master 2 Branches 3 Tags Go to file Go to file Code

snesrev Add BPS support fbbb3f9 · last year 314 Commits

.github	Added Nintendo Switch build target. (#…	last year
assets	Move around files	last year
other	Add support for german translation	last year
saves/ref	Initial version	2 years ago
snes	Move around files	last year
src	Add BPS support	last year
third_party	Add support for german translation	last year
.gitignore	Move around files	last year
LICENSE.txt	Opus MSU support and MSU Deluxe s…	2 years ago
Makefile	Move around files	last year
README.md	Move around files	last year
Zelda3.sln	Add MSVC configuration to deploy all/…	2 years ago
extract_assets.bat	Move around files	last year
packages.config	Merge pull request #7 from kageurufu/…	2 years ago
requirements.txt	Add requirements.txt. Fixes #60	2 years ago
run_with_tcc.bat	Move around files	last year
zelda3.ini	Add support for german translation	last year
zelda3.vcxproj	Add BPS support	last year
zelda3.vcxproj.filters	Move around files	last year

README License

Zelda3

A reimplementation of Zelda 3.

Our discord server is: https://discord.gg/AJJbJAzNNJ

About

This is a reverse engineered clone of Zelda 3 - A Link to the Past.

It's around 70-80kLOC of C code, and reimplements all parts of the original game. The game is playable from start to end.

You need a copy of the ROM to extract game resources (levels, images). Then once that's done, the ROM is no longer needed.

It uses the PPU and DSP implementation from LakeSnes, but with lots of speed optimizations. Additionally, it can be configured to also run the original machine code side by side. Then the RAM state is compared after each frame, to verify that the C implementation is correct.

I got much assistance from spannerism's Zelda 3 JP disassembly and the other ones that documented loads of function names and variables.

Additional features

A bunch of features have been added that are not supported by the original game. Some of them are:

Support for pixel shaders.

Support for enhanced aspect ratios of 16:9 or 16:10.

Higher quality world map.

Support for MSU audio tracks.

About

discord.gg/AJJbJAzNNJ

Readme
View license
Activity
4.3k stars
91 watching
355 forks
Report repository

Releases 1

v0.3 Latest
on Aug 17, 2023

Packages

No packages published

Contributors 25

+ 11 contributors

Languages

C 92.7% Python 6.9% Other 0.4%

Secondary item slot on button X (Hold X in inventory to select).

Switching current item with L/R keys.

How to Play:

Option 1: Launcher by RadzPrower (windows only) https://github.com/ajohns6/Zelda-3-Launcher

Option 2: Building it yourself

Visit Wiki for more info on building the project: https://github.com/snesrev/zelda3/wiki

Installing Python & libraries on Windows (required for asset extraction steps)

1. Download Python installer and install with "Add to PATH" checkbox checked
2. Open the command prompt
3. Type `python -m pip install --upgrade pip pillow pyyaml` and hit enter
4. Close the command prompt

Compiling on Windows with TCC (1mb Tiny C Compiler)

1. Download the project by clicking "Code > Download ZIP" on the github page
2. Extract the ZIP to your hard drive
3. Place the USA rom named `zelda3.sfc` in the root directory.
4. Double-click `extract_assets.bat` in the main dir to create `zelda3_assets.dat` in that same dir
5. Download TCC and extract to the "\third_party" subfolder
6. Download SDL2 and extract to the "\third_party" subfolder
7. Double-click `run_with_tcc.bat` in the main dir to create `zelda3.exe` in that same dir
8. Configure with `zelda3.ini` in the main dir

Compiling on Windows with Visual Studio (4.5gb IDE and compiler)

Same Steps 1-4 above
8. Double-click `Zelda3.sln`
9. Install the **Desktop development with C++** workload with the VS Installer if you don't have it already (it should prompt you to do this).
10. Change "debug" to "release" in the top dropdown
12. Choose "build > build Zelda3" in the menu to create `zelda3.exe` in the "/bin/release" subfolder
13. Configure with `zelda3.ini` in the main dir

Installing libraries on Linux/MacOS

1. Open a terminal
2. Install pip if not already installed

```
python3 -m ensurepip
```

3. Clone the repo and `cd` into it

```
git clone https://github.com/snesrev/zelda3
cd zelda3
```

4. Install requirements using pip

```
python3 -m pip install -r requirements.txt
```

5. Install SDL2

- Ubuntu/Debian `sudo apt install libsdl2-dev`
- Fedora Linux `sudo dnf install SDL2-devel`
- Arch Linux `sudo pacman -S sdl2`
- macOS: `brew install sdl2` (you can get homebrew here)

Compiling on Linux/MacOS

1. Place your US ROM file named `zelda3.sfc` in `zelda3`
2. Compile

```
make
```

▶ Advanced make usage ...

Nintendo Switch

You need DevKitPro and Atmosphere installed.

```
(dkp-)pacman -S git switch-dev switch-sdl2 switch-tools
cd platform/switch
make # Add -j$(nproc) to build using all cores ( Optional )
# You can test the build directly onto the switch ( Optional )
nxlink -s zelda3.nro
```

More Compilation Help

Look at the wiki at https://github.com/snesrev/zelda3/wiki for more help.

The ROM needs to be named `zelda3.sfc` and has to be from the US region with this exact SHA256 hash `66871d66be19ad2c34c927d6b14cd8eb6fc3181965b6e517cb361f7316009cfb`

In case you're planning to move the executable to a different location, please include the file `zelda3_assets.dat`.

Usage and controls

The game supports snapshots. The joypad input history is also saved in the snapshot. It's thus possible to replay a playthrough in turbo mode to verify that the game behaves correctly.

The game is run with `./zelda3` and takes an optional path to the ROM-file, which will verify for each frame that the C code matches the original behavior.

Button	Key
Up	Up arrow
Down	Down arrow
Left	Left arrow
Right	Right arrow
Start	Enter
Select	Right shift
A	X
B	Z
X	S
Y	A
L	C
R	V

The keys can be reconfigured in zelda3.ini

Additionally, the following commands are available:

Key	Action
Tab	Turbo mode
W	Fill health/magic
Shift+W	Fill rupees/bombs/arrows
Ctrl+E	Reset
P	Pause (with dim)
Shift+P	Pause (without dim)
Ctrl+Up	Increase window size
Ctrl+Down	Decrease window size
T	Toggle replay turbo mode
O	Set dungeon key to 1
K	Clear all input history from the joypad log
L	Stop replaying a shapshot
R	Toggle between fast and slow renderer
F	Display renderer performance
F1-F10	Load snapshot
Alt+Enter	Toggle Fullscreen
Shift+F1-F10	Save snapshot

Key	Action
Ctrl+F1-F10	Replay the snapshot
1-9	Load a dungeons playthrough snapshot
Ctrl+1-9	Run a dungeons playthrough in turbo mode

License