Game Dev: PhysFS

Ricard Pillosu - UPC

Abstracting Access to Files

- Every OS has a different mix File Systems:
  - Windows / Xbox: NTFS, FAT, FAT32, exFAT
  - Linux / PlayStation: ext3, ext4, Reiser
  - MacOS / iOS: HFS+, HFSX, APFS
- They basically improve on the previous one supporting bigger file names, deeper directory nesting, **Journaling**, etc.

Major differences

*NIX	WINDOWS
Case sensitive	Case insensitive
Use “/” as separator	Use “\” as separator
Drives are mounted as directories	Drives use letters (C: D: ...)
Hidden files start with “.” (.secrets)	Hidden files use a metadata flag
Very long file names supported	Supported as workaround the 8.3 MS-DOS format
Extensions are not a standard	Extensions are well spread as a standard
Native support for symbolic links	Symbolic links not supported

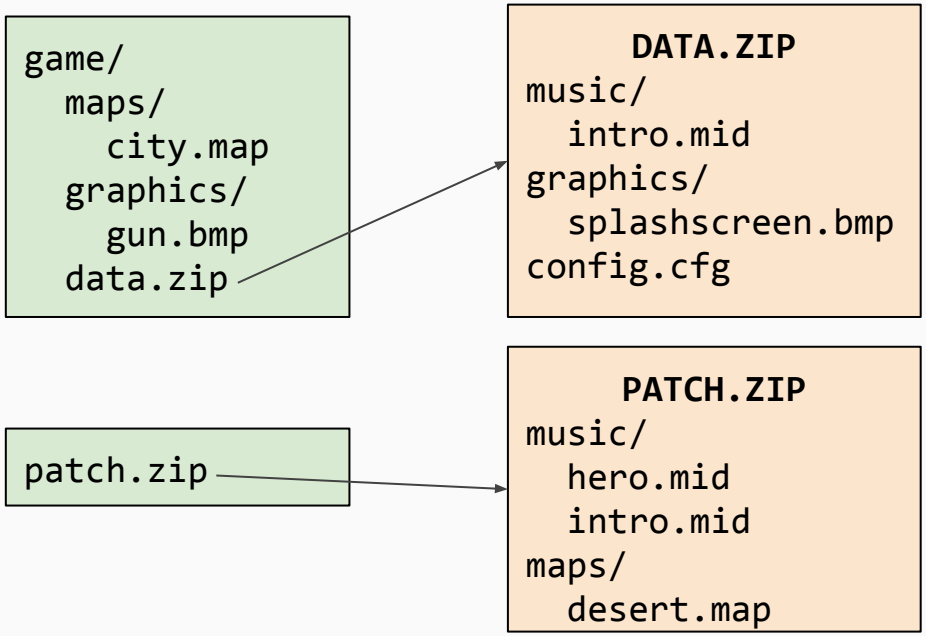
File System concepts

- Current folder is expressed with “.” (e.g. *./hello.txt*)
- Previous folder is expressed with “..” (e.g. *../../folder/file.txt*)
- When executing our game:
  - Installation directory: where the .exe file is located
  - Working directory: the directory from that .exe was called.

Introducing PhysFS

- Access transparently ZIP/7Z/PAK/… packages
- Abstracts access to windows / *nix filesystems (ported to ps/xbox)
- Substitutes fopen()/fread()/fwrite()/fclose()/mkdir()/... functions
- Supports directory listing, mounting and buffering
- Limits writing to specific folder, stopping hacker behaviour

Virtual File System - Quake style



Search for “/*”:

- maps/
- graphics/
- music/
- config.cfg

Search for “maps/*”:

- desert.map
- city.map

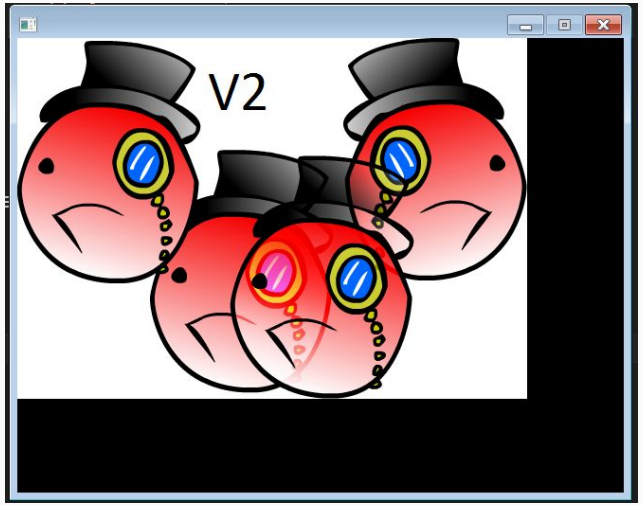
Search for “graphics/*”:

- gun.bmp
- splashscreen.bmp

Search for “music/*”:

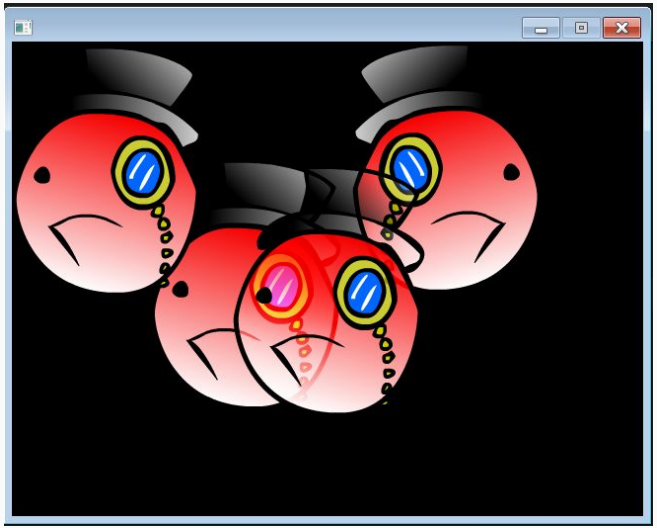
- intro.mid *(from patch.zip)*
- hero.mid

Open Handout zip and execute game/solution.exe



Files in the folder take precendence over the contents of data.zip

Try changing the name of *game_test/textures/test.png*



TODO 1

“Init PhysFS lib / stop PhysFS lib”

- Check the documentation here
- You will need to use
  - int PHYSFS_init (const char *argv0)
  - int PHYSFS_deinit (void)

TODO 2

*“Mount directory "**.**", then mount "**data.zip**" files in the folder should take precendence over the zip file!”*

- Read carefully the documentation for PHYSFS_mount()
- Now use it to “mount” the current folder “.”
- The mount data.zip, making sure it will looked at *last*

TODO 3

“Read a file and allocate needed bytes to buffer and write all data into it return the size of the data”

- You will need to use PHYSFS_openRead()
- Calculate the total size in bytes with PHYSFS_fileLength()
- Allocate memory on *buffer and fill it calling PHYSFS_read()
- Make sure to check for errors after every call using PHYSFS_getLastError()

TODO 4

“Using our previous Load method create a sdl rwops using SDL_RW_From_ConstMem() and return it if everything goes well”

- We just want to copy all data to memory
- Then we create a RWops for sdl with a call to *SDL_RW_From_ConstMem()*
- Use the little function *close_sdl_rwops()* as a callback to delete memory

TODO 5

“Instead of reading directly from HD, use our new load methods from filesystem module You will need to use IMG_Load_PNG_RW() instead”

- UYou need to dwap one line
- We should never access the filesystem directly anymore
- Everything goes via PHYSFS now
- Use *IMG_Load_PNG_RW()* and pass it the resulting RWops from Load method on filesystem module

TODO 6

“Same as with the textures, use the proper RW loading function”

- Just for the music right now
- Find the proper RW function inside the Mix library

Documentation

- Documentation on SDL RWops
- PhysFS
- PhysFS API
- PhysFS Tutorial

Homework

- Load audio fx via RWops
- Add utility methods:
  - Exist(), Size()
  - CreateDir(), IsDir()
  - Delete()
- Enable a writing directory (check **SDL_GetPrefPath()**)
- Add a method to write a buffer into a file