Graphics Pipeline

CS7GV6 2021/2022

Lecturer:

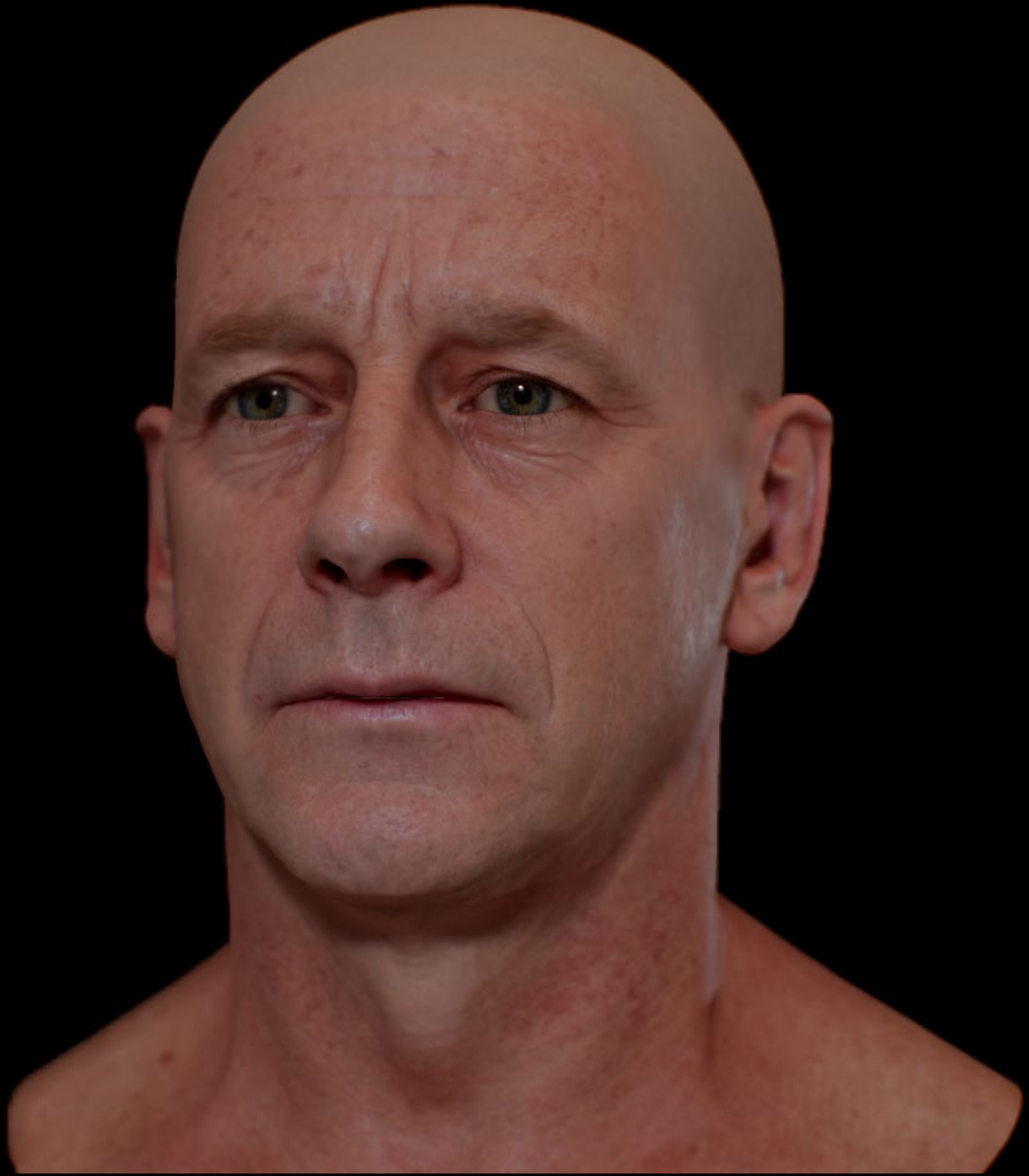
Prof. Carol O’Sullivan, carol.osullivan@tcd.ie

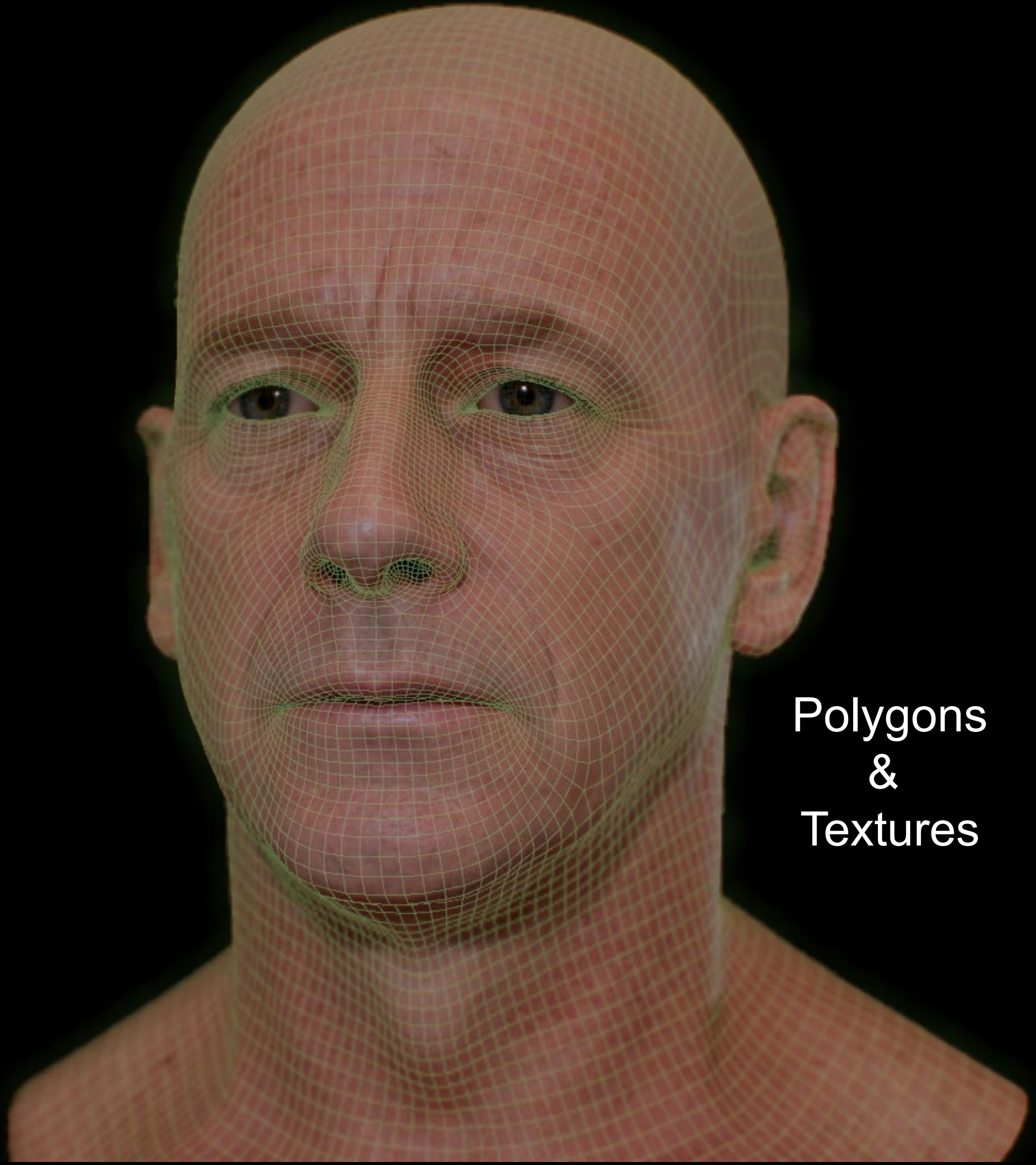
Demonstrator:

Bharat Vyas, vyasb@tcd.ie

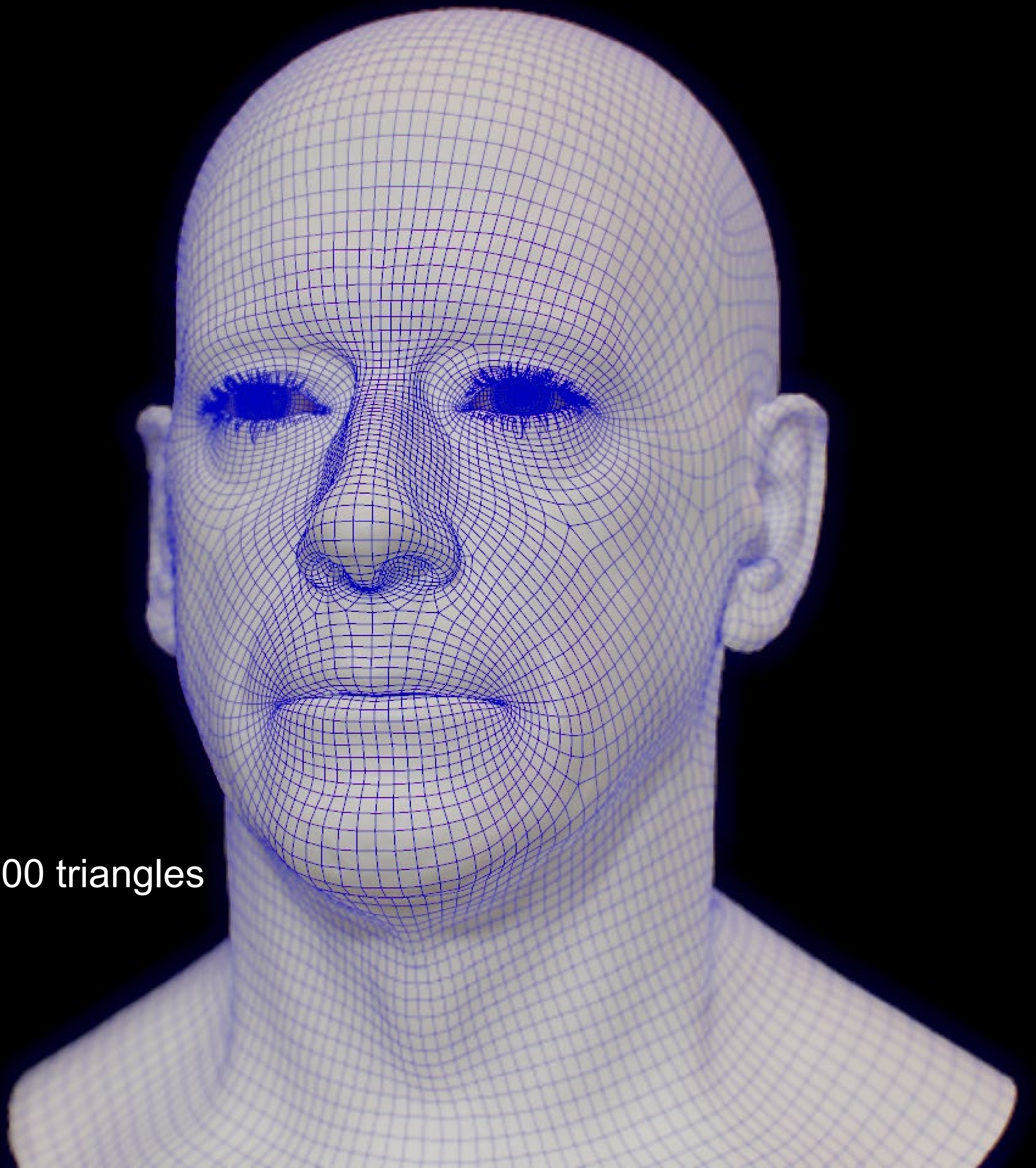
Course Content: Blackboard

3D Model





Polygons
&
Textures



24,800 triangles

Polygons

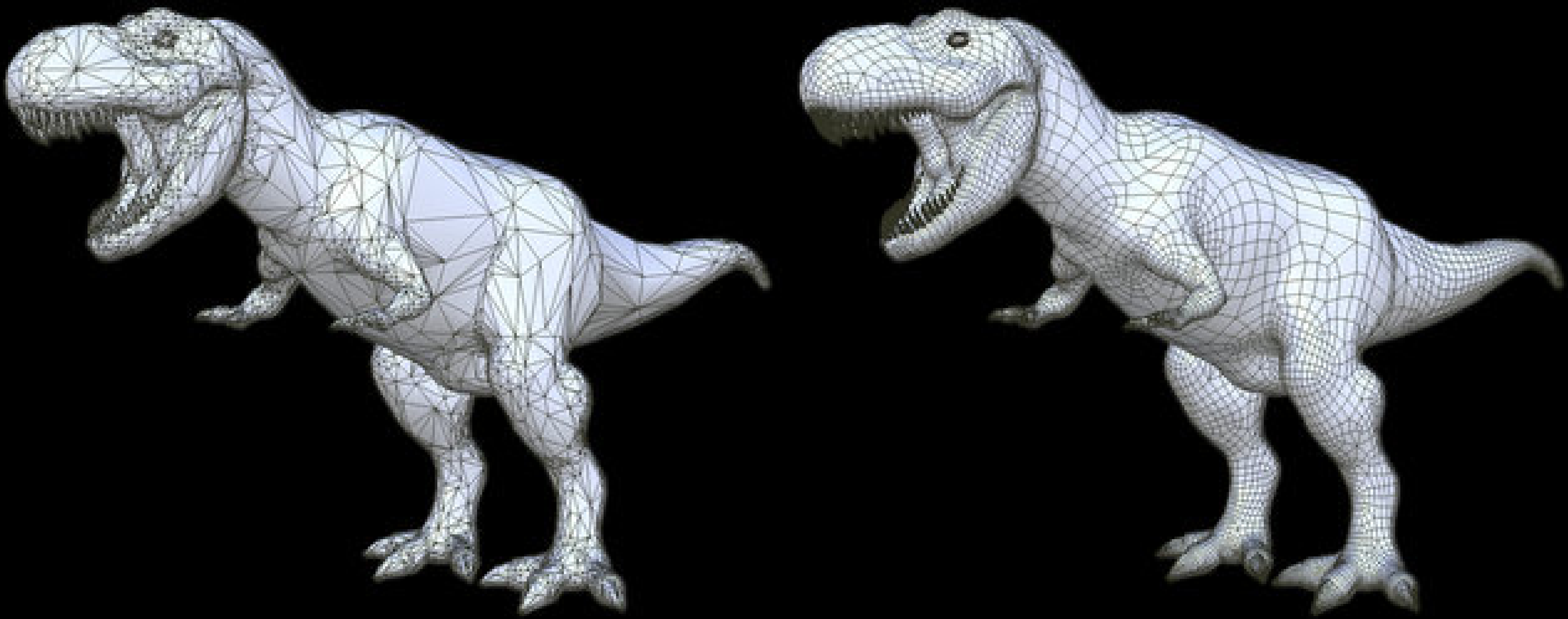
Triangle △

Quadrilateral □

Pentagon ⬠

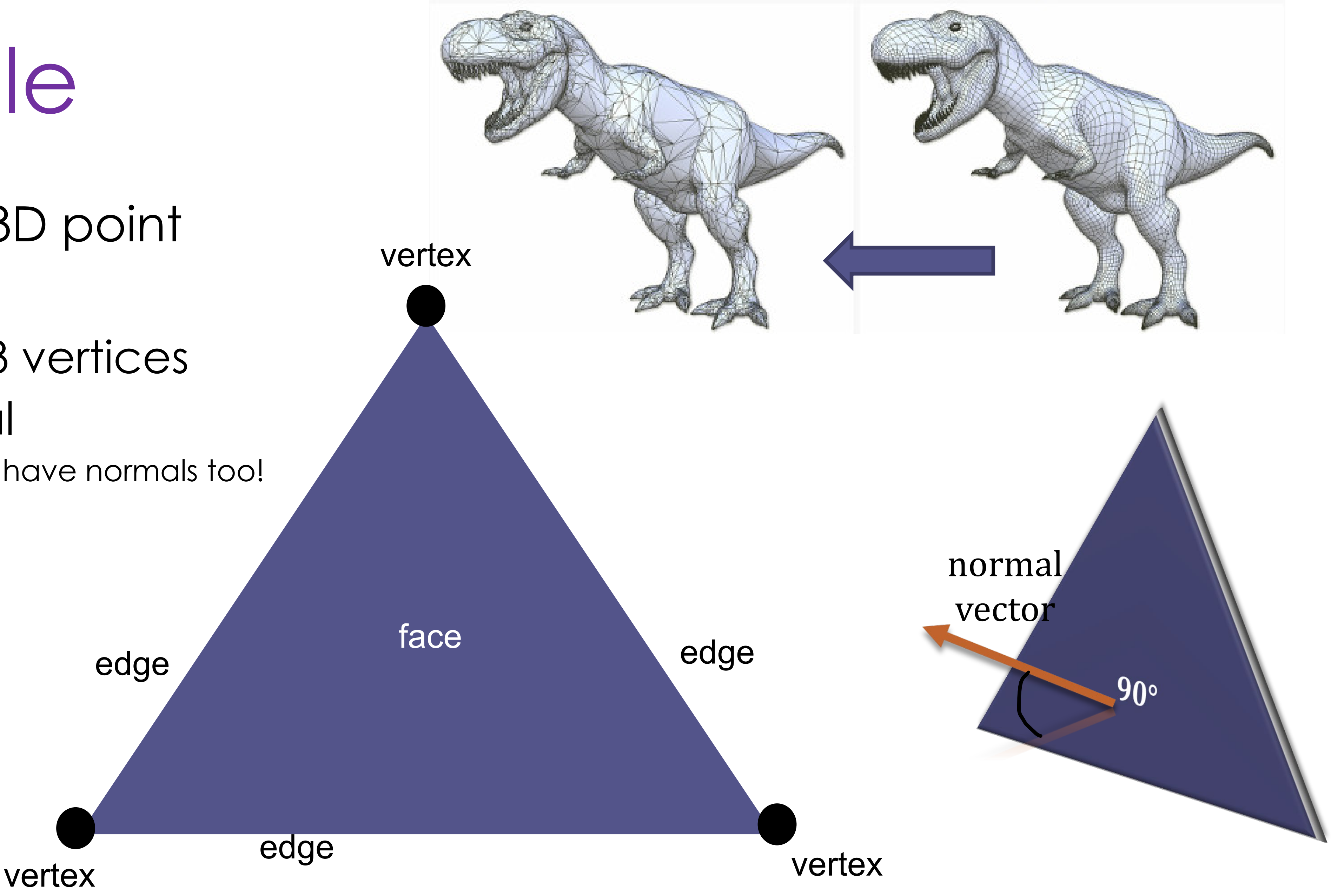
Hexagon ⬡

Triangulate



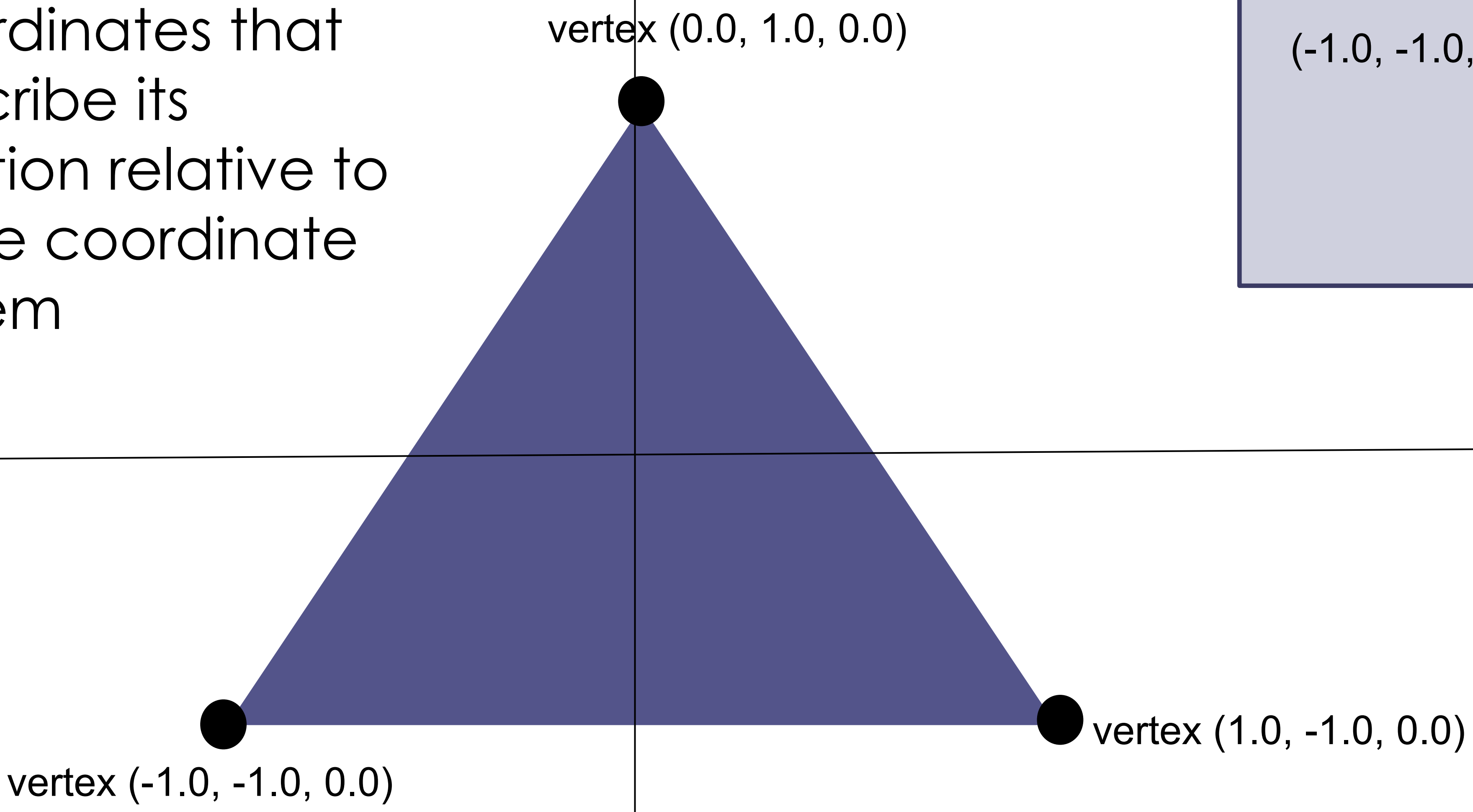
Triangle

- A vertex is a 3D point
- A triangle:
  - Made from 3 vertices
  - Has a normal
  - Note: vertices can have normals too!



Vertex buffer

- A vertex has 3 coordinates that describe its position relative to some coordinate system



VERTEX BUFFER

(0.0, 1.0, 0.0)

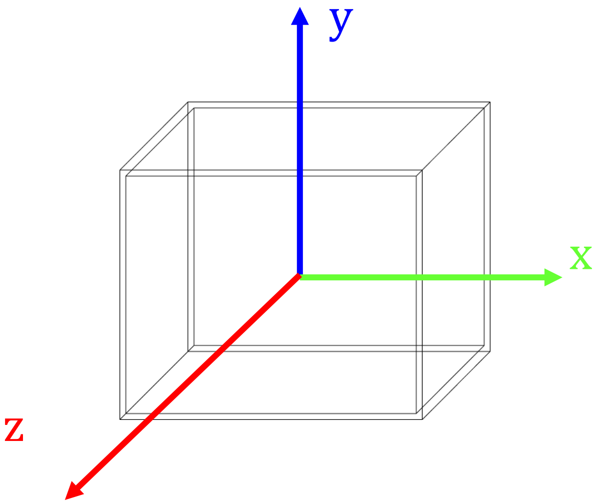
(1.0, -1.0, 0.0)

(-1.0, -1.0, 0.0)

Sources of 3D data

Directly specify the Three-Dimensional data

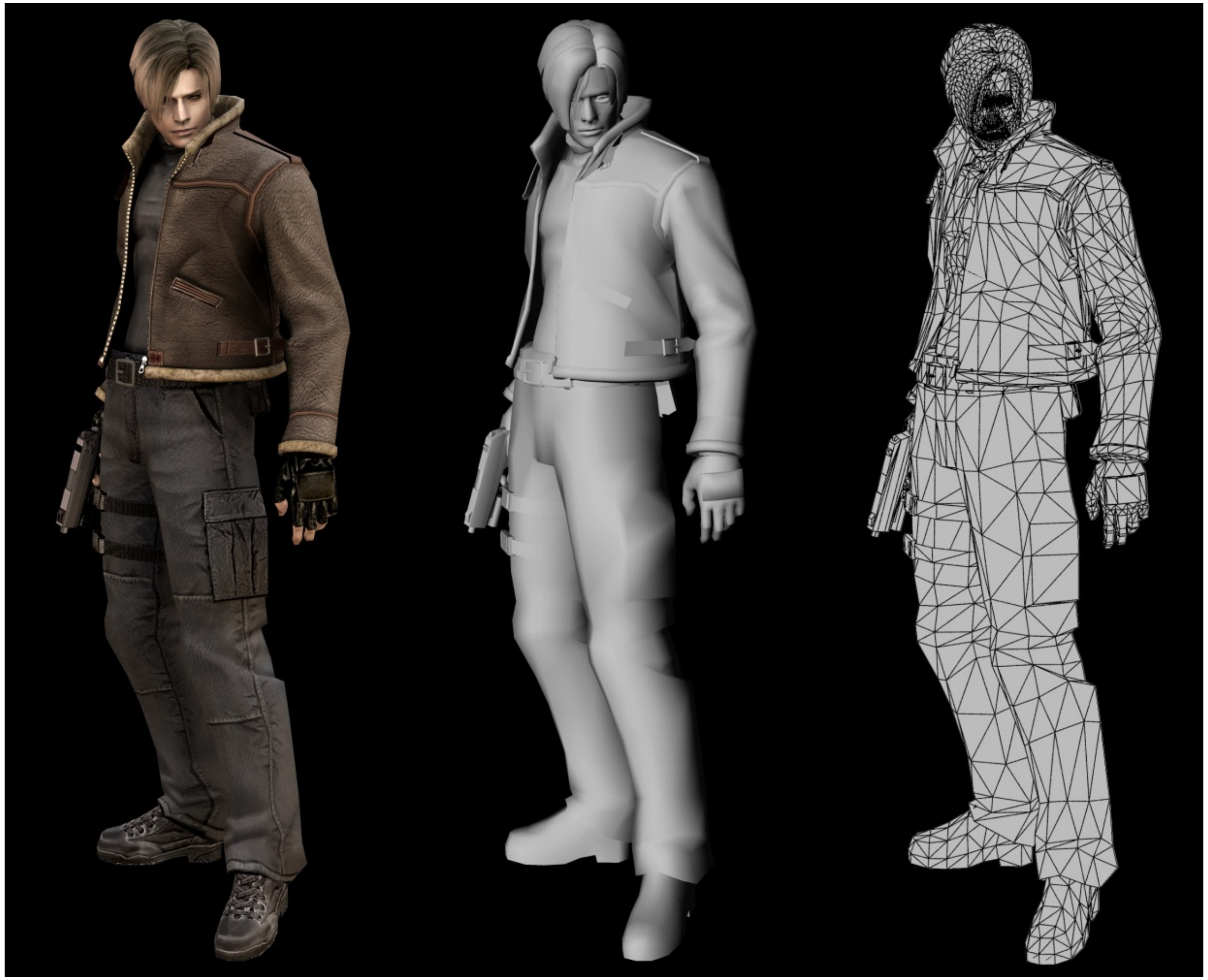
Fine for this:



=

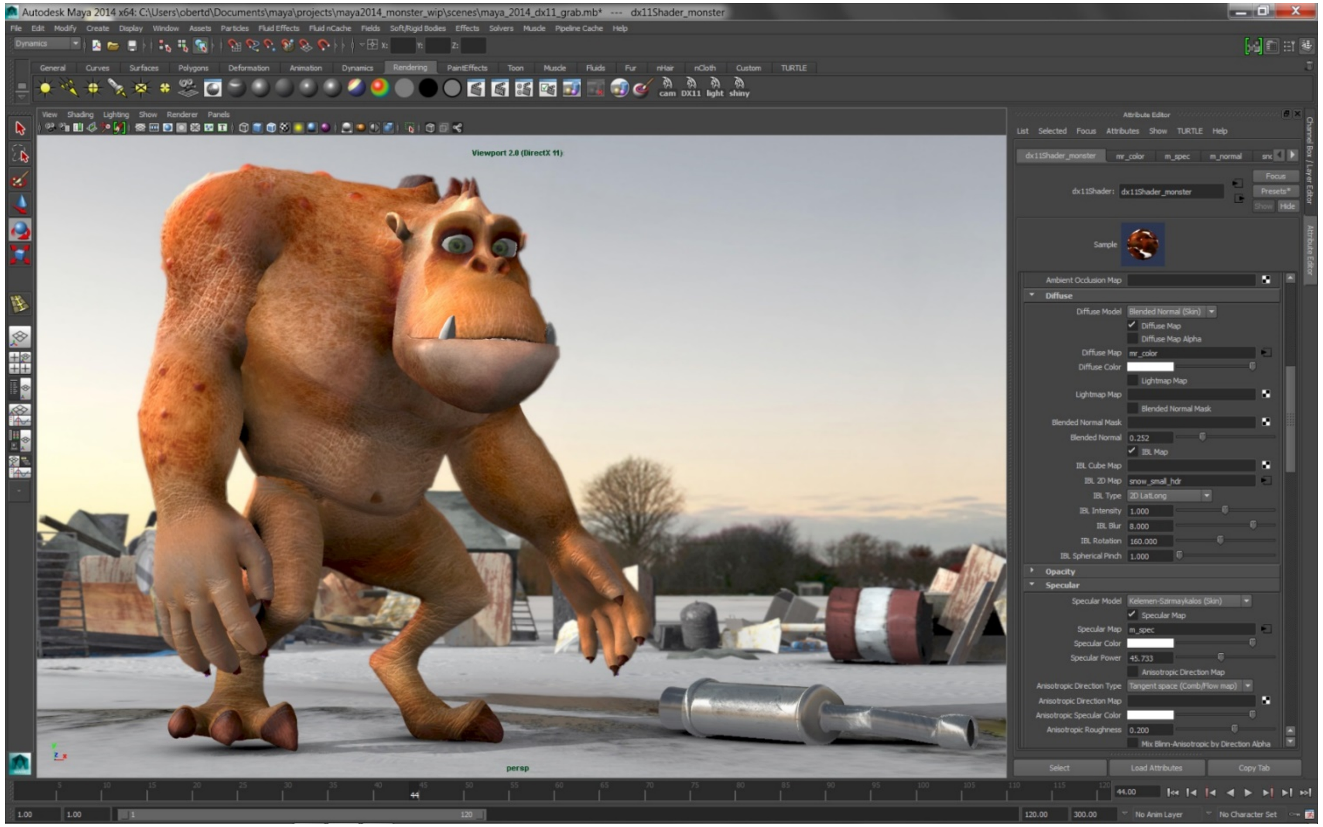
(-1, -1, -1)
(1, -1, -1)
(1, 1, -1)
(-1, 1, -1)
(-1, -1, 1)
(1, -1, 1)
(1, 1, 1)
(-1, 1, 1)

... But not for this!



Modelling Program

- 3ds Max, Maya, Softimage, Blender, Auto CAD, Mudbox, etc.


File Edit Modify Create Display Window Assets Particles Fluid Effects Fluid nCache Fields Soft/Rigid Bodies Effects Solvers Muscle Pipeline Cache Help
Dynamics
General Curves Surfaces Polygons Deformation Animation Dynamics Rendering PaintEffects Toon Muscle Fluids Fur nHair nCloth Custom TURTLE
cam DX11 light shiny
View Shading Lighting Show Renderer Panels
Viewport 2.0 (DirectX 11)
persp
Attribute Editor
List Selected Focus Attributes Show TURTLE Help
dx11Shader_monster mr_color m_spec m_normal snc
Focus
Presets*
Show Hide
Sample
Channel Box / Layer Editor
Attribute Editor
Select Load Attributes Copy Tab
No Anim Layer No Character Set

Autodesk Maya 2014 x64: C:\Users\obertd\Documents\maya\projects\maya2014_monster_wip\scenes\maya_2014_dx11_grab.mb* --- dx11Shader_monster

dx11Shader: dx11Shader_monster

Ambient Occlusion Map

Diffuse

Diffuse Model: Blended Normal (Skin)

☑ Diffuse Map

□ Diffuse Map Alpha

Diffuse Map: mr_color

Diffuse Color

□ Lightmap Map

Lightmap Map

□ Blended Normal Mask

Blended Normal Mask

Blended Normal: 0.252

☑ IBL Map

IBL Cube Map

IBL 2D Map: snow_small_hdr

IBL Type: 2D LatLong

IBL Intensity: 1.000

IBL Blur: 8.000

IBL Rotation: 160.000

IBL Spherical Pinch: 1.000

Opacity

Specular

Specular Model: Kelemen-Szirmaykalos (Skin)

☑ Specular Map

Specular Map: m_spec

Specular Color

Specular Power: 45.733

□ Anisotropic Direction Map

Anisotropic Direction Type: Tangent space (Comb/Flow map)

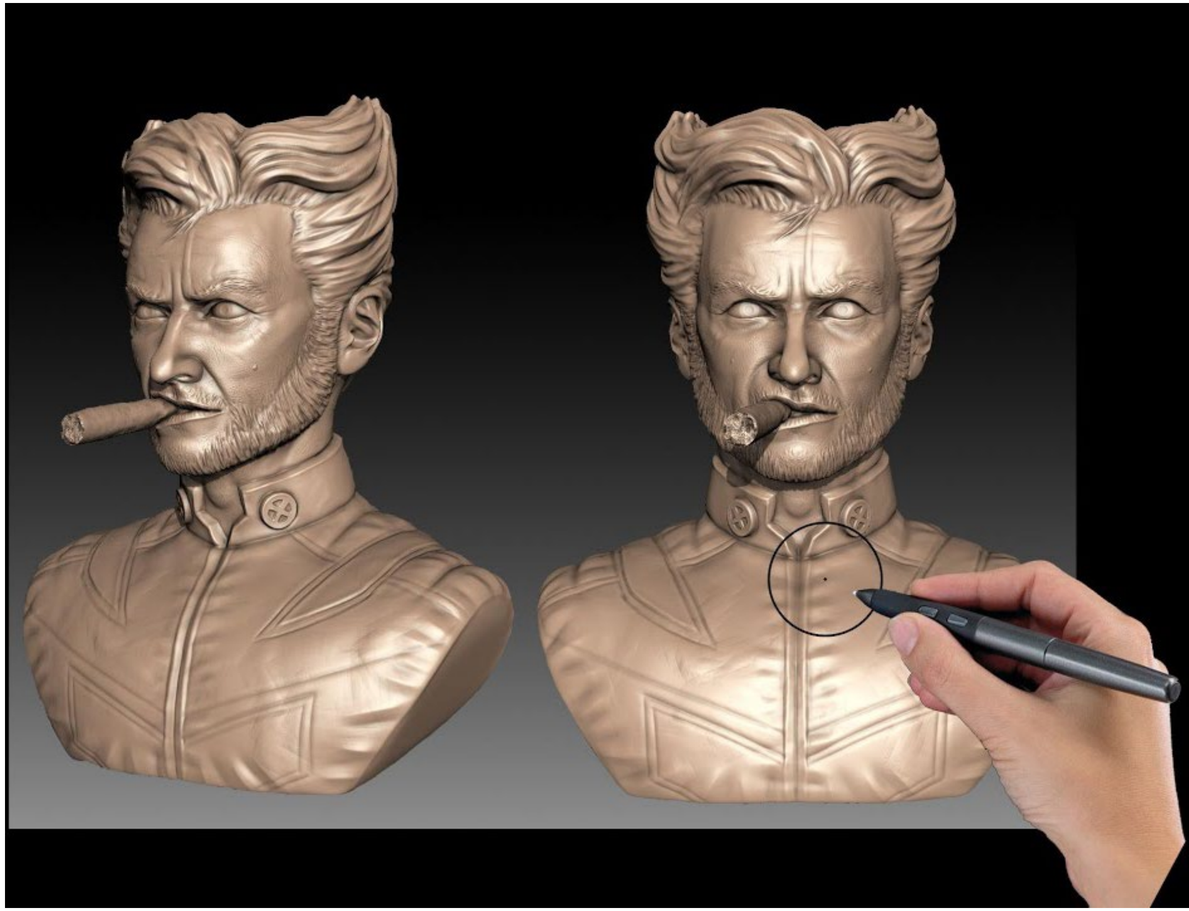
Anisotropic Direction Map

Anisotropic Specular Color

Anisotropic Roughness: 0.200

□ Mix Blinn-Anisotropic by Direction Alpha

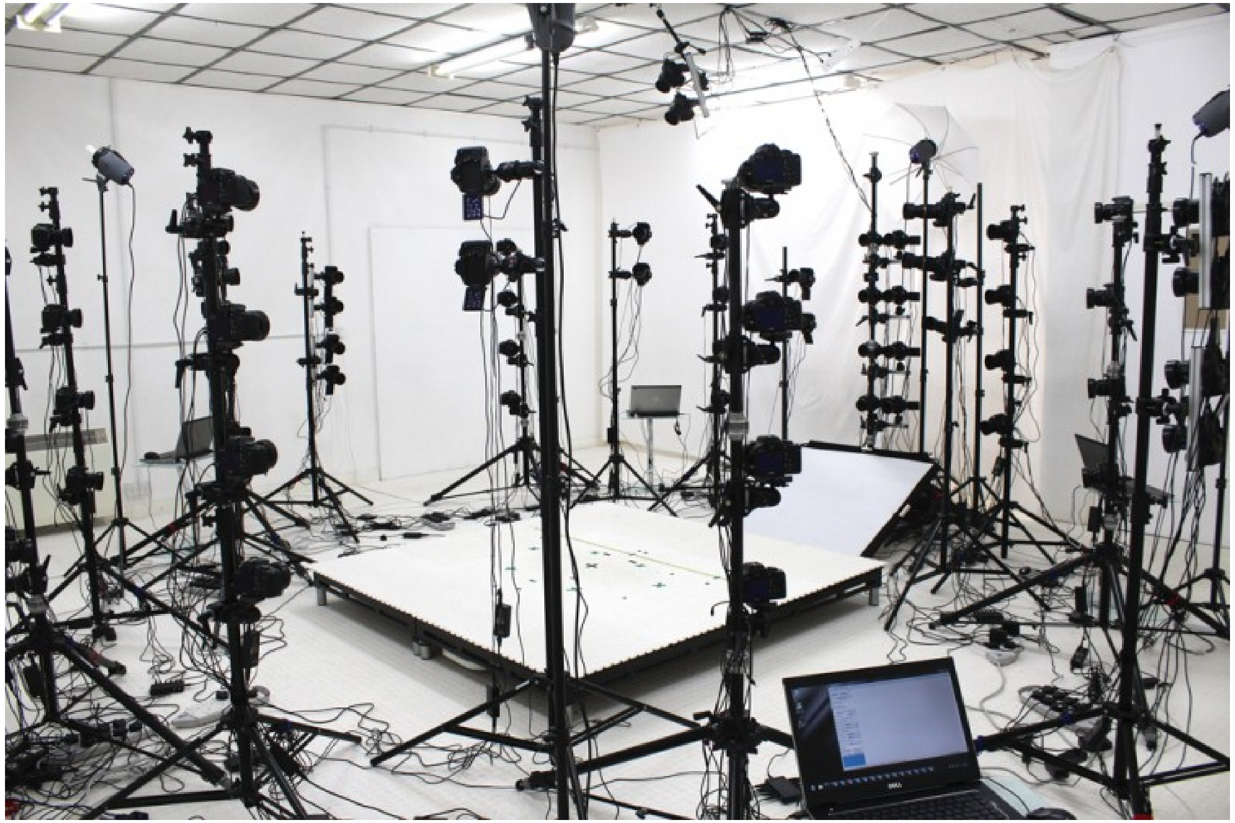
44.00 | 1.00 | 1.00 | 1 | 120 | 120.00 | 300.00



https://www.autodesk.com/education/free-software/all

Scanning Technologies

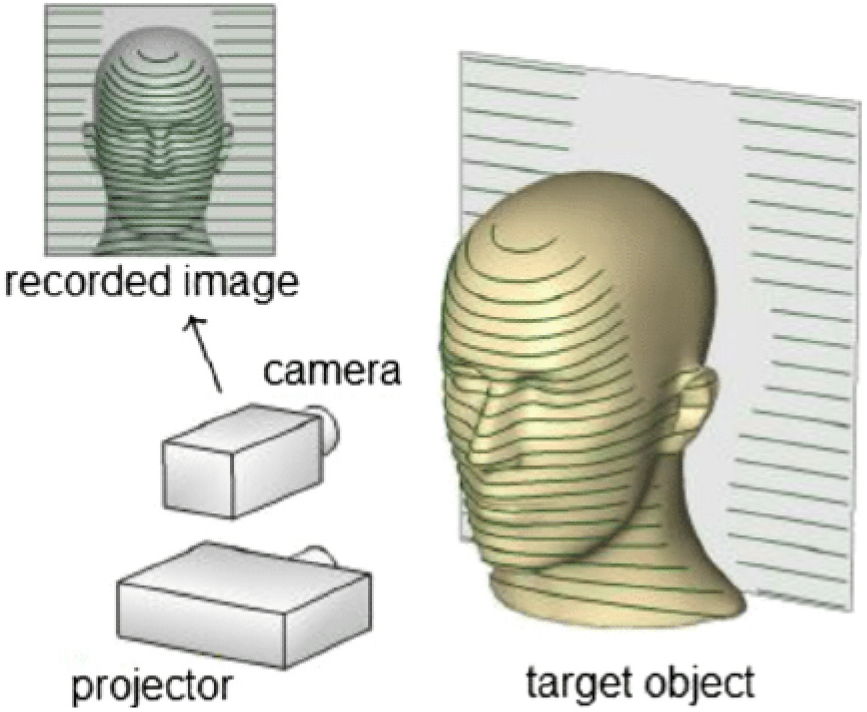
- 1. Photogrammetry



https://www.capturingreality.com/
https://itseez3d.com/

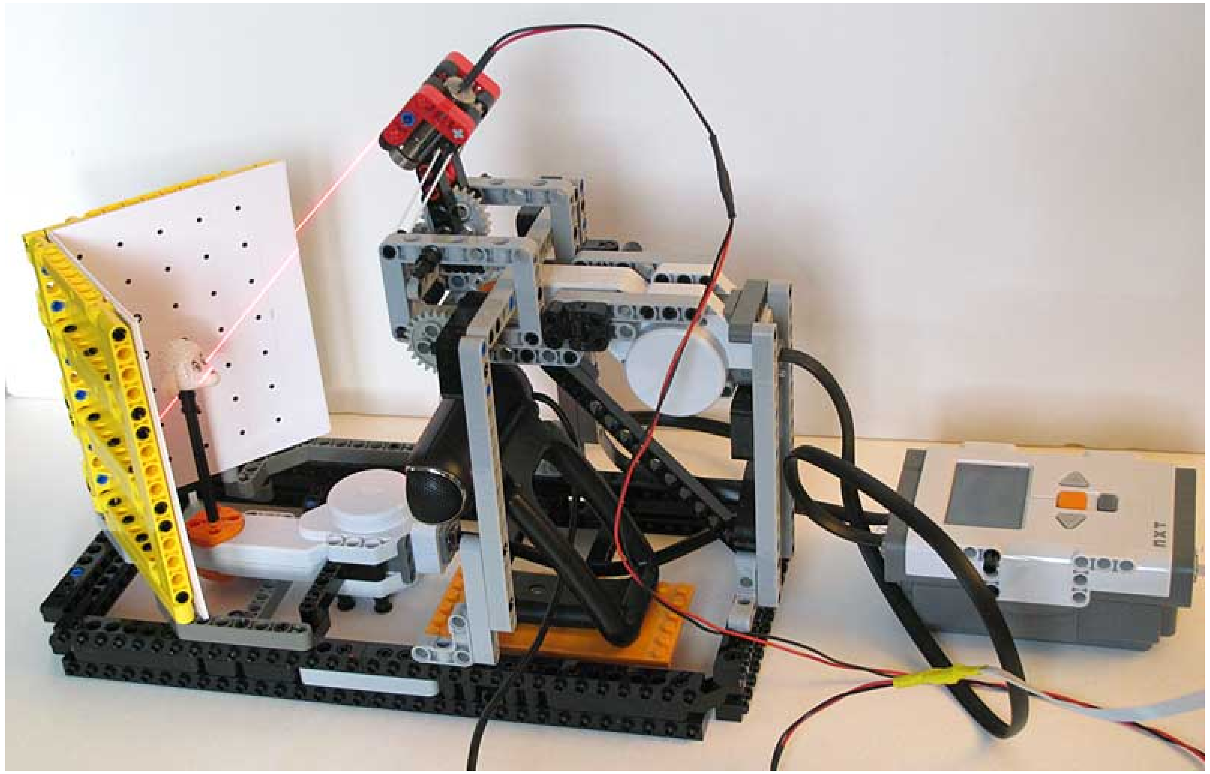
Scanning Technologies

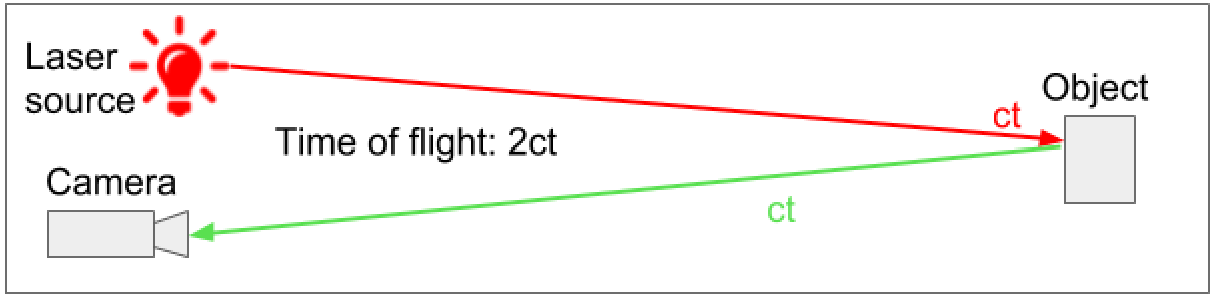
- 2. Structured light scanning



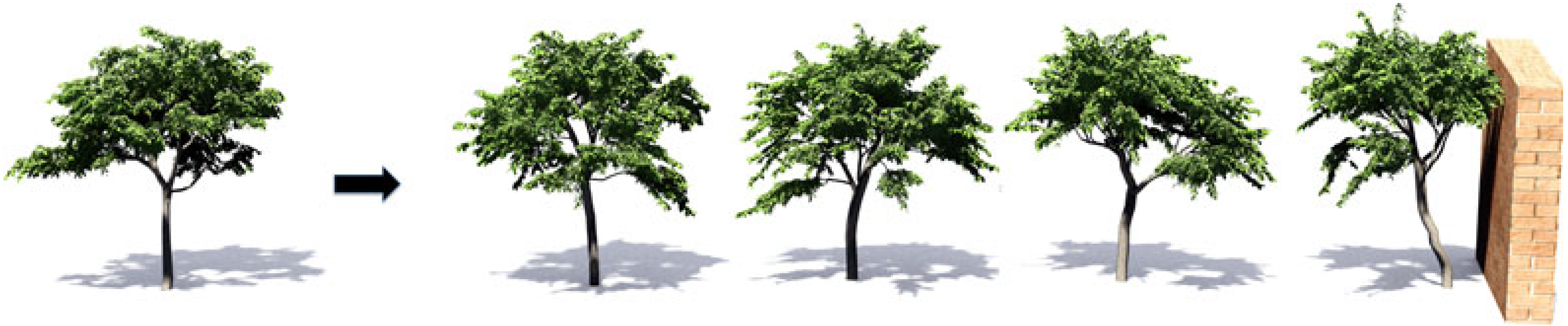
Scanning Technologies

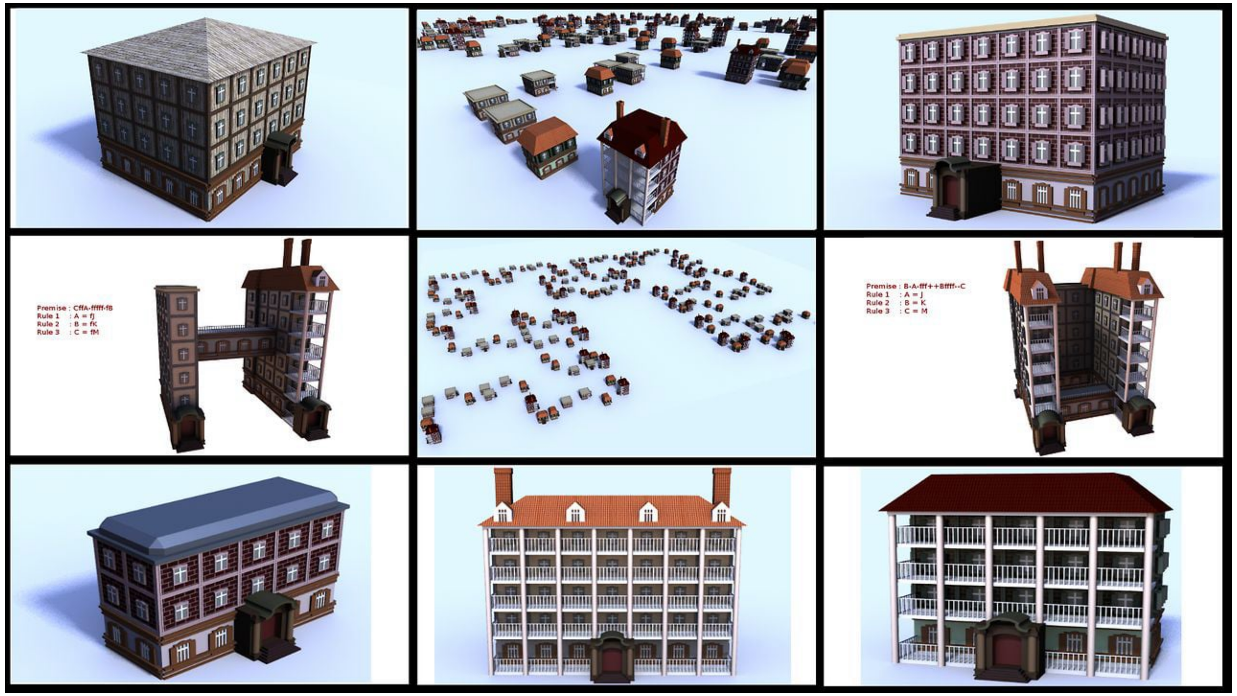
- 3. Laser scanning

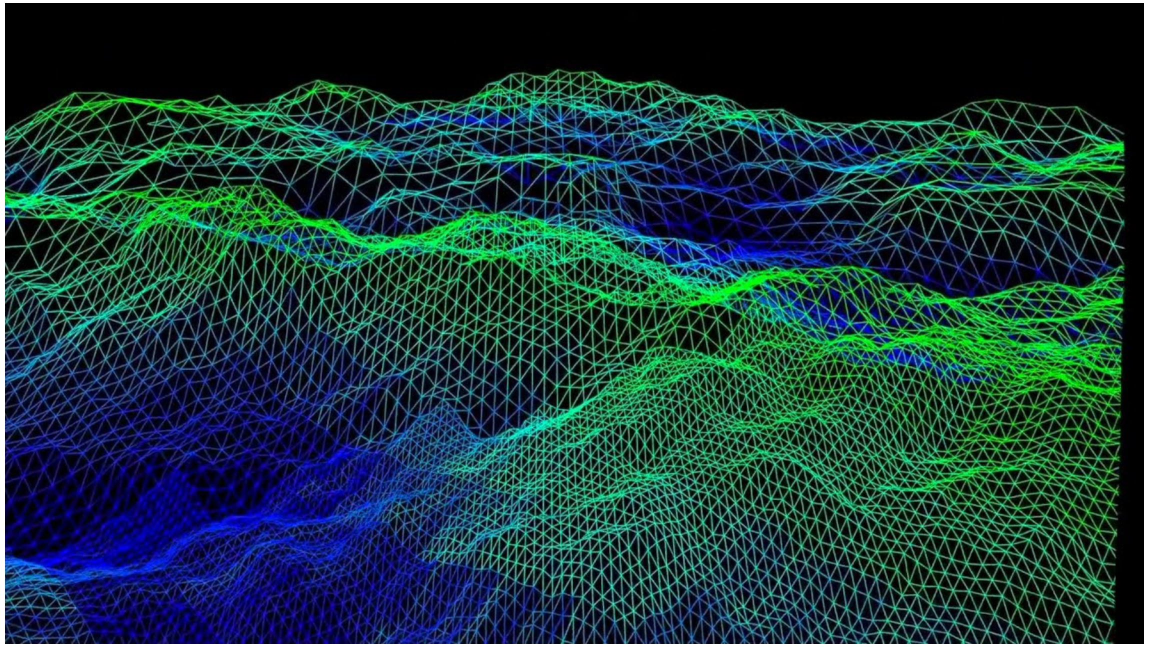




Procedural Models



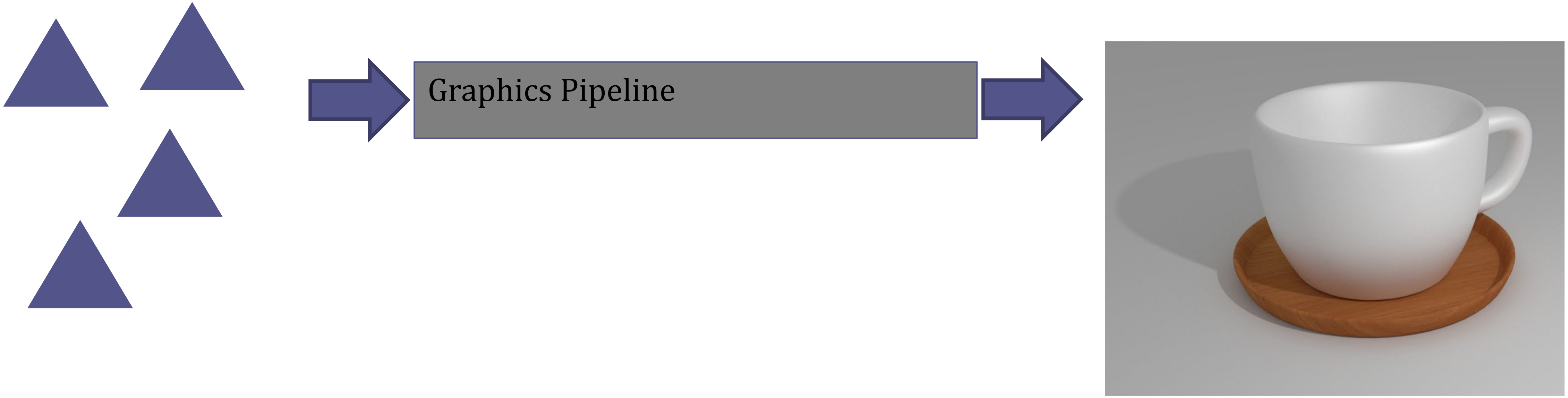




Algorithmic rules to generate complex models

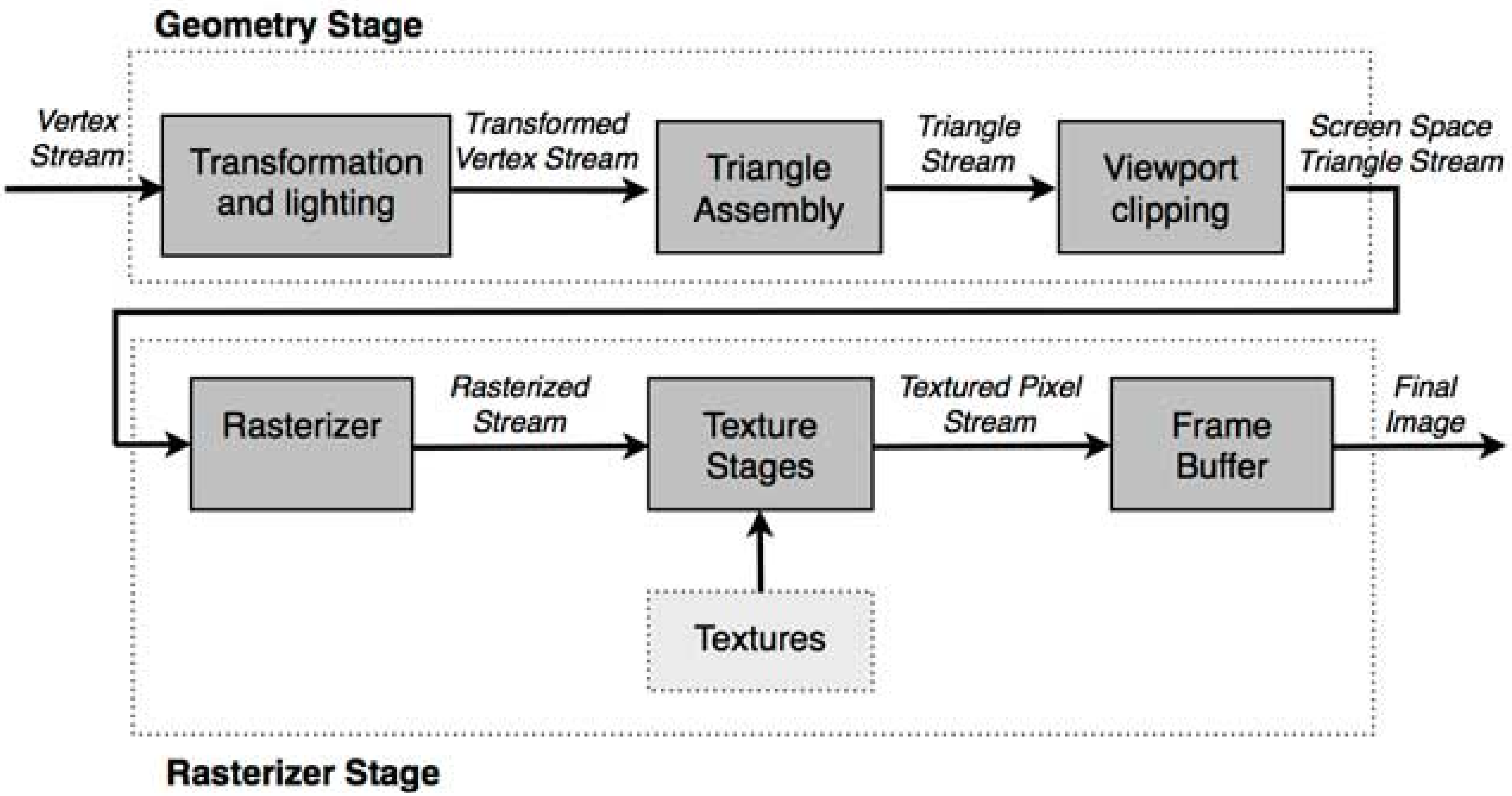
Rendering

- *Rendering* is the process by which a computer creates images from models or objects.
- The final rendered image consists of pixels drawn on the screen



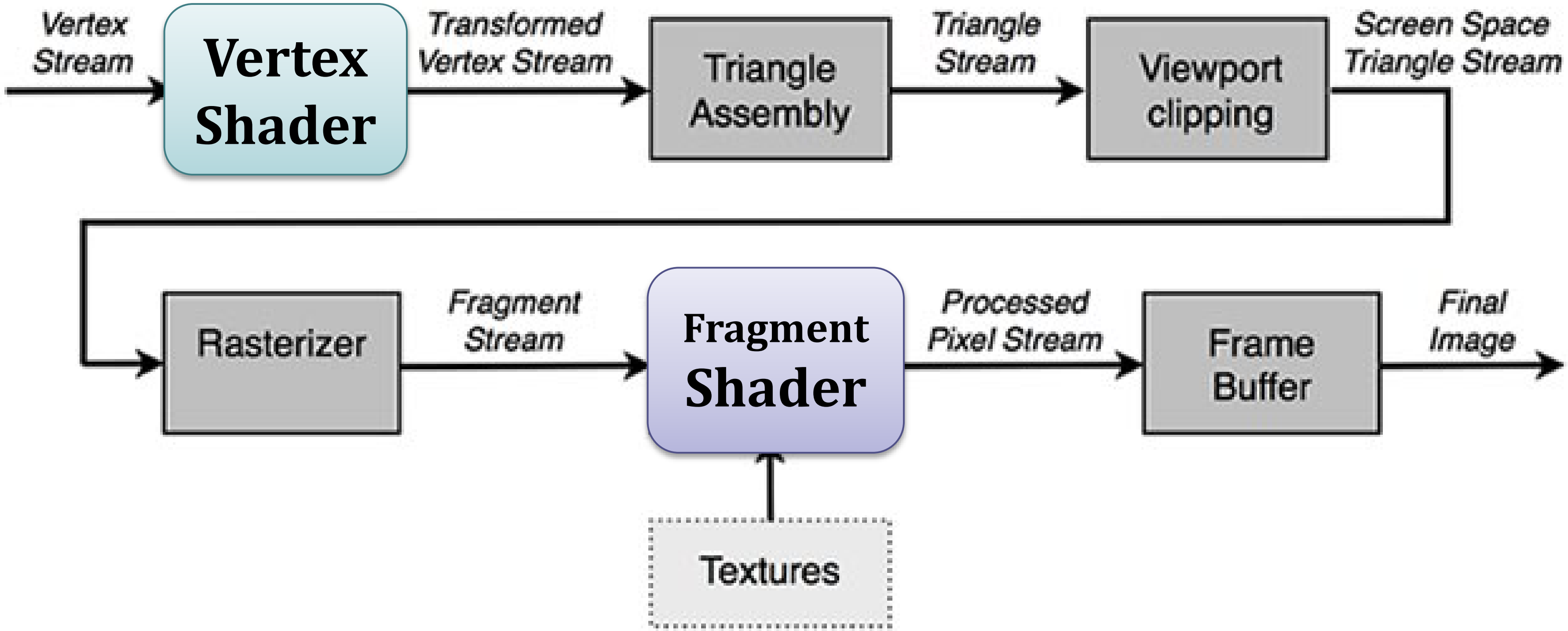
Graphics Pipeline

Fixed Function Pipeline



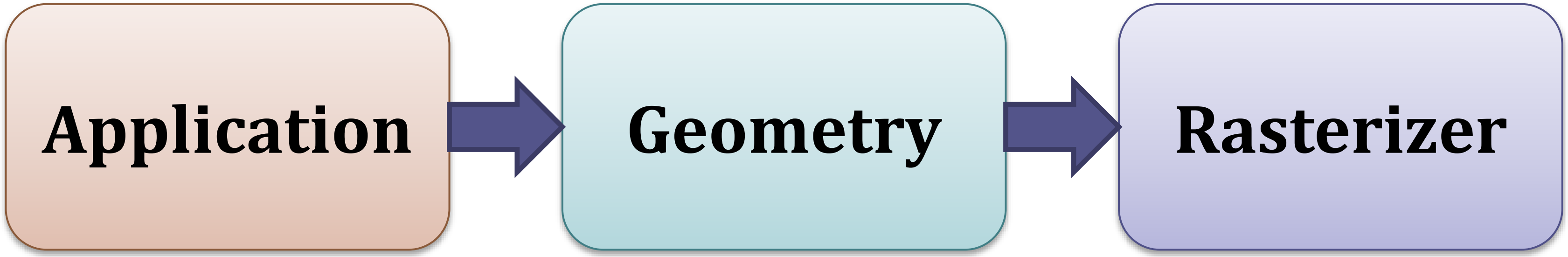
http://www.adobe.com/devnet/flashplayer/articles/how-stage3d-works.html

Graphics Programmable Pipeline



Graphics Pipeline Overview

- Coarse Division
- Each stage is a pipeline in itself

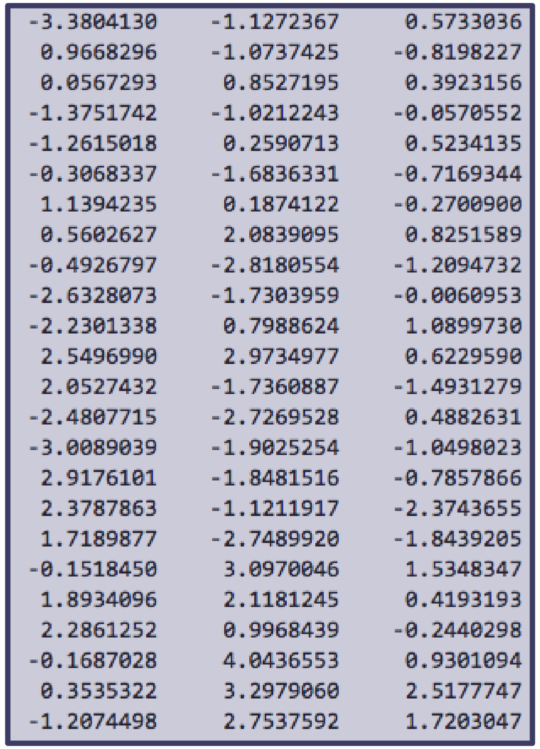


- The slowest pipeline stage determines the *rendering speed (fps)*

Real-Time Rendering, 3rd Edition, Akenine-Moller

The Application Stage

VERTEX BUFFER

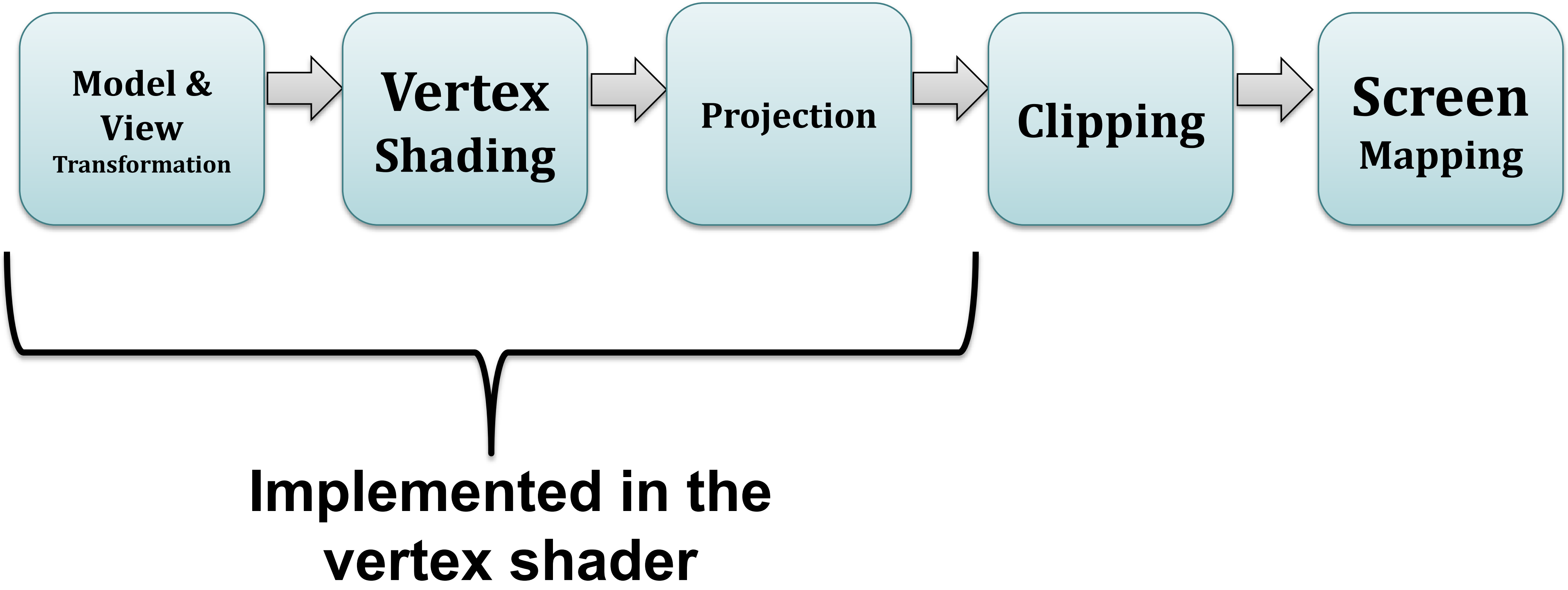


```
-3.3804130   -1.1272367    0.5733036
 0.9668296   -1.0737425   -0.8198227
 0.0567293    0.8527195    0.3923156
-1.3751742   -1.0212243   -0.0570552
-1.2615018    0.2590713    0.5234135
-0.3068337   -1.6836331   -0.7169344
 1.1394235    0.1874122   -0.2700900
 0.5602627    2.0839095    0.8251589
-0.4926797   -2.8180554   -1.2094732
-2.6328073   -1.7303959   -0.0060953
-2.2301338    0.7988624    1.0899730
 2.5496990    2.9734977    0.6229590
 2.0527432   -1.7360887   -1.4931279
-2.4807715   -2.7269528    0.4882631
-3.0089039   -1.9025254   -1.0498023
 2.9176101   -1.8481516   -0.7857866
 2.3787863   -1.1211917   -2.3743655
 1.7189877   -2.7489920   -1.8439205
-0.1518450    3.0970046    1.5348347
 1.8934096    2.1181245    0.4193193
 2.2861252    0.9968439   -0.2440298
-0.1687028    4.0436553    0.9301094
 0.3535322    3.2979060    2.5177747
-1.2074498    2.7537592    1.7203047
```

- Developer has full control
- Executes on the CPU
- At the end of the application stage, the rendering primitives are fed to the geometry stage

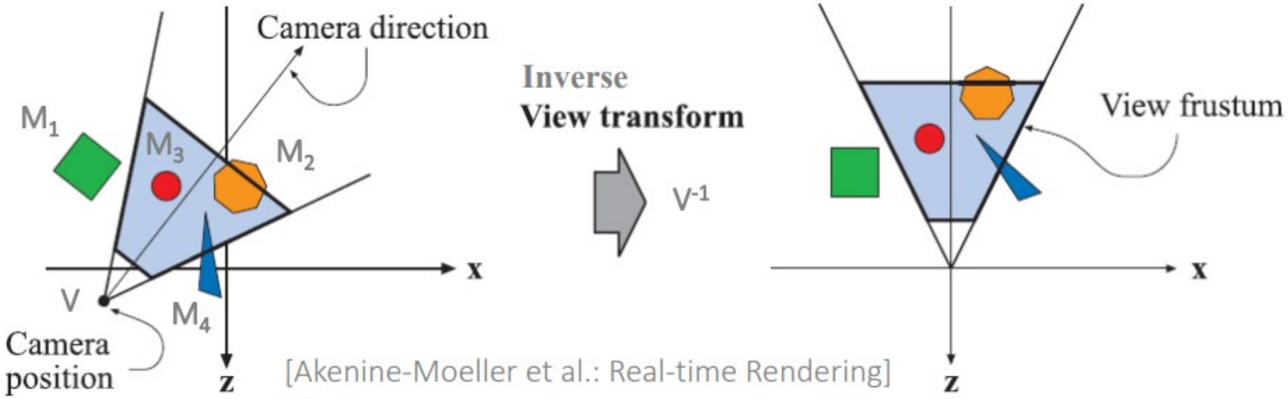
The Geometry Stage

- Responsible for the per-polygon and per-vertex operations



Model & View Transform

- Models are transformed into several *spaces* or *coordinate systems*
- Models initially reside in *model space*
    - i.e. no transformation
- "*Model transform*" positions the object in *world coordinates* or *world space*
- The *view transform* places the camera at the origin and aims it, to make it look in the direction of the negative z-axis

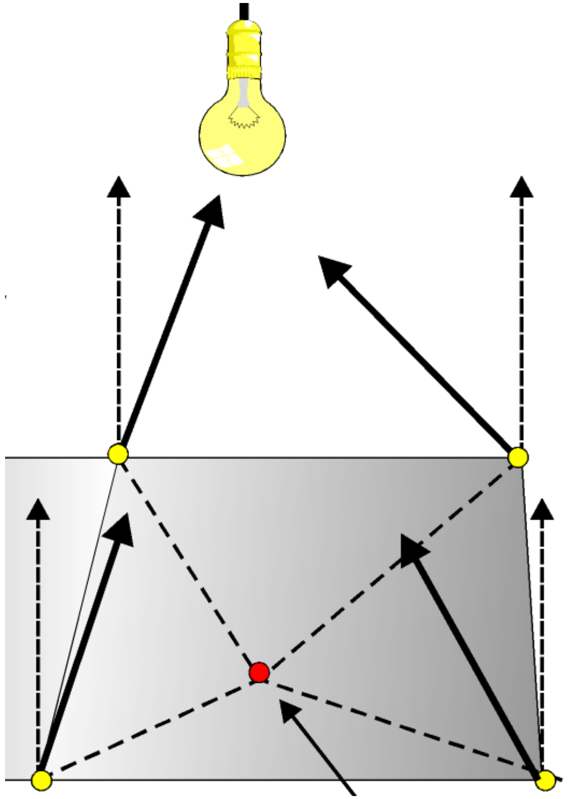


[Akenine-Moeller et al.: Real-time Rendering]

Vertex Shading

Vertex Shading

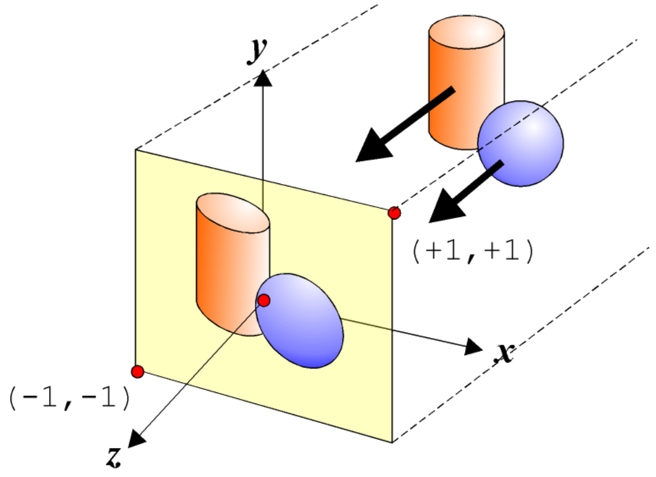
- Shading means determining the effect of a light on a material
- A variety of material data can be stored at each vertex
  - Points location
  - Normal
  - Color
- Vertex shading results (colors, vectors, texture coordinates, or any other kind of shading data) are then sent to the rasterization stage to be interpolated



Projection

Projection

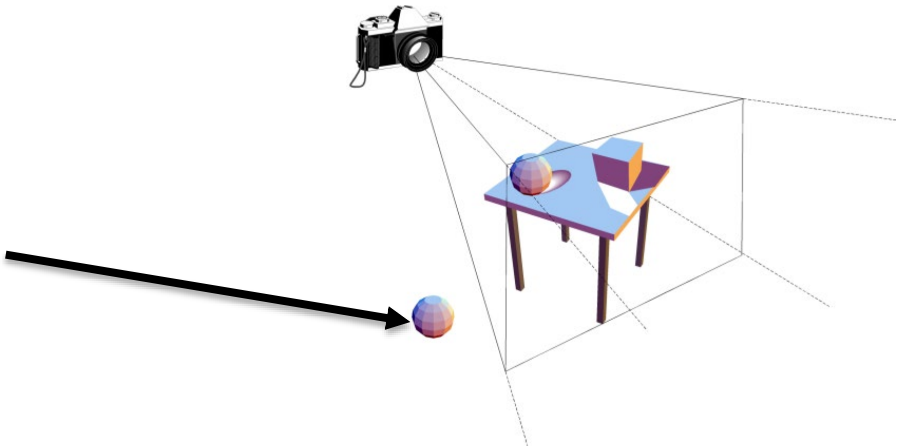
- After shading, rendering systems perform *projection*
- Models are projected from three to two dimensions
- *Perspective* or *orthographic* viewing



Clipping

Clipping

- The computer may have model, texture, and shader data for all objects in the scene in memory
- The virtual camera viewing the scene only “sees” the objects within the field of view
- The computer does not need to transform, texture, and shade the objects that are behind or on the sides of the camera
- A clipping algorithm skips these objects making rendering more efficient

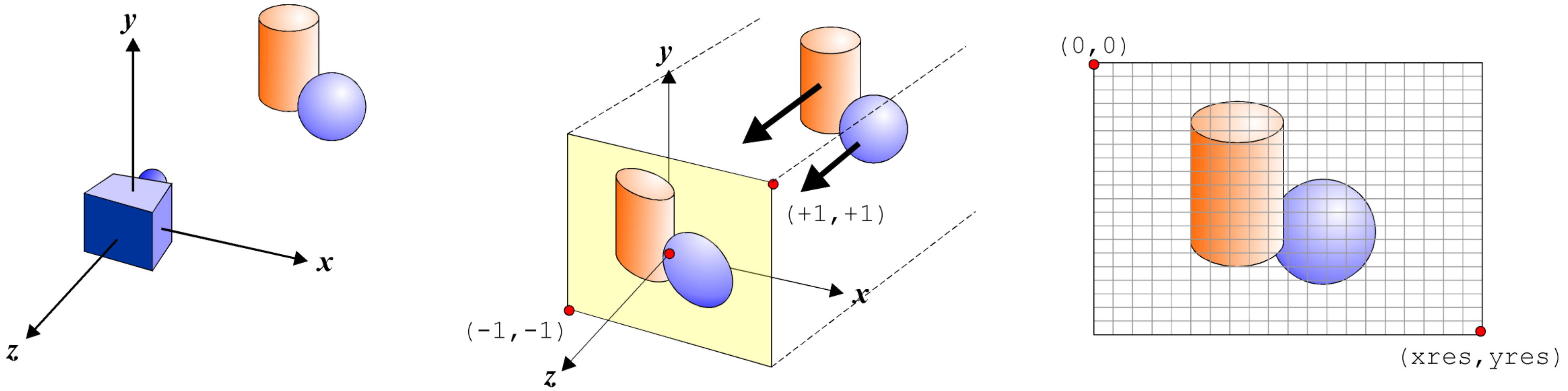


Outside view so must be clipped

Screen
Mapping

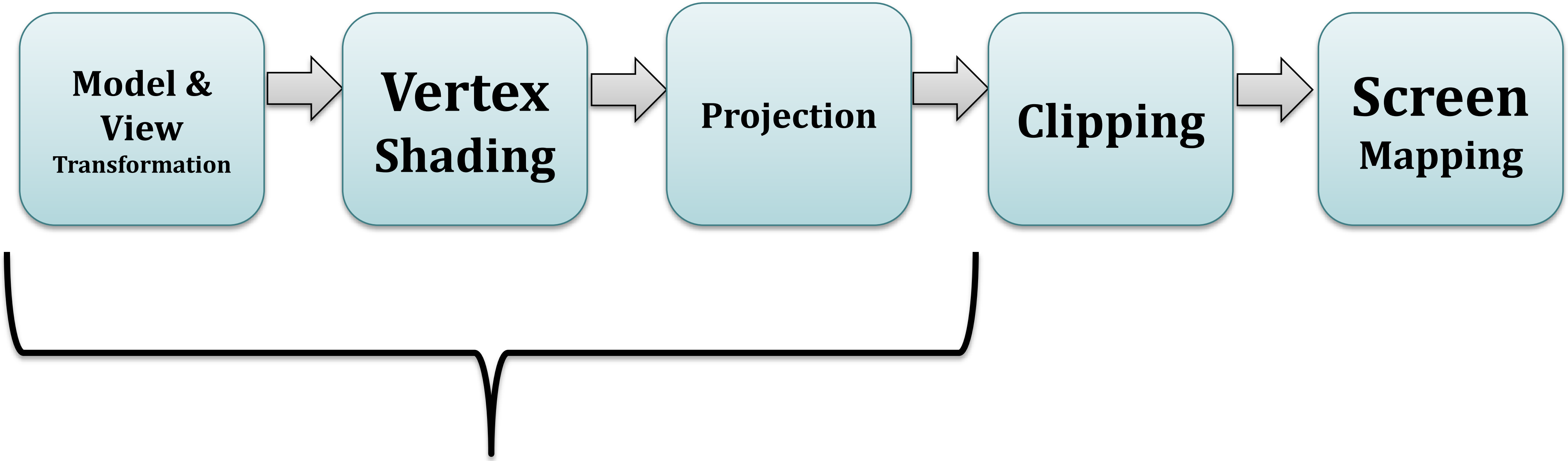
Screen Mapping

- Only the clipped primitives inside the view volume are passed to this stage
- Coordinates are in 3D
- The x- and y-coordinates of each primitive are transformed to the *screen coordinates*



The Geometry Stage

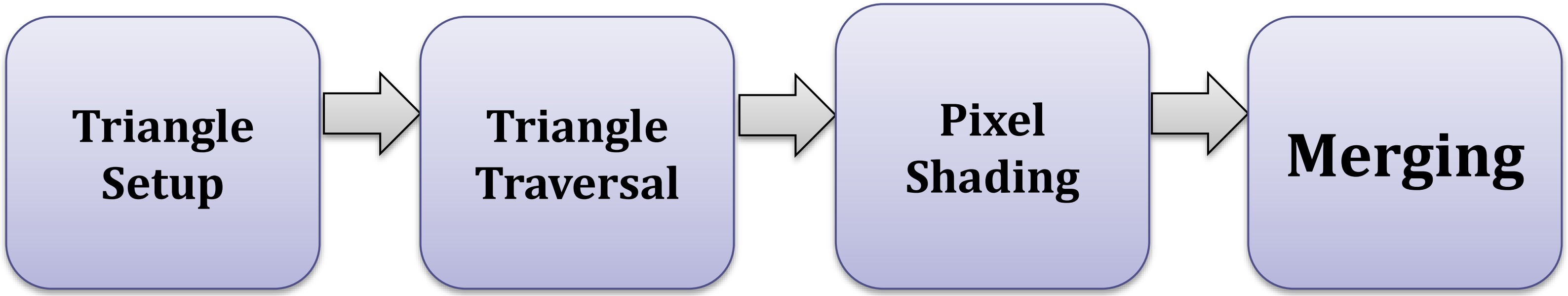
- Responsible for the per-polygon and per-vertex operations



Implemented in the vertex shader

The Rasterizer Stage

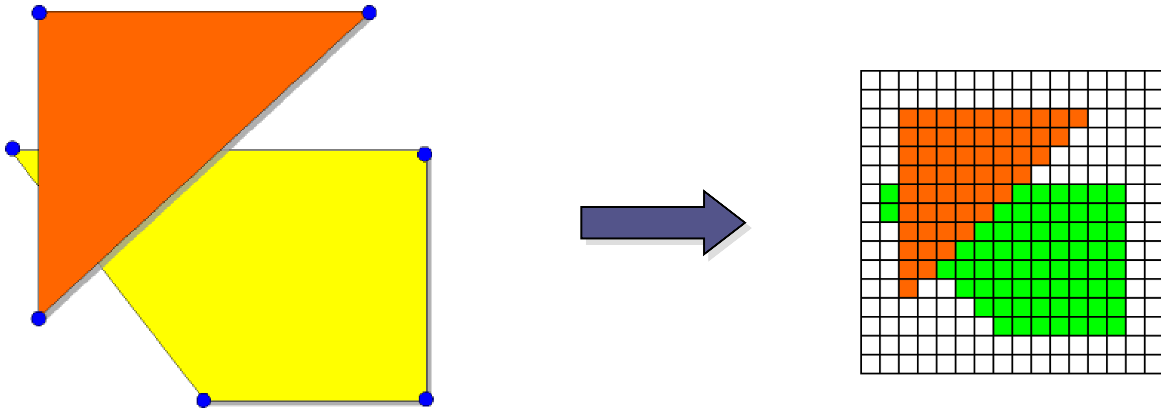
- Given the transformed and projected vertices with their associated shading data (from geometry stage)



- The goal of the rasterizer stage is to compute and set colors for the pixels covered by the object
- *Rasterization:* conversion from 3D vertices in screen-space to pixels on the screen

Triangle Traversal

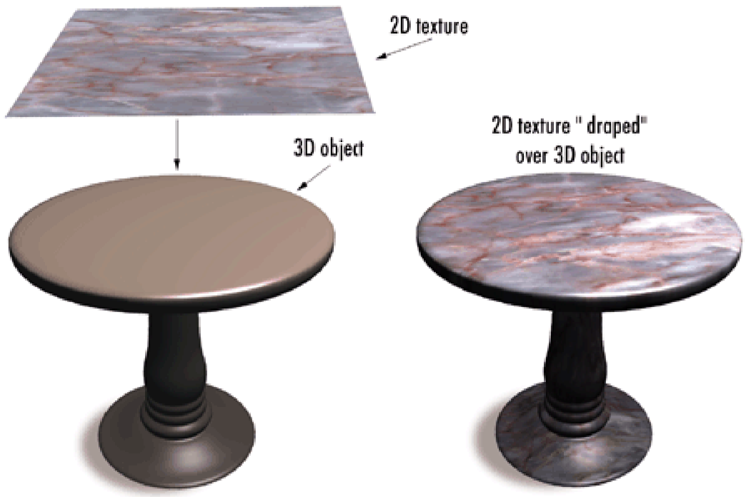
- Which pixels are inside a triangle?
- Each pixel that has its centre covered by the triangle is checked
- A *fragment* is generated for the part of the pixel that overlaps the triangle
- Triangle vertices interpolation



Pixel Shading

Pixel Shading

- Per-pixel shading computations are performed here
- End result is one or more colours to be passed to the next stage
- Executed by programmable GPU cores
- NB: Texturing is employed here



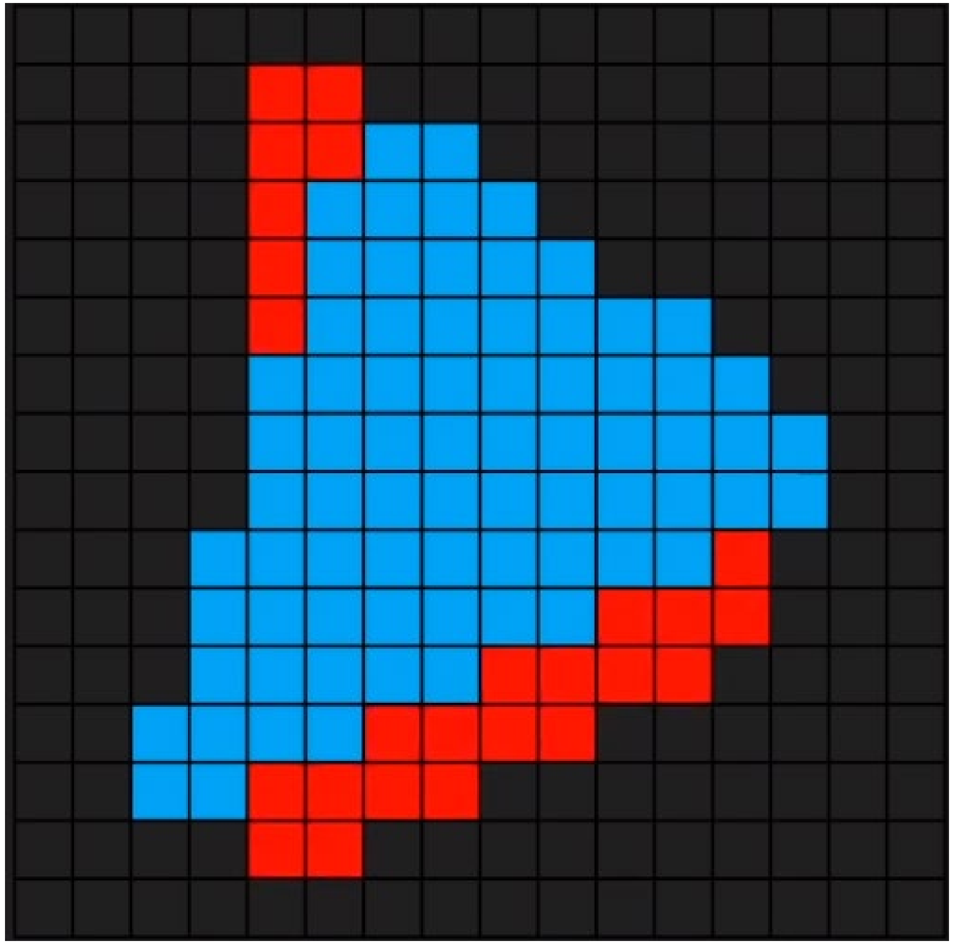
Merging

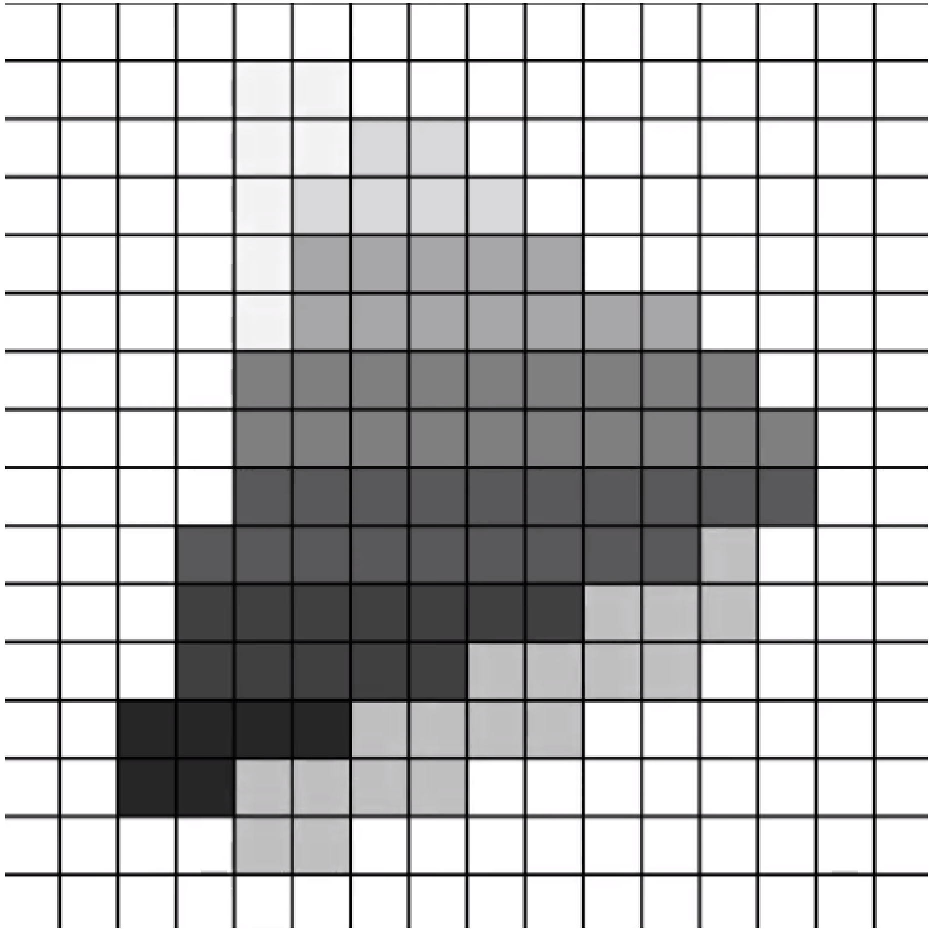
Merging

- Information for each pixel is stored in the *colour buffer* (a rectangular array of colours)
- Combine the fragment colour produced by the shading stage with the colour currently stored in the buffer

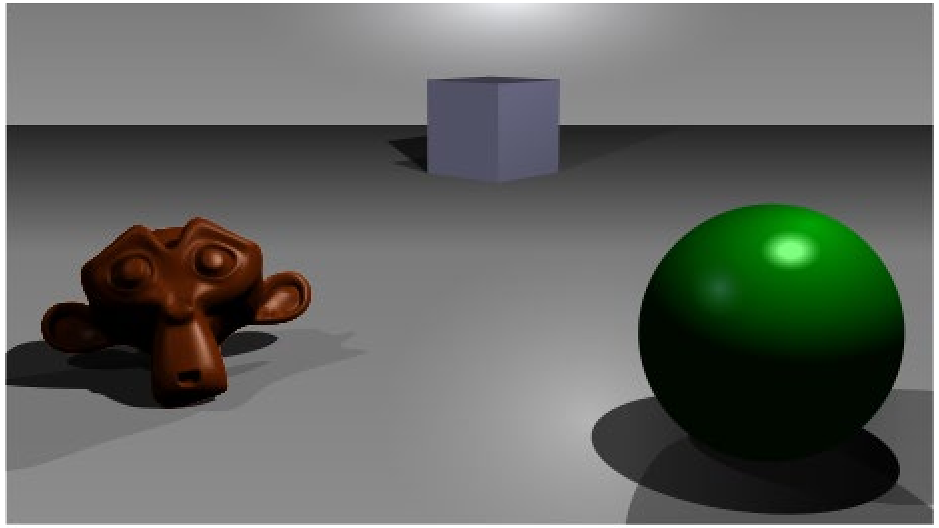
- This stage is also responsible for resolving visibility
  - Using the z-buffer

Z-Buffer

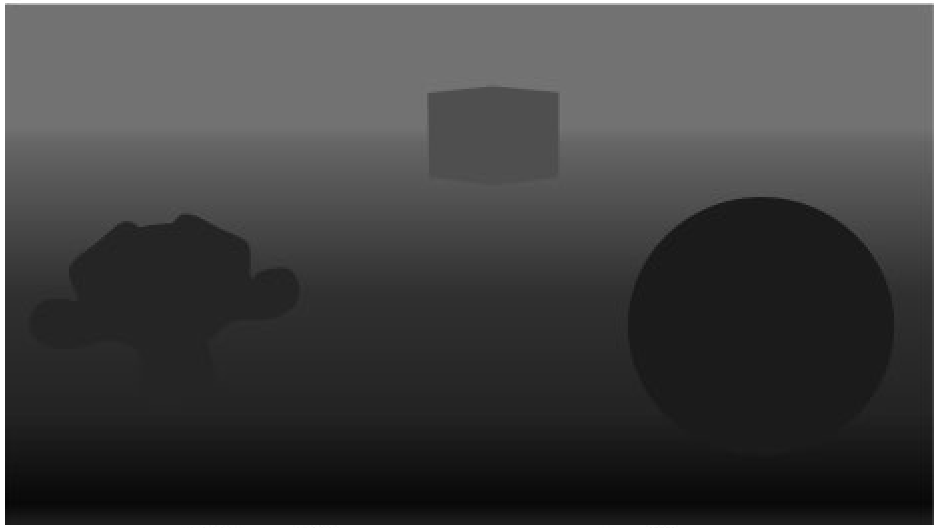




Z-Buffer



A simple three-dimensional scene



Z-buffer representation

Double Buffering

- The screen displays the contents of the color buffer
- To avoid perception of primitives being rasterized, *double buffering* is used
- Rendering takes place off screen in a *back buffer*
- Once complete, contents are swapped with the *front buffer*

Summary

- 3D models
- Model acquisition
- Fixed function vs. Programmable pipeline
- Stages of the programmable pipeline