Computer Graphics: Rendering (Model)

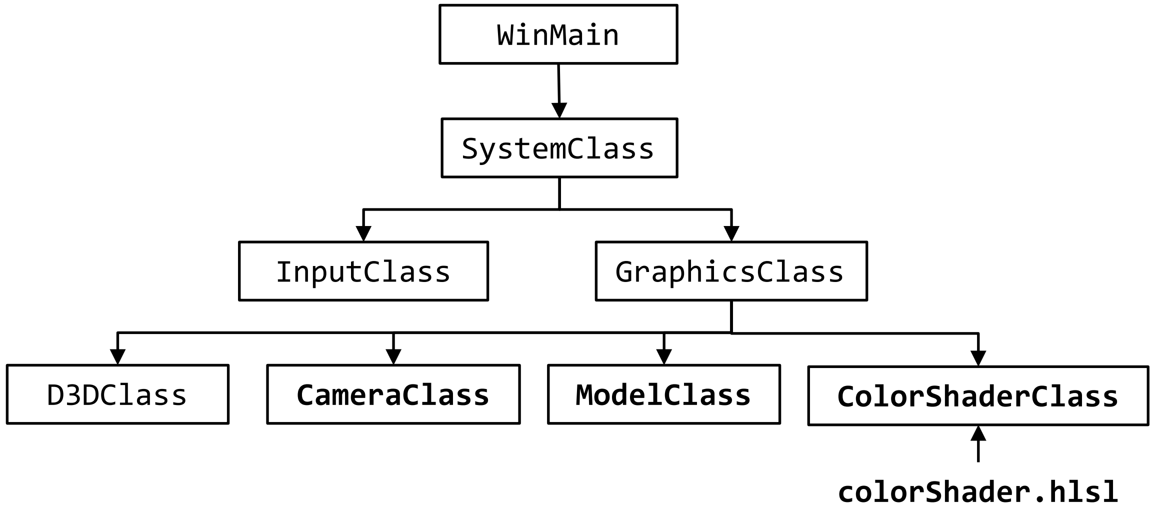
Dept. of Game Software
Yejin Kim

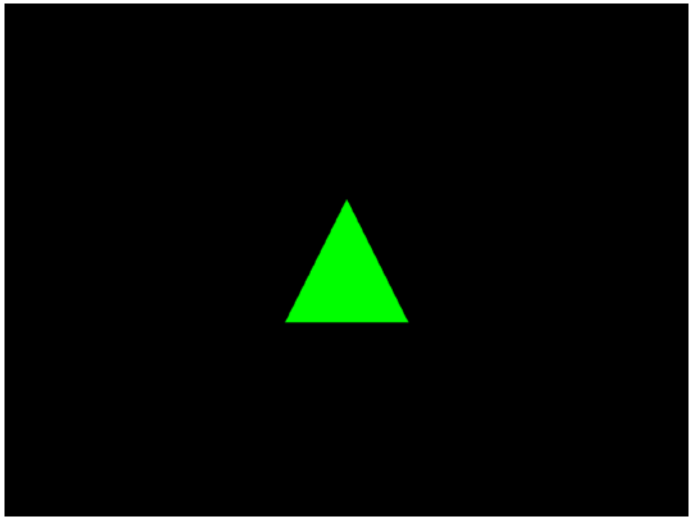
Tutorials

- Shaders
- 3D Model (OBJ)
- Instancing
- 3D Model (FBX)

2-1 Shaders

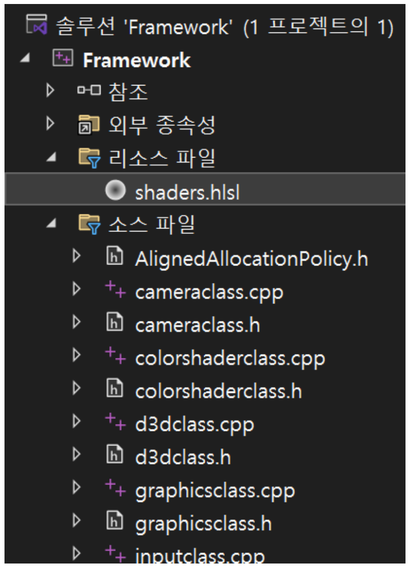
- Adding rendering classes to the Framework
  - AlignedAllocationPolicy: allocate a memory in 16-bit alignment
    - Necessary for GPU programming which processes data in 16-bits
  - CameraClass: handle the camera in the 3D space
  - ModelClass: handle the 3D models
  - ColorShaderClass: render the model using HLSL (shaders.hlsl)

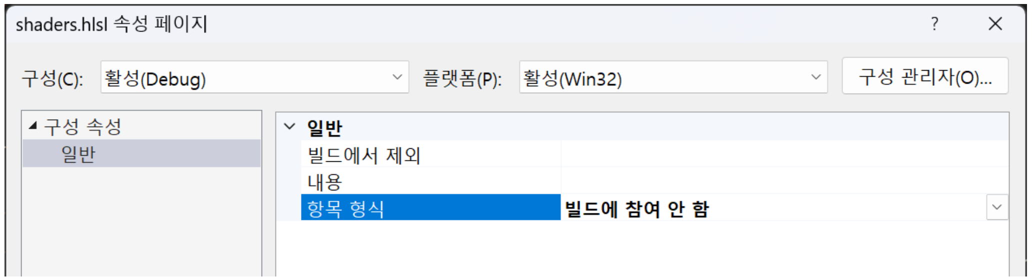




2-1 Shaders

- Shader file: *.hlsl
  - A shader file can be defined with any extension name
    - e.g. effects.fx, colorShader.hlsl, colors.shader, etc.
  - A vertex and a pixel shader is defined as a function respectively in the shader file
    - These functions can be defined in a separate file: shaders.vs, shaders.ps
  - A shader file should be excluded from project build
    - Project → shaders.hlsl → 속성 → 항목 양식: 빌드에 참여 안 함


솔루션 'Framework' (1 프로젝트의 1)
Framework
참조
외부 종속성
리소스 파일
shaders.hlsl
소스 파일
AlignedAllocationPolicy.h
cameraclass.cpp
cameraclass.h
colorshaderclass.cpp
colorshaderclass.h
d3dclass.cpp
d3dclass.h
graphicsclass.cpp
graphicsclass.h
inputclass.cpp


shaders.hlsl 속성 페이지

구성(C): 활성(Debug)   플랫폼(P): 활성(Win32)   구성 관리자(O)...

구성 속성
일반

일반
빌드에서 제외
내용
항목 형식: 빌드에 참여 안 함

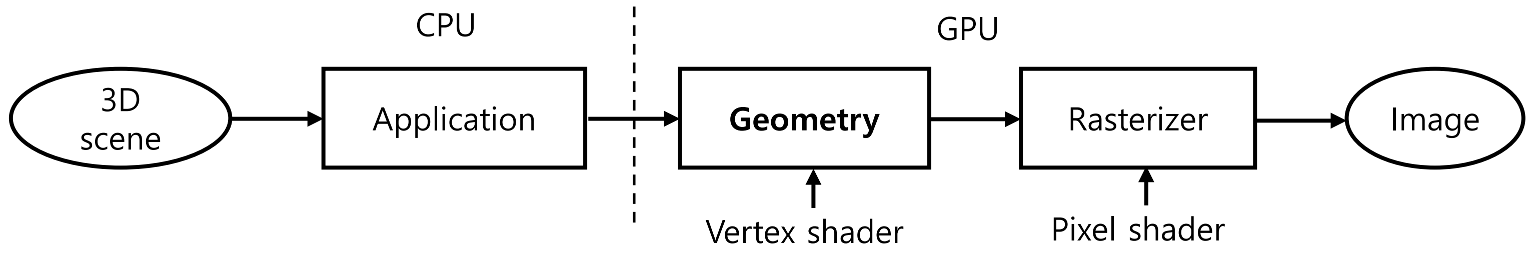
2-1 Shaders

- Geometry data buffers
  - Vertex: a data array for a vertex list
  - Index: a data array to find a vertex in the vertex buffer
  - ID3D11Device::CreateBuffer()

```
// Set up the description of the static vertex(or index) buffer
D3D11_BUFFER_DESC BufferDesc;
BufferDesc.Usage = D3D11_USAGE_DEFAULT;
BufferDesc.ByteWidth = sizeof(VertexType) * object[0].m_vertexCount;
BufferDesc.BindFlags = D3D11_BIND_VERTEX_BUFFER;
BufferDesc.CPUAccessFlags = 0;
BufferDesc.MiscFlags = 0;
BufferDesc.StructureByteStride = 0;

// Give the subresource structure a pointer to the vertex(or index) data
D3D11_SUBRESOURCE_DATA Data;
Data.pSysMem = object[0].vertices;
Data.SysMemPitch = 0;
Data.SysMemSlicePitch = 0;

// Create the vertex(or index) buffer
ID3D11Buffer Buffer;
ID3D11Device::CreateBuffer(&BufferDesc, &Data, &Buffer);
```

2-1 Shaders

- Geometry stage

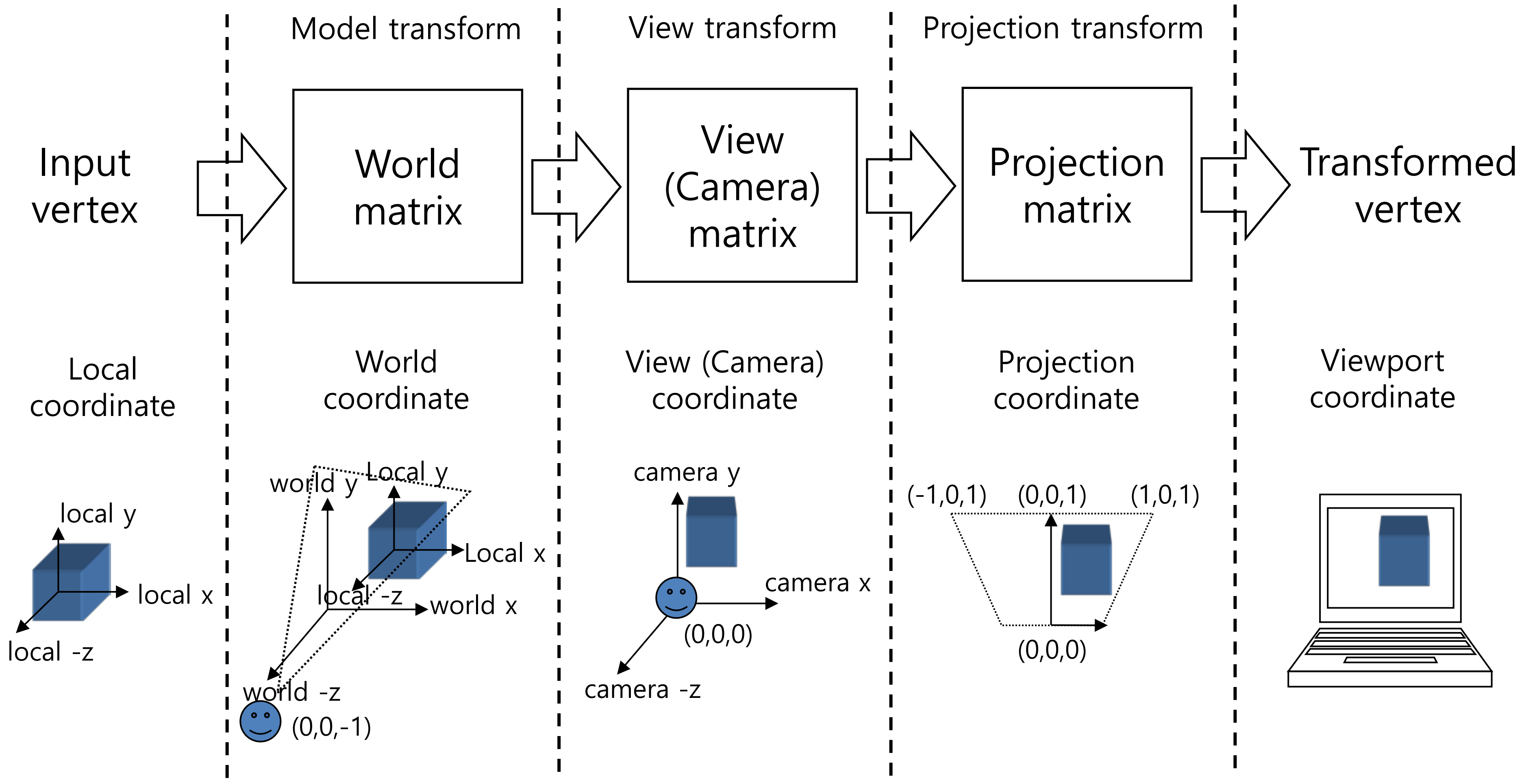


  - Vertex: handles the processing of individual vertices
  - Pixel: colors the polygons

- Vertex transformation matrices
  - World matrix
  - Camera (View) matrix
  - Projection matrix
    - Perspective projection
    - Orthogonal (parallel) projection

2-1 Shaders

- Geometry stage: Vertex transformation matrices
  - Maps an input vertex on a screen



2-1 Shaders

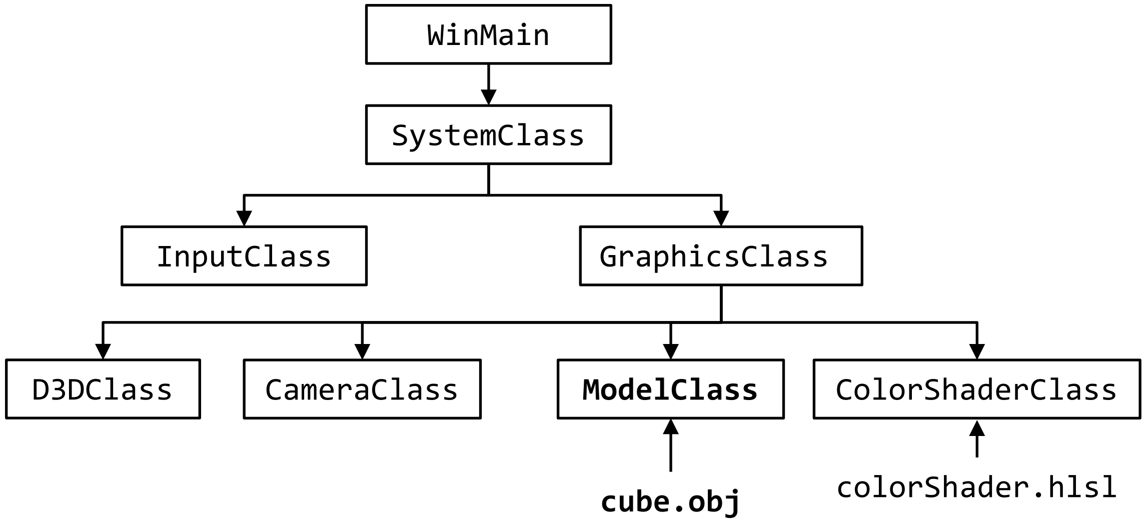
- Model translation, rotation, and scale
  - XMatrixTranslation(x, y, z)
  - XMatrixRotationX(radian)
  - XMatrixRotationY(radian)
  - XMatrixRotationZ(radian)
  - XMatrixScaling(x, y, z)

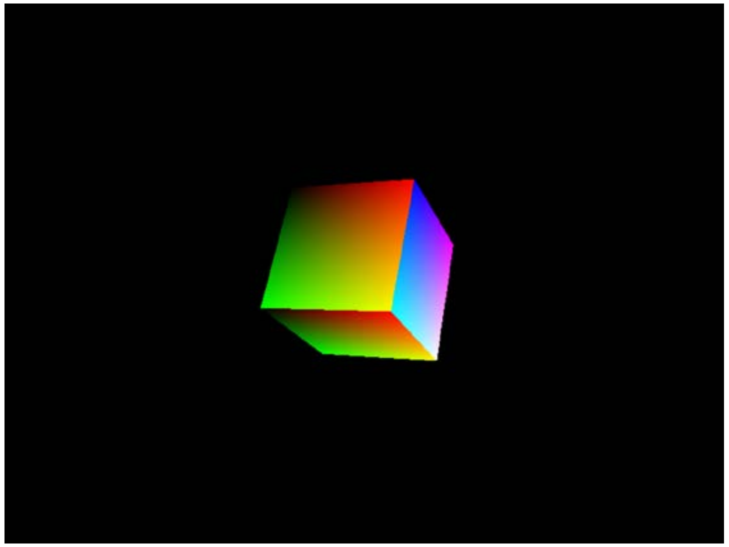
```
XMMATRIX worldMatrix;

m_D3D->GetWorldMatrix(worldMatrix);
worldMatrix *= XMMatrixScaling(0.5f, 0.5f, 0.5f);
worldMatrix *= XMMatrixRotationY(180.0f/XM_PI);       // Radian angle
worldMatrix *= XMMatrixTranslation(-2.0f, 0.5f, 0.0f);
```

2-2 3D Model (OBJ)

- Loading a 3D model from an external file
  - ModelClass: loads 3D model data from an OBJ file

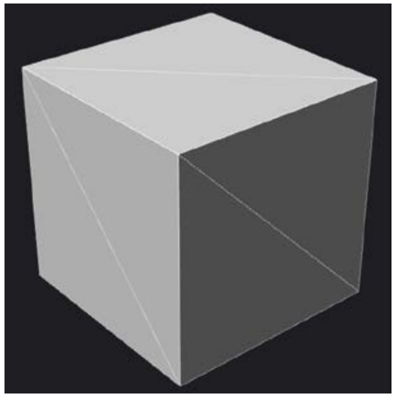
- Rotating and translating the loaded model
  - Use a world matrix





2-2 3D Model (OBJ)

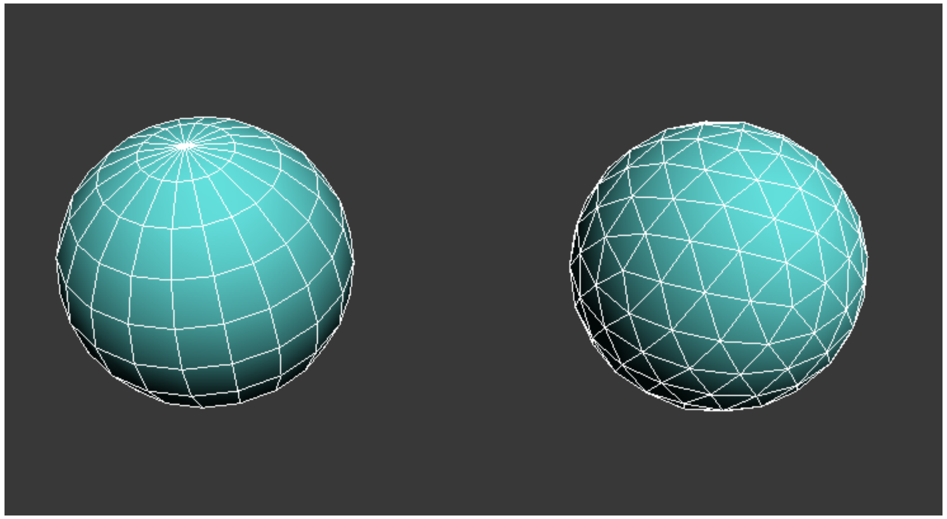
- Loading an external model: OBJ
  - Geometry: "cube.obj"
  - Material: "cube.mtl" (*Not necessary)
    - Include image file names
  - Image: use DDS
    - JPG/JPEG, PNG, GIFF, TIFF → DDS
  - Do NOT include an OBJ file in the "솔류션 탐색기" of a VS project → Build error…!



```
mtllib cube.mtl
g default
v −0.500000 −0.500000 0.500000
…

vt 0.001992 0.001992
…

vn 0.000000 0.000000 1.000000
…

s 1
g pCube1
usemtl file1SG
f 1/1/1 2/2/2 3/3/3
f 3/3/3 2/2/2 4/4/4

s 2
f 3/13/5 4/14/6 5/15/7
f 5/15/7 4/14/6 6/16/8
…
```

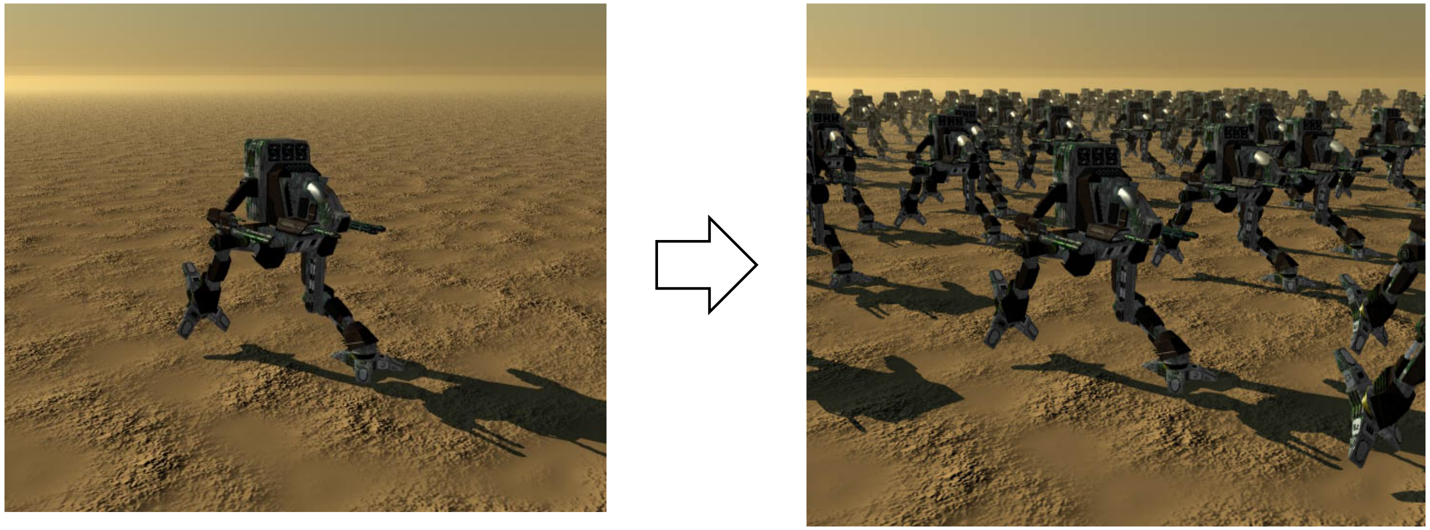
2-2 3D Model (OBJ)

- Loading an external model file
  - Using triangular mesh models
    - deviceContext->IASetPrimitiveTopology(D3D11_PRIMITIVE_TOPOLOGY_TRIANGLELIST);
  - Using quadrilateral mesh models
    - deviceContext->IASetPrimitiveTopology(D3D11_PRIMITIVE_TOPOLOGY_TRIANGLESTRIP);
  - Convert Quad → Triangle using 3D modeling tool
    - Blender, Max, Maya, etc.



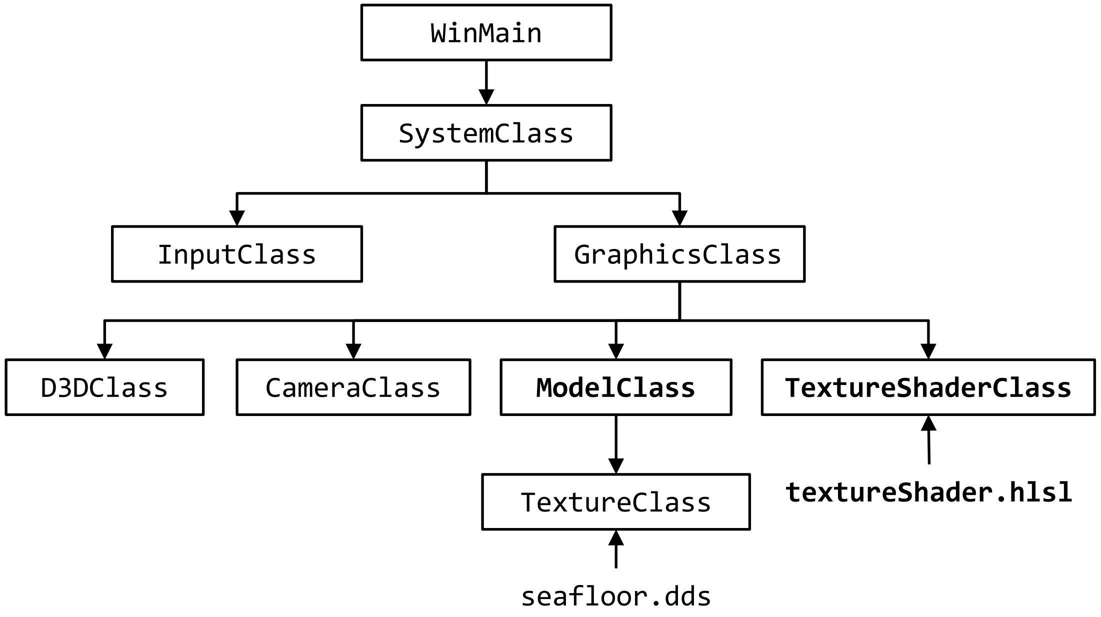
2-3 Instancing

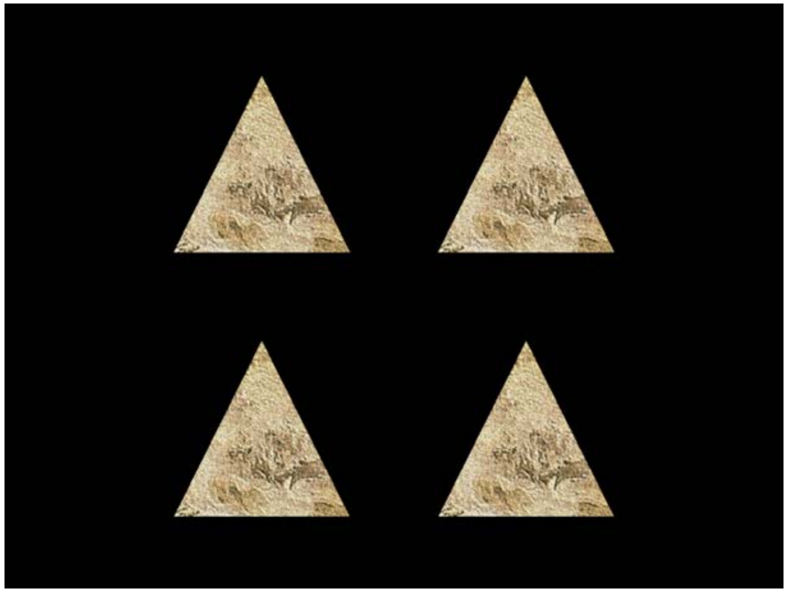
- Mesh Instances
  - Render multiple copies of the same geometry with just changes in position, scale, color, etc.
  - Single vertex buffer + **instance** buffer



2-3 Instancing

- Adding multiple instances to the Framework
  - ModelClass: handles an instance buffer
  - TextureShaderClass: handles setting up instances for the shader



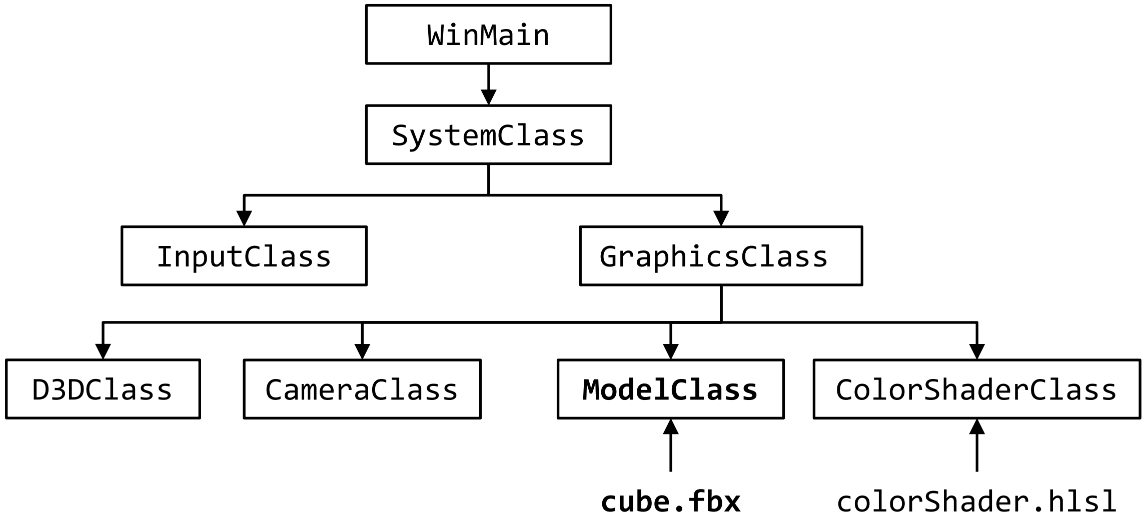


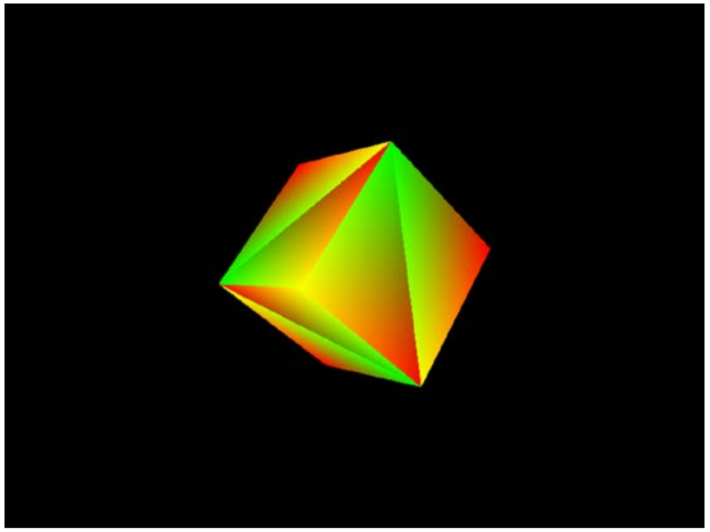
2-4 3D Model (FBX)

- Other 3D file formats: Open Asset Import Library (ASSIMP)
  - https://assimp.org/
  - Written in C++
  - Under a liberal BSD license
  - Loads all input model formats into one straightforward data structure for further processing
  - Augmented by various post processing tools, including frequently-needed operations such as computing normal and tangent vectors.
  - Supports more than 40 file formats

2-4 3D Model (FBX)

- Loading a 3D model from an external file
  - ModelClass: loads 3D model data from a FBX file
  - [Project] 속성 → C/C++ → 추가 포함 디렉터리: include
  - .₩lib₩assimp-vc142-mtd.lib: a ASSIMP library file (x86 Debug)
  - .₩include₩assimp: header files for ASSIMP





References

- Wikipedia
  - www.wikipedia.org
- Introduction to DirectX 11
  - www.3dgep.com/introduction-to-directx-11
- Raster Tek
  - www.ratertek.com
- Braynzar Soft
  - www.braynzarsoft.net)
- CS 445: Introduction to Computer Graphics *[Aaron Bloomield]*
  - www.cs.virginia.edu/~asb/teaching/cs445-fall06

Q & A