Computer Graphics Practice

Lecture 05

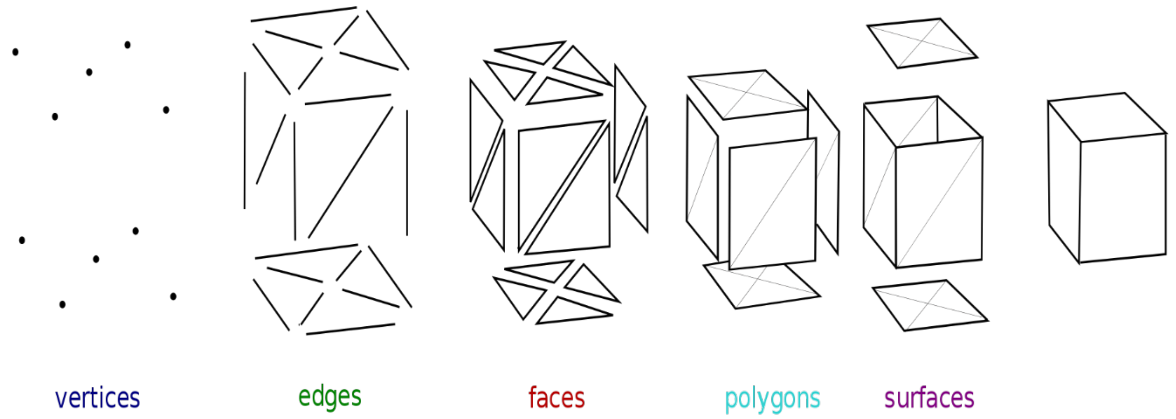
Dept. of Game Software
Yejin Kim

Plan

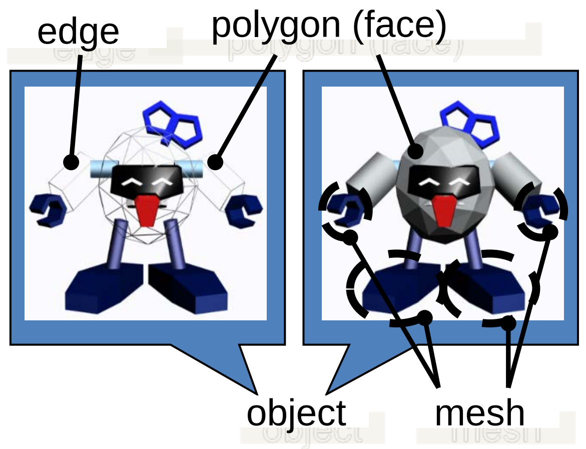
- 3D Geometry
  - Polygon
  - Mesh Representation
  - D3D Buffers and Shaders
- Tutorial
  - Buffers, Shaders, and HLSL
  - 3D Model Rendering (*After Tutorial: texturing)
  - Instancing (*After Tutorial: texturing)

3D Geometry

- Polygon
  - A collection of vertices and edges: triangles, quadrilaterals(quads)
  - HW support for rendering: 3 or 4-sided faces

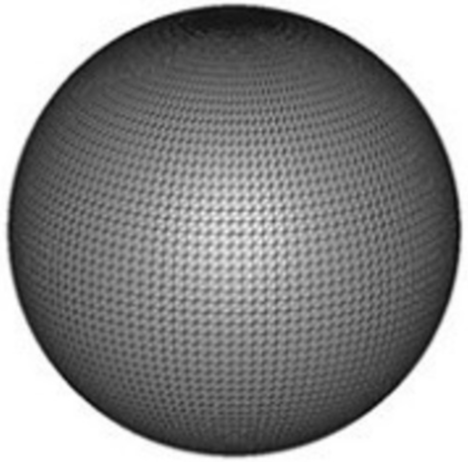
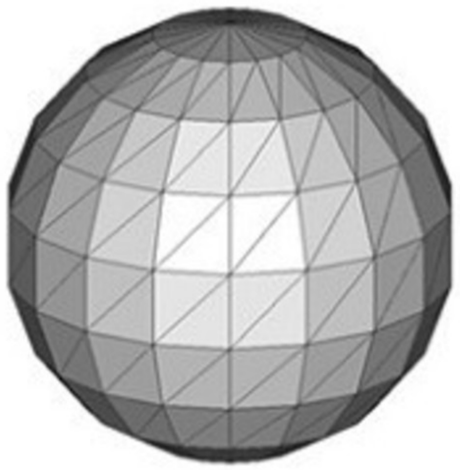


- (Polygonal) Mesh
  - Surface: a collection of polygons
- Object
  - A collection of meshes



3D Geometry

- Surface representations
  - Analytical form (분석적 형태)
  - Collection of patches (패치들의 집합)
  - Triangle mesh (삼각 메쉬)

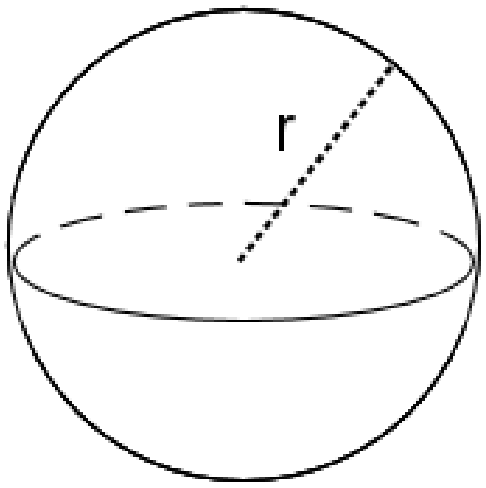



3D Geometry

- Mesh representation: Analytical form
  - Parametric surface equation (표면 방정식)

    e.g. Sphere equation:

$$x^2 + y^2 + z^2 = r^2$$

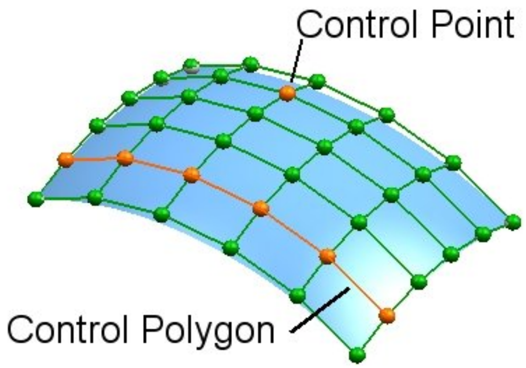


3D Geometry

- Mesh representation: Collection of patches
  - Patch: a curved plane composed of a set of rectangles
  - Similar to quilt

    e.g. NURBS, Bezier surfaces, subdivision surface

Quilt



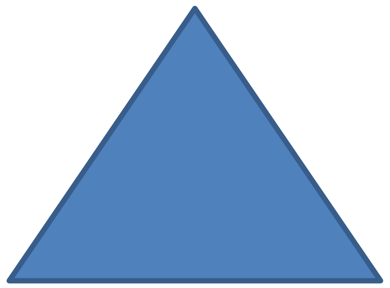
NURBS surface

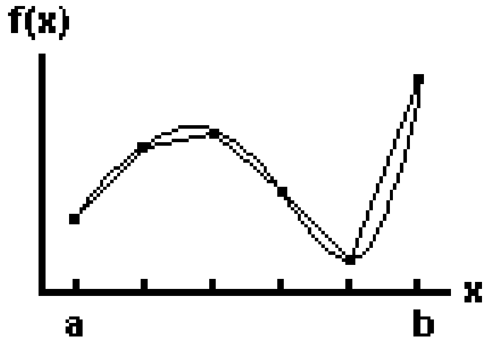
Artwork posted at CGbrainchild gallery
Jinwoo Lee © jermain72@dreamwiz.com

NURBS modeling

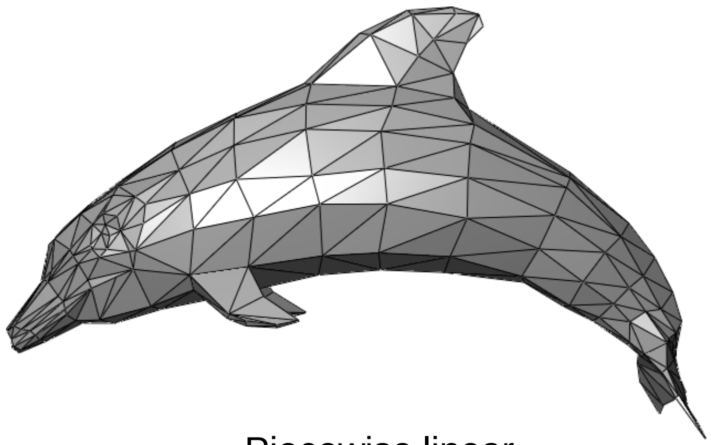
3D Geometry

- Mesh representation: Triangle mesh
  - Simplest type of polygon
  - Always planar
  - Still a triangle after transformations
    - e.g. Affine (projective) transformation
  - Hardware acceleration support
  - Piecewise-linear approximation(구분적 선형 근사)
    - used for curved objects





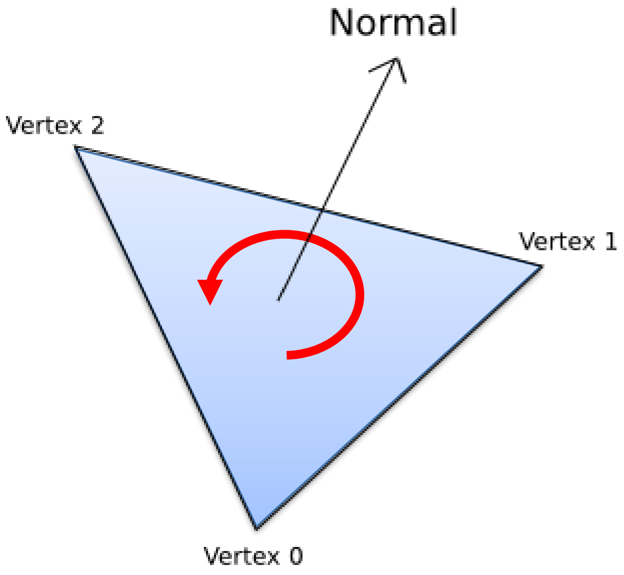
Piecewise linear
approximation to a function



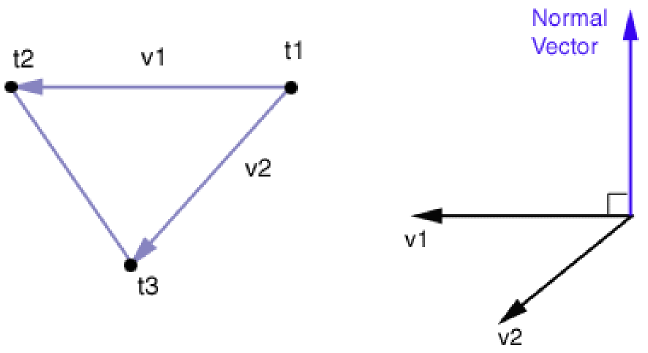
Piecewise linear
approximation to a surface

3D Geometry

- Triangle mesh representation: winding order
  - Decide a polygon (front or back) side
  - Counterclockwise(CCW) or clockwise(CW)



winding order: CCW



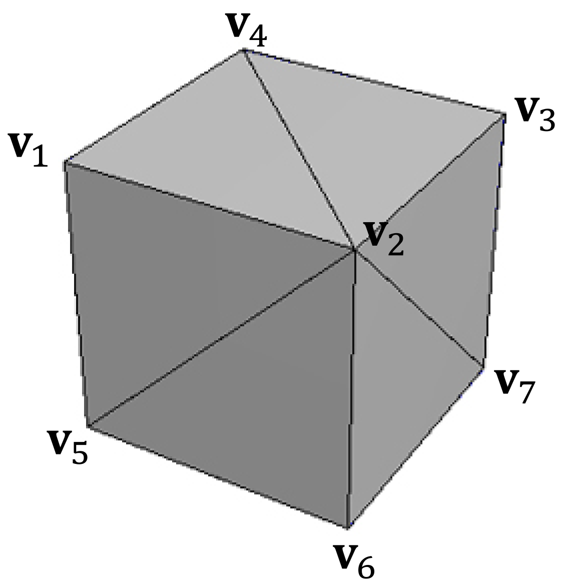
$$\mathbf{n} = \frac{\mathbf{v}_1 \times \mathbf{v}_2}{|\mathbf{v}_1 \times \mathbf{v}_2|}$$

computing normal direction

3D Geometry

- Triangle mesh representation
  - Indexed triangle list

$\mathbf{v}_i$: vertex 3D position



vertex list:

0	1	2	3	4	5	6	7
$\mathbf{v}_1$	$\mathbf{v}_2$	$\mathbf{v}_3$	$\mathbf{v}_4$	$\mathbf{v}_5$	$\mathbf{v}_6$	$\mathbf{v}_7$	$\mathbf{v}_8$

indexed triangle list:

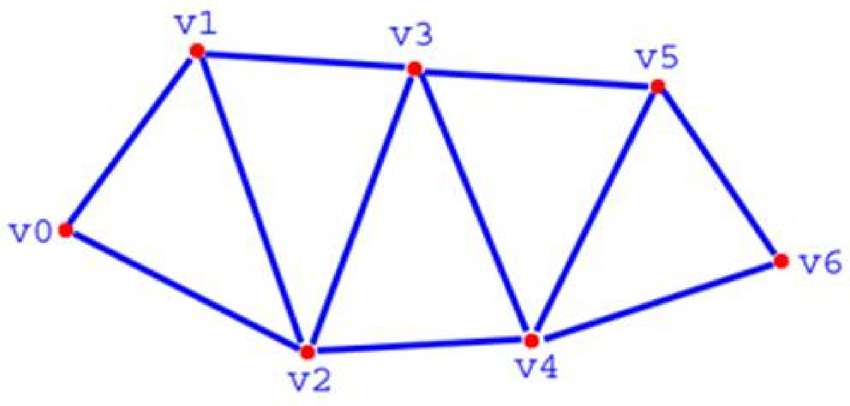
0	1	3	1	2	3	0	4	1	…

3D Geometry

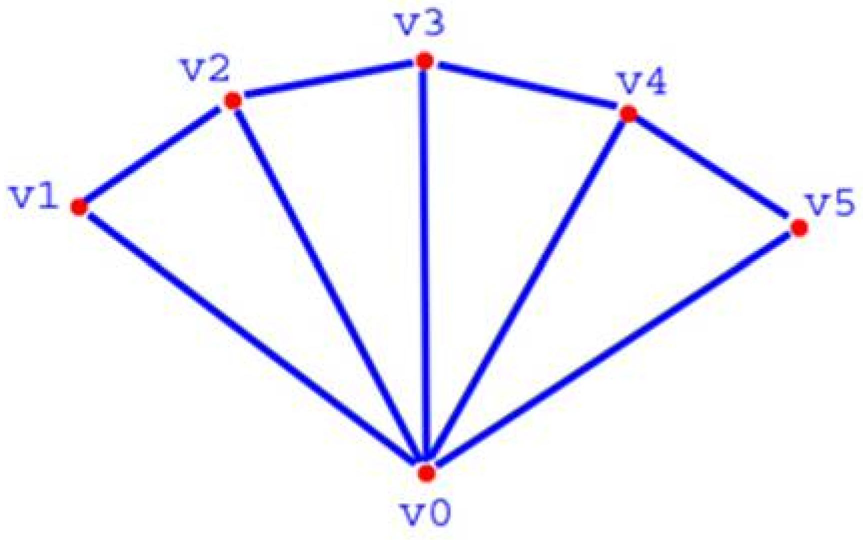
- Triangle mesh representation
  - Triangle strip
  - Triangle fan

Vertex list:

$\mathbf{v}_0$	$\mathbf{v}_1$	$\mathbf{v}_2$	$\mathbf{v}_3$	$\mathbf{v}_4$	$\mathbf{v}_5$	$\mathbf{v}_6$



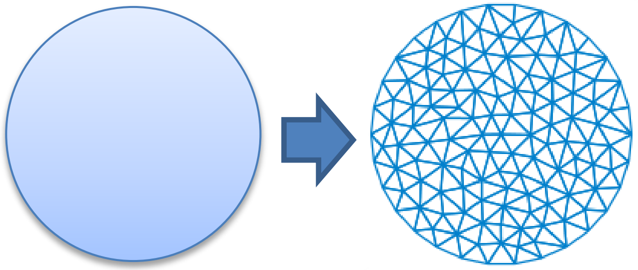
triangle strip



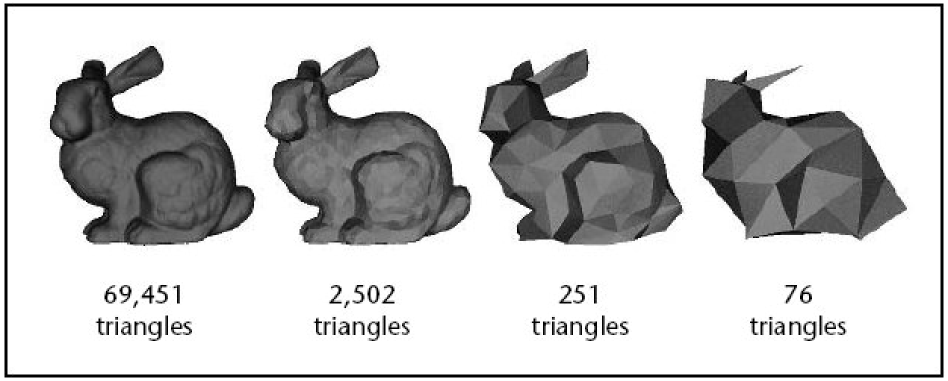
triangle fan

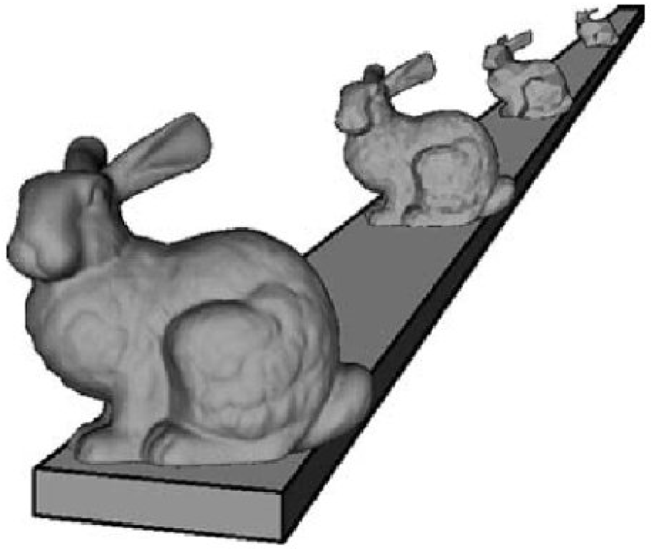
3D Geometry

- Triangle mesh representation: tessellation
  - Divide a surface into a collections of triangles



e.g. Level-of-detail(LOD)



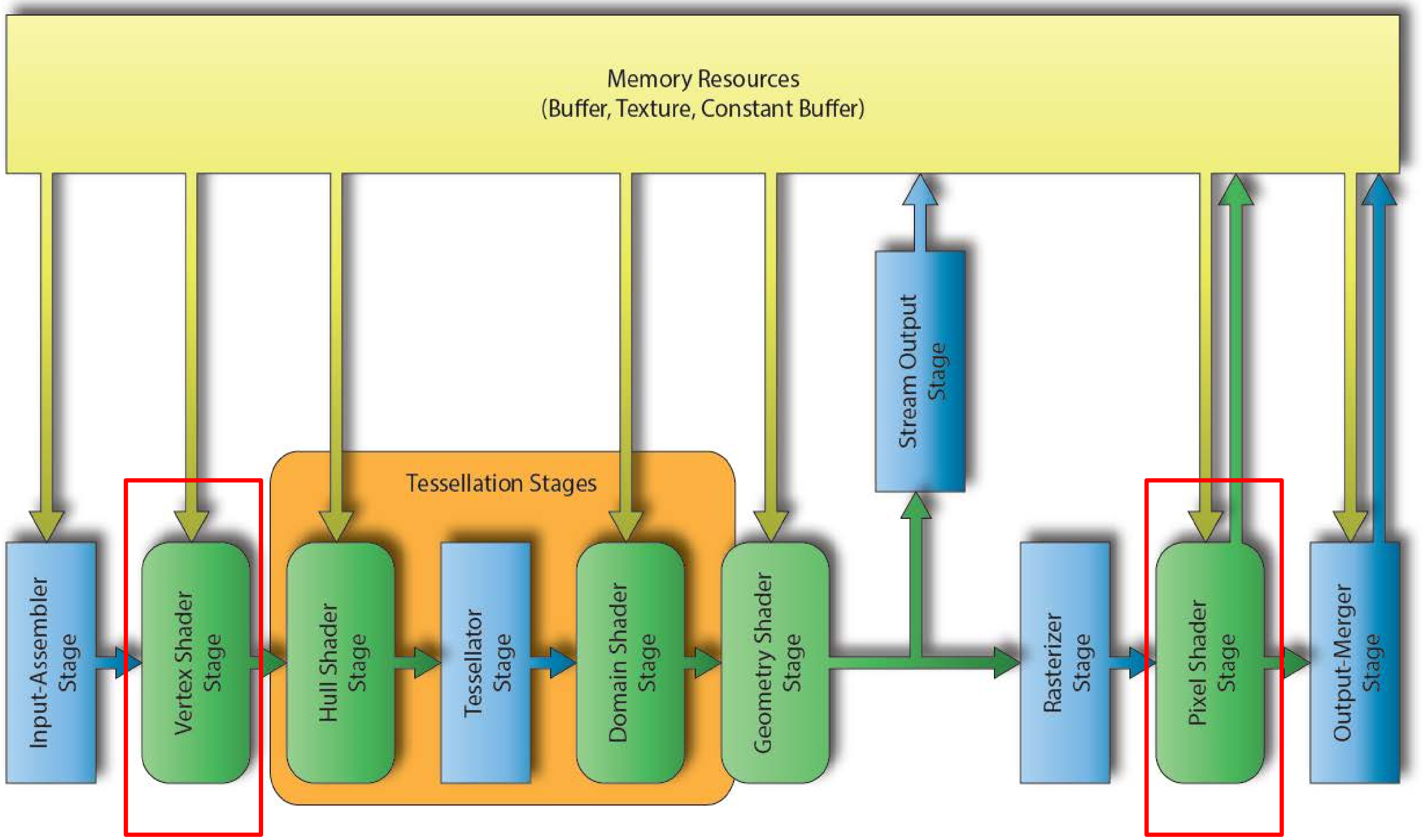


Tutorial: Buffers, Shaders, and HLSL

- Shader Models
  - DirectX 1~7 (1995~1999): fixed-function rendering pipeline
  - DirectX 8 (2000): Shader Model 1
    - vertex shader
  - DirectX 9 (2002): Shader Model 2
    - vertex, pixel shaders
  - DirectX 9c (2004): Shader Model 3
    - extended vertex, pixel shaders
  - DirectX 10 (2006): Shader Model 4
    - vertex, pixel, geometry shaders
    - effect file: .fx
  - DirectX 11 (2009): Shader Model 5
    - vertex, pixel, geometry, tessellation shaders
  - DirectX 12 (2014): Shader Model 6
    - extended vertex, pixel, geometry, tessellation shaders
    - raytracing

Tutorial: Buffers, Shaders, and HLSL

- DirectX 11 Rendering Pipeline

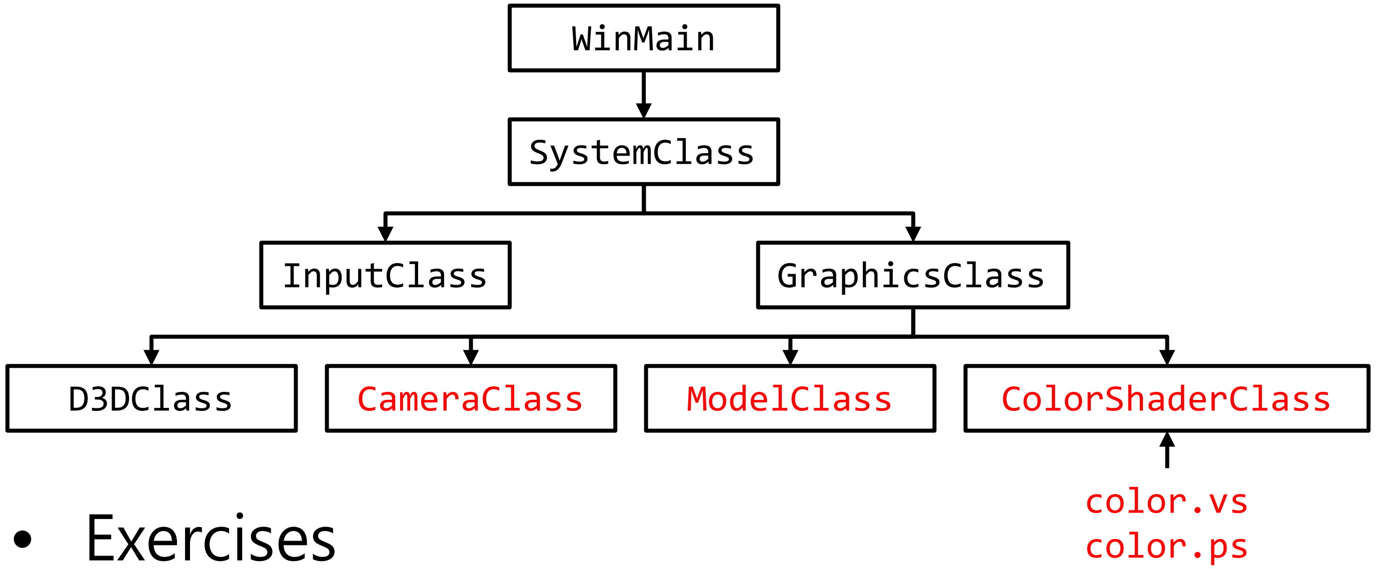


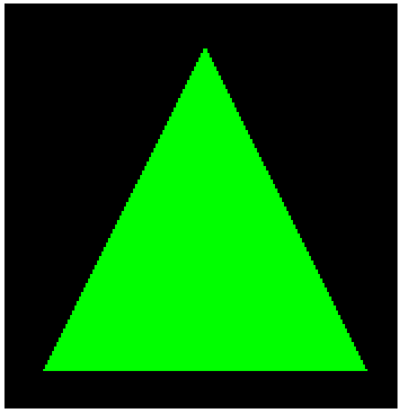
Tutorial: Buffers, Shaders, and HLSL

- Buffers, Shaders, and HLSL
  - **Vertex buffer**: a data array for a vertex list
  - **Index buffer**: a data array to find a vertex in the vertex buffer
    - Increase the possibility of caching the vertex data in faster locations in video memory
  - **Vertex shader**: a programmable stage in the rendering pipeline that handles the processing of individual vertices
    - Transform the vertices from the vertex buffer into 3D space
    - Manipulate vertex properties: position, color, texture coordinates, etc.
  - **Pixel (fragment) shader**: a programmable stage in the rendering pipeline that colors the polygons
    - Coloring, texturing, lighting, and other effects on the polygon pixels
  - **HLSL**: high-level shader language
    - Similar to C language with predefined types

Tutorial: Buffers, Shaders, and HLSL

- Adding geometry classes to the Framework
  - **CameraClass**: handle the camera in the 3D space
  - **ModelClass**: handle the 3D models
  - **ColorShaderClass**: render the model using HLSL

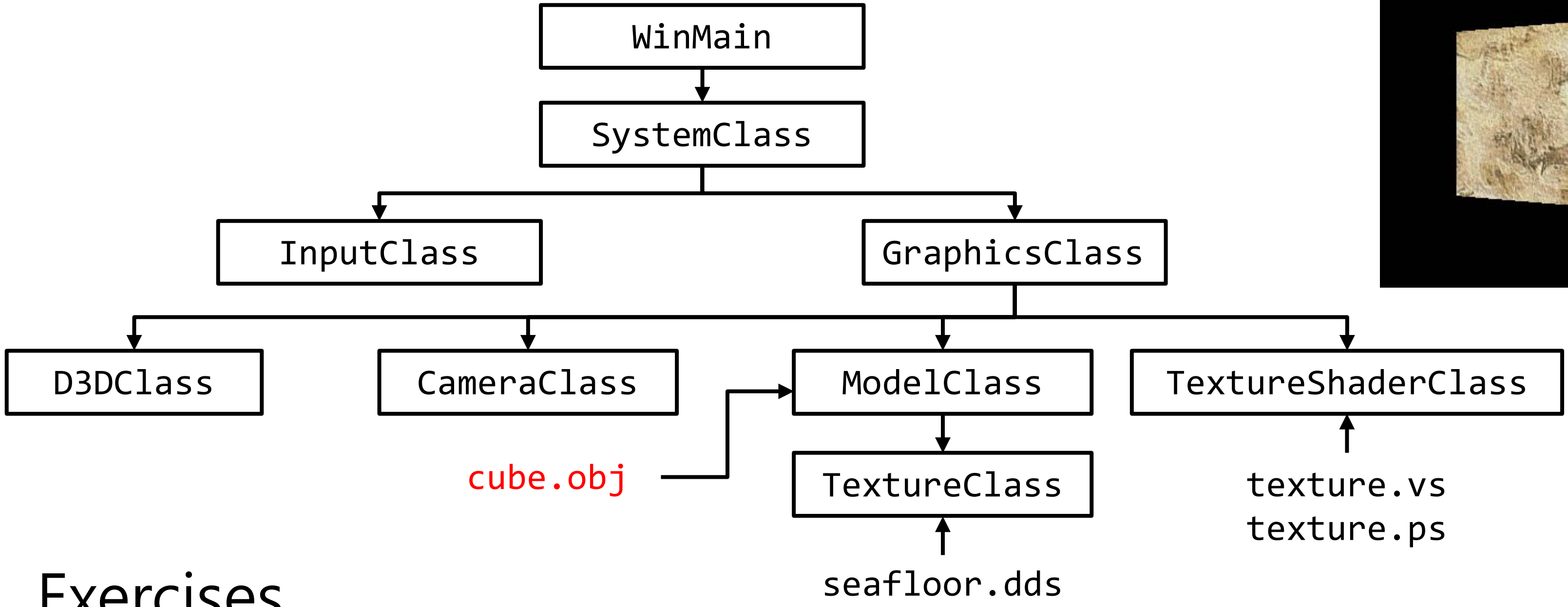


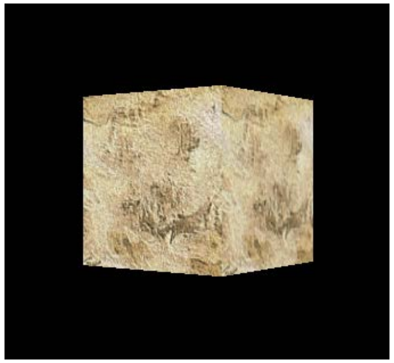


- Exercises
  - Add more polygons with different shapes and colors
  - Change the pixel shader to output the color half as bright. (Hint: multiply something in ColorPixelShader)

Tutorial: 3D Model Rendering

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

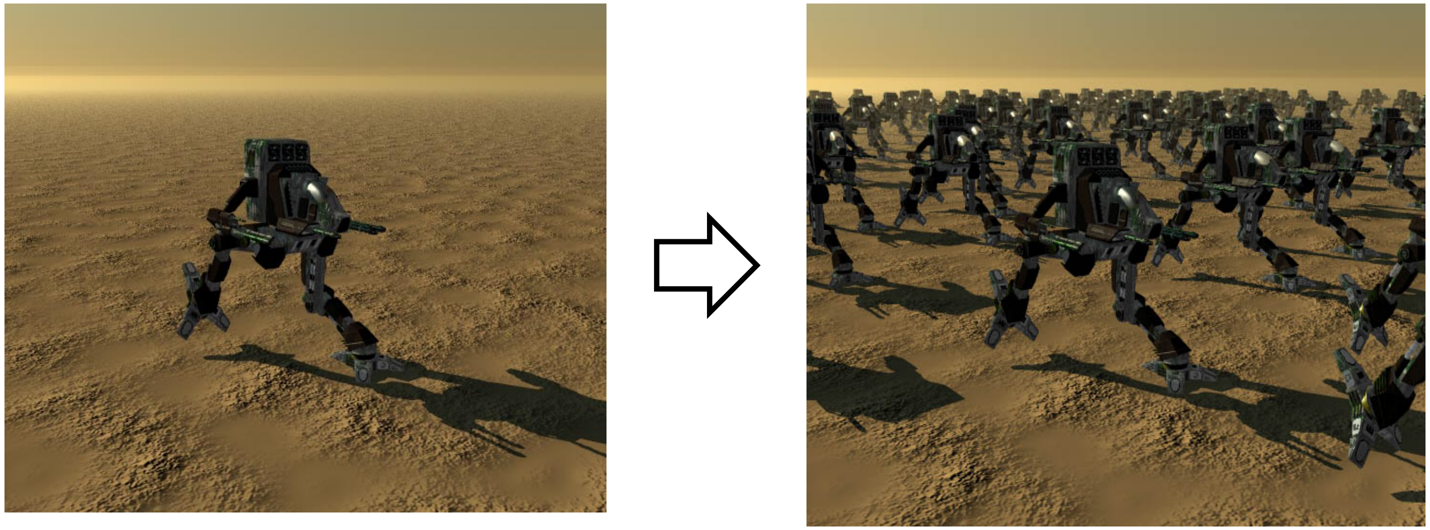




- Exercises
  - Add multiple 3D models to the current scene
  - Rotate each model with different orientations

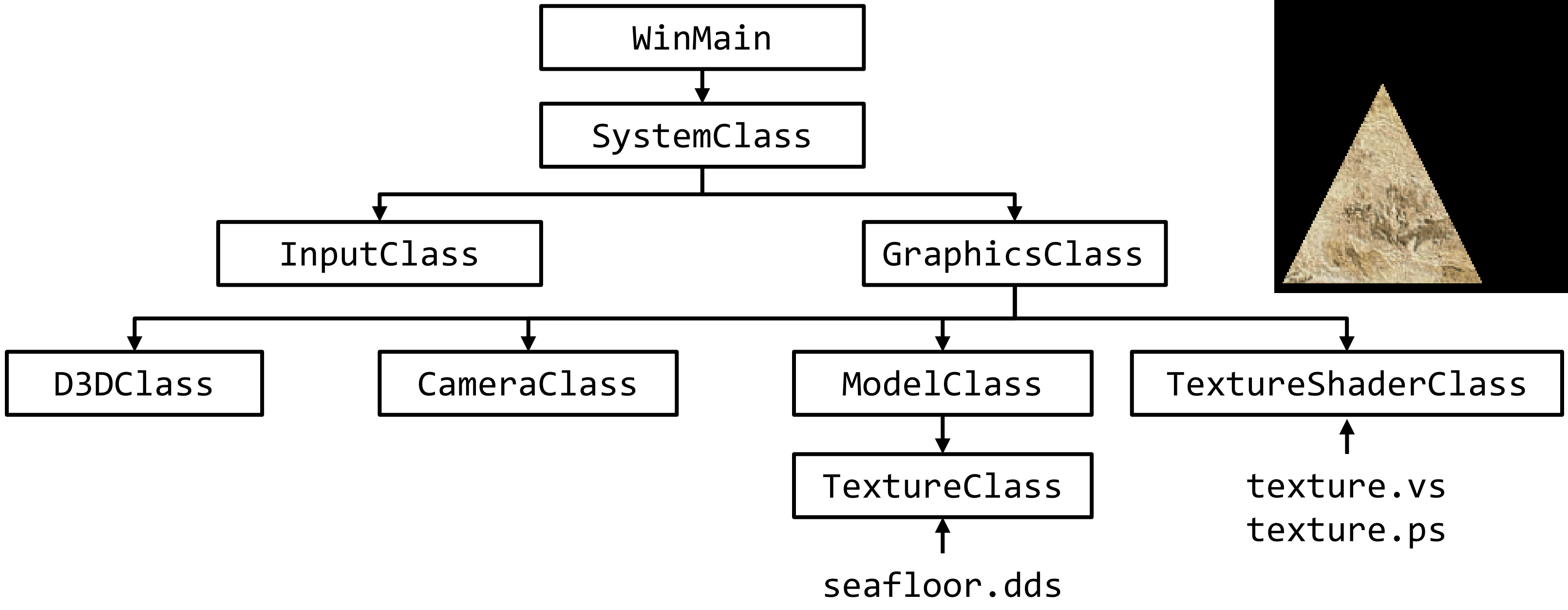
Tutorial: Instancing

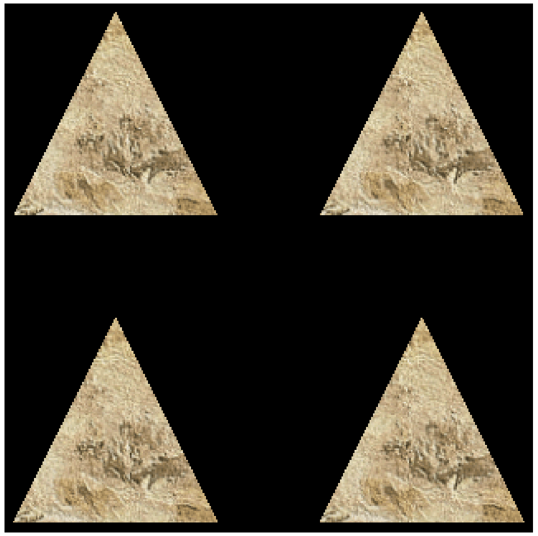
- Mesh Instances
  - Render multiple copies of the same geometry with just changes in position, scale, color, etc.
  - Single vertex buffer + **instance buffer**



Tutorial: Instancing

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





Q & A