Binocle

Valerio Santinelli

Apr 05, 2022

CONTENTS:

This is the main documentation of Binocle, a C engine mainly aimed at game development.

The previous incarnation was a C++ engine with way more features than this one, but I wanted to get back to the basics and trim everything down to a more manageable framework without all the bloat that C++ carries around.

It's born out of the need for the following features:

- Cross-platform compilation (macOS, Windows, iOS, Android, Web)
- OpenGL ES 2 support (but you can use any variant of OpenGL as long as it's supported by your hardware)

Nothing too fancy, but still something I always need when I make 2D or 3D games and prototypes.

Warning: The API is evolving all the time but the core is pretty stable. I keep adding and tweaking stuff based on my needs, so things may change without notice.

CHAPTER ONE

FEATURES

- Cross-platform: macOS, Windows, iOS, Android, Web (Linux planned)
- OpenGL API (ES 2/3 on mobile platforms)
- 2D Sprites
- Spritesheets (TexturePacker format. LibGDX format is in the works)
- Sprite batching
- Music and sound effects
- Bezier paths that can be used for anything
- BitmapFont fonts
- 2D Camera
- 2D Collisions (boxes and circles)
- Easing functions
- Entity Component System
- Timing functions
- Viewport adapters for 2D pixel perfect images
- Experimental hot code reloading for game code
- Lua scripting through LuaJIT on supported platforms

1.1 Third party libraries

Binocle sits on the shoulders of giants. I tried to keep the amount of external libraries to a minimum. The current libraries are the following:

- SDL by the almighty Ryan C. Gordon (OS abstraction)
- miniaudio (cross-platform audio support)
- zlib
- Vorbis by the Xiph.Org Foundation
- OGG by the Xiph.Org Foundation
- FreeType
- Dear ImGui

- glew (for Windows OpenGL support)
- Kazmath by Luke Benstead
- stbimage
- parson
- LuaJIT
- sokol_time

1.2 Coordinate system

Binocle uses a right-handed coordinate system which is the same used by OpenGL

CHAPTER

TWO

INSTALLING

The easiest way to work with Binocle is to use the CLI project manager called *bone*. With *bone* you can initialize a new project and automate the compilation and linking scripts. It's a quite young tool so please remember to make regular backups of your projects.

2.1 Installing *bone*

You can grab *bone* from the GitHub repo. You can either download the binaries on the Releases page or compile it yourself.

2.2 Creating, building, running, updating and upgrading a project

Please refer to the documentation of bone to setup your first Binocle application.

CHAPTER

THREE

BUILDING THE LIBRARY

The whole Binocle toolchain is based on CMake and makes it quite easy to build for different architectures. I usually suggest to use *bone* to build your project, but if you really need to compile the library by hand, here are the steps to follow for each and every supported platform.

3.1 macOS

```
cd build/macosx/gen
cmake -G Xcode -D DEBUG=1 ../../..
```

3.2 Windows

I usually run the CMake GUI tool and select the Visual Studio generator there. That's pretty much all that's needed.

3.3 Android

You will need the Android SDK and NDK and the correct environment variables for this to work.

```
cd build/android/gen
cmake -D DEBUG=1 -D ANDROID_ABI=armeabi -D ANDROID_STL=c++_static -D ANDROID_
↪PLATFORM=android-21 -D CMAKE_TOOLCHAIN_FILE=../../cmake/android.toolchain.cmake ../../.
↪.
make
cmake -D DEBUG=1 -D ANDROID_ABI=armeabi-v7a -D ANDROID_STL=c++_static -D ANDROID_
↪PLATFORM=android-21 -D CMAKE_TOOLCHAIN_FILE=../../cmake/android.toolchain.cmake ../../.
↪.
make
cmake -D DEBUG=1 -D ANDROID_ABI=x86_64 -D ANDROID_STL=c++_static -D ANDROID_
↪PLATFORM=android-21 -D CMAKE_TOOLCHAIN_FILE=../../cmake/android.toolchain.cmake ../../.
↪.
make
cd ../android-project
./gradlew installDebug
```

3.4 iOS

You will need the latest Xcode and its command line tools.

```
cd build/ios/gen
cmake -G Xcode -D DEBUG=1 -D IOS=1 ../../..
```

3.5 Emscripten (web)

You need a recent version of Emscripten installed on your system. If you're using macOS, just do a *brew install emscripten* to set it up.

```
cd build/emscripten/gen
emcmake cmake ../../.. -DCMAKE_BUILD_TYPE=Release
make -j8
cd example/src
python -m SimpleHTTPServer 8000
open http://localhost:8000/ExampleProject.html
```

CHAPTER

FOUR

API REFERENCE

api/app

api/atlas

api/audio

api/bezier

api/bitmapfont

api/blend

api/camera

api/collision

api/color

api/easing

api/ecs

api/fs

api/game

api/gd

api/image

api/input

api/log

api/lua

api/material

api/math

api/platform

api/render_state

api/sdl

api/shader

api/sprite

api/subtexture

api/texture

api/timer

api/viewport_adapter

api/vpct

api/window

CHAPTER FIVE

INDICES AND TABLES

- genindex
- modindex
- search