Model / Assimp

-큐브같이 간단한 정적인 구조를 가지고 있는 경우에는 모든 좌표를 정의할 수 있었음

-그러나 사람, 차량 같은 복잡한 도형의 모든 정점, 법선벡터, 텍스처좌표를 정의하는 것은 매우 힘든 일임

-이러한 모델을 사용하는 것은 응용프로그램에 import하여 사용하는 것

-모델들

-모델들은 3D 아티스트들이 여러 툴을 사용하여 디자인 한 것들

-추출된 모델에서는 모든 기술적인 관점들이 숨겨져 있음, 이러한 기술적인 상세부분을 신경써야함

-Wavefront.obj 와 같은 포멧

-Color, Diffuse, Specular Map 같은 작은 material 정보들을 포함

-XML기반의 COllada 파일 포멧

-model, light, 다양한 종류의 materials, 애니메이션 데이터, 카메라, 완전한 scene 정보 등 광범위한 정로를 가지고있음

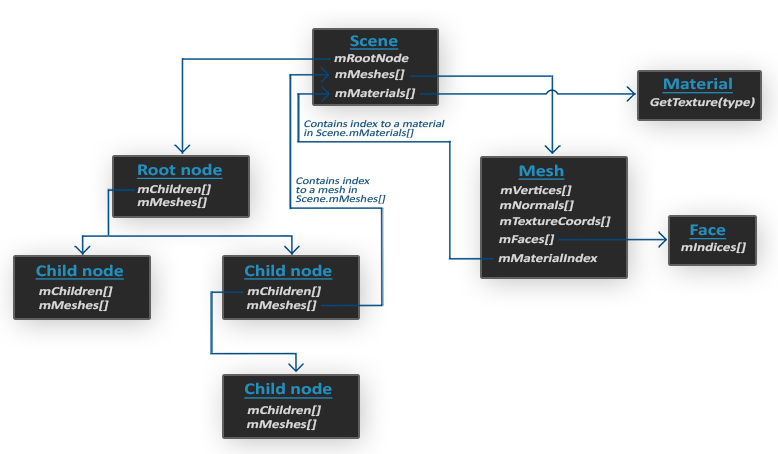
-WaveFront 오브젝트 포멧이 분석하기 쉬운 모델 포멧

-여러 포멧의 모델을 추가할려면 importer을 작성해야 하는데, 이런 일을 해주는 라이브러리가 잇음

A Model loading library

-가장 많이 쓰이는 라이브러리 Assimp (Open Asset Import Library)

-모든 모델의 데이터들을 데이터 구조로 불러와 import, 추출할 수 있음



-Scene/Model 의 모든 데이터는 Material, Mesh 들과 마찬가지로 Scene 객체에 포함됨, 루트 노트에 대한 참조를 가지고 있음

-Scene의 루트 노드는 자식 노드들을 포함(다른 모든 노드들과 마찬가지로)할 수 있고 scene 객체의 mMeshes 배열 안의 데이터를 가리키는 인덱스들의 모음을 가지고 있음. 이 루트 노드의 mMeshes 배열은 실제 Mesh 객체들을 가지고 있고 일반 노드의 mMeshesM 배열에 들어있는 값은 오직 scene의 mesh 배열에 대한 index들만 가지고 있음

-Mesh 객체는 렌더링하는 데에 필요한 모든 관련 데이터들을 포함. 오브젝트의 vertex 위치, 법선 벡터, 텍스처 좌표, 면, material ...

-Mesh는 여러개의 면들을 가집니다. 면(Face)은 렌더링 기본 오브젝트(삼각형, 사각형, 점)를 나타냄. 면은 primitive를 형성하기 위한 vertex들의 index를 가지고 있음. vertex들과 index들이 분리되어있기 때문에 index 버퍼를 통해 렌더링하는 것을 쉽게 만들어줌

-마지막으로 mesh는 Material 객체도 가짐. 이 객체는 오브젝트의 material 속성들을 얻기위한 여러가지 함수들을 관리(텍스처 map(diffuse, specular map)과 컬러)

-Assimp 빌드하기

-본 강의에서 사용하는 Assimp 빌드 방법은 Visual Studio를 사용한 CMake로 빌드하였음

-현 사용하고 있는 프로그램은 Visual Studio Code를 사용, Mingw32를 사용하여 빌드 중

-따라서 개별적으로 빌드하는데 시행착오, 좀 오랜 시간이 걸렸음

-Assimp 소스코드를 설치

-Assimp폴더 내부에 build폴더 생성

-Cmake를 이용해 Assimp폴더를 build폴더 내부에 빌드

-빌드 할 때, configure에서 MinGW makefile을 이용해 빌드해야함

-cmd창으로 build폴더 내에서 Mingw32-make 명령어를 이용, 빌드

-빌드 도중 오류가 중간 중간 생기는데, cmd창에 나오는 오류 내역을 보고 수정

-빌드 완성

-완성된 Assimp 폴더에 헤더파일이 있는 include폴더를 작업중인 StudyOpenGl폴더에 넣음

-Assimp/build/code 폴더에 있는 libassimp.dll.a, libassimp.dll 파일을 가져옴

-Assimp/contrib/zlib 폴더에 있는 libzlibstatic.a 파일을 가져옴

-각각 자기가 작업중인 환경에 맞는 위치에 넣어주고, task.json파일을 수정

-간단한 코드 실행으로 적용이 되었는지 확인

|  |
| --- |
| #include <assimp/Importer.hpp>  int main(){  Assimp::Importer imp;  return 0;  } |

-빌드, excute 전부 실행 되는 것을 확인함