port **4** Branches **2** Tags Go to file Go to file Code ···

This branch is 844 commits ahead of, 60 commits behind `n64decomp/perfect_dark:master` . #507

fgsfdsfgs port: rename random to rngRandom to avoid conflicts with libc ✓

06dcbcc · yesterday 7,532 Commits

.github/workflows	ci: only build the port branch for nswitch	2 days ago
cmake	port: build: move to CMake	last week
dist	port: nswitch: basic Nintendo Switch port	5 days ago
docs	Name almost all BG symbols	last year
include	port: 64-bit: replace another long	last week
ld	Rename vm.c to vminit.c, tlb.s to vm.s …	last year
port	port: bring back the host_ structs for fil…	4 days ago
src	port: rename random to rngRandom to…	yesterday
tools	ci: nswitch: add autobuilds	2 days ago
.gitignore	Merge branch 'fgs-port' into port-x64	last month
.gitmodules	Remove support for qemu-irix and req…	2 years ago
CMakeLists.txt	build: nswitch: use actual AND in the a…	3 days ago
LICENSE	Add MIT license	2 years ago
README.md	port: fix typo	2 days ago
checksums.jpn-final.md5	Merge lang json files	2 years ago
checksums.ntsc-1.0.md5	Merge lang json files	2 years ago
checksums.ntsc-beta.md5	Merge lang json files	2 years ago
checksums.ntsc-final.md5	Merge lang json files	2 years ago
checksums.pal-beta.md5	Merge lang json files	2 years ago
checksums.pal-final.md5	Merge lang json files	2 years ago
stagetable.txt	Initial commit	5 years ago

About

work in progress port of n64decomp/perfect_dark to modern platforms

Readme
MIT license
Activity
1.2k stars
45 watching
74 forks
Report repository

Releases 1

Latest automatic build (port br… Latest
on May 14

Packages

No packages published

Languages

● **C** 90.8%	● **Python** 4.2%
● **C++** 2.5%	● **Assembly** 2.4%
● **CMake** 0.1%	● **Shell** 0.0%

README MIT license

Perfect Dark port

This repository contains a work-in-progress port of the Perfect Dark decompilation to modern platforms.

To run the port, you must already have a Perfect Dark ROM, specifically one of the following:

- `ntsc-final` / `US V1.1` / `US Rev 1` (md5 `e03b088b6ac9e0080440efed07c1e40f`).
  This is the recommended version to use.
  Called `NTSC version 8.7 final` on the boot screen.
- `ntsc-1.0` / `US V1.0` (md5 `7f4171b0c8d17815be37913f535e4e93`).
  Technically supported, but not recommended.
  Called `NTSC version 8.7 final` on the boot screen as well.
- `jpn-final` (md5 `538d2b75945eae069b29c46193e74790`).
  Technically supported, but requires a separate custom-built executable.
  Called `JPN version 8.9 final` on the boot screen.
- `pal-final` (md5 `d9b5cd305d228424891ce38e71bc9213`).
  Technically supported, but requires a separate custom-built executable.
  Called `PAL 8.7 final` on the boot screen.

Status

The game is in a mostly functional state, with both singleplayer and split-screen multiplayer modes fully working. There are minor graphics- and gameplay-related issues, and possibly occasional crashes.

The following extra features are implemented:

- mouselook;
- dual analog controller support;
- widescreen resolution support;
- configurable field of view;
- 60 FPS support, including fixes for some framerate-related issues;
- fixes for a couple original bugs and crashes;
- basic mod support, currently enough to load a few custom levels;
- slightly expanded memory heap size;
- experimental high framerate support (up to 240 FPS):
  - set `Game.TickRateDivisor` to `0` in `pd.ini` to activate;
  - in practice the game will have issues running faster than ~165 FPS, so use VSync or `Video.FramerateLimit` to cap it.
- emulate the Transfer Pak functionality the game has on the Nintendo 64 to unlock some cheats automatically.

Currently both 32-bit and 64-bit platforms should be supported.
The port is tested on i686 and x86_64, both on Windows and on Linux.
ARM platforms might be supported, but are mostly untested.

Download

Latest automatic builds for supported platforms:

- x86_64-windows
- i686-windows
- x86_64-linux
- i686-linux
- arm64-nswitch

If you are looking for netplay builds (the `port-net` branch), see this link.

Running

You must already have a Perfect Dark ROM to run the game, as specified above.

This assumes that you're using an x86_64 build. If you aren't, replace `x86_64` below with your arch (e.g. `i686`).

1. Create a directory named `data` next to `pd.x86_64` if it's not there.
2. Put your Perfect Dark NTSC ROM named `pd.ntsc-final.z64` into it.
3. Run the `pd.x86_64` executable.

If you want to use a PAL or JPN ROM instead, put them into the `data` directory and run the appropriate executable:

- PAL: ROM name `pd.pal-final.z64`, executable name `pd.pal.x86_64`.
- JPN: ROM name `pd.jpn-final.z64`, executable name `pd.jpn.x86_64`.

Optionally, you can also put your Perfect Dark for GameBoy Color ROM named `pd.gbc` in the `data` directory if you want to emulate having the Nintendo 64's Transfer Pak and unlock some cheats automatically.

Additional information can be found in the wiki.

A GPU supporting OpenGL 3.0 or above is required to run the port.

Installing the Nintendo Switch version

The Nintendo Switch build ZIP comes with all 3 regions in different folders: `perfectdark`, `perfectdark_pal` and `perfectdark_jpn`.

Take the folder for the region you want and put it into the `/switch` folder on your SD card, then put your ROM into the `data` folder inside of the folder you extracted as described above.

Controls

1964GEPD-style and Xbox-style bindings are implemented.

N64 pad buttons X and Y (or `X_BUTTON`, `Y_BUTTON` in the code) refer to the reserved buttons `0x40` and `0x80`, which are also leveraged by 1964GEPD.

Support for one controller, two-stick configurations are enabled for 1.2.

Note that the mouse only controls player 1.

Controls can be rebound in `pd.ini`. Default control scheme is as follows:

Action	Keyboard and mouse	Xbox pad	N64 pad
Fire / Accept	LMB/Space	RT	Z Trigger
Aim mode	RMB/Z	LT	R Trigger
Use / Cancel	E	N/A	B

Action	Keyboard and mouse	Xbox pad	N64 pad
Use / Accept	N/A	A	A
Crouch cycle	N/A	L3	`0x80000000` (Extra)
Half-Crouch	Shift	N/A	`0x40000000` (Extra)
Full-Crouch	Control	N/A	`0x20000000` (Extra)
Reload	R	X	X `(0x40)`
Previous weapon	Mousewheel forward	B	D-Left
Next weapon	Mousewheel back	Y	Y `(0x80)`
Radial menu	Q	LB	D-Down
Alt fire mode	F	RB	L Trigger
Alt-fire oneshot	`F + LMB` or `E + LMB`	`A + RT` or `RB + RT`	`A + Z` or `L + Z`
Quick-detonate	`E + Q` or `E + R`	`A + B` or `A + X`	`A + D-Left` or `A + X`

Building

Windows

1. Install MSYS2.
2. Open the `MINGW64` prompt if building for x86_64, or the `MINGW32` prompt if building for i686. (**NOTE:** *do not* use the `MSYS` prompt)
3. Install dependencies:
   `pacman -S mingw-w64-x86_64-toolchain mingw-w64-x86_64-SDL2 mingw-w64-x86_64-zlib mingw-w64-x86_64-cmake mingw-w64-x86_64-python3 mingw-w64-i686-toolchain mingw-w64-i686-SDL2 mingw-w64-i686-zlib mingw-w64-i686-cmake mingw-w64-i686-python3 make git`
4. Get the source code:
   `git clone --recursive https://github.com/fgsfdsfgs/perfect_dark.git && cd perfect_dark`
5. Run `cmake -G"Unix Makefiles" -Bbuild .`.
   - Add `-DROMID=pal-final` or `-DROMID=jpn-final` at the end of the command if you want to build a PAL or JPN executable respectively.\
6. Run `cmake --build build -j4 -- -O`.
7. The resulting executable will be at `build/pd.x86_64.exe` (or at `build/pd.i686.exe` if building for i686).
8. If you don't know where you downloaded the source to, you can run `explorer .` to open the current directory.

Linux

1. Ensure you have gcc, g++ (version 10.0+), make, cmake, git, python3 and SDL2 (version 2.0.12+), libGL and ZLib installed on your system.
   - If you wish to crosscompile, you will also need to have libraries and compilers for the target platform installed, e.g. `gcc-multilib` and `g++-multilib` for x86_64 -> i686 crosscompilation.
2. Get the source code:
   `git clone --recursive https://github.com/fgsfdsfgs/perfect_dark.git && cd perfect_dark`
3. Run the following command:
   - `cmake -G"Unix Makefiles" -Bbuild .`
   - Add `-DROMID=pal-final` or `-DROMID=jpn-final` at the end of the command if you want to build a PAL or JPN executable respectively.
   - Add `-DCMAKE_C_FLAGS=-m32 -DCMAKE_CXX_FLAGS=-m32` at the end of the command if you want to crosscompile from x86_64 to x86.
4. Run `cmake --build build -j4`.
5. The resulting executable will be at `build/pd.<arch>` (for example `build/pd.x86_64`).

Nintendo Switch

1. Set up the devkitA64 environment.
   - On Windows you can do it under MSYS2 or WSL, usually MSYS2 is recommended.
   - If using MSYS2, make sure to use the **MSYS2** shell, **not** MINGW32 or MINGW64.
2. Install host dependencies:
   - On MSYS2: execute command `pacman -Syuu && pacman -S git make cmake python3`
   - On Linux: use your package manager as normal to install the above dependencies.
3. Install Switch toolchain and dependencies:
   - Execute commands:
     ```
     dkp-pacman -Syuu
     dkp-pacman -S devkitA64 libnx switch-zlib switch-sdl2 switch-cmake dkp-toolchain-vars
     ```
   - If in MSYS2 or `dkp-pacman` doesn't work, replace it with just `pacman`.
4. Ensure devkitA64 environment variables are set:
   - Execute command: `source /opt/devkitpro/switchvars.sh`
   - If your `$DEVKITPRO` path is different, substitute that instead or set the variables manually.

5. Configure:
    - Execute command: `aarch64-none-elf-cmake -G"Unix Makefiles" -Bbuild .`
    - Add `-DROMID=pal-final` or `-DROMID=jpn-final` at the end of the command if you want to build a PAL or JPN executable respectively.
6. Build:
    - Execute command: `make -C build -j4`
7. The resulting executable will be at `build/pd.arm64.nro`.

Notes

Alternate compilers or toolchains can be specified by passing `-DCMAKE_TOOLCHAIN_FILE=whatever` as normal. The port does not build with Visual Studio.

You will need to provide a `jpn-final` or `pal-final` ROM to run executables built for those regions, named `pd.jpn-final.z64` or `pd.pal-final.z64`.

It might be possible to build and run the game on an ARM/AArch64 platform, but this has not been tested.

Credits

- the original decompilation project authors;
- Ryan Dwyer for the above, additional help, and `pd-extract`;
- doomhack for the only other publicly available PD porting effort I could find;
- sm64-port authors for the audio mixer and some other changes;
- Ship of Harkinian team, Emill and MaikelChan for the libultraship version of fast3d that this port uses;
- lieff for minimp3;
- Mouse Injector and 1964GEPD authors for some of the 60FPS- and mouselook-related fixes;
- Raf for the 64-bit port;
- NicNamSam for the icon;
- everyone who has submitted pull requests and issues to this repository and tested the port;