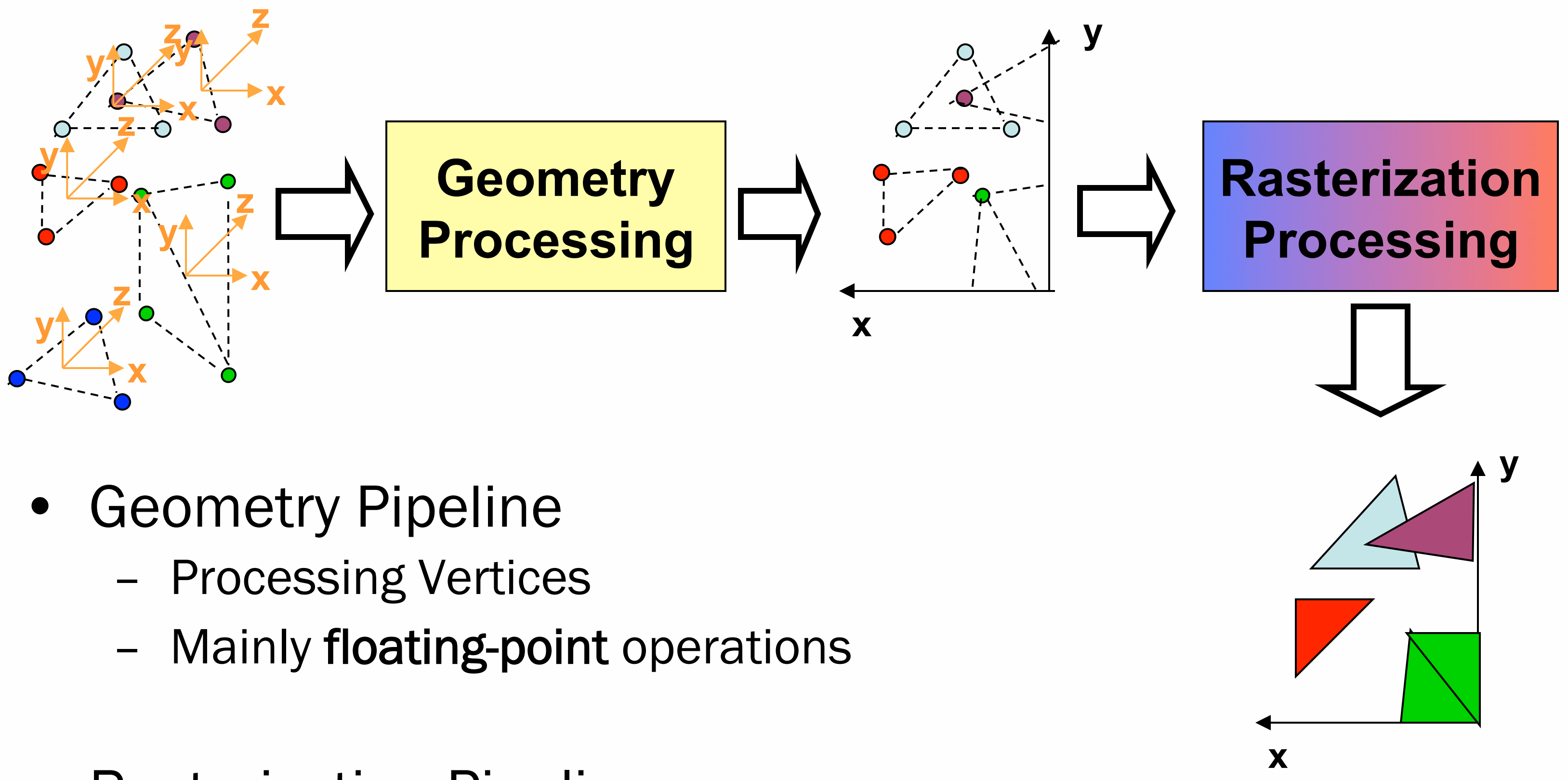
ECE4893A/CS4803MPG: Multicore and GPU Programming for Video Games

3D Coordinates & Transformations

Prof. Aaron Lanterman
(Based on slides by Prof. Hsien-Hsin Sean Lee)
School of Electrical and Computer Engineering
Georgia Institute of Technology

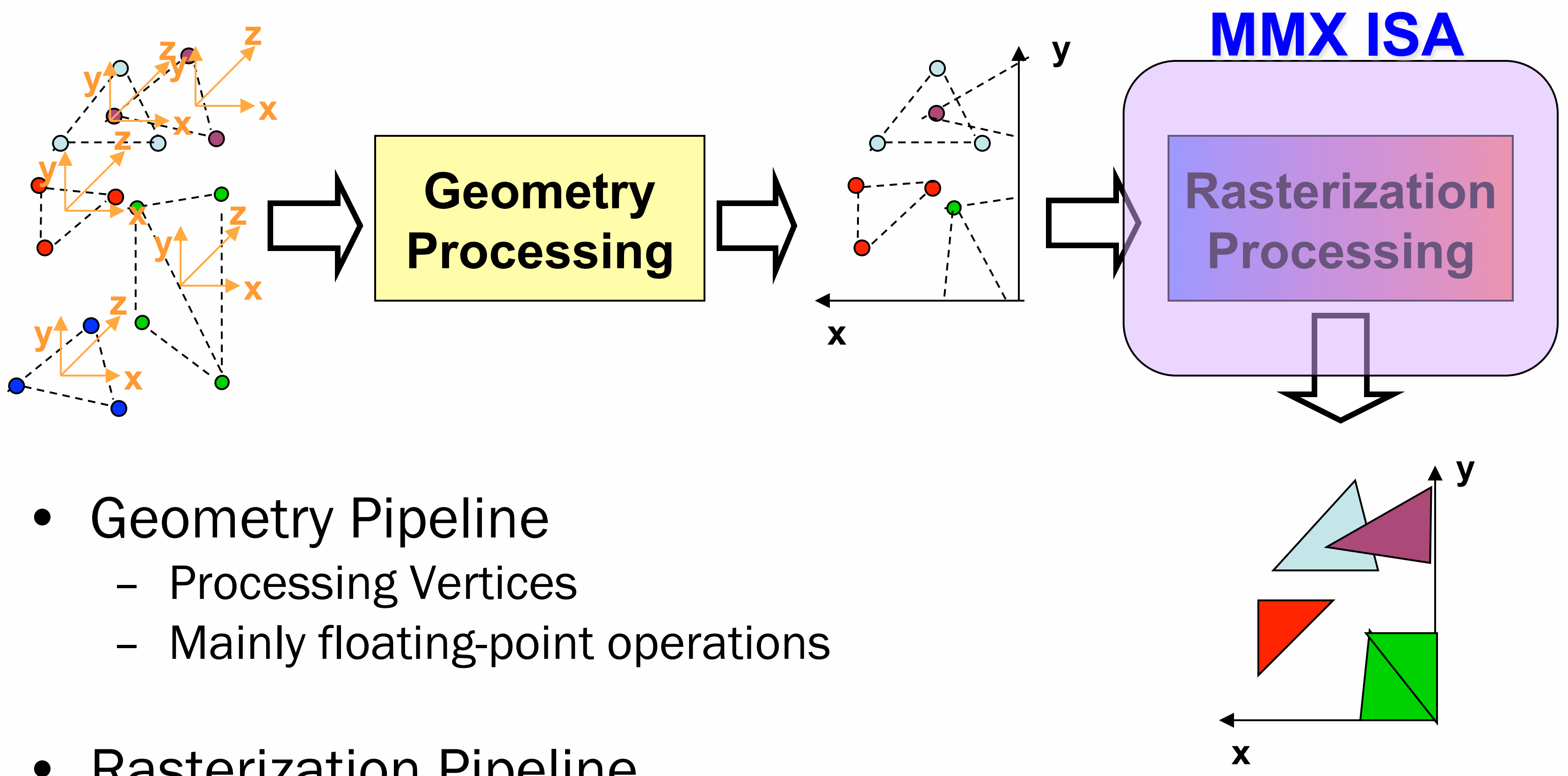
Georgia Institute of Technology

3D graphics rendering pipeline (1)



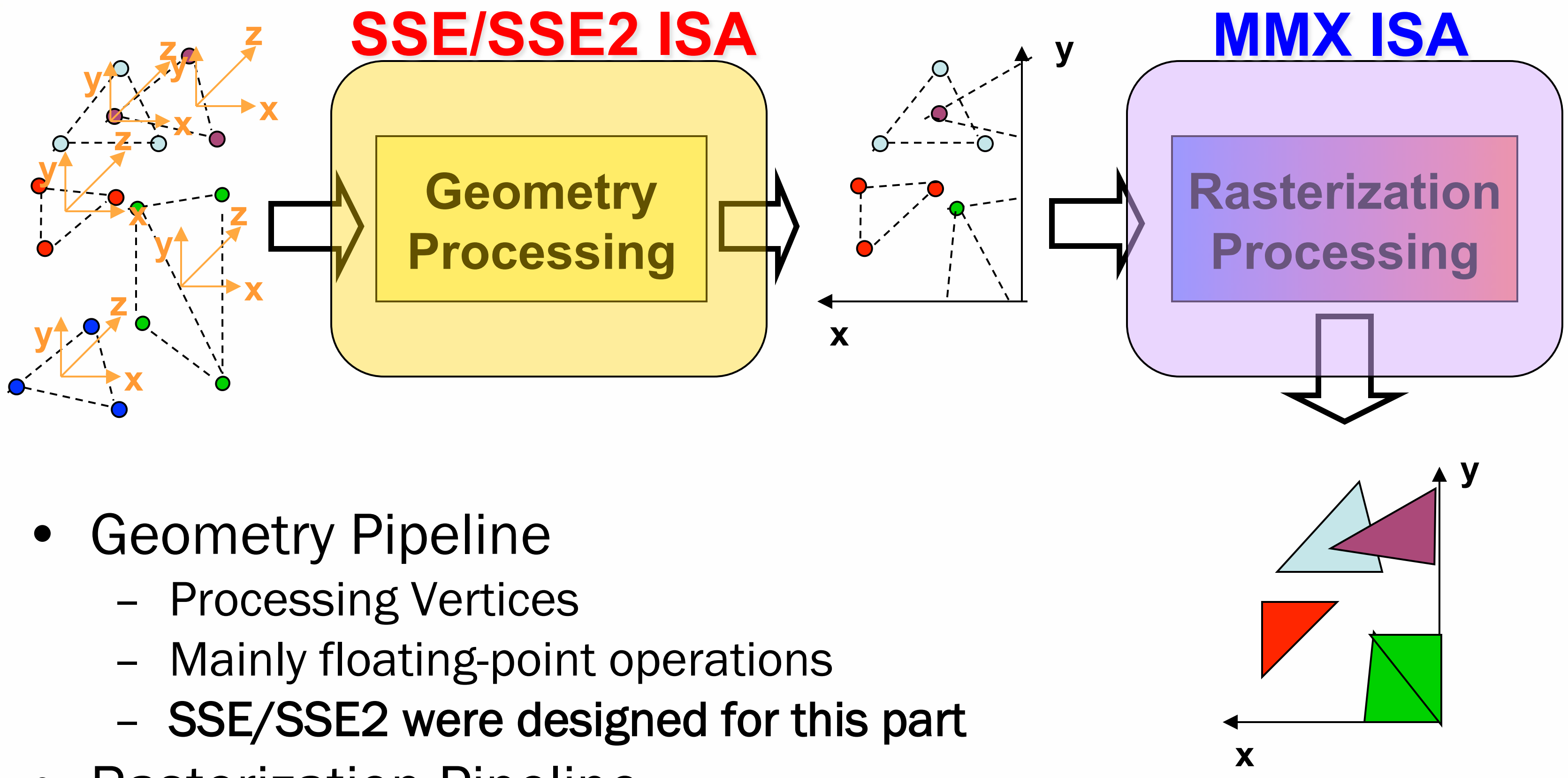
- Geometry Pipeline
  - Processing Vertices
  - Mainly **floating-point** operations
- Rasterization Pipeline
  - Processing Pixels
  - Mainly dealing with **Integer** operations

3D graphics rendering pipeline (2)



- Geometry Pipeline
  - Processing Vertices
  - Mainly floating-point operations
- Rasterization Pipeline
  - Processing Pixels
  - Mainly dealing with Integer operations
  - **MMX was originally designed to accelerate this type of functionality**

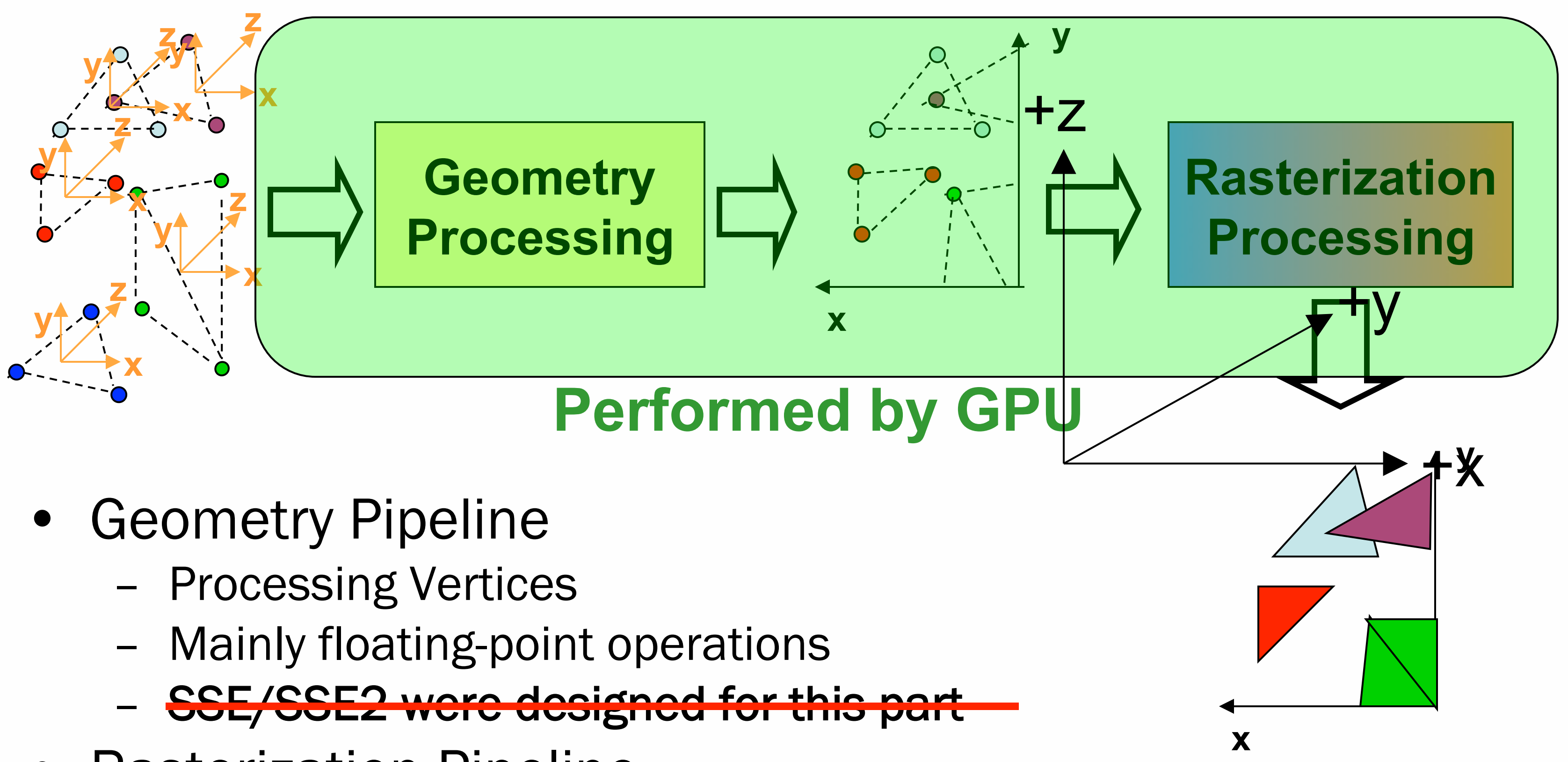
3D graphics rendering pipeline (3)



- Geometry Pipeline
  - Processing Vertices
  - Mainly floating-point operations
  - SSE/SSE2 were designed for this part
- Rasterization Pipeline
  - Processing Pixels
  - Mainly dealing with Integer operations
  - MMX was originally designed to accelerate this type of functionality

Georgia Institute of Technology

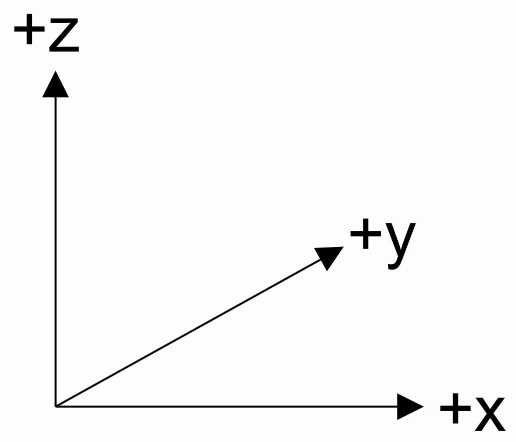
Fixed-function 3D graphics pipeline



- Geometry Pipeline
  - Processing Vertices
  - Mainly floating-point operations
  - ~~SSE/SSE2 were designed for this part~~
- Rasterization Pipeline
  - Processing Pixels
  - Mainly dealing with Integer operations
  - ~~MMX was originally designed to accelerate this type of functio~~nality

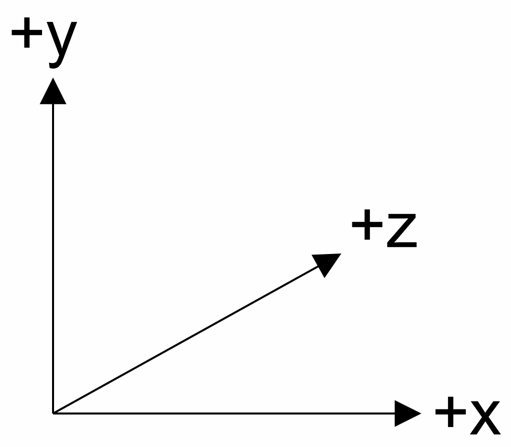
Georgia Institute of Technology

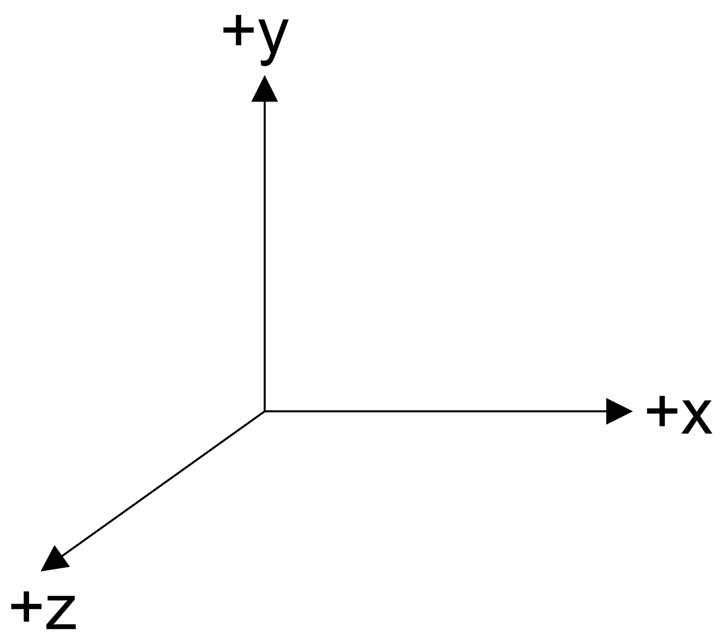
3D Coord: Math textbooks use z-up



Z-up, Right-Handed System

3D Coord: Real games tend to use y-up





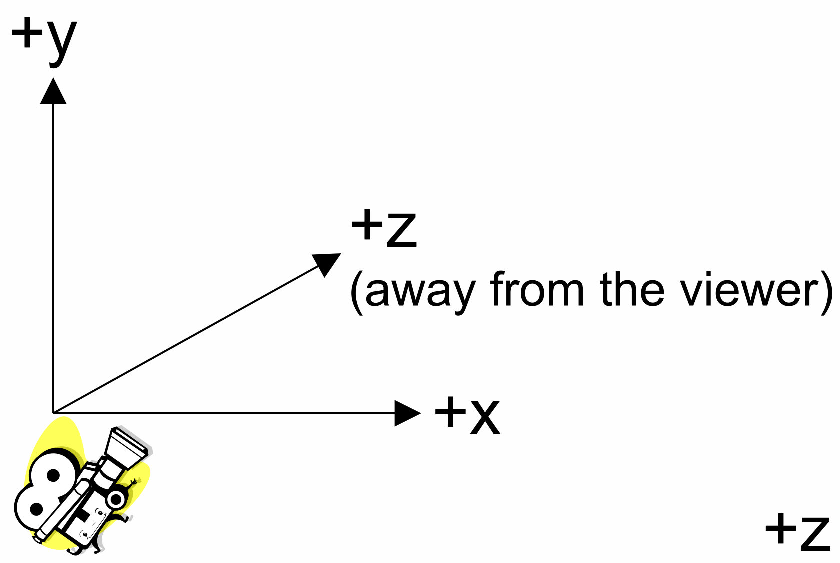
Left-Handed System

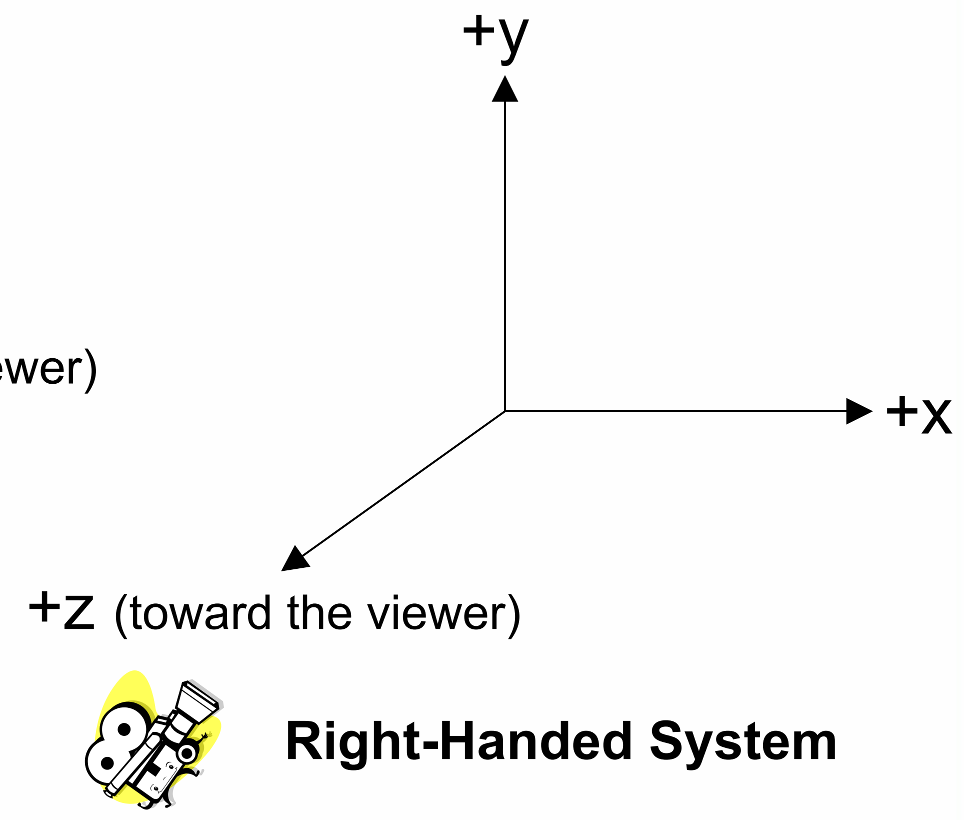
- Direct3D
- Unity3D

Right-Handed System

- OpenGL
- XNA

X-Y natural for screen coordinates





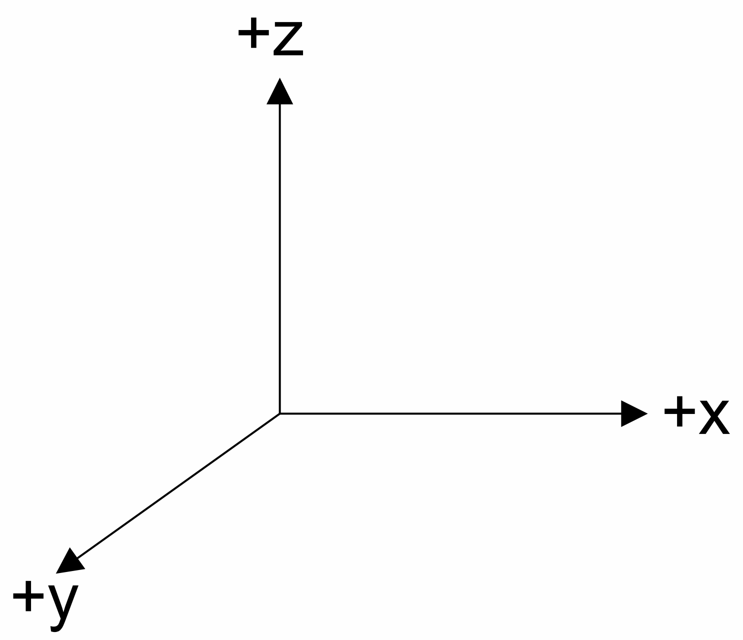
Left-Handed System

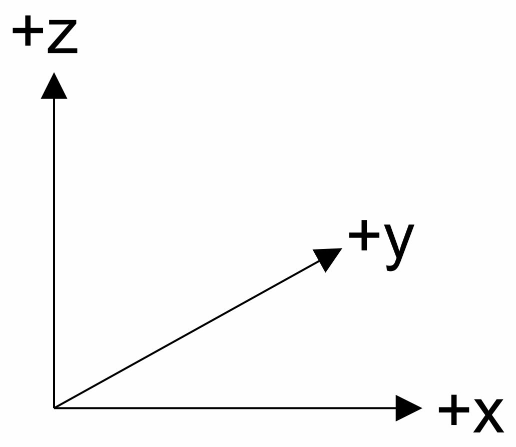
- Direct3D
- Unity3D

Right-Handed System

- OpenGL
- XNA

Some use Z-up for world coordinates



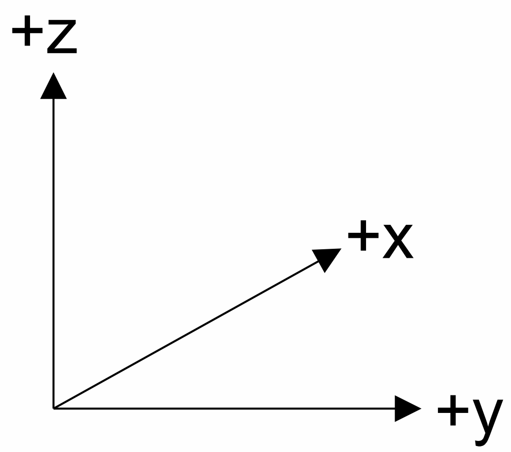


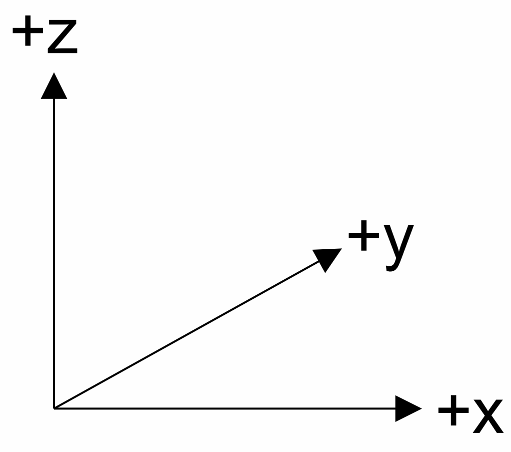
Left-Handed System **Right-Handed System**

- Z-up, LHS: Unreal
- Z-up, RHS: Quake/Radiant, Source/Hammer, C4 Engine
- Nearly everything still uses Y-up for screen coordinates!

Georgia Institute of Technology

Another view





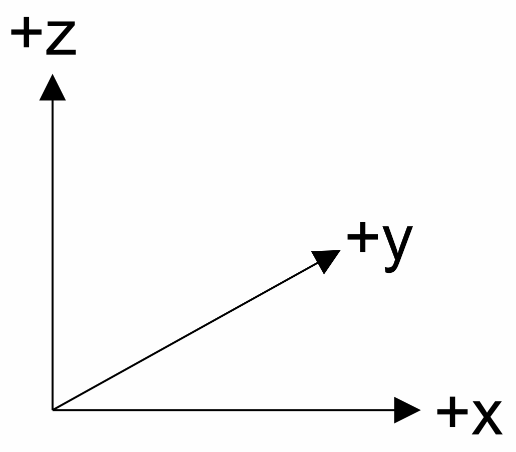
Left-Handed System

- Unreal

Right-Handed System

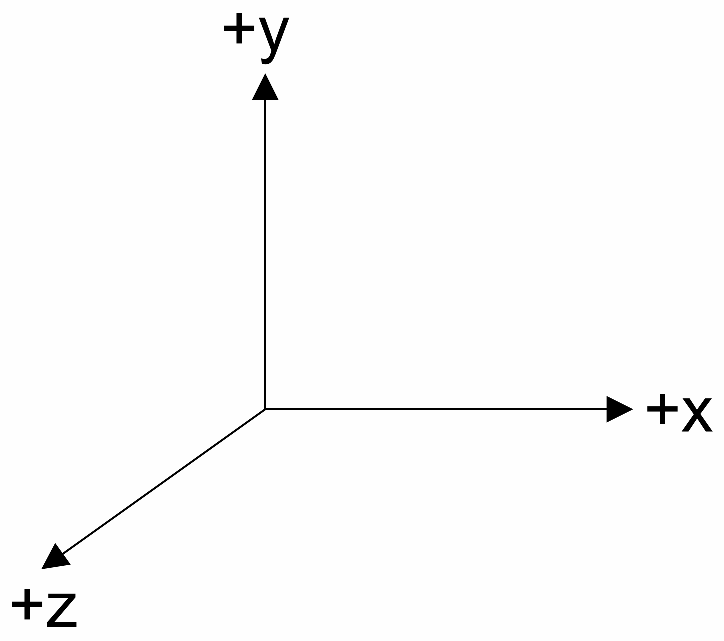
- Quake/Radiant
- Source/Hammer
- C4 Engine

3D "object" modeling software



Right-Handed System

3D Studio Max, Blender

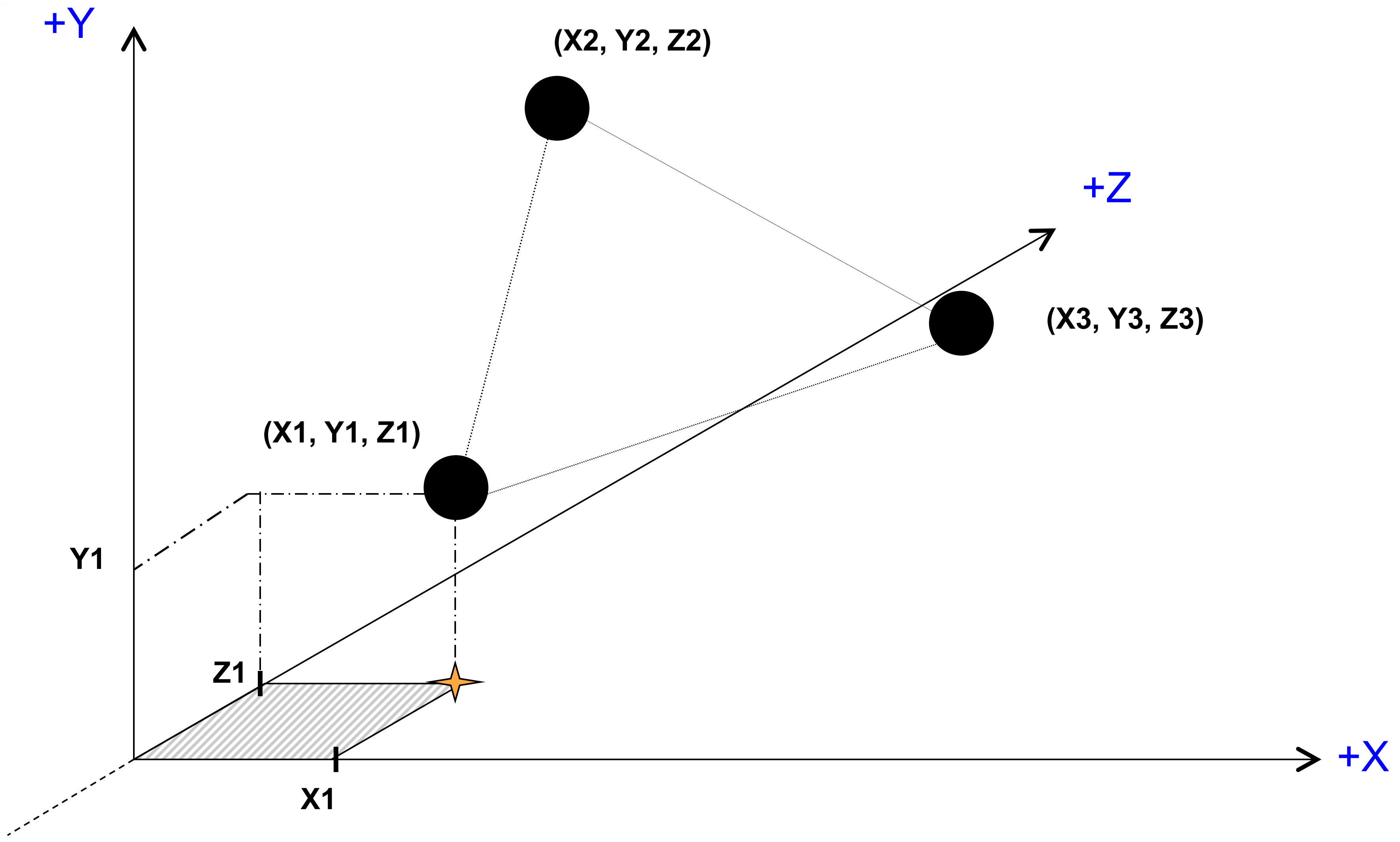


Right-Handed System

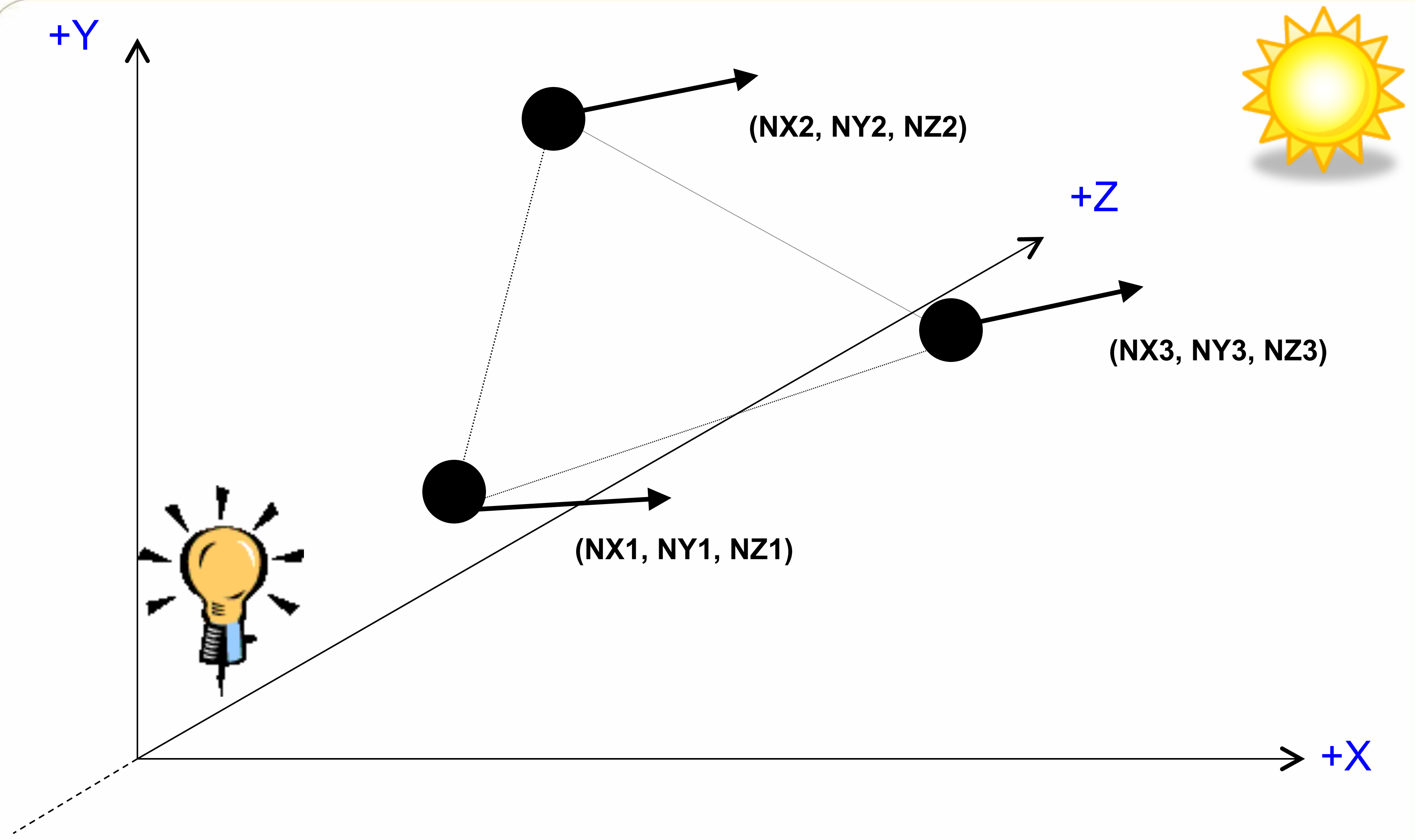
Maya, Milkshape

Georgia Institute of Technology

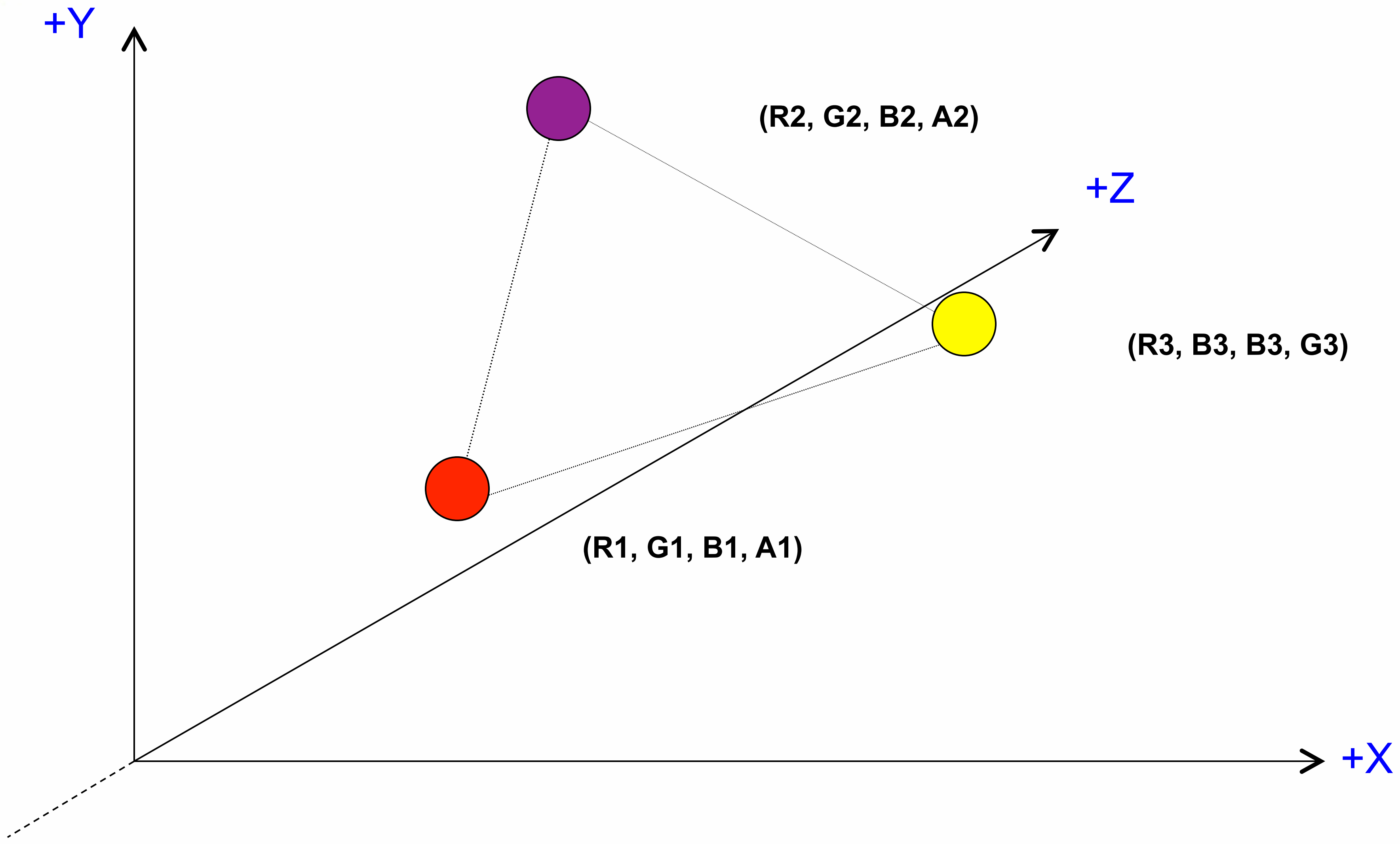
Geometry format – vertex coordinates



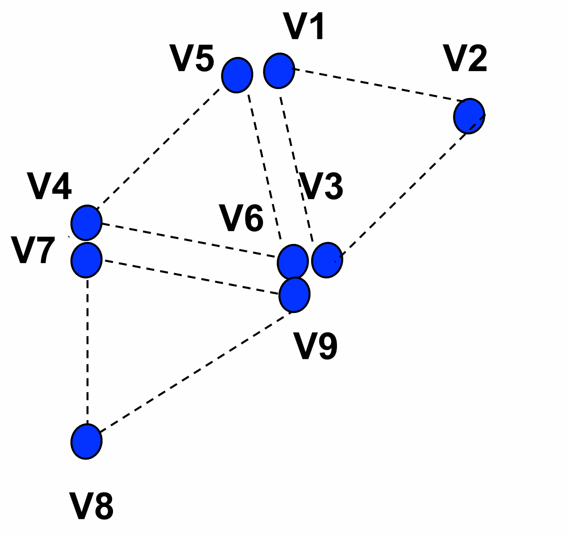
Geometry format – vertex normals



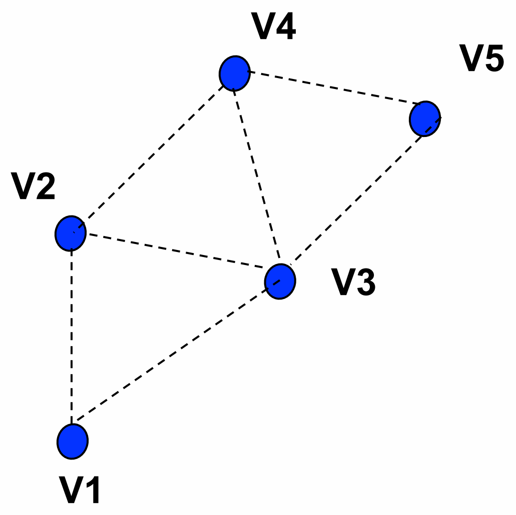
Geometry format – vertex colors



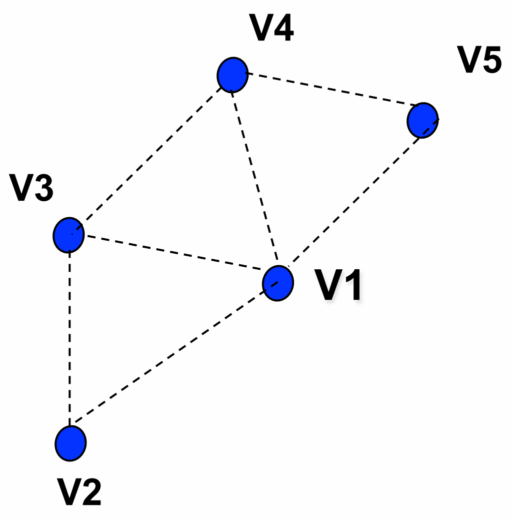
Triangle-based geometry representation



Triangle List
(note the vertex order)



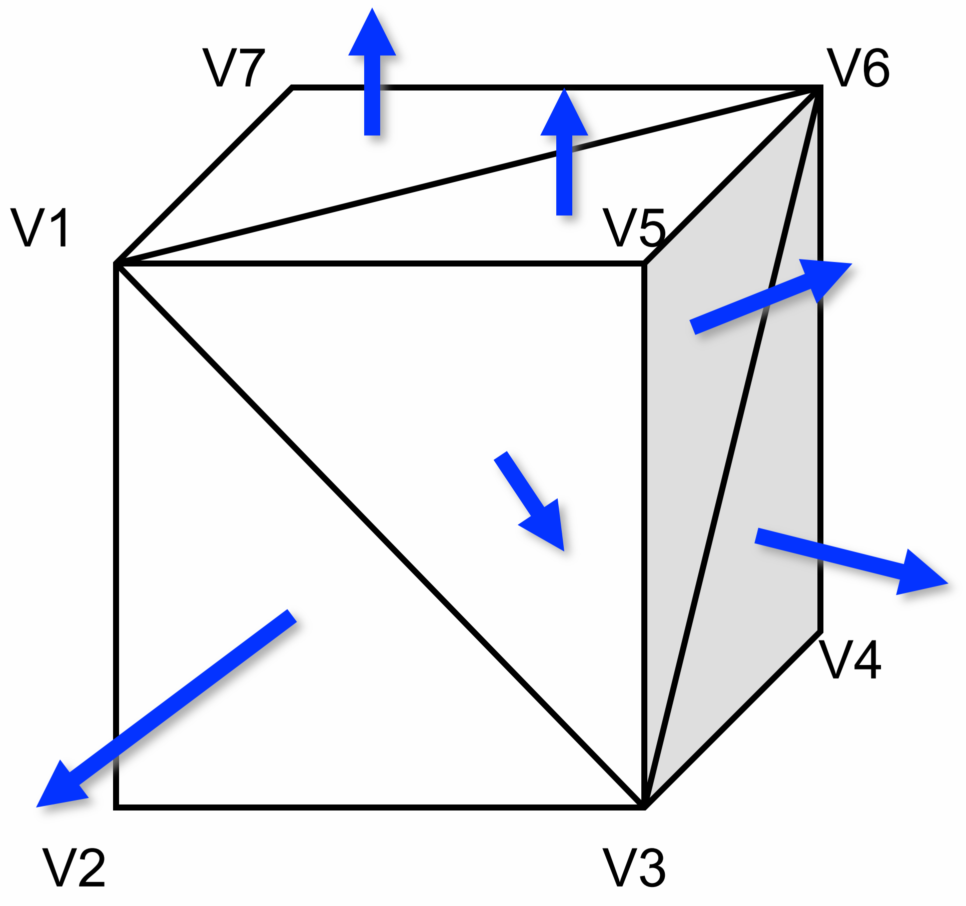
Triangle Strip



Triangle Fan

Georgia Institute of Technology

Specifying a 3D object (1)

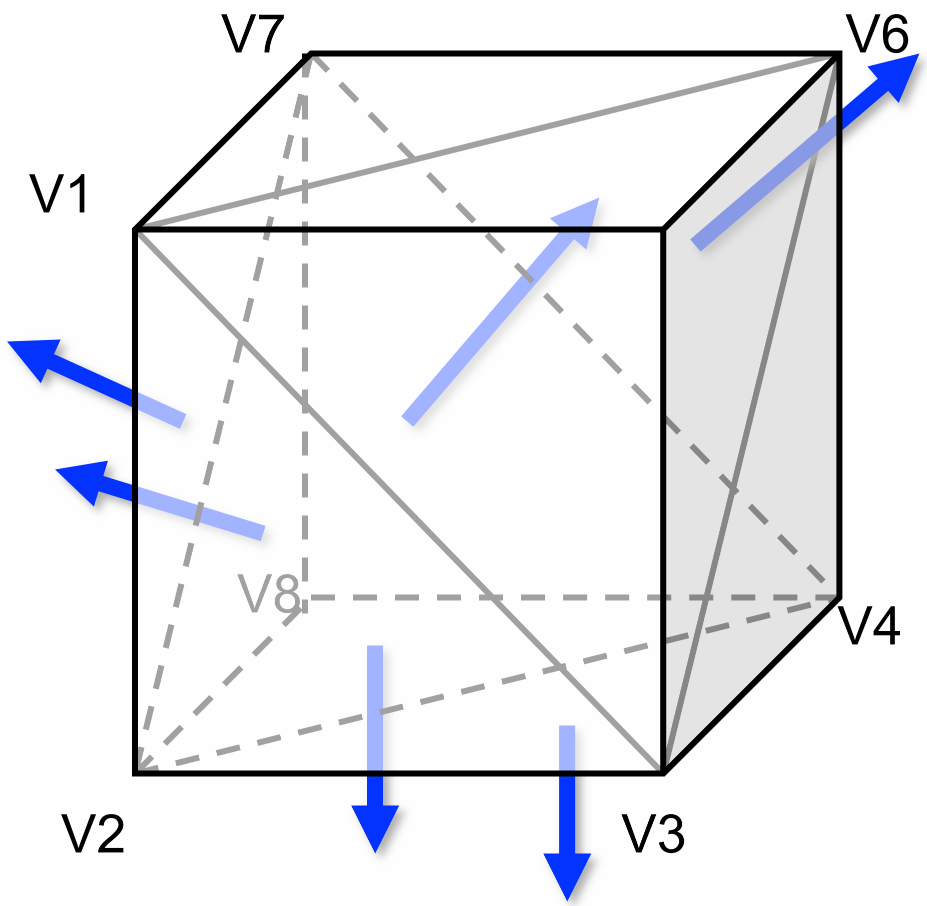


Triangle list
{v1, v3, v2},
{v1, v5, v3},
{v5, v6, v3},
{v4, v3, v6},
{v1, v7, v6},
{v1, v6, v5}

Triangle strip
{v5, v3, v1, v2},
{v5, v6, v3, v4},
{v7, v6, v1, v5}

- Vertex ordering is critical when culling mode enabled
- We will discuss normal computation later

Specifying a 3D object (2)

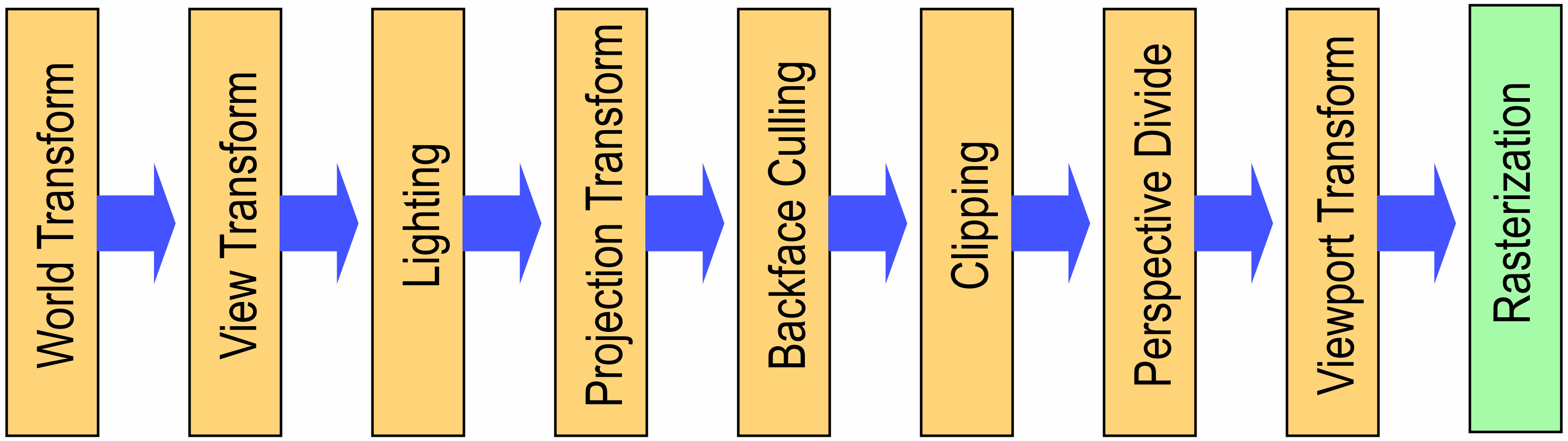


Triangle list
{v1, v2, v7},
{v2, v8, v7},
{v2, v3, v4},
{v2, v4, v8},
{v4, v7, v8},
{v4, v6, v7}

Triangle strip
{v1, v2, v7, v8},
{v3, v4, v2, v8},
{v6, v7, v4, v8}

- Vertex ordering is critical when culling mode enabled
- We will discuss normal computation later

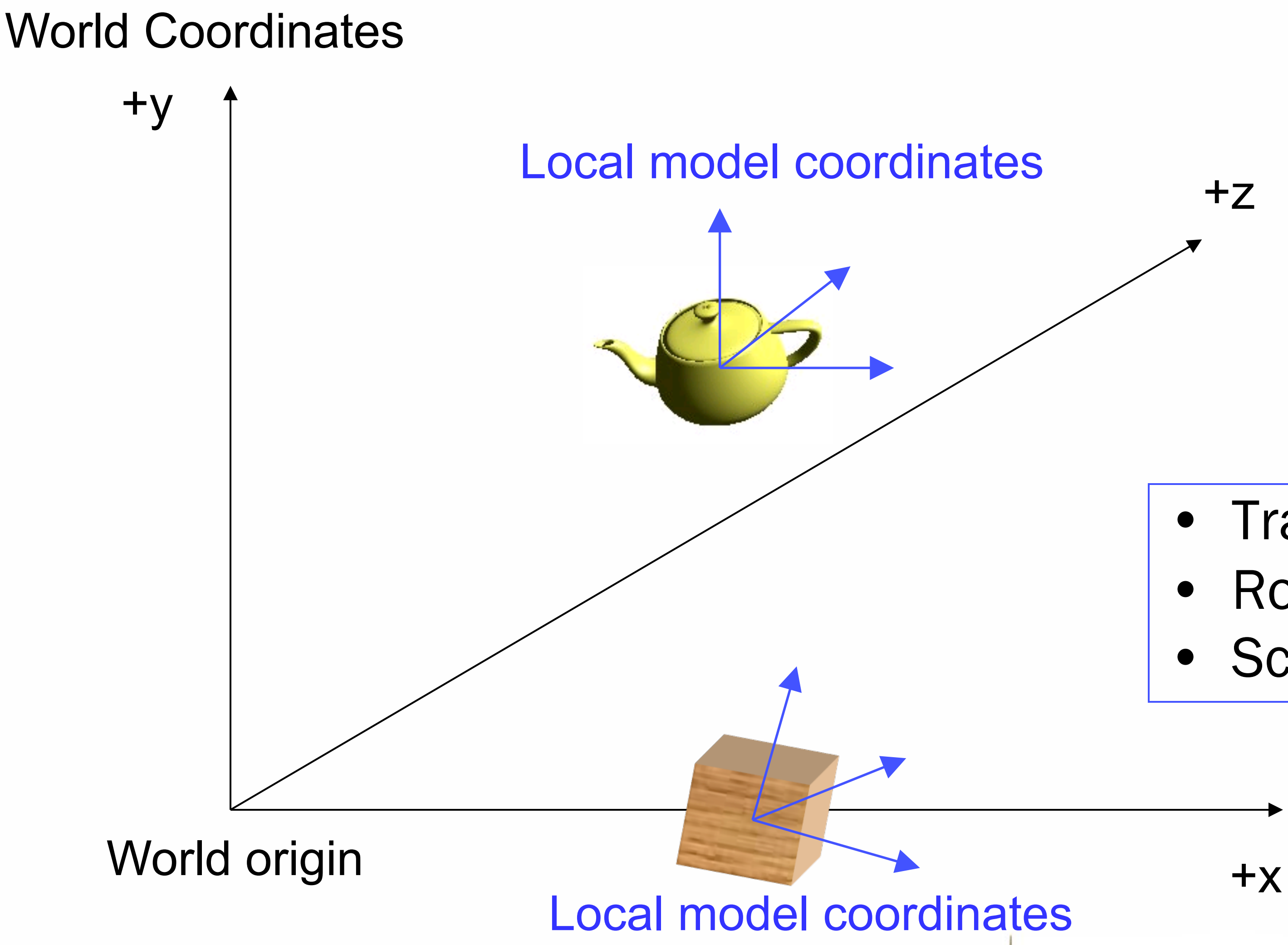
3D rendering pipeline



Transformation pipeline

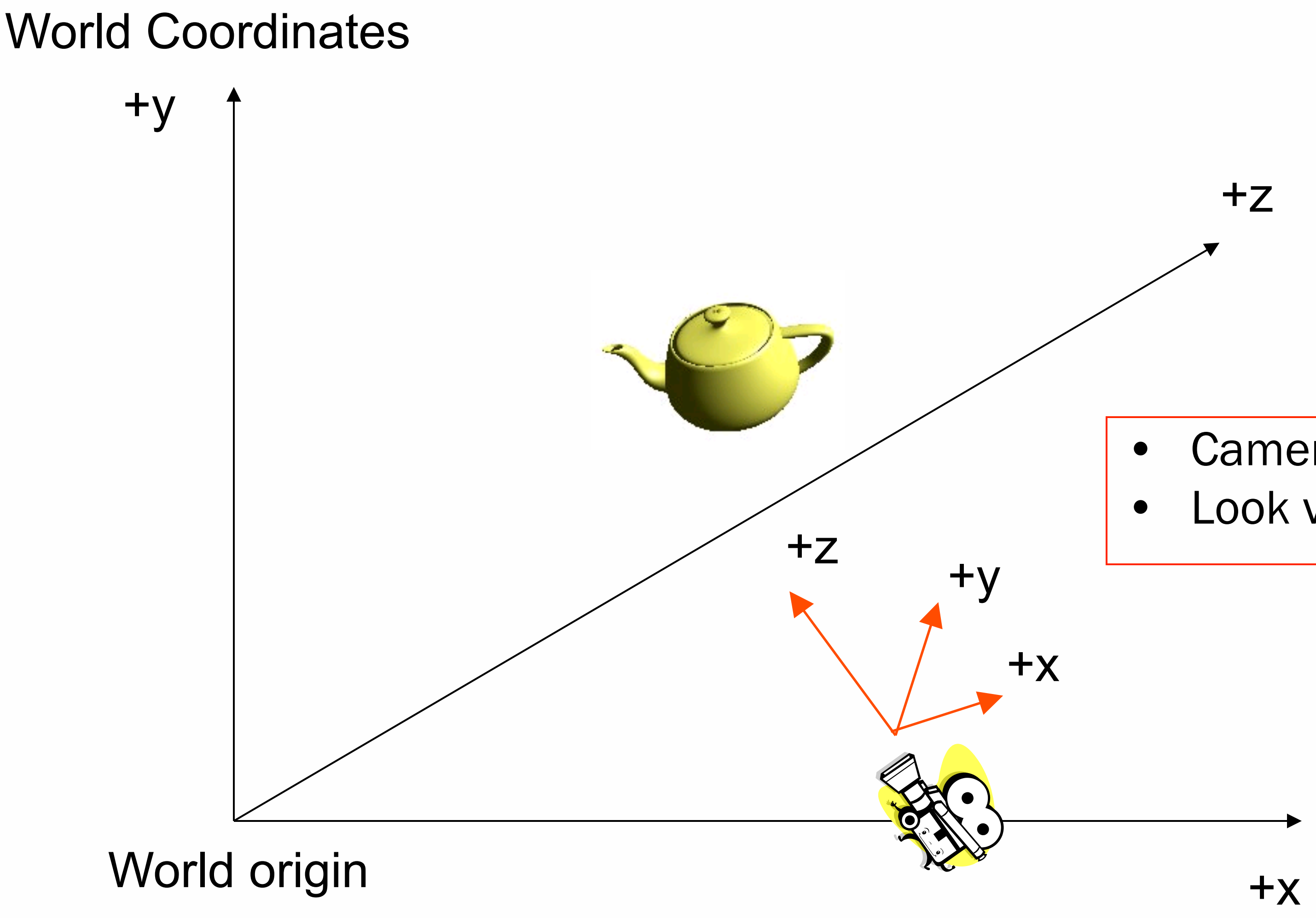
- World Transformation
  - Model coordinates → World coordinates
- View Transformation
  - World coordinates → Camera space
- Projection Transformation
  - Camera space → View plane
- These are a series of matrix multiplications

World transformation



- Translation
- Rotation
- Scaling

View transformation



- Camera position
- Look vector

Projection transformation

- Set up camera internals
- Set up
  - Field of View (FOV)
  - View frustum
  - View planes
- Will discuss in the next lecture

Georgia Institute of Technology

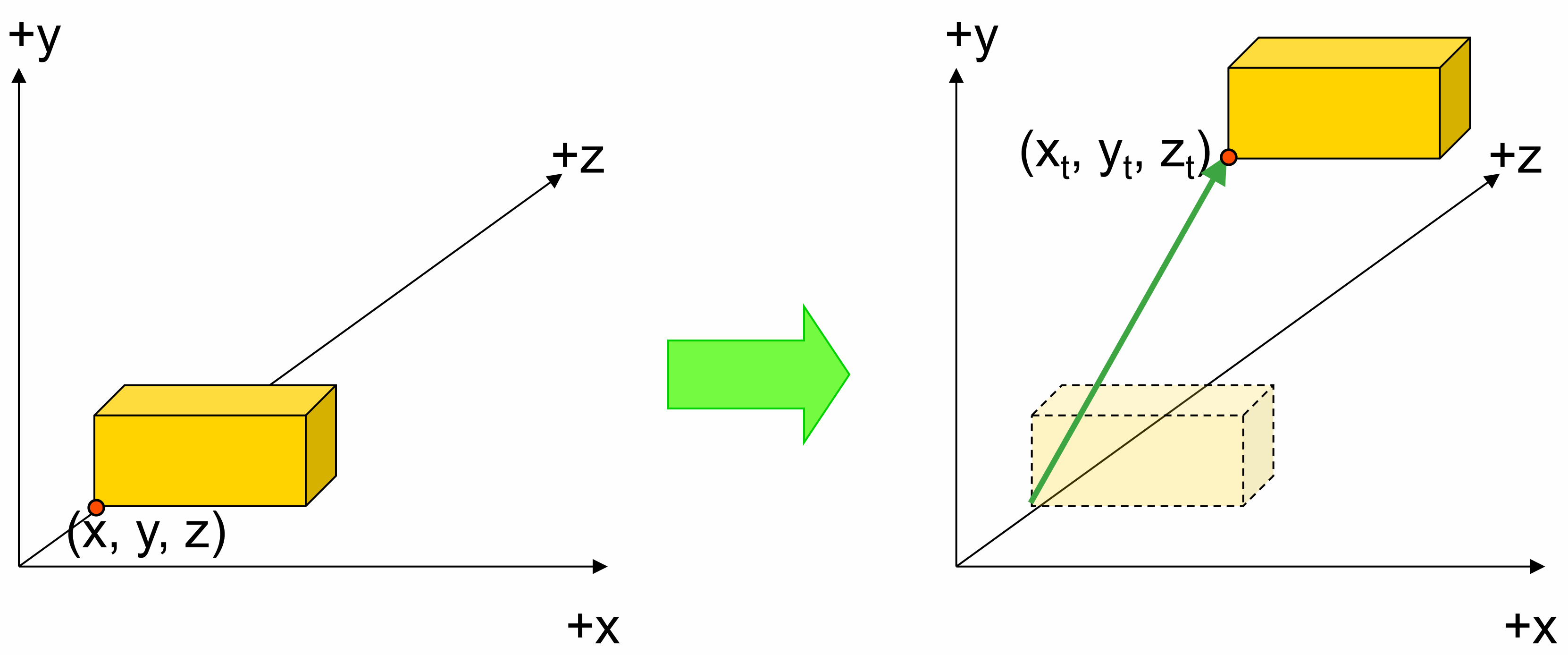
Homogeneous coordinates

- Enable all transformations to be done by "*multiplication*"
  - Primarily for translation (see next few slides)

- Add one coordinate (w) to a 3D vector

- Each vertex has [x, y, z, w]
  - w will be useful for perspective projection
  - w should be 1 in a Cartesian coordinate system

Transformation 1: translation (Offset)



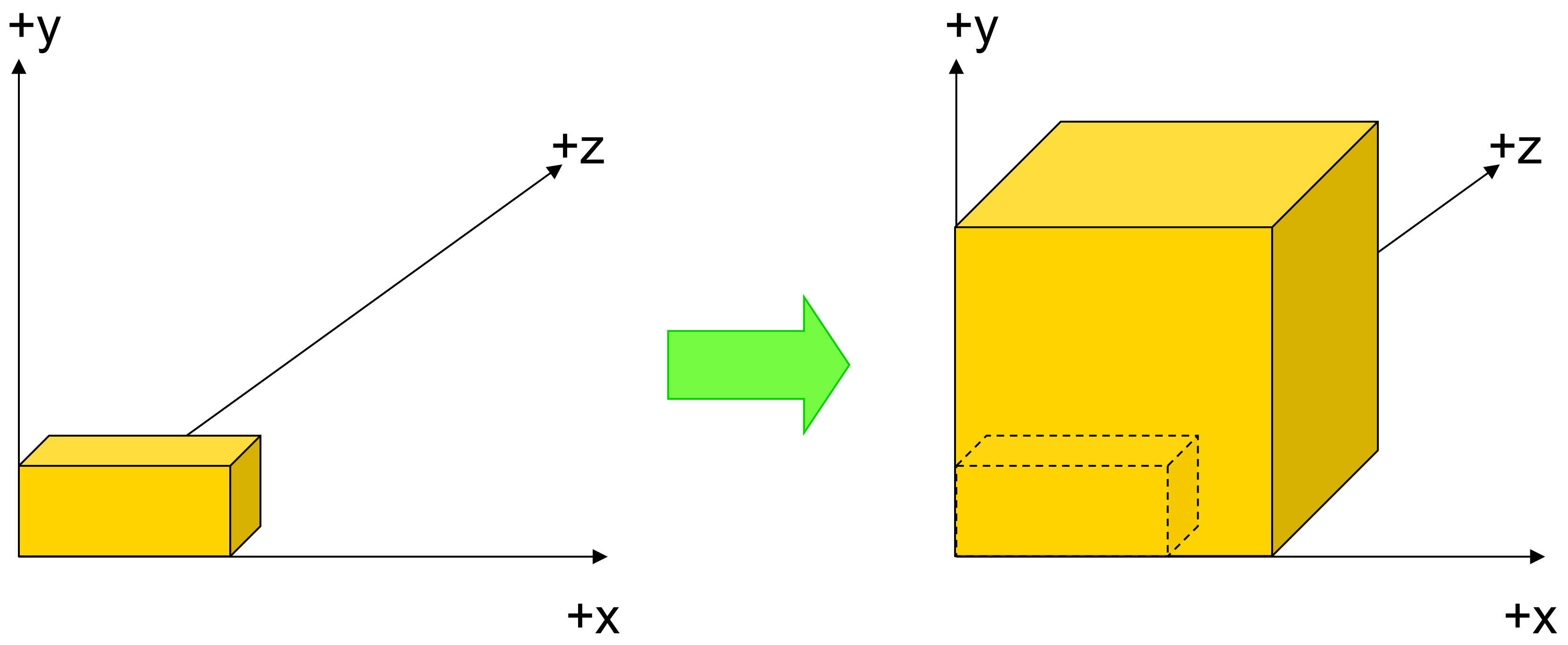
Translation matrix

$$[x_t, y_t, z_t, 1] = [x, y, z, 1] \cdot \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ T_x & T_y & T_z & 1 \end{bmatrix}$$

- Example of a row-coordinate convention
- Direct3D & XNA use row coordinates
- OpenGL uses column coordinates

Georgia Institute of Technology

Transformation 2: scaling

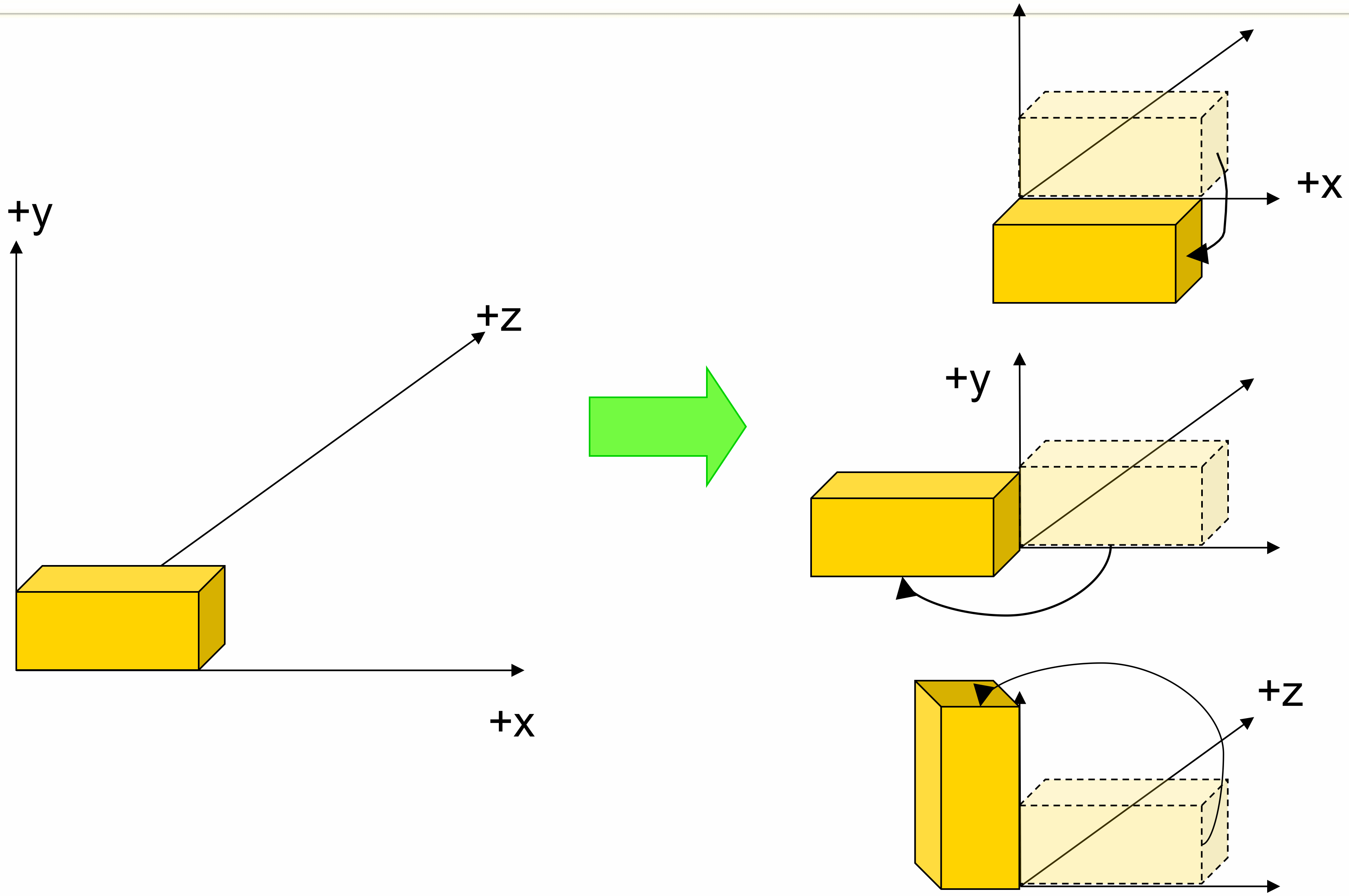


Scaling matrix

$$[x_s, y_s, z_s, 1] = [x, y, z, 1] \cdot \begin{bmatrix} S_x & 0 & 0 & 0 \\ 0 & S_y & 0 & 0 \\ 0 & 0 & S_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

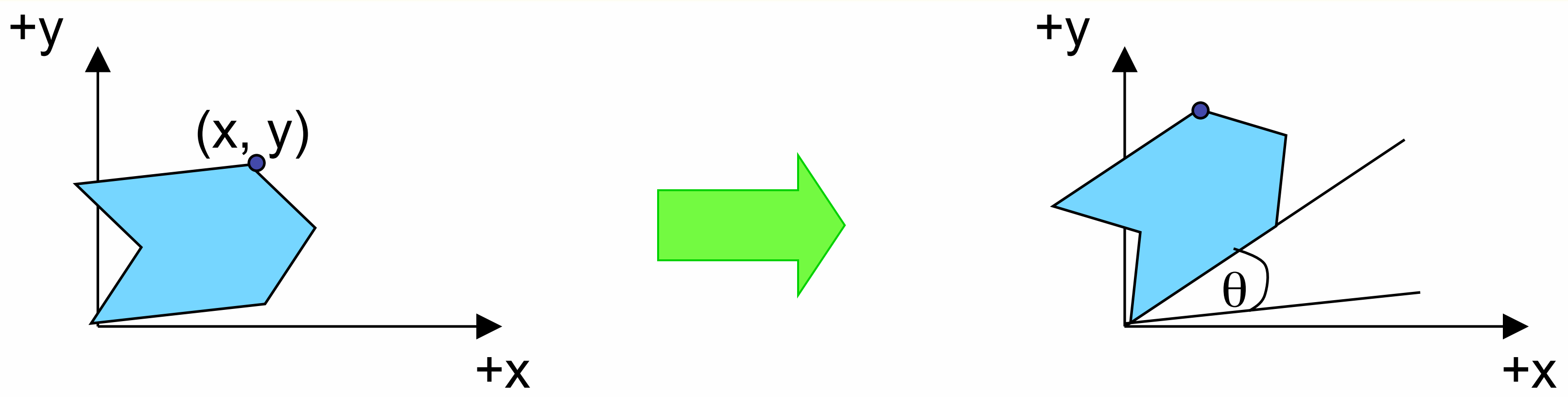
Georgia Institute of Technology

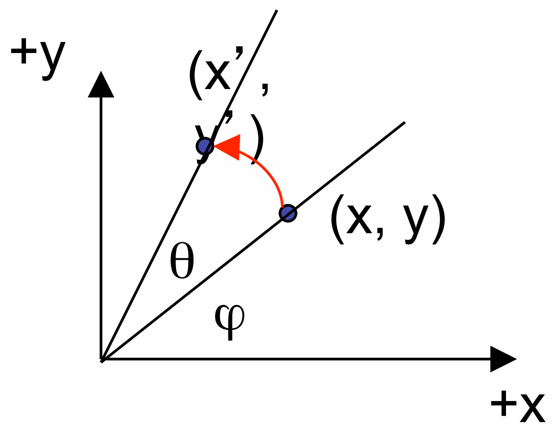
Transformation 3: rotation



Georgia Institute of Technology

2D rotation





$$[x', y', 1] = [x, y, 1] \cdot \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Rotate along which axis?

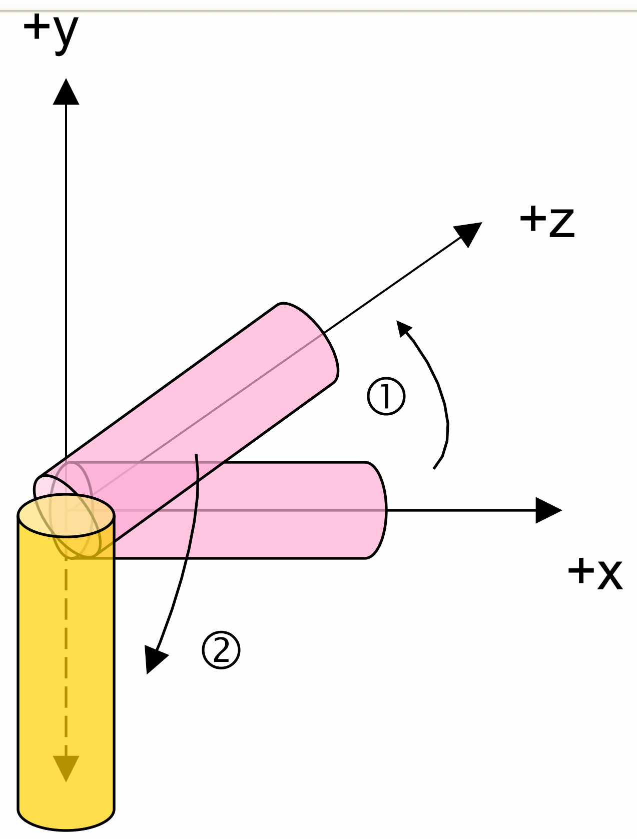
3D rotation matrix (LHS)

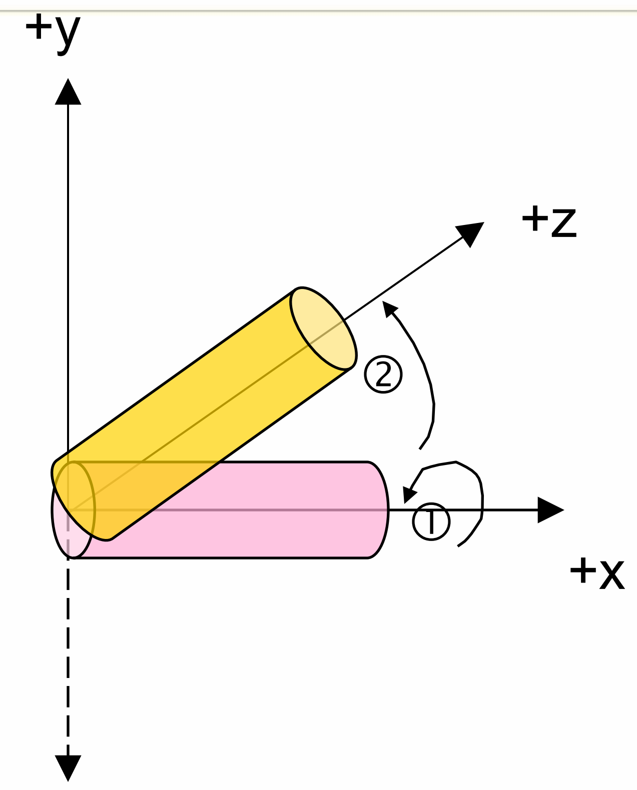
Rotation along **Z** axis
$$[x', y', z', 1] = [x, y, z, 1] \cdot \begin{bmatrix} \cos\theta & \sin\theta & 0 & 0 \\ -\sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Rotation along **Y** axis
$$[x', y', z', 1] = [x, y, z, 1] \cdot \begin{bmatrix} \cos\theta & 0 & -\sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ \sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Rotation along **X** axis
$$[x', y', z', 1] = [x, y, z, 1] \cdot \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & \sin\theta & 0 \\ 0 & -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Non-commutative property (1)

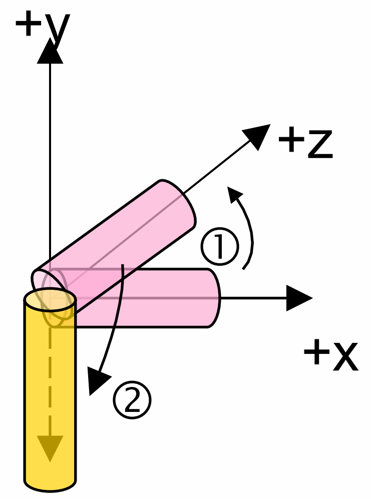




1. Counter-clockwise 90^o along y
2. Clockwise 90^o along x

1. Clockwise 90^o along x
2. Counter-clockwise 90^o along y

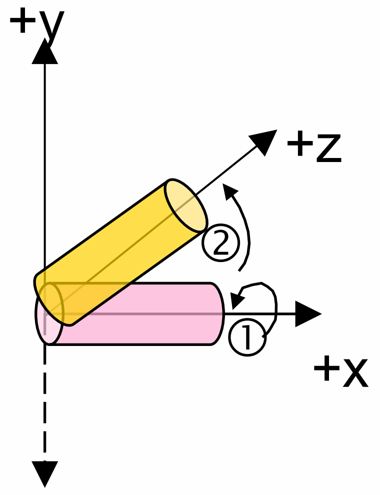
Non-commutative property (2)



① ②

$$\begin{bmatrix} \cos(-\frac{\pi}{2}) & 0 & -\sin(-\frac{\pi}{2}) & 0 \\ 0 & 1 & 0 & 0 \\ \sin(-\frac{\pi}{2}) & 0 & \cos(-\frac{\pi}{2}) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\frac{\pi}{2}) & \sin(\frac{\pi}{2}) & 0 \\ 0 & -\sin(\frac{\pi}{2}) & \cos(\frac{\pi}{2}) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

(x'', y'', z'') = (-z, -x, y)

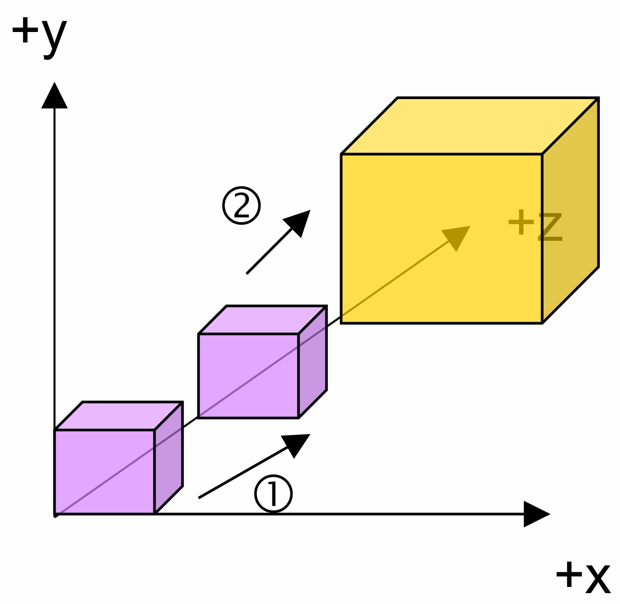


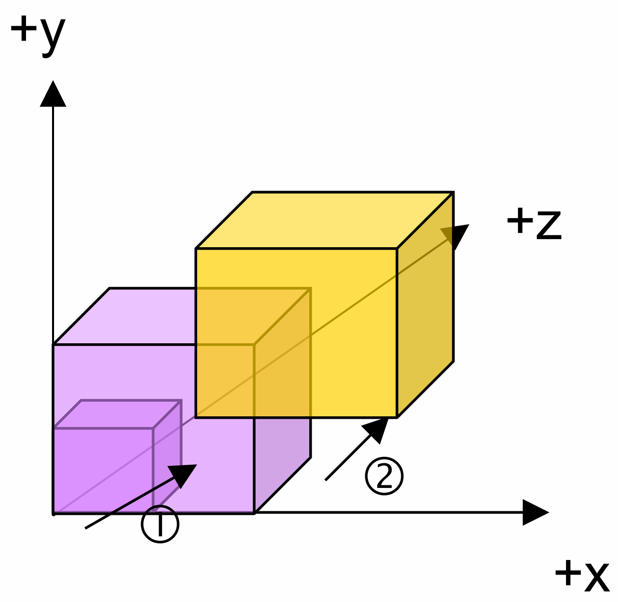
① ②

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\frac{\pi}{2}) & \sin(\frac{\pi}{2}) & 0 \\ 0 & -\sin(\frac{\pi}{2}) & \cos(\frac{\pi}{2}) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \cos(-\frac{\pi}{2}) & 0 & -\sin(-\frac{\pi}{2}) & 0 \\ 0 & 1 & 0 & 0 \\ \sin(-\frac{\pi}{2}) & 0 & \cos(-\frac{\pi}{2}) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

(x'', y'', z'') = (-y, -z, x)

Non-commutative property (3)



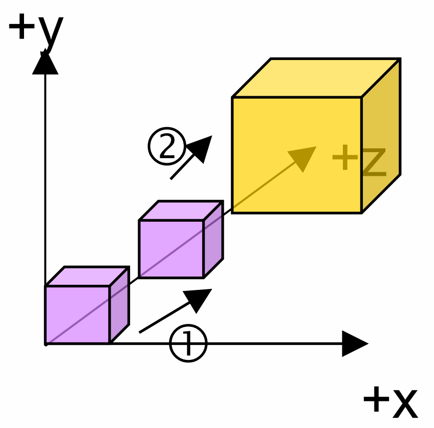


1. Translation by (x, y, z)
2. Scale by 2 times

1. Scale by 2 times
2. Translation by (x, y, z)

Georgia Institute of Technology

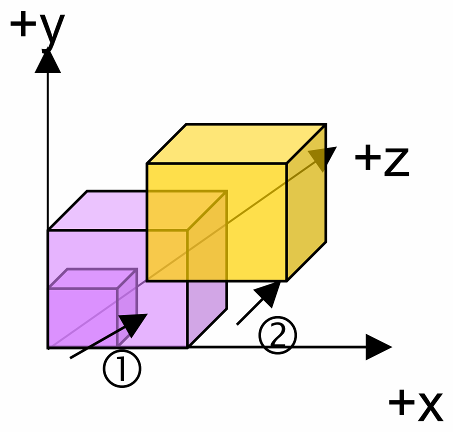
Non-commutative property (4)



$$\begin{array}{cccc} ① & & ② & \\ \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ Tx & Ty & Tz & 1 \end{bmatrix} & \cdot & \begin{bmatrix} Sx & 0 & 0 & 0 \\ 0 & Sy & 0 & 0 \\ 0 & 0 & Sz & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} & = \begin{bmatrix} Sx & 0 & 0 & 0 \\ 0 & Sy & 0 & 0 \\ 0 & 0 & Sz & 0 \\ SxTx & SyTy & SzTz & 1 \end{bmatrix} \end{array}$$

(x’ ’ , y’ ’ , z’ ’) = (x*Sx+Sx*Tx, y*Sy+Sy*Ty, z*Sz+Sz*Tz)

Offsets were scaled as well



$$\begin{array}{cccc} ① & & ② & \\ \begin{bmatrix} Sx & 0 & 0 & 0 \\ 0 & Sy & 0 & 0 \\ 0 & 0 & Sz & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} & \cdot & \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ Tx & Ty & Tz & 1 \end{bmatrix} & = \begin{bmatrix} Sx & 0 & 0 & 0 \\ 0 & Sy & 0 & 0 \\ 0 & 0 & Sz & 0 \\ Tx & Ty & Tz & 1 \end{bmatrix} \end{array}$$

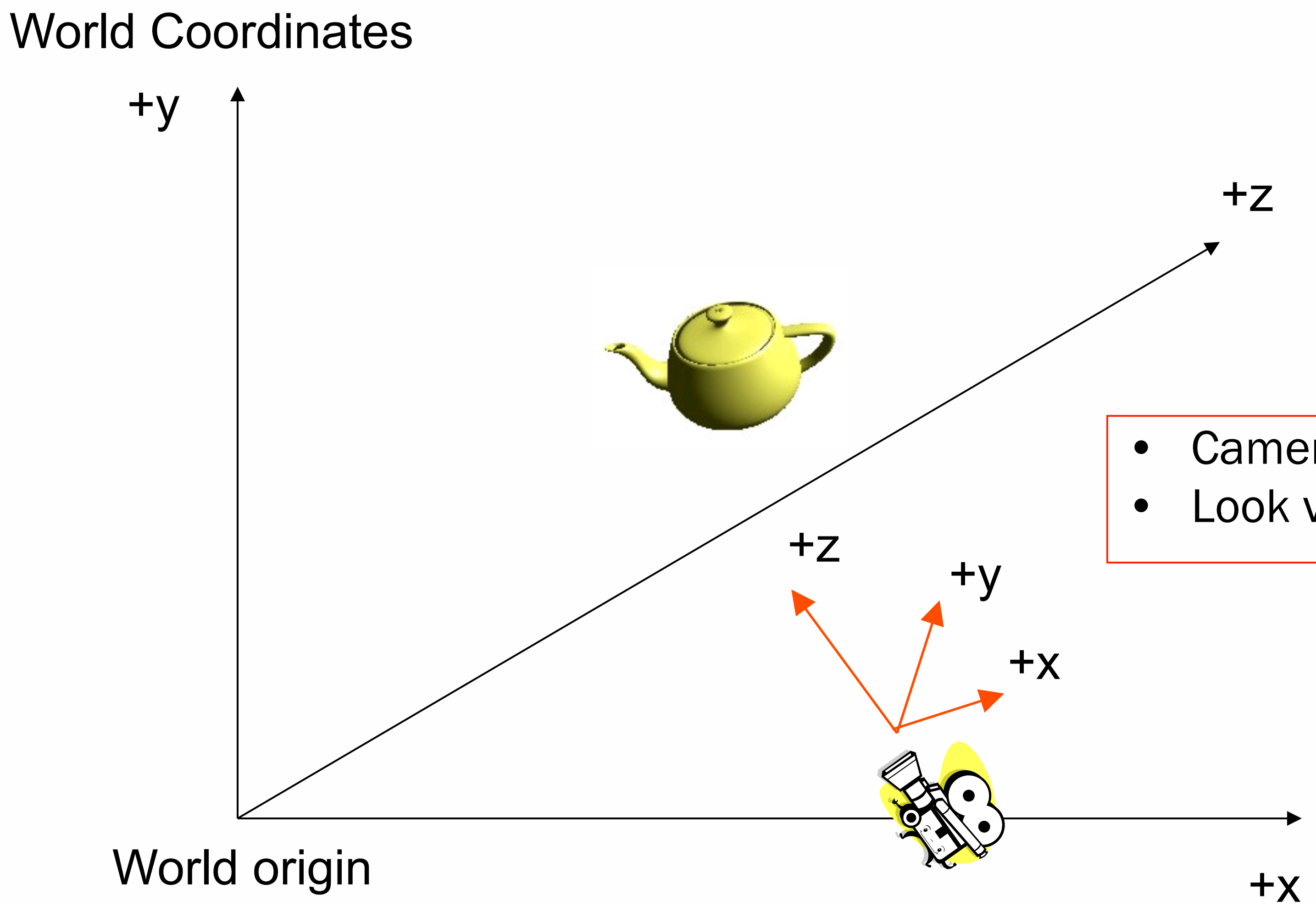
(x’ ’ , y’ ’ , z’ ’) = (x*Sx+Tx, y*Sy+Ty, z*Sz+Tz)

Georgia Institute of Technology

Non-commutative property (5)

- Ordering matters !
- Be careful when performing matrix multiplication

View transformation revisited



- Camera position
- Look vector

Specifying the view transformation

- Most commonly parameterized by:
  - Position of camera
  - Position of point to look at
  - Vector indicating “up” direction of camera
- In Direct3D: `D3DXMatrixLookAtLH`
  - D3D uses a LHS, but also have `D3DXMatrixLookAtRH`
- In XNA: `Matrix.CreateLookAt` (RHS)
- In OpenGL: `gluLookAt` (RHS)
- Can also build a rotation+translation matrix as if the camera was an object in scene, then take the inverse of that matrix

msdn.microsoft.com/en-us/library/bb205342(VS.85).aspx
msdn.microsoft.com/en-us/library/bb205343(VS.85).aspx

Georgia Institute of Technology