GameEngine5

2024年7月12日 15:06

----------一些维护和更改----------

》》》》 Made Win-GenProjects.bat work from every directory

代码更改：

```
@echo off
->pushd ..\
->pushd %~dp0\..\
call vendor\bin\premake\premake5.exe vs2019
popd
PAUSE
```

为什么要做这样的更改？

当你通过命令行提示符打开并运行该文件时，这个批处理文件会在命令行提示符被打开的位置被运行，这会导致 pushd ..\这个语句原本的语义错误。（在命令行提示符的路径下临时切换工作目录至相对于命令提示符的上一级 '..\'，而不是相较于批处理文件的上一级）

为了避免这样的情况发生，我们需要将启用该批处理文件后的语句改为绝对的、固定的切换目录操作。

%~dp0 的含义：

- %0 是批处理文件本身的名称。
- %~dp0 是批处理文件的完整路径，包括驱动器号和路径。它扩展为批处理文件所在的目录。这个路径在批处理文件内部是固定的，不会改变。

%~dp0\ 有什么意义？

pushd ..\	是将当前目录更改为上一级目录。
pushd %~dp0\..\	是将当前目录更改为批处理文件所在目录的上一级目录（绝对路径）。

Eg.

```
@echo off
echo this is %%cd%% %cd%
echo this is %%~dp0 %~dp0
```

当你在其他目录（比如 C:\）运行这个批处理文件时，这两个路径会不同

%cd%	显示的是当前目录
%~dp0	显示的是批处理文件所在的目录

》》》》 Fix Build warnings to do with BufferElement Offset Type

WIN64 size_t =>	unsigned __int64	intptr_t =>	__int64
ELSE size_t =>	unsigned int	intptr_t =>	int

In Any Case Uint32_t =>	unsigned int

(const void*)(intptr_t)element.Offset 中，Offset 仅仅是一个数值，没有任何与内存地址相关的含义。在这种情况下，intptr_t 的转换也是不必要的，因为它不会改变你正在做的事情的本质。

》》》》 Basic ref-counting system to terminate glfw

之前在 WindowsWindow.cpp 中，仅仅判断了GLFW窗口是否已经初始化？是否需要初始化上下文？然后根据判断结果进行 glfwDestroyWindow(m_Window); 这仅仅只是销毁了窗口。而且我们并没有通过 glfwTerminate(); 函数对GLFW 库进行终止，并释放资源。

这一次，我们判断打开的 GLFW 窗口是否全部关闭，如果全部关闭，则对 GLFW 库进行终止。若对一个窗口关闭之后，仍有正在运行的窗口，则仅销毁需要关闭的窗口即可。

Uint8_t 的定义：	typedef unsigned char uint8_t;

》》》》 Added file reading check

```
std::string OpenGLShader::ReadFile(const std::string& filepath)
{
    std::string result;
    std::ifstream readIn(filepath, std::ios::in | std::ios::binary);
```

```
        if (readIn)                             //是否成功打开
        {
            readIn.seekg(0, std::ios::end);
            size_t size = readIn.tellg();
            if (size != -1) {                   //是否成功获取
                ...
            }
            else { NUT_CORE_ERROR("Failed to read file from : '{0}'", filepath); }
        }
        else { NUT_CORE_ERROR("Could not open file form : '{0}'", filepath); }
    }
```

》》》》Code maintenance (#165)

构造函数 A() = {} 和 A() = default 有什么区别吗?

- 实现上:

A() = {}	是显式手动定义一个空的默认构造函数，不依赖于编译器生成。
A() = default	是告诉让编译器来生成默认构造函数。

- 性能上:
  这两种方式行为相同，运行时也生成相同的机器码。因此，在生成的代码执行效率上，不会有显著的区别。

- 结论：二者没有什么区别。