License to Salp: The Salp Wars User's Guide

Tyler Goen

June 27, 2002

Contents

List of Figures

List of Tables

Abstract

Help has arrived! Geared toward players who have some game development knowledge, this manual addresses basic end-user concerns regarding the game Salp Wars. After reading, players should be able to install and run a copy of the game.

Much of the information inside has been leeched from the brain of Tyler Goen (lead programmer). However, a perusal of the bibliography items may help put the report's non-Salp-specific terminology into context.

Beginner topics such as how to download, install, configure and play the game are covered, as well as game types and terminology.

Though Salp Wars takes a unique approach to gaming, there are still some hurdles in the way of its mass-consumption. This documentation should give users the extra push necessary to become comfortable with the game.

To get the most out of the reading, the user should already be familiar with how to bring up a command-line console. Though it is not critical to the understanding of the game, familiarity with the Concurrent Versioning System (CVS) and GNU/Linux will also improve the reader's experience.

1 Introduction

Hi Taery. May I call you Taery? Thanks. Taery, meet Salp Wars. Salp Wars, meet Taery.

Whereas you are likely at least ten years old, Salp Wars (hereafter, SW) is barely a year old, so please play nicely with him. Taking care of computer games is sometimes a tricky business, but luckily for you, SW comes with this handy manual.

In this manual, I, the omnipotent-yet-nameless narrator, will describe the history and future of the game as well as most of the information you'll need to begin playing.

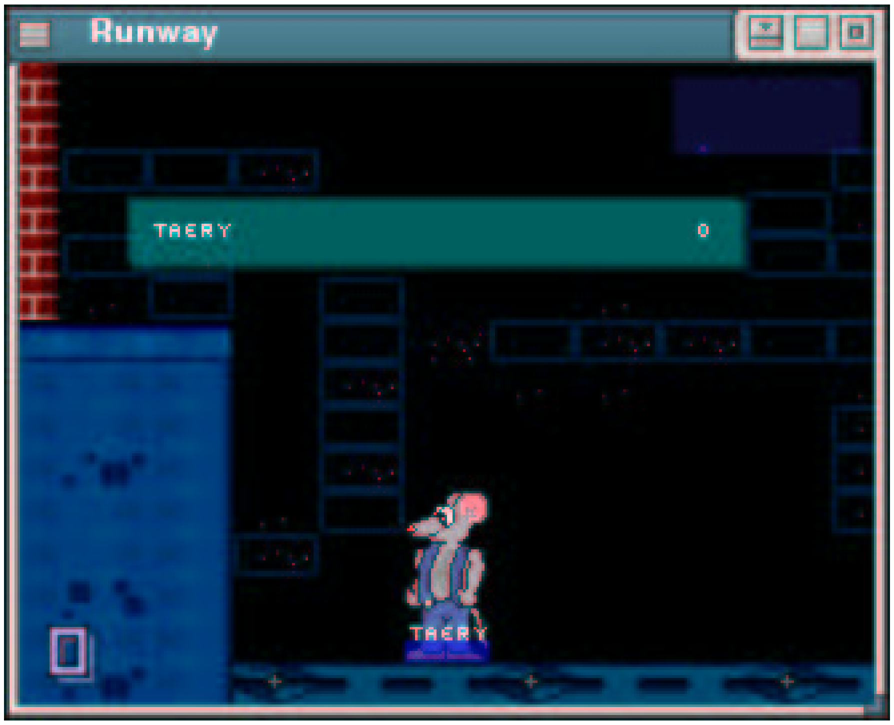


Figure 1: Taery waits patiently to play.

1.1 Key Terminology

Before we get too deep, let's clear up the Taer and salp references.

The Taer

You must be wondering why your name is Taery.[1] Well, it's because you're a *Taer*! This term evolved slowly from a series of inside jokes among the developers, long before SW was even conceived of.

The name itself is derived from the word "rat" by reversing it (to become "tar") and then adding an "e" (to differentiate it from the gooey black stuff). The resulting word is only half of the phrase "taer drattsab," the origin of which will be left as an exercise to the reader.

In 1998, when developing the textual Multi-User Dungeon (MUD) *Rock II: Crashed Plane*, Ryan Karr (content developer) and Tyler Goen (programmer) first introduced players to the Taer as a rat-like character. Though Ryan's description of the Taer as shown in Figure 2 (page 2) is not fully translated into the graphical SW, the two versions definitely approximate each other.

Just in case it comes up in a game of *SW Trivia* (hey, it could happen), the plural of Taer is the same as its singular form.

[1]If you're female, your name is more likely to be Taera instead.

Originating from the Veralian Plane located on the outer rim of the dimensional web, the Taer are considered one of the most vile and disgusting of sentient races. They dwell in underground tunnels, though most of their agriculture is located on the surface. Endless grape vineyards cover most of their land, stretching across miles and miles with only the occasional swamp to break the endless rows.

Their diet consists of bulbous green frogs which dwell within the deep swamps and the grapes which are grown year round. It is this diet of amphibians and fruit which accounts for their odd ability known as salping in most civilized planes. Combined with their unique rat-like biology, their diet creates a corrosive slime within their stomach. This slime forms along the stomach's wall, and can build up for over a week before it must be released. Taer can force this slime up, using it to 'salp' victims. Once leaving the stomach, the acidic slime seems to lose potency quite rapidly, though during its short-lived usefulness it can easily eat through several layers of thick metal. Its thick and sticky consistency can grasp almost any surface, and the quantity of the slime provides enough coverage to engulf the upper portion of an average sized human.

The Taer resemble humanoid rats, standing four to five feet tall on average. They are bipedal, relying on their thick cord-like tail to provide extra balance. The facial structure is more similar to a rodent, with an elongated snout and whiskers. Taer are covered with thick fur, ranging drastically in color from light tans to black.

Figure 2: Description of the Taer from Rock II: Crashed Plane

Salping

Figure 2 (page 2) should clue you into what is meant by "salping" as well.

As a noun, *salp* refers to a combination of frogs and grapes. In the special case of a Taer, it can also refer to vomit consisting of frogs and grapes.

As a verb, *salping* refers to the process of throwing salp (in either acidic or whole form) at some victim. In case you're wondering what this might look like, see Figure 3.

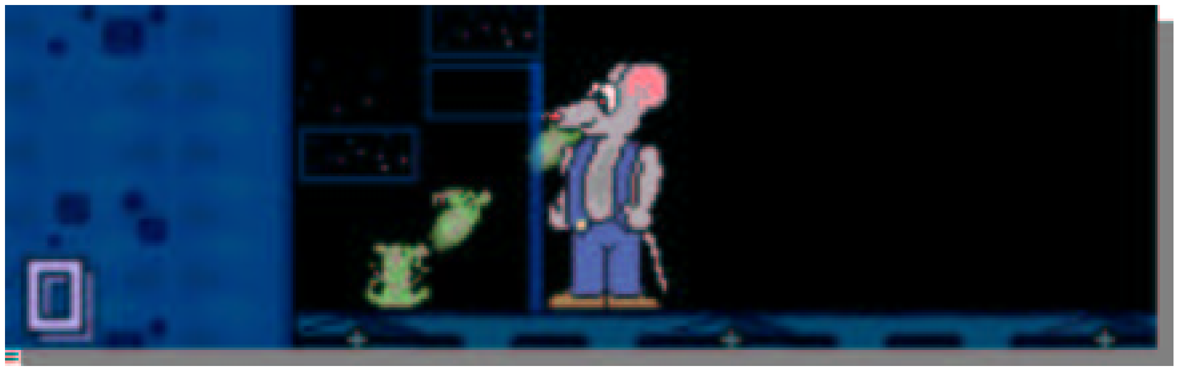


Figure 3: Taery spews up some acidic salp.

This term also has some wacky origins. Several friends and I used to log onto a local Bulletin Board System[2] (BBS).

When playing the trivia game *Farwest Trivia*, players could type "action" commands to interact with other players. For example, one might type "grin bob", after which a message like "Taery shows Bob a huge grin." would be shown to all other users in the room. Many different actions existed: hug, laugh, nudge, ..., and slap.

Since players would have to speed up their typing to keep up with the trivia, sometimes these commands would get mistyped. In this case, "slap" would sometimes be typed as "salp." Obviously, "salp" wasn't a valid action and so the associated message was never displayed on the screen.

[2]A BBS is a computer service typically accessed via modem, where users can chat, post messages, send e-mail and play games. Most BBS's have a local feel to it, compared to Internet chatrooms.

After players persistently errored on typing "slap," eventually a "salp" action was added to the game, which displayed a message similar to "Taery is salping Bob!" The new message needed a meaning. What exactly *was* salping?

Following some thought, two of the players agreed that the term meant to throw frogs and grapes. Why? Even *I* have no clue, but it stuck.

Finally, why do Taers salp? It's simply the combination of two lengthy, trivial inside jokes.

1.2 About the Documentation

The *Salp Wars User's Guide* is meant to help SW players become acquainted with the game; it does not address many development concerns. Expect a *Salp Wars Developer Guide* by the end of 2002.

This documentation was written in LaTeX and is available from the official Salp Wars website (`http://www.salpwars.com/doc/`) in both PDF and HTML formats. This may be distributed freely in any form, whether or not it accompanies the game code.

Please note that this manual attempts to document the *latest*, bleeding-edge[3] version of Salp Wars. There is a chance that the version you downloaded may not act as described in this manual. If you notice an inconsistency, please first make sure that you have downloaded the latest *alpha* version from our website, as described on page 7.

If such inconsistencies still exist, please alert a documentation or project coordinator (a list of whom is on page 3) or submit a bug report as described on page 11.

To paraphrase Dick Brandon, "Documentation is like chocolate: when it is good, it is very, very good; and when it is bad, it is better than nothing."

1.3 About the Team

Computer games don't just grow on trees.[4] Through intense development, rigorous testing and steadfast Mountain Dew drinking, these fine folks have helped make SW possible:

Tyler Goen (`mrplate@users.sourceforge.net`) began the SW project in an effort to dabble in graphical, action-game programming. He had previously coded online text games like *Rock: Conflict of Interests* and *Rock II: Crashed Plane*, as well as other experimental games like *Hang!* (a multiplayer hangman clone). His Perl programming skills are stronger than his C++, but he's learning quickly. Tyler plays the role of Project Coordinator and concentrates on code development.

Daniel Lindsley (`polarcowz@yahoo.com`) often tests bleeding-edge builds of the game, catching bugs and gameplay issues before the general public encounters them. He is a prolific writer of feature requests and other game adjustments. Having learned about Tile Studio (a design tool used in creating game tiles and maps), several of Daniel's levels may appear in future releases of the game.

Jeff Lucas (`enstrim@yahoo.com`) joined the SW project to provide bitmap graphics and (possibly) some code. He's written various small programs himself (*Circtris*, a tetris clone with circular unit, and a brief graphical Final Fantasy clone) but nothing on the scale of SW. His C++ skills are ironically better than his graphics skills, and his skills of "being busy" exceed all. Nonetheless, years of playing around in Q-BASIC and Kidpix and designing his own bit-mapped characters on graph paper have given him a slight ability to, in his own words, "Uh... draw stuff." Jeff plays the role of graphic designer.

Thankfully, these were not the only people generous enough to contribute to the project. A full list of contributors can be found in the game's "CONTRIB" file (in the "salp" directory, wherever you install the game). If you're crazy enough to help out, you might find your name in there!

[3] "Bleeding-edge" refers to the latest, freshest possible version of a product.
[4] If games grew on trees, I'd be first in line to buy some seeds.

1.4 Game Genre

What happens when you place games like *Contra* and *Unreal Tournament* in a *Super Mario Brothers* environment? That's what the SW team aims to find out. The game is a two-dimensional, real-time simultaneous multiplayer network combat platform game. Just in case that barrage of adjectives flew above and past your head, let's break down what is meant by each term.

two-dimensional Gameplay takes place on two axes. Sure, if you're playing a game on a standard monitor, all games are two-dimensional. Typically when we refer to two-dimensional games, though, it is implied that the player can only move on two axes. In SW, your character can move left and right (the X axis) as well as up and down (the Y axis).

Three-dimensional games such as *Unreal Tournament* allow you to travel with some depth (the Z axis) as well.

real-time The game involves no distinct turn-taking, where you must wait for other players to move before you are allowed to move (as in checkers, poker and many other other board or card games). In a real-time game, it is "always" your turn. Consider soccer, *Unreal Tournament* and *Contra* as examples of such games.

simultaneous multiplayer More than one person can play the game at the same time. You don't need to take turns controlling your character. This differs from the "real-time" adjective in that it refers to the ability to have more than one player in the game environment at the same time.

It is important to note that Salp Wars ONLY has multiplayer support. That is, there are no single-player levels or tutorials through which you can play the game by yourself.

network You play the game over a network of computers. Typically, each player has his or her own computer, and all players' computers are connected via a network (e.g. through Ethernet).

Contrast this with most console games, where all players share the same screen.

combat The key solution to conflicts is fighting. Kids, don't try this at home.

platform The player's character can walk on horizontal surfaces. *Contra* and *Super Mario Bros.* are classic examples of platform games.

Of course, you already know what a game is.

1.5 Development Status

SW is not finished, and is undergoing heavy development. However, you may still download the source code and alpha versions of the software, as described on page 7.

So as not to mislead you, let's look at a few qualities of an alpha software release:

1. It is the first major release of its kind, and is in need of testing. Often the code development is incomplete compared to a "final" (also called "production") release.
2. It can cause unexpected things to happen on your system. That is, it is more likely to crash than "beta" or "final" releases.
3. Its documentation is likely incomplete or erroneous, similar to the state of the software.

Don't get me wrong, the team does significant testing of the game before posting the latest alpha release, but it is quite possible that unforeseen errors[5] can sneak into such releases.

The game code is written in C++, compiled with *g++* for MacOS/Linux and *Microsoft Visual C++* for Windows platforms. Simple DirectMedia Layer (SDL) libraries are used to provide cross-platform support without having to write separate sets of graphical/GUI code.

Since the program has been released under the terms of the GNU General Public License, you are free to own as many copies of the game as you wish. The source is available as well, so you can make your own tweaks to the game or hunt for easter eggs.[6] If you're interested in helping develop the game, see page 11.

[5]Yes, errors are often called "bugs," though the latter term seems to downplay the seriousness of such errors. Flaws should not appear cute and cuddly.

[6]Easter eggs are "amusing tidbit[s] that creators hid in their creations."

1.6 Gameplay in a Nutshell

No, you don't actually play the game in a nutshell; it's just an expression.

The game is primarily combat-based. In all anticipated game types, you play the role of a character (e.g. a Taer) who fights other players. A variety of weapons will aid you, including spread-shot guns, shotguns, and of course, salp.

There are no plans to make a single-player version, so to have any fun, you'll need some living, breathing friends.[7]

Unlike the games *Super Mario Bros.* and *Contra*, winning the game is not dependent on your reaching point B from point A. Rather, you are free to traverse the map any way you please, accumulating points for your team. How these points are scored depends on the active game type. Here are several planned game types (though currently only *Deathmatch* is supported):

Capture the Flag (CTF) Each team owns a base where their flag is stored. A team scores a point whenever the team carries an enemy team's flag back to where their own flag is kept. Flag-carrying players drop their flags upon death, and bonus points are rewarded for such kills. Flags are returned to the owner team's base whenever an owner team's player runs over the flag.

Deathmatch Each player belongs to their own team. Every time a player kills another player, the killer scores a point for himself.

Fishing A Taer isn't just any kind of rat; it's a pack rat. In this game type, prized items appear throughout the map (for example, perhaps apples from a tree, or fish from a stream). Players must catch these and drag them back to their base without dying. The player or team scores a point for each item that reaches their base. Like *Capture the Flag*, enemies can attack you in order to make you drop your bounty.

King of the Hill Certain areas of the map are considered "the hill." A team is considered to be "king of that hill" when *only* its members are touching that part of the map. Points are scored based on the amount of time each team has been "king of the hill." For example, if Red Team is king for 30 seconds, they will have 30 points.

Team Deathmatch Like *Deathmatch*, but players are teamed together in groups, and the sum of each of the individual scores represents the team score.

The game ends once some combination of the following events occurs:

- A time limit has been reached (e.g. five minutes after the game begins)
- A score threshold has been broken (e.g. a team reaches 10 points)
- The disparity between the top two scores has broken a threshold (e.g. the top-scoring team is five or more points ahead of the second-highest-scoring team), similar to how tennis is played in close matches
- It is impossible for any other team to win (for example, if there is not enough time left for the second-place team to score more points than the highest-scoring team in King of the Hill)

With each of these game types, the player or team with the highest score is declared the winner.

2 Installation

So you want to travel to the vomit-laden land of the Taers? No sweat, we'll get you there in style. This section details everything you need to get started.

2.1 Licensing

Salp Wars is released under the terms of the GNU General Public License (Version 2 or later).

In short, you can freely use this software and its source code. The only catch is that if you make your own (derivative) version of Salp Wars available to the public, the source code to this derivative work must also be made available to the public. Cool stuff!

[7]For an additional fee, we may be able to arrange a friend-leasing agreement, whereby enthusiastic companions are shipped to your house for a monthly fee.

As you'd probably expect, the download and execution of this software is to be done at your own risk. There aren't any intended flaws or virii associated with the files linked to by this page, but the author isn't to be held responsible in case something slips by his or your virus scanning software.

Finally, for those of you who prefer scary, intimidating legal talk, try this on for size (quoted from the game's opening message):

> Salp Wars comes with ABSOLUTELY NO WARRANTY. This is free software, and you are welcome to redistribute it under certain conditions. See 'LICENSE' and 'COPYING' files for more details.

The "LICENSE" and "COPYING" files are included in each distribution of the game, in the "salp" directory.

2.2 System Requirements

The following are required in order to happily play Salp Wars. If you're using anything less, consider yourself lucky.

- An x86 (e.g. Pentium II, Pentium III, Athlon, Duron, etc) or PowerPC (603e, 604e, G3, G4, etc) processor of at least 200MHz clock speed.
- An operating system; Microsoft Windows and GNU/Linux are supported.
- A high-speed ($>$ 56Kbps) Internet or network connection. The game may be played over the modem, but you will not have a very good time.
- Gamepad (optional). For proper gameplay, the gamepad should have at least:
  - One 4-way directional pad (as found on the left half of a Super Nintendo pad)
  - Four buttons to the right of the directional pad (like a Super Nintendo's X, Y, A and B buttons)
  - Two buttons at the top of the controller (like a Super Nintendo's L and R buttons)

Microsoft Windows

- Windows 98 or later (Windows 2000, Windows ME, Windows XP).
- A variety of SDL DLLs, similar to those found in the Linux Requirements section. To save you time hunting for the proper files, we've packaged these up into an easy, one-time download. Download these at: `http://www.salpwars.com/download/salp-dll.zip`

GNU/Linux

- GNU C++ Compiler (`g++`)
- SDL 1.2[8]
- SDL_net[9]
- SDL_mixer[10]
- SDL_image[11]

Note: To compile Salp Wars, you will need the proper SDL headers, so make sure you download the proper version of each module above.

[8] See `http://www.libsdl.org/download-1.2.html`
[9] See `http://www.libsdl.org/projects/SDL_net/index.html`
[10] See `http://www.libsdl.org/projects/SDL_mixer/index.html`
[11] See `http://www.libsdl.org/projects/SDL_image/index.html`

2.3 Downloading the Game

To download the game, simply head to one of the two websites:

`http://www.salpwars.com/download/` has bleeding-edge, *alpha* builds of the game. These are typically updated every night that the game code has been modified. They are *not* "final releases" and should not be treated as shiny final copies. They are mainly used by developers and testers to prepare the game for a stable release. This page also explains which files to download for each platform.

`https://sourceforge.net/project/showfiles.php?group_id=47786` archives stable releases of the game and related files. At the time of writing, no files are available at this URL. Don't worry, though; once a version of the game is ready for public consumption, it will be here. We promise. Until then, feel free to wet your appetite through the bleeding-edge builds.

Downloading via Concurrent Versioning System

If you're a developer, or simply a CVS freak (you'd know if you were), you might want to download the freshest version of the game through CVS. Of course, you'll have to compile the game yourself. If you're not familiar with CVS, then just ignore this section; it's not meant for you. If you *are* that CVS freak, however, here's what you want to do. We assume here that you are using UNIX environment (Linux, BSD, Cygwin) and already have a CVS client installed.

1. By the end of this, a new directory called "salp" will be created. `cd` to the directory *inside* of which the "salp" directory should be placed. For example, if you typed: `cd  /programs/`, then your "salp" directory will be found at `/programs/salp/`.
2. Type the proper CVS command, depending on whether you are a developer registered with the Salp Wars project:
   - If you are *NOT* registered with our project, use anonymous CVS by typing *all on one line*:

     ```
     cvs -z3 -d:pserver:anonymous@cvs.salpwars.com:/cvsroot/salpwars co salp
     ```

   - If you *ARE* registered with our project, type these *two* lines (changing `developername` to your login name):

     ```
     export CVS_RSH=ssh
     cvs -z3
     -d:ext:developername@cvs.salpwars.com:/cvsroot/salpwars co salp
     ```

     If this gives you an error message saying that a certain directory cannot be found, try typing:

     ```
     ssh developername@cvs.salpwars.com
     ```

     You'll get disconnected as quickly as you've connected there, but you should have more success if you try typing the previous `cvs` command again.
3. You're done!

2.4 Decompressing Game Files

Of course, downloading the files isn't enough; you have to decompress them too. The process for this varies depending on which operating system and decompression utilities you use.

Microsoft Windows

You've got some unzipping to do!

1. Unzip the `salp.zip` file you downloaded in section 2.3.[12] For the purposes of this example, we'll assume that you have unzipped the file in the directory `C:SALP`, so that the `SALPWARS.EXE` file can be found at `C:\SALP\BIN\SALPWARS.EXE`.

[12] You should already know how to decompress a .ZIP file. If you're looking for tools to do so, you might want to consider WinZip, WinRAR, or Aladdin Expander (which can be found http://www.download.com/). Using any of these tools to extract the game files should be fairly straightforward. If you experience trouble with this process, please e-mail us and we will expand on this topic.

2. Unzip the `salp-dll.zip` file you downloaded so that its contents fall into the `C:\SALP\BIN` directory. Without these files, you may get some "Required DLL" error messages when trying to run `SALPWARS.EXE`. For information on installing the DLLs, see the file `Readme.txt`, included in `salp-dll.zip`.

GNU/Linux

If you're a "leet UNIX haxor,"[13] you already know what to do with a `*.tar.gz` file. Just in case you've somehow forgotten, though, here's what you want to do:

1. In the command prompt, `cd` to the directory where you've downloaded your `salp.tar.gz` file.
2. Type the following to untar the files into a new `./salp/` directory: `tar -xzvf salp.tar.gz`

That wasn't so hard, was it? Of course not, the real challenge is in getting this beast to compile.

2.5 Compiling the Game

If you're using the Windows version of Salp Wars, don't worry about this; you shouldn't need to compile your own version. We've done that for you.

GNU/Linux users, on the other hand, need to compile the game in order to run it. Here's what you need to do:

1. Make sure you have all the programs and libraries listed in section 2.2 (page 6).
2. Once you've downloaded the game, enter the `salp/src/` directory and type "`make`" (without the quotes). If all goes well, you should see a message like:

```
### SALP WARS HAS BEEN COMPILED SUCCESSFULLY. YAY!
### Either type "make fun" to play, or run "./salp"
### from "../bin/". Have fun!
```

Should you encounter errors in compiling the game, send us e-mail and we'll try to help you out (and update the documentation accordingly).

3 Running the Game

Once you've downloaded, decompressed and (for GNU/Linux users) compiled the game, you're *finally* ready to play! Sections 3.3 (page 9) and 3.2 (page 9) describe all you typically need to get the game running.

When played over the LAN, typically one player starts the game in server mode (that is, he becomes the server), and all other players join that game in client mode.

You may read some references to the "command-line" interface[14] (e.g. "MS-DOS Prompt," "Xterm" or "Konsole"). If you are unfamiliar with this, you may ignore the command-line parts of the discussion. In the end, though, the command-line interface allows for much more customization of the game, and it is recommended that you learn to use it.

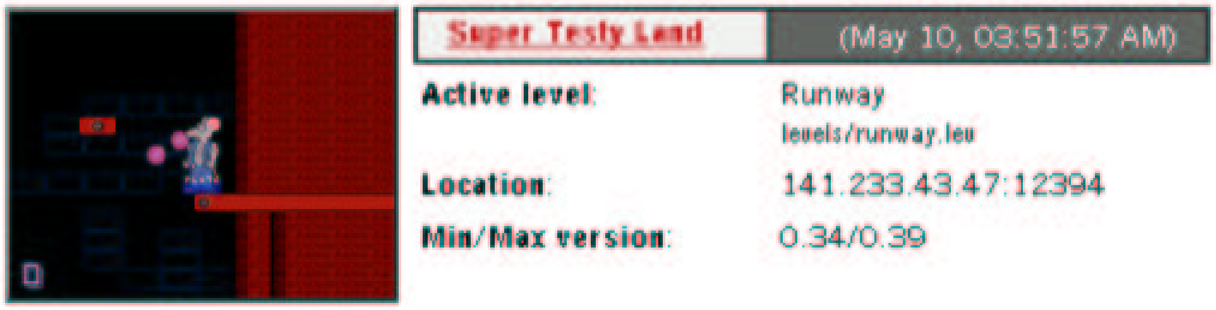
3.1 The Metaserver

Typically, whenever a server hosts a game, the game is automagically[15] registered on the official "Salp Wars Metaserver" webpage. A metaserver can be thought of as a "server server". It is a central hub, serving a list of servers for those who wish to join an open game (like the game displayed on page 9, Figure 4 (page 9)).

The official metaserver can be viewed at:

```
http://www.salpwars.com/metaserver/
```

[13] If you're not sure what a "leet haxor" is, then you probably aren't one.
[14] We assume that you are familiar with the command-line interface.
[15] *Automagically.* You know, so automatic it looks like magic.



Super Testy Land (May 10, 03:51:57 AM)

Active level: Runway
levels/runway.lev

Location: 141.233.43.47:12394

Min/Max version: 0.34/0.39

Figure 4: Sample game listed by the official metaserver

3.2 Playing as Server

If you want to host a new game of your own, you'll need to run SW in server mode. The easiest way to do this is to double-click the `SALPWARS.EXE` you see in the "salp/bin" directory. This will start a new game, playing the default level.

If you'd like to use a different level, try invoking SW from the command line, changing the level parameter as follows:

```
salpwars level=levels/salptower.lev
```

3.3 Playing as Client

In Windows, it's extremely easy to play as a client if the server updates its status on our metaserver (by default).

In fact, all you have to do is click the "salp://" links from the metaserver page. To use Figure 4 as an example, clicking the underlined "Super Testy Land" would launch a game of SW, attempting to connect to that server.

Command-line stuff isn't all that difficult either. You need to remember to tell SW that you are running the game in client mode, and need to specify which host (server) to connect to, as in:

```
salpwars mode=client host=141.233.43.47
```

To run the game using User Datagram Protocol[16] (UDP) mode, try:

```
salpwars mode=client host=141.233.43.47 udp=yes
```

With some luck, you should have a game up and running in no time flat.

3.4 Command-Line Arguments

When running SW from the command prompt, you can give the game certain extra commands. These commands are in the form of "`attribute=value`", and appear on the same line as your command to start Salp Wars. Multiple attribute/value pairs are separated by spaces, and can appear in any order.

For example, in Windows you might type the following to run the game in full screen mode:

```
salpwars screen=full
```

If your computer has a slower processor, you might improve performance by having the game omit background and super-foreground drawing by typing:

```
salpwars drawbg=no drawfg=no
```

Notice how the two arguments were separated by whitespace. The *drawbg*, *drawfg* and *screen* arguments are just the tip of the iceberg, though. Table 1 (page 10) has a comprehensive lists of command-line arguments you can play with. If you're not in the mood to mess with it, don't worry. You don't *need* to know how to do this in order to play the game.

[16]Using UDP might make the game seem to react faster to your controls. Depending how far away the server is, it could also make the game movement incredibly crazy and jumpy. You will probably have problems using UDP mode from behind a firewall.

Table 1: Command Line Arguments

Attribute Name	Value Description
`audio`	Specifies whether *any* audio hooks should be used by this copy of the game. To turn audio off, set it to `no`. Note that if a server has `audio` set to `no`, it will still be smart enough to send audio signals to clients (who may have audio hooks enabled). Set to `yes` by default.
`clmode`	Shortcut for `audio=no usejoystick=no screen=no`. Use this (by setting to `yes`) if you want to run the game from the command line (that is, no window and stuff). The "`cl`" stands for "command line." Great for servers who don't want to waste extra CPU cycles on drawing the screen or waste RAM by loading extra sounds and graphics - or for UNIX servers who can't/won't run X. Set to `no` by default.
`debug`	When set to `yes`, extra debug information is displayed while playing the game (mostly to the command line). Set to `no` by default.
`drawbg`	Determines whether background tiles (unnecessary for gameplay) are drawn to the screen. Set to `yes` (default) to have the tiles drawn, or `no` to skip them and speed up gameplay.
`drawfg`	Determines whether super-foreground tiles (unnecessary for gameplay) are drawn to the screen. Set to `yes` (default) to have the tiles drawn, or `no` to skip them and speed up gameplay.
`frame_delay`	Specifies the amount of time to sleep between "frames" (both drawn and animated) in the game, measured in milliseconds. Set to `30` by default. If you want the game to run as fast as possible, set this to `0`.
`height`	Sets the height of the game screen. The higher the screen is, the more you can see and the slower the game will run. Common values are `240` (default) and `480`.
`host`	In *client* mode, this is the Internet Protocol (IP) address of the server you wish to connect to. This address may be either the actual numeric IP address (e.g. `141.233.43.47`), or the domain that resolves to that IP address (e.g. `pizza.potsit.com`). Set to `localhost` by default.
`level`	In *server* mode, specifies the level to be loaded, as a relative path from the game executable. For a list of possible levels, look in your `bin/levels/` directory. Set to `levels/runway.lev` by default. Note that this default value seems to change often during game testing.
`mode`	Determines whether the game will be run in client or server mode. When in server mode, you are actually both a server *and* a client (connected to yourself). Set to `server` (default) for server mode, or `client` for client mode.
`port`	In *client* mode, this is the port the server you wish to connect to. This will usually be close to `12394`, but can vary. In *server* mode, this is the port you wish to use to accept connections. If the desired port is already in use, the game will try incrementally higher port(s). Set to `12394` by default.
`safemode`	Salp Wars attempts to play nicely with your system wherever possible. However, some of its warnings (like "don't run this as root") - though very helpful - may annoy you. To get rid of some warnings, set `safemode` to `no`. It is set to `yes` by default.
`screen`	Toggles full screen mode. Set to `window` (default) to show the game in a separate window, `full` to have the game take up the full screen, or `no` to show no graphics whatsoever.
`tcp_nodelay`	When set to `yes`, sockets will attempt to disable Nagle's algorithm. This basically means that TCP packets will be sent as soon as possible, regardless of their size. This will waste your bandwidth (imagine sending a 3-byte message with a header several times that size), but could improve response times. This works for both client and server sockets; both sides will probably have to enable this before you notice a difference. Set to `no` by default.
`width`	Sets the width of the game screen. The wider the screen is, the more you can see and the slower the game will run. Common values are `320` (default) and `640`.
`udp`	When set to `yes` and if `mode=client`, the client will *request* that the server send it game state info via UDP where possible. This can speed up your response rate, but if you are on a bad (non-LAN) connection, could result in dropped packets, and jumpy character movement - worse than the default TCP setup. Set to `no` by default.

continued on next page

continued from previous page

Attribute Name	Value Description
`updateenv`	When set to `yes`, Salp Wars will try to configure your operating system so that bonus features will work with your game. For example, it will try configuring the registry on Windows machines so that you can click *salp://* URLs in Internet Explorer to join games. Set to `no` by default.
`usejoystick`	When set to `yes` (default), Salp Wars will try using one of your joysticks (or gamepads, or whatever). Set this to `no` to avoid this. You might set this to `no` when running two simultaneous instances of the game on the same box, with one player using the keyboard and the other using a gamepad to play the game.

4 Preference Files

An advanced user might want to customize some of the default game settings. By no means would you *need* to do this, but the option is open should you prefer it.

All preference files mentioned can be found in the "salp/bin/prefs" directory.

4.1 general.ini

Most of the fun preferences are found in the `general.ini` file.

This file contains several lines of key/value pairs. All keys are completely uppercase. The domain of each value depends on its corresponding key. An example general.ini file might look like:

```
PLAYERGENDER         male
PLAYERNAME           Noob
SERVERNAME           Noob's House of Salp
SS_HOST              www.salpwars.com
SS_PATH_REGISTER     /cgi-bin/register_game
SS_UPDATEINTERVAL    4
SS_UPDATETYPE        text
```

In the example above, the "keys" are the upper-case symbols on the left column; corresponding "values" are found in the second column. See Table 4.1 for a description of all the general.ini options.

5 Contributing through SourceForge

Do you feel the urge to have your name forever immortalized in the world of SW? If so, there's no easier way than by directly contributing to the SW project. For a better grasp of what contributions you might be able to make, please visit our SourceForge presence at:

`http://sourceforge.net/projects/salpwars/`

5.1 Reporting Bugs

"A bug in the hand is better than one as yet undetected." - *Anonymous*

Software errors (a.k.a. bugs) are unimpressive, to say the least. Should you notice any errors in the software, please report them through our SourceForge bug tracking system at:

`http://sourceforge.net/tracker/?atid=450787&group_id=47786&func=browse`

Table 2: general.ini Settings

Key Name	Description
`CLIENT_MAX_PORT_DRIFT`	The maximum number of TCP ports to *try* connecting to, from the base port, when setting up a client. Note that the client will only actually connect to one of these ports. Default value is `10`.
`PLAYERGENDER`	The gender of your player. Choose from `male` (default), `female`, or `neuter`.
`PLAYERNAME`	The name of your player. Choose whatever name you want, but keep it short if you don't want to anger fellow players. Max size is 30. Default value is `Noob`.
`SERVERNAME`	The name of your server, as it is listed on the slot server. Default value is `Noob's House of Salp`.
`SERVER_MAX_PORT_DRIFT`	The maximum number of TCP ports to *try* listening to, from the base port, when setting up a server. Note that the server will only actually bind to and listen on one of these ports. Default value is `10`.
`SS_HOST`	In server mode, the hostname of the slot server used to list your game. Default value is `www.salpwars.com`.
`SS_PATH_REGISTER`	In server mode, the path of the CGI script used to list your game on the slot server. Default value is `/cgi-bin/register_game`.
`SS_UPDATEINTERVAL`	In server mode, the integral number of minutes the server should wait before updating its listing on the slot server. The more often you do this, the more bandwidth you waste. When set to `0`, the slot server is not made aware of your game's existence. Default value is `10`.
`SS_UPDATETYPE`	In server mode, the graphic detail that is sent to the slot server at *each* update of the game (that is, every `SS_UPDATEINTERVAL` minutes). From least to most bandwidth usage, acceptable values are: `text` to withhold screenshots (default), `grayscale` to send grayscale screenshots, or `color` to send color screenshots.

5.2 Feature Requests

It's been repeated more times than a Britney Spears song, but *this game is still under development.* Clearly, then, there is ample room for new features.

If the Einstein or Edison in you feels like redefining the world as we know it, there's no better place to begin than in the feature request tracking system at SourceForge:

`http://sourceforge.net/tracker/?atid=450787&group_id=47786&func=browse`

Though the team can't guarantee that every request will be fulfilled, new ideas never hurt anyone - so submit away!

5.3 Tested Platforms

Which team members are testing SW for which platforms?[17] Take a peek at Table 3:

[17]A platform is a combination of the operating system (e.g. Windows98) and computer hardware (e.g. a Pentium III).

Table 3: Tested Platforms

Team Member	Platform	CPU	RAM
Jeff Gauthier	OS X 10.1	???MHz G4	384MB
Tyler Goen	Win98SE	800MHz PIII	256MB
Tyler Goen	Mandrake 8.1	750MHz Duron	384MB
Tyler Goen	Mandrake 8.0	210MHz 604e (PowerPC)	192MB
Daniel R. Lindsley	Slackware 8	400MHz K6-III	256MB
Daniel R. Lindsley	Win2k	850MHz Duron	256MB
Jeff Lucas	Win98SE	300MHz PII	128MB

6 Level Design

6.1 What is a Level?

You can think of a level as a scenario in the game.

6.2 .LEV Files

.LEV files describe the main components of the scenario, bringing together the game maps, starting points, and describing how all the pieces of the puzzle come together to form the scenario.

The file contains one command on each line. Comments may be added *on their own line* by prefixing a hash mark ("#", pound sign, number sign, etc) before the comments. The following commands can be used in level files; though the order of many of these commands doesn't matter, pay special attention to the commands where order *does* matter.

`AUTHOR` ***author_name***

Sets the author of the level to *author_name*, a string. Clearly, credit should be given where it is due. This is a great way to be recognized for your efforts in contributing toward a level.

Feel free to comma-separate names where multiple contributors exist (e.g. "Bob Odenkirk, David Cross"), to use nicknames (e.g. "Quantum Cow, Plat") or omit last names ("Lindsley & Goen"). In short, there are no rules in how you should attribute work, so do it in whichever way you enjoy it the most.

`BACKGROUND` ***tileset_file map_id***

Like the `MAP` command, but the map loaded is treated as a background for the last `MAP` loaded. Thus, you cannot place a `BACKGROUND` command in the file before `MAP` commands are found. You may add multiple backgrounds to a map; when doing so, the first background specified will be drawn on top of the other backgrounds.

`FOREGROUND` ***tileset_file map_id***

Like the `MAP` command, but the map loaded is treated as a foreground for the last `MAP` loaded. Thus, you cannot place a `FOREGROUND` command in the file before `MAP` commands are found. You may add multiple foregrounds to a map; when doing so, the first foreground specified will be drawn on top of the other foregrounds.

`MAP_ALPHA` ***alpha***

Sets the alpha value of the last `MAP`, `BACKGROUND` or `FOREGROUND` loaded to *alpha*, an integer value ranging from 1 (almost transparent) to 254 (almost opaque). If you want a map to be completely transparent, then don't load it at all! Maps are opaque (255) by default.

`MAP_BGALPHA` ***alpha***

Sets the alpha value of the *background colors of the* last `MAP`, `BACKGROUND` or `FOREGROUND` loaded to *alpha*, an integer value ranging from 1 (almost transparent) to 254 (almost opaque). Background colors are opaque (255) by default.

Note that this only sets the alpha value of the background color specified by the `MAP_BGCOLOR` command. If you want to change the alpha value of the map tiles themselves, use the `MAP_ALPHA` command.

`MAP_BGCOLOR` ***r g b***

Sets the background color of the last `MAP`, `BACKGROUND` or `FOREGROUND` to (r, g, b), all integers. If the back-most map shown on the screen has some transparency to it, make sure that you set a `MAP_BGCOLOR` for the map. Otherwise, some weird blurring effects could occur.

`MAP` ***tileset_file map_id***

Loads a new map of id *map_id* found in *tileset_file*. Note that *tileset_file* should NOT contain a suffix. For example, you might set *tileset_file* to “tiles/Runway”, and *map_id* to “Runway1”.

`STARTMAP`

Signifies that the last `MAP` loaded is the starting map in the level. This command is currently required, but may be ignored in the near future when mapcodes are used to determine starting locations.

`TITLE` ***title***

Sets the title of the level to *title*, a string. This title is displayed in the game window’s title bar, and might be used to identify the level through various front-ends (e.g. the Metaserver).

`VERSION` ***version_num***

Sets the version of the level to *version_num*, an integer value. The version number does not currently affect how the game uses the level file.

7 Map Design

7.1 Standard Mapcodes

What follows is a set of standard mapcodes which you can use in your Tile Studio map design. Be warned that the term “standard” is actually a very loose one. Since the support for these is still fairly basic, the meanings of each mapcode could change with time.

If the mapcode tables should change once there is a large userbase, a command will be provided to automagically remap (no pun intended) these mapcodes.

If a mapcode does not perform how you’d expect it to, first please check if this table is out of sync with the source code. You can find the definitions in the file src/Salp/Tile/Map.h

8 Parting Comments

Wow, you made it through the manual! How do you feel? You have all the tools to play a round of the game, so what are you waiting for? Get to it!

Table 4: Standard Mapcodes

Mapcode (hex)	Significance
d0	Spawnpoint for +20 health box
e0	Spawnpoint for shotgun weapon
e1	Spawnpoint for spreadshot weapon
e2	Spawnpoint for frog-salp ammo/weapon
e3	Spawnpoint for grape-salp ammo/weapon
e4	Spawnpoint for homing missle weapon
f0	Starting location(s) for Team A
f0	Starting location(s) for Team A
f1	Starting location(s) for Team B
f2	Starting location(s) for Team C
f3	Starting location(s) for Team D

Bibliography

Callahan, Bryan. *What is the Difference Between Alpha, Beta and Final Software?*. 9 May 2002
`<http://www.white-dwarf.com/support/kb/articles/00,00.php?rate=1>`

Free Software Foundation, Inc. *GNU General Public License*. 9 May 2002
`<http://www.gnu.org/licenses/gpl.txt>`

Glass, John C. *How to be a L33t H4X0r*. 9 May 2002
`<http://www.johncglass.com/leethaxors.htm>`

Karr, Ryan. *Taer*. 9 May 2002
`<http://web.archive.org/web/19991007103523/www.zilla.net/r2/help/taer.shtml>`

Pesch, Roland. *CVS – Concurrent Versions System*. 3 Mar. 2002
`<http://www.loria.fr/~molli/cvs/doc/cvs_toc.html>`

SourceForge, Inc. *SourceForge CVS Repository*. 9 May 2002
`<http://sourceforge.net/cvs/?group_id=47786>`

Wolfsites, LLC. *The Easter Egg Archive*. 9 May 2002
`<http://www.eeggs.com/>`

Index