Liquid War - a unique multiplayer wargame

Christian Mauduit

v5.4.3

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 who?

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

"ufoot@ufoot.org"
"http://www.ufoot.org"

8 Allee de la Fresnaie
95120 Ermont
FRANCE
```

2.3 Other contributors

As Liquid War is now free software, protected by the GPL, anyone is allowed to view, edit, modify and re-compile the source code, 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 Linux.
- Cort Danger Stratton : helped me setting up network support.

Chapter 3

Game options and menus

3.1 Introduction

This section describes how the GUI and menus work. 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!

3.2 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.

3.3 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.

To activate a team, that is to say to add this team to the list of the teams which are playing, you have to click on one of the 12 colored buttons of this team. Then the zone representing the team should be colored whith the color you just clicked on.

To disable a team, just click on the dark button which is on the left of the 12 colored buttons.

You can toggle the "Human/Cpu" mode by clicking one the button which displays this info. If you read "Human", it means that the team will be controled by a player, but if you read "Cpu", it means that the computer will handle this team. And he will do it poorly, so remember that Liquid War is basically a multiplayer game, and that this cpu control is dedicated tobeginnersonly.

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.

3.4 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.

3.5 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.

3.6 Rules menu

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

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 "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.

3.7 Speeds menu

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.

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.

3.8 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.

3.9 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.

Chapter 4

Network game

4.1 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 Linux versions will allow you to do it.
- 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-)

4.2 Getting started

4.2.1 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 Linux to get this information if you don't have it.

4.2.2 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 Linux or "lwwinsrv.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 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 4. The server really needs to know how many teams will be in the game: when enough teams are connected, the game starts. It's possible to skip this question by typing "liquidwar-server -4" instead of a plain "liquidwar-server".

```
Listening on port 8035...
Waiting for 4 teams...
```

Now the server is ready to accept clients. By default it listens to clients on port 8035. As of release 5.4.0, it's not possible to change this setting, but 5.4.1 should certainly allow it 8-)

4.2.3 Starting the clients

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

Go to the "Teams" menu and select 2 teams, red and blue for instance. If you don't know how to do this, then try and play Liquid War on a single computer first.

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 nickname.
- Change the IP address of the server.
- Change the default communication port.

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

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.
- Change the nickname to "B".

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.

At this point, you are almost ready to play. Simply click on "Start game" on computer A and then on computer B. The server console should display the following:

```
Waiting for 4 teams...
Connection from 127.0.0.1:3098
Accepted "A"
Waiting for 2 teams...
Connection from 192.168.1.2:3101
Accepted "B"
Talking to "A"
Talking to "B"
Game start
```

And the game should start on both computers A and B.

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

4.3 Options

4.3.1 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.

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

4.3.2 Client options

The "-netlog" option works for the client too. Otherwise, all the options can be set from the "Net Game" menu.

Note that when you play a network game, the settings such as those found in the "Rules" menu must be the same on every computer. However, Liquid War does this automatically. But a consequence is that after a network game, your settings might have changed if you were not on the machine from which the settings have been taken and then dispatched to every player.

4.4 About Liquid War's network implementation

4.4.1 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 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.

4.4.2 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 5, you can theorically play at a rate of 25 rounds/sec.

On one hand, setting the lag 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 dratically he number of rounds per second.

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 (5), 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.

4.5 Troubleshooting

4.5.1 General information

Network support in 5.4 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.
- Send me a bug report at "ufoot@ufoot.org" so that I can fix the bugs if there are some.

4.5.2 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.

Chapter 5

Command line parameters

5.1 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...

5.2 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.

5.3 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 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.

5.4 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.
- "-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 thesafemode.

- "-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.

5.5 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.

Chapter 6

Platform specific issues

6.1 General remarks

Liquid War is now a cross-platform game, thanks to Allegro. So now you can play under 3 different OS, without rebooting. I personnally never use the Windows port, since the DOS one works better. Anyways I almost never run Windows for I have a nice Linux box runnning all the time, and I find it much more adapted to my needs.

The same code will compile under the 3 platforms, but with slight differences when running. C defines are used to code the specific stuff.

As I said, I try to use the same code for all platforms. This is in the long term the best solution. 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). Therefore, 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.

6.2 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 (but unsafe) 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.

6.3 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.

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

6.4 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 exe in:

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

and the configuration file is

```
~/.liquidwarrc
```

Since not all Linux distributions have /usr/games in their path, I also put a symbolic link to the binaries in /usr/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/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 safe under Linux. You should also know that the 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 Linux.

Chapter 7

User levels

7.1 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. 2-color bitmaps will probably 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!

7.2 Maps

Liquid War 5 does many checking 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/liquidwar/map" on Linux.

7.3 Textures

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

7.4 Send me your levels

Maybe you will find that my levels are ugly and unplayable. Well, if you have made user levels and think they are great, just send them to me by attaching them to an E-mail ("ufoot@ufoot.org"). Please use only 256 colors bitmap and zip them before sending them, for I sincerly don't want my provider's mail server to explode. If I get enough new user levels, I will try and include them in the next release of Liquid War.

Chapter 8

Core algorithm

8.1 Introduction

8.1.1 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.

8.1.2 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.

8.1.3 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.

8.1.4 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.

8.2 Mesh

8.2.1 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)

8.2.2 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).

8.2.3 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.

8.3 Gradient

8.3.1 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?

8.3.2 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

8.3.3 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...

8.3.4 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.

8.4 Move

8.4.1 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.

8.4.2 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.

8.4.3 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 9

Source code

9.1 General remarks

Liquid War 5 is basically a big C program, and this section explains what you can find in source files. 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-(

9.2 List of C source files

9.2.1 advanced.c / advanced.h

Contains the GUI advanced options menu.

9.2.2 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.

9.2.3 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.

9.2.4 army.c / army.h

Functions to create the armies, and place them on the battlefield.

9.2.5 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.

9.2.6 back.c / back.h

This modules displays the background image.

9.2.7 base.h

Contains global constants used in many different files.

9.2.8 basicopt.c / basicopt.h

Handles basic command line parameters such as "-v" or "-h".

9.2.9 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.

9.2.10 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.

9.2.11 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.

9.2.12 cursor.c / cursor.h

Contains the code to init the cursors and place them on the battlefield at the beginning of the game.

9.2.13 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.

9.2.14 dialog.c / dialog.h

Contains code for standard dialog boxes.

9.2.15 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.

9.2.16 disp.c / disp.h

Contains functions to display the battlefield.

9.2.17 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...

9.2.18 error.c / error.h

Contains functions to display error messages once the game is in graphical mode.

9.2.19 exit.c / exit.h

Contains code that is executed when the game ends, it shuts down Allegro and displays messages on the console.

9.2.20 fighter.c / fighter.h

Contains code to move the armies, once the gradient has been calculated.

9.2.21 game.c / game.h

Contains the main game loop.

9.2.22 gfxmode.c / gfxmode.h

Contains code to set up the various video modes, and defines which modes are available for each platform.

9.2.23 glouglou.s / glouglou.h

Assembly module, it is a replacement for some functions of distor.c. It goes much faster but does the same.

9.2.24 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...).

9.2.25 graphics.c / graphics.h

Code for the graphic options menu.

9.2.26 help.c / help.h

Generic functions to display the various help pages.

9.2.27 httputil.c / httputil.h

Low level functions to handle http requests to "http://www.ufoot.org".

9.2.28 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.

9.2.29 init.c / init.h

Contains code to initialize Allegro with proper options and analyze failures.

9.2.30 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.

9.2.31 keyboard.c / keyboard.h

Contains code to handle key presses.

9.2.32 keyexch.c / keyexch.h

Functions to send and receive keys to the server. Used on the client.

9.2.33 level.c / level.h

Contains code for the menu where the player can select a level and its options (texture or color).

9.2.34 log.h

Common header for logcli.c and logsrv.c.

9.2.35 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.

9.2.36 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.

9.2.37 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.

9.2.38 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.

9.2.39 menu.c / menu.h

Contains the code for the main menu.

9.2.40 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.

9.2.41 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.

9.2.42 monster.s / monster.h

Assembly functions to speed-up the game. It's a replacement for some fighter.c functions.

9.2.43 mouse.c / mouse.h

Wraps the mouse movements to virtual keyboard keys. This way the mouse can be used to control the players.

9.2.44 move.c / move.h

Provides an API to move the cursors.

9.2.45 netconf.c / netconf.h

Code to send and receive the config of the clients over the network.

9.2.46 netgame.c / netgame.h

Contains the code for the net game menu.

9.2.47 netkey.c / netkey.h

Contains some tools to manipulate key strokes over the network.

9.2.48 netmap.c / netmap.h

Code to send and receive the maps over the network.

9.2.49 netmess.c / netmess.h

Contains a parser to interpret plain text messages. Used when exhanging information over the network.

9.2.50 network.c / network.h

Contains some network related functions and constants used on the client.

9.2.51 options.c / options.h

Contains the code for the options menu.

9.2.52 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.

9.2.53 parser.c / parser.h

Contains code to parse and analyze the command line parameters.

9.2.54 pion.c / pion.h

Contains code to display the cursors.

9.2.55 play.c / play.h

Contains the code which ties the menu to the main gameloop.

9.2.56 profile.c / profile.h

Provides tools to calculate how fast the game is runnning and what operations slow it down.

9.2.57 protocol.c / protocol.h

Contains the sequence of messages send and recevied by the client when connecting on the server.

9.2.58 rules.c / rules.h

Code for the rules menu.

9.2.59 score.c / score.h

Functions to display the scores at the end of the game.

9.2.60 server.c / server.h

Main code for the server (equivalent of main.c for the client).

9.2.61 sock2cli.c

Code used to wrap low-level network function on the client.

9.2.62 sock2gen.h

Header for sock2cli.c and sock2srv.c.

9.2.63 sock2srv.c

Code used to wrap low-level network function on the server.

9.2.64 sockdos.c

Network API for DOS.

9.2.65 sockex.c

Netowrk routines shared by sockunix and sockw32.

9.2.66 sockgen.h

header for sockdos.c, sockunix.c and sockw32.c.

9.2.67 sockunix.c

Network API for UNIX.

9.2.68 sockw32.c

Network API for Win32.

9.2.69 sound.c / sound.h

Functions to play sound.

9.2.70 speeds.c / speeds.h

Contains the code for the speeds menu.

9.2.71 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.

9.2.72 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.

9.2.73 srvcont.c / srvcont.h

Global network controler used on the server.

9.2.74 srvteam.c / srvteam.h

Code used to handle teams on the server.

9.2.75 srvtime.c / srvtime.h

Code used to handle time on the server, where Allegro's functions are not available.

9.2.76 startup.c / startup.h

Analyzes the command line parameters and stores them into global variables.

9.2.77 team.c / team.h

Code for the team menu.

9.2.78 texture.c / texture.h

Contains code to handle textures. Textures are stored in a special format which uses 5 bits per pixel.

9.2.79 thrdgen.h

Header fo,r thrdunix.c and thrdw32.c.

9.2.80 thrdunix.c

Provides thread support on UNIX.

9.2.81 thrdw32.c

Provides thread support on Win32.

9.2.82 ticker.c / ticker.h

Sets up a timer callback.

9.2.83 time.c / time.h

Functions to know how long the game has been running, knowing that it can be interrupted.

9.2.84 volume.c / volume.h

Code for the sound menu.

9.2.85 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.

9.2.86 watchdog.c / watchdog.h

This module waits for "secret codes" to be typed while the game is running, and traps them.

9.2.87 wave.c / wave.h

Code for the wave menu.

9.2.88 wwwcli.c / wwwcli.h

Code used on the client to communicate with "http://www.ufoot.org".

9.2.89 wwwsrv.c / wwwsrv.h

Code used on the server to communicate with "http://www.ufoot.org".

Chapter 10

Bugs

10.1 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 me a mail (ufoot@ufoot.org) with a (precise...) decription of your problem. I'll try and see what I can do about it.

10.2 Memory bugs

Memory allocation of Liquid War 5 can get very buggy, especially when you run it in a DOS box.

The main reason is that Liquid War keeps on allocating and freeing bitmaps. And sometimes there is not enough DPMI memory to create bitmaps. If this happens, Liquid War should exit and give you a short description of the problem.

But it is also possible that Liquid War exits with a page fault error without warning you. This is often because the GUI manager of Allegro NEEDS free memory to work correctly. And if this memory is not available, it crashes... I have tried hard to work this arround but could not make it. All you can do to avoid this is to give Liquid War more DPMI memory and try and reduce the "-mem n" command line parameter.

10.3 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.

Basically, network support in 5.4.0 and 5.4.1 is half-broken, and setting up an internet game is pretty hard.

10.4 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 Linux - at least if you do not run 300 daemons together on your machine 8-)

10.5 Mouse cursor display

Most of the time, the mouse cursor will not be displayed correctly, more precisely, weird squares can appear now an then where the mouse cursor was. I acknowledge this looks ugly but at least it's not a "blocking" bug... ...it's ugly, that's all!

Chapter 11

To do

11.1 Bug-fixing

Now that Allegro 4 is finally out, Liquid War should be quite stable. Still, I doubt I'm already done with bug-fixing, and I fear this will remain one of my most time-consuming tasks =8-)

11.2 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 could be a nice contribution too.

11.3 Network enhancements

The network code could be improved, it needs to be more flexible and a few options must be added. Basically, I need to try the game over the Internet, get feedback from players about what's wrong and then I'll see what I'll do 8-)

11.4 GUI improvement

Well, the GUI I coded is OK, but still it could be even better. The ideal thing would be to be able to change settings during a game, instead of having to stop the game. One should also find better labels for buttons and sliders.

11.5 Choose another language

I'm considering re-coding Liquid War using another programming language. As of today, the best candidate seems to be Python. Basically, I would code a low-level Python object in C, which then could be used in any Python program. But this represents quite a lot of work...