mmatyas / **supermariowar** Public

ma… · **23** Branches · **12** Tags · Go to file · Go to file · Code

mmatyas Removed some unused externs ✓ a04d6f4 · last month · 1,604 Commits

.github/workflows	Run the Windows tests on AppVeyor	6 months ago
cmake	Show the Git revision info in the title bar	8 months ago
data @ 1139d89	Updated the game data	last month
dependencies	Added doctest 2.4.11	6 months ago
docs	Allow server config to have txt extension	3 years ago
resources	Fixed missing launch argument in the …	3 years ago
src	Removed some unused externs	last month
tests	Improved the tokenizer	last month
.appveyor.yml	Run the Windows tests on AppVeyor	6 months ago
.clang-format	Fixed a minor style issue	7 months ago
.gitattributes	Correctly classify C++ header files on …	7 months ago
.gitignore	Replaced data.zip with a separate sub…	4 years ago
.gitmodules	Updated yaml-cpp to 0.8	last year
CHANGELOG	Always compile netplay code	11 years ago
CMakeLists.txt	Make sure non-void functions always r…	5 months ago
CREDITS	Proper re-organization of text docume…	11 years ago
README.ko.md	Update linux build instructions	7 months ago
README.md	Update linux build instructions	7 months ago
testmaps.zip	Re-added test maps.	9 years ago

About

A fan-made multiplayer Super Mario Bros. style deathmatch game

smwstuff.net

#game #mario #arm #arena #cross-platform #cpp #multiplayer #sdl #emscripten #hacktoberfest #deathmatch #fangame

Readme
Activity
442 stars
35 watching
74 forks
Report repository

Releases 9

Continuous build Latest
on Oct 13, 2020

+ 8 releases

Packages

No packages published

Contributors 13

Languages

● **C++** 98.2% ● **CMake** 1.8%

README

Super Mario War

Build - Linux passing · Build - Windows passing · build passing · Discord

Super Mario War is a fan-made multiplayer Super Mario Bros. style deathmatch game in which players try to beat one another in a variety of gameplay modes. You can play on teams, design your own levels, design your own worlds, and much more!

Read this page in other languages: 한국어

History

The original *Mario War* was created by *Samuele Poletto* in 2002, in which four Super Mario characters could fight on various levels by jumping on each other's head. It was written in Pascal/Assembly, and was released for DOS. Later versions also included a map editor.

In 2004, *Florian Hufsky*, founder of the *72dpiarmy* forum started working on an open-source rewrite, which became *Super Mario War*. This version introduced custom characters, additional gameplay modes and map mechanics, items and powerups. Custom user contents were stored on (the now defunct) `smwstuff.com` site, with thousands of maps and skins available. Due to its open-source nature, the game has been ported to pretty much every desktop and console system.

At the end of 2009, Florian died. The development of the game slowed down and eventually stopped, with SMW 1.8 beta 2 being the last official release. While there have been several attempts for continuing or restarting the development, none of them seem to have succeeded in the long term. In addition, due to technical issues the whole `smwstuff.com` site and all of its contents has also been lost, with partial backups from the uploaders available on the forum.

This is a fork I've started working on around 2014, with the initial goal of implementing network multiplayer support that would work cross-platform between different devices. I've also wanted to make a site for collecting the scattered content from the forum in one place once again. This became `smwstuff.net`.

As for the game, unfortunately it became clear very soon that the quality and structure of the original source code will make effective work impossible: most source files were in the "around 10000 lines" category, with 1000+ LOC functions being common, most of them modifying global variables, and there was even an `if-else` with a hundred branches.

Eventually I've managed to make the network multiplayer working, but it's far from perfect. With good conditions, on local networks it may work fine, but subtle bugs and lag usually makes the gameplay over the internet far from optimal. A proper implementation would likely require a redesign of several core parts of the game and another several hundred clean up/refactoring patches to make the code maintainable. And, with this kind of project of course, there's also a chance that next morning you get a cease-and-desist letter from whatever company.

For me, this was the point where I've stopped putting more time on this project (2016). In the long term, a complete rewrite might be faster and more effective than patching the original game for years. Either way, I hope I've left the project in a better state than it was, and that it will be of use for future developers. Have fun!

Old releases for Mario War and SMW can be found here.

Main changes since 1.8

- Experimental online mode, including an online lobby server
- Lots and lots and lots of bugfixes (seriously, there's at least 200)
- Added Web, ARM and (an experimental) Android ports, and improved support for Windows, Linux and OSX
- Improved music and sound quality
- Ported all parts of the game to SDL2, for hardware accelerated drawing and improved support for many platforms
- Major cleanup of the source code and the build system

About

Super Mario War is a Super Mario multiplayer game. The goal is to stomp as many other Mario's as possible to win the game. There is a range of different gameplay modes in the game, like Capture-The-Flag, King of The Hill, Deathmatch, Team Deathmatch, Tournament Mode, Collect The Coins, Race, and many more. The game also includes a level editor which lets you create your own maps from scratch, or re-create sections from your favorite Mario game, your imagination is the limit! Recently included is a world editor, which strings a bunch of levels with specified conditions to create an SMB3-like experience merged with tournament like play. The game is more importantly a tribute to Nintendo and the original fangame game Mario War by Samuele Poletto.

The game uses artwork and sounds from Nintendo games. We hope that this noncommercial fangame qualifies as fair use work. We just wanted to create this game to show how much we adore Nintendo's characters and games.

Features

- Up to four players deathmatch fun
- a whole bunch of game modes (featuring GetTheChicken, Domination, CTF, ...)
- online and local multiplayer
- Comes with the leveleditor - you can create your own maps...
- ... and a lot of people did so. There are currently over 1000 maps
- More fun than poking a monkey with a stick
- The whole source code of the game is available, for free
- uses SDL and is fully portable to windows, linux, mac, ...
- CPU Players
- will make you happy and gives you a fuzzy feeling

Supported platforms

- Linux
- Windows
- Mac OS X
- ARM devices (eg. Raspberry Pi)
- XBox (?)
- Android (experimental)
- asm.js (experimental)

Building instructions

Requirements

- C++11 supporting compiler (eg. gcc-4.8)
- CMake (>= 2.6)
- SDL (1.2 or 2.0), with
  - SDL_image
  - SDL_mixer
- zlib
- yaml-cpp (included)
- ENet (optional, included)

You can use package managers for getting these dependencies:

- Debian-based: `apt-get install cmake libsdl1.2-dev libsdl-image1.2-dev libsdl-mixer1.2-dev zlib1g-dev`
- Fedora/RPM: `yum install cmake SDL-devel SDL_image-devel SDL_mixer-devel zlib-devel`
- Arch: `pacman -S cmake sdl sdl_image sdl_mixer zlib`
- MSYS2: `pacman -S mingw-w64-x86_64-SDL mingw-w64-x86_64-SDL_image mingw-w64-x86_64-SDL_mixer mingw-w64-x86_64-zlib`

For other systems, you can download the development files manually from:

- http://www.cmake.org/cmake/resources/software.html#latest
- http://www.libsdl.org/download-1.2.php
- http://www.libsdl.org/projects/SDL_image/release-1.2.html
- http://www.libsdl.org/projects/SDL_mixer/release-1.2.html
- http://zlib.net

Get the code

This repository contains some submodules which you can use if the dependencies are not available for your OS, are outdated or you simply don't want to install them on your system. To use the included libraries, do a recursive cloning:

`git clone --recursive https://github.com/mmatyas/supermariowar.git`

Alternatively, you can also initialize the submodules manually:

```
git clone https://github.com/mmatyas/supermariowar.git
cd supermariowar
git submodule update --init
```

If you'd rather use the system libraries, please see the Build configuration section for disabling this feature.

Linux

Create a build directory and run CMake there to configure the project. Then simply call `make` every time you want to build. In short:

```
mkdir build && cd build
cmake ..
make -j4 # -jN = build on N threads
./smw --datadir ../data
```

The main build targets for `make` are:

- smw
- smw-leveleditor
- smw-worldeditor
- smw-server

If you prefer to work within an IDE (CodeBlocks, Eclipse, ...), you can also generate project files for it using CMake. You can find more information in Build configuration.

To create installable packages, simply run `make package`. This will create TGZ, DEB and RPM archives.

Mac OSX

You'll probably need Xcode and its command line tools; you can install SDL and CMake manually from its site, or you can get them with Homebrew: `brew install cmake sdl2 sdl2_image sdl2_mixer`. Then follow the Linux instructions to build SMW.

Windows

If you're using MinGW Shell/MSYS or Cygwin, you can follow the Linux guide. You can also generate a project file with CMake for various IDEs, such as CodeBlocks, Eclipse or Visual Studio.

For more details, see the wiki: Building on Windows.

ARM devices

You can build SMW on ARM devices, like the Raspberry Pi, following the Linux instructions. If you know how to do it, you can also cross-compile the usual way, either by setting up a cross toolchain or emulating your device. For more details, see the wiki: Cross compiling to ARM.

The build configuration contains some default compiler flags already, but since there are many possible combinations (hard float, Thumb, NEON, ...), you might want to use custom parameters. In this case, define the CFLAGS and CXXFLAGS vars, and run CMake with the `DISABLE_DEFAULT_CFLAGS` option (see Build configuration).

Android

The Android port uses a different build system, you can find more details here.

Emscripten

SMW can be build to run in your browser using Emscripten. You can find the build instructions in the wiki.

Other devices

You should be able to port SMW to any device where SDL (either 1.2 or 2.0) works. Generally, this involves the following steps:

- get the cross compiler toolchain of your device
- cross-compile the SDL libs, if they are not included
    - build SDL
    - build SDL_image with at least PNG support
        - requires zlib
        - requires libpng
    - build SDL_mixer with at least OGG support
        - requires libogg
        - requires libvorbis
- create a CMake toolchain file and define your compiler and paths
- build using the toolchain file

Build configuration

TODO: expand this section

You can change the build configuration by setting various CMake flags. The simplest way to do this is by running `cmake-gui ..` from the `Build` directory. You can read a short description of an element by hovering the mouse on its name too.

Alternatively, you can pass these options directly to CMake as `-DFLAGNAME=VALUE` (eg. `cmake .. -DUSE_SDL2_LIBS=1`).

How to play

Please see documentation in the docs/ directory.