Metal学习笔记

Metal-cpp

引用计数在NSObject中管理，使用官方提供的NSSharedPtr.hpp不会有额外指针空间开销(关于更多ARC机制和内存查看OC学习笔记)。
NS::SharedPtr是引用计数的共享指针，重载了运算符管理NSObject的引用计数。
NS::TransferPtr()可以将普通NSObject指针转化为NS::SharedPtr，不增加引用计数，但销毁会减少引用计数
NS::RetainPtr()可以将普通NSObject指针转化为NS::SharedPtr，增加引用计数，销毁会减少引用计数

Cpp和OC交互

桥接的关键字可以帮助您在 objc 和 C++ 之间进行转换，还可以帮助您转移对象所有权，_bridge转换可以在objc和cpp对象之间进行强制转换，但是不会进行ARC所有权转移，__bridge_retained可以将objc指针强制转换为cpp指针，并从 ARC 获得所有权，__bridge_transfer能够将cpp指针移动到objc，并将所有权转移给ARC。

Metal编译

Xcode在编译.metal文件时：
如果metal文件和目标源文件一起编译，那么metal文件会被编译为default.metallib，程序运行时可以使用newDefaultLibrary来获取默认metal库。

```
auto mtLibrary = TransferPtr(mtDevice->newDefaultLibrary());
```

如果metal预先编译为.metallib静态库，当连接到目标源文件时，会打包在应用资源文件夹下，可以通过newLibraryWithURL来获取此静态库。

```
const char* cPath = Bundle::mainBundle()->resourcePath()->utf8String();
std::string pathStr(cPath);
pathStr += "/metal-shader.metallib";
auto mtLibURL = TransferPtr(URL::alloc()->initFileURLWithPath(String::string(pathStr.c_str(), UTF8StringEncoding)));
auto mtLibrary = TransferPtr(mtDevice->newLibrary(mtLibURL.get(), &e));
```

如果不预编译，要在运行时编译，可通过传入metal数据字符串来获取metal库。

```
const char* metalSrc = R"(metal代码)";
auto mtLibrary = TransferPtr(mtDevice->newLibrary(String::string(metalSrc, UTF8StringEncoding), &e));
```

文章：

https://www.jianshu.com/p/cddf73c6c05e
https://github.com/LeeTeng2001/metal-cpp-cmake
https://nicolaschavez.com/projects/metal-cpp-extensions/
https://nicolaschavez.com/projects/metal-toolkit/