Mesh Surfaces, Coordinate Systems & Model Transformations

CSE606: Computer Graphics
Jaya Sreevalsan Nair, IIIT Bangalore
January 22, 2025

Mesh Surfaces

Mesh Surfaces

While parametric surfaces provide a simple mathematical formulation and simplify evaluation, they are difficult to use for complex - and real-life - shapes.

Beziers, NURBS, etc are not convenient for modelling such objects, as well as clothing, etc.

Further, we sometimes have data of the points on a surface (e.g. from scanning) and need to compute a surface that “fits” these points

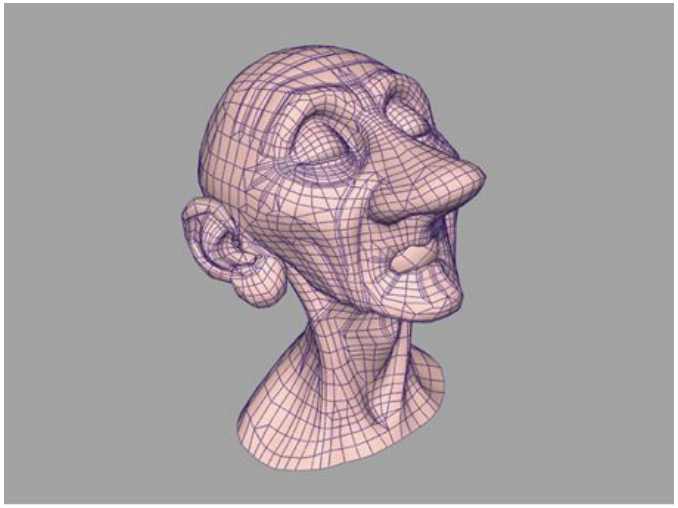


Image from: *Subdivision Surfaces in Character Animation*, Tony DeRose, Michael Kass, Tien Truong; Pixar Animation Studios

Mesh Surfaces

Model such objects directly as a 2D mesh:

- Surface defined by set of vertices, and connectivity between vertices that define triangles or quadrilaterals.

https://youtu.be/uMVtpCPx8ow?si=IQhrroeS9QtH3hGc

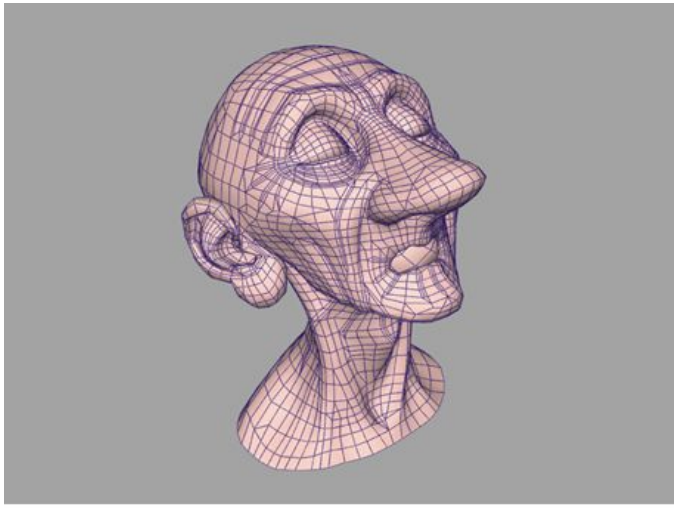


Image from: *Subdivision Surfaces in Character Animation*, Tony DeRose, Michael Kass, Tien Truong; Pixar Animation Studios

Mesh Surfaces

- Represent a curve or surface as connected sets of piecewise linear elements
  - Line segments (polylines) for curves (1D)
  - Polygon elements for surfaces (2D)
- Triangles and quadrilaterals most commonly used for surface meshes.
  - Other polygons reduced to some combination of the basic types.
- A preferred mechanism for modeling “natural” (i.e. not “manufactured”) surfaces, which typically do not have convenient analytical formulations.
- Special data structures such as half-edge, winged-edge, lath, etc used to enable efficient processing of large meshes
- Geometric properties of the object are derived from the mesh representation directly or through refinement of the mesh (e.g. using subdivision schemes)

Mesh Surfaces

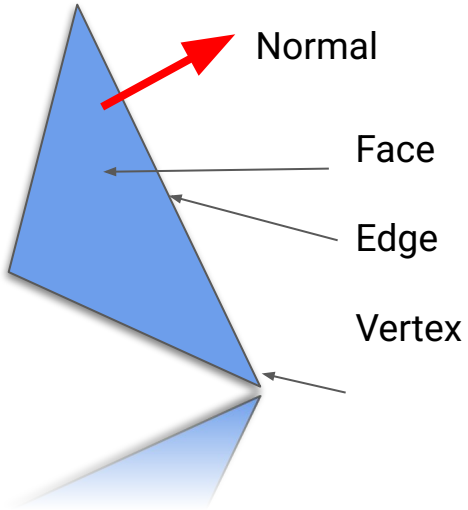
Mesh representation - vertices and connectivity - provides information on the shape of the object.

Vertices - Edges - Faces

* *Depending on the application, edges may not be explicitly represented*

Additional computations needed to compute properties such as surface normal and curvature

1. Normal at a vertex can be computed as a weighted average of the normals of the mesh elements (triangles) at that vertex
2. Use refinement techniques to compute better approximations of the mesh (e.g. subdivision schemes) and use these to compute normals etc



Rendering Mesh Surfaces

OpenGL/WebGL provide mechanisms to render triangles as groups of vertices. Thus, generate a vertex array from coordinates of the vertices and render in WebGL using **WebGLRenderingContext.drawElements()** with gl.TRIANGLES as the primitive mode. Also, TRIANGLE_STRIP, TRIANGLE_FAN

Options:

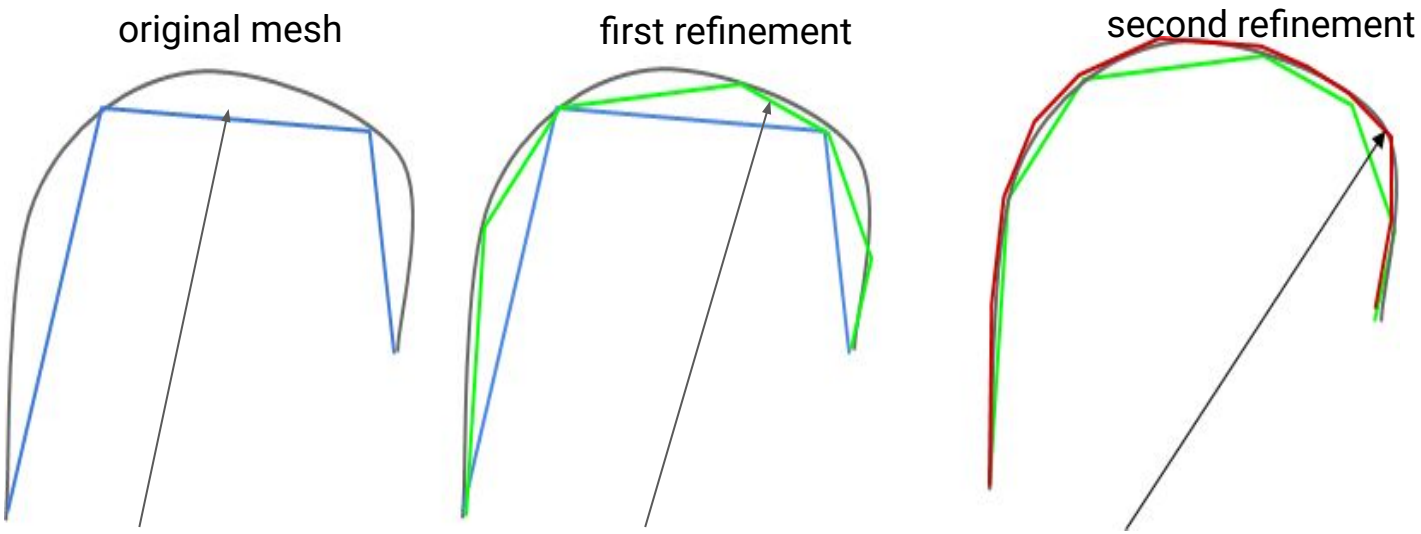
1. List the vertices for each triangle separately. That is, repeat vertices for each triangle that they bound. Provides flexibility in color and other computations
2. Have one copy of each vertex in the vertex array. Elements array provides triples of indices into the array. Optimizes on number of vertex computations

Vertex normals can be computed for the vertices of the model and sent in as attributes, to be used in the shader

Subdivision of Meshes

Subdivision defines a smooth curve (or surface) as the limit of a sequence of successive refinements. At each step, new vertices are generated and their coordinates computed based on local/neighbourhood geometry

Each refinement step produces a better piece-wise linear approximation of the “limit surface” – the hypothetical curve (or surface) obtained after an infinite number of subdivision steps.



Subdivision Surfaces

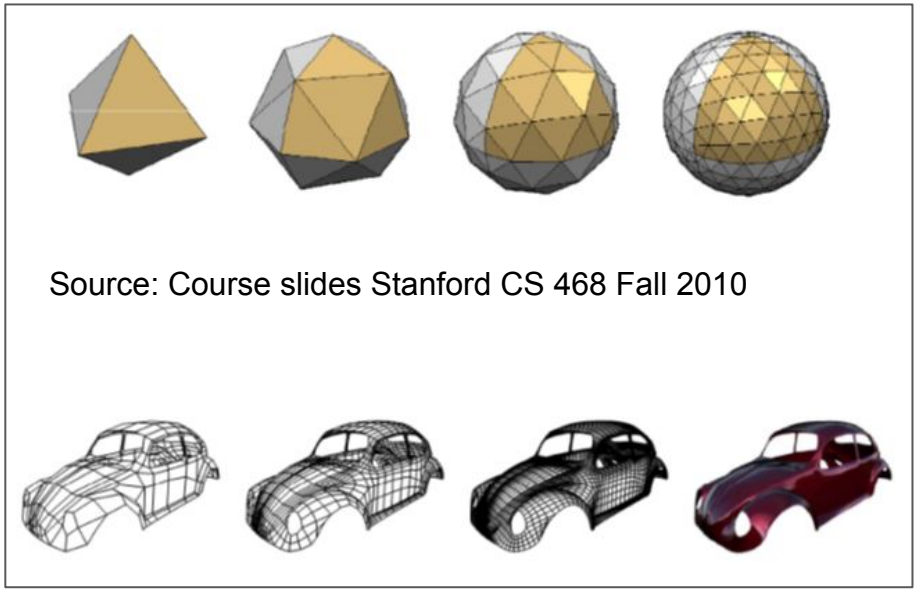
Successive refinement of meshes:

1. Add vertices in the interior of triangles.
2. Update coordinates of new vertices

Coordinates of new vertices computed through a weighted average of neighbour vertices. Existing vertices may also be “smoothed”

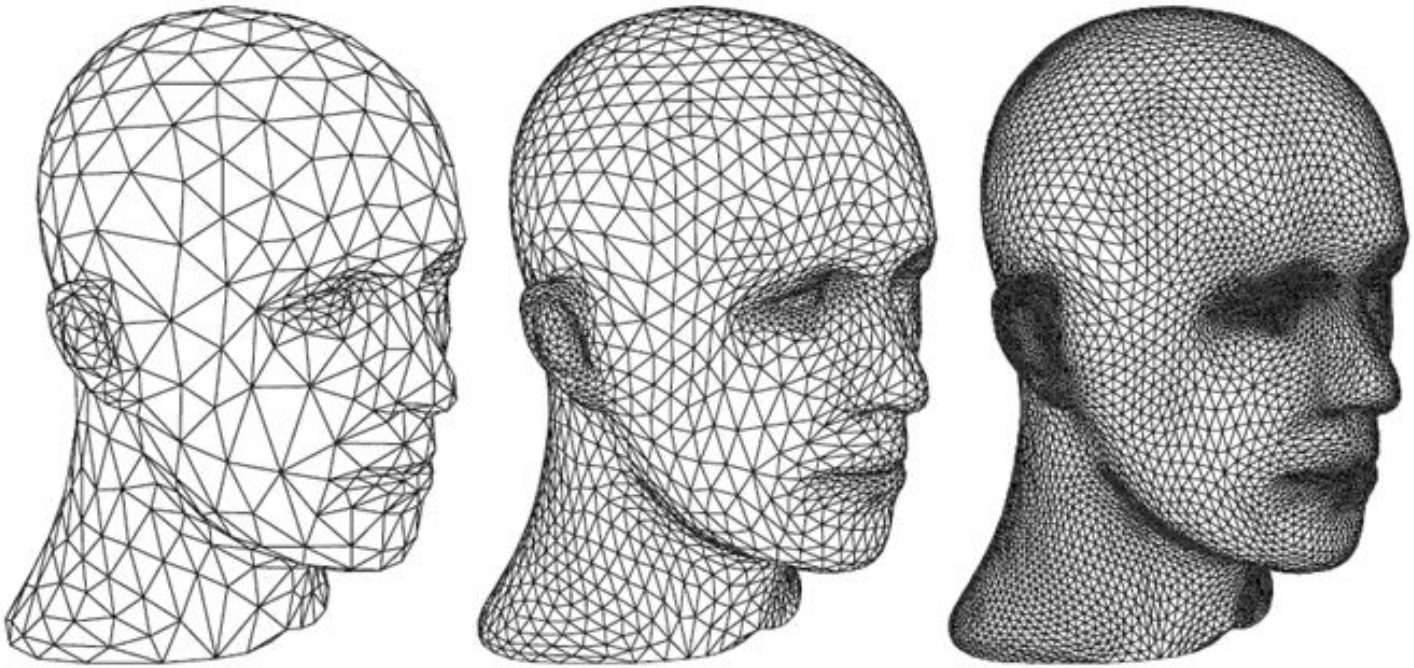
The smoothing depends on subdivision technique being used (Examples: Loop, Catmull-Clark)

Converges to a smooth surface



Source: Course slides Stanford CS 468 Fall 2010

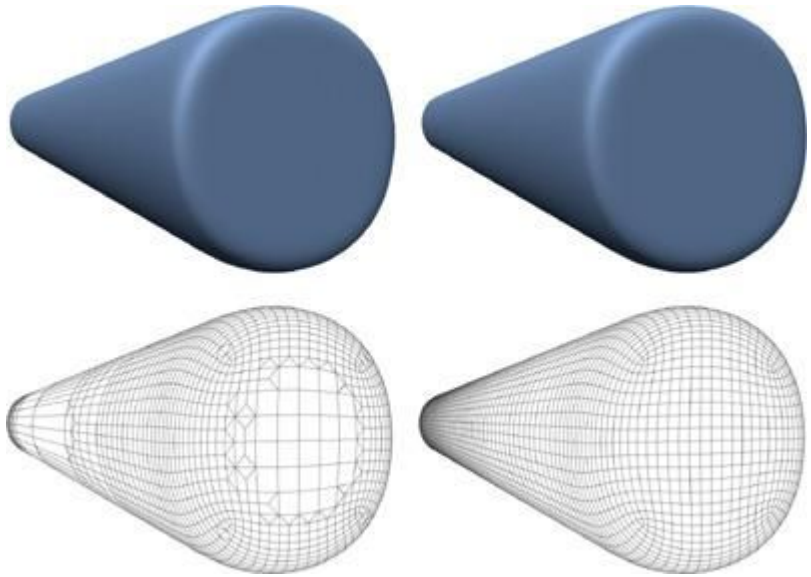
Refinement of Surface Meshes



The second and third images are produced by subdividing each triangle of the previous image into 4 triangles using a specific rule (Loop subdivision). SIGGRAPH 2000 Course Notes on Subdivision for Modeling and animation, D. Zorin and P. Shroeder

Adaptive Subdivision

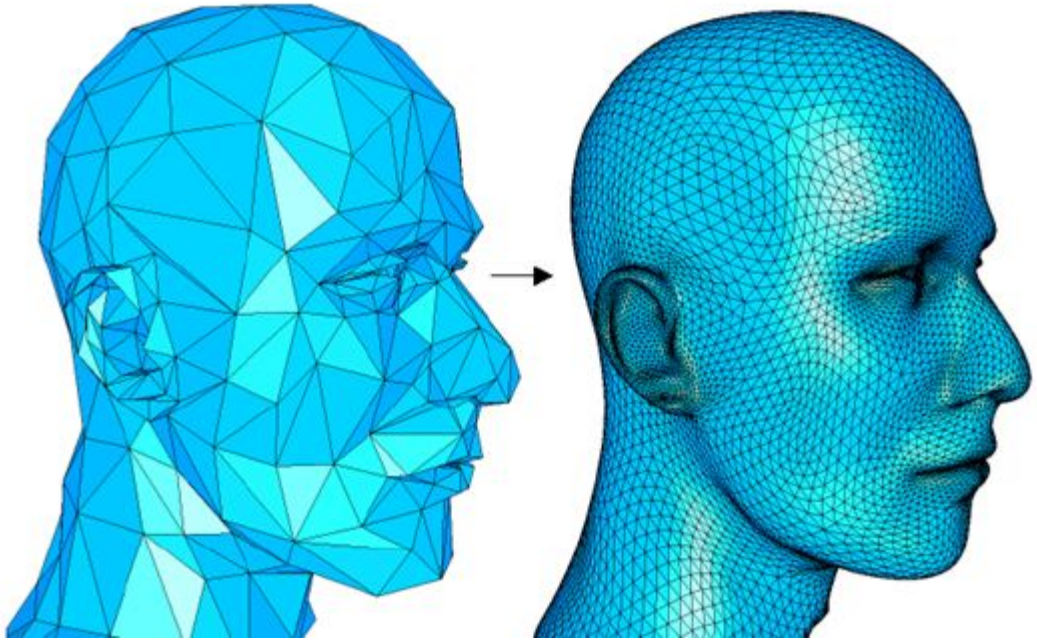
Note: subdivision need not be uniform. Certain triangles could be refined much finer than others, based on curvature or other features. Reduces size of mesh



Adaptive vs Uniform tessellation

(Source: Nvidia GPU Gems)

View dependent Adaptive Subdivision



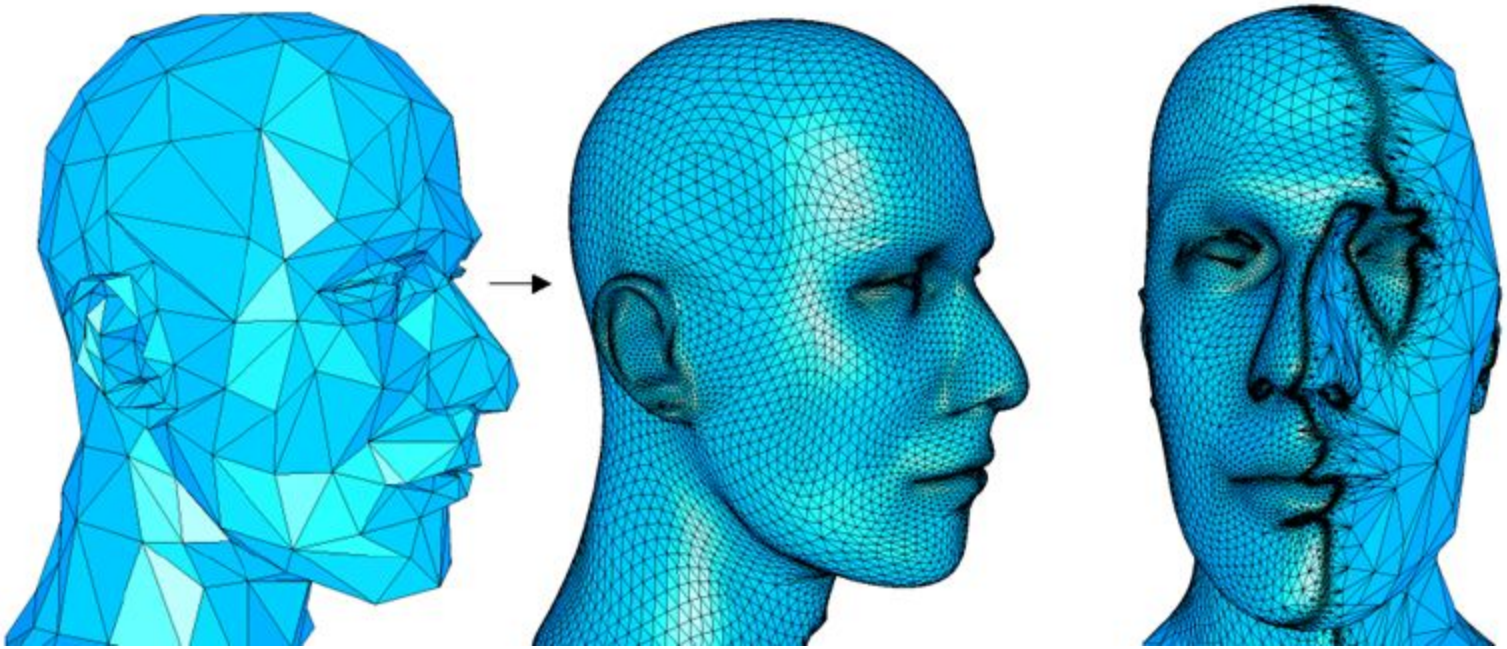
Mesh density needs to be greater where:

- significant features
- higher curvature
- region is visible
- along silhouettes
- higher pixel area on screen

From: Efficient View-Dependent Refinement of 3D Meshes using $\sqrt{3}$ – Subdivision

Pierre Alliez, Nathalie Laurent, Henri Sanson, Francis Schmitt

View dependent Adaptive Subdivision



From: Efficient View-Dependent Refinement of 3D Meshes using √3 – Subdivision

Pierre Alliez, Nathalie Laurent, Henri Sanson, Francis Schmitt

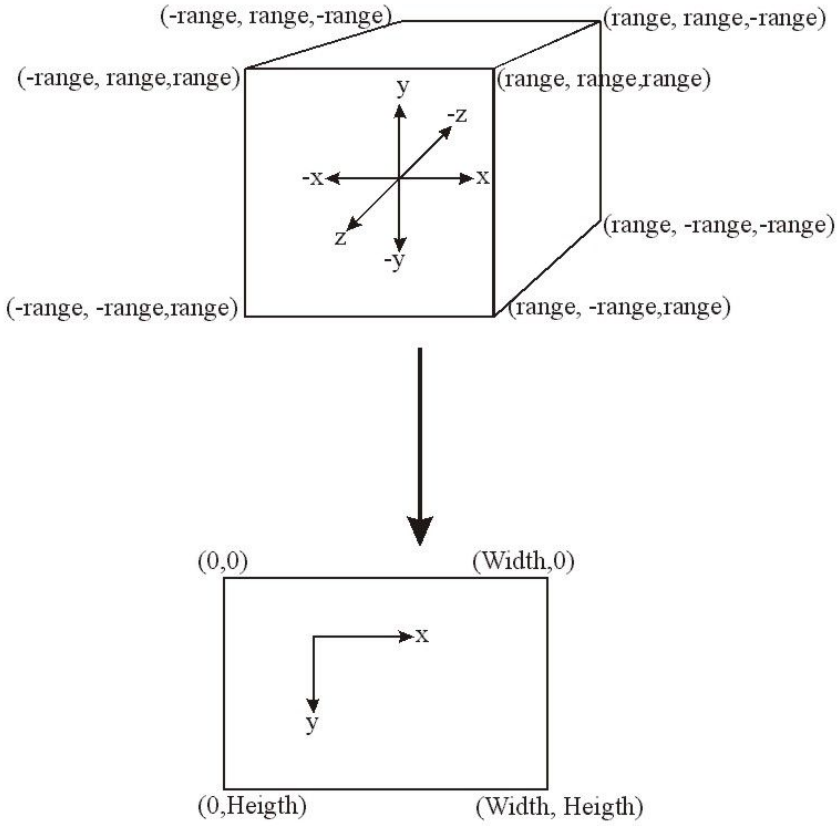
Coordinate Systems for Viewing

Coordinate Systems

Device-independent graphics: Advanced software system, where the application programmers need not worry about input/output device specifications.

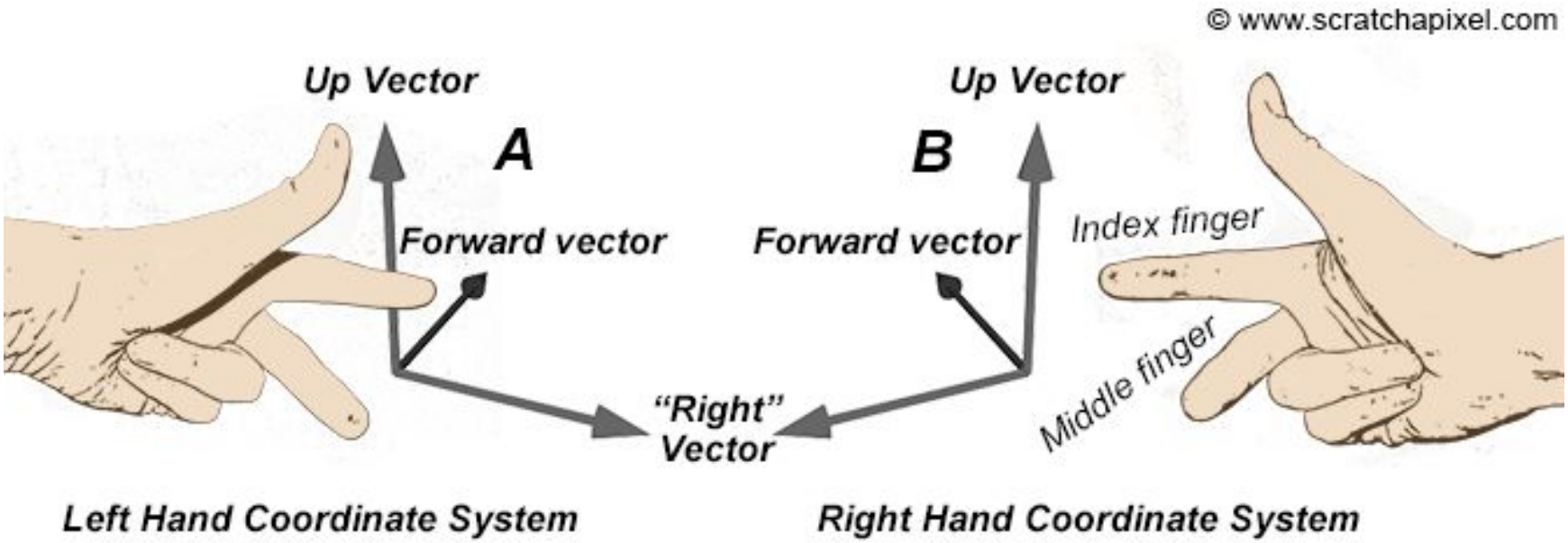
Graphics system converts world coordinates to screen coordinates.

In general, OpenGL uses right-handed coordinate system.



(Image courtesy: OpenGL Tutorial.)

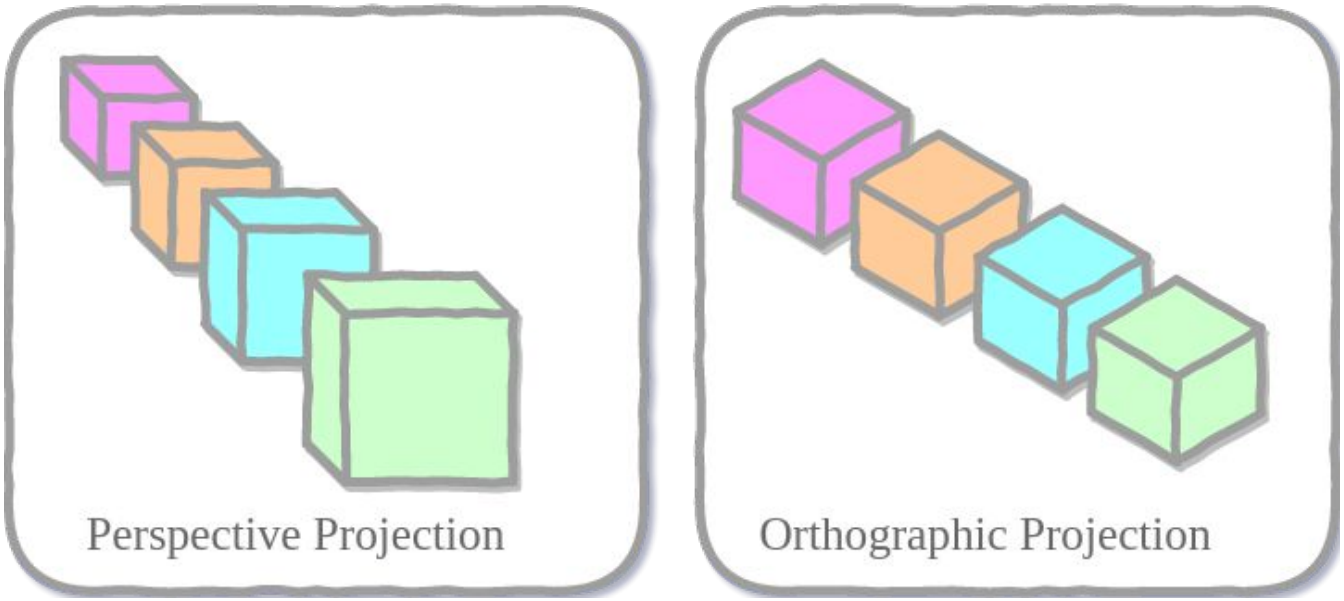
Handedness in Coordinate Systems



Coordinate Systems for defining the scene/object

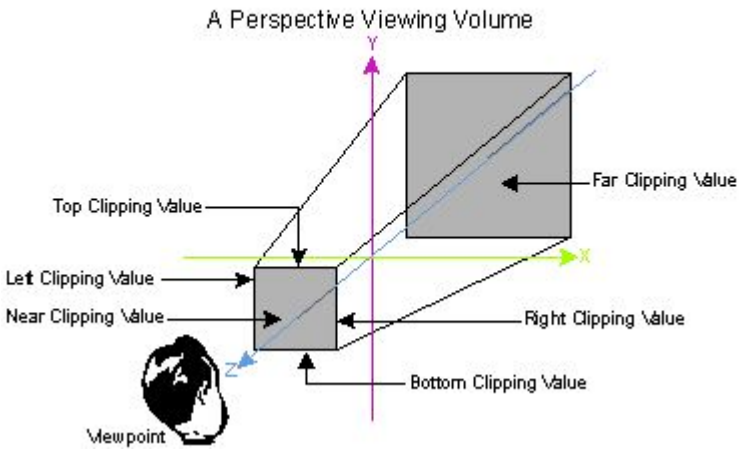
https://www.scratchapixel.com/lessons/mathematics-physics-for-computer-graphics/geometry/coordinate-systems

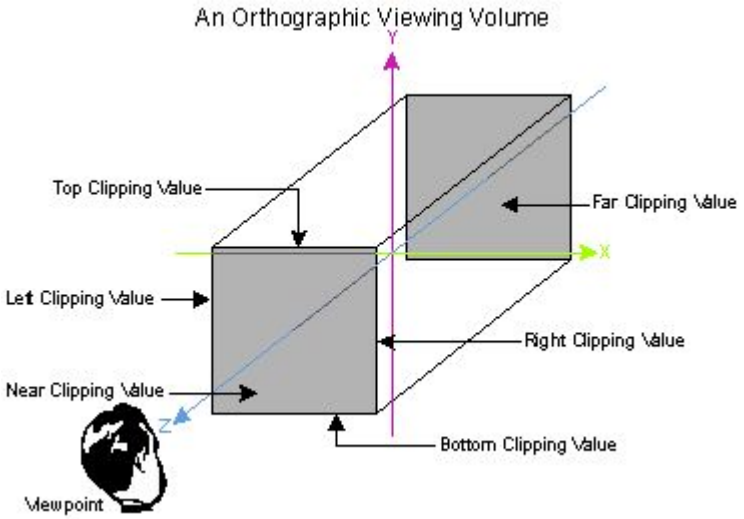
Viewing: Defining Different Views



https://techno-pundit.blogspot.com/2019/06/your-first-threejs-scene-hello-cube.html

Viewing: Defining the Viewing Volume





http://www.c-jump.com/bcc/common/Talk3/Math/Matrices/W01_0220_orthographic_projecti.htm

Viewing: Synthetic Camera Model

Synthetic-camera model ensures that object specifications and camera specifications are completely independent.

- Orthographic View: Simplest & default view.
- Camera is placed infinitely far from object.
- Center of projection replaced by direction of projection.
- Camera points towards -z axis.
- (x, y, z) gets projected to (x, y, 0) on projection plane, z=0.

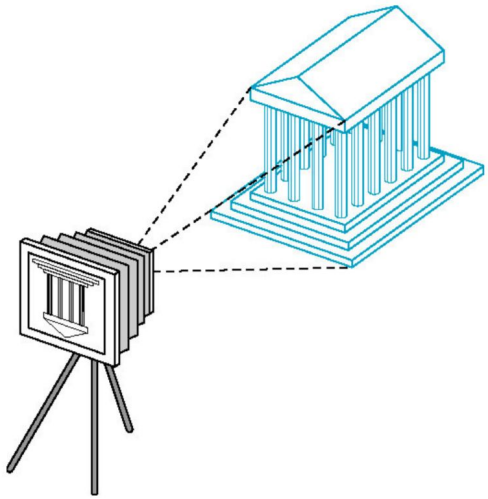


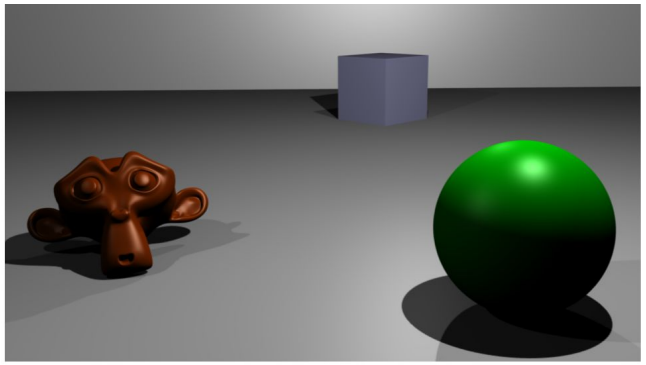
Figure 1: Model of a synthetic camera (E. Angel, Interactive Computer Graphics - 5th edition).

Hidden Surface Removal

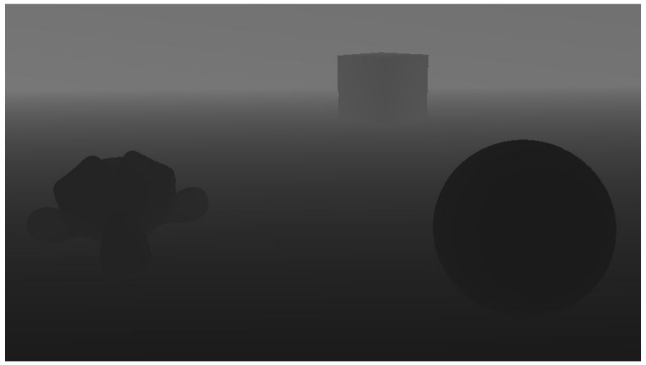
Surfaces at the back hidden by those in the front are ordered using hidden-surface removal or visible-surface algorithms.

OpenGL uses z-buffer algorithm: z-buffer is part of FB.

z-buffer is a state parameter that has to be enabled/disabled, at either the main or the initialization functions.



A simple three-dimensional scene



Z-buffer representation

(Image courtesy: Wikimedia Commons.)

Viewing Matrices

Modelview Matrix: applies to viewing and model transformation matrices. Describes where the viewer stands with the camera and the direction one points it.

Projection Matrix: applies to projection matrices. Describes the attributes of the camera, such as field of view, focal length, fisheye lens, etc.

These matrices are state parameters.

Description courtesy: OpenGL FAQ.

Aspect Ratio and Viewport

- Viewport and viewing rectangle (from field of view) should have same aspect ratio.
- Viewport is part of OpenGL state.
- Multiple viewports can be achieved by rendering an image, changing viewport, rendering next image, etc.

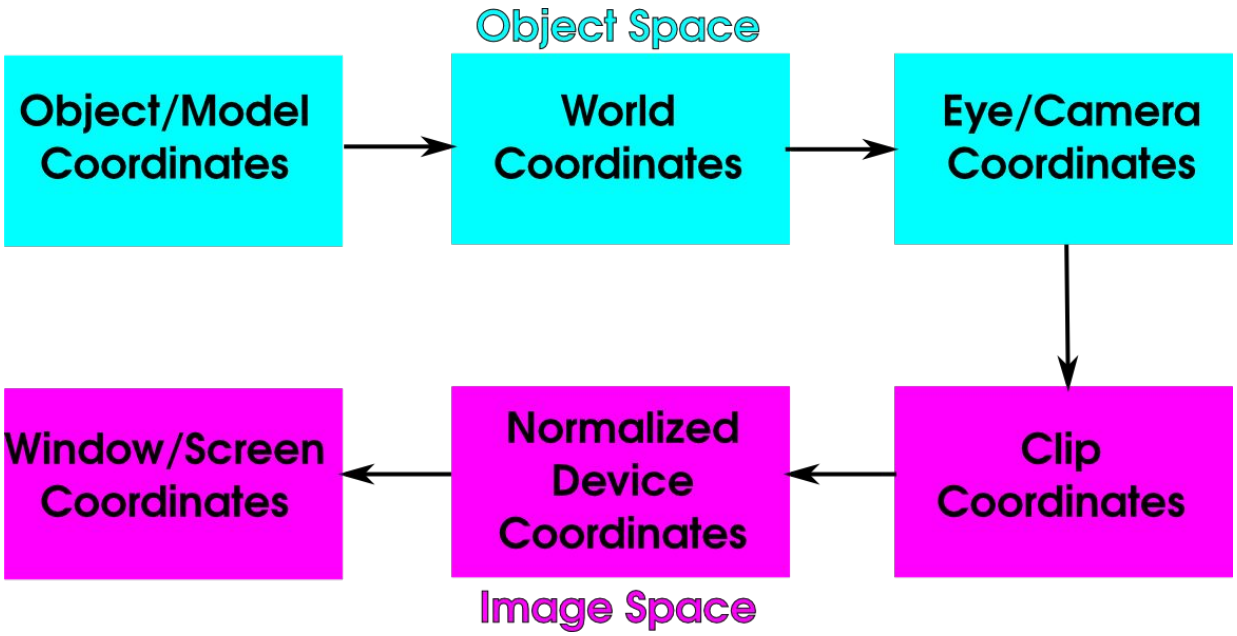
Normal (Correct AR)

Extended (Skewed AR)

Compressed (Skewed AR)

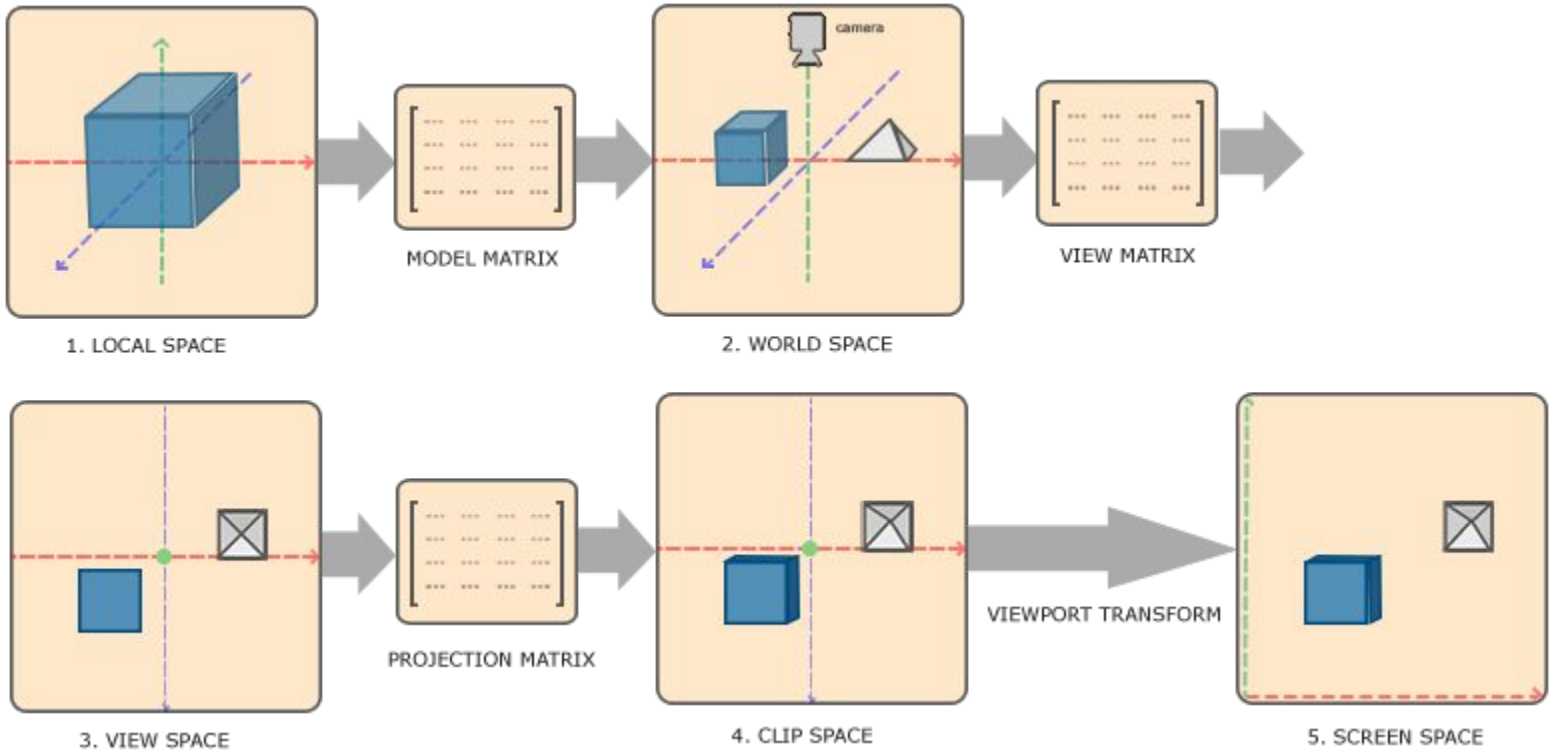
Image Courtesy: Adobe

Coordinate Systems in Computer Graphics



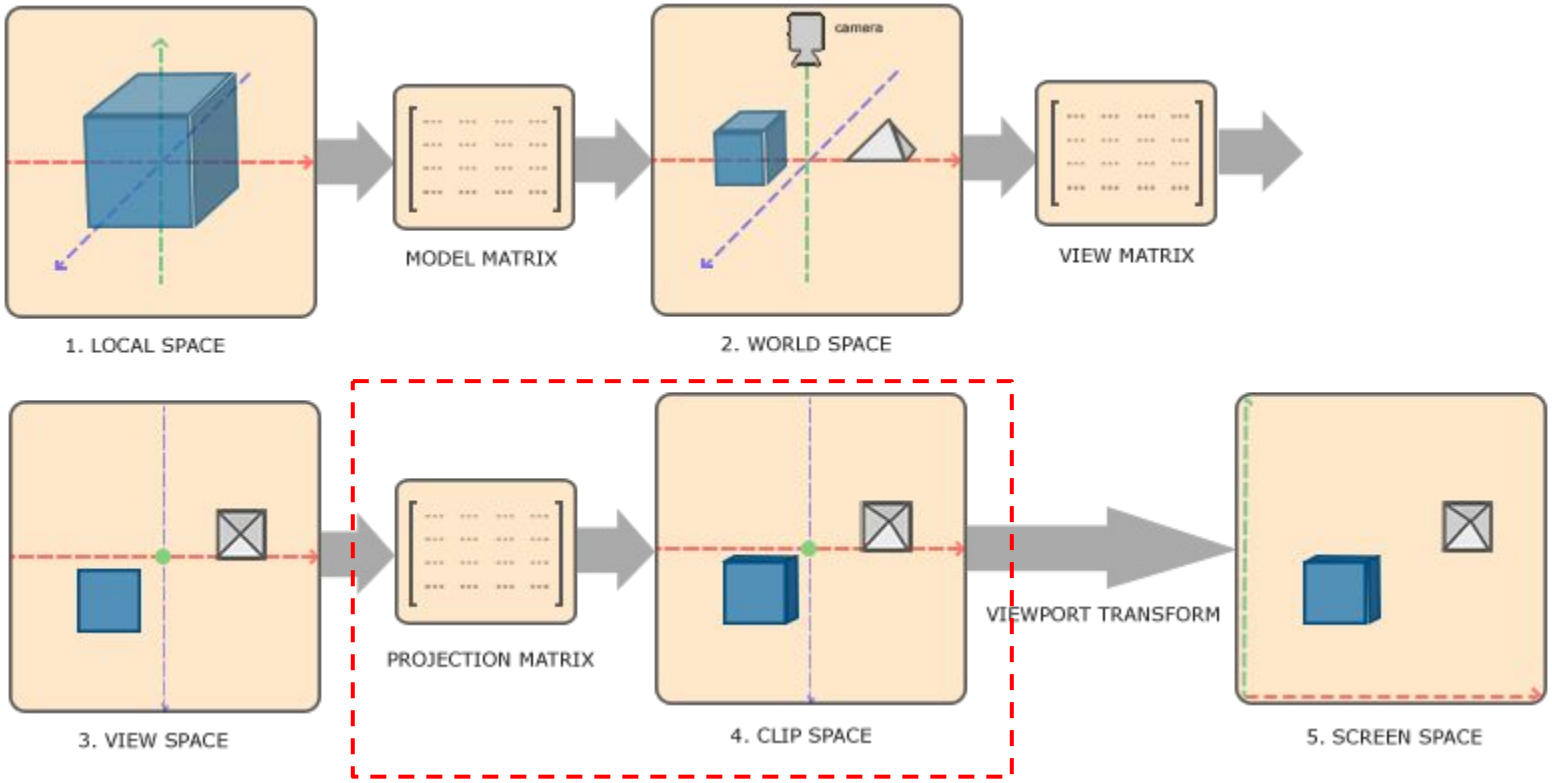
Coordinate Systems in Computer Graphics: Implementation

https://learnopengl.com/Getting-started/Coordinate-Systems



Coordinate Systems in Computer Graphics: Implementation

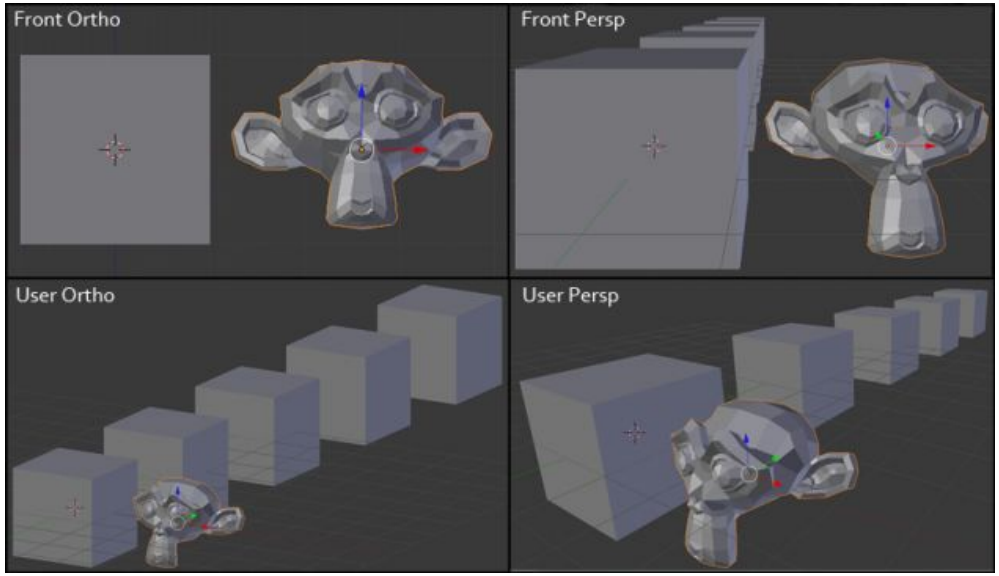
https://learnopengl.com/Getting-started/Coordinate-Systems



Orthographic vs Perspective Projection

Orthographic projection preserves the property of parallelism of lines.

Perspective projection has the concept of vanishing points, as parallel lines “appear” to meet at the vanishing point, in the horizon.



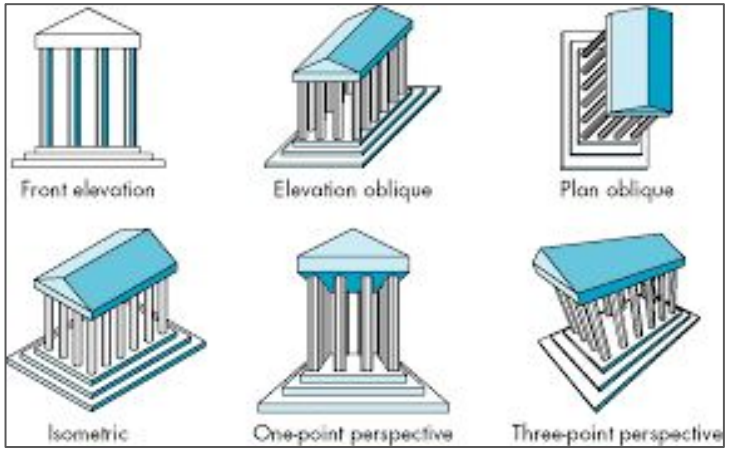
https://wiki.blender.jp/Doc:2.6/Manual/3D_interaction/Navigating/3D_View

Orthographic vs Perspective Projection

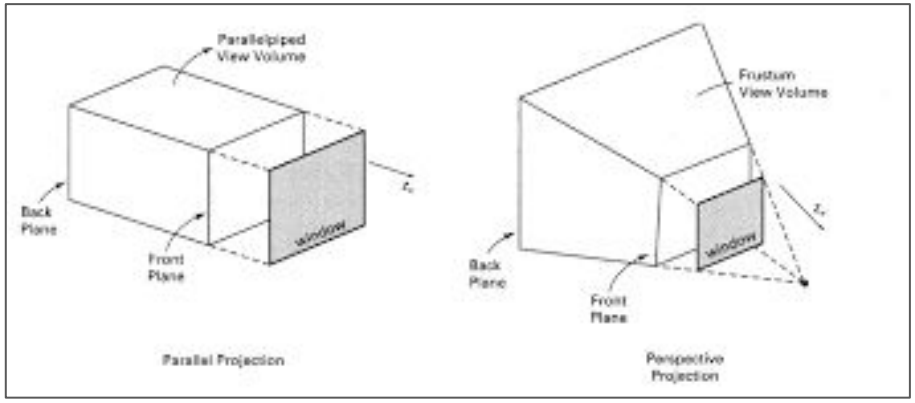
Orthographic projection preserves the property of parallelism of lines.

Perspective projection has the concept of vanishing points, as parallel lines "appear" to meet at the vanishing point, in the horizon.

The choice of projection is determined by the choice of the **viewing volume**.



Edward Angel, Textbook



http://ivl.calit2.net/wiki/images/5/55/04_VertexTransformF12.pdf

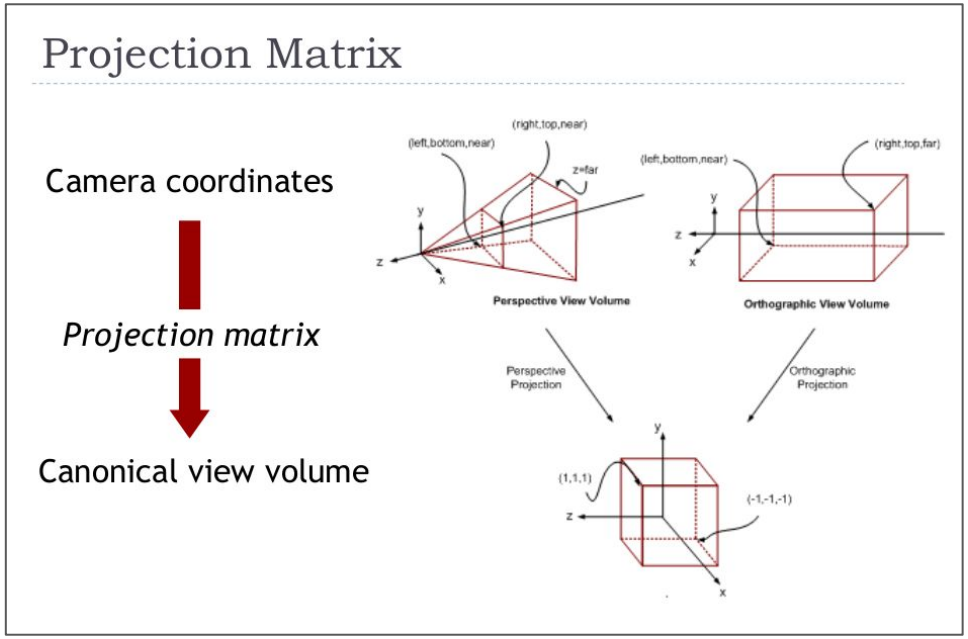
Orthographic vs Perspective Projection

Orthographic projection preserves the property of parallelism of lines.

Perspective projection has the concept of vanishing points, as parallel lines “appear” to meet at the vanishing point, in the horizon.

The choice of projection is determined by the choice of **viewing volume**.

Projection matrix implements the choice of projection.



http://ivl.calit2.net/wiki/images/5/55/04_VertexTransformF12.pdf

Orthographic vs Perspective Projection

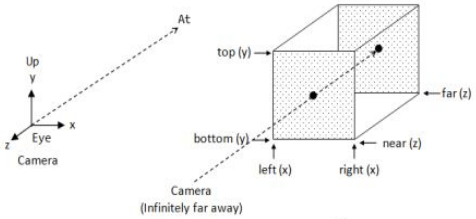
Orthographic projection preserves the property of parallelism of lines.

Perspective projection has the concept of vanishing points, as parallel lines “appear” to meet at the vanishing point, in the horizon.

The choice of projection is determined by the choice of **viewing volume**.

Projection matrix implements the choice of projection.

Orthographic Projection Matrix



In OpenGL:
glOrtho(left, right, bottom, top, near, far)

$$\mathbf{P}_{ortho}(right,left,top,bottom,near,far) = \begin{bmatrix} \frac{2}{right-left} & 0 & 0 & -\frac{right+left}{right-left} \\ 0 & \frac{2}{top-bottom} & 0 & -\frac{top+bottom}{top-bottom} \\ 0 & 0 & \frac{2}{far-near} & \frac{far+near}{far-near} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\mathbf{P}_{ortho}(width,height,near,far) = \begin{bmatrix} \frac{2}{width} & 0 & 0 & 0 \\ 0 & \frac{2}{height} & 0 & 0 \\ 0 & 0 & \frac{2}{far-near} & \frac{far+near}{far-near} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

http://ivl.calit2.net/wiki/images/5/55/04_VertexTransformF12.pdf

Orthographic vs Perspective Projection

Orthographic projection preserves the property of parallelism of lines.

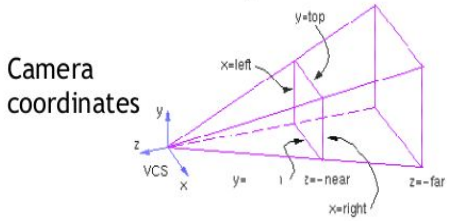
Perspective projection has the concept of vanishing points, as parallel lines “appear” to meet at the vanishing point, in the horizon.

The choice of projection is determined by the choice of **viewing volume**.

Projection matrix implements the choice of projection.

Perspective Projection Matrix

- General view frustum with 6 parameters



$$\mathbf{P}_{persp}(left, right, top, bottom, near, far) =$$

$$\begin{bmatrix} \frac{2near}{right-left} & 0 & \frac{right+left}{right-left} & 0 \\ 0 & \frac{2near}{top-bottom} & \frac{top+bottom}{top-bottom} & 0 \\ 0 & 0 & \frac{-(far+near)}{far-near} & \frac{-2far \cdot near}{far-near} \\ 0 & 0 & -1 & 0 \end{bmatrix}$$

In OpenGL:
glFrustum(left, right, bottom, top, near, far)

http://ivl.calit2.net/wiki/images/5/55/04_VertexTransformF12.pdf

Orthographic vs Perspective Projection

Orthographic projection preserves the property of parallelism of lines.

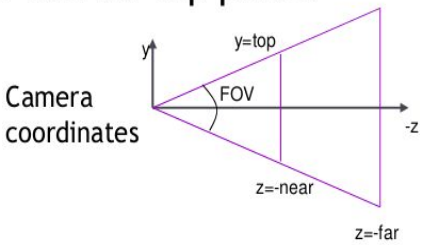
Perspective projection has the concept of vanishing points, as parallel lines “appear” to meet at the vanishing point, in the horizon.

The choice of projection is determined by the choice of **viewing volume**.

Projection matrix implements the choice of projection.

Perspective Projection Matrix

- Symmetrical view frustum with field of view, aspect ratio, near and far clip planes



$$\mathbf{P}_{persp}(FOV, aspect, near, far) = \begin{bmatrix} \frac{1}{aspect \cdot \tan(FOV/2)} & 0 & 0 & 0 \\ 0 & \frac{1}{\tan(FOV/2)} & 0 & 0 \\ 0 & 0 & \frac{near+far}{near-far} & \frac{2 \cdot near \cdot far}{near-far} \\ 0 & 0 & -1 & 0 \end{bmatrix}$$

In OpenGL:
gluPerspective(fov, aspect, near, far)

http://ivl.calit2.net/wiki/images/5/55/04_VertexTransformF12.pdf

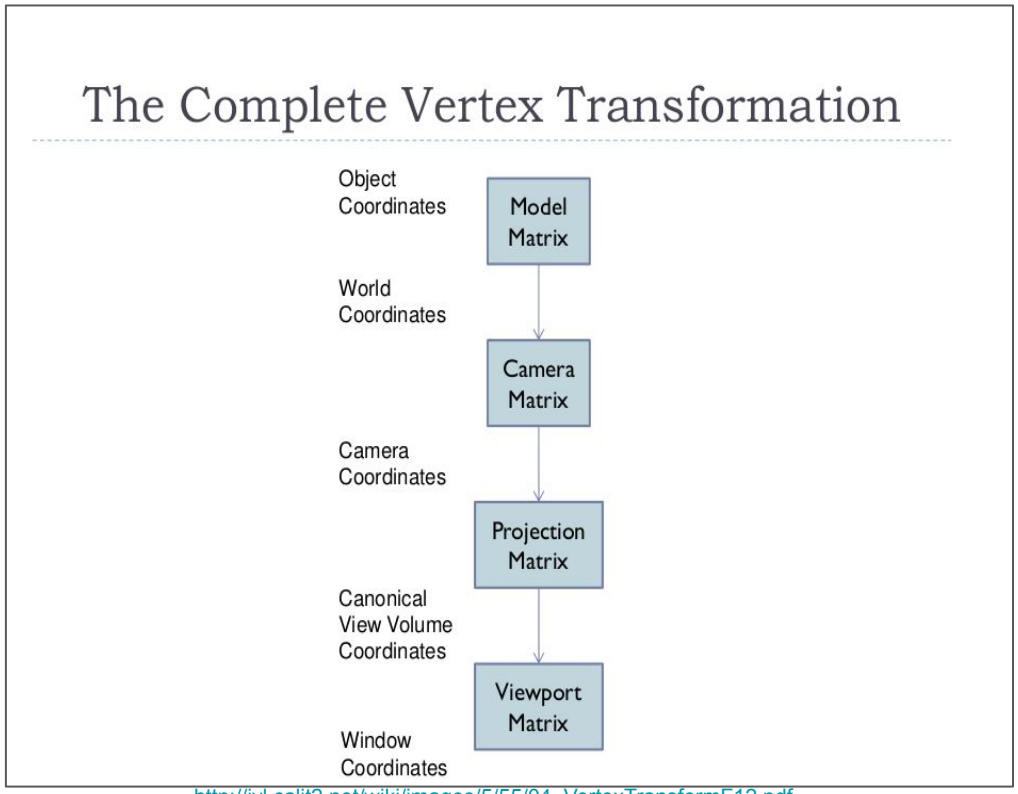
Orthographic vs Perspective Projection

Orthographic projection preserves the property of parallelism of lines.

Perspective projection has the concept of vanishing points, as parallel lines “appear” to meet at the vanishing point, in the horizon.

The choice of projection is determined by the choice of **viewing volume**.

Projection matrix implements the choice of projection.

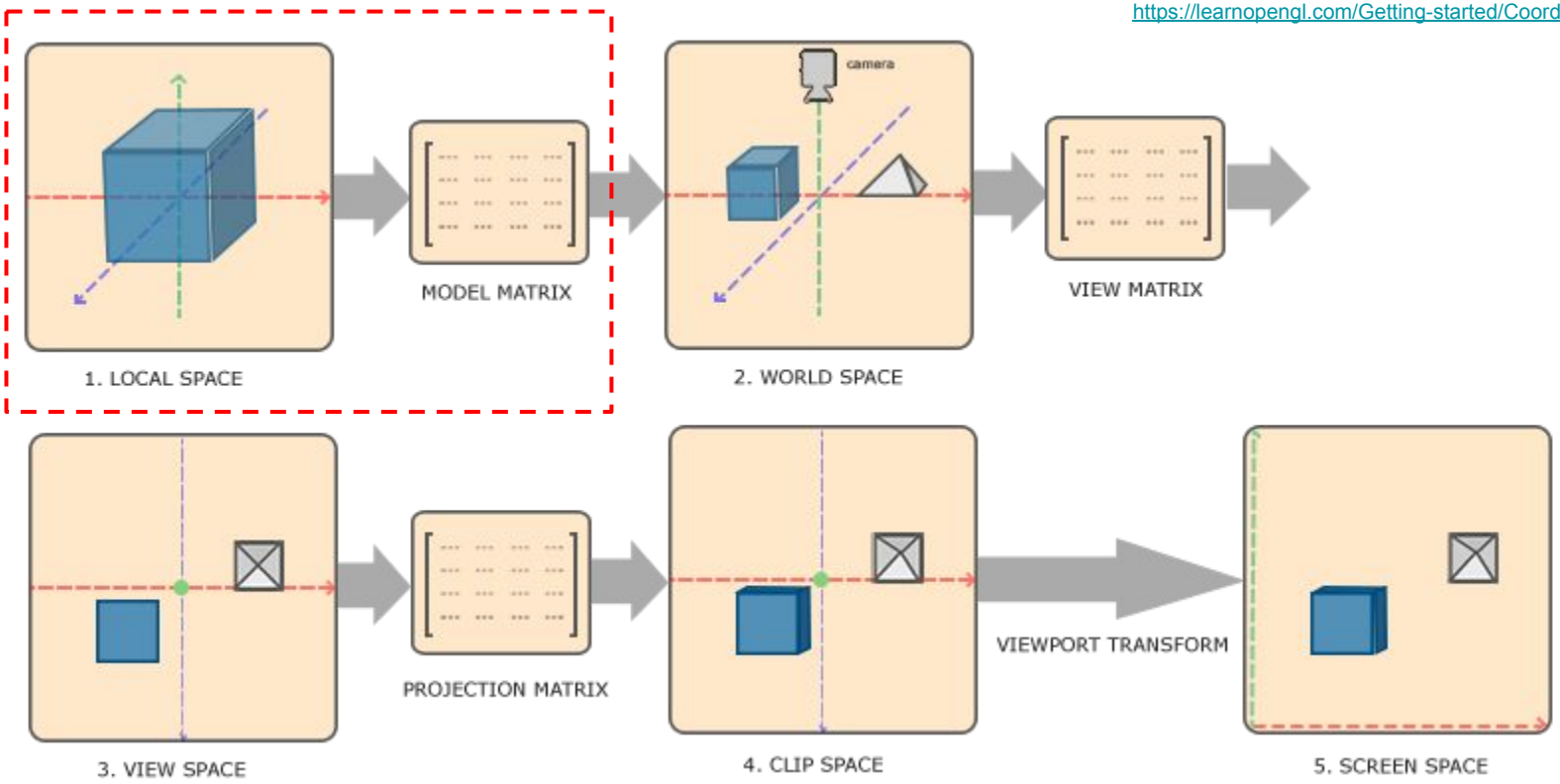


http://ivl.calit2.net/wiki/images/5/55/04_VertexTransformF12.pdf

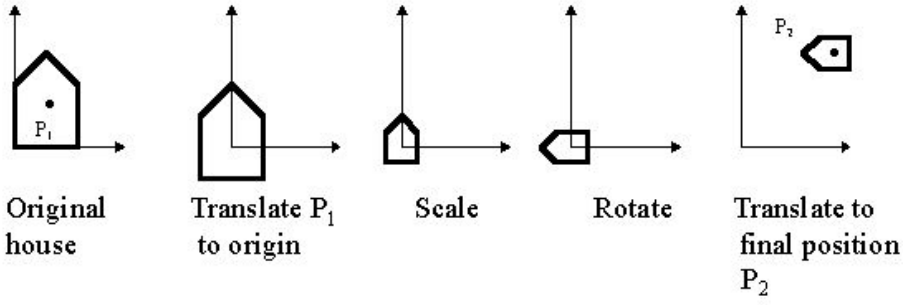
Model Transformations

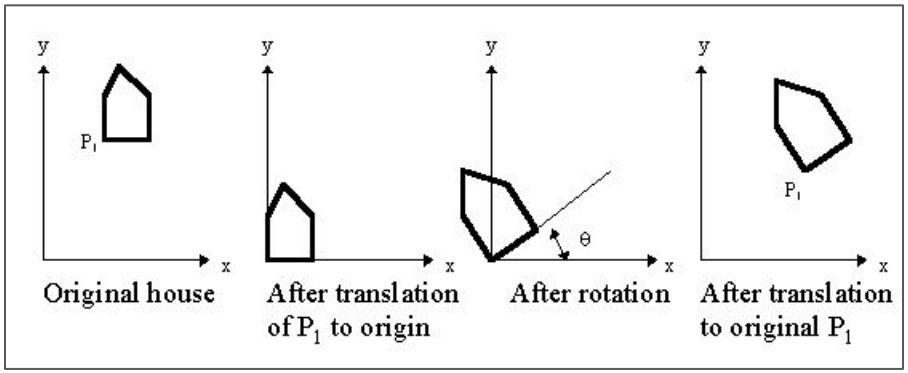
Coordinate Systems in Computer Graphics: Implementation

https://learnopengl.com/Getting-started/Coordinate-Systems

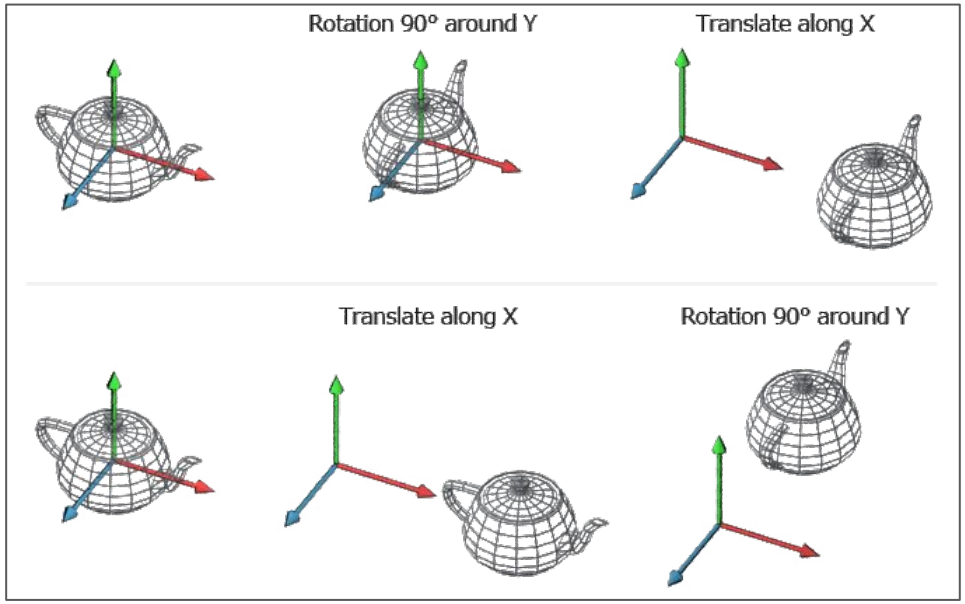


Model Transformations: Examples - 2D and 3D





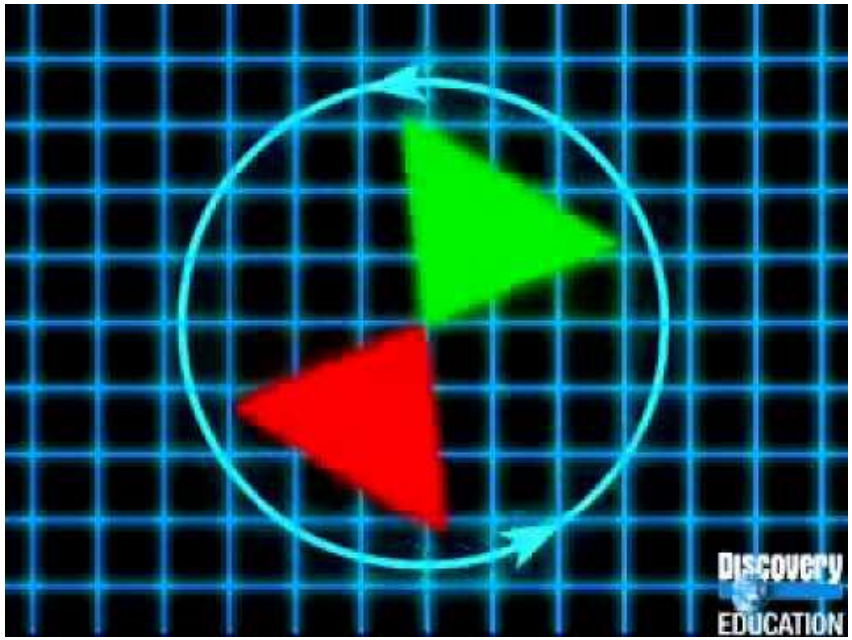
Model transformations fall under the class of affine transformations.



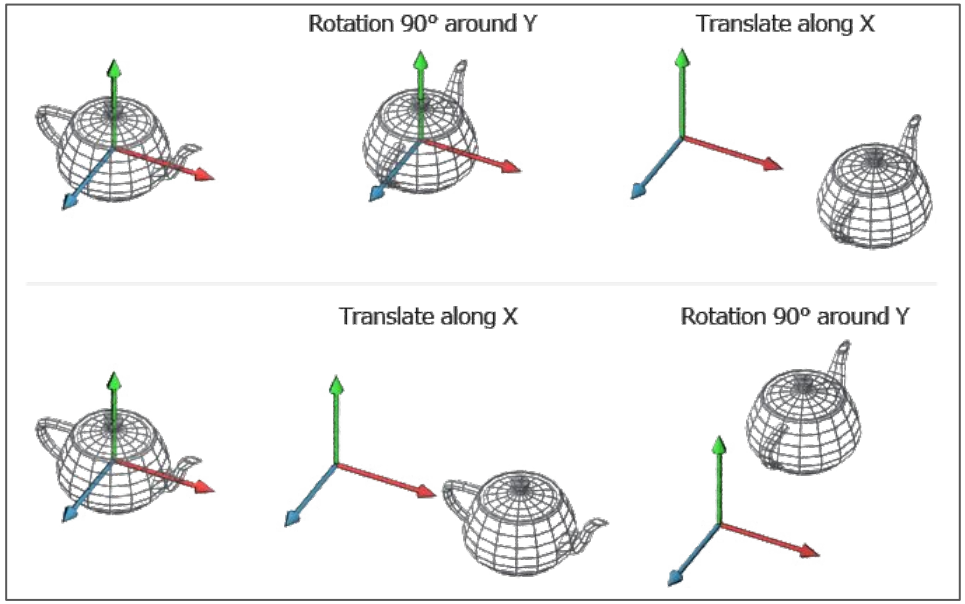
https://www.docjava.com/java/javalect/ub/cs411x/lect11/lect11.htm

http://www.codinglabs.net/article_world_view_projection_matrix.aspx

Model Transformations: Examples - 2D and 3D



Model transformations fall under the class of affine transformations.



https://www.youtube.com/watch?v=NY2cDTpsvBA

http://www.codinglabs.net/article_world_view_projection_matrix.aspx

Topics Covered Today

- Mesh Surfaces
  - Representation
  - Rendering in WebGL
  - Subdivision Surfaces
- Coordinate systems for viewing
  - Handedness
  - Viewing – projection and camera – matrices
  - Hidden surface removal
  - Viewing – display/screen
  - Coordinate systems in CG
    - Implementation
  - Projection Transformations
- Model Transformations
  - Sequence of affine transformations