

----- SPIR-V & New shader system -----

》》》这次 Cherno 做了很多提交，所以我的笔记可能篇幅较长，但我会仔细记录。  
请认真浏览。

》》》 basic architecture layout of this episode (本集基本构架)  
(前庭仅供个人参考，并无侵犯版权的想法，若违反版权条款，并非本人意愿)

个人在学习过程中觉得值得查阅的几个文档：

游戏开发者大会文档 (关于 SPIR-V 与 渲染接口 OpenGL/Vulkan 、 GLSL/HLSL 之间的关系, SPIR-V 的工具及其执行流程)	<a href="https://www.neilhenning.dev/wp-content/uploads/2015/03/AnIntroductionToSPIR-V.pdf">https://www.neilhenning.dev/wp-content/uploads/2015/03/AnIntroductionToSPIR-V.pdf</a>
俄勒冈州立大学演示文档 ( SPIR-V 与 GLSL 之间的关系, SPIR-V 的实际使用方法: Win10 )	<a href="https://web.engr.oregonstate.edu/~mjb/cs557/Handouts/VulkanGLSL1pp.pdf">https://web.engr.oregonstate.edu/~mjb/cs557/Handouts/VulkanGLSL1pp.pdf</a>
Vulkan 官方 Github Readme 文档 ( GLSL 与 SPIR-V 之间的映射关系, 以及可以在线使用的编辑器, 非常好用 )	<a href="https://github.com/KhronosGroup/Vulkan-Guide/blob/main/chapters/mapping_data_to_shaders.adoc">https://github.com/KhronosGroup/Vulkan-Guide/blob/main/chapters/mapping_data_to_shaders.adoc</a>
	在线文档示例 ( <a href="https://godbolt.org/z/oMys8a78T">https://godbolt.org/z/oMys8a78T</a> )
大原Khronos开发者大会 (SPIR-V 语言的规范, 及其意义)	<a href="https://www.lunarg.com/wp-content/uploads/2023/05/SPIRV-Osaka-MAY2023.pdf">https://www.lunarg.com/wp-content/uploads/2023/05/SPIRV-Osaka-MAY2023.pdf</a>

前 33 分钟，基本上讲述以下几点：

<p>1.着色器将会支持 OpenGL 和 Vulkan ，故着色器中做了更改（涉及到 OpenGL 和 Vulkan 在着色器语法上的不同：比如 Uniform 的使用）</p> <p>2.为了避免性能浪费，并高效的使用数据/统一变量，将采用 UniformBuffer 这种高级 GLSL。</p> <p>(参考文献1-来自 LearnOpenGL 教程：<a href="https://learnopengl-cn.github.io/04%20Advanced%20OpenGL/08%20Advanced%20GLSL/">https://learnopengl-cn.github.io/04%20Advanced%20OpenGL/08%20Advanced%20GLSL/</a> )</p> <p>(参考文献2-来自 Vulkan 教程：<a href="https://vulkan-tutorial.com/Uniform_buffers/Descriptor_layout_and_buffer#page-Uniform-buffer">https://vulkan-tutorial.com/Uniform_buffers/Descriptor_layout_and_buffer#page-Uniform-buffer</a> )</p> <p>建议阅读全文，这样理解更加深刻。</p>	<p>• Uniform buffer</p> <h3>使用Uniform缓冲</h3> <p>我们已经讨论了如何在着色器中定义Uniform块，并设定它们的内存布局了，但我们还没有讨论如何使用它们。</p> <p>首先，我们需要调用 <code>glGenBuffers</code>，创建一个Uniform缓冲对象。一旦我们有了一个缓冲对象，我们需要将它绑定到 <code>GL_UNIFORM_BUFFER</code> 目标，并调用 <code>glBufferData</code>，分配足够的内存。</p> <pre>unsigned int uboExampleBlock; glGenBuffers(1, &amp;uboExampleBlock); glBindBuffer(GL_UNIFORM_BUFFER, uboExampleBlock); glBufferData(GL_UNIFORM_BUFFER, 152, NULL, GL_STATIC_DRAW); // 分配152字节的内存 glBindBuffer(GL_UNIFORM_BUFFER, 0);</pre> <p>现在，每当我们需要对缓冲更新或插入数据，我们都会绑定到 <code>uboExampleBlock</code>，并使用 <code>glBufferSubData</code> 来更新它的内存。我们只需要更新这个Uniform缓冲一次，所有使用这个缓冲的着色器就都使用的是更新后的数据了。但是，如何才能让OpenGL知道哪个Uniform缓冲对应的是哪个Uniform块呢？</p> <p>在OpenGL上下文中，定义了一些绑定点(Binding Point)，我们可以将一个Uniform缓冲中链接至它。在创建Uniform缓冲之后，我们将它绑定到其中一个绑定点上，并将着色器中的Uniform块绑定到相同的绑定点，把它们连接到一起。下面的这个图示例了这个：</p> <p>• Uniform buffer.</p> <h3>Uniform buffer 均匀缓冲</h3> <p>In the next chapter we'll specify the buffer that contains the UBO data for the shader, but we need to create this buffer first. We're going to copy new data to the uniform buffer every frame, so it doesn't really make any sense to have a staging buffer. It would just add extra overhead in this case and likely degrade performance instead of improving it.</p> <p>在下一章中，我们将指定包含着色器 UBO 数据的缓冲。但我们需要首先创建该缓冲。我们将每帧复制数据到该缓冲。因此，我们不需要 staging 缓冲，因为它在这种情况下只会增加额外的开销，并可能会降低性能而不是提高性能。</p> <p>We should have multiple buffers, because multiple frames may be in flight at the same time and we don't want to update the buffer in preparation of the next frame while a previous one is still reading from it! Thus, we need to have as many uniform buffers as we have frames in flight, and write to a uniform buffer that is not currently being read by the GPU</p> <p>我们应该有多个缓冲。因为多个帧可能同时在飞行，我们不想在上一帧仍在读取时更新缓冲以准备下一帧！因此，我们需要拥有与飞行中的帧一样多的统一缓冲，并写入 GPU 当前未读取的统一缓冲。</p> <p>To that end, add new class members for <code>uniformBuffers</code>, and <code>uniformBuffersMemory</code>:</p> <p>为此，为 <code>uniformBuffers</code> 和 <code>uniformBuffersMemory</code> 添加新的类成员：</p> <pre>VkBuffer IndexBuffer; VkDeviceMemory IndexBufferMemory;  std::vector&lt;VkBuffer&gt; uniformBuffers; std::vector&lt;VkDeviceMemory&gt; uniformBuffersMemory; std::vector&lt;uint&gt; uniformBuffersMapped;</pre> <p>Similarly, create a new function <code>createUniformBuffers</code> that is called after <code>createIndexBuffer</code> and allocates the buffers:</p> <p>类似地，创建一个新函数 <code>createUniformBuffers</code>，该函数在 <code>createIndexBuffer</code> 之后调用并分配缓冲：</p> <pre>void InitVulkan() {     ...     createVertexBuffer();     createIndexBuffer();     createUniformBuffers(); }</pre>
<p>3.OpenGL 和 Vulkan 在着色器语言上的使用规范，还有不同之处。</p> <p>参考文献：OpenGL教程 ( <a href="https://learnopengl-cn.github.io/02%20Lighting/03%20Materials/">https://learnopengl-cn.github.io/02%20Lighting/03%20Materials/</a> )</p> <p>参考文献：俄勒冈州立大学演示文件《GLSL For Vulkan》 ( <a href="https://eecs.oregonstate.edu/~mjb/cs557/Handouts/VulkanGLSL1pp.pdf">https://eecs.oregonstate.edu/~mjb/cs557/Handouts/VulkanGLSL1pp.pdf</a> )</p> <p>附录： 参考文献：Github 中文 Readme ( <a href="https://github.com/zenny-chen/GLSL-for-Vulkan">https://github.com/zenny-chen/GLSL-for-Vulkan</a> )</p>	<p>• GLSG 中的结构体示例：</p> <pre>#version 330 core struct Material {     vec3 ambient;     vec3 diffuse;     vec3 specular;     float shininess; };  uniform Material material;</pre> <p>在片段着色器中，我们创建一个结构体(struct)来保存物体的材质属性。我们也可以把它们存储为独立的uniform值。但是作为一个结构体来保存会更好一些。我们首先定义结构体的布局(layout)，然后简单地以刚创建的结构体作为类型声明一个uniform变量。</p> <p>• 如果查看 Vulkan API 在编写着色器时使用 GLSL 的语法规则，可以查看 Github 仓库 (中文：<a href="https://github.com/zenny-chen/GLSL-for-Vulkan">https://github.com/zenny-chen/GLSL-for-Vulkan</a>) 或查看 Vulkan 的官方入门指南 ( <a href="http://vulkan-tutorial.com/Introduction">http://vulkan-tutorial.com/Introduction</a> )</p>

参考文献: Vulkan 教程官网 ( <https://vulkan-tutorial.com/Introduction> )

或前往 Vulkan! 教程门户网站 ( <https://vulkan-tutorial.com/Introduction/> )

•不同之处:

### How Vulkan GLSL Differs from OpenGL GLSL

4

Detecting that a GLSL Shader is being used with Vulkan/SPIR-V:

- In the compiler, there is an automatic `#define VULKAN 100`


Vulkan Vertex and Instance Indices:

```
gl_VertexIndex
gl_InstanceIndex
```

- Both are 0-based

gl\_FragColor:

- In OpenGL, `gl_FragColor` broadcasts to all color attachments
- In Vulkan, it just broadcasts to color attachment location `0`
- Best idea: don't use it at all – explicitly declare out variables to have specific location numbers



oeb – December 17, 2020

### How Vulkan GLSL Differs from OpenGL GLSL

5

Shader combinations of separate texture data and samplers:

```
uniform sampler s;
uniform texture2D t;
vec4 rgba = texture( sampler2D( t, s ), vST );
```

Descriptor Sets:

```
layout( set=0, binding=0 ) ... ;
```

Push Constants:

```
layout( push_constant ) ... ;
```

Specialization Constants:


```
layout( constant_id = 3 ) const int N = 5;
```

- Only for scalars, but a vector's components can be constructed from specialization constants

Specialization Constants for Compute Shaders:

```
layout( local_size_x_id = 0, local_size_y_id = 16 );
```

- This sets `gl_WorkGroupSize.x` and `gl_WorkGroupSize.y`  
`gl_WorkGroupSize.z` is set as a constant



oeb – December 17, 2020

4.SPIR-V的使用思路，使用逻辑。

参考文献: SPIR-V 官网 ( [https://www.khronos.org/api/index\\_2017/spir](https://www.khronos.org/api/index_2017/spir) )

参考文献: Vulkan 教程 ( [https://vulkan-tutorial.com/Drawing\\_a\\_triangle/Graphics\\_pipeline\\_basics/Shader\\_modules](https://vulkan-tutorial.com/Drawing_a_triangle/Graphics_pipeline_basics/Shader_modules) )

参考文献: Vulkan 指南 ( [https://docs.vulkan.org/guide/latest/what\\_is\\_spirv.html](https://docs.vulkan.org/guide/latest/what_is_spirv.html) )

参考文献: 俄勒冈州立大学演示文件 ( <https://web.engr.oregonstate.edu/~mjb/cs557/Handouts/VulkanGLSL1pp.pdf> )

参考文献: 2016 年 3 月 - 游戏开发者大会 ( <https://www.neilheming.dev/wp-content/uploads/2015/03/AnIntroductionToSPIR-V.pdf> )

附件: 关于 SPIR-V 也可以参考 SPIR-V 的 github 仓库: ( <https://github.com/KhronosGroup/SPIRV-Guide> )

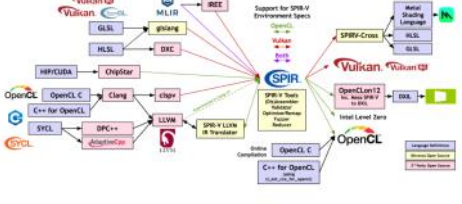
• SPIR-V 的生态系统:

### SPIR-V Language Ecosystem

#### SPIR-V 语言生态系统

The SPIR-V ecosystem includes a rich variety of language front-ends (compilers), development tools and run-times (consumers).

SPIR-V 生态系统包括丰富的语言前端(编译器)、开发工具和运行时(消费者)。



• SPIR-V 的概念:

Unlike earlier APIs, shader code in Vulkan has to be specified in a bytecode format as opposed to human-readable syntax like **GLSL** and **HLSL**. This bytecode format is called **SPIR-V** and is designed to be used with both Vulkan and OpenCL (both Khronos APIs). It is a format that can be used to write graphics and compute shaders, but we will focus on shaders used in Vulkan's graphics pipelines in this tutorial.

与早期的 API 不同, Vulkan 中的着色器代码必须以字节码格式指定,而不是像 **GLSL** 和 **HLSL** 这样的人类可读语法。这种字节码格式称为 **SPIR-V**, 现在与 Vulkan 和 OpenCL (均为 Khronos API) 一起使用。它是一种可用于编写图形和计算着色器的格式。但在本教程中我们将重点关注 Vulkan 图形管道中使用的着色器。

• SPIR-V 管线:

### Vulkan Guide / Logistics Overview / What is SPIR-V

#### What is SPIR-V 什么是 SPIR-V

**NOTE**

Please read the [SPIRV Guide](#) for more in detail information about SPIR-V

请阅读[SPIRV 指南](#)。了解有关 SPIR-V 的更详细的信息。

**SPIR-V** is a binary intermediate representation for graphical-shader stages and compute kernels. With Vulkan, an application can still write their shaders in a high-level shading language such as GLSL or HLSL, but a SPIR-V binary is needed when using `vkCreateShaderModule`. Khronos has a very nice [white paper](#) about SPIR-V and its advantages, and a high-level description of the representation. There are also two great Khronos presentations from Vulkan DevDay 2016 [here](#) and [here](#) (video of both).

**SPIR-V** 是面向着色器阶段和计算内核的二进制中间表示。使用 Vulkan, 应用程序仍然可以使用高级着色语言(例如 GLSL 或 HLSL)编写着色器。但使用 `vkCreateShaderModule` 时需要 SPIR-V 二进制文件。Khronos 有一些关于 SPIR-V 及其优点的非常好的[白皮书](#), 以及对表示的清晰描述。[这里](#)和[这里](#)还有 2016 年 Vulkan DevDay 的演讲视频和 Khronos 演示 [\(两者的视频\)](#)。



	<p>从 OpenGL 4.5 开始，OpenGL 也支持通过 SPIR-V 加载编译好的着色器二进制文件。流程与 Vulkan 类似，只不过 OpenGL 在内部做了更多的高层封装。</p> <p>加载过程：</p> <p>示例：</p> <pre>GLuint program = glCreateProgram();  // 加载 SPIR-V 二进制文件 GLuint shader = glCreateShader(GL_VERTEX_SHADER); glShaderBinary(1, &amp;shader, GL_SHADER_BINARY_FORMAT_SPIR_V, spirvData, spirvDataSize); glSpecializeShader(shader, "main", 0, nullptr, nullptr);  // 绑定和链接程序 glAttachShader(program, shader); glLinkProgram(program);</pre> <p>-----</p> <p>3.2 在 Vulkan 中使用 SPIR-V</p> <p>加载过程：</p> <p>创建一个 VkShaderModule 对象，该对象包含 SPIR-V 二进制代码。</p> <p>使用 SPIR-V 二进制代码来创建 Vulkan 着色器管线（例如，创建顶点着色器和片段着色器的管线）。</p> <p>示例：Vulkan 使用 SPIR-V</p> <pre>// 加载 SPIR-V 文件（假设你已经将 shader.spv 文件加载为二进制数据） VkShaderModuleCreateInfo createInfo = {}; createInfo.sType = VK_STRUCTURE_TYPE_SHADER_MODULE_CREATE_INFO; createInfo.codeSize = shaderData.size(); createInfo.pCode = reinterpret_cast&lt;const uint32_t*&gt;(shaderData.data());  // 创建着色器模块 VkShaderModule shaderModule; VkResult result = vkCreateShaderModule(device, &amp;createInfo, nullptr, &amp;shaderModule);  // 使用这个 shaderModule 来创建图形管线</pre>
4. 执行着色器程序	<p>在 OpenGL 中，SPIR-V 着色器程序被链接到程序对象中，并通过调用 <code>glUseProgram</code> 来激活该程序，之后通过绘制调用来执行。</p> <p>在 Vulkan 中，着色器被绑定到渲染管线或计算管线中，随后可以通过绘制命令（例如 <code>vkCmdDraw</code>）或计算命令（例如 <code>vkCmdDispatch</code>）来执行。</p>

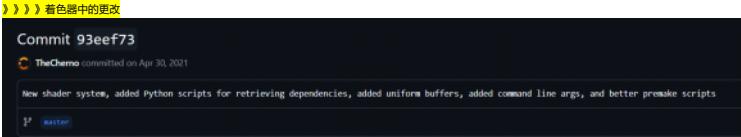
》》》上述涉及语言的纵向对比图

GLSL	<pre>#version 330 core  in vec3 fragColor; // 从顶点着色器传递过来的颜色 out vec4 FragColor; // 输出颜色到屏幕  void main() {     FragColor = vec4(fragColor, 1.0); // 输出最终颜色 }</pre>
<p>SPIR-V</p> <p>SPIR-V 本身的核心是一个二进制格式，然而为了便于开发和调试，SPIR-V 也可以以类似汇编语言的文本形式表达，这种形式通常称为 SPIR-V Assembly。</p> <p>它是 SPIR-V 的一种可读性较好的文本表示方式，开发者可以通过这种形式来编写、调试和优化 SPIR-V 代码，然后再将其转换为二进制格式以供图形 API 使用。</p> <p>实际上，SPIR-V Assembly 代码最终还是会通过工具（如 spirv-as）转化为二进制格式，供 Vulkan 或 OpenGL 使用。</p>	<p>SPIR-V</p> <pre>#302 2307 0000 0100 0100 0000 0e00 0000 0000 0000 1100 0200 0100 0000 0000 0000 0100 0000 474c 534c 2e73 7464 2e34 3530 0000 0000 0e00 0300 0000 0000 0100 0000 0f00 0000 0400 0000 0400 0000 6d61 695e 0000 0000 0900 0000 1000 0300 0400 0000 0700 0000 0300 0300 0200 0000 8c00 0000 0500 0400 0400 0000 6d61 695e 0000 0000 0500 0000 0900 0000 675c 5f46 7261 6743 6ffc 6f72 0000 0000 1300 0200 0200 0000 2100 0300 0300 0000 0200 0000 1600 0300 0600 0000 2000 0000 1700 0400 0700 0000 0600 0000 0400 0000 2000 0400 0800 0000 0300 0000 0700 0000 3b00 0400 0800 0000 0900 0000 0300 0000 2b00 0400 0800 0000 0a00 0000 cdc c3e 2b00 0400 0600 0000 0b00 0000 cdc c43f 2b00 0400 0600 0000 0c00 0000 0000 803f 2c00 0700 0700 0000 0d00 0000 6a00 0000 0a00 0000 0000 0000 0e00 0000 3600 0500 0200 0000 0400 0000 0000 0000 0300 0000 f800 0200 0500 0000 1e00 0100 0900 0000 0000 0000 f000 0100 3800 0100</pre> <p>SPIR-V Assembly</p> <pre>; SPIR-V ; Version: 1.0 ; Generator: Khronos GLSLang Reference Front End; 1 ; Bound: 14 ; Schema: 0  OpCapability Shader %1 = OpExtInstImport "GLSL.std.450" OpMemoryModel Logical GLSL450 OpEntryPoint Fragment %4 "main" %9 OpExecutionMode %4 OriginUpperLeft OpSource GLSL 450 OpName %4 "main" OpName %9 "out_colour" OpDecorate %9 Location 0  %2 = OpTypeVoid %3 = OpTypeFunction %2 %6 = OpTypeFloat 32 %7 = OpTypeVector %6 4 %8 = OpTypePointer Output %7 %9 = OpVariable %8 Output %10 = OpConstant %6 0.4 %11 = OpConstant %6 0.8 %12 = OpConstant %6 1 %13 = OpConstantComposite %7 %10 %10 %11 %12 %4 = OpFunction %2 None %3 %5 = OpLabel     OpStore %9 %13     OpReturn OpFunctionEnd</pre>
OpenGL	<pre>GLuint shaderProgram = glCreateProgram(); glAttachShader(shaderProgram, vertexShader); glAttachShader(shaderProgram, fragmentShader); glLinkProgram(shaderProgram); glUseProgram(shaderProgram);  // 主要渲染循环 while (!glfwWindowShouldClose(window)) {     glClear(GL_COLOR_BUFFER_BIT);     glUseProgram(shaderProgram);</pre>

Vulkan

```
VkInstance instance;
VkApplicationInfo appInfo = {};
appInfo.sType = VK_STRUCTURE_TYPE_APPLICATION_INFO;
appInfo.pApplicationName = "Vulkan 示例";
appInfo.applicationVersion = VK_MAKE_VERSION(1, 0, 0);
appInfo.pEngineName = "No Engine";
appInfo.engineVersion = VK_MAKE_VERSION(1, 0, 0);
appInfo.apiVersion = VK_API_VERSION_1_0;

VkInstanceCreateInfo createInfo = {};
createInfo.sType = VK_STRUCTURE_TYPE_INSTANCE_CREATE_INFO;
createInfo.pApplicationInfo = &appInfo;
```



以下是详情解释:

1 premake脚本更改  
(and better premake scripts)

2 + -- Hazel Dependencies  
3 +  
4 + VULKAN\_SDK = os.getenv("VULKAN\_SDK")

```
15 + IncludeDir["shaders"] = "${wks.location}/Hazel/vendor/shaders/include"
16 + IncludeDir["SPIRV_Cross"] = "${wks.location}/Hazel/vendor/SPIRV-Cross"
17 + IncludeDir["VulkanSDK"] = "${VULKAN_SDK}/include"
18 +
19 + LibraryDir = {}
20 +
21 + LibraryDir["VulkanSDK"] = "${VULKAN_SDK}/lib"
22 + LibraryDir["VulkanSDK_Debug"] = "${wks.location}/Hazel/vendor/VulkanSDK/lib"
23 +
24 + Library = {}
25 + Library["Vulkan"] = "${LibraryDir.VulkanSDK}/vulkan-1.lib"
26 + Library["VulkanGLES"] = "${LibraryDir.VulkanSDK}/VkLayer_utility.lib"
27 +
28 + Library["ShaderC_Debug"] = "${LibraryDir.VulkanSDK_Debug}/shader_shared.lib"
29 + Library["SPIRV_Cross_Debug"] = "${LibraryDir.VulkanSDK_Debug}/spirv-cross-core.lib"
30 + Library["SPIRV_Cross_GLES_Debug"] = "${LibraryDir.VulkanSDK_Debug}/spirv-cross-gles.lib"
31 + Library["SPIRV_Tools_Debug"] = "${LibraryDir.VulkanSDK_Debug}/SPIRV_Tools.lib"
32 +
33 + Library["ShaderC_Release"] = "${LibraryDir.VulkanSDK}/shader_shared.lib"
34 + Library["SPIRV_Cross_Release"] = "${LibraryDir.VulkanSDK}/spirv-cross-core.lib"
35 + Library["SPIRV_Cross_GLES_Release"] = "${LibraryDir.VulkanSDK}/spirv-cross-gles.lib"
```

premake5.lua

```
1 --
2 -- @ - 3.4 - 4.5 88
3 include "${vendor/premake/premake_customization/solution_items.lua}"
4 + include "Dependencies.lua"
5
6 workspace "Hazel"
7
8 architecture "x86_64"
9
10 @ - 23,17 +24,6 @ workspace "Hazel"
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
```

```
73 +         "H[library.ShaderC_Debug]",
74 +         "H[library.SPIRV_Cross_Debug]",
75 +         "H[library.SPIRV_Cross_GLSL_Debug]"
76 +     }
77 +
78 +     filter "configurations:Release"
79 +     defines "H_RELEASE"
80 +     runtime "Release"
81 +     optimize "on"
82 +
83 +     links
84 +     [
85 +         "H[library.ShaderC_Release]",
86 +         "H[library.SPIRV_Cross_Release]",
87 +         "H[library.SPIRV_Cross_GLSL_Release]"
88 +     ]
89 +
90 +     filter "configurations:Dist"
91 +     defines "H_DIST"
92 +     runtime "Release"
93 +     optimize "on"
94 +
95 +     links
96 +     [
97 +         "H[library.ShaderC_Release]",
98 +         "H[library.SPIRV_Cross_Release]",
99 +         "H[library.SPIRV_Cross_GLSL_Release]"
100 +     ]
```

2.py脚本  
(Python scripts for retrieving dependencies)

```
1 + import subprocess
2 + import pkg_resources
3 +
4 + def install(package):
5 +     print(f"Installing {package} module...")
6 +     subprocess.check_call([python, "-m", "pip", "install", package])
7 +
8 + def validate_package(package):
9 +     required = { package }
10 +     installed = {pkg.key for pkg in pkg_resources.working_set}
11 +     missing = required - installed
12 +
13 +     if missing:
14 +         install(package)
15 +
16 + def validate_packages():
17 +     validate_package('requests')
18 +     validate_package('fake-useragent')
```

1. 确保在执行过程中 requests 和 fake-useragent 这两个模块已经安装。如果没有安装，它会自动使用 pip 安装它们。

```
1 + import os
2 + import subprocess
3 + import CheckPython
4 +
5 + # Make sure everything we need is installed
6 + CheckPython.ValidatePackages()
7 +
8 + import Vulkan
9 +
10 + # Change from Scripts directory to root
11 + os.chdir('../')
12 +
13 + if (not Vulkan.CheckVulkanSDK()):
14 +     print("Vulkan SDK not installed.")
15 +
16 + if (not Vulkan.CheckVulkanSDKDebugLibs()):
17 +     print("Vulkan SDK debug libs not found.")
18 +
19 + print("Running premake...")
20 + subprocess.call(["cmd", "premake5.exe", "vs2019"])
```

- 1. 确保所需的 Python 包已经安装。
- 2. 检查 Vulkan SDK 是否安装，并确保 Vulkan SDK 的调试库存在。
- 3. 改变当前工作目录到项目根目录。
- 4. 使用 premake 工具生成 Visual Studio 2019 项目的构建文件。

```
1 + import requests
2 + import sys
3 + import time
4 +
5 + from fake_useragent import UserAgent
6 +
7 + def DownloadFile(url, filepath):
8 +     with open(filepath, 'wb') as f:
9 +         ua = UserAgent()
10 +         headers = {'User-Agent': ua.stream}
11 +         response = requests.get(url, headers=headers, stream=True)
12 +         total = response.headers.get('content-length')
13 +
14 +         if total is None:
15 +             f.write(response.content)
16 +         else:
17 +             downloaded = 0
18 +             total = int(total)
19 +             startTime = time.time()
20 +             for data in response.iter_content(chunk_size=max(int(total/1000), 1024*1024)):
21 +                 downloaded += len(data)
22 +                 f.write(data)
23 +                 done = int(10*downloaded/total)
24 +                 percentage = (downloaded / total) * 100
25 +                 elapsedTime = time.time() - startTime
26 +                 avgPerSecond = (downloaded / 1024) / elapsedTime
27 +                 avgDownloadString = '{:2f} MB/s'.format(avgPerSecond)
28 +                 if (avgPerSecond > 1024):
29 +                     avgPerSecond = avgPerSecond / 1024
30 +                 avgDownloadString = '{:2f} MB/s'.format(avgPerSecond)
```

DownloadFile(url, filepath) 函数的作用是从指定 URL 下载文件，并显示实时的下载进度（包括下载进度条和速度）。  
YesOrNo() 函数用于与用户进行交互，获取用户的确认输入，返回布尔值表示“是”或“否”。

```

1 // 0.0 - 1.00 00
2 #include <...>
3 #include <...>
4 #include <...>
5 #include <...>
6 #include <...>
7 #include <...>
8 #include <...>
9 #include <...>
10 #include <...>
11 #include <...>
12 #include <...>
13 #include <...>
14 #include <...>
15 #include <...>
16 #include <...>
17 #include <...>
18 #include <...>
19 #include <...>
20 #include <...>
21 #include <...>
22 #include <...>
23 #include <...>
24 #include <...>
25 #include <...>

```

用于检查和安装 Vulkan SDK

InstallVulkanSDK(): 下载并运行 Vulkan SDK 安装程序。

InstallVulkanPrompt(): 提示用户是否安装 Vulkan SDK。

CheckVulkanSDK(): 检查 Vulkan SDK 是否安装并且版本是否正确。

CheckVulkanSDKDebugLibs(): 检查 Vulkan SDK 的调试库是否存在，如果缺失则下载并解压缩。

3 Application 中的 ApplicationCommandLineArgs  
( added command line args)

```

1 // 13.7 - 13.8 00 namespace Hazel {
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20

```

```

1 // 13.7 - 13.8 00 namespace Hazel {
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100

```

```

1 // 13.7 - 13.8 00 namespace Hazel {
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100

```

```

1 // 13.7 - 13.8 00 namespace Hazel {
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100

```

```

1  #include "HazelnetApp.hpp"
2
3  namespace Hazel {
4
5      class Hazelnet : public Application
6      {
7      public:
8
9          Hazelnet()
10         {
11             Application("Hazelnet")
12             Hazelnet(ApplicationCommandLineArgs args)
13             Application("Hazelnet", args)
14         {
15             PushLayer(new EditorLayer());
16         }
17     };
18
19     namespace Hazel {
20
21         Application* CreateApplication()
22         Application* CreateApplication(ApplicationCommandLineArgs args)
23         {
24             return new Hazelnet();
25             return new Hazelnet(args);
26         }
27     }
28 }

```

#### 4 Uniform Buffer 的定义以及使用, 包括着色器更新 (added uniform buffers)

```

1 // Hack1/src/Hack1/Hack1UniformBuffer.h
2
3 #pragma once
4
5 #include "Hack1/Core/Hack1.h"
6
7 namespace Hack1 {
8
9     public:
10         virtual ~UniformBuffer() {}
11         virtual void SetData(const void* data, uint32_t size, uint32_t offset = 0);
12
13         static RefUniformBuffer Create(uint32_t size, uint32_t binding);
14     };
15 }

```

```

1 // src/Phase/Renderer/UniformBuffer.cpp
2
3 #include "Render.h"
4 #include "UniformBuffer.h"
5 #include "Scene/Renderer/Renderer.h"
6 #include "Utils/OpenGL/OpenGLUniformBuffer.h"
7
8 namespace Hazel {
9
10     RefUniformBuffer UniformBuffer::Create(uint32_t size, uint32_t binding)
11     {
12         auto Renderer = GetAPI();
13
14         case Renderer.API::None: return Hazel_CBO_Assert(false, "RendererAPI: There is no
15         case Renderer.API::OpenGL: return CreateOpenGLUniformBuffer(size, binding);
16     }
17
18     Hazel_CBO_Assert(false, "Unknown RendererAPI!");
19     return nullptr;
20 }
21
22 }

```

```

1 // Hazel/vec/Platform/OpenGL/OpenGLUniformBuffer.h
2
3 #pragma once
4
5 #include "Hazel/Renderer/UniformBuffer.h"
6
7 namespace Hazel {
8
9     class OpenGLUniformBuffer : public UniformBuffer
10     {
11     public:
12         OpenGLUniformBuffer(uint32_t size, uint32_t binding);
13         virtual ~OpenGLUniformBuffer();
14
15         virtual void setData(const void* data, uint32_t size, uint32_t offset = 0) override;
16
17         uint32_t m_RendererID = 0;
18     };
19 }

```

```

1 // Main.cpp Platform/OpenGL/OpenGLVertexBuffer.cpp
2
3 #include <stdio.h>
4 #include "Numpy.h"
5 #include "OpenGLVertexBuffer.h"
6
7 #include <glad/glad.h>
8
9 namespace Hazel {
10
11     OpenGLVertexBuffer::OpenGLVertexBuffer(uint32_t size, uint32_t binding)
12     {
13         GLVertexBuffer* v = new Hazel::VertexBuffer();
14         glBindBuffer(GL_ARRAY_BUFFER, binding);
15         glBufferData(GL_ARRAY_BUFFER, size, nullptr, GL_DYNAMIC_DRAW); // TODO: improve
16         glBindBuffer(GL_UNIFORM_BUFFER, binding, m_RendererID);
17     }
18
19     OpenGLVertexBuffer::~OpenGLVertexBuffer()
20     {
21         glBindBuffer(GL_UNIFORM_BUFFER, m_RendererID);
22     }
23
24     void OpenGLVertexBuffer::SetData(const void* data, uint32_t size, uint32_t offset)
25     {
26         glBindBuffer(GL_UNIFORM_BUFFER, m_RendererID, offset, size, data);
27     }
28 }

```

```

1  #include "Hazel/Renderer/Renderer2D.cpp"
2
3  +
4  3
5  #include "Hazel/Renderer/VertexArray.h"
6  #include "Hazel/Renderer/Shader.h"
7  #include "Hazel/Renderer/VertexBuffer.h"
8  #include "Hazel/Renderer/RenderCommand.h"
9  #include <glm/gtc/matrix_transform.hpp>
10 #include <glm/gtx/type_ptr.hpp>
11
12 namespace Hazel {
13
14 +
15 @@ -41,6 +45,13 @@ namespace Hazel {
16
17 41 45 glm::vec4 QuadVerticesPosition[4];
18
19 42 46 Renderer2D::Statistics Stats;
20
21 43 47
22 44 48 +
23 45 49 struct CameraData
24 46 50 +
25 47 51 +
26 48 52 glm::mat4 ViewProjection;
27 49 53 +
28 50 54 +
29 51 55 +
30 52 56 Ref<RenderTarget> CameraRenderTarget;
31
32 53 57 };
33
34 54 58
35 55 59
36 56 60
37 57 61
38 58 62
39 59 63
40 60 64
41 61 65
42 62 66
43 63 67
44 64 68
45 65 69
46 66 70
47 67 71
48 68 72
49 69 73
50 70 74
51 71 75
52 72 76
53 73 77
54 74 78
55 75 79
56 76 80
57 77 81
58 78 82
59 79 83
60 80 84
61 81 85
62 82 86
63 83 87
64 84 88
65 85 89
66 86 90
67 87 91
68 88 92
69 89 93
70 90 94
71 91 95
72 92 96
73 93 97
74 94 98
75 95 99
76 96 100
77 97 101
78 98 102
79 99 103
80 100 104
81 101 105
82 102 106
83 103 107
84 104 108
85 105 109
86 106 110
87 107 111
88 108 112
89 109 113
90 110 114
91 111 115
92 112 116
93 113 117
94 114 118
95 115 119
96 116 120
97 117 121
98 118 122
99 119 123
100 120 124
101 121 125
102 122 126
103 123 127
104 124 128
105 125 129
106 126 130
107 127 131
108 128 132
109 129 133
110 130 134
111 131 135
112 132 136
113 133 137
114 134 138
115 135 139
116 136 140
117 137 141
118 138 142
119 139 143
120 140 144
121 141 145
122 142 146
123 143 147
124 144 148
125 145 149
126 146 150
127 147 151
128 148 152
129 149 153
130 150 154
131 151 155
132 152 156
133 153 157
134 154 158
135 155 159
136 156 160
137 157 161
138 158 162
139 159 163
140 160 164
141 161 165
142 162 166
143 163 167
144 164 168
145 165 169
146 166 170
147 167 171
148 168 172
149 169 173
150 170 174
151 171 175
152 172 176
153 173 177
154 174 178
155 175 179
156 176 180
157 177 181
158 178 182
159 179 183
160 180 184
161 181 185
162 182 186
163 183 187
164 184 188
165 185 189
166 186 190
167 187 191
168 188 192
169 189 193
170 190 194
171 191 195
172 192 196
173 193 197
174 194 198
175 195 199
176 196 200
177 197 201
178 198 202
179 199 203
180 200 204
181 201 205
182 202 206
183 203 207
184 204 208
185 205 209
186 206 210
187 207 211
188 208 212
189 209 213
190 210 214
191 211 215
192 212 216
193 213 217
194 214 218
195 215 219
196 216 220
197 217 221
198 218 222
199 219 223
200 220 224
201 221 225
202 222 226
203 223 227
204 224 228
205 225 229
206 226 230
207 227 231
208 228 232
209 229 233
210 230 234
211 231 235
212 232 236
213 233 237
214 234 238
215 235 239
216 236 240
217 237 241
218 238 242
219 239 243
220 240 244
221 241 245
222 242 246
223 243 247
224 244 248
225 245 249
226 246 250
227 247 251
228 248 252
229 249 253
230 250 254
231 251 255
232 252 256
233 253 257
234 254 258
235 255 259
236 256 260
237 257 261
238 258 262
239 259 263
240 260 264
241 261 265
242 262 266
243 263 267
244 264 268
245 265 269
246 266 270
247 267 271
248 268 272
249 269 273
250 270 274
251 271 275
252 272 276
253 273 277
254 274 278
255 275 279
256 276 280
257 277 281
258 278 282
259 279 283
260 280 284
261 281 285
262 282 286
263 283 287
264 284 288
265 285 289
266 286 290
267 287 291
268 288 292
269 289 293
270 290 294
271 291 295
272 292 296
273 293 297
274 294 298
275 295 299
276 296 300
277 297 301
278 298 302
279 299 303
280 300 304
281 301 305
282 302 306
283 303 307
284 304 308
285 305 309
286 306 310
287 307 311
288 308 312
289 309 313
290 310 314
291 311 315
292 312 316
293 313 317
294 314 318
295 315 319
296 316 320
297 317 321
298 318 322
299 319 323
300 320 324
301 321 325
302 322 326
303 323 327
304 324 328
305 325 329
306 326 330
307 327 331
308 328 332
309 329 333
310 330 334
311 331 335
312 332 336
313 333 337
314 334 338
315 335 339
316 336 340
317 337 341
318 338 342
319 339 343
320 340 344
321 341 345
322 342 346
323 343 347
324 344 348
325 345 349
326 346 350
327 347 351
328 348 352
329 349 353
330 350 354
331 351 355
332 352 356
333 353 357
334 354 358
335 355 359
336 356 360
337 357 361
338 358 362
339 359 363
340 360 364
341 361 365
342 362 366
343 363 367
344 364 368
345 365 369
346 366 370
347 367 371
348 368 372
349 369 373
350 370 374
351 371 375
352 372 376
353 373 377
354 374 378
355 375 379
356 376 380
357 377 381
358 378 382
359 379 383
360 380
```



5 着色器系统更新:  
(New shader system)

```
1 // src/MyClass/Timer.h
2 #pragma once
3 #include <chrono>
4 #include <string>
5 namespace Pencil {
6
7     class Timer
8     {
9     public:
10         Timer()
11         {
12             Reset();
13         }
14
15         void Timer::Reset()
16         {
17             m_Start = std::chrono::high_resolution_clock::now();
18         }
19
20         float Timer::Elapsed()
21         {
22             return std::chrono::duration_cast<std::chrono::microseconds>
23                 (std::chrono::high_resolution_clock::now() - m_Start).count() / 1000.0f;
24         }
25
26         float Timer::ElapsedMillis()
27         {
28             return Elapsed() * 1000.0f;
29         }
30
31     private:
32         std::chrono::time_point<std::chrono::high_resolution_clock> m_Start;
```

```

7 // ResultType(PlatformSpecific)OpenGLShader_Type.cpp
8
9 #if GL_VERSION_4_0 || GL_EXT_gpu_shader4
10
11 #include <glm/glm.hpp>
12 #include <glm/ext/matrix_float3x2.hpp>
13 #include <glm/ext/vector_float3.hpp>
14 #include <glm/ext/quaternion_float.hpp>
15 #include <glm/ext/vector_float3_spherical_coordinates.hpp>
16 #include <glm/ext/vector_float3_radial_coordinates.hpp>
17 #include <glm/ext/vector_float3_linear_space.hpp>
18 #include "Header/Common.h"
19
20 namespace Hazel {
21
22     static GLenum ShaderTypeFromString(const std::string& type)
23     {
24         if (type == "vertex")
25             return GL_VERTEX_SHADER;
26         else if (type == "fragment" || type == "pixel")
27             return GL_FRAGMENT_SHADER;
28     }
29
30     namespace Utils {
31
32         static GLenum ShaderTypeFromString(const std::string& type)
33         {
34             if (type == "vertex")
35                 return GL_VERTEX_SHADER;
36             else if (type == "fragment" || type == "pixel")
37                 return GL_FRAGMENT_SHADER;
38             else
39                 return GL_NONE;
40         }
41
42         void CreateShader(GLuint shaderID, const std::string& type, const std::string& source)
43         {
44             switch (type)
45             {
46                 case "vertex":
47                     glCompileShader(shaderID);
48                     break;
49                 case "fragment":
50                     glCompileShader(shaderID);
51                     break;
52                 default:
53                     throw std::runtime_error("Invalid shader type");
54             }
55         }
56
57         void LinkProgram(GLuint programID)
58         {
59             glLinkProgram(programID);
60         }
61
62         void ValidateProgram(GLuint programID)
63         {
64             glValidateProgram(programID);
65         }
66
67         void DeleteShader(GLuint shaderID)
68         {
69             glDeleteShader(shaderID);
70         }
71
72         void DeleteProgram(GLuint programID)
73         {
74             glDeleteProgram(programID);
75         }
76
77         void AttachShader(GLuint programID, GLuint shaderID)
78         {
79             glAttachShader(programID, shaderID);
80         }
81
82         void DetachShader(GLuint programID, GLuint shaderID)
83         {
84             glDetachShader(programID, shaderID);
85         }
86
87         void UseProgram(GLuint programID)
88         {
89             glUseProgram(programID);
90         }
91
92         void UnuseProgram()
93         {
94             glUseProgram(0);
95         }
96
97         void GetAttribLocation(GLuint programID, const std::string& name, GLint& location)
98         {
99             glGetAttribLocation(programID, name.c_str(), &location);
100         }
101
102         void GetUniformLocation(GLuint programID, const std::string& name, GLint& location)
103         {
104             glGetUniformLocation(programID, name.c_str(), &location);
105         }
106
107         void SetUniformVec3(GLint location, const glm::vec3& value)
108         {
109             glUniform3f(location, value.x, value.y, value.z);
110         }
111
112         void SetUniformMat3x3(GLint location, const glm::mat3x3& value)
113         {
114             glUniformMatrix3fv(location, 1, GL_FALSE, value.data());
115         }
116
117         void SetUniformMat4x4(GLint location, const glm::mat4x4& value)
118         {
119             glUniformMatrix4fv(location, 1, GL_FALSE, value.data());
120         }
121
122         void SetUniformVec4(GLint location, const glm::vec4& value)
123         {
124             glUniform4f(location, value.x, value.y, value.z, value.w);
125         }
126
127         void SetUniformMat2x2(GLint location, const glm::mat2x2& value)
128         {
129             glUniformMatrix2fv(location, 1, GL_FALSE, value.data());
130         }
131
132         void SetUniformMat3x4(GLint location, const glm::mat3x4& value)
133         {
134             glUniformMatrix3x4fv(location, 1, GL_FALSE, value.data());
135         }
136
137         void SetUniformMat4x3(GLint location, const glm::mat4x3& value)
138         {
139             glUniformMatrix4x3fv(location, 1, GL_FALSE, value.data());
140         }
141
142         void SetUniformMat2x3(GLint location, const glm::mat2x3& value)
143         {
144             glUniformMatrix2x3fv(location, 1, GL_FALSE, value.data());
145         }
146
147         void SetUniformMat3x2(GLint location, const glm::mat3x2& value)
148         {
149             glUniformMatrix3x2fv(location, 1, GL_FALSE, value.data());
150         }
151
152         void SetUniformMat4x2(GLint location, const glm::mat4x2& value)
153         {
154             glUniformMatrix4x2fv(location, 1, GL_FALSE, value.data());
155         }
156
157         void SetUniformMat2x4(GLint location, const glm::mat2x4& value)
158         {
159             glUniformMatrix2x4fv(location, 1, GL_FALSE, value.data());
160         }
161
162         void SetUniformMat3x1(GLint location, const glm::mat3x1& value)
163         {
164             glUniformMatrix3x1fv(location, 1, GL_FALSE, value.data());
165         }
166
167         void SetUniformMat4x1(GLint location, const glm::mat4x1& value)
168         {
169             glUniformMatrix4x1fv(location, 1, GL_FALSE, value.data());
170         }
171
172         void SetUniformMat1x3(GLint location, const glm::mat1x3& value)
173         {
174             glUniformMatrix1x3fv(location, 1, GL_FALSE, value.data());
175         }
176
177         void SetUniformMat1x4(GLint location, const glm::mat1x4& value)
178         {
179             glUniformMatrix1x4fv(location, 1, GL_FALSE, value.data());
180         }
181
182         void SetUniformMat1x2(GLint location, const glm::mat1x2& value)
183         {
184             glUniformMatrix1x2fv(location, 1, GL_FALSE, value.data());
185         }
186
187         void SetUniformMat2x1(GLint location, const glm::mat2x1& value)
188         {
189             glUniformMatrix2x1fv(location, 1, GL_FALSE, value.data());
190         }
191
192         void SetUniformMat3x1(GLint location, const glm::mat3x1& value)
193         {
194             glUniformMatrix3x1fv(location, 1, GL_FALSE, value.data());
195         }
196
197         void SetUniformMat4x1(GLint location, const glm::mat4x1& value)
198         {
199             glUniformMatrix4x1fv(location, 1, GL_FALSE, value.data());
200         }
201
202         void SetUniformMat1x3(GLint location, const glm::mat1x3& value)
203         {
204             glUniformMatrix1x3fv(location, 1, GL_FALSE, value.data());
205         }
206
207         void SetUniformMat1x4(GLint location, const glm::mat1x4& value)
208         {
209             glUniformMatrix1x4fv(location, 1, GL_FALSE, value.data());
210         }
211
212         void SetUniformMat1x2(GLint location, const glm::mat1x2& value)
213         {
214             glUniformMatrix1x2fv(location, 1, GL_FALSE, value.data());
215         }
216
217         void SetUniformMat2x1(GLint location, const glm::mat2x1& value)
218         {
219             glUniformMatrix2x1fv(location, 1, GL_FALSE, value.data());
220         }
221
222         void SetUniformMat3x1(GLint location, const glm::mat3x1& value)
223         {
224             glUniformMatrix3x1fv(location, 1, GL_FALSE, value.data());
225         }
226
227         void SetUniformMat4x1(GLint location, const glm::mat4x1& value)
228         {
229             glUniformMatrix4x1fv(location, 1, GL_FALSE, value.data());
230         }
231
232         void SetUniformMat1x3(GLint location, const glm::mat1x3& value)
233         {
234             glUniformMatrix1x3fv(location, 1, GL_FALSE, value.data());
235         }
236
237         void SetUniformMat1x4(GLint location, const glm::mat1x4& value)
238         {
239             glUniformMatrix1x4fv(location, 1, GL_FALSE, value.data());
240         }
241
242         void SetUniformMat1x2(GLint location, const glm::mat1x2& value)
243         {
244             glUniformMatrix1x2fv(location, 1, GL_FALSE, value.data());
245         }
246
247         void SetUniformMat2x1(GLint location, const glm::mat2x1& value)
248         {
249             glUniformMatrix2x1fv(location, 1, GL_FALSE, value.data());
250         }
251
252         void SetUniformMat3x1(GLint location, const glm::mat3x1& value)
253         {
254             glUniformMatrix3x1fv(location, 1, GL_FALSE, value.data());
255         }
256
257         void SetUniformMat4x1(GLint location, const glm::mat4x1& value)
258         {
259             glUniformMatrix4x1fv(location, 1, GL_FALSE, value.data());
260         }
261
262         void SetUniformMat1x3(GLint location, const glm::mat1x3& value)
263         {
264             glUniformMatrix1x3fv(location, 1, GL_FALSE, value.data());
265         }
266
267         void SetUniformMat1x4(GLint location, const glm::mat1x4& value)
268         {
269             glUniformMatrix1x4fv(location, 1, GL_FALSE, value.data());
270         }
271
272         void SetUniformMat1x2(GLint location, const glm::mat1x2& value)
273         {
274             glUniformMatrix1x2fv(location, 1, GL_FALSE, value.data());
275         }
276
277         void SetUniformMat2x1(GLint location, const glm::mat2x1& value)
278         {
279             glUniformMatrix2x1fv(location, 1, GL_FALSE, value.data());
280         }
281
282         void SetUniformMat3x1(GLint location, const glm::mat3x1& value)
283         {
284             glUniformMatrix3x1fv(location, 1, GL_FALSE, value.data());
285         }
286
287         void SetUniformMat4x1(GLint location, const glm::mat4x1& value)
288         {
289             glUniformMatrix4x1fv(location, 1, GL_FALSE, value.data());
290         }
291
292         void SetUniformMat1x3(GLint location, const glm::mat1x3& value)
293         {
294             glUniformMatrix1x3fv(location, 1, GL_FALSE, value.data());
295         }
296
297         void SetUniformMat1x4(GLint location, const glm::mat1x4& value)
298         {
299             glUniformMatrix1x4fv(location, 1, GL_FALSE, value.data());
300         }
301
302         void SetUniformMat1x2(GLint location, const glm::mat1x2& value)
303         {
304             glUniformMatrix1x2fv(location, 1, GL_FALSE, value.data());
305         }
306
307         void SetUniformMat2x1(GLint location, const glm::mat2x1& value)
308         {
309             glUniformMatrix2x1fv(location, 1, GL_FALSE, value.data());
310         }
311
312         void SetUniformMat3x1(GLint location, const glm::mat3x1& value)
313         {
314             glUniformMatrix3x1fv(location, 1, GL_FALSE, value.data());
315         }
316
317         void SetUniformMat4x1(GLint location, const glm::mat4x1& value)
318         {
319             glUniformMatrix4x1fv(location, 1, GL_FALSE, value.data());
320         }
321
322         void SetUniformMat1x3(GLint location, const glm::mat1x3& value)
323         {
324             glUniformMatrix1x3fv(location, 1, GL_FALSE, value.data());
325         }
326
327         void SetUniformMat1x4(GLint location, const glm::mat1x4& value)
328         {
329             glUniformMatrix1x4fv(location, 1, GL_FALSE, value.data());
330         }
331
332         void SetUniformMat1x2(GLint location, const glm::mat1x2& value)
333         {
334             glUniformMatrix1x2fv(location, 1, GL_FALSE, value.data());
335         }
336
337         void SetUniformMat2x1(GLint location, const glm::mat2x1& value)
338         {
339             glUniformMatrix2x1fv(location, 1, GL_FALSE, value.data());
340         }
341
342         void SetUniformMat3x1(GLint location, const glm::mat3x1& value)
343         {
344             glUniformMatrix3x1fv(location, 1, GL_FALSE, value.data());
345         }
346
347         void SetUniformMat4x1(GLint location, const glm::mat4x1& value)
348         {
349             glUniformMatrix4x1fv(location, 1, GL_FALSE, value.data());
350         }
351
352         void SetUniformMat1x3(GLint location, const glm::mat1x3& value)
353         {

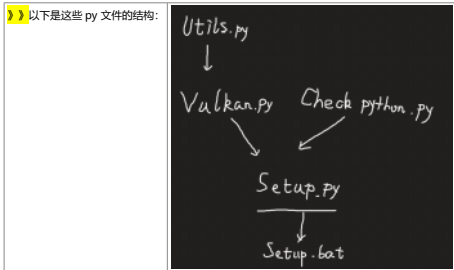
```

## 6 平台工具的更新（打开或保存文件）

```

> H:\src\Platform\Windows\WindowsPlatformWin.cpp
18 19
19 20     m_responseHeader {
21 21
22 22         std::optional<string> ffileDialogs{SendFile(const char* filter)
23 23         + std::string ffileDialogs{SendFile(const char* filter)
24 24
25 25         OPENFILENAME ofn;
26 26         CHM sFile[50] = { 0 };
27 27
28 28         m_responseHeader {
29 29
30 30         if (GetOpenFileName(&ofn) == TRUE)
31 31             return ofn.lpstrFile;
32 32
33 33         return std::nullopt;
34 34
35 35         return std::string{};
36 36
37 37         std::optional<string> ffileDialogs{SendFile(const char* filter)
38 38         + std::string ffileDialogs{SendFile(const char* filter)
39 39
40 40         OPENFILENAME ofn;
41 41         CHM sFile[50] = { 0 };

```

[illegible]

运行脚本时，请关闭代理。

**问题二**

创建好 VulkanSDK 文件夹之后，重新运行 Setup.py，脚本运行之后开始尝试运行 Vulkan installer:

这可能是 Vulkan.py 中存放的 VulkanSDK 下载地址不适合 64 位系统，我将其更新为 2023 年的某一版本。

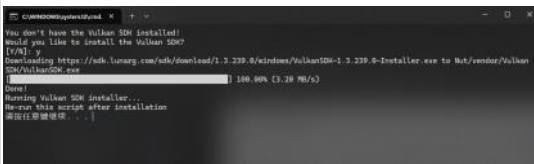
附录: 如果你想进入官网查看适合你系统的 SDK, 以下是网址 -> (<https://vulkan.lunarg.com/sdk/home>)



当前我只更新了 SDK Installer 的安装地址, 但是我还没有更新随后的 debug lib.zip, 这是下一个问题会出现的地方, 现在先不讨论。

```
48 vulkanSDKDebugLibsURL = "https://files.lunarg.com/SDK-1.2.179.0/vulkansdk-1.2.179.0-debug-lib.zip"
49 OutputDirectory = "H:/vendor/VulkanSDK"
50 TempZipFile = ("OutputDirectory/VulkanSDK.zip")
51
52
53 def CheckVulkanSDKDebugLibs():
54     shadercdlib = Path(f"({OutputDirectory})/lib/shadercd.lib")
55     if (not shadercdlib.exists()):
56         print("No Vulkan SDK debug libs found, (checked {shadercdlib})")
57         print("Downloading", VulkanSDKDebugLibsURL)
58         with urlopen(VulkanSDKDebugLibsURL) as zipresp:
59             with ZipFile(BytesIO(zipresp.read())) as zfile:
60                 zfile.extractall(OutputDirectory)
61         print("Vulkan SDK debug libs located at {OutputDirectory}")
62     return True
```

我们先重新运行一遍, 使用更新之后的 SDK install,

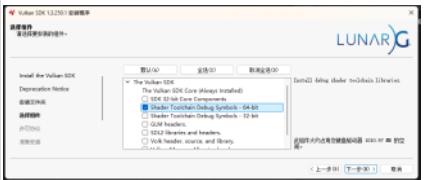


于是运行后出现这样的窗口:

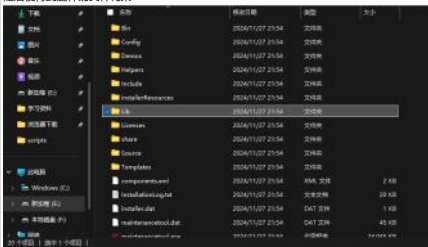


安装 vulkan SDK

我目前没有选择任何拓展, 但在安装过程中, 我不是很确定这个拓展和 DebugLibs 有没有什么直接关系, 就先标注一下, (毕竟这将会占用我1G空间 bushi)



随后便得到这样的文件结构:



### 问题三

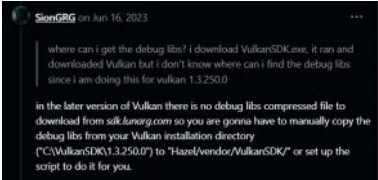
我们发现 Cherno 另外下载了一个 Debuglib.zip, 并对其进行了一些处理。

但是在1.2.198.1版本之后, lunarg 公司不再支持 debuglibs 的单独下载。现在 SDK 中的调试库通常随着 Vulkan 库一起分发, 不再单独打包成一个 zip 文件。

所以现在, 这些文件通常直接包含在 Vulkan SDK 的核心目录下, 特别是在 lib 目录中

我们也可以从评论中窥见这一更改。 (@SionGRG)

```
48 vulkanSDKDebugLibsURL = "https://files.lunarg.com/SDK-1.2.179.0/vulkansdk-1.2.179.0-debug-lib.zip"
49 OutputDirectory = "H:/vendor/VulkanSDK"
50 TempZipFile = ("OutputDirectory/VulkanSDK.zip")
51
52
53 def CheckVulkanSDKDebugLibs():
54     shadercdlib = Path(f"({OutputDirectory})/lib/shadercd.lib")
55     if (not shadercdlib.exists()):
56         print("No Vulkan SDK debug libs found, (checked {shadercdlib})")
57         print("Downloading", VulkanSDKDebugLibsURL)
58         with urlopen(VulkanSDKDebugLibsURL) as zipresp:
59             with ZipFile(BytesIO(zipresp.read())) as zfile:
60                 zfile.extractall(OutputDirectory)
61         print("Vulkan SDK debug libs located at {OutputDirectory}")
62     return True
```



现在我们需要更改这个函数

```
vulkanSDKDebugLibsURL = "https://files.lunarg.com/SDK-1.2.179.0/vulkansdk-1.2.179.0-debug-lib.zip"
```

( CheckVulkanSDKDebugLibs ) 的逻辑

```
outputDirectory = "Nuts/vendor/VulkanSDK"
zipFile = ("outputDirectory)/VulkanSDK.zip"

def CheckVulkanSDKDebugLibs():
    shadercdlib = Path(("outputDirectory)/lib/shadercdlib")
    if (not shadercdlib.exists()):
        print("No Vulkan SDK debug libs found. (checked (shadercdlib))")
        print("Downloading VulkanSDKDebugLibs")
        with urlopen(VulkanSDKDebugLibURL) as zipresp:
            with zipfile.Zipfile(zipresp.read()) as zfile:
                zfile.extractall(outputDirectory)
        print("Vulkan SDK debug libs located at (outputDirectory)")
        return True
```

首先，我对这个 shadercdlib 的路径有点疑惑：因为我的确查找到了 shadercdlib 这个库，而不是shadercdlib。

```
>>> def CheckVulkanSDKDebugLibs():
54     shadercdlib = Path(("outputDirectory)/lib/shadercdlib")
55     if (not shadercdlib.exists()):
```



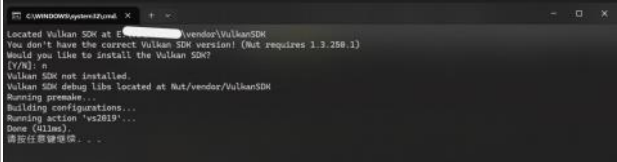
现在我开始更改，不过我发现原先的逻辑是：如果没有找到调试库，就在线去下载。  
但现在这些文件将会在安装 Vulkan SDK 时，同步安装在文件夹中，所以如果没有找到的话，一定是安装是出了什么问题。

我便做了以下更改：（仅仅是口头提醒一下~）

```
>>> VulkanSDKDebugLibURL = "https://files.lunarg.com/sdk/1.2.139.0/VulkanSDK-1.2.139.0-DebugLibs.zip"
outputDirectory = "Nuts/vendor/VulkanSDK"

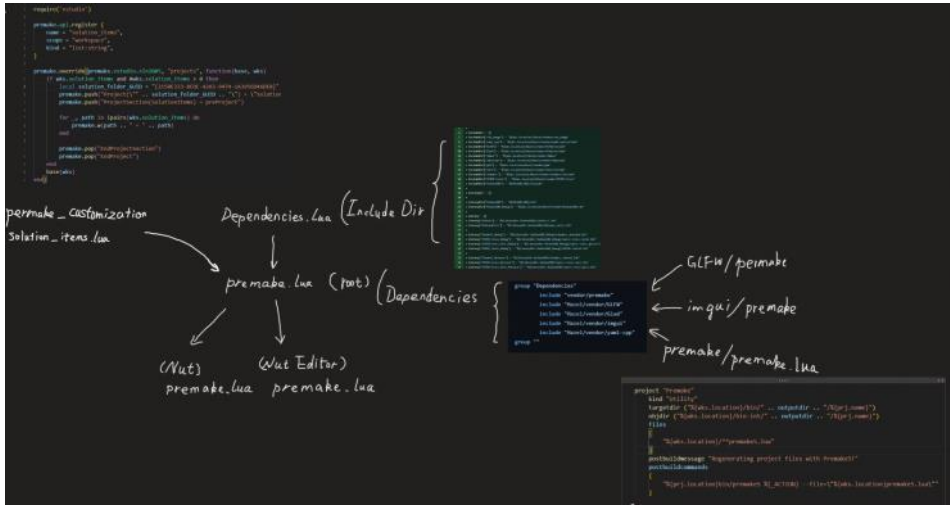
def CheckVulkanSDKDebugLibs():
    shadercdlib = Path(("outputDirectory)/lib/shadercdlib")
    if (not shadercdlib.exists()):
        print("No Vulkan SDK debug libs found. (checked (shadercdlib))")
        # print("Downloading VulkanSDKDebugLibs")
        # with urlopen(VulkanSDKDebugLibURL) as zipresp:
        #     with zipfile.Zipfile(zipresp.read()) as zfile:
        #         zfile.extractall(outputDirectory)
        # print("Please check you Vulkan SDK files. (checked https://vulkan.lunarg.com/sdk/home and find you version, Or reinstall you Vulkan SDK by installer)")
        # 提示用户检查 Vulkan SDK 文件
        raise RuntimeError("Vulkan SDK debug libs not found, process aborted.")
    print("Vulkan SDK debug libs located at (outputDirectory)")
    return True
```

随后我重新运行 Setup.bat，并拒绝再次安装 installer，便得到这样的结果：



我想应该是对的。

》》》现在我们已成功安装了 Vulkan，现在则需要更新 premake 文件内容。  
这是将要实现的 premake 文件构架图（以及细则）

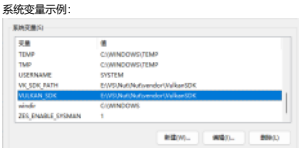
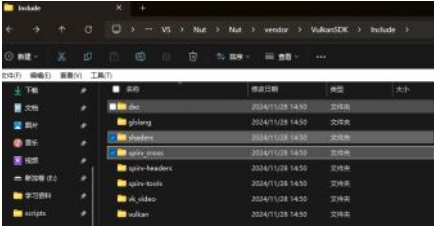


》》》接下来我先更新 Premake Dependencies.lua 文件（这里为预处理，实际操作步骤在后面）。

第一步，我们在项目的根目录下重新编写一个premake文件，这个文件主要用来索引vendor中的外部库（API）

```
1 --!- Not Dependent!
2
3 VULKAN_SDK = os.getenv("VULKAN_SDK")
4
5 IncludeDir = {}
6 IncludeDir["vulkan_image"] = "%(vks.location)/vulkan/vendor/vulkan_image"
7 IncludeDir["vulkan_core"] = "%(vks.location)/vulkan/vendor/vulkan_core/include"
8 IncludeDir["vulkan"] = "%(vks.location)/vulkan/vendor/vulkan/include"
9 IncludeDir["glad"] = "%(vks.location)/vulkan/vendor/glad/include"
10 IncludeDir["imgui"] = "%(vks.location)/vulkan/vendor/imgui"
11 IncludeDir["imgui_impl_glfw"] = "%(vks.location)/vulkan/vendor/imgui_impl_glfw"
12 IncludeDir["glsl"] = "%(vks.location)/vulkan/vendor/glsl"
13 IncludeDir["entt"] = "%(vks.location)/vulkan/vendor/entt/include"
14 IncludeDir["shaderc"] = "%(vks.location)/vulkan/vendor/shaderc/include"
15 IncludeDir["SPIRV_cross"] = "%(vks.location)/vulkan/vendor/SPIRV_cross"
16 IncludeDir["vulkan_sdk"] = "%(vks.location)/vulkan_sdk/include"
17
18 LibraryDir = {}
19
20 LibraryDir["vulkan_sdk"] = "%(vks.location)/vulkan_sdk/lib"
21 LibraryDir["vulkan_sdk_debug"] = "%(vks.location)/vulkan_sdk/lib"
22
23 Library = {}
24
25 Library["vulkan"] = "%(librarydir.vulkan_sdk)/vulkan-1.lib"
26 Library["vulkan_imgui"] = "%(librarydir.vulkan_sdk)/vulkan_imgui.lib"
27
28 Library["shaderc_debug"] = "%(librarydir.vulkan_sdk_debug)/shaderc_shared.lib"
29 Library["SPIRV_cross_debug"] = "%(librarydir.vulkan_sdk_debug)/SPIRV_cross_core.lib"
30 Library["SPIRV_cross_gsl_debug"] = "%(librarydir.vulkan_sdk_debug)/SPIRV_cross_gsl.lib"
31 Library["SPIRV_tools_debug"] = "%(librarydir.vulkan_sdk_debug)/SPIRV_tools.lib"
32
33 Library["shader_release"] = "%(librarydir.vulkan_sdk)/shader_shared.lib"
34 Library["SPIRV_cross_release"] = "%(librarydir.vulkan_sdk)/SPIRV_cross_core.lib"
35 Library["SPIRV_cross_gsl_release"] = "%(librarydir.vulkan_sdk)/SPIRV_cross_gsl.lib"
```

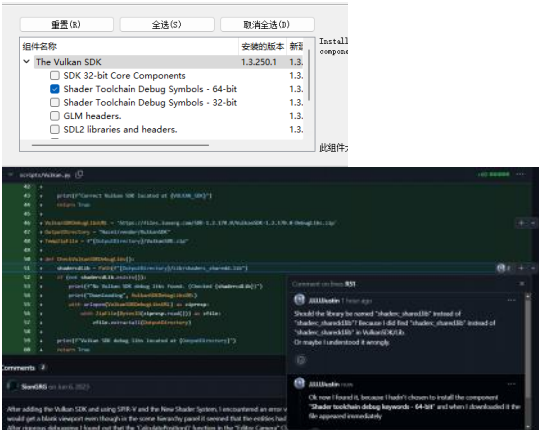
但我发现有些问题，比如shaderc和spirv\_cross的路径已经发生改变，参考1.3.250.1版本：这两个文件位于VulkanSDK/Include下



而且由于我没有下载某些组件，这使很多文件并不存在。（我将其标注出来）

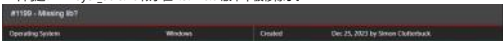
```
1 --!- Not Dependent!
2
3 VULKAN_SDK = os.getenv("VULKAN_SDK")
4
5 IncludeDir = {}
6 IncludeDir["vulkan_image"] = "%(vks.location)/vulkan/vendor/vulkan_image"
7 IncludeDir["vulkan_core"] = "%(vks.location)/vulkan/vendor/vulkan_core/include"
8 IncludeDir["vulkan"] = "%(vks.location)/vulkan/vendor/vulkan/include"
9 IncludeDir["glad"] = "%(vks.location)/vulkan/vendor/glad/include"
10 IncludeDir["imgui"] = "%(vks.location)/vulkan/vendor/imgui"
11 IncludeDir["imgui_impl_glfw"] = "%(vks.location)/vulkan/vendor/imgui_impl_glfw"
12 IncludeDir["glsl"] = "%(vks.location)/vulkan/vendor/glsl"
13 IncludeDir["entt"] = "%(vks.location)/vulkan/vendor/entt/include"
14 IncludeDir["shaderc"] = "%(vks.location)/vulkan/vendor/shaderc/include"
15 IncludeDir["SPIRV_cross"] = "%(vks.location)/vulkan/vendor/SPIRV_cross"
16 IncludeDir["vulkan_sdk"] = "%(vks.location)/vulkan_sdk/include"
17
18 LibraryDir = {}
19
20 LibraryDir["vulkan_sdk"] = "%(vks.location)/vulkan_sdk/lib"
21 LibraryDir["vulkan_sdk_debug"] = "%(vks.location)/vulkan_sdk/lib"
22
23 Library = {}
24
25 Library["vulkan"] = "%(librarydir.vulkan_sdk)/vulkan-1.lib"
26 Library["vulkan_imgui"] = "%(librarydir.vulkan_sdk)/vulkan_imgui.lib"
27
28 Library["shaderc_debug"] = "%(librarydir.vulkan_sdk_debug)/shaderc_shared.lib"
29 Library["SPIRV_cross_debug"] = "%(librarydir.vulkan_sdk_debug)/SPIRV_cross_core.lib"
30 Library["SPIRV_cross_gsl_debug"] = "%(librarydir.vulkan_sdk_debug)/SPIRV_cross_gsl.lib"
31 Library["SPIRV_tools_debug"] = "%(librarydir.vulkan_sdk_debug)/SPIRV_tools.lib"
32
33 Library["shader_release"] = "%(librarydir.vulkan_sdk)/shader_shared.lib"
34 Library["SPIRV_cross_release"] = "%(librarydir.vulkan_sdk)/SPIRV_cross_core.lib"
35 Library["SPIRV_cross_gsl_release"] = "%(librarydir.vulkan_sdk)/SPIRV_cross_gsl.lib"
```

于是我决定下载拓展(shader toolchain debug symbols)，这一步通过运行maintenancetool.exe文件实现：



```
1 --!- Not Dependent!
2
3 VULKAN_SDK = os.getenv("VULKAN_SDK")
4
5 IncludeDir = {}
6 IncludeDir["vulkan_image"] = "%(vks.location)/vulkan/vendor/vulkan_image"
7 IncludeDir["vulkan_core"] = "%(vks.location)/vulkan/vendor/vulkan_core/include"
8 IncludeDir["vulkan"] = "%(vks.location)/vulkan/vendor/vulkan/include"
9 IncludeDir["glad"] = "%(vks.location)/vulkan/vendor/glad/include"
10 IncludeDir["imgui"] = "%(vks.location)/vulkan/vendor/imgui"
11 IncludeDir["imgui_impl_glfw"] = "%(vks.location)/vulkan/vendor/imgui_impl_glfw"
12 IncludeDir["glsl"] = "%(vks.location)/vulkan/vendor/glsl"
13 IncludeDir["entt"] = "%(vks.location)/vulkan/vendor/entt/include"
14 IncludeDir["shaderc"] = "%(vks.location)/vulkan/vendor/shaderc/include"
15 IncludeDir["SPIRV_cross"] = "%(vks.location)/vulkan/vendor/SPIRV_cross"
16 IncludeDir["vulkan_sdk"] = "%(vks.location)/vulkan_sdk/include"
17
18 LibraryDir = {}
19
20 LibraryDir["vulkan_sdk"] = "%(vks.location)/vulkan_sdk/lib"
21 LibraryDir["vulkan_sdk_debug"] = "%(vks.location)/vulkan_sdk/lib"
22
23 Library = {}
24
25 Library["vulkan"] = "%(librarydir.vulkan_sdk)/vulkan-1.lib"
26 Library["vulkan_imgui"] = "%(librarydir.vulkan_sdk)/vulkan_imgui.lib"
27
28 Library["shaderc_debug"] = "%(librarydir.vulkan_sdk_debug)/shaderc_shared.lib"
29 Library["SPIRV_cross_debug"] = "%(librarydir.vulkan_sdk_debug)/SPIRV_cross_core.lib"
30 Library["SPIRV_cross_gsl_debug"] = "%(librarydir.vulkan_sdk_debug)/SPIRV_cross_gsl.lib"
31 Library["SPIRV_tools_debug"] = "%(librarydir.vulkan_sdk_debug)/SPIRV_tools.lib"
32
33 Library["shader_release"] = "%(librarydir.vulkan_sdk)/shader_shared.lib"
34 Library["SPIRV_cross_release"] = "%(librarydir.vulkan_sdk)/SPIRV_cross_core.lib"
35 Library["SPIRV_cross_gsl_release"] = "%(librarydir.vulkan_sdk)/SPIRV_cross_gsl.lib"
```

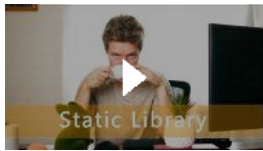
虽然下载了一个组件可以解决但部分问题，但是尽管在之后我下载了其余的所有组件，VulkanSDK/Include/Lib这个路径都不存在（但是Vulkan/Lib这个路径存在，且







(顺便一提，如果需要打开的话，还需要额外进行动态链接的配置操作，具体可以回看Cherno的视频： [Static Libraries and ZERO Warnings I Game Engine series](#) )



》》》接下来谈谈 vendor/premake 文件中我们新添的两个文件：premake5.lua 和 premake\_customization/solution\_items.lua 具体的 PullRequests 记载于 #301 ( <https://github.com/TheCherno/Haze/pull/301> )

Premake5.lua	定义一个工具类型的项目 Premake，并且在构建后通过 premake5 工具来重新生成或更新项目文件。  这个脚本的目的是 <b>生成或重新生成构建项目文件</b> （如 Visual Studio 工程文件、Makefile 等），使用的是 <b>premake5</b> 工具。它是一个自动化构建的过程，通常用于生成构建系统（如 Makefile 或 Visual Studio 工程文件）等。
solution_items.lua	这段代码的作用是为 Visual Studio 解决方案文件（.sln）添加一个新的部分，称为 <b>Solution Items</b> ，并将工作区中指定的文件（通过 solution_items 命令）添加到这个部分中。  解决方案项是指那些不是属于任何特定项目的文件，例如文档、配置文件等，通常用于存储一些和整个解决方案相关但不属于某个单独项目的文件。  这添加了对 Visual Studio 解决方案项（solution items）的支持。文档、配置文件、README 或其他相关文件可以被作为解决方案项添加到解决方案中。

## 》》》Application 中的 ApplicationCommandLineArgs (added command line args) 命令行参数

### 》》流程与定义的概述

首先，我们位于入口点的主函数中使用了 (argc, argv) 来获取命令行信息。并且将参数传入到 CreateApplication() 中，以便后续使用这些信息：

```
int main(int argc, char** argv)
{
    // 将其设置为 windows 平台上的 WinMain
    Nut::CreateApplication(ApplicationCommandLineArgs);

    Nut::Log::Info("Initialised log");
    Nut::Info("Goodbye World");

    Nut::Core::Warn("Command line args:");
    for (int i = 0; i < argc; i++) {
        Nut::Core::Trace("Argument (%d): %s", i, argv[i]);
    }

    Nut::Profile::BeginSession("Startup", "NutProfile-Startup.json");
    auto app = Nut::CreateApplication(ApplicationCommandLineArgs); // 这里我们传入的 argc 和 argv 就是这里传入的
    Nut::Profile::EndSession();


    Nut::Profile::BeginSession("Runtime", "NutProfile-Runtime.json");
    app->Run();
    Nut::Profile::EndSession();

    Nut::Profile::BeginSession("Shutdown", "NutProfile-Shutdown.json");
    delete app;
    Nut::Profile::EndSession();

    return 0;
}
```

在入口点使用的 `Nut::CreateApplication((argc, argv))`，实际上是在构造一个 `ApplicationCommandLineArgs` 类型的对象，并将 `argc` 和 `argv` 传递给它。

管线流程：

<pre>Application* CreateApplication(ApplicationCommandLineArgs args) {     return new NutEditor(args); }</pre>	这里是 CreateApplication() 的定义。  CreateApplication() 中使用了 NutEditor()
<pre>class NutEditor : public Application { public:     NutEditor(ApplicationCommandLineArgs args) : Application("Nut Editor", args) {}     PushLayer(new EditorLayer());      ~NutEditor() {} };</pre>	这里是 NutEditor() 的定义。  NutEditor() 是 Application() 的子类，故 NutEditor() 的构造函数会自动先使用父类 Application() 的构造函数，我们可以通过这个特性将 args 参数传给 Application() 的构造函数，并实现一些目的。
<pre>class Application { public:     Application(const std::string&amp; name = "Nut App", ApplicationCommandLineArgs args = ApplicationCommandLineArgs()) : m_Name(name), m_Args(args) {}     virtual ~Application() {}      void OnEvent(Event e); // 事件分发      void PushLayer(Layer* layer);     void PushOverlay(Layer* overlay);      inline Window* GetWindow() { return m_Window; }     inline Window* GetEditorLayer() { return m_EditorLayer; } // 返回下面这个指向 Window 的指针     inline static Application* Get() { return m_Instance; } // !!! 返回的是 s_Instance 这个指向 Application 的指针     inline Application* GetInstance() { return m_Instance; } // (为什么函数是引用传递？因为 Application 是一个单例)     inline ApplicationCommandLineArgs GetCommandLineArgs() const { return m_CommandLineArgs; }      auto sceneFilePath0 = commandLineArgs[0];      Nut::Core::Warn(sceneFilePath0); }</pre>	这是父类 Application() 构造函数的新定义。  同时我们新添了一个 GetCommandLineArgs() 的函数，用于获取私有变量 m_CommandLineArgs 中存放的数据。  

我们可以在运行时查看 argv 获取到的信息是什么。

## 》》》知识点

### 》》》关于 Argc, Argv

#### 1. argc 和 argv 的含义

定义：

在 C 和 C++ 程序中，argc 和 argv 是由编译器（如 GCC、Clang 或 Visual Studio）在程序启动时自动传递给程序的 main 函数的两个参数，用于传递命令行的输入参数。

- argc：是 argument count 的缩写，表示命令行参数的数量。它是一个整数，包含程序名和任何附加的命令行参数。
- argv：是 argument vector 的缩写，表示命令行参数的数组。它是一个字符指针数组，每个元素是一个指向命令行参数的字符串。

例如，当你运行一个程序 `/myapp input.txt --verbose` 时，argc 和 argv 的内容如下：

argc = 3，因为有三个参数（程序名、input.txt 和 --verbose）	argv[0] = "/myapp"，表示程序的路径。 argv[1] = "input.txt"，表示第一个参数（输入文件）。 argv[2] = "--verbose"，表示第二个参数（开启调试模式）。
---	---

<p>1. 内容传递。 (什么时候传递? 传递什么内容?)</p>	<p><code>argc</code> 和 <code>argv</code> 是由操作系统在启动程序时根据命令行输入自动传递的, 不需要手动获取。</p> <p>程序中 <code>argc</code> 和 <code>argv</code> 的值取决于你启动程序时后台输入到命令行中的命令或参数内容。在不同的操作系统上, 命令行参数的格式和解释规则可能会有所不同。</p> <p>比如:</p> <ul style="list-style-type: none"> <li>• 在 <b>Windows</b> 上, 命令行参数是由 <b>命令提示符 (cmd.exe)</b> 或 <b>PowerShell</b> 等工具传递给程序的。</li> <li>• 在 <b>Unix/Linux</b> 上, 命令行参数是由 <b>shell (如 Bash)</b> 传递给程序的。</li> </ul>
<p>2. 内容 (内容什么时候被确定? 是否可以被随时改变?)</p>	<p><code>argc</code> 和 <code>argv</code> 是实时的, 但它们是<b>程序启动时</b>由操作系统从命令行提取的参数, 并且在程序执行过程中保持不变。</p> <p>所以一旦程序开始执行, <code>argc</code> 和 <code>argv</code> 的值就固定了, 不能在程序运行过程中改变。</p>

C++ 标准库没有其他内建的类似 `argc` 和 `argv` 的机制。`argc` 和 `argv` 是 `main` 函数的参数，是 C++ 标准定义的，通常用于处理命令行参数。

例如，在当前情况下，我们可以在 EditorLayer.cpp 中实时的获取到命令行参数信息并将其打印在控制台上：

```
void EditorLayer::OnAttach()
{
    MUT_PROFILE_FUNCTION();

    m_Framebuffer = Framebuffer::Create({ 1280, 720, 1 }, Framebuffer::Format_RGBA);

    m_Texture = Texture2D::Create("assets/textures/Checkerboard.png");
    m_Repo1 = Texture2D::Create("assets/textures/emoji.png");

    m_ActiveScene = CreateRef(Scene)();

    CommandList& Iskray = Application::Get().GetCommandList(Iskray);
    if (!m_Iskray.Iskry.Count > 1)
    {
        auto SceneFilePath = CommandList& Iskray{};
        SceneSerializer serializer(m_ActiveScene);
        serializer.Serialize(sceneFilePath);

        auto sceneFilePathO = CommandList& Iskray{};

        MUT_OVER_WRITE(sceneFilePathO);
    }
}
```

```
void EditorLayer::OnInit()
{
    NUT_PROFILE_FUNCTION();

    m_Framebuffer = Framebuffer::Create(1280, 720, 1, Framebuffer::
        m_Texture = Texture2D::Create("assets/textures/Checkboard.png");
    m_Esmpi = Texture2D::Create("assets/textures/empty.png");
    m_ActiveScene = CreateRef(Scene0);

    m_CommandBuffer = Application::Get().GetCommandBuffer();
    if (m_CommandBuffer.Count() > 1)
    {
        NUT_WARN("Path %s Command Buffer %s",
            serializer.Serialize(m_ActiveScene),
            serializer.Serialize(m_CommandBuffer));
    }

    m_CommandBuffer.Push = m_CommandBuffer.Count();
    NUT_WARN("Path %s", m_CommandBuffer.Count());
}
```

```
void ButtonLayer::OnAttach()
{
    NUT_PROFILE_FUNCTION();

    m_Framebuffer = Framebuffer::Create((1280, 720, 1), Framebuffer::
    m_Texture = Texture2D::Create("assets/textures/Checkerboard.png");
    m_Kaoji = Texture2D::Create("assets/textures/emoj.png");
    m_ActiveScene = CreateRef(Scene00);

    m_CommandBuffer = Application::Get().GetCommandBuffer();
    if (m_CommandBuffer.Count() > 31)
    {
        auto sceneFilePath = CommandBuffer::
        SceneSerializer.Serialize(m_ActiveScene,
        serializer.Serialize(m_ActiveScene,
        serialiser.Serialize(m_ActiveScene,

    auto sceneFileId = CommandBuffer::

    m_NUT_CORE_NAME(sceneFilePath);
}
```

[illegible]

**实际使用时发生的情况：**

现在 Cherno 设置了命令行参数的新功能，但其实并不是想通过在某处修改命令内容，或者实时根据命令的变化进行一些操作。而是为了在命令行中运行指令时，开启引擎并进入页面的时候，能够自动预先加载一个场景，让我们查看效果以了解详情：

<b>(旧)</b>	<b>(新)</b>
一个黄褐色头发的男人，他打开了 cmd，想要运行 Nut-Editor 应用，于是他输入了一句指令：	但是在新功能的加持下，如果我们在该指令之后添加了一个来自场景的目录：





))) "/"= 运算符重载

Eg.  
"m\_CurrentDirectory /= path.filename();"

/= 运算符的重载

概念:  
在 C++17 的 std::filesystem::path 中, /= 运算符是被重载的, 用于拼接路径。其功能是将路径对象 path 中的部分与左侧的路径进行合并。

使用要求:  
m\_CurrentDirectory 是一个表示当前目录的路径, 通常是一个 std::filesystem::path 类型的对象。path.filename() 返回的是 path 对象中的文件名部分, 且其类型也是 std::filesystem::path。

示例说明:

假设

m_CurrentDirectory	C:\Projects\MyGame\Assets.
path	C:\Projects\MyGame\Assets\Models\Character.obj.
path.filename()	Character.obj.

那么, m\_CurrentDirectory /= path.filename(); 的结果会是 **m\_CurrentDirectory 等于 C:\Projects\MyGame\Assets\Character.obj**

))) ImGui::Columns(columnCount, 0, false);

ImGui::Columns()

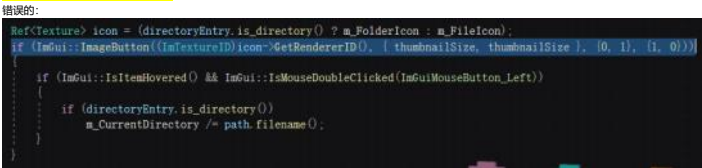
原型:  
void ImGui::Columns(int columns\_count = 1, const char\* id = NULL, bool border = true);

参数解释:

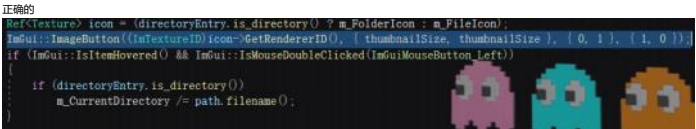
columns_count (类型: int, 默认值: 1)	功能: 指定列的数量。默认值是 1, 表示只有一列。如果你想创建多个列, 可以设置为大于 1 的数字。
id (类型: const char*, 默认值: NULL)	功能: 这是一个可选的字符串, 用来指定一个唯一的 ID。 如果多个列使用相同的 ID, ImGui 会为它们创建一个统一的状态。这个 ID 在 ImGui 的内部用于区分不同的列布局, 但不需要区分, 可以传入 NULL 或忽略它。
border (类型: bool, 默认值: true)	功能: 指定是否显示列之间的边框。如果为 true, 列之间会有一个分隔线。如果为 false, 则没有边框, 列之间没有分隔线。

示例:	示例: ImGui::Columns(3) 表示创建 3 列布局。
	示例: ImGui::Columns(3, "MyColumns"), 通过指定 ID, 可以在后续的操作中区分不同的列布局。
	示例: ImGui::Columns(3, NULL, false) 表示创建 3 列, 并且不显示列间的边框。

))) 一段错误代码诱发的思考:



如果将 ImGui::ImageButton() 放在条件判断中, 会导致优先判断按钮是否被单击, 随后才会判断使用者是否在指定区域双击图标, 这会导致鼠标双击的逻辑不能正常触发。



))) ImGui::TextWrapped()

概念:  
ImGui::TextWrapped() 是一个用于在 ImGui 中显示文本的函数, 主要特点是当文本内容超出当前窗口或控件的宽度时, 会自动换行显示。  
这个特性适用于显示多行文本, 因为文本宽度是动态的, 可以适应父容器的大小, 这避免了手动计算的麻烦。

函数原型:

void ImGui::TextWrapped(const char\* fmt, ...);  
void ImGui::TextWrapped(const std::string& str);

参数:

- fmt: 一个格式化字符串, 允许你使用 ImGui 的格式化语法来插入变量。例如, 可以传入一个字符串, 或者传入多个参数, 通过 fmt 来格式化它们。
- str: 传入一个 std::string 对象。它会自动转化为 C 字符串并显示在界面上。

用法:

1. 基本用法:	ImGui::TextWrapped("This is a very long line of text that will automatically wrap when it reaches the edge of the window.");
2. 与格式化字符串一起使用: 你可以通过格式化字符串来显示动态内容。例如显示文件名、错误信息等。	const char* filename = "example.txt"; ImGui::TextWrapped("The file %s has been loaded successfully.", filename);
3. 使用 std::string: 如果你有一个 std::string 对象, 也可以直接传给 TextWrapped。	std::string filename = "example.txt"; ImGui::TextWrapped(filename); // 直接显示 std::string 的内容

》》》DragFloat 和 SliderFloat 的区别。

ImGui::DragFloat 和 ImGui::SliderFloat 的区别

<p>DragFloat:</p> <p>既可以通过鼠标在输入框中直接滑动, 也可以通过输入值。</p>	
<p>SliderFloat :</p> <p>只能操作滑块来改变大小。</p>	

-----Content browser panel (Drag & drop) -----

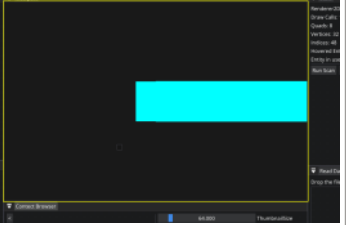
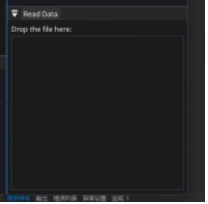
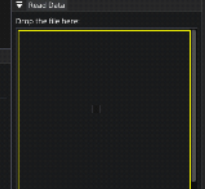
》》》BeginDragDropTarget()使用细节

如果手动跟进了 Chernor 的代码, 我们会发现, 使用 DragDrop 功能只需要两步操作: 设置拖动源、设置拖动目标。

<p>拖动源的设置</p>	<pre>// Allow drag &amp; drop function if (ImGui::BeginDragDropSource()) {     const wchar_t* item_path = relativePath.c_str();     ImGui::SetDragDropPayload("CONTENT_BROWSER_ITEM", item_path, (wcslen(item_path) * sizeof(wchar_t)));     ImGui::EndDragDropSource(); }</pre>
<p>拖动目标的设置</p>	<pre>ImGui::SetWindowPos(ImGui::GetCurrentWindow(), ImVec2(ImGui::GetCurrentWindow()-&gt;GetWindowPos().x, ImGui::GetCurrentWindow()-&gt;GetWindowPos().y + 1), ImGuiCond_IfNotSet); // Confirm boundary values ImGui::SetWindowPos(ImGui::GetCurrentWindow(), ImVec2(ImGui::GetCurrentWindow()-&gt;GetWindowPos().x, ImGui::GetCurrentWindow()-&gt;GetWindowPos().y + 1), ImGuiCond_IfNotSet); // Viewport bounds int x = ImGui::GetIO().DisplaySize.x; int y = ImGui::GetIO().DisplaySize.y; int w = ImGui::GetIO().DisplaySize.x; int h = ImGui::GetIO().DisplaySize.y; int x1 = 0; int y1 = 0; int x2 = x; int y2 = y; // Iframe::BeginDragDropTarget() if (ImGui::AcceptDragDropPayload("CONTENT_BROWSER_ITEM")) {     const wchar_t* path = (const wchar_t*)ImGui::GetDragDropPayloadData();     OpenFile(path, ImGui::GetCurrentWindow(), path);     ImGui::EndDragDropTarget(); }</pre>

》》可是还需要注意一点:

在使用 BeginDragDropTarget() 之前, 需要绘制一个有效的交互区域。

<p>比如在视口的设置之后, 我们使用了BeginDragDropTarget(), 你会发现拖动文件到视口区域时, 视口的可用区域会高亮, 并且能够处理后续文件拖入操作。</p> <p>可是如果注释掉 ImGui::Image() 这一行代码, 你会发现拖动文件的功能会无响应。</p> <p>这是因为 ImGui::Image 不仅显示了图像, 还会自动处理它的交互区域, 因此它是一个“有效”的拖放目标。</p>	
<p>如果你只绘制了一个窗口, 或者在窗口中放置了Text,Child等“不可交互”的空间, 可用区域高亮便不会出现。同样的, 文件拖动也会不起作用。</p> <p>Eq.</p> <pre>ImGui::Begin("Read Data") {     ImGui::Text("Read the file here.");     ImGui::Text("Drop the file here.");     ImGui::Text("Drop the file here.");     // 创建一个子窗口, 用于显示拖放目标区域     if (ImGui::BeginChild("DropTarget", ImVec2(100, 100)))     {         // 通过ImGui::Text绘制一个可交互的区域         ImGui::Text("Drop the file here."); // 只创建一个交互区域     }     // Allow drag &amp; drop     if (ImGui::BeginDragDropTarget())     {         if (ImGui::AcceptDragDropPayload("CONTENT_BROWSER_ITEM"))         {             const wchar_t* path = (const wchar_t*)ImGui::GetDragDropPayloadData();             ReadFile(path, ImGui::GetCurrentWindow(), path);             ImGui::EndDragDropTarget();         }     }     ImGui::End(); }</pre>	
<p>此时便需要我们创建一个可交互的区域: ImGui::Button, ImGui::Dummy 等等控件, 以此来完善文件拖动的功能。</p> <p>Eq.</p> <pre>// 创建一个子窗口, 用于显示拖放目标区域 if (ImGui::BeginChild("DropTarget", ImVec2(100, 100))) {     // 通过ImGui::Text绘制一个可交互的区域     ImGui::Text("Drop the file here."); // 只创建一个交互区域 } // Allow drag &amp; drop if (ImGui::BeginDragDropTarget()) {     if (ImGui::AcceptDragDropPayload("CONTENT_BROWSER_ITEM"))     {         const wchar_t* path = (const wchar_t*)ImGui::GetDragDropPayloadData();         ReadFile(path, ImGui::GetCurrentWindow(), path);         ImGui::EndDragDropTarget();     } }</pre>	



》》什么是 ImGui::Dummy

<b>概念:</b>	
ImGui::Dummy 是 ImGui 提供的一个函数, 用于创建一个 "占位符" 或 "虚拟" 元素, 它不会渲染任何实际的内容, 但可以用来占据空间或提供一个交互区域。	
<b>主要用途:</b>	占位符: ImGui::Dummy 可以作为一个占位符, 帮助你设置一些占用空间但不渲染任何实际内容的区域。这对于需要控制布局、调整空间或创建拖放目标区域非常有用。 控制布局: 通过 ImGui::Dummy, 你可以创建精确的布局区域, 而不会干扰其他控件的显示。例如, 当你需要创建一个特定大小的区域来接收拖放操作时, 可以使用 Dummy 来占据空间。
<b>语法:</b>	void ImGui::Dummy(const ImVec2& size);
<b>参数:</b>	size: 指定占位符的大小, 通常是一个 ImVec2 (x 和 y 坐标)。这定义了 Dummy 占据的区域的大小。
<b>示例:</b>	假设你想在 ImGui 窗口中创建一个区域, 它不会显示任何内容, 但你希望它占据一个特定的空间:  ImGui::Begin("Example Window");  // 创建一个大小为 200x200 的占位符区域 ImGui::Dummy(ImVec2(200, 200));  ImGui::End();