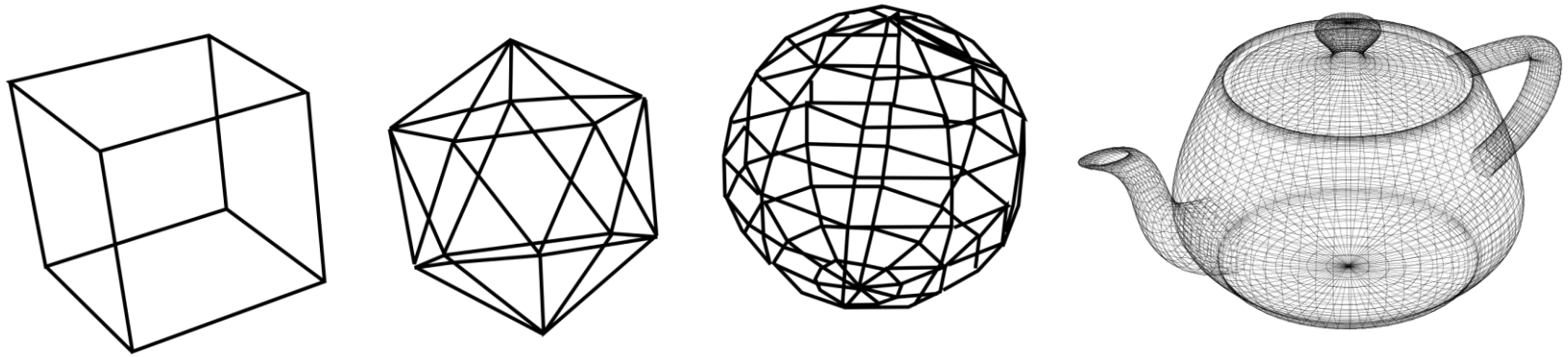
VIETNAM ACADEMY OF SCIENCE AND TECHNOLOGY
UNIVERSITY OF SCIENCE AND TECHNOLOGY OF HANOI

COMPUTER GRAPHICS

Lecture 1: Introduction to Computer Graphics

Lecturer: Dr. NGUYEN Hoang Ha



Course information

- Credit: 3
- Prerequisites: C++, Linear Algebra
- Moodle page: https://moodle.usth.edu.vn/enrol/index.php?id=488
  - Materials
  - Assignment submissions
- Grading
  - Attendance: 10%
  - Middle term: 40%
  - Final project: 50%
  - Rewards (+2, +1), Penalties (-2, - 1)

Textbooks

- Prescribed:
  - JungHyun Han. 3D Graphics for Game Programming (1st ed.). Chapman & Hall/CRC 2011.
- Recommended:
  - Hughes, J.F. and van Dam, A. and Foley, J.D. and Feiner, S.K. *Computer Graphics: Principles and Practice (3rd Edition)*, Addison-Wesley, 2014, ISBN: 9780321399526.
  - Donald D. Hearn and M. Pauline Baker. *Computer Graphics with OpenGL (4th Edition)*. 2010. Prentice Hall Professional Technical Reference.

Agenda

- Computer Graphics Overview
- Graphics Pipeline & Model creation

Computer Graphics Overview

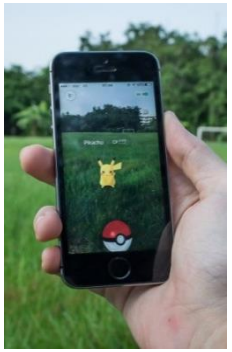
Reference: Computer Graphics: Principles and Practice (3rd Edition, 2014)

What is Computer Graphics?

- Process of **generating images using computers**
  - Creation, manipulation, storage, and visualization of object models
- Study of digital **synthesis** of, **interaction** with, and **manipulation** of **visual content**

→ Computer graphics consists of:

- Modeling (representations)
- Rendering (display): usually “computer graphics” refers to rendering
- Interaction (user interfaces)
- Animation (combination of 1-3)
- Others: color theory, AR/VR



Computer Graphics is everywhere

- GUIs on PC, smartphones, automobile dashboards…
- Games
- CAD/CAM
- Architecture
- Movie industry
- Scientific Visualization and Analysis
- Medicine and Virtual Surgery

Games

- Entertainment games

- For fun
- Rich experiences

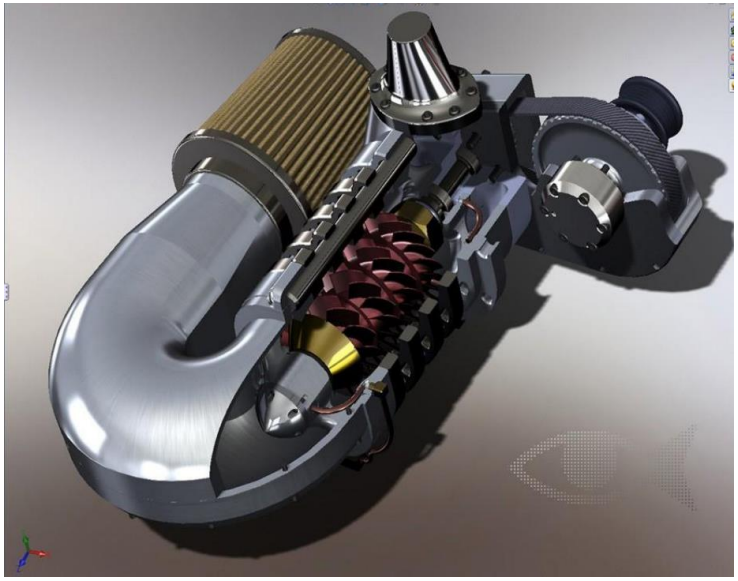
- Serious games

- For problem solving
- Important & precise experience

8

CAD/CAM

- Virtual prototypes

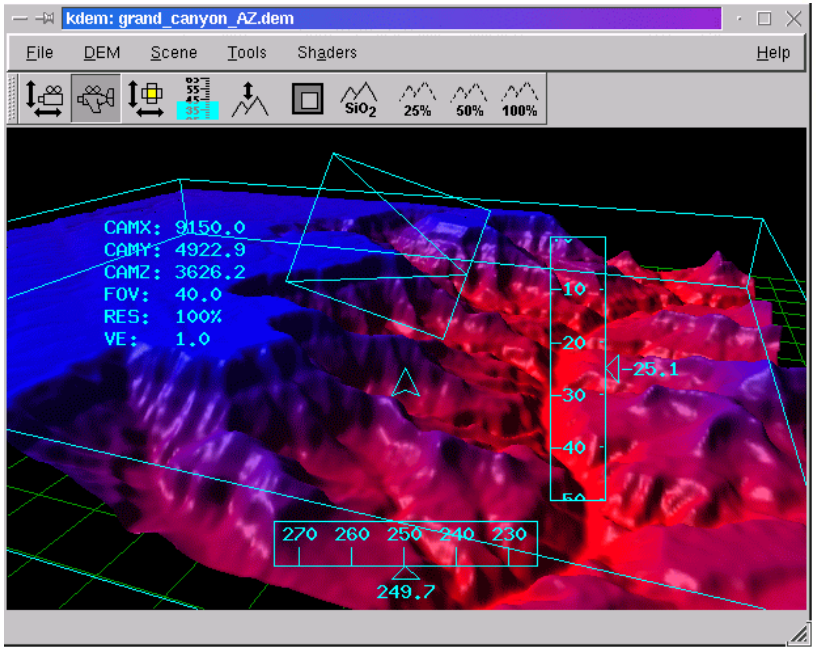


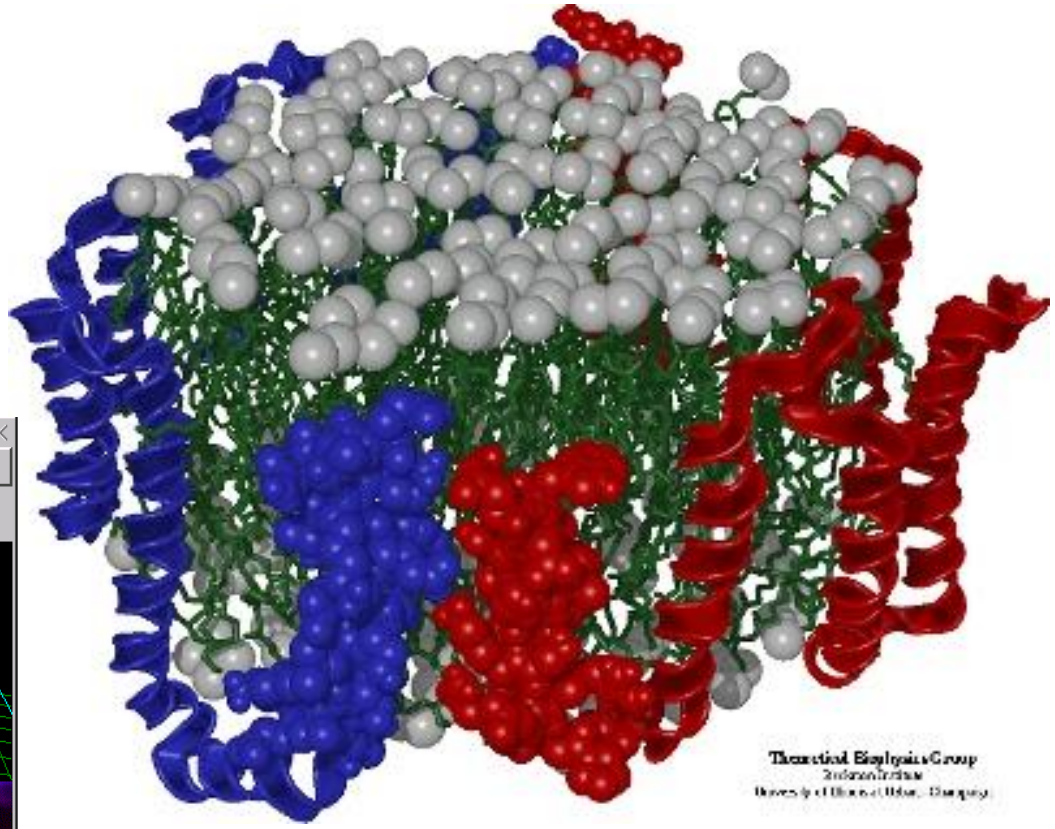
Architecture

Movie industry

- Computer-Generated Imagery (CGI)

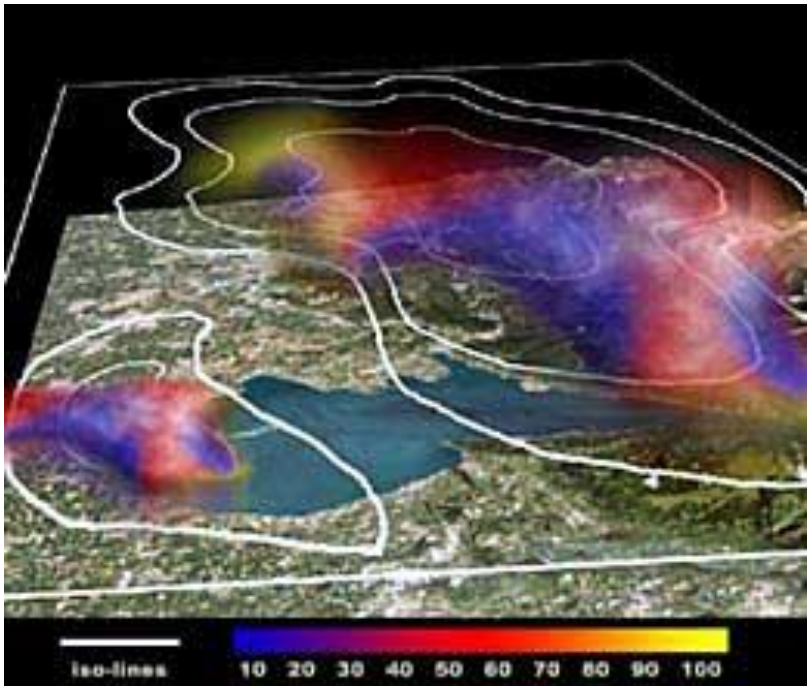
Scientific Visualization and Analysis



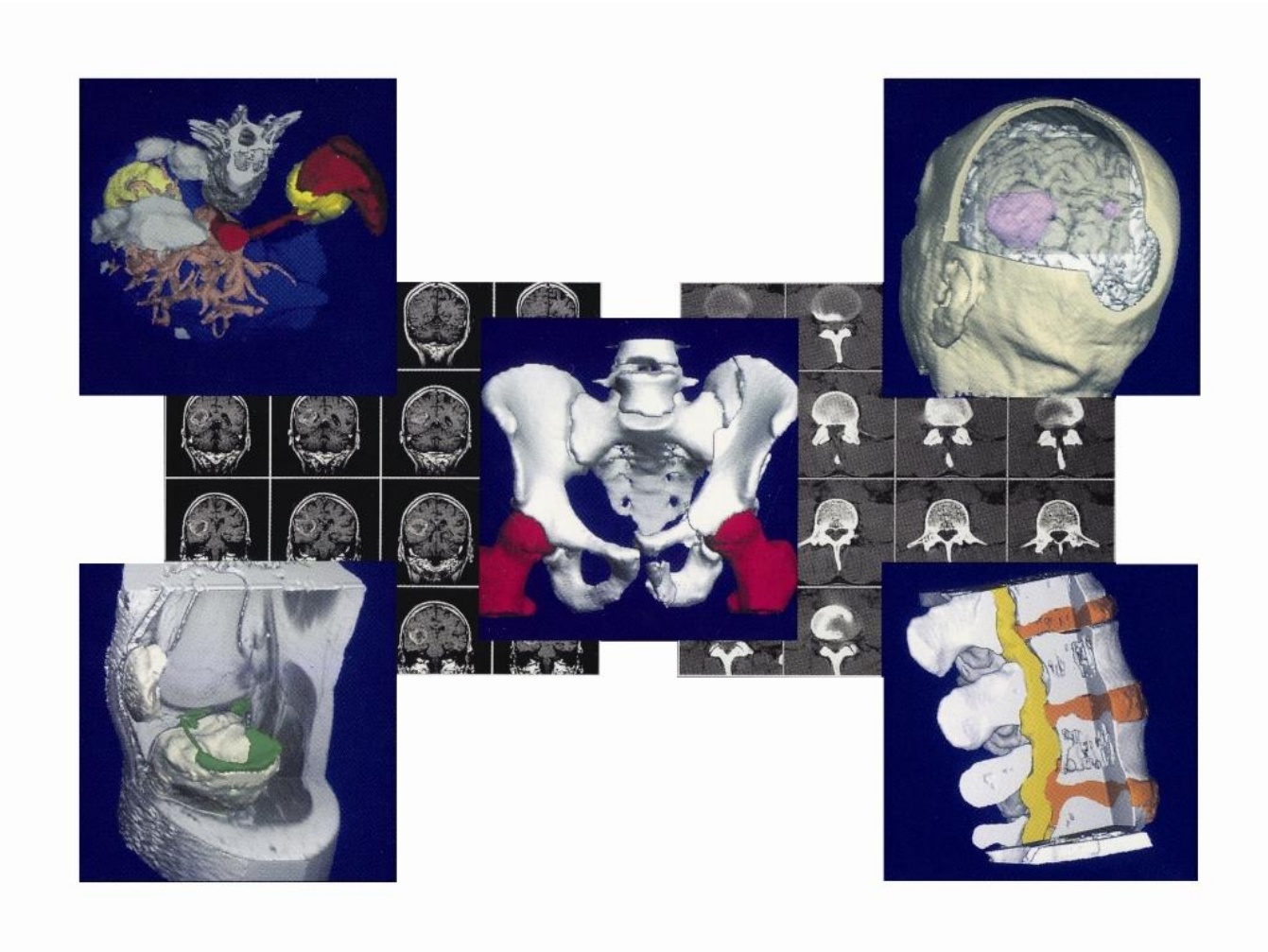


Scientific Visualization and Analysis

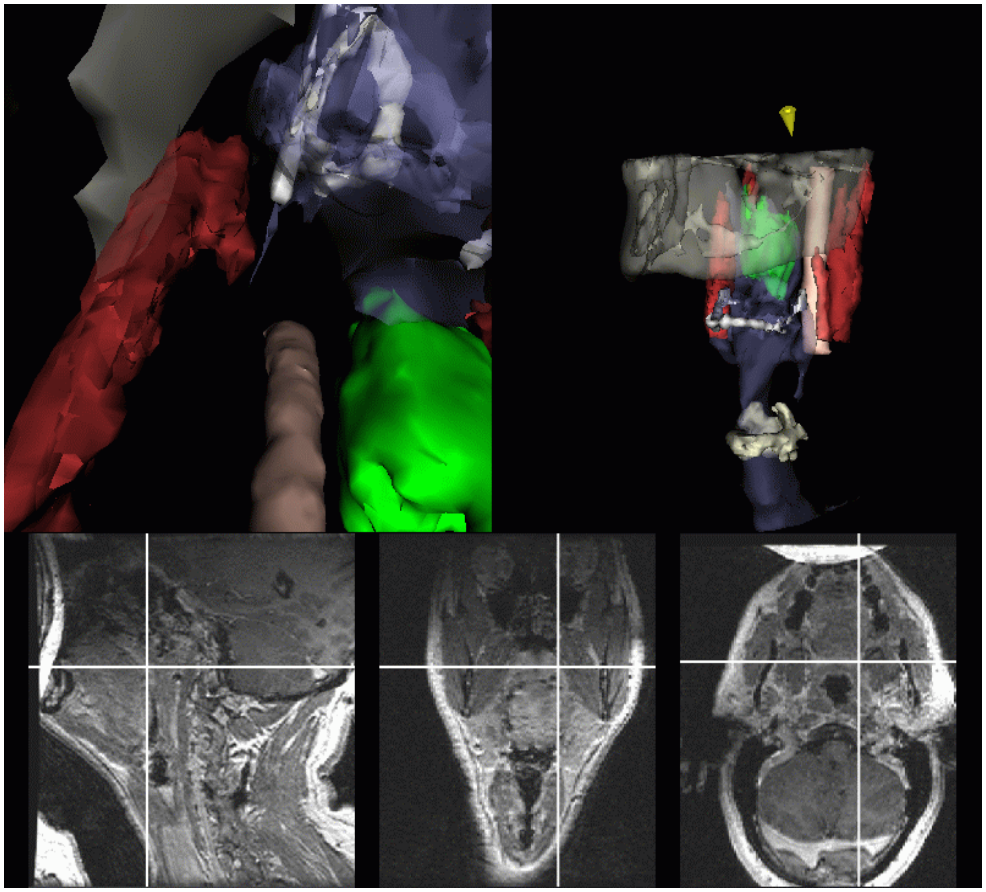
- Data Visualization
  - Scientific, Engineering, Medical data
  - Visualizing millions to billions of data points
  - See trends
  - Different schemes

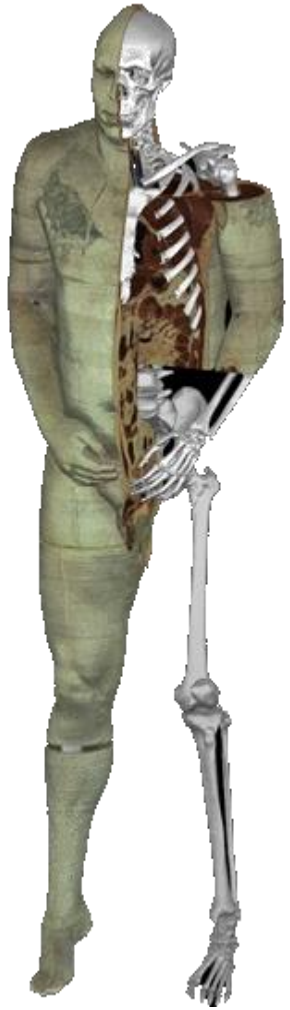


Medical images rendering



Medicine and Virtual Surgery

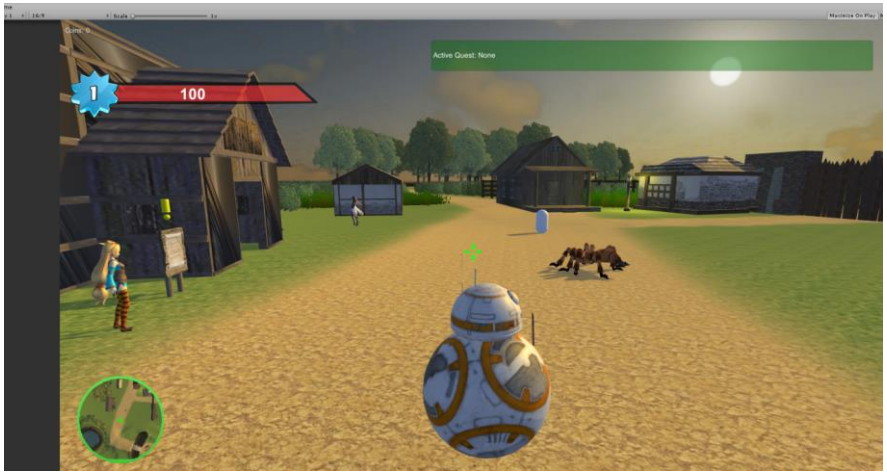




The Visible Human project

Why Computer Graphics is fun?

- Inter-discipline:
  - Within computer science: systems, hardware, compilers, algorithms, numerical methods, image processing, computer vision.
  - Outside computer science: math, physics, art, architecture…
- You can see the results, create your own “virtual” world



Course objectives

- Theory
  - Understand mathematical aspects and algorithms underlying 3D graphics systems
- Practice
  - Write 3D graphics programs

→ Prepare the foundation for

- Game programming
- AR/VR, metaverse

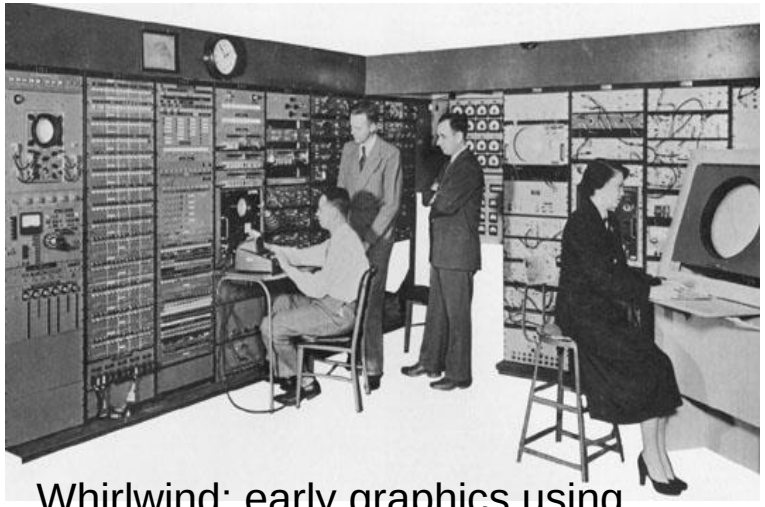
What to learn?

- Graphics pipeline
- Fundamentals of CG algorithms
- Concentrate on 3D, not 2D illustration
- Basic OpenGL, Web 3D

What NOT to learn?

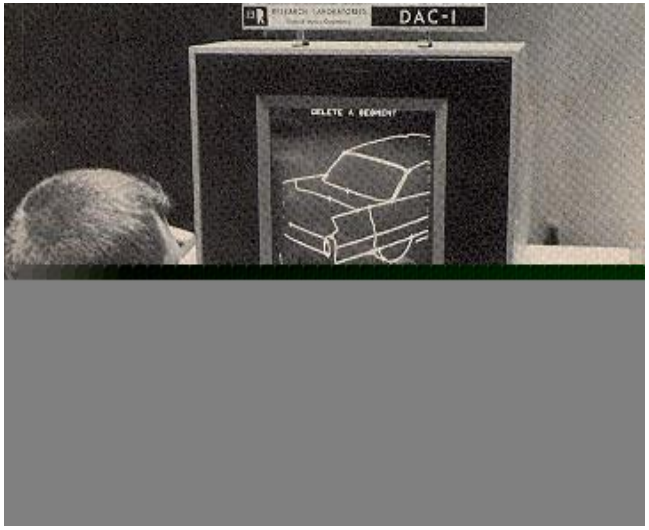
- Image processing
- Game design
- Artistic skills
- Software packages
  - Photoshop
  - CAD, CAM
  - Maya…

History

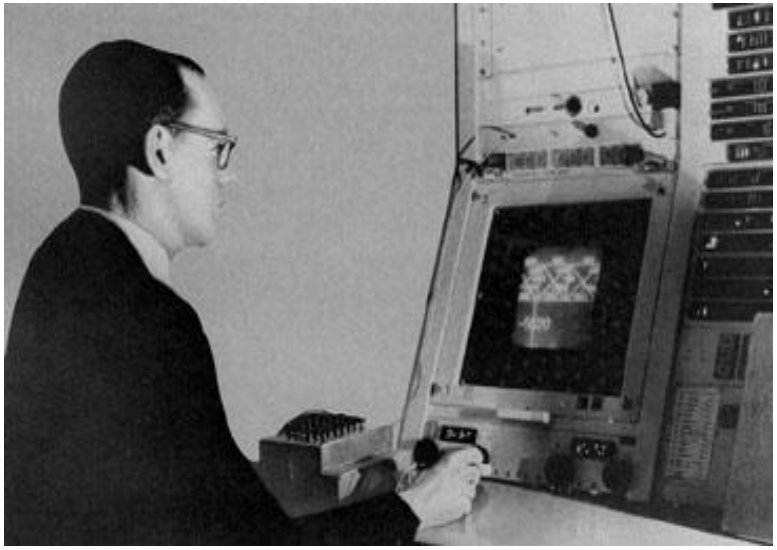


Whirlwind: early graphics using VectorScope (1951)

Spacewars: first computer graphics game (MIT 1961

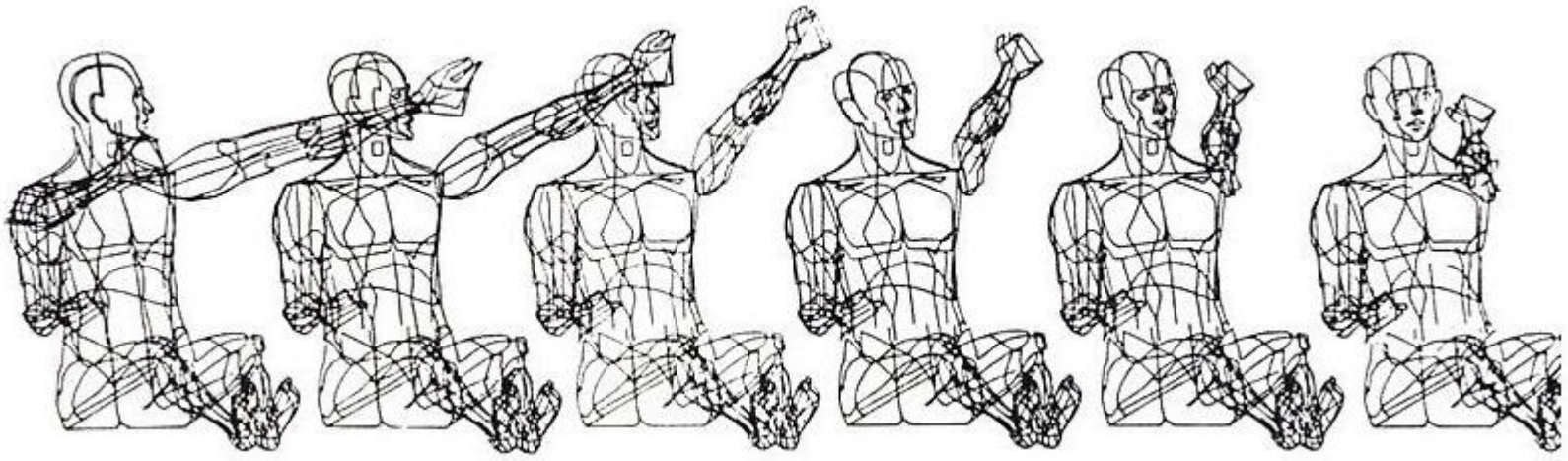


first CAD system (IBM 1959)



SketchPad: first interactive graphics (1963)

History

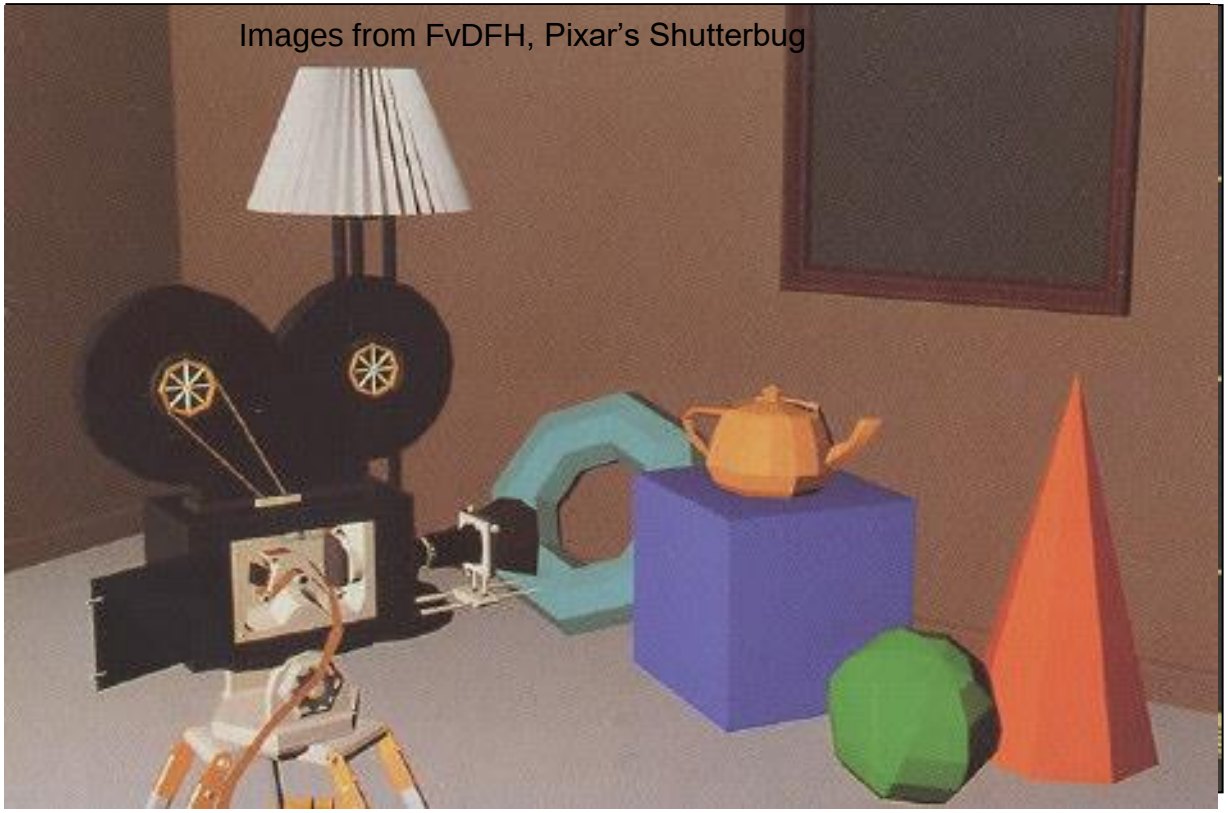


The term “computer graphics” was coined in 1960 by William Fetter to describe new design methods that he was developing at Boeing.

He created a series of widely reproduced images on a plotter exploring cockpit design using a 3D model of a human body

Rendering: 1960s (visibility)

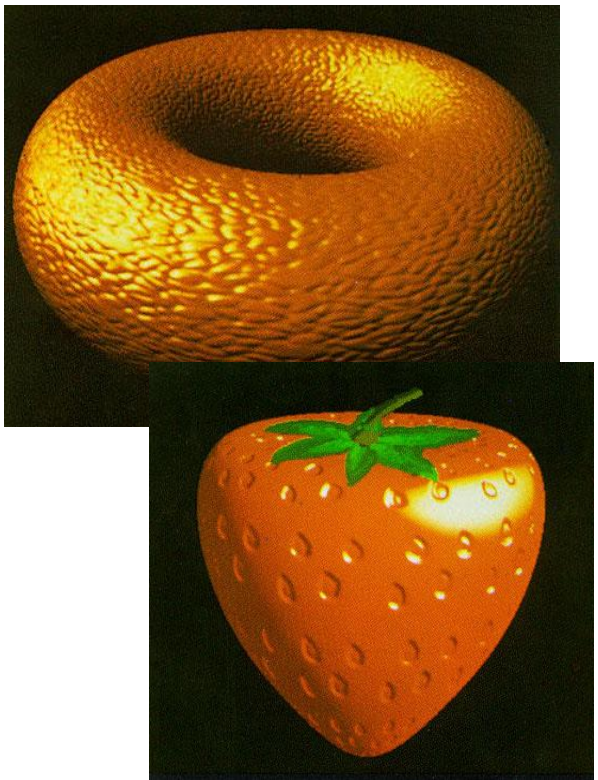
- Roberts (1963), Appel (1967) - hidden-line algorithms
- Warnock (1969), Watkins (1970) - hidden-surface



Rendering: 1970s (lighting)

1970s - raster graphics

- Gouraud (1971) - diffuse lighting, Phong (1974) - specular lighting
- Blinn (1974) - curved surfaces, texture
- Catmull (1974) - Z-buffer hidden-surface algorithm

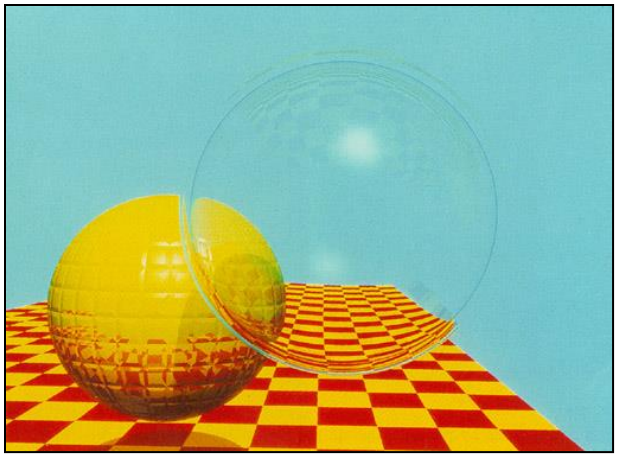


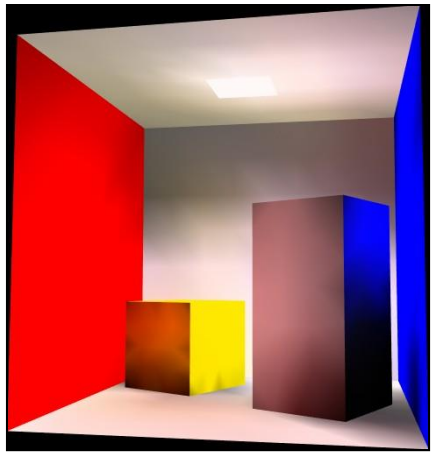
First bump-mapped images (Blinn 1978)

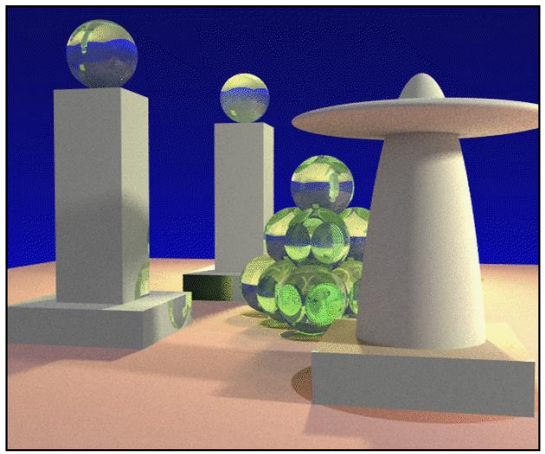
Rendering (1980s, 90s: Global Illumination)

early 1980s - global illumination

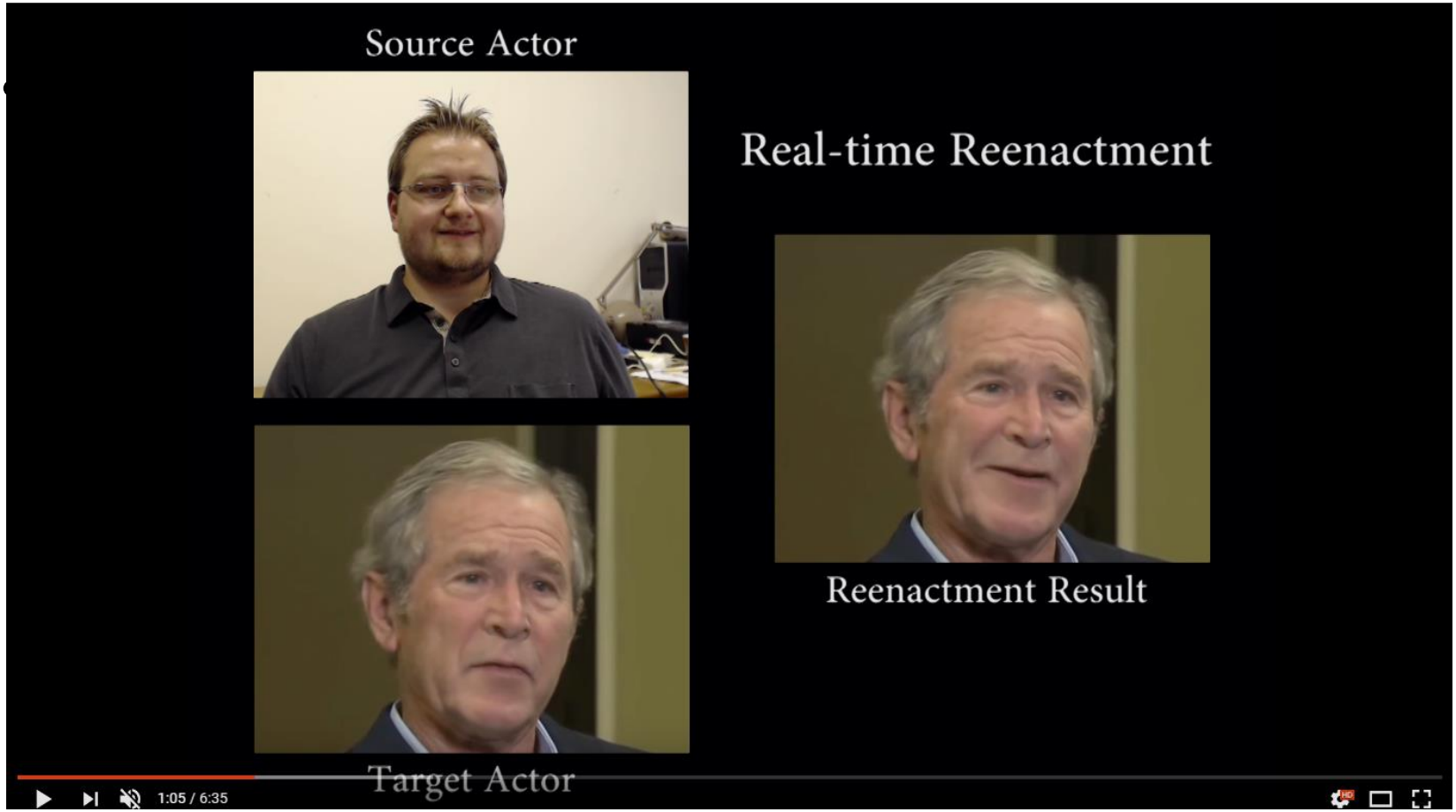
- Whitted (1980) - ray tracing
- Goral, Torrance et al. (1984) radiosity
- Kajiya (1986) - the rendering equation







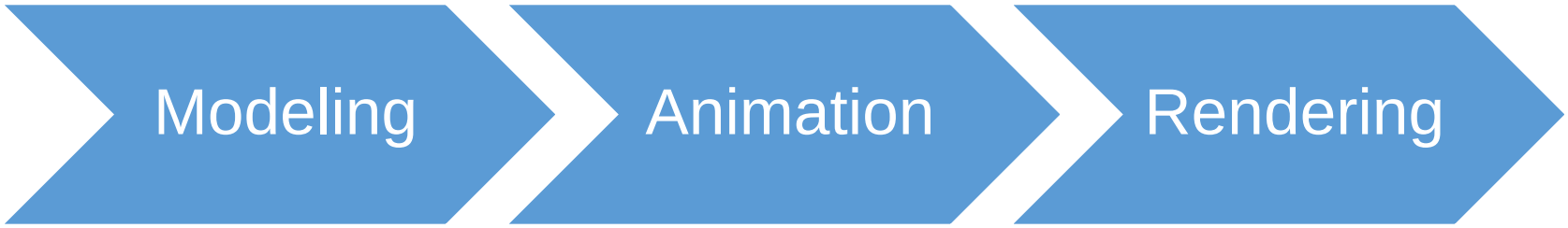
2010s: realistic images



Graphics pipeline & Model Creation

Reference: Chapter 1 3D graphics for Game Programming

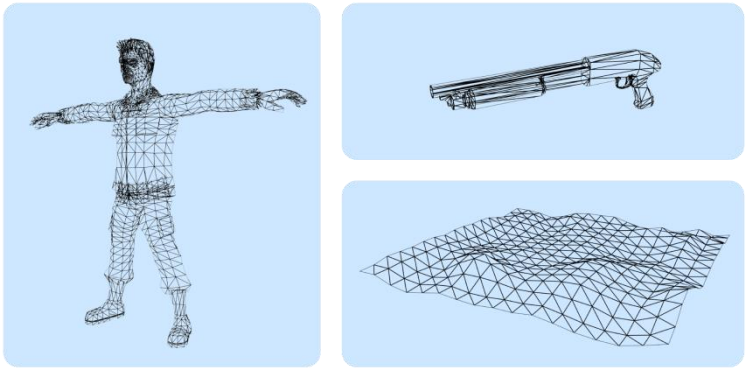
Graphics pipeline



- Artists: in charge of modelling, off-line task of animation
- Software engineers: run-time task of animation, rendering

Modeling

- *Model* is referred to as a computer representation of an object, and *modeling* is the process of creating the components of the virtual environment.
- The most popular modeling method in games is using *polygons*, and the model is called *polygon meshes*.



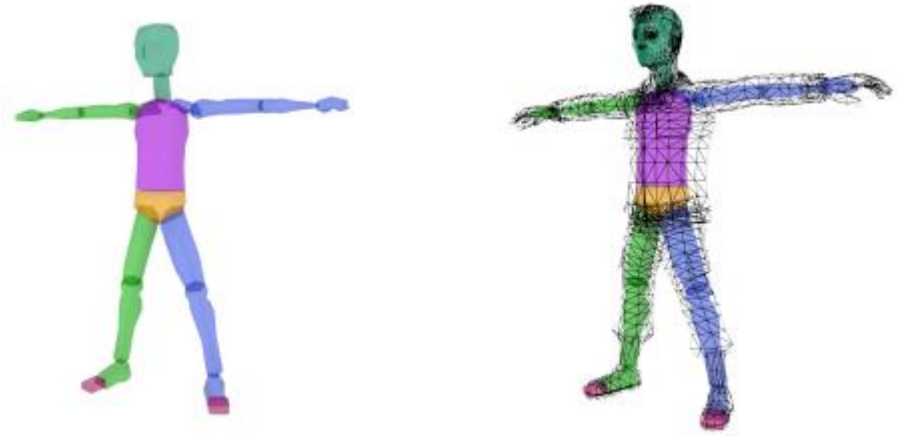
texture image

- creating *textures* to increase their realism.

Animation

- Eg: soldier model should be able to walk, run, and crawl.
  - → specify the *skeletal structure* of the soldier, define how the skeletal motion deforms the soldier's polygon mesh.

→This process is referred to as *rigging*.



skeleton skeleton embedded into the mesh

Animation (cont')

- A rigged model is *animated* by artists
  - 3ds Max and 3ds Maya are popularly used for modeling and off-line animation.

Rendering

- Generate a sequence of images (a.k.a frame) to illustrate the character animation
- Rendering: process of generating 3D image from 3D data

- Take into account: 3D models, texture, lighting condition, camera pose (position, aperture, lens, iso…)

Rendering

- Unlike modeling and off-line animation conducted by artists, run-time animation and rendering are executed by software engineers.
- A game program is usually built upon graphics APIs such as Direct3D and OpenGL.
  - Direct3D is part of Microsoft's DirectX API, and is available only for Microsoft platforms.
  - OpenGL (Open Graphics Library) is managed by a non-profit consortium, the Khronos Group, and is a cross-platform standard API.
- Graphics APIs
  - They provide application programmers with essential graphics functions. Today, such functions are implemented in a processor specialized for graphics, named GPU (Graphics Processing Unit).
  - A graphics API can be taken as a software interface of the GPU. The API translates the application's graphics command to instructions that can be executed by the GPU.

3D model: polygon mesh

- In real-time applications such as games, the *polygon mesh* representation dominates. It is not an accurate representation but an approximate one, where the mesh vertices are the points *sampling* the smooth surface.

$$(x - C_x)^2 + (y - C_y)^2 + (z - C_z)^2 = r^2$$

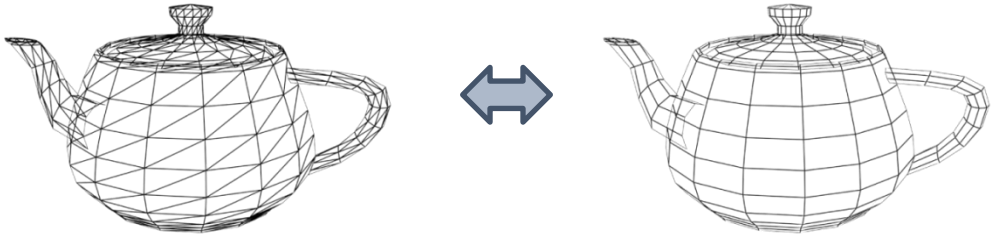
Polygon Mesh (cont')

- In OpenGL, the polygons should be convex and planar.

convex polygons

concave polygons

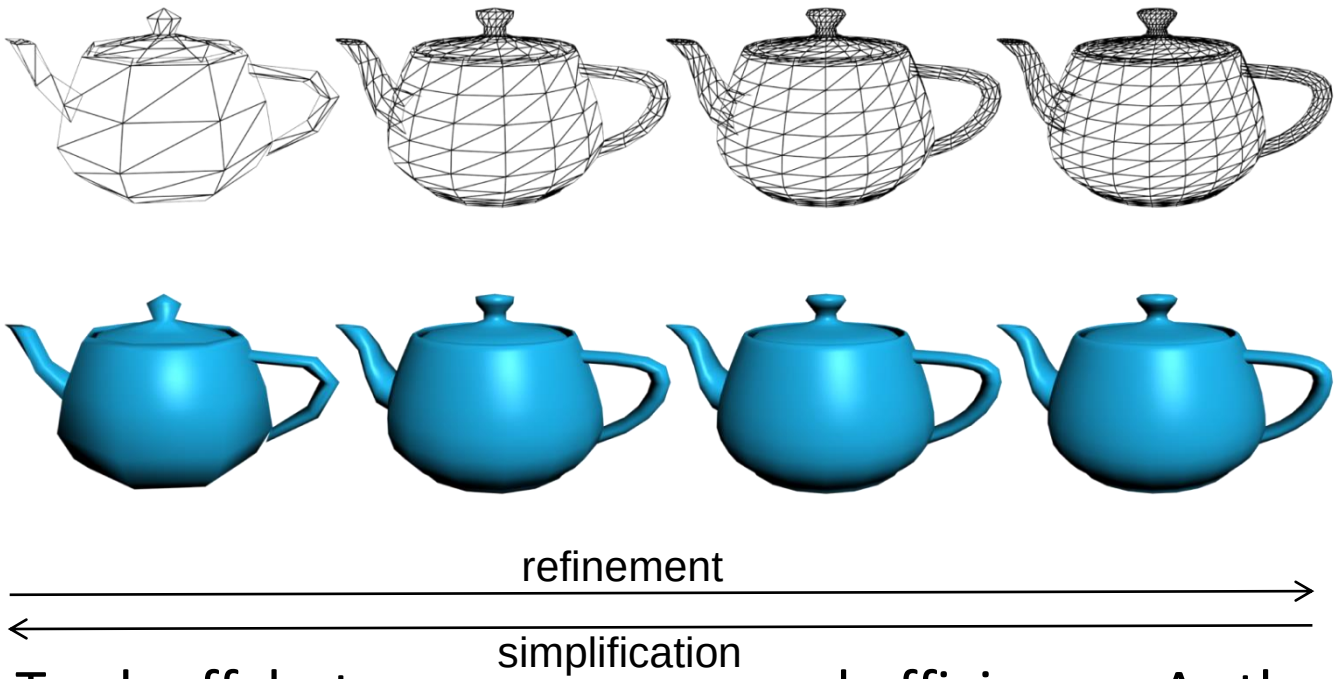
- Direct3D supports only triangles. Note that the simplest polygon is a triangle, and triangles are both convex and planar.



triangle mesh

quad mesh

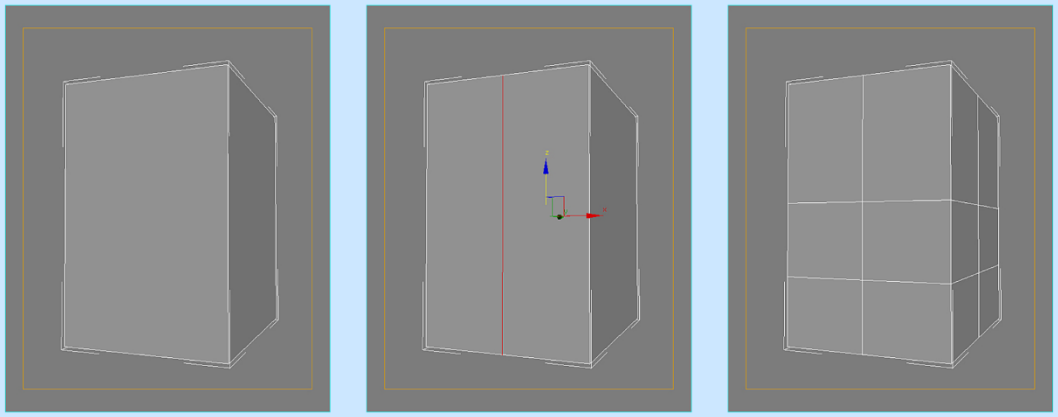
Level of detail (LOD) of Polygon Mesh



- Tradeoff between accuracy and efficiency: As the resolution is increased, the mesh becomes closer to the original curved surface, but the time needed for processing the mesh is increased.

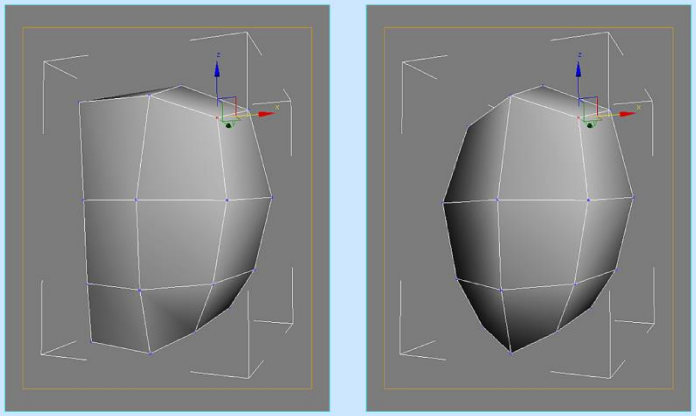
Polygon Mesh Creation

- In most cases, the polygon mesh is interactively created using graphics packages such as 3Ds Max.

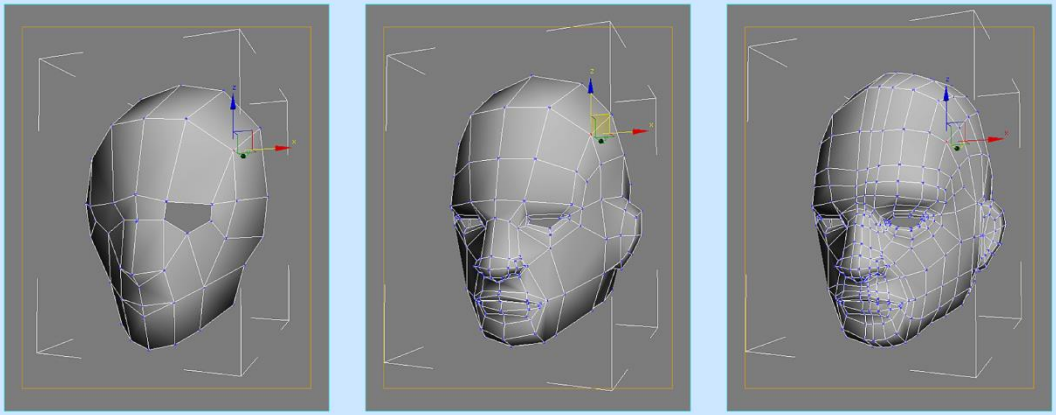


(a) Edges are selected, and then connected to refine the box mesh.

Polygon Mesh Creation (cont')

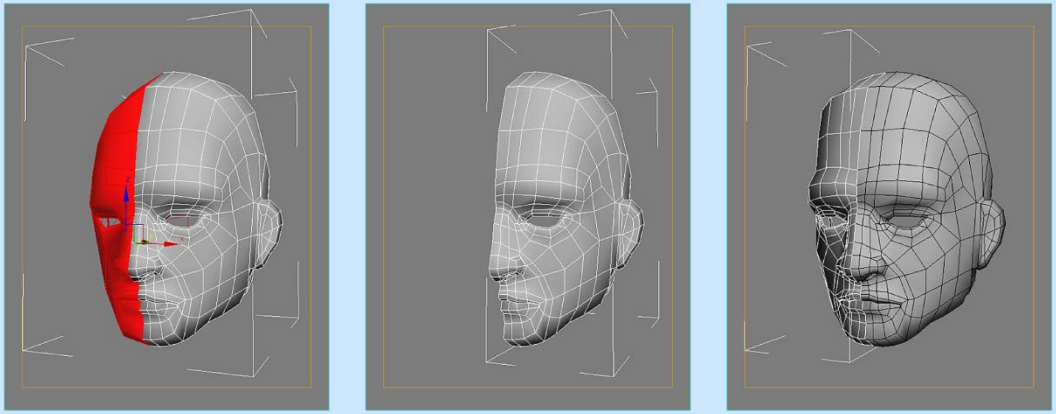


(b) Vertices are selected and moved to change the geometry of the mesh.

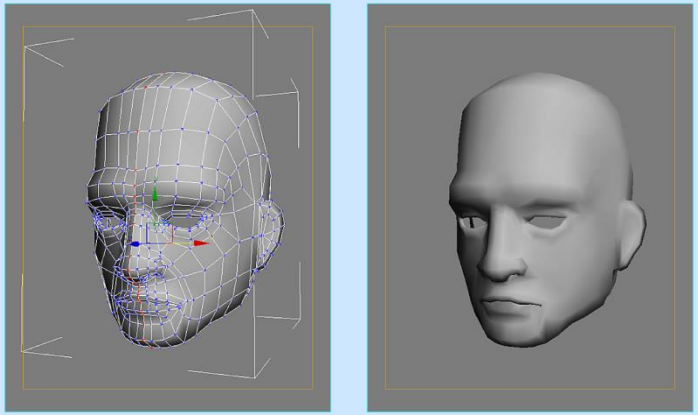


(c) A polygon is cut out for creating an eye, and the mesh is repeatedly refined.

Polygon Mesh Creation (cont')

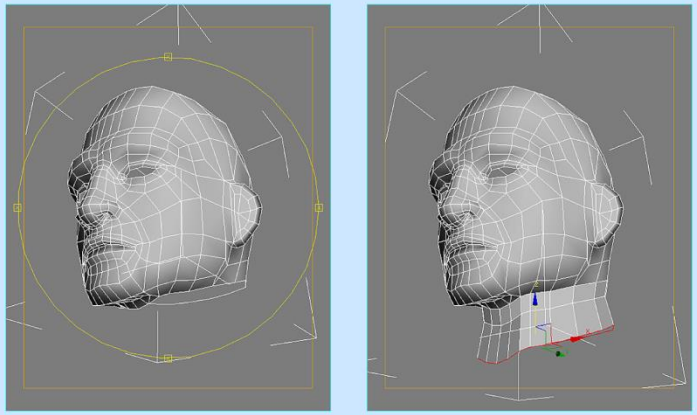


(d) The right half of the head is removed, and the left half is copied and reflected to make the head symmetric.



(e) Two separate meshes of the head are combined through a welding operation to produce a single mesh.

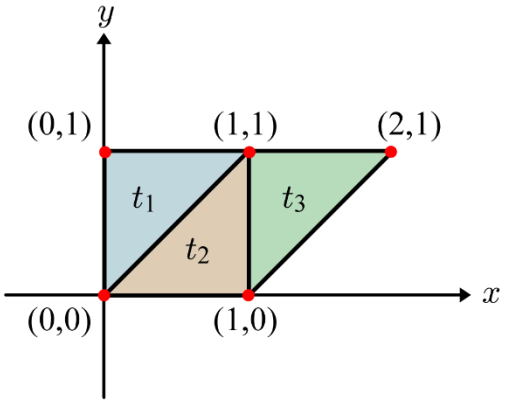
Polygon Mesh Creation (cont')



(f) The neck part of the mesh is extruded.

Polygon Mesh: Non-indexed Triangle List

- Non-indexed triangle list
  - The vertices are enumerated in a memory space, named *vertex buffer*.
  - Three vertices are read in linear order to make up a triangle.



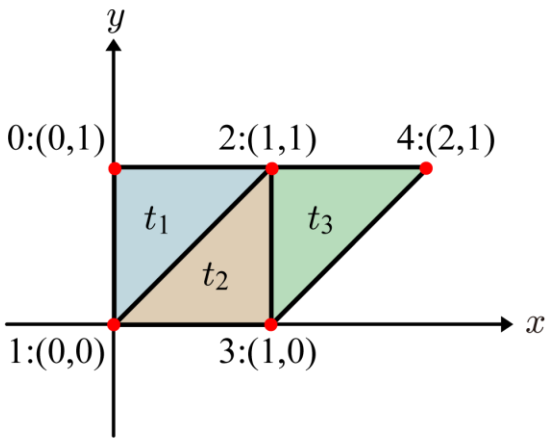
vertex buffer

(0,1)	t_1
(0,0)	
(1,1)	
(0,0)	t_2
(1,0)	
(1,1)	
(1,1)	t_3
(1,0)	
(2,1)	

- It is inefficient because the vertex buffer contains redundant data.

Polygon Mesh: Indexed Triangle List

- Separate *index buffer (indexed triangle list*) is better:
  - A vertex appears only once in the vertex buffer.
  - Three indices per triangle are stored in the index buffer.



vertex buffer

0	(0,1)
1	(0,0)
2	(1,1)
3	(1,0)
4	(2,1)

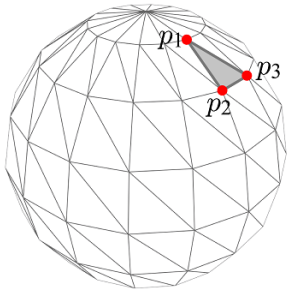
index buffer

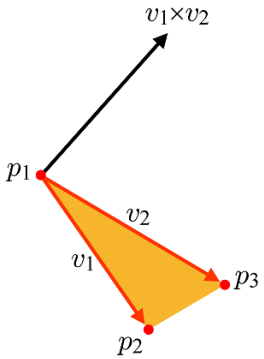
0	t_1
1	
2	
1	t_2
3	
2	
2	t_3
3	
4	

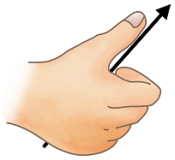
- The vertex data stored in the vertex buffer include not only positions but also normals, texture coordinates, and many more. Therefore, the vertex buffer storage saved by removing the duplicate data outweighs the additional storage needed for the index buffer.

Surface Normal

- Surface normals play a key role in computer graphics.
- Given triangle $\langle p_1, p_2, p_3 \rangle$, let v_1 denote the vector connecting the first vertex (p_1) and the second (p_2). Similarly, the vector connecting the first vertex (p_1) and the third (p_3) is denoted by v_2. Then, the *triangle normal* is computed using the *cross product* based on the *right-hand rule*.





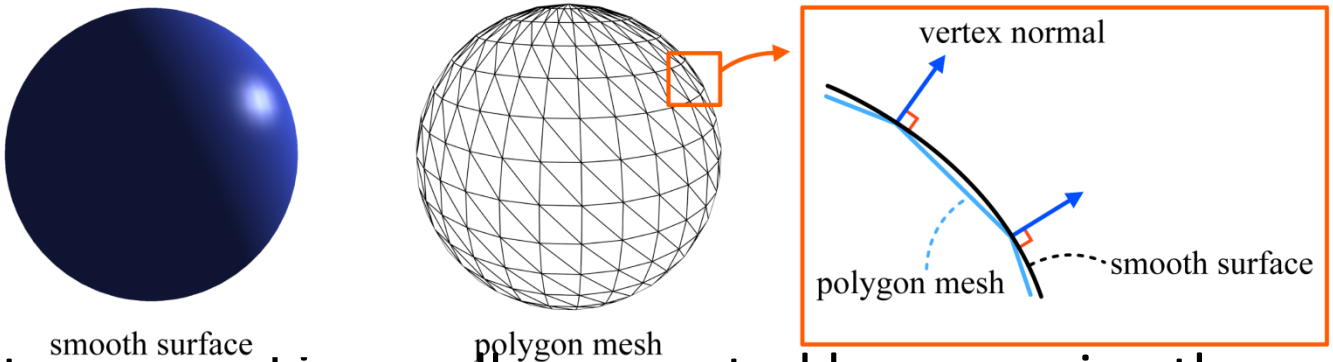


$$\frac{v_1 \times v_2}{\|v_1 \times v_2\|}$$

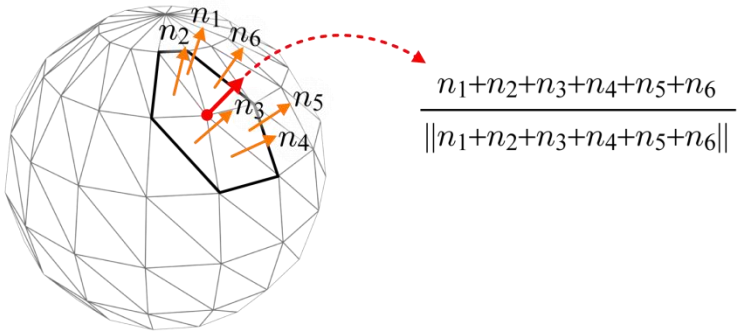
- When v_1 and v_2 are represented in $(x_1\ y_1\ z_1)$ and $(x_2\ y_2\ z_2)$, respectively, $v_1 \times v_2$ is defined as $(y_1z_2\text{-}z_1y_2\ \ z_1x_2\text{-}x_1z_2\ \ x_1y_2\text{-}y_1x_2)$.
- Every normal vector is made to be a unit vector by default.
- Note that p_1, p_2, and p_3 are ordered *counter-clockwise* (CCW).

Surface Normal (cont')

- Vertex normal should approximate the normal of the smooth surface at the vertex position.

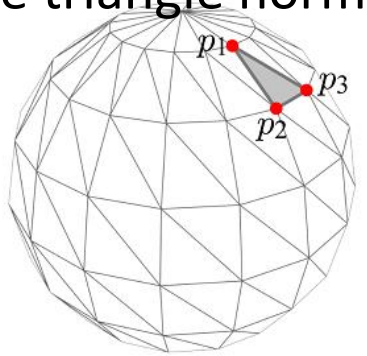


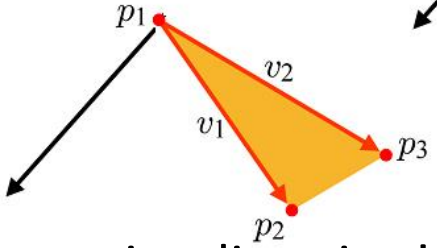
- A vertex normal is usually computed by averaging the normals of all polygons sharing the vertex.

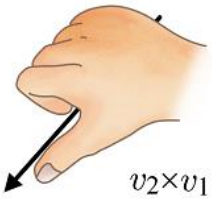


Surface Normal (cont')

- The vector connecting the first vertex (p_1) and the second (p_3) and that connecting the first vertex (p_1) and the third (p_2) define the triangle normal.





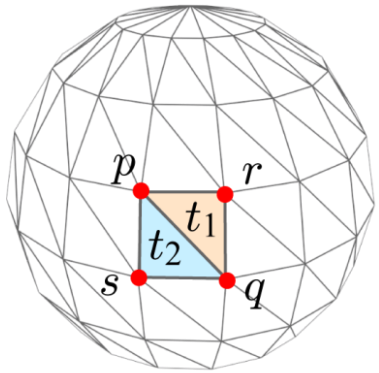


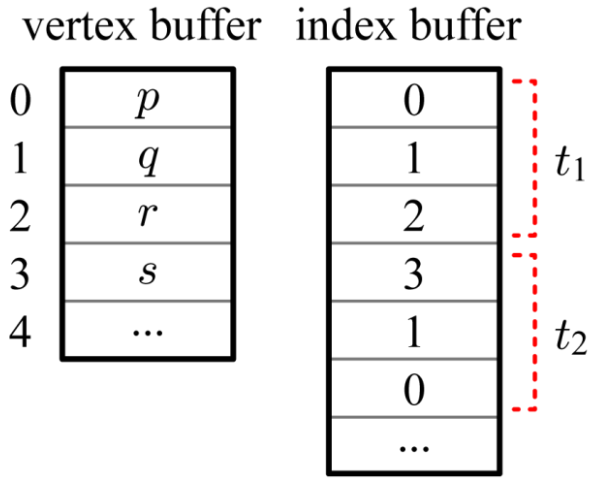
$$\frac{v_2 \times v_1}{||v_2 \times v_1||}$$

- The normal is in the opposite direction!
- The normal direction depends on the vertex order, i.e., whether $\langle p_1, p_2, p_3 \rangle$ or $\langle p_1, p_3, p_2 \rangle$.
- In computer graphics, surface normals are supposed to point out of the polyhedron. For this, we need CCW ordering of the vertices, i.e., $\langle p_1, p_2, p_3 \rangle$, instead of $\langle p_1, p_3, p_2 \rangle$.

Surface Normal (cont')

- In the index buffer, the vertices are ordered CCW.

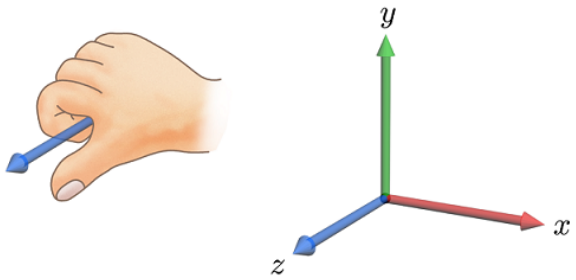




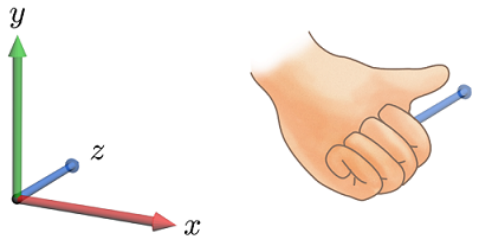
Coordinate System's Handedness and Vertex Ordering

- Right-hand system vs. left-hand system

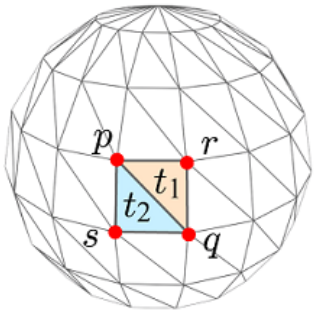
right-hand system (RHS)



left-hand system (LHS)



- To make the normals point out of the object, RHS adopts CCW ordering whereas LHS adopts CW ordering.



for RHS

$t_1 = \langle p,q,r \rangle$

$t_2 = \langle s,q,p \rangle$

for LHS

$t_1 = \langle p,r,q \rangle$

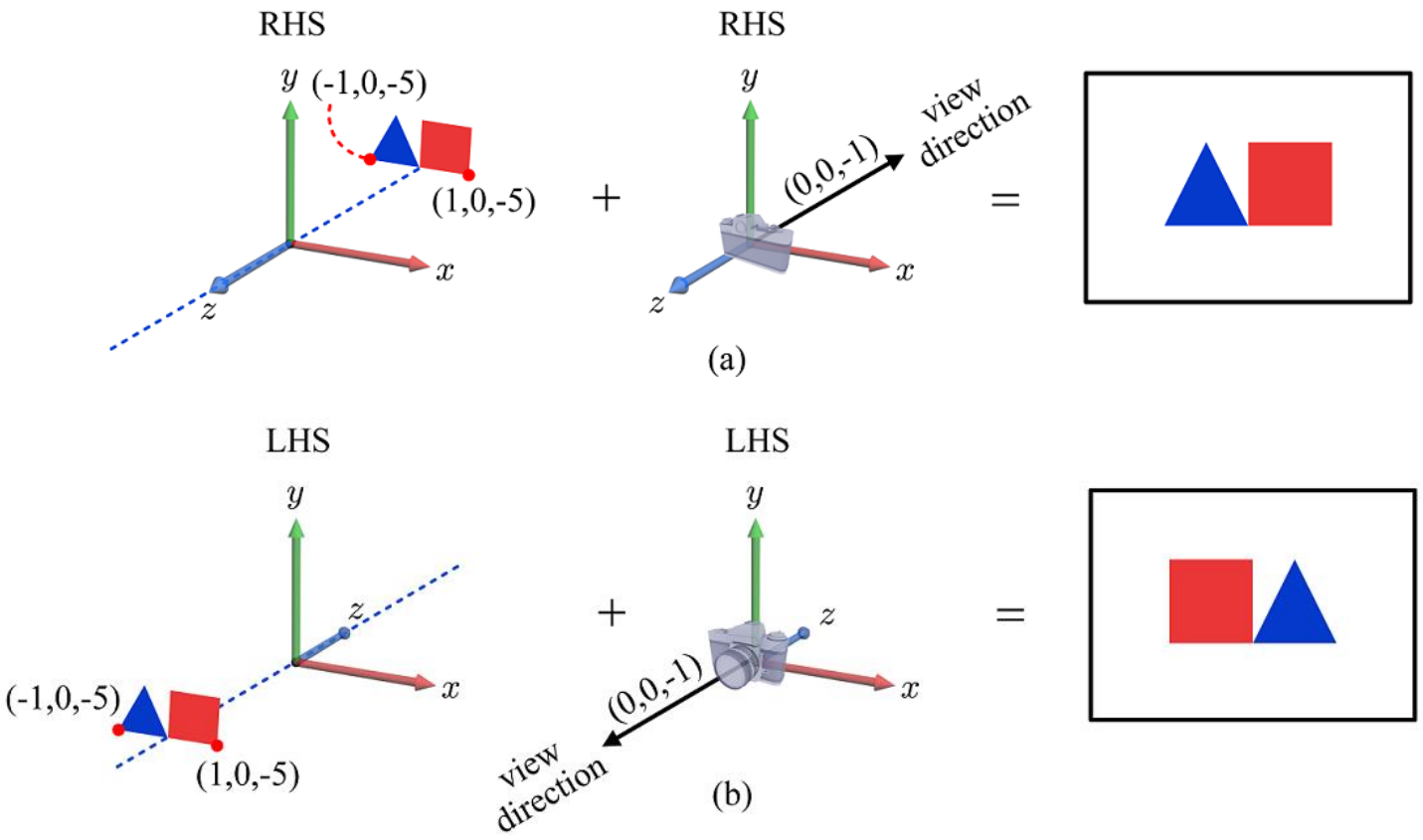
$t_2 = \langle s,p,q \rangle$

Conversion between RHS and LHS

- RHS is the default coordinate system of OpenGL, and LHS is that of Direct3D. Porting between RHS and LHS needs to be clearly understood.
- The difference constructed a barrier hard to cross over for inexperienced programmers. However, the barrier has crumbled away, and we can run an RHS-based application on top of Direct3D, for example. It has been due to GPU's programmability.

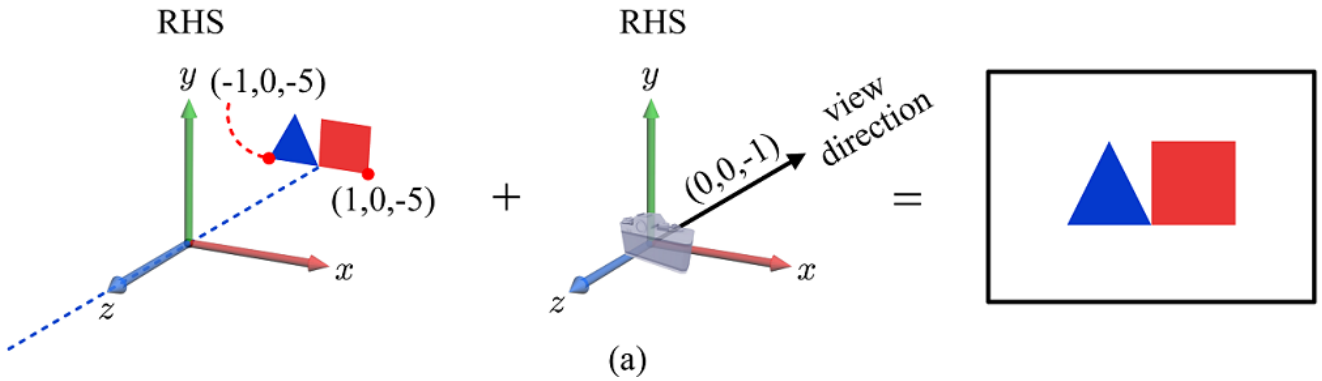
Conversion between RHS and LHS (cont')

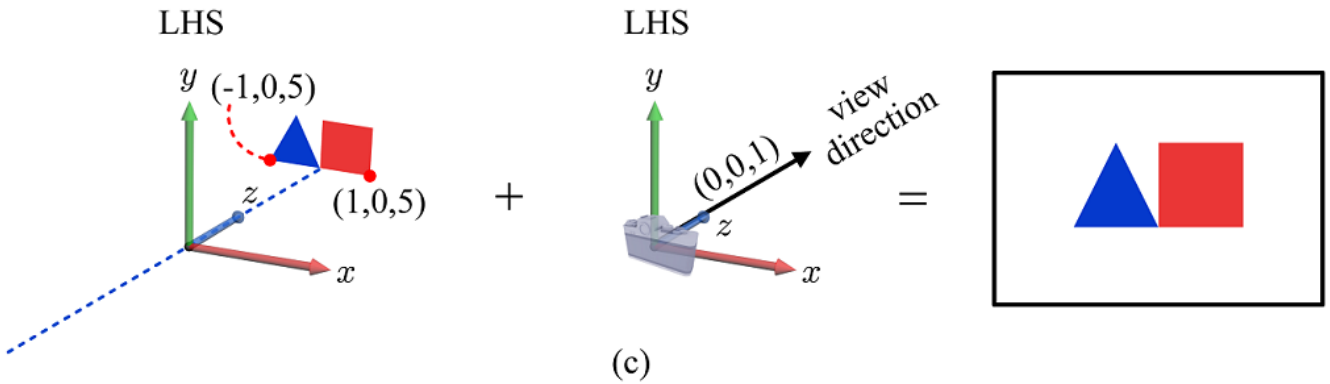
- Problem: flipped image could be negated



Conversion between RHS and LHS (cont')

- Porting an application between RHS and LHS needs two tasks.
  - The order of triangle vertices has to be changed: $\langle p_1, p_2, p_3 \rangle \rightarrow \langle p_1, p_3, p_2 \rangle$
  - The *z*-coordinates (objects and view parameters) are negated.





Exercise

Consider a *tetrahedron* composed of four vertices, (0,0,0), (1,0,0), (0,1,0), (0,0,1).

- (a) tetrahedron is exported to a *right-hand* system, draw the vertex and index buffers for its indexed triangle list.
- (b) Assuming that the tetrahedron is exported to a *left-hand* system, do the same.