CS580:
Classic Rendering Pipeline

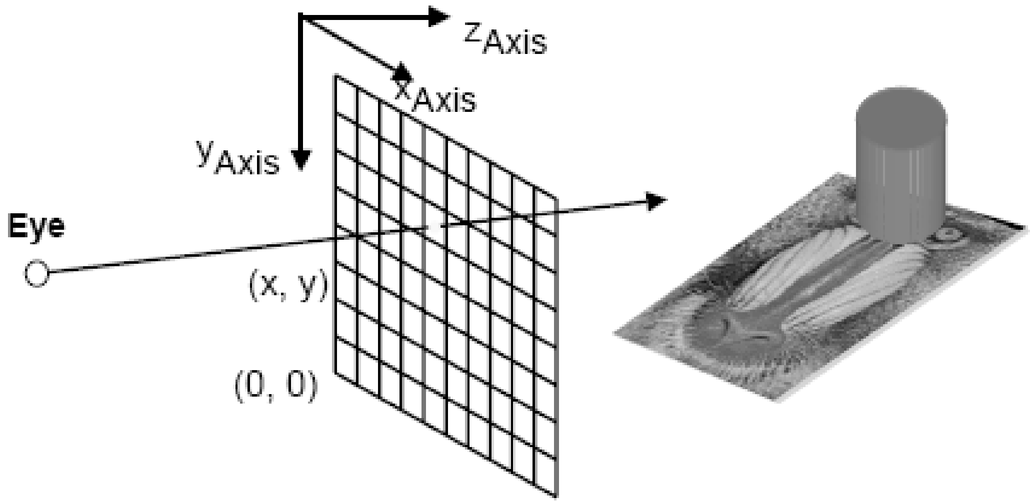
Sung-Eui Yoon
(윤성의)

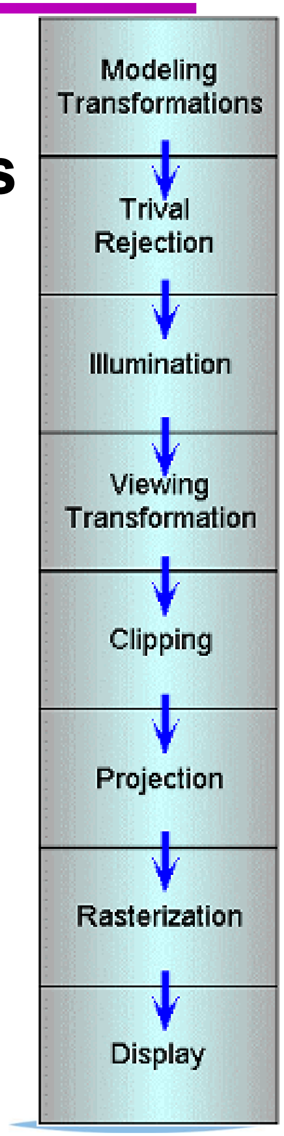
Course URL:
http://sglab.kaist.ac.kr/~sungeui/GCG/

KAIST

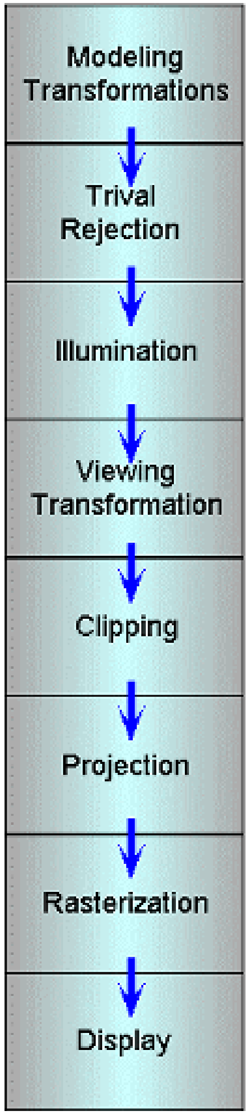
Course Objectives

- **Understand classic rendering pipeline**
  - **Just high-level concepts, not all the details**
  - **Brief introduction of common under. CG**
- **Know its pros and cons**





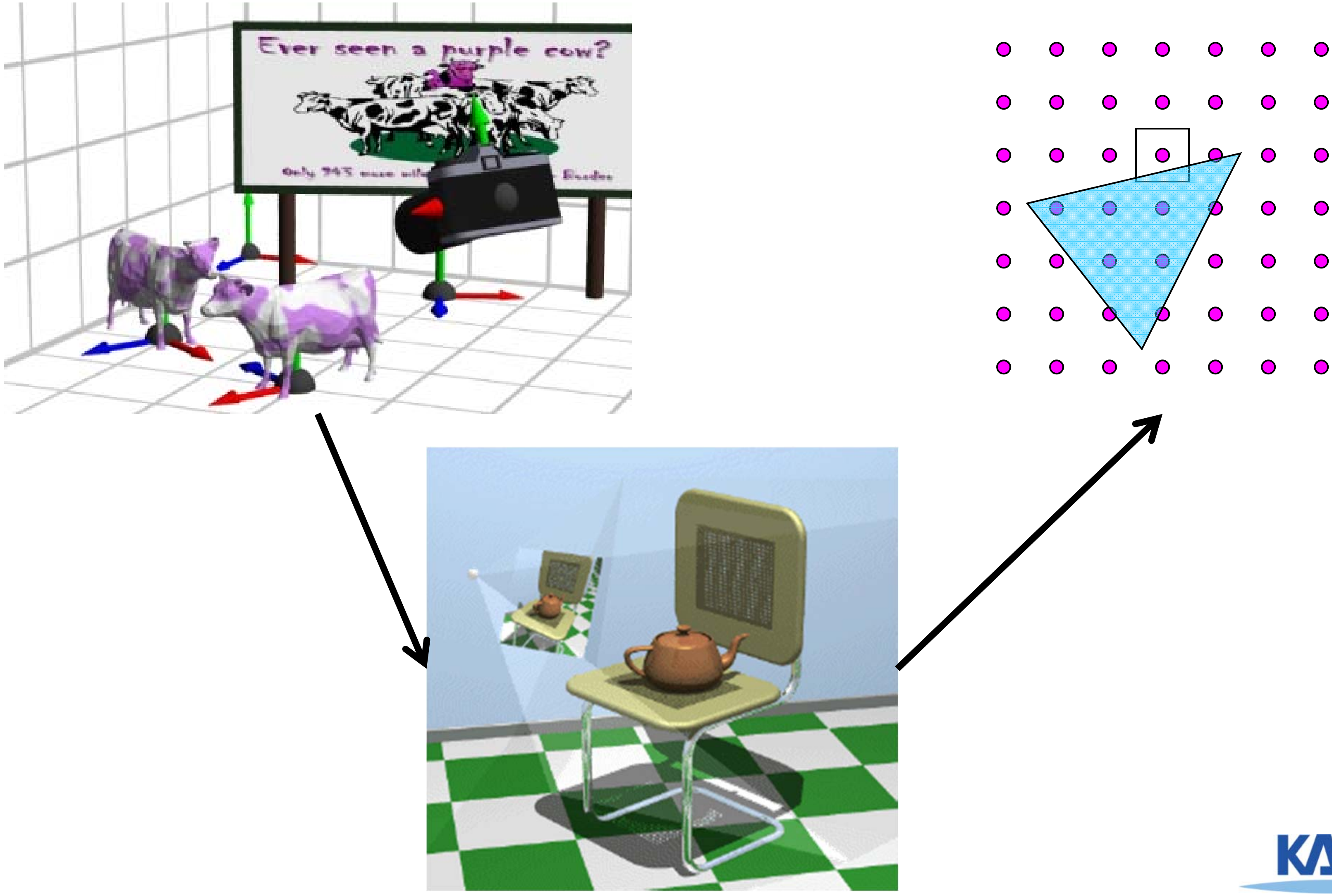
The Classic Rendering Pipeline



- **Adopted in OpenGL and DirectX**
  - **Most of games are based on this pipeline**

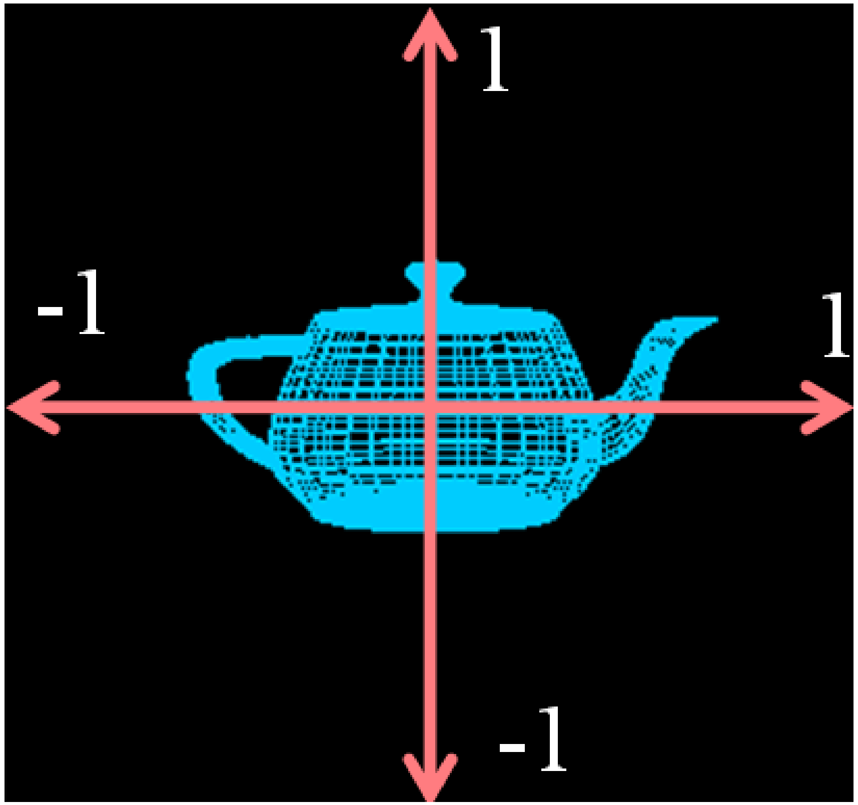
- **Object primitives defined by vertices fed in at the top**
- **Pixels come out in the display at the bottom**

The Classic Rendering Pipeline

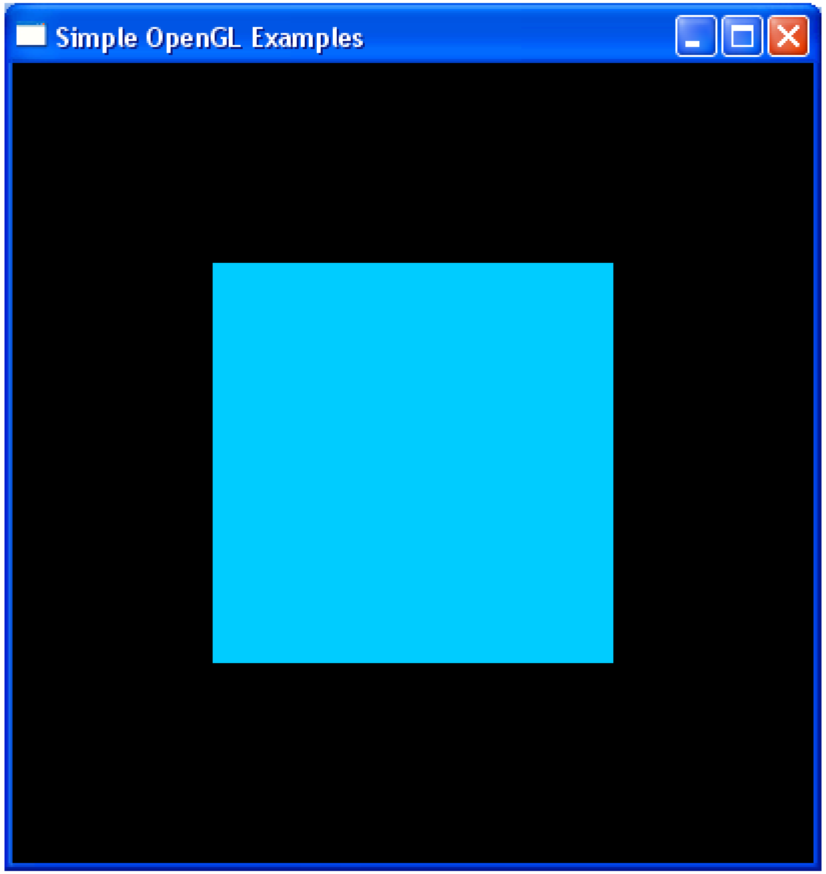


Your New World in OpenGL

- A 2D square ranging from (-1, -1) to (1, 1)
- You can draw in the box with just a few lines of code



Code Example



OpenGL Code:

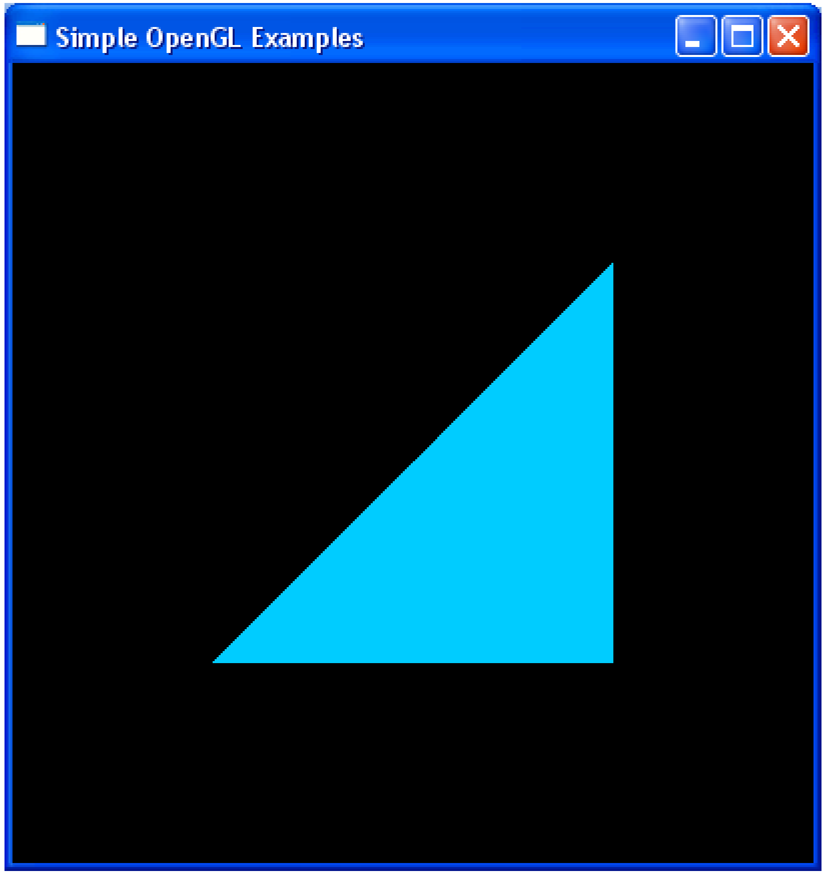
```
glColor3d(0.0, 0.8, 1.0);

glBegin(GL_POLYGON);
        glVertex2d(-0.5, -0.5);
        glVertex2d( 0.5, -0.5);
        glVertex2d( 0.5,  0.5);
        glVertex2d(-0.5,  0.5);
glEnd();
```

OpenGL Command Syntax

- **glColor3d(0.0, 0.8, 1.0);**

Suffix	Data Type	Corresponding C-Type	OpenGL Type
b	8-bit int.	singed char	GLbyte
s	16-bit int.	short	GLshort
i	32-bit int.	int	GLint
f	32-bit float	float	GLfloat
d	64-bit double	double	GLdouble
ub	8-bit unsinged int.	unsigned char	GLubyte
us	16-bit unsigned int.	unsigned short	GLushort
ui	32-bit unsigned int.	unsigned int	GLuint

Another Code Example



OpenGL Code:

```
glColor3d(0.0, 0.8, 1.0);

glBegin(GL_POLYGON);
      glVertex2d(-0.5, -0.5);
      glVertex2d( 0.5, -0.5);
      glVertex2d( 0.5,  0.5);
glEnd()
```

Drawing Primitives in OpenGL

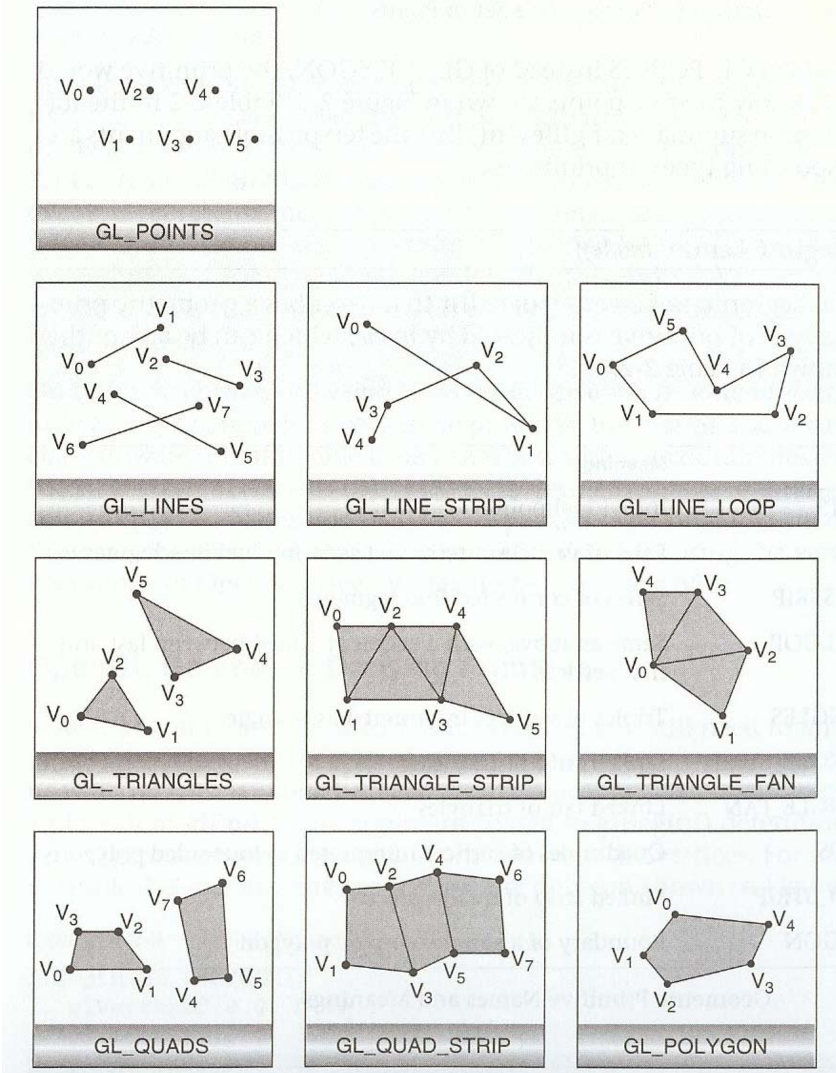
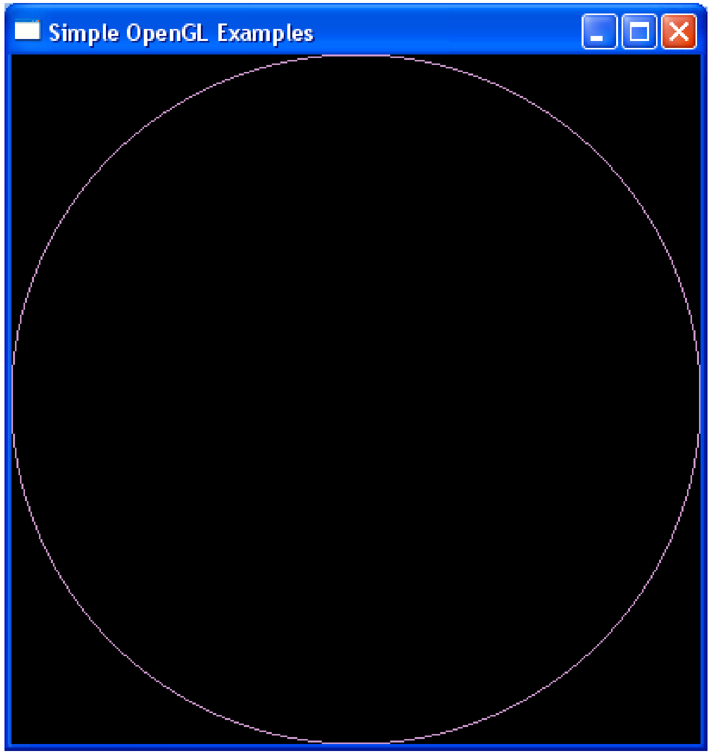


Figure 2-7 Geometric Primitive Types

The red book

Yet Another Code Example

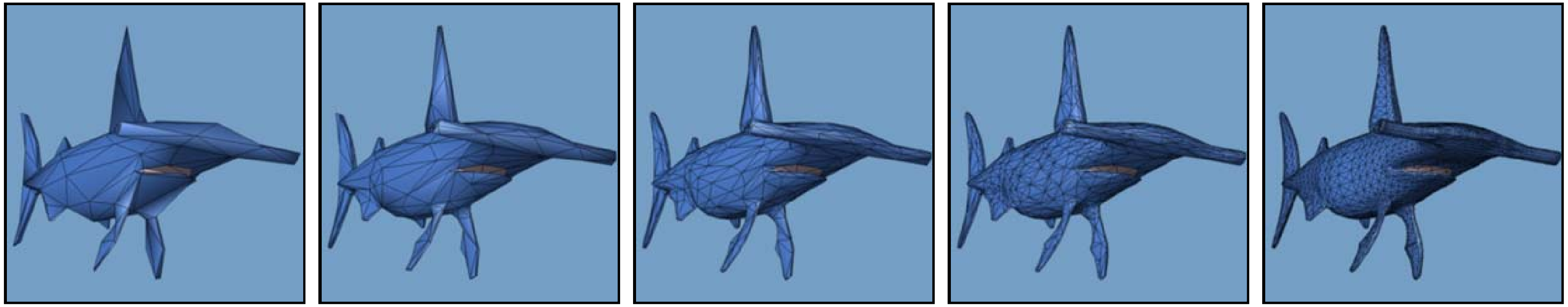


OpenGL Code:

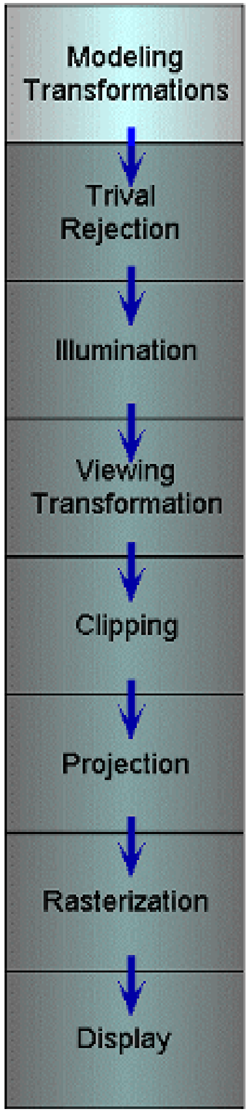
```
glColor3d(0.8, 0.6, 0.8);


glBegin(GL_LINE_LOOP);
for (i = 0; i < 360;i = i + 2)
{
   x = cos(i*pi/180);
   y = sin(i*pi/180);
   glVertex2d(x, y);
}
glEnd();
```

Triangle Representation, Mesh

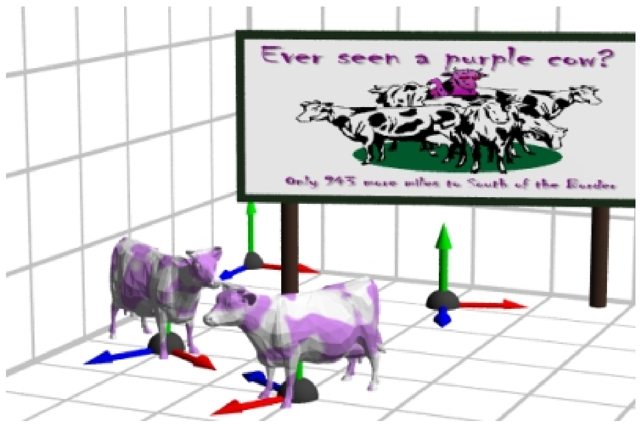
- Triangles can approximate any 2-dimensional shape (or 3D surface)
  - Polygons are a locally linear (planar) approximation
- Improve the quality of fit by increasing the number edges or faces



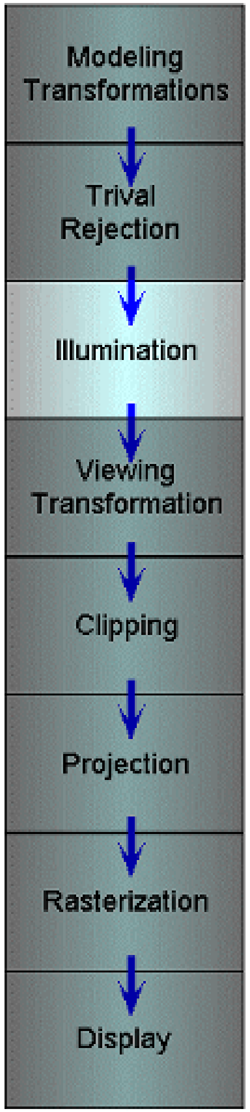
Modeling Transforms



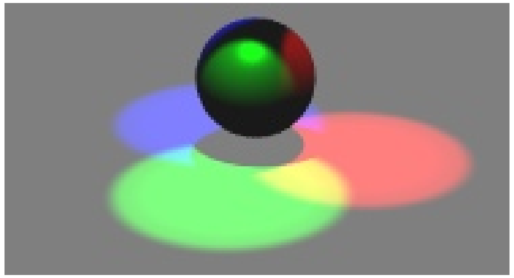
- **Start with 3D models defined in modeling spaces with their own modeling frames**
- **Modeling transformations orient models within a common coordinate frame called world space**
  - **All objects, light sources, and the camera live in world space**

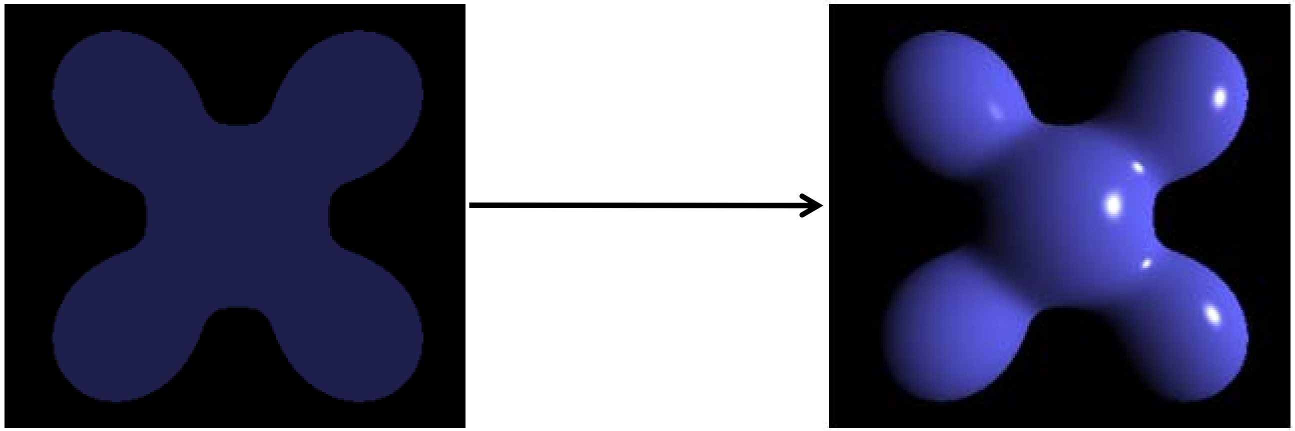


Illumination

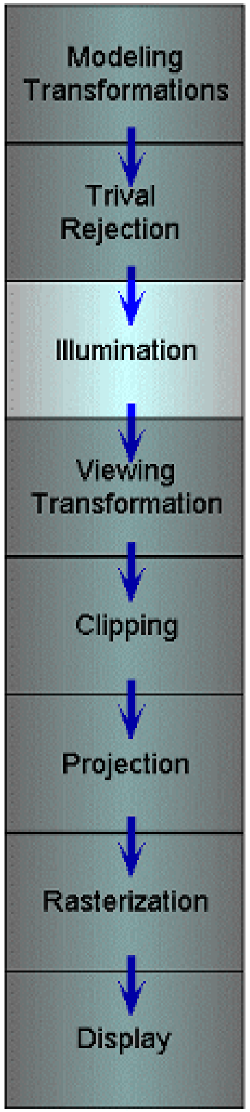


- **Illuminate potentially visible objects**
- **Final rendered color is determined by object's orientation, its material properties, and the light sources in the scene**

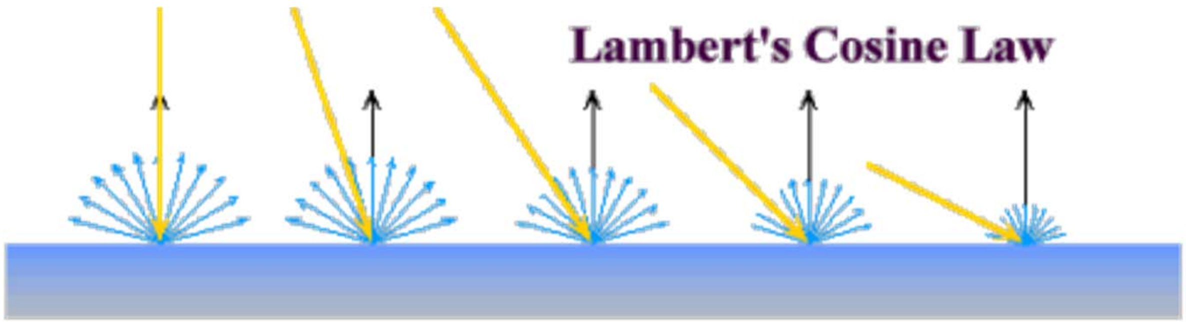




Illumination

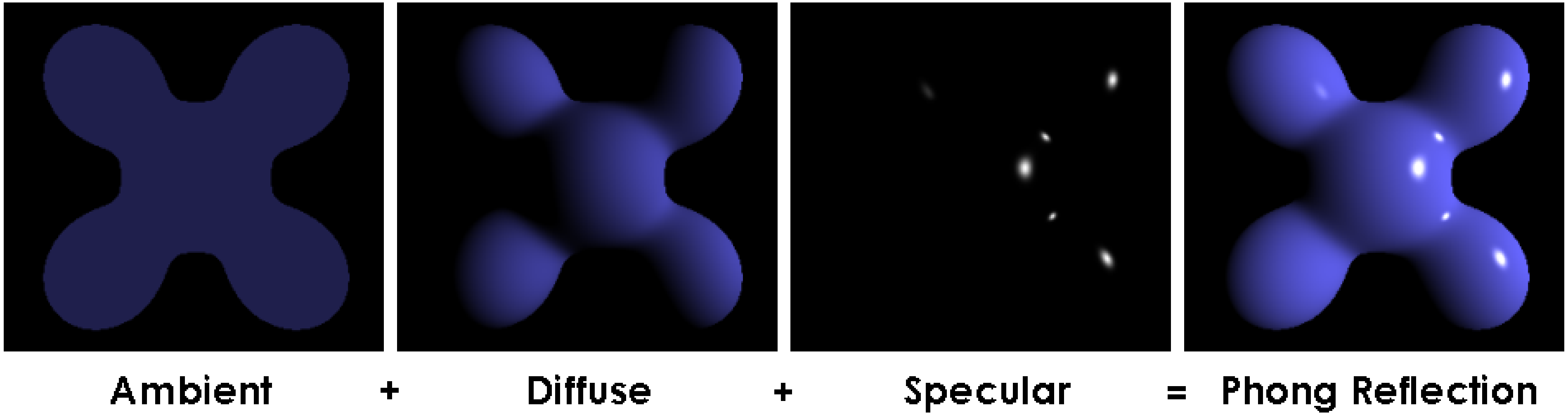


- Illuminate potentially visible objects
- Final rendered color is determined by object's orientation, its material properties, and the light sources in the scene
- **Lighting**
  - Local shading model considering direct illumination



OpenGL's Illumination Model

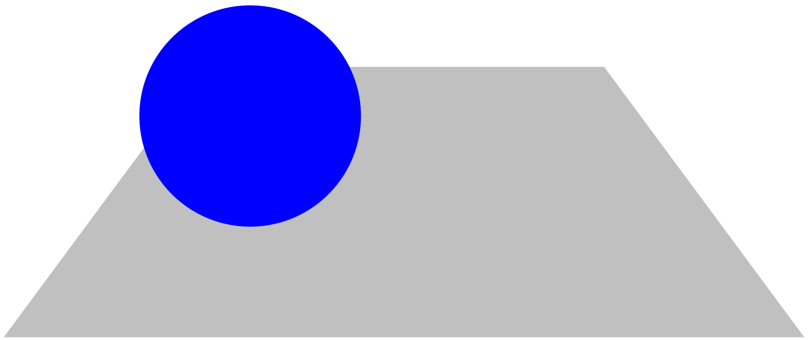
$$I_r = \sum_{j=1}^{numLights} (k_a^j I_a^j + k_d^j I_d^j \max((\hat{N} \bullet \hat{L}_j),0) + k_s^j I_s^j \max((\hat{V} \bullet \hat{R})^{n_s},0))$$



From Wikipedia

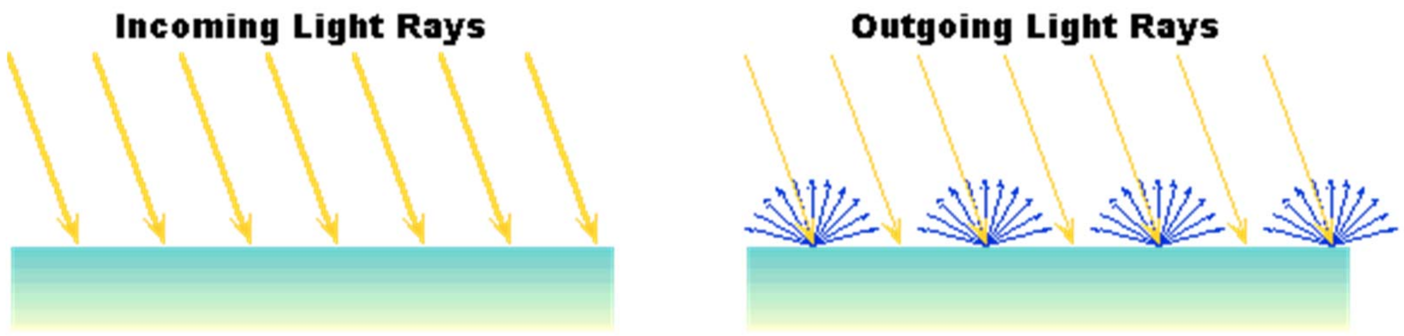
Ambient Light Source

- A simple <u>hack</u> for indirect illumination
  - Incoming ambient illumination ($I_{i,a}$) is constant for all surfaces in the scene
  - Reflected ambient illumination ($I_{r,a}$) depends only on the surface's ambient reflection coefficient (k_a) and not its position or orientation $I_{r,a} = k_a I_{i,a}$
  - These quantities typically specified as (R, G, B) triples



Ideal Diffuse Reflection

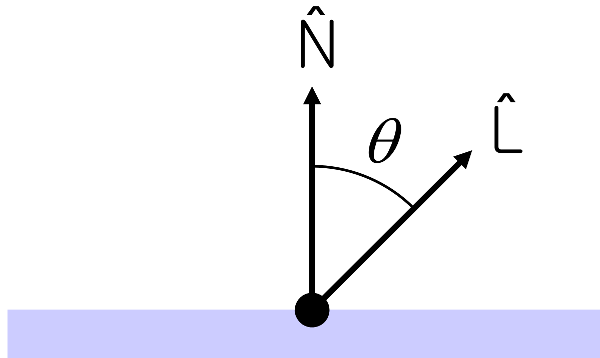
- Ideal diffuse reflectors (e.g., chalk)
  - Reflect uniformly over the hemisphere
  - Reflection is view-independent
  - Very rough at the microscopic level
- Follow Lambert's cosine law

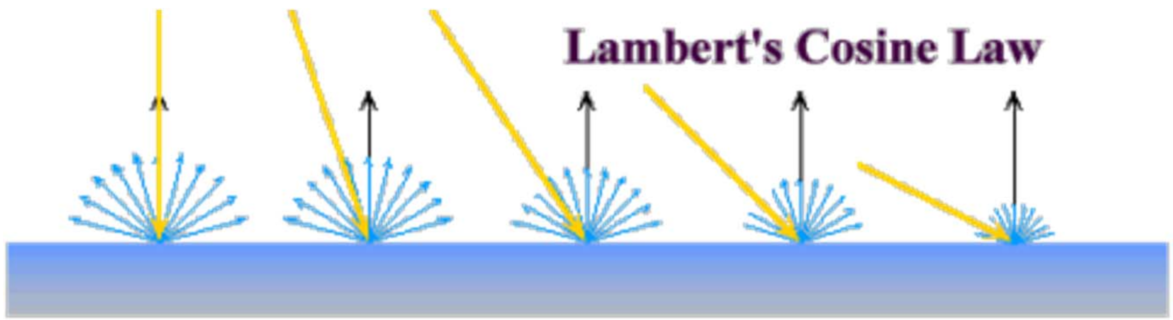


Lambert's Cosine Law

- **The reflected energy from a small surface area from illumination arriving from direction $\hat{L}$ is proportional to the cosine of the angle between $\hat{L}$ and the surface normal**

$$I_r \approx I_i \cos\theta$$
$$\approx I_i(\hat{N} \bullet \hat{L})$$





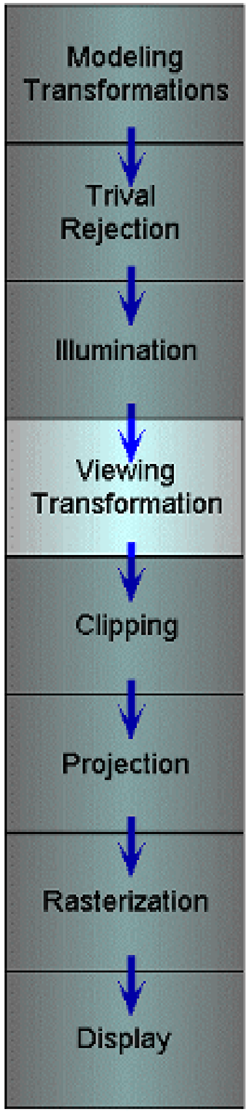
Specular Reflection

- Specular reflectors have a bright, view dependent highlight
  - E.g., polished metal, glossy car finish, a mirror
  - At the microscopic level a specular reflecting surface is very smooth
  - Specular reflection obeys Snell's law

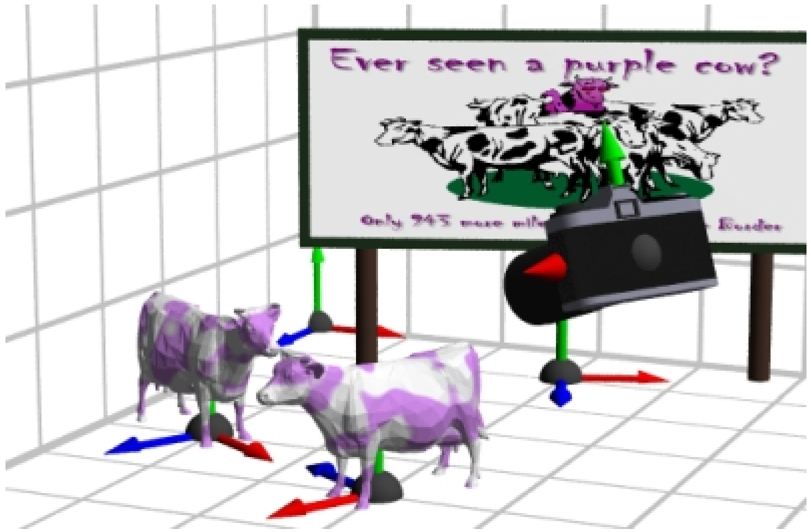
Image source: astochimp.com and wiki

KAIST

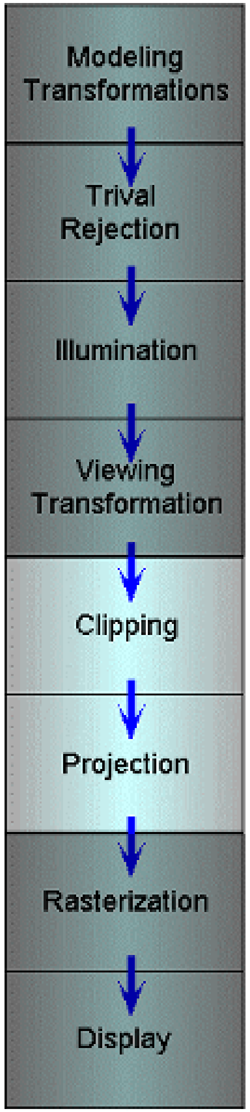
Viewing Transformations



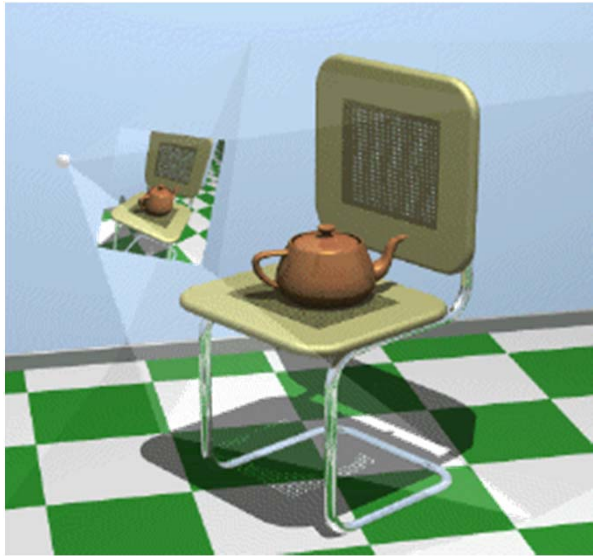
- **Maps points from world space to eye space**
  - **Viewing position is transformed to the origin**
  - **Viewing direction is oriented along some axis**



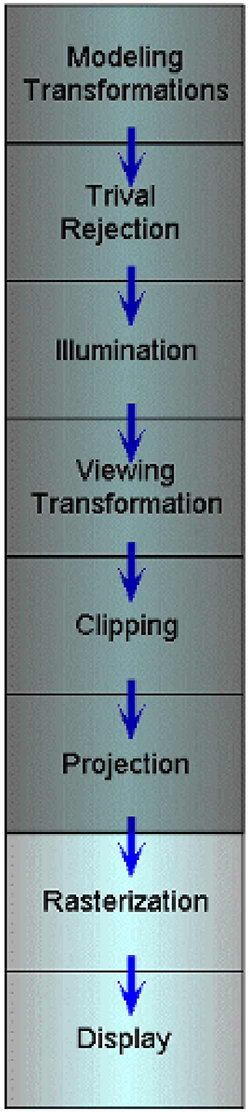
Clipping and Projection



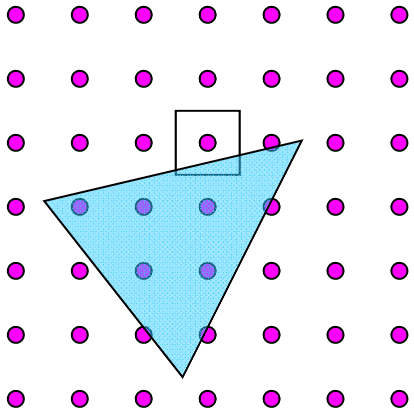
- We specify a volume called a *viewing frustum*
- Map the view frustum to the unit cube
- Clip objects against the view volume, thereby eliminating geometry not visible in the image
- Project objects into two-dimensions



Rasterization and Display



- Transform to screen space
- Rasterization converts objects pixels

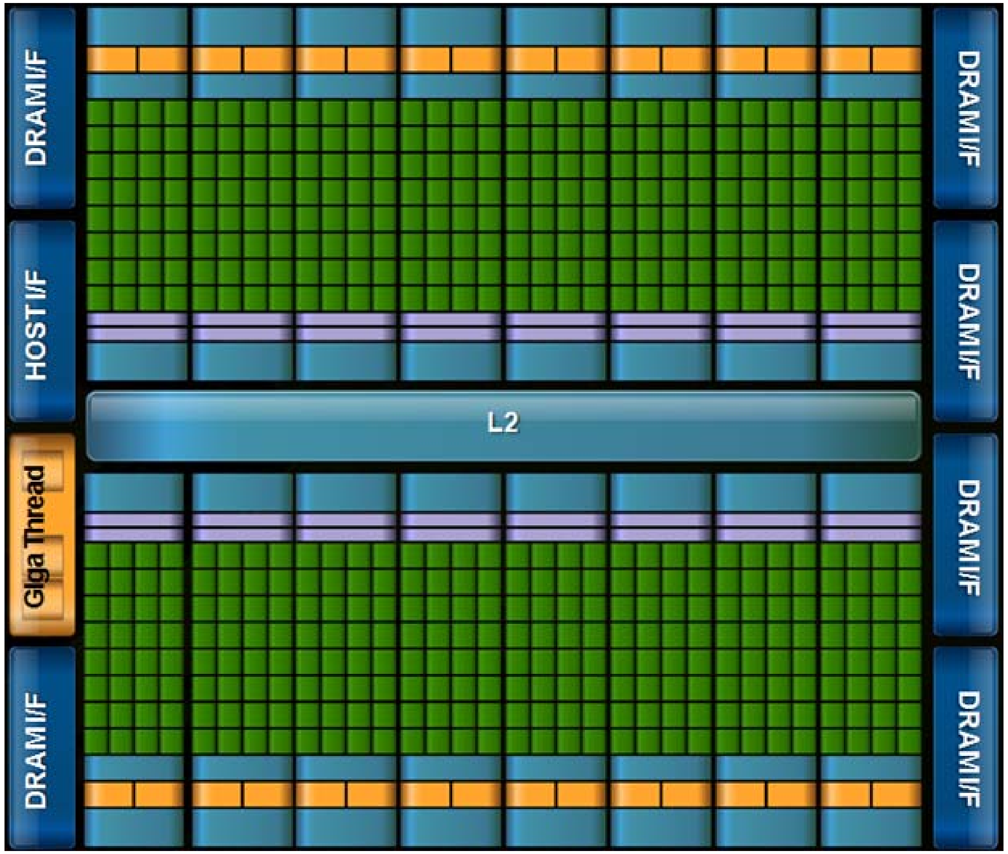


Why we are using rasterization?

- **Efficiency**
- **Reasonably quality**

Fermi GPU Architecture

16 SM (streaming processors)



512 CUDA cores

Memory interfaces

Where Rasterization Is

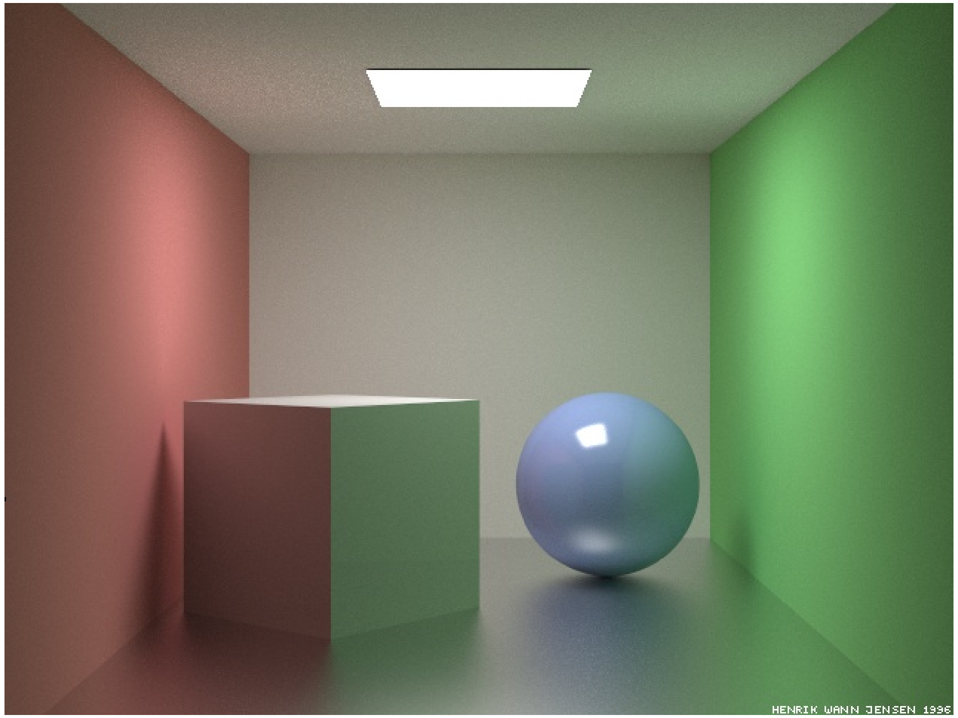
From Battlefield: Bad Company, EA Digital Illusions CE AB

From Eric Haines

But what about other visual cues?

- Lighting
  - Shadows
  - Shading: glossy, transparency
- Color bleeding, etc.

- Generality

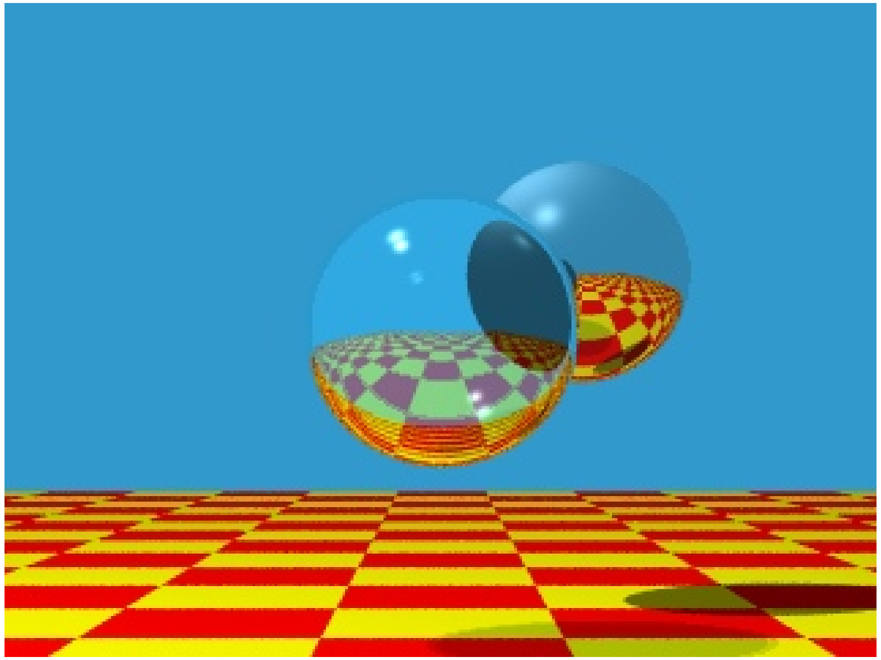


from Henrik's homepage

KAIST

Recursive Ray Casting

- **Gained popularity in when Turner Whitted (1980) recognized that *recursive* ray casting could be used for global illumination effects**



Any Questions?

- **Come up with one question on what we have discussed in the class and submit at the end of the class**
  - **1 for already answered questions**
  - **2 for typical questions**
  - **3 for questions with thoughts**
  - **4 for questions that surprised me**

Homework

- **Go over the next lecture slides before the class**
- **Watch 2 SIGGRAPH videos and submit your summaries every Tue. class**
  - **Just one paragraph for each summary**

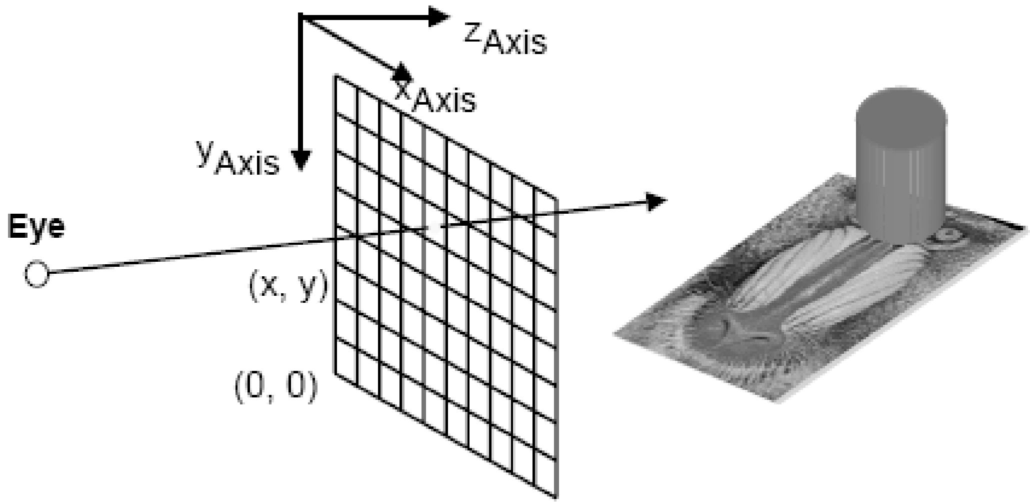
Example:

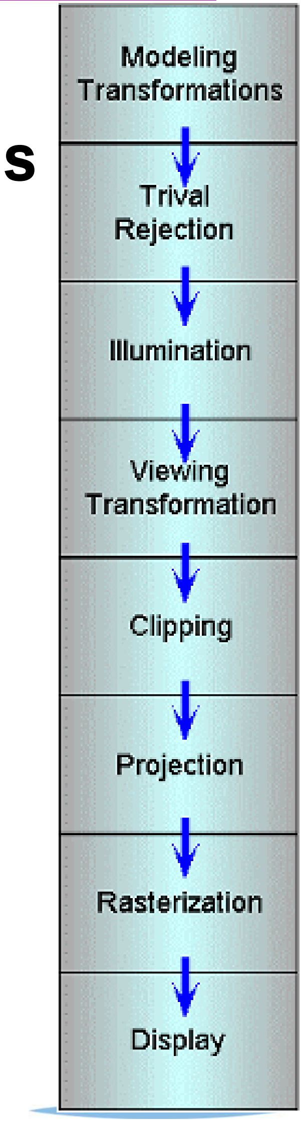
Title: XXX XXXX XXXX

Abstract: this video is about accelerating the performance of ray tracing. To achieve its goal, they design a new technique for reordering rays, since by doing so, they can improve the ray coherence and thus improve the overall performance.

Course Objectives were:

- Understand classic rendering pipeline
  - Just high-level concepts, not all the details
  - Brief introduction of common under. CG
- Know its pros and cons





Next Time

- **Ray tracing**