIX platforms, and the other one is "misc/lwpopup.js", which is automatically used by "misc/lwserver.bat", and works on Windows.

5.4.2 Common options

These options work on both client and server, even if they are rather "server-oriented".

- "-metaserver url" : redefines the URL of the meta-server. Usefull if you want to use your own meta-server.

- "-netlog" : if you use this option, the server will dump all the network traffic on the standard output. This is usefull for debugging.

- "-nobeep" : Disables the system beeps that the application might fire. These beeps are mostly used on the server to notify clients connections.

5.5 About Liquid War's network implementation

5.5.1 Basics

Liquid War uses TCP sockets, and a single-threaded server. This implies that:

- The game can sometimes get blocked if you play on Internet.
- The server can't talk simultaneously with several clients.

I needed to use TCP sockets, since LW's algorithm can not cope with any data loss and it's not a reasonnable to try and anticipate what the map would be like if the player did not move etc...

I did not implement any complex multithreaded stuff since I'm lazy and however, clients need to have informations about all the other before something can be done. However, implementing a mutltithreaded server could have advantages over the current solution.

5.5.2 What is this lag stuff anyway?

In Liquid War, all the clients send their key presses to the server, and then the server dispatches this information to everyone. This has to be done for every round.

You can easily imagine that if a player has a poor connection, with a very long "ping delay", it can take quite a long time to send the information to the server, and then get it back.

So what Liquid War does is that at the beginning of the game, the server sends a couple of "blank" key strokes to the clients. This way, clients receive data from the server before thay have sent any. The number of key strokes sent at the beginning of the game is called the "lag".

So if it takes 200 msec to send and then receive data from the server (approx the time returned by the "ping" command) then with a lag of 6, you can theorically play at a rate of (1/0.2)*6=30 rounds/sec.

On one hand, setting the lag parameter to a high value will avoid many network errors and allow you to play at a very fast pace, but the big drawback is that there will be quite a long time between the instant you send a key stroke to the server and the moment it comes back to you. On the other hand, setting the lag to a low value will limit drastically he number of rounds per second, but make the game more "responsive".

However, since release 5.4.1, the "lag" is modified automatically and should adapt itself to the situation. I've not been able to test it in real conditions yet, but it should work 8-)

Still, setting the lag to a sensible default value can save you some trouble. Indeed, by default, Liquid War will choose a value (6), but it can not guess if you are playing on Internet or on a 100 Mbit LAN, and it can take quite a long time before Liquid War automatically finds the right value. To know the right value which should be used with the "-lag" option, simply play a few games and watch the average lag (which is displayed on the server console every minute) at the end of the game.

5.5.3 Performance issues

Liquid War uses a "light" server, and one of the advantages of this solution is that it allows you to run the server on low-end computers. I personnally run a permanent server on a 486 DX2, and it runs like a charm.

The only thing you have to take care of when running a server is bandwidth. Don't worry, you won't need a 10Mbit connection, basically, each clients sends and receives 12 bytes of data at each round. If you add TCP/IP headers and the facts that stuff will probably be bundled in bigger packets, a client must deliver about 15 Kbit/sec (up and down) for a game that runs at 100 frames/sec. A 56K V90 modem is enough for this.

So if you run a server with 2 clients connected, the server will need to deliver 30 Kbit/sec in both ways. A 56K V90 modem _can_ do that, but your provider needs to be a good one 8-)

And if you run a server with 6 clients, you simply won't be able to reach the 100 frames/sec with a 56K V90 modem. It will necessarly drop to something less than 30 frames/sec, and is likely to drop to about 15 frames/sec. OK this is not a big deal, since few Internet games run at more than 30 frames/sec, but well, if the server has troubles receiving/sending data, everyone will wait, and the fun will go away.

As a conclusion: if you have the choice, choose the friend who has the best bandwidth to run the server, without even considering the power of his computer.

5.6 Troubleshooting

5.6.1 General information

Network support in 5.4 and 5.5 is still experimental in many ways, so you might get weird behaviors. Basically, if you have a problem, just do the following:

- Stop and restart the server when something goes wrong. To stop it, use CTRL-C.
- Check out that you have entered the correct IP addresses.
- Try and start the client and the server using the "-netlog" option to have an idea about what's happening.

5.6.2 Bugs in 5.4.x corrected in 5.4.2

Liquid War 5.4.0 and 5.4.1 were very hard to play over the Internet. The reason is that the network routines did not do enough error checking, and therefore there were very often errors when sending and/or receiving the map to the server. Hopefully, this bug should not appear anymore in 5.4.2 or any other recent release.

5.7 About security

5.7.1 Network games passwords

As you might have noticed, under the box where you can enter the password, a little notice explains that you must choose a "weak" password. Now you'll tell me -> people keep on explaining me that passwords must be something complex like "aS\r!Y9p" and now I'm told to use "hello", what's up?

OK, keep in mind Liquid War is a game. This password stuff is just a way to be able to play with your friends only and keep on using the meta-server's services. Liquid War does not encrypt data and I can see no good reason to do it for, so the password is stored and sent to the server in clear, as plain text.

The consequence is that if you use a valuable password - for instance the one you use to log in on your computer - the guy who runs the server will see your password in the log file if he wishes to. Therefore, use something weak, something that if someones finds out what it is, you won't really care. So "hello" is a wise choice.

5.7.2 Is Liquid War likely to have security holes?

Yes.

Any program is likely to have security holes, especially when it's networked. However, I have good reasons to think that Liquid War is safe enough for a game. At least I find it safe enough to run a permanent public server on my personnal computer 8-)

FYI, here are some things which I think make Liquid War rather safe to run:

- Liquid War does not store anything on your hard drive that would have been received from the network. The maps are kept in RAM. So you won't download any virus playing Liquid War on Internet.
- Liquid War does not transmit any sort of code on the network. All the transmitted bytes represent plain data. So you're not likely to execute any arbitrary code - virus, worm - when playing on the Net.
- Liquid War receives network packets in static buffers, and if the received data is too big, it is truncated. One consequence is that Liquid War has a bunch of "limits". You can't send hudge maps over the network, you can't have long nicknames, and so on. But another consequence is that if you try to send garbage in the buffer, it will be truncated. Liquid War will protest with a "network error" message and the connection will be closed, but there will be no easy exploit possible here.
- Liquid War does not use the sprintf, strcpy and strcat functions, which are known as being rather unsecure since they can lead to buffer overflows. Instead, it uses the equivalent functions snprintf, strncpy and strncat. On platforms where these functions are supported natively, the game will use the default system functions, otherwise it will use a free implementation by Mark Martinec. FYI, Windows does not support snprintf-like functions natively, that is, it's a piece of crap.
- Liquid War is Free Software, so I'm not likely to have put backdoors in it myself, since anyone can look at the source code 8-)

However, I have not - and I know nobody who has - audited Liquid War for security holes. So there might be some. Therefore you should respect a few things while running Liquid War:

- Never run Liquid War as root or administrator. This is obvious but I still mention it. If you want to run a Liquid War daemon on UNIX, run it as user "nobody" or something approaching. If "root" or "administrator" does not make sense on your system (DOS, Win98...) then I assume you're not _really_ concerned about security anyway 8-P
- If you run a server 7/7 24/24, use the "-log" option to log everything in a file. This way you'll keep a trace of network activity, and if something goes wrong, you might get a chance to see it.
- If you use passwords in network games, *never* choose a valuable password. Use something simple like "hello" or "goodbye".

- Keep in mind that Liquid War is a game, and not a bullet proof professionnal server.

Last point: you should be aware that version 5.4.5 of Liquid War has been proved to be vulnerable to a local buffer overflow, and one should run at least 5.5.9 to get rid of this problem. FYI by the time the exploit was found on 5.4.5, 5.5.9 was already out 8-) See http://www.securityfocus.com/bid/8629 and http://www.securityfocus.com/bid/9453 for more informations.

5.7.3 Can people cheat when playing on the Net?

No.

Or at least, not really. In fact, you can still find the following types of lamers:

- A guy who lets the CPU play at his place. He'll loose anyway because the CPU is definitely not a great Liquid War Master 8-)
- A guy who tweaks the game and gets all his bots fight anyone he wishes. That's mean.
- A guy who manages to let you have a 500msec lag while he does not have any lag at all.

Apart from this, I can hardly see any way to cheat. Why? Because the Liquid War server does not store any information about the game. It's not aware of who wins, who looses, it knows nothing. The only thing it does is to transmit key presses between client computers.

This way, if someone plays with a tweaked release of Liquid War, thinking he will fool you, then he will fool you on his computer only... On your computer, everything will be fine. After some time, your screen and his screen will have nothing in common, and both players are likely to think they have won. Except the lamer will stay a lamer.

This also explains why it's required to play with the very same versions of the game during network games. If you plug a 5.5.2 with a 5.5.1, after a minute the screens will be completely different on each client, since there are subtle differences between the 5.5.1 and the 5.5.2 engine. However, you shouldn't be able to do this, since a network error will stop you before you can start to play.

Additionnally, versions 5.5.5 and higher have a checksum system. Every 100 rounds, each client calculates a checksum with its local map, and sends it to the server. If the checksum is incorrect, the server will log a message like:

```
Checksum error on client "192.168.1.1:1098"
```

If you see this, then you're in one of the following situations:

- There's a bug in the game
- A lamer tries to cheat

FYI, all releases from 5.4.0 to 5.5.4 have a bug which causes clients to desynchronize after a while...

Chapter 6

Command line parameters

6.1 Introduction

When you launch Liquid War 5, you can use command line options. If you have no problems launching Liquid War, this section should not interest you very much.

You can use several options at the same time. The basic syntax for options looks like this:

```
lw -option1 -option2 parameter2 -option3 parameter3 -option4 -option5
```

Note that most of the options are legacy options which where usefull with the initial releases of Liquid War, when you had to run in a Windows DOS box, and when there were still plenty of 486 computers with only 8Mb ram...

6.2 Version checking

These are basic options which can be usefull to figure out which release of Liquid War is installed.

- "-v" : returns the version number of the program.
- "-h" : displays a short description and copyright information.

6.3 Changing default paths

Very usefull options, especially if you can not install Liquid War in default directories or want to put the game in a special place.

- "-cfg myconfigfile.cfg" : causes Liquid War to use the specified config file.
- "-dat mydatafilefile.dat" : causes Liquid War to use the specified datafile. This might be a very interesting option if you run Liquid War on a GNU/Linux box where you do not have root access and therefore can not put the datafile in /usr.

- "-map mycustommapdir" : causes Liquid War to use the specified directory as the user map directory. The user map directory is where you can put plain bitmaps to be used as maps.
- "-tex mycustomtexturedir" : causes Liquid War to use the specified directory as the user texture directory. The user texture directory is where you can put plain bitmaps to be used as textures.
- "-mid mycustommusicdir" : causes Liquid War to use the specified directory as the user music directory. Any midi file placed in this directory will be added to the list of available musics.
- "-server myliquidwarserverfile" : causes Liquid War to use the specified file as the server executable. This option has no effect for now since the server is not launched by the client.
- "-mapgen mylwmapgenfile" : causes Liquid War to use the specified file as the "lwmapgen" executable. "lwmapgen" is an utility by David Redick which generates random maps automatically for Liquid War. You might wish to change this option if random map generation does not work for some reason, or if you want to use your own map generator.

6.4 Troubleshooting switches

These options give you control on how Liquid War treats initialisation errors, how much memory it should reserve, what kind of video mode sit should not choose etc...

- "-vga" : This option forces Liquid War to use your video card as if it was only a basic VGA card. This option is required if you play Liquid War from Windows NT.
- "-no400300" : This option disables the VGA 400x300 video mode. I created this options for I know that some video cards/monitors don't support the 400x300 mode.
- "-silent" : With this option, Liquid War will not play any sound. It will not search for any sound card. This can be interesting if you don't have any sound card or if Liquid War doesn't handle your card correctly.
- "-nowater" : Causes Liquid War not to load any water sound. Use this if Liquid War runs short of memory, and you should gain about 850kb.
- "-nosfx" : Causes Liquid War not to load any sound fx. Use this if Liquid War runs short of memory, and you should gain about 150kb.
- "-nomusic" : Causes Liquid War not to load any midi music.
- "-mem n" : The parameter "n" sets the amount of memory (in Mb) Liquid War will allocate to do all its calculus. If this number is too small, you won't be able to play on all the levels. If it is too high, Liquid War may not start at all or crash while you are playing. The default value is 8. If you play Liquid War from Windows and Liquid War refuses to run because this parameter is too high, then try and give more dpmi memory to Liquid War.
- "-nojoy" : This option disables joystick support.
- "-noback" : Causes Liquid War not to load the background image. Use this if Liquid War runs short of memory, and you should gain about 300kb.
- "-notex" : Causes Liquid War not to load any texture. Use this if Liquid War runs short of memory, and you should gain about 750kb.

- "-auto" : If you set this option, Liquid War won't generate any error while allocating memory or loading data.
- "-safe" : With this option, you will play with a very reduced version of Liquid War. It looks rather ugly but should work in a DOS box with only 4Mb of DPMI memory. Use this if you experience serious memory or device problems. If Liquid War doesn't start with this option turned on, I really don't think I can do anything for you...
- "-nice" : With this option, Liquid War will use a mode which is between the default mode and the "safe" mode.
- "-check" : With this option, Liquid War will stop as soon as it detects something strange while initializing.
- "-stop" : If you set this option, Liquid War will prompt you for a key when the init process is completed.
- "-c" : This is a weird option, if you turn it on, the game will only use fonctions which are programmed in C langage. The default behaviour is to use some functions I rewrote in assembly langage, so that the game is a little faster.

6.5 Debug options

These options are usefull if you want to debug the game and trace what's happening.

- "-netlog" : Dumps all the network traffic on the standard output. This can help finding problems when trying to connect to the server in a network game.

6.6 Server options

The server command line options are described in the section which deals which network games in general. IMHO you need to understand how network works in LW before trying to tweak the server 8-)

6.7 Other options

Everything else 8-)

- "-capture" : Activates the capture mode. In this mode, the game will dump a .bmp file on the disk several times per second, which is usefull if you want to create an mpeg movie of your game session afterwards. You can also activate this mode interactively by pressing F9 within the game.
- "-metaserver url" : redefines the URL of the meta-server. Usefull if you want to use your own meta-server.
- "-nobeep" : Disables the system beeps that the application might fire. These beeps are mostly used on the server to notify clients connections.
- "-tombola" : Activates a special mode where scores are not displayed normally. Instead, the game displays 3 random numbers between 1 - 500.

6.8 lwmapgen options

The external lwmapgen program by David Redick accepts a number of command line options. You can type:

```
liquidwar-mapgen --help
```

to get a list of all available options under UNIX. Under Windows the command would be:

```
lwwinmap --help
```

Note that lwmapgen is called automatically by Liquid War when you click on the "Random map" button.

The idea behind lwmapgen is that you choose a function which sets the type of map generated, and then pass various arguments to this function. If you do not explicitly choose a function, one will be picked up for you randomly, and all other parameters will use default values.

The most usefull and common options are:

- "–help" : displays help.
- "–out filename" : save bitmap to filename.
- "–size n" : bitmap size (an integer between 0 and 5).
- "–list" : list all available functions.
- "–function function_name" : which function to use.

Here's a typical use of lwmapgen:

```
liquidwar-mapgen --size 3 --function hole --out test.bmp
```

Chapter 7

Platform specific issues

7.1 General remarks

Liquid War is now a cross-platform game, thanks to Allegro. So now you can play under different OS.

The same source tree will compile on all supported platforms, but with slight differences when running. C preprocessor #defines are used to code some platform specific stuff, and in some cases there are different files for the DOS, Windows and UNIX versions.

As I said, I try to use the same code for all platforms. This is in the long term the best choice. Otherwise there would different branches of the source tree, and I don't think this is a very good solution.

Therefore some optimizations that were performed in the old DOS-only version have been totally removed, for they were 100% platform dependent (ie mode-X asm coding). So the new versions are all a little slower than the old 5.1 stuff, but the performance loss is only about 20%, which is not significant with today's PCs. And anyways the performance loss is most of the time limited to the goog old VGA 320x200x8 mode-X, which starts being kind of obsolete.

7.2 DOS

This is the original version. It's the fastest one as far as I know, the safest one and it will always be I think, since Allegro was first designed for DOS, and DOS allows a full unconditionnal access to all the hardware ressources LW requires. LW doesn't use any hardware acceleration and it's not been designed to do so. Unfortunately there's no network support for the DOS version of Liquid War.

7.3 Windows

When running under a Windows box, the DOS release used to be safer than the native Windows port. Now that DOS support is getting really poor with recent versions of Windows, the native Windows release of Liquid War starts begin the good choice for Windows users. And Allegro for Windows is getting quite stable in the 4.x series.

The other reason to choose this release rather than the DOS release is that it has network support.

If you have problems running Liquid War under Windows, please check out the "data\lwwin.log" file which should be written each time you run the game. It contains the information which is displayed on the console under other platforms, and might give you a clue about what's going wrong.

7.4 GNU/Linux

This port is the most recent one, and also the one I prefer. Paths have been changed to an UNIXish style, ie the data is stored in:

```
/usr/local/share/games/liquidwar
```

the executable in:

```
/usr/local/games
```

and the configuration file is

```
~/.liquidwarrc
```

Since not all GNU/Linux distributions have /usr/local/games in their path, I also put a symbolic link to the binaries in /usr/local/bin. I believe Liquid War is quite FHS compliant, so if its default directories do not match your configuration, blame your distro for not following the standards 8-) AFAIK the only touchy directory is /usr/local/share/pixmaps which I've seen on many distribution but does not seem to be referenced in the FHS.

With the latest releases of Allegro, Liquid War is becoming pretty stable under GNU/Linux. You should also know that the GNU/Linux port is usually the most up to date, since I very very seldom boot Windows at home and do most of the coding under GNU/Linux.

7.5 FreeBSD

This is the latest port, so I expect it to be a little touchy to install and/or run for some time.

Note that to compile the game you'll need to install GNU tools like gmake and gcc. Liquid War won't compile with the genuine make and cc commands.

One thing you might ask is: "why do you provide the binary as a plain .tgz file, it would be much nicer if a standard FreeBSD port was provided instead!". The answer is that the statically linked binary should work flawlessly and does not raise any dependency problem. Also I don't know how to make a BSD port and I'm not really interested in doing it. If it's easy to do, then someone can simply do it and send it back to me. If it's hard to do, then I do not really have the time nor motivation to do it. What I did is make the code and install scripts FreeBSD friendly so that it would be possible to compile the game under FreeBSD. Packaging is another story.

7.6 Mac OS X

There's currently a beta version of a Mac OS X port for Liquid War. Ryan D. Brown nicely managed to compile and run the game under Mac OS X, and the latest news was that it does basically work. Still, this port did not go through intensive testing, so there might still be some bugs, expecially concerning networking.

There were some byte endianess problems in previous (<=5.5.8) releases of LW, but I tried to fix them and they should be gone now.

As of today, we're trying to find out a convenient way to package and release the Mac OS X version of LW. You can contact us on the mailing list if you're interested in this port.

Chapter 8

User levels

8.1 A piece of advice

You can use your own levels whith Liquid War 5. The only thing you have to do is to put your own 256-colors vbitmap files in a special directory, and the program will use them. Currently, BMP, LBM, PCX, and TGA files are supported. It is a good thing to use 256 colors bitmaps, for they waste less disk space than truecolor bitmaps, and Liquid War 5 converts all bitmaps to 32 colors bitmaps. Additionnally, truecolor bitmaps might cause the DOS version to crash randomly... 2-color bitmaps will also cause the program to crash. I warned you!

The best thing you can do to create your user levels is to have a look at the few user files I put in the .zip file and try at first to do something that looks about the same!

8.2 Maps

Liquid War 5 does many checks on user levels and is much safer than Liquid War 3. Still, try and help the program not to crash, if possible.

Liquid War considers that dark colors are walls and bright colors are associated to the playable area. So you can draw your walls in black, dark blue, etc... And the rest of the map can be of any bright color such as white or yellow.

You can draw a small map on a big bitmap, as long as you use a bright background color. Liquid War will autodetect the range of your map and add the border line if necessary.

Liquid War re-orders all the maps, so that the smallest ones are on the left and the most complicated ones on the right when you choose them with the slider in the "map" menu. So if you can't find the map you just draw, don't worry, it is probably just mixed with the levels from the .dat file.

The default path for maps is "custom\map\" on windows, and "/usr/local/share/games/liquidwar/map" on GNU/Linux.

8.3 Textures

All you have to do is put a bitmap in the default directory which is "custom\texture\" on windows, and "/usr/local/share/games/liquidwar/texture" on GNU/Linux.

8.4 Textures

As of Liquid War 5.6.x, it's possible to associate a map with a texture. All you need is to call the maps with the same name (case is important, lowercase recommendend...). Look at the "meditate" custom map which is shipped with Liquid War for instance.

This also works with internal builtin maps (those stored in the datafile) so for instance if you name a custom texture "world1.bmp" it will automatically be associated to the builtin map world1 (to figure out internal names you have to unpack the source distribution and look in the ./data directory). This also works the other way, if you name a map "wood2.bmp" it will be automatically associated with the builtin texture wood2.

Associating a texture with a map requires more work than designing a simple map, but the results is usually much nicer. See the "Kasper - ..." series of maps for instance.

8.5 Send your levels

Maybe you will find that the original levels are ugly and unplayable. Well, if you have made user levels and think they are great, just send them to the Liquid War user mailing list. Please use only 256 colors bitmap and zip them before sending them, or else they might be blocked by my provider...

As of today, dozens of user maps have already been included in Liquid War, this is very nice for it happens that every map designer has his own personnal tastes, so the maps all look different.

Still, to be included in Liquid War's mainstream distribution, your maps will need to be placed under the terms of the GNU General Public License, or at least a compatible license. You should have received a copy of this license with Liquid War anyway. Read it 8-)

Of course, you can use *any* map when playing. You can even play with a bitmap you got from a proprietary source - such a proprietary game you bought for instance - but the point is that I can't - and you can't either - distribute such a map along with Liquid War.

However, this is enough legal boring stuff! What you should keep in mind is that I'm always happy when I receive maps from players, and it's a pleasure for me to include them in the mainstream distribution.

Chapter 9

Core algorithm

9.1 Introduction

9.1.1 General remarks

If you have played Liquid War, you must have noticed that your army always takes the shortest way to reach the cursor. So the fundamental stuff in Liquid War is path-finding. Once you've done that the game is quite easy to code. Not harder than any other 2D game. Still the path finding algorithm is an interesting one, for it's not a common method that we used.

Basically, at each round (by round I mean a game logical update, this occurs 10 or 100 times/sec depending on the level and/or your machine), the distance from all the points of the level to your cursor is calculated. Now the point is to calculate this fast, real fast. In fact, a "gradient" is calculated for all the points of the level, and the value of this gradient is the distance required for a little pixel/fighter to reach your cursor, assuming that he takes the shortest way. Liquid War does this with a 10% error tolerance, and it's enough for keeping the game interesting.

Once you have this gradient calculated, it's not hard to move your fighters. Basically, you just have to move them toward the adjacent point that has the lowest gradient value, ie is the closest to your cursor.

9.1.2 History

The Liquid War algorithm has been invented by my friend Thomas Colcombet In fact the Liquid War algorithm has been invented before the game itself. The game came as a consequence of the algorithm, he just thought "mmm, cool, we could make a game with that!".

Later, I enhanced the algorithm, as I coded it. The consequences were a performance increase, especially on simple but big levels. I mean levels with wide areas for teams to move. Still the basis of the algorithm remained the same.

9.1.3 Pros

The Liquid War algorithm for path-finding is very efficient:

- When you have to move lots of different points toward one single point. Good thing that's the rule of Liquid War!
- When you have no clue about how your map will look like, ie if the walls are randomly placed. The complexity of the level doesn't influence much the speed of the algorithm. The size does, but the complexity, ie the number of walls, is not so important.

9.1.4 Cons

The Liquid War algorithm is very poor compared to other algorithms when:

- You have several target destinations, that's to say Liquid War would be really slow if there were 100 teams with 10 players only.
- You want to move one single point only.
- > You want the exact (100% sure) path. In fact, this algorithm finds solutions which approach the best one but you can never figure out if the solution you found is the best, and the algorithm never ends. In the long term, the algo will always find the best solution or something really close but I don't know any easy way to figure out when you have reached this state.

9.2 Mesh

9.2.1 Introduction

The first Liquid War algorithm used to calculate the gradient (the distance from a point to your cursor) for every single point of the map.

With Liquid War 5, I used a mesh system. This mesh system is a structure of squares connected together. Squares may be 1,2,4,8 or 16 units large or any nice value like that, and the gradient is only calculated once for each square. Squares have connections between them, and each connection is associated to a direction.

There are 12 directions:

- North-North-West (NNW)
- North-West (NW)
- West-North-West (WNW)
- West-South-West (WSW)
- South-West (SW)
- South-South-West (SSW)
- South-South-East (SSE)
- South-East (SE)
- East-South-East (ESE)

- East-North-East (ENE)
- North-East (NE)
- North-North-East (NNE)

9.2.2 Example

Well, let me give you an example, supposing that you level structure is:

```
**********
*        *
*        *
*       **
*        *
**********
```

The * represent walls, that's to say squares where fighters can not go.

Then the mesh structure would be:

```
**********
*11112233*
*11112233*
*1111445**
*i1114467*
**********
```

In this mesh, there are 7 zones:

- zone 1 has a size of 4. It's linked with zones 2 (ENE) and 4 (ESE).
- zone 2 has a size of 2. It's linked with zones 3 (ENE,ESE), 5 (SE), 4 (SSE,SSW) and 1 (SW,WSW,WNW).
- zone 3 has a size of 2. It's linked with zones 5 (SSW), 4 (SW) and 2 (WSW,WNW).
- zone 4 has a size of 2. It's linked with zones 2 (NNW,NNE), 4 (NE), 5 (ENE), 6 (ESE) and 1 (WSW,WNW,NW).
- zone 5 has a size of 1. It's linked with zones 3 (NNW,NNE,NE), 7 (SE), 6 (SSE,SSW), 4 (SW,WSW,WNW) and 2 (NW).
- zone 6 has a size of 1. It's linked with zones 5 (NNW,NNE), 7 (ENE,ESE) and 4 (WSW,WNW,NW).
- zone 7 has a size of 1. It's linked with zones 5 (NW) and 6 (WSW,WNW).

9.2.3 Why such a complicated structure?

Because it allows the module which calculates the gradient to work much faster. With this system, the number of zones is reduced a lot, and calculus on the mesh can go very fast. At the same time, this mesh structure is complicated to understand by us humans but it's very easy for the computer.

9.3 Gradient

9.3.1 Introduction

For each zone defined in the mesh, LW calculates an estimation of the distance between the cursor and this zone.

The algorihm is based on the fact that to cross a zone which size is n, n movements are required. Easy, eh?

9.3.2 Description

Here's the way the algorithm works:

for each turn of the game, do:

- pick up a direction between the 12 defined directions. They have to be chosen is a peculiar order to avoid weird behaviors from fighters, but let's suppose we just pick up the "next" direction, ie if WSW was chosen the last time, we pick up WNW.

and then for each zone in the mesh, do:

- Compare the potential of the current zone with that of its neighbor zone. The neighbor zone to be chosen is the one which corresponds to the direction which has been previously picked up, and by potential I mean "the distance to the cursor, estimated by the algorithm's last pass".
- If potential_of_the_neighbor_zone > (potential_of_the_current_zone + size_of_the_current_zone) then potentiel_of_the_neighbor_zone = potential_of_the_current_zone + size_of_the_current_zone

9.3.3 How can this work?

Well, just ask my friend thom-Thom, he's the one who had the idea of this algorithm!

The basic idea is that by applying this simple rule to all the zones, after a certain amount of time, it's impossible to find any place in the mesh where the rule is not respected. And at this time, one can consider the potiential is right in any point.

Of course when the cursor moves the potential has to be recalculated, but you see, cursors move really slowly in Liquid War, so the algorithm has plenty of time to find a new stable solution...

9.3.4 Demo

It's possible to see this algorithm working by typing:

```
ufootgrad[n]
```

while playing, where [n]is the number of the team the gradient of which you want to view. The game is still running but you view a team's gradient being calculated in real time instead of seeing the fighters.

If you type ufootgrad0 the display comes back to normal mode.

9.4 Move

9.4.1 Introduction

Once the gradient is calculated for any zone on the battlefield, it's quite easy to move the fighters, hey?

The following method is used to move the players:

- A "main direction" is chosen for the fighter, this direction is chosen using the gradient calculated on the mesh.
- Knowing which direction is the main one, a "level of interest" is applied to the 12 defined directions.

There are 4 "level of interest" for directions:

- Main directions: the direction calculated.
- Good directions: these directions should lead the fighter to the cursor.
- Acceptable directions: ok, one can use this direction, since the fighter shouldn't loose any time using it.
- Unpossible directions: wether there's a wall or using this direction means the fighter will be farer from his cursor than before, it always means that this direction will not be used, never.

9.4.2 Rules

The fighters will try to find any matching situation in this list, and chose the first one.

- The main direction is available, no one on it, OK, let's follow it.
- There's a good direction with no one on it, OK, let's follow it.
- There's an acceptable direction with no one on it, OK, let's follow it.
- The main direction is available, but there's an opponent on it, I attack! By attacking, one means that energy is drawned from the attacked fighter and transmitted to the attacker. When the attacked fighter dies, he belongs to the team which killed him.
- A good direction is available, but there's an opponent on it, I attack!
- The main direction is available, but there's a mate on it, I cure him. That's to say that energy is given to the mate. This way, when there's a big pool of fighters from the same team, they re-generate each other.
- None of the previous situations found, do nothing.

9.4.3 Tips and tricks

The behavior of the armies is quite tricky to set up. I had myself to try many algorithms before I came to something nice. In fact, I had to introduce some "random" behaviors. They are not really random for I wanted the game to behave the same when given the same keyboard input, but for instance, fighters will prefer NNW to NNE sometimes, and NNE to NNW some other times. By the way, I think Liquid War could stand as a nice example of the thoery of chaos.

Chapter 10

Source code

10.1 General remarks

10.1.1 Modularity

Liquid War 5 is basically a big C program. I've splitted the source code in many small files for I do not like to have to handle big monolithic sources, but this does not mean Liquid War is very modular. In fact Liquid War 5 is quite bloated with global variables and other ugly stuff 8-(

10.1.2 Coding style

To be honest, it's a big mess. You won't find 2 files coded in the same maner... OK, I'm exagerating a bit. From now I try to make an effort and stick to basic rules such as:

- use the GNUish-style indentation - the default Emacs mode in fact
- prefix global functions / variables / constants / types with lw_<NAME_OF_THE_file>_. For instance, a "do_it" function in myfile.c will be called lw_myfile_do_it
- use capitals for constants, globals and types only. All functions are in lowercase with "_" to separate words
- keep on using 8.3 filenames for .c source files. This is for better DOS integration. DOS version of Liquid War is still maintained, you know 8-)
- use English only for code and comments

I might decide to rename and cleanup everything some day, for it would help other coders to understand what I wrote, but well, this is certainly not a thrilling task 8-/

10.2 Source files organization

10.2.1 Main game code

Here you'll find the main() function, the main game loop, application-wide constants and other global stuff.

It might be a good start if you want to hack the code.

- base.h: contains global constants used in many different files.
- game.c / game.h: contains the main game loop.
- main.c / main.h: the file where the main C function is declared. Doesn't contain much except calling init functions and running the GUI.

10.2.2 Menus

The menus are coded using the Allegro GUI system. While this system is very powerfull, it's IMHO not adapted to very complex GUIs, and one of its drawbacks is that it's not so easy to redesign something once you've coded it.

Besides, when I started coding the GUI in 1998, I did it in a rather ugly way, and now I'm paying for my being lazy at that time, since I spent hours coding when I want to change something 8-/

- about.c / about.h: contains the code for the about menu.
- advanced.c / advanced.h: contains the GUI advanced options menu.
- connect.c / connect.h: contains code for the "connect" menu which displays which players are connected to the server, before the game actually starts.
- controls.c / controls.h: contains the code for the controls menu.
- graphics.c / graphics.h: code for the graphic options menu.
- internet.c / internet.h: contains the code for the "Search for Internet games" menu, where one can pick up a running server automatically with the help of the meta-server.
- language.c / language.h: contains the code for the "Language" menu.
- level.c / level.h: contains code for the menu where the player can select a level and its options (texture or color).
- menu.c / menu.h: contains the code for the main menu.
- netgame.c / netgame.h: contains the code for the net game menu.
- options.c / options.h: contains the code for the options menu.
- play.c / play.h: contains the code which ties the menu to the main gameloop.
- rules.c / rules.h: code for the rules menu.
- score.c / score.h: functions to display the scores at the end of the game.

- speeds.c / speeds.h: contains the code for the speeds menu.
- team.c / team.h: code for the team menu, where one choses which teams will play.
- volume.c / volume.h: code for the sound menu.
- wave.c / wave.h: code for the wave menu.

10.2.3 GUI tools

These files contain various utilities which are used in the menus.

- alleg2.c / alleg2.h: contains some tweaked allegro functions. I wanted to use bitmaps with sevral colors for my fonts, and change some of the allegro default behavior. So rather than modifying the allegro source code right in the library I copied it in this file and then modified it.
- back.c / back.h: this modules displays the background image.
- dialog.c / dialog.h: contains code for standard dialog boxes.
- error.c / error.h: contains functions to display error messages once the game is in graphical mode.
- help.c / help.h: generic functions to display the various help pages.

10.2.4 Core algorithm

Here's *the* interesting part. All the rest of the code is just sugar coat to display stuff, receive players commands, communicate with other computers, handle errors, etc... But the real thing is here!

It's funny to note that these files have almost not been modified since Liquid War 5.0.

It's also interesting to note that they represent a small percentage of the total amount of code in the game. This tends to prove - and I'm convinced of it - that game programming does not only consists in having great ideas, but also requires a lot of "dirty" and boring work. Honestly, coding an option menu is as boring as coding Liquid War algorithm is fun.

- fighter.c / fighter.h: contains code to move the armies, once the gradient has been calculated.
- grad.c / grad.h: this module calculates the gradient for each team. One could say it's the "kernel" of the game, since most of the CPU time is spent in this module (except if you have a slow display...).
- mesh.c / mesh.h: contains code to set up a usable mesh with a map. Mesh are re-calculated at each time a new game is started, the reason for this being that meshes are *very* big so it would not be reasonnable to save them directly on the HD.
- monster.s / monster.h: assembly functions to speed-up the game. It's a replacement for some fighter.c functions.
- spread.s / spread.h: contains assembly replacements for some functions of grad.c. These replacements do the same than the original ones from grad.c, but faster. Could still be optimized.

10.2.5 Moving cursors

It looks like nothing, but moving a cursor and deciding where it should go if there's a wall in front of it is not that easy, especially if you want things to work nicely.

- autoplay.c / autoplay.h: contains the code for the computer AI. This module simulates keypresses from the computer, then the computer is handled as any other player.
- move.c / move.h: provides an API to move the cursors.

10.2.6 User input

Until 5.4.0, Liquid War did not have network support. As it is designed to be multiplayer, one needed to have several players on the same computer. The mouse also needed to be handled in a special way since cursors can *not* pass walls in Liquid War. Additionnally, I wanted all input channels (keyboard mouse and joystick) to be handled in a unified way.

This explains why there's so much code for user input, when one would think at first sight that "polling the keyboard is enough".

- joystick.c / joystick.h: contains code to support joystick input. It wraps joystick buttons to virtual keyboard keys, so that joystick and keyboard behave exactly the same.
- keyboard.c / keyboard.h: contains code to handle key presses.
- mouse.c / mouse.h: wraps the mouse movements to virtual keyboard keys. This way the mouse can be used to control the players.

10.2.7 Initialisations

These files contain functions to intialize various game components. 100% boring code.

- area.c / area.h: contains functions to create the game area. Basically it contains functions to create the data structures in which the level is stored during the game.
- army.c / army.h: functions to create the armies, and place them on the battlefield.
- asm.c / asm.h: various constants, macros and utilities to ensure that asembly code works correctly.
- bigdata.c / bigdata.h: I had a really hard time with the malloc function with DJGPP under Win95 dos box. I tried to have it working for hours and hours but my program kept being buggy. So I decided to allocate the memory myself, in a memory zone I create at startup. This is what this module does: create a huge memory zone and then give parts of it to the rest of the program.
- config.c / config.h: contains everything that is related to the game configuration. This module contains in global variables all the parameters that are stored in the config file.
- cursor.c / cursor.h: contains the code to init the cursors and place them on the battlefield at the beginning of the game.

- decal.c / decal.h: This module makes the link between teams and players. Its coding is quite ugly, for some modules in LW assume that when 2 teams are playing they are always teams 0 and 1. So when 3 teams are playing are playing and the second team loses, one has to make team 2 become team 1. That's what this module is for.
- exit.c / exit.h: contains code that is executed when the game ends, it shuts down Allegro and displays messages on the console.
- gfxmode.c / gfxmode.h: contains code to set up the various video modes, and defines which modes are available for each platform.
- init.c / init.h: contains code to initialize Allegro with proper options and analyze failures.
- palette.c / palette.h: contains function to set up the current color palette. Liquid War uses different palettes, depending on what colors are chosen for teams.

10.2.8 Graphics

Here lies most of the graphic functions in Liquid War. There's not that much code since Liquid War's strength is not its visual effects, but rather its gameplay.

The only "funny" thing is the wave effect. I'm quite happy with it, and honestly, I do think it is rather fast, given the fact that it uses no 3D hardware at all.

- disp.c / disp.h: contains functions to display the battlefield.
- distor.c / distor.h: this module contains code to create the "wave effect". It uses a lot of data tables, and is quite complicated to understand...
- glouglou.s / glouglou.h: assembly module, it is a replacement for some functions of distor.c. It goes much faster but does the same.
- info.c / info.h: contains code to display the info bar. The info bar is the bar which display the time left and the amount of players for each team while the game is running.
- message.c / message.h: provides an API to display messages during the game. Very useful if you want to debug the game: you can trace and display anything.
- pion.c / pion.h: contains code to display the cursors.
- viewport.c / vieport.h: code to allocate and resize the zone where the map is displayed, also called "viewport".

10.2.9 Sound and music

Sound and music routines required some encapsulation, since the game must be able to run even if the sound and/or music did not load correctly.

- music.c / music.h: contains the code to control MIDI playback.
- sound.c / sound.h: functions to play sound.

10.2.10 Data management

These functions handle the datafile contents and also the custom data.

Note that the various utilities such as liquidwarcol, liquidwarmap and liquidwartex do not share code with the main executable. This is obviously a design error, for liquidwarmap will handle maps in a very poor way and is unable to autodetect map errors, whereas the game does it rather well. Blame the programmer.

- disk.c / disk.h: contains all the code to access data from the hard drive. In fact, all the HD access is done at startup.
- map.c / map.h: contains code to load the maps from a datafile raw data or a user defined bitmap to a usable structure in RAM.
- maptex.c / maptex.h: contains code to handle the "use default texture" option, and associate a map with a given texture automatically.
- texture.c / texture.h: contains code to handle textures. Textures are stored in a special format which uses 5 bits per pixel.

10.2.11 Random map generator

Liquid War has a "generate random map" feature which is available within the game and also as an external program. The source code for the external program is in ./utils/lwmapgen in Liquid War source distribution. This program has been coded by David Redick, is also available on http://www.cs.clemson.edu/~dredick/lwmapgen/ and works on GNU/Linux. Compiling this program under DOS and/or Windows is untested and unsupported.

The random map generator within Liquid War - which of course works on any platform support by LW - uses for its greater part the same source code as the external lwmapgen program.

- random.c / random.h: wrapper for the map generator written by David Redick. It basically does the same as ./utils/lwmapgen/main.c except that it does it within Liquid War as it is running and not in an external independant program.

10.2.12 Time handling

Time handling is fundamental in a game. Time is used for visual effects (waves...) during the game, it's used to generate some pseudo random stuff, well, it's used everywhere!

Note that on the client, I use 2 "different" clocks. The first counts the "real" time, in seconds. The second one is counts "rounds" and is incremented by 1 at each game round.

- srvtime.c / srvtime.h: code used to handle time on the server, where Allegro's functions are not available.
- ticker.c / ticker.h: sets up a timer callback.
- time.c / time.h: functions to know how long the game has been running, knowing that it can be interrupted.

10.2.13 In-game utilities

These are various utilities use to monitor and control the game while one's playing.

- capture.c / capture.h: code used to capture the video output of the game and store it in .bmp files while playing.
- checksum.c / checksum.h: utilities to generate a checksum from a given game state. Used in network code to make sure all the clients stay synchronized.
- code.c / code.h: This file contains the code to handle key presses during the game. That's to say the pause key for instance.
- profile.c / profile.h: provides tools to calculate how fast the game is runnning and what operations slow it down.
- watchdog.c / watchdog.h: this module waits for "secret codes" to be typed while the game is running, and traps them.

10.2.14 Command line handling

OK, now to all the UNIX guys, I *know* there are many ways to do things in a better and simple way than I did. But keep in mind that in 1998, under DOS, I had a rotten command line and even now I need everything to work on both UNIX and Microsoft platforms.

These utilities are not perfect, but they work, that's all I ask them.

- basicopt.c / basicopt.h: handles basic command line parameters such as "-v" or "-h".
- parser.c / parser.h: contains code to parse and analyze the command line parameters.
- startup.c / startup.h: analyzes the command line parameters and stores them into global variables.

10.2.15 Locale support

Liquid War now has locale support. Basically, all the labels and texts in the UI are stored in constants. There's simply file per language.

Note to translators: if you decide to translate the menus in another language, keep in mind that all the translations must fit in the various buttons and textboxes. The best resolution to test this - the one where letters take most place - is 640x480.

- lang.c / lang.h: contains code to handle language dependant stuff.
- langen.c / langen.h: contains code to handle English specific stuff.
- langfr.c / langfr.h: contains code to handle French specific stuff.

10.2.16 Log and various messages

OK, the API of the log routines is a piece of crap. Now I'm simply too lazy to change it. It works, that's all I ask.

BTW, there's a clear advantage in using custom-made log functions instead of plain calls to "fprintf(stderr,...". It might not be obvious for UNIX users, but think about Windows. Nothing like a "tail -f" there, nor a proper output redirection system. When a user clicks on the Liquid War icon, I want "console" information to be logged in a file!

- log.h: common header for logcli.c and logsrv.c.
- logcli.c: contains code to display messages on the console. It's usefull for console may have different behaviors when the games is used on different platforms. This file is used to compile the client.
- logsrv.c: contains code to display messages on the console. This file is used to compile the server, which does not use Allegro at all.
- popupgen.h: common header for popup functions.
- popupw32.c: code to handle popup on the Win32 platform. Popups are a must-have under Windows for error diagnostics, since the average Windows user never gives any look at any log file...

10.2.17 Macros, utilities and string support

As usual, I needed to prepare a small set of usefull macros.

- macro.h: contains basic wrappers/macros for snprintf like functions. This mostly to ease up string manipulation which is - as always - a nightmare in standard C.
- path.c / path.h: code used to handle filenames and paths, for instance remove path and file extension from a filename.

It's also important to note that Liquid War uses snprintf instead of sprintf, for using the latter is very likely to cause buffer overflows. Under Linux glibc provides this function but Microsoft does not provide it natively on Windows. Therefore I used a third party snprintf implementation by Mark Martinec: http://www.ijs.si/software/snprintf/ and its source is available in the ./utils directory of Liquid War source distribution.

10.2.18 Byte order and endianess

As you might know, PC Intel based computers are "little-endian" while Sun Sparc stations and Mac computers are "big-endian". This is an issue for LW since in network games maps are transmitted in binary format. Therefore I needed to set up some (un)serialization fonctions.

- serial.c / serial.h: code used to transform integers and map headers into an uniform cross-platform byte stream which is readable by both little and big endian machines.

10.2.19 Thread support

Liquid War does have thread support, but it is a "limited" thread support. I mean that the game is generally monothreaded, but a few functions use threads. For instance, calls to the meta-server are done within threads.

Basically, I do not really enjoy programming in a multithreaded environnement. So when possible, I chose the monothread path, and used threads only where I simply would not be able to find another acceptable solution.

I also needed to use some mutexes to prevent crashes in the user interface.

- mutxdos.c: provides fake mutex support under DOS. This module is here only to make compilation easier.
- mutxgen.h: header for mutxdos.c, mutxunix.c and mutxw32.c.
- mutxunix.c: provides mutex support on UNIX.
- mutxw32.c: provides mutex support on Win32.
- thrddos.c: provides fake thread support under DOS. This module is here only to make compilation easier.
- thrdgen.h: header for thrddos.c, thrdunix.c and thrdw32.c.
- thrdunix.c: provides thread support on UNIX.
- thrdw32.c: provides thread support on Win32.

10.2.20 Launching external programs

Liquid War might sometimes launch external programs. This is (for security reason) not a default behavior and has to be activated and configured by yourself, using the "-callback" command line option on the server for instance.

- execgen.h: header for execunix.c and execw32.c.
- execunix.c: code to launch external programs on UNIX.
- execw32.c: code to launch external programs on Win32.
- exec2.c: code to launch external programs within the client, without any interaction with the user, ie no unwanted popping window for instance.

10.2.21 Low-level network code

There are network packages for Allegro, but I decided not to use them. Socket support is not that hard to implement under UNIX and Win32 and besides, I've done it for my job recently, so I just knew how to do it.

Another reason which decided me to code my own toolbox is that I did not want Liquid War to have external dependencies - except Allegro of course. This way, UNIX gamers to not have to set up and/or download a specific network library. It's also easier to integrate the game in projects like Debian if it has few dependencies.

This network code is not a masterpiece, it's just a little set of tools that have proven to work. That's all.

BTW, it's important to notice that when linking with Allegro, most blocking UNIX calls ("sleep" or "recv" for instance) stop working: they alwasys return immediately. This led me to implement weird ugly hacks, like calling "recv" in a loop until it gets what it wants... This is theorically and practically a performance killer, but I found no other way to fix this. And FYI, this is not an Allegro bug, it's a feature 8-)

- dnsutil.c / dnsutil.h: wrapper code to issue DNS requests, without having to handle the hostent struct.
- sock2cli.c: sode used to wrap low-level network function on the client.
- sock2gen.h: header for sock2cli.c and sock2srv.c.
- sock2srv.c: code used to wrap low-level network function on the server.
- sockdos.c: network API for DOS.
- sockex.c: netowrk routines shared by sockunix and sockw32.
- sockgen.h: header for sockdos.c, sockunix.c and sockw32.c.
- sockunix.c: network API for UNIX.
- sockw32.c: network API for Win32.

10.2.22 High-level network code

These files contains network utilities which are Liquid War specific.

- chat.c / chat.h: functions used to handle chat messages in network games.
- keyexch.c / keyexch.h: functions to send and receive keys to the server. Used on the client.
- netconf.c / netconf.h: code to send and receive the config of the clients over the network.
- netkey.c / netkey.h: contains some tools to manipulate key strokes over the network.
- netmap.c / netmap.h: code to send and receive the maps over the network.
- netmess.c / netmess.h: contains a parser to interpret plain text messages. Used when exhanging information over the network.
- netplay.c / netplay.h: contains the code to set up and start network games.
- network.c / network.h: contains some network related functions and constants used on the client.
- ping.c / ping.h: code used on the client to estimate the average ping time with a server.
- protocol.c / protocol.h: contains the sequence of messages send and recevied by the client when connecting on the server.
- startinf.c / startinf.h: contains struct and tools to handle some network informations while starting a network game.

10.2.23 Communication with the meta-server

The meta-server is called by both client and server. Basically, the server registers itself, and the client asks for a list of servers.

The meta-server itself is just a set of simple PHP scripts with a simple MySQL database. I chose PHP because my provider allows execution of PHP pages, that's all.

The protocol is *very* basic, and uses HTTP 1.0 for requests. Answers are received in plain text, with one information per line. There's no garantee that this would work with any HTTP server, but experience proved that it works with my provider 8-)

- httputil.c / httputil.h: low level functions to handle http requests.
- wwwcli.c / wwwcli.h: code used on the client to communicate with the meta-server.
- wwwsrv.c / wwwsrv.h: code used on the server to communicate with the meta-server.

10.2.24 Server code

The Liquid War server is a rather small program. The only thing it does is accept new players, transmit map and game parameters between them, and then "replicate keys".

By "replicate keys" I mean that the server asks each client what keys have been pressed during the last round, and then dispatches this informations to all clients. This implies that the server has absolutely no idea of who's loosing, who's winning, etc...

All the "logic" of the server is coded in these files, the rest is only utilities and helper functions.

- server.c / server.h: main code for the server (equivalent of main.c for the client).
- srvchan.c / srvchan.h: code used to handles channels on the server. A channel is associated to a given computer and may manage several teams.
- srvcont.c / srvcont.h: global network controler used on the server.
- srvteam.c / srvteam.h: code used to handle teams on the server.

Chapter 11

Bugs

11.1 Report a new bug

If you have troubles with Liquid War 5, if you think it is a bug, and if it is not described in this file, then just send a (precise...) decription of your problem to the Liquid War user mailing list.

11.2 Network

Network support in Liquid War is far from being perfect, so there are a bunch of little problems which can appear. Basically, once the game is correctly started on a LAN, you should have no problems, but getting the game started might be difficult.

11.3 Mouse does not work

Some users reported that they were unable to control the Liquid War cursor with the mouse. Well, the answer is a typical Microsoftish "this ain't a bug, it's a feature!".

More seriously, you're supposed to move the cursor with the keyboard in Liquid War. There's no way to handle the cursor "like a mouse pointer" (*). This is due to:

- Severe limitations in the Liquid War core algorithm.
- The fact that moving the cursor "too fast" would really change the gameplay of Liquid War. As a Liquid War integrist 8-) I can tell you the game would really not be the same if you could move the cursor as fast as you wish. It's part of the game that sometimes it takes you ages to recover from a strategical mistakes. You need to think twice before going to the very end of a level. That's strategy. At least that's how I view things... Anyways as I mentionned above there's a limitation in the core algorithm.

(*) This is not perfectly true, there's a way to control the cursor with the mouse, but it's designed for the case "4 people want to play on the same computer and one single keyboard is not enough". Controlling the cursor with the mouse in Liquid War is possible but yet rather hard to master 8-/ Try it and you'll understand what I mean. This mode can be set up in the "Teams" menu.

11.4 Game does not start

On non UNIX platforms such as Windows or DOS, Liquid War is distributed in a .zip file. It's IMPORTANT that you unzip the .zip files with an "unzipper" which preserves the directory structure. Most install problems under Windows come from broken unzipping programs which extract all files in the same directory... WinZip 8.x or the unzip32.exe utility that comes with DJGPP are both able to uncompress Liquid War .zip files correctly.

On Liquid War 5.5.9 and later, the Windows version should detect this problem automatically and warn you with a message which basically says something like "Unable to load datafile. Are you sure Liquid War is correctly installed?". If you get this message, you need to reinstall the game by unzipping it with a "correct" unzipping program which does not wreck directory structrure up.

11.5 Interference with other Windows programs

It's been reported that Liquid War can run very slowly on Windows when some other programs (Mozilla for instance) are running. So if Liquid War's menus seem to be really really slow, then try to shut down other applications and run the game again.

This problem does not seem to apply on GNU/Linux - at least if you do not run 300 daemons together on your machine 8-)

11.6 Datafile bugs

Sometimes there are some problems when compiling the datafile, this includes:

- The liquidwarcol, liquidwarmap and liquidwartex utilities might freeze or segfault. Typing "make" again often solves the problem.
- The background image sometimes ends up using the wrong palette, which has a very nasty consequence: it looks ugly.

These bugs are quite hard to get rid off, since I can not reproduce them easily. The good solution would be to completely rewrite the liquidwarcol, liquidwarmap and liquidwartex utilities.

11.7 Midi does not work on OSS

11.7.1 Preamble

IF your midi music on Liquid War, or indeed any other Allegro game, doesn't work and you are using the OSS (Open Sound System) drivers (these are the sound drivers which come with the standard kernel distribution), this may well be because Allegro only supports "FM synthesis" and not "wavetable" when it is using OSS. FM synthesis is a very old method of making sound from MIDI and has long since been replaced by wavetable synthesis, with the net result that it's quite possible you've got OSS MIDI working nicely in other applications without having FM support set up at all. This is what I found. (It has to be said that I didn't find the FM sound quality quite as bad as people have said, though).

In this situation, it looks to me like you have the following choices:

11.7.2 Hack Allegro...

...to implement wavetable midi on OSS :-)

and for the rest of us...

11.7.3 Use Allegro's DIGMID midi driver...

...which creates audio from MIDI using a set of patches (more info here: http://www.talula.demon.co.uk/allegro/digmid.html) and plays back through your sound card's audio.

11.7.4 Get an FM driver up and running...

...Which is comprised of the following steps:

- Find out which FM driver is appropriate for your sound card. If you have distribution-specific tools and docs for setting up sound, try those. If not, you will need to be familiar with the knowledge in the Sound-HOWTO and Kernel-HOWTO i.e. know how to compile kernels and modules and deal with sound drivers.
- Look through the OSS modules in 'make menuconfig' and see if anything catches your eye. See if there is any specific documentation on your sound card on http://www.linuxdoc.org. Do a few web searches. For my AWE64, I use the OPL3 driver.
- Compile and install the FM driver module, or set up your system to use the new kernel if you want to compile the driver in.
- Load the module, or boot your new kernel. It is very important that you pay attention to what is said in the 'help' for your FM driver in 'make menuconfig' and read any necessary files in the Documentation/sound/ directory. For example, I just had a nice half-hour wondering why the hell my FM wasn't working now when it had been before - with the OPL3 driver, you have to give the option io=0x388 to insmod. Which is stated nice and clear in the docs, but of course I had forgotten since then. You can prevent such happenings by recording options permanently in /etc/modules.conf - see the manpage etc.
- Try the game. If it's worked you will hear particularly beepy music. Enjoy!

11.7.5 Opl3 occult FAQ

–IMPORTANT– If you are using Liquid War, your FM will only work if you go to the map 'Elephant inside a boa' and proceed to chase each other round in circles for at least 10 minutes. This cures a bug in the design of the OPL3 interface which conflicts badly with the core Liquid War algorithms. How the hell the music hardware even knows about the core algorithms I don't know, but that's what I made of the now-defunct opl3-occult-FAQ, from which here is an excerpt:

```
Many roads a man must take. Those with one-track minds are DOOMED, I tells ya.
```

—- The Liquid War algorithm calculates distances to one place, the cursor.

And:

`Man or machine, face or code, must stand strong and solid; must not just ooze away as slime.`

—- We think it might just take objection to the whole 'slimy' nature of the LW beings. As well as it being LIQUID War.

So, our carefully tailored approach, is to firstly have the players going in all the possible different directions evenly by moving around the map in circles, and secondly to divert the opl3's attention from the general slimy liquidness of it all by emphasizing the solidity, reality, and natural goodness of that classic tapestry: an elephant inside a boa.

That and it's a f***ing ace level.

11.8 Checksum errors

The Liquid War server is a "light" servers which - to some extent - has no idea about what is going on in the game. It simply replicates key strokes between clients and each client maintains its own game state. Normally, the game is designed so that given the same user input, it will behave exactly the same.

However, it happens that sometimes 2 clients can behave differently, and this is a (severe) bug. One consequence is that messages reporting "Checksum errors" appear on the server's and on the client's console output. This bug appears when using non-default rules settings. Basically, if someones tweaks his rules, then the checksum errors appear. Of course I double-triple checked that options were correctly sent on the network, but, well, could not fix the bug. Yet. The short term solution seems to play with default factory settings...

I'm highly interested in bug-reports concerning this problem.

Chapter 12

To do

12.1 Bug-fixing

In its latest releases Liquid War is quite stable IMHO. However there are still some issues with network under Windows for instance. I'm aware of these bugs and I'm trying to fix them but this does really take time.

I always welcome bug-reports and patches, as making Liquid War W 5.x.x as stable and bug-free as possible is really important to me - and most of the time players also appreciate stable programs 8-)

12.2 Artwork

It's hard to find people to do that kind of thing. Artists are indeed pretty rare, at least artists who wish to create stuff freely... So if you feel like adding some theme support to Liquid War, this could *really* help. I used the textures I had but it would be nice if one could have a "space" ambiance, an "ocean" ambiance or a "desert" ambiance. This could really make the game better I think.

Musics (in midi format) are also appreciated. Tim Chadburn spontaneously contributed the first midi files, and it really pleased me 8-)

12.3 New features

I regularly receive requests for new features in Liquid War. Of course I do not have enough time to implement them all, so they land in my "todo" list. However, most "major" evolutions are now planned for Liquid War 6, since the code in Liquid War 5 is bloated as hell. It's all right to debug it and code minor evolutions, but that's all.

However, here's a list of what "could be" in the next Liquid War 5.x.x releases, although it might never be implemented there and come later with Liquid War 6:

- Network enhancements. Network in LW5 is somewhat buggy and hard to use, improvements wouldn't harm.
- Theme support. This would be a way to get rid of the ugly current fonts and menus.

12.4 Liquid War 6

Liquid War 6 *is* planned.

This being said, it's important to note that:

- Not one single line of code has been typed yet 8-)
- Until 6.0.0 is released, 5.x.x remains *the* stable and useable branch. 5.x.x will be maintained as long as needed and will probably be maintained (bugfixes I mean) even when 6.0.0 is ready. So if you want to contribute some maps, graphics, or anything else for the current 5.x.x branch, don't fear that your work will be lost before 2 months because all interest went off 5.x.x and everyone is using 6.0.0 instead. 5.x.x is here to stay.
- Liquid War 6 will be an almost complete rewrite. I mean that common code between branches 5 and 6 might end up in representing 0% of the total code. I think this is a wise decision, for the current code is really hard to maintain, and would not survive any serious cleanup. LW5 was first written in 1998, for DOS, when I had much less experience in programming. In 5 years I - and other people as well - hacked major enhancements in it such as cross-platform support, network games, and if you compare release 5.0 with the latest 5.x.x release, you'll see that a bunch of things have changed. I had never expected I would patch and fix this game for so long, and it's no surprise that it's bloated today.
- Liquid War 6 won't run on "slow" computers anymore. I mean it will require some accelerated display (OpenGL for instance) to run. This is IMHO acceptable since LW5 will remain available. People with slow computers will always be able to run LW5 instead of LW6. But if some nice fancy features are to be implemented in 6.x.x, one needs to drop support for low-class machines.

I also plan to use different technologies and languages to code LW6. As of today, I have a rather precise idea of what tools I'm going to use, but *before* I really start coding LW6, I'd like to test these tools on a "real life example". This "real life example" will probably be a rewrite of the good old HP48 Babal (http://www.ufoot.org/babal/) for desktop computers (GNU/Linux and Windows). Here are the tools I plan to use:

- "GNU framework". I mean that I'll try to make LW6 as close as possible to the GNU standards. This has the advantage that it's then much easier to make the program operate in a GNU environment. For instance LW6 will use GNU automake/autoconf, GNU gettext, and have a GNU-ish source tree. This should save time in the long run, the experience I have on LW5 teaches me that there are a bunch of low-level boring tasks (generating documentation in several formats, generating source tarballs, handling i18n/l10n) which are done automatically by GNU tools, and it's a good idea to re-use other people's work and be standard compliant.
- Standard C language for the core algorithm. This is where I might use a few dozen lines of code from LW5. C is fast, portable, and very adapted to this kind of code.
- Assembly for the core algorithm on Intel architectures. LW5 does this too, and it's even faster than C. However, this assembly code will be written only when the equivalent algorithm is available in C. Think of it as an optimization, not as a base component.
- OpenGL display. I'll probably use Glut (http://www.opengl.org/developers/documentation/glut.html) which is portable and does enough stuff for me. I do not think I'll need a wrapper on it and/or a more general library like SDL.

- "mod" music. Let's be clear: MIDI sucks. Well, more precisely, it sucks on standard desktop computers (the kind of computer everyone uses). Most of the time mod files sound much better, and have the advantage of sounding the same on every computer. MikMod (http://mikmod.raphnet.net) will probably be used.
- Scheme as the glue language for all this. I'll very certainly use Guile (http://www.gnu.org/software/guile/guile.html). I'm strongly convinced that C is absolutely *not* adapted to code large scale programs, unless one uses an extension language. Guile solves this issue by offering a way to code some parts in scheme. You might wonder why I do not use another extension language. The excellent Python language (which I know better than scheme by now) would be a very good contender for instance. Well, to be honest, I'm very interested by Scheme and Lisp-like languages in general, and I've good reasons to think that once I master them, they'll simply be the languages I prefer. But this is enough Scheme advocacy 8-)

Of course, all this complete rewrite and new tools are not a goal in themselves. So here's the list of new features which are planned to be in LW6:

- Truecolor support. LW5 has strong color limitations (32 colors, not 32 bits colors but really *only* 32 colors) for textures and this won't be anymore in LW6.
- Map wrapping/tiling. You could have a map where the right and left border would "communicate". Any fighter leaving the map on the right side would reappear on the left.
- Several cursors per team. This feature is easy to implement even with the current algorithm, but isn't possible to implement in a reasonnable time because LW5 is bloated. LW6 should have it. The fighters would simply follow the closest cursor.
- Fancy multiplayer options such as team alliance.
- Fancy 3D displays. This could be very nice combined with map wrapping/tiling. You could imagine playing on a sphere, on a tore, on a Moebius ring...
- Maps could be deeper in some places. In a deep place armies would take less surface but would be stronger.
- Of course, anything that LW5 does (unless maybe running under DOS), LW6 will do it 8-)
- Any idea is welcome! Although I can't garantee all ideas will be implemented, there are at least more chances that it happens given the fact that LW6 should be much easier to hack and enhance.

So, that's it for LW6 for now. It's still vaporware, but believe me, some day you'll be able to download liquidwar-6.0.0.tar.gz. Until then, any idea, suggestion or help is welcome.

Chapter 13

Copying

Liquid War is a multiplayer wargame.