bgfx 初步使用讲解

Dezeming Family

2023 年 1 月 5 日

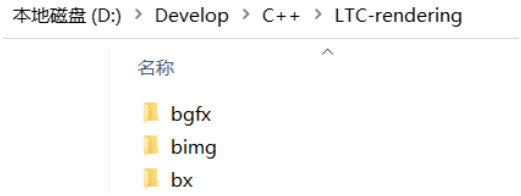
DezemingFamily 系列书和小册子因为是电子书，所以可以很方便地进行修改和重新发布。如果您获得了 DezemingFamily 的系列书，可以从我们的网站 [https://dezeming.top/] 找到最新版。对书的内容建议和出现的错误欢迎在网站留言。

bgfx 是一个跨平台的图形 API，可以用来构建属于自己风格的渲染引擎，支持 Direct3D、OpenGL、Metal 和 WebGL 等渲染底层。支持 Linux、Windows 和 Android 等平台，并且支持多种编程语言。

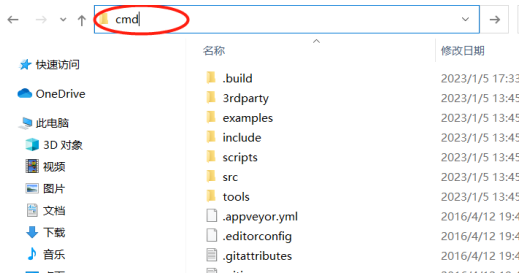
一　建立工程

bgfx 使用 GENie - Project generator tool 来生成工程，该工具可以生成 GNU Makefile、XCode 以及 VS 工程等。需要 Visual Studio 版本：VS2017 及以上。

我们只介绍 windows 平台下的操作。首先把这三个工程目录放在一起：



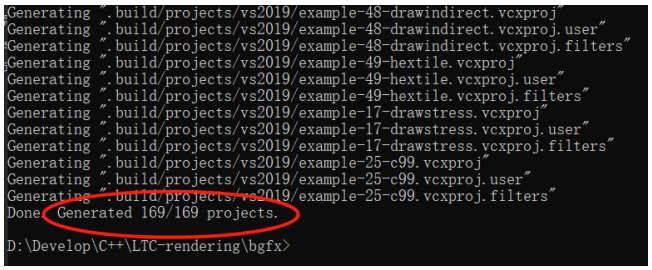
然后进入 bgfx 目录，在目录地址处输入”cmd”，然后回车：



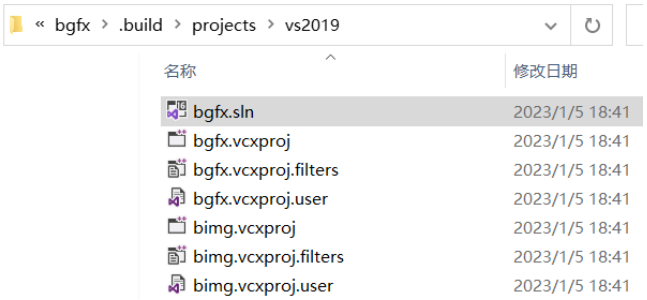
在弹出的 cmd 命令行中输入（我搭建的环境是 VS2019）：

```
1 ..\bx\tools\bin\windows\genie --with-examples vs2019
```

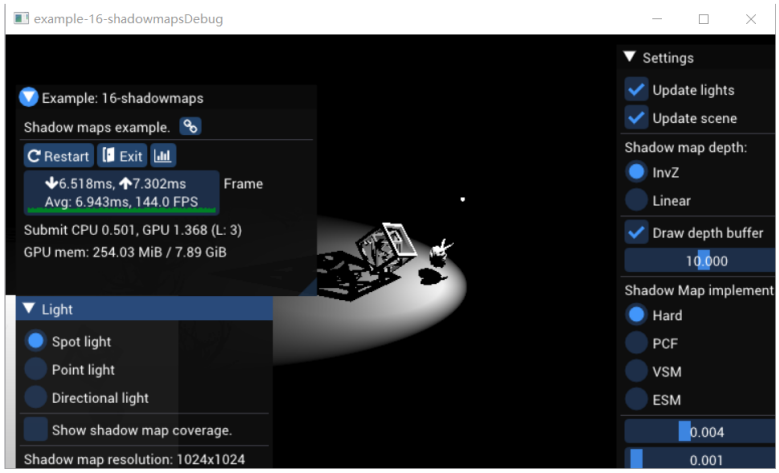
如果提示没有 –with-examples 这个选项，说明这个 bgfx 版本比较老，可以去官网下个比较新的版本。正确生成以后的结果：



打开 bgfx/.build/projects/vs2019 目录，打开 bgfx.sln 工程：



随便找一个代码，然后生成。如果生成成功，即可运行：



bgfx 有一些支持的工具，比如 shaderc，可以将 shader 编译到不同的平台下，还是像之前一样，进入 bgfx，然后输入命令：

```
..\bx\tools\bin\windows\genie --with-tools vs2019
```

即可编译生成这些工具。

二 制作新工程

生成工程需要使用 GENie，我研究了大半天才终于找到了生成工程的相关方法。在 bgfx/scripts 目录下，有一个文件叫 genie.lua 该文件定义了要生成的工程，比如：

```
if _OPTIONS["with-profiler"] then
......
end
if _OPTIONS["with-examples"]
or _OPTIONS["with-combined-examples"] then
......
end
```

当输入的参数中有 with-examples 时，就把下面的内容生成工程：

```
exampleProject(_OPTIONS["with-combined-examples"]
                , "00-helloworld"
                , "01-cubes"
                , "02-metaballs"
        ......)
```

这些都是在 bgfx/examples 文件夹下的目录，我们在后面添加上我们的工程目录。然后将我们的工程放在 bgfx/examples 文件夹下即可。

genie.lua 定义了一些生成工程的准则，比如把.c 和.h 文件作为代码文件添加到工程里之类的：

```
files {
    path.join(BGFX_DIR, "examples", name, "**.c"),
    path.join(BGFX_DIR, "examples", name, "**.cpp"),
    path.join(BGFX_DIR, "examples", name, "**.h"),
}
```

之后再在 cmd 里执行一次下面的命令即可：

```
..\bx\tools\bin\windows\genie --with-examples vs2019
```

三　初步了解 bgfx

当前的 bgfx 版本 (1.118.8367) 中，所有的自己实现的工程都首先继承 entry::AppI，比如：

```
class ExampleHelloWorld : public entry::AppI{
public:
ExampleHelloWorld(const char* _name, const char* _description, const char*
    _url) : entry::AppI(_name, _description, _url){}
}
```

_name 表示这个例子的名称，_description 表示对这个例子的描述，_url 表示网站链接。在 ENTRY_IMPLEMENT_MA
函数中会传入相关参数：

```
ENTRY_IMPLEMENT_MAIN(
    ExampleHelloWorld
    , "00-helloworld"
    , "Initialization and debug text."
    , "https://bkaradzic.github.io/bgfx/examples.html#helloworld"
    );
```

然后需要覆盖下面的几个函数：

```
// 负责管理初始化
void init(int32_t _argc, const char* const* _argv, uint32_t _width, uint32_t
    _height) override;
// 负责关闭后释放资源等
virtual int shutdown() override;
// 负责更新每一帧内容
bool update() override;
```

3 1　init 函数

我们看一下 ExampleHelloWorld 项目中的 init() 函数，该函数中有几个需要解释一下的地方。m_debug 表示要调试的类型，bgfx 中支持好几种类型，可以在 [2] 中找到。

bgfx::Init 结构可以参考 [2] 中的描述，该结构中有不少默认的设定方式，我们可以直接拿来使用，而不必追究细节。

3 2　shutdown 函数

该函数处理关闭以后的资源释放等，由于 ExampleHelloWorld 项目没有申请什么资源，所以这里仅仅是做一些基本处理：

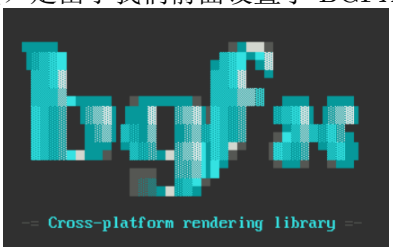
```
// 关闭imgui
imguiDestroy();
// 关闭bgfx
bgfx::shutdown();
```

3 3　update 函数

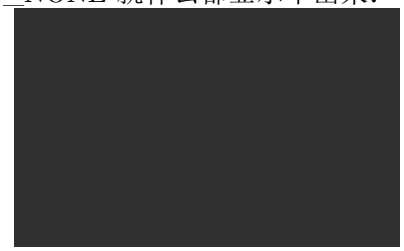
update 函数中比较难懂的是 dbgTextImage() 函数（dbg 是 debug 的意思），这里解释一下。

s_logo[4000] 是一副图像，每行有 160 个元素，一共 25 列。不过由于有些区域属于空白区，无需显示，所以并不用显示上来（只显示每行前 40 个字符以及前 12 行）。这里 s_logo[4000] 中每两个元素表示一个像素值，这里的取值是一个编码，而非 RGB 格式，因此一行有 80 个像素编码。

这里显示的格式（比如像素格式）是由于我们前面设置了 BGFX_DEBUG_TEXT，所以是这样：



如果设置的是 BGFX_DEBUG_NONE 就什么都显示不出来：



比如 ExampleDenoise 就不需要任何 debug，而是直接显示渲染结果，因此就用的 BGFX_DEBUG_NONE。

大家可以把所有的例子都自己演示演示，后面我会根据结合 LearnOpenGL[3] 来讲解这些例子。

参考文献

[1] https://github.com/bkaradzic/bgfx

[2] https://bkaradzic.github.io/bgfx/bgfx.html#resources

[3] https://learnopengl.com/