UNIVERSITY of HOUSTON

DEPARTMENT OF COMPUTER SCIENCE

COSC 4370 Fall 2023

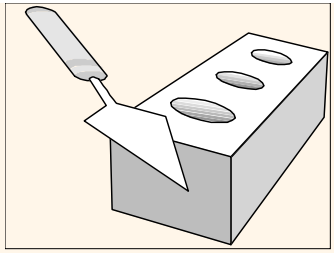
Interactive Computer Graphics

M & W 5:30 to 7:00 PM

Prof. Victoria Hilford

PLEASE TURN your webcam ON

NO CHATTING during LECTURE



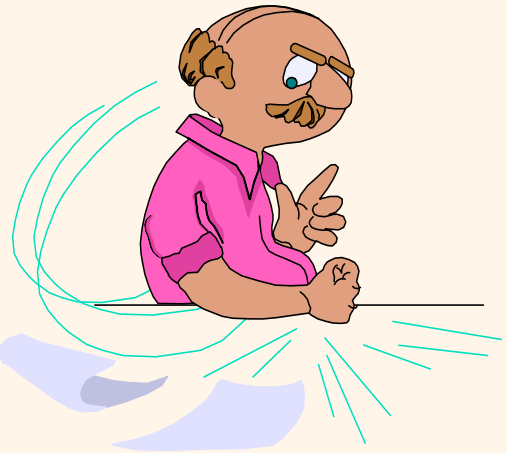
COSC 4370

5:30 to 7

PLEASE

LOG IN

CANVAS



Please close all other windows.

From 5:30 to 6:30 PM – 60 minutes.

08.23.2023 (W 5:30 to 7) (2)		Lecture 1
08.28.2023 (M 5:30 to 7) (3)	Homework 1	Lecture 2
08.30.2023 (W 5:30 to 7) (4)		Math Review 1
09.06.2023 (W 5:30 to 7) (5)	Homework 2	Lecture 3
09.11.2023 (M 5:30 to 7) (6)		Math Review 2
09.13.2023 (W 5:30 to 7) (7)	Homework 3	Lecture 4
09.18.2023 (M 5:30 to 7) (8)		PROJECT 1
09.20.2023 (W 5:30 to 7) (9)		EXAM 1 REVIEW
09.25.2023 (M 5:30 to 7) (10)		EXAM 1

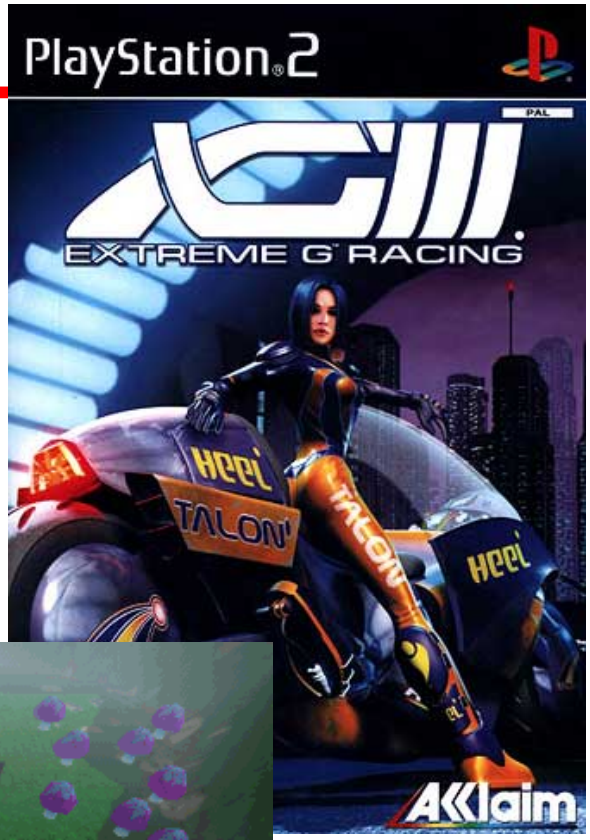
COSC 4370 – Computer Graphics

Lecture 1

Chapter 1

What is CG used for?

- computer games

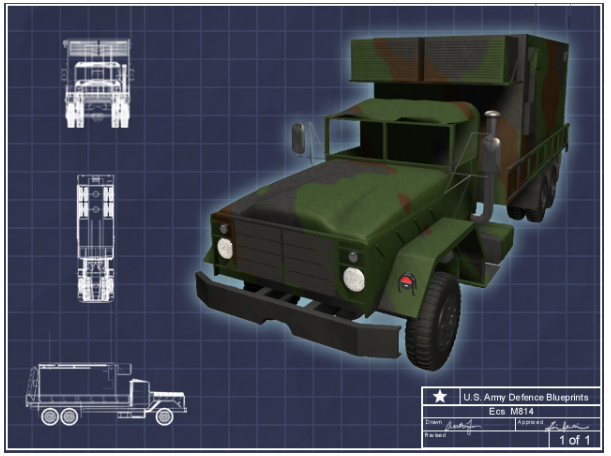


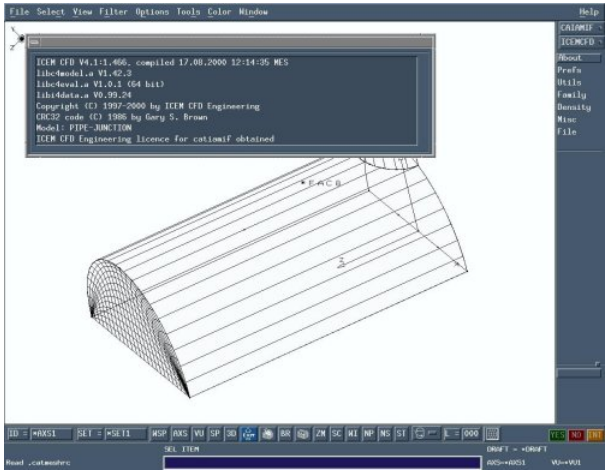
20 Billion Industry
Bypass movie & music Industry

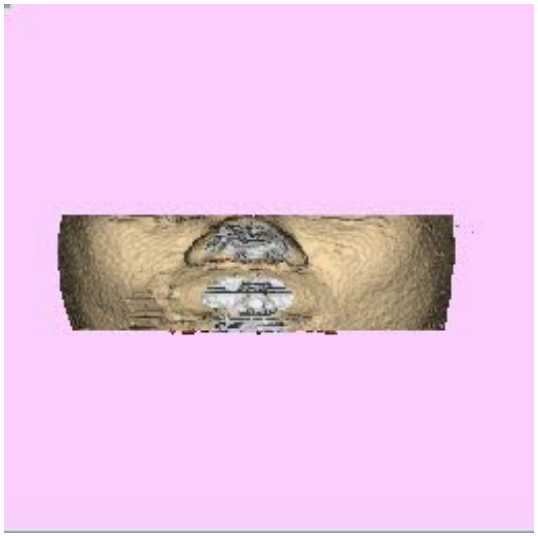
The University of New Mexico

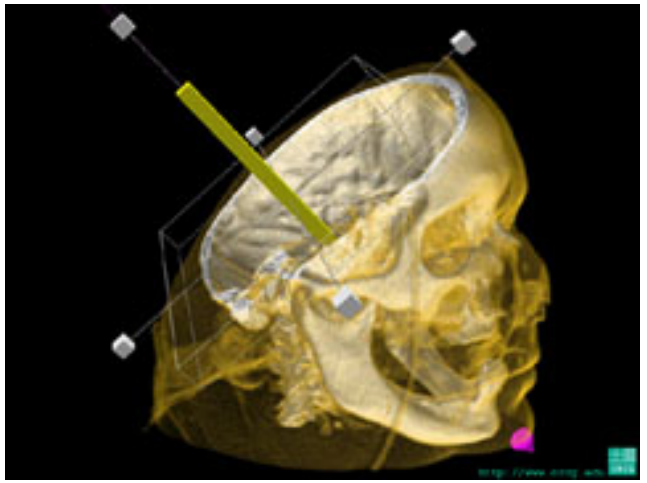
What is CG used for?

- Graphical User Interfaces (**GUI**)
  - modeling systems
  - applications
- Simulation & Visualization









What is CG used for?

- movies
  - animation
  - special effects

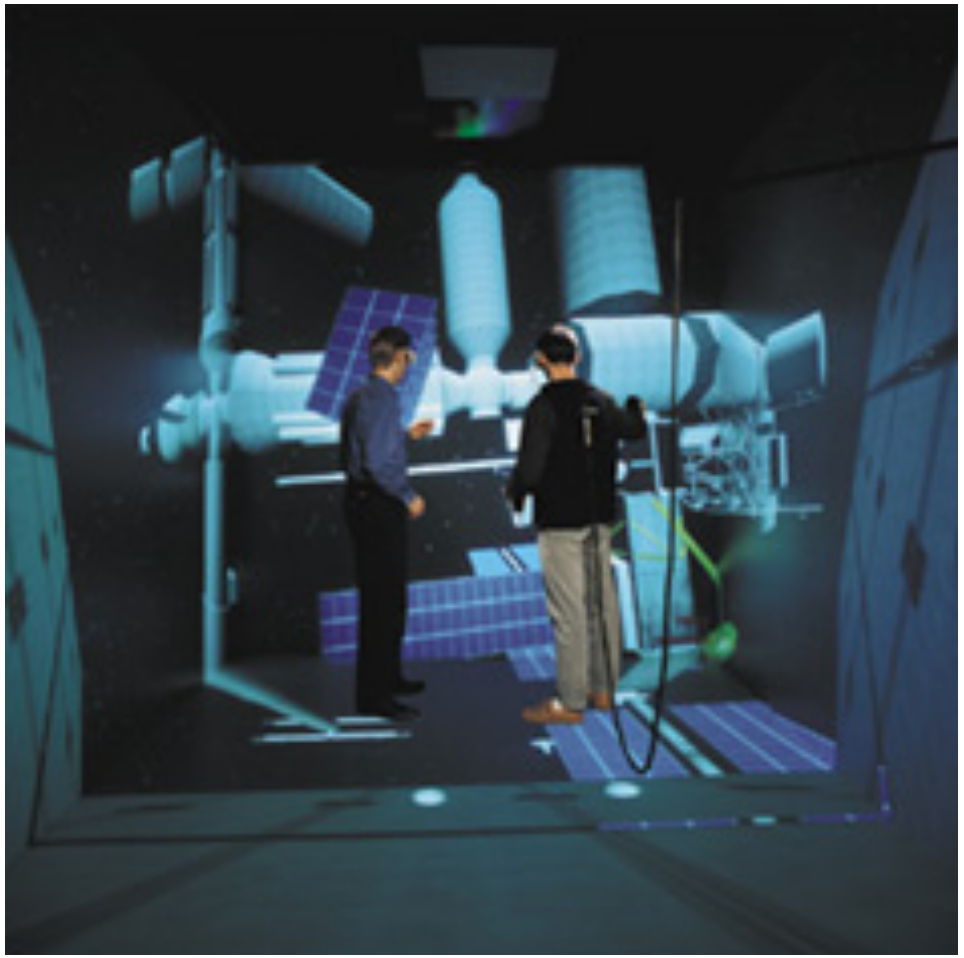
Inspector Gadget © 1999 Walt Disney Pictures.
Visual Effects by Dream Quest Images.

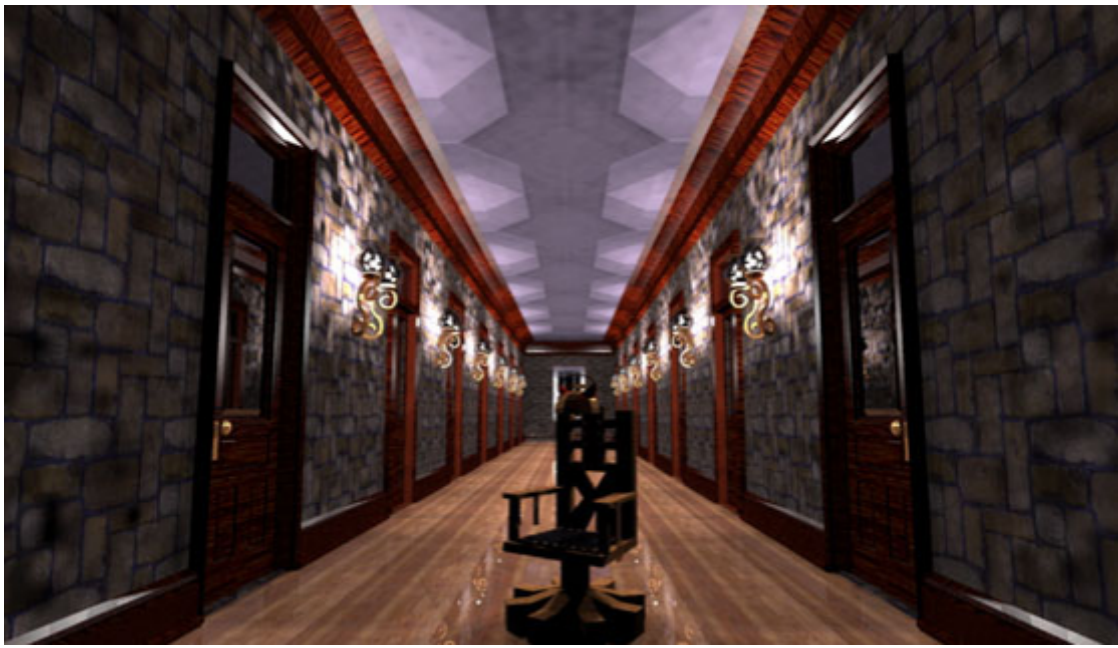
What is CG used for?

- images
  - advertising
  - design
  - art

What is CG used for?

- virtual reality / immersive displays

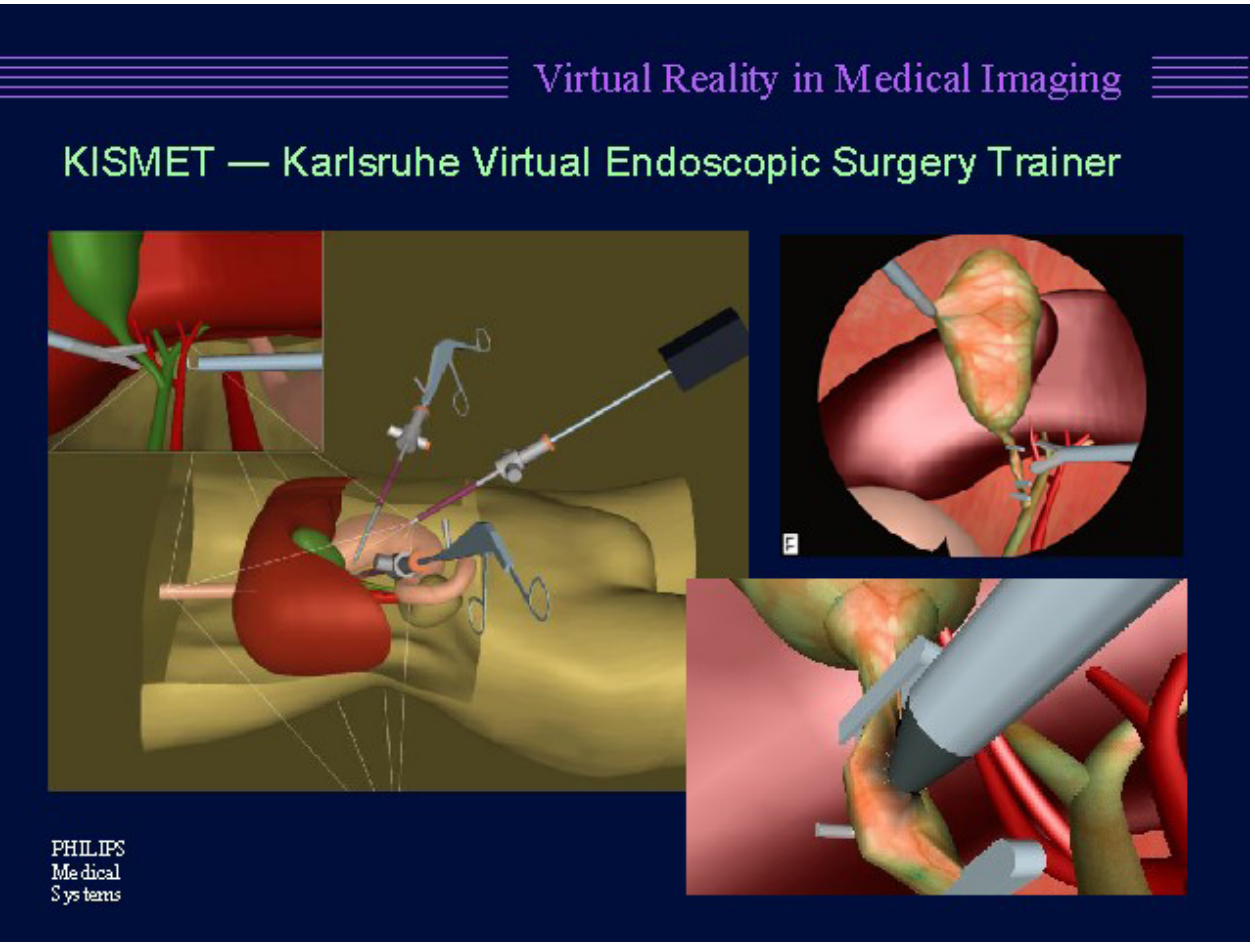


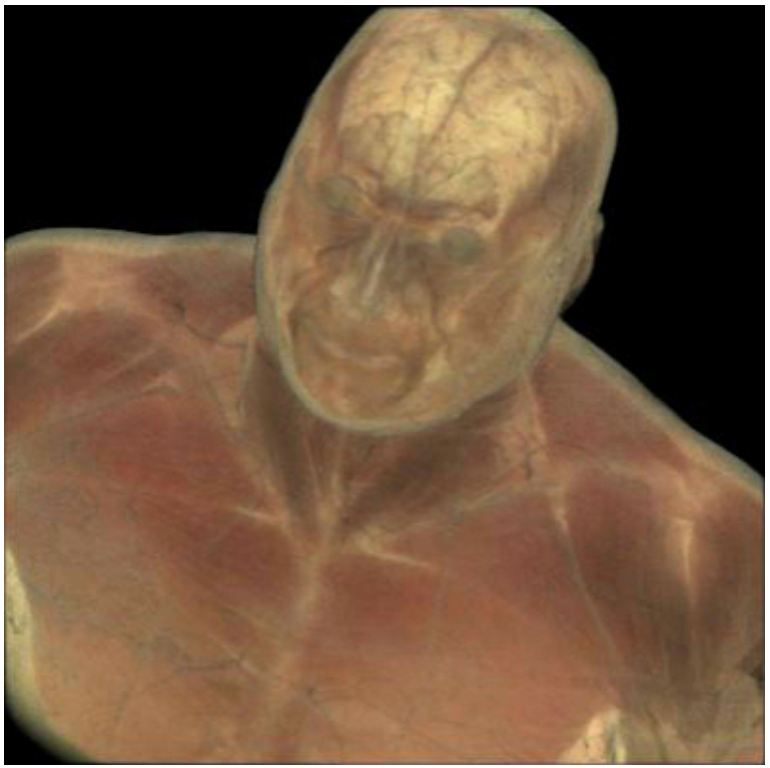


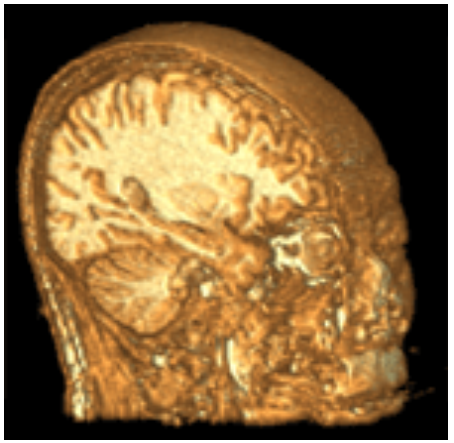
The University of New Mexico

What is CG used for?

- Virtual Surgery

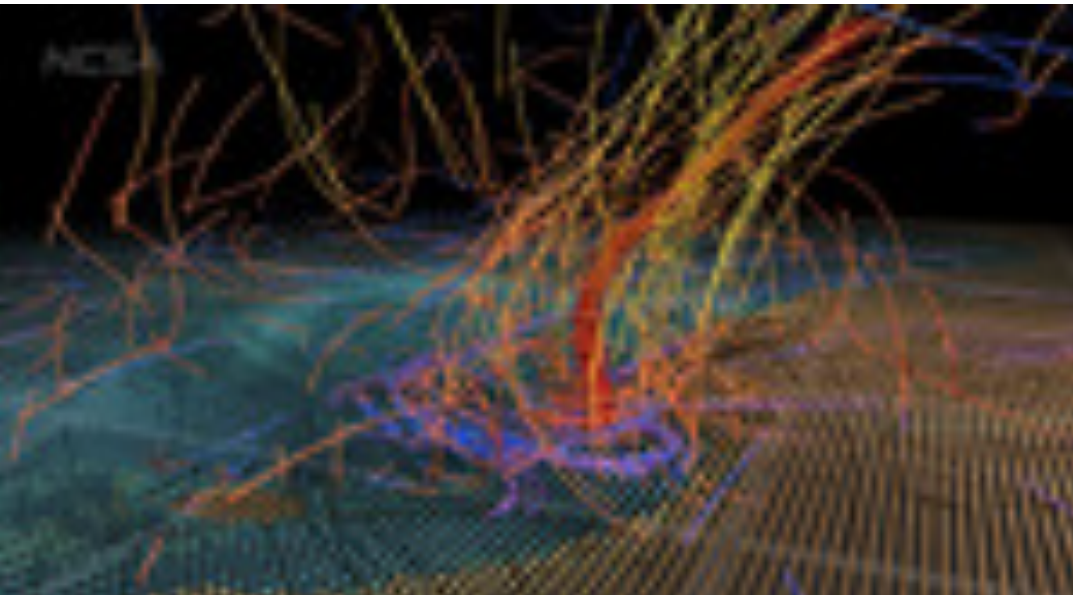


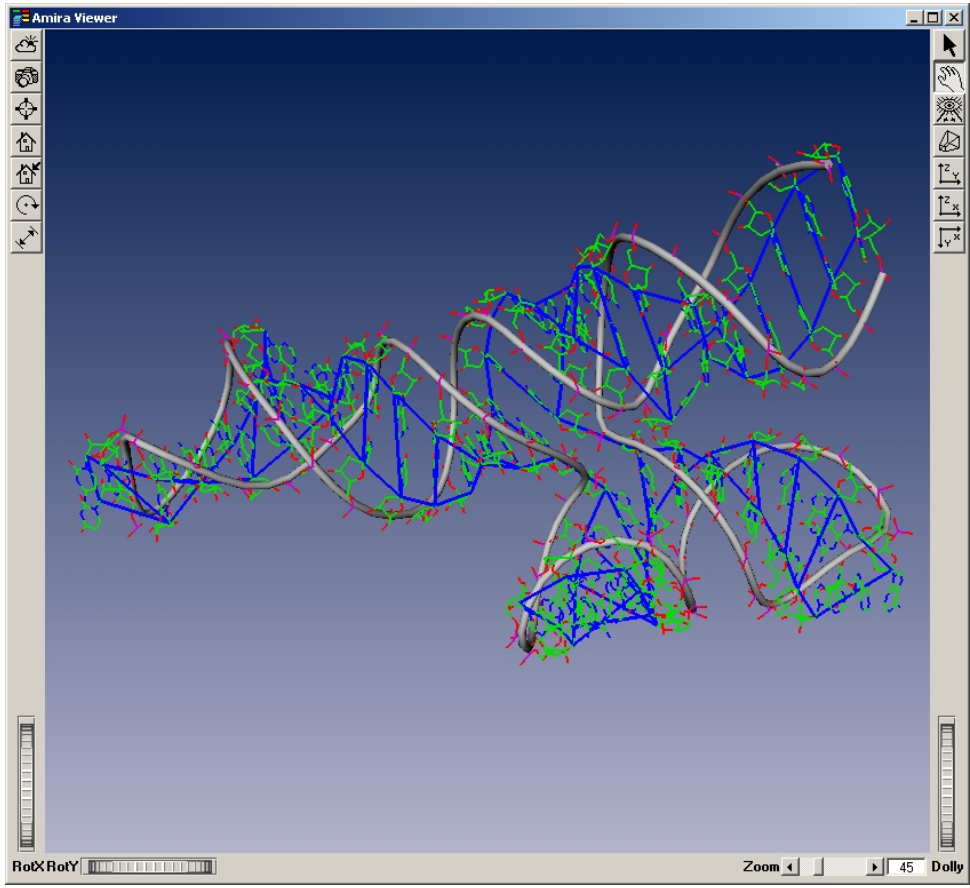




What is CG used for?

- Visualization Tools
  - Meteorology
  - Flow display



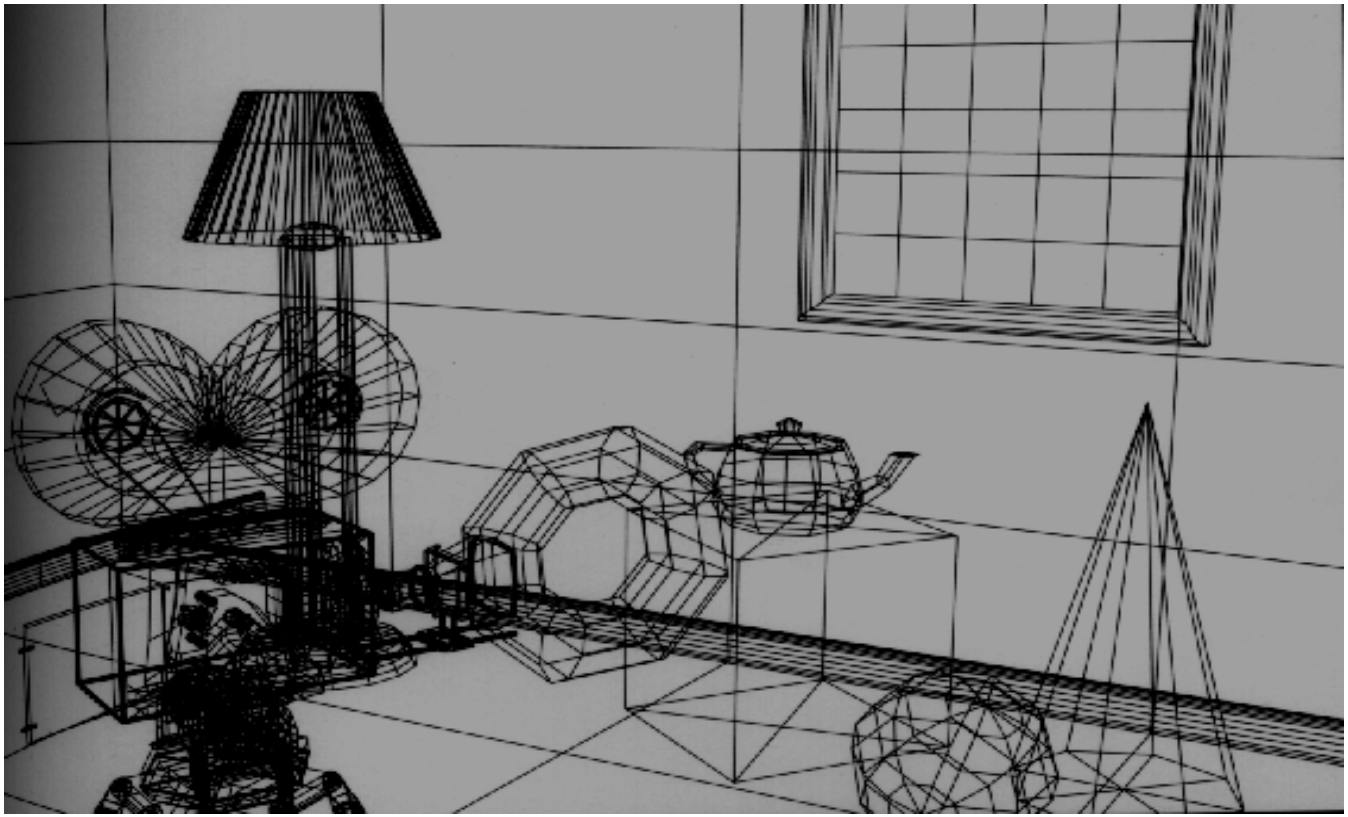


What is CG used for?

- generating **Models**

  lines, curves, polygons, smooth surfaces

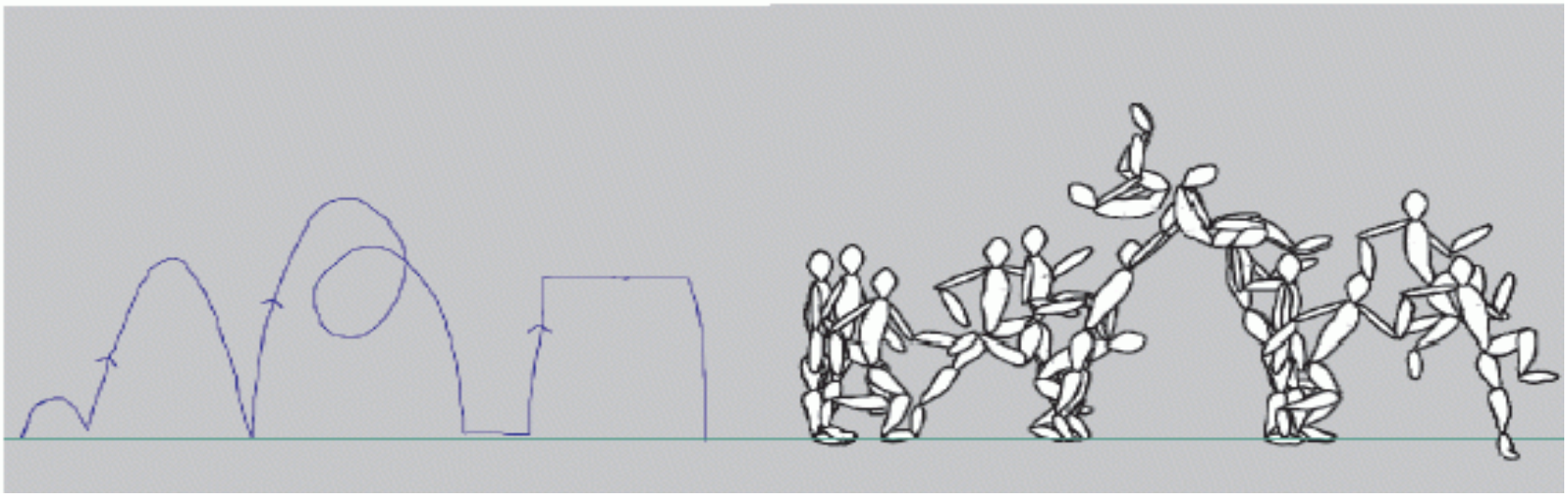
  digital geometry



What is CG used for?

- **Animation**: generating motion

  interpolating between frames, states



Motion Doodles: An Interface for Sketching Character Motion

http://www.cs.ubc.ca/~van/papers/doodle.html

CG or Real?

http://www.autodesk.com/eng/etc/fakeorfoto/quiz.html

1

CG or Real?

http://www.autodesk.com/eng/etc/fakeorfoto/quiz.html

2

CG or Real?

http://www.autodesk.com/eng/etc/fakeorfoto/quiz.html

3

CG or Real?

http://www.autodesk.com/eng/etc/fakeorfoto/quiz.html

4

CG or Real?

http://www.autodesk.com/eng/etc/fakeorfoto/quiz.html

5

Real

CG

Rendering

creating images from models

geometric objects

- lines, polygons, curves, curved surfaces

camera

- pinhole camera, lens systems, orthogonal

shading

- light interacting with material

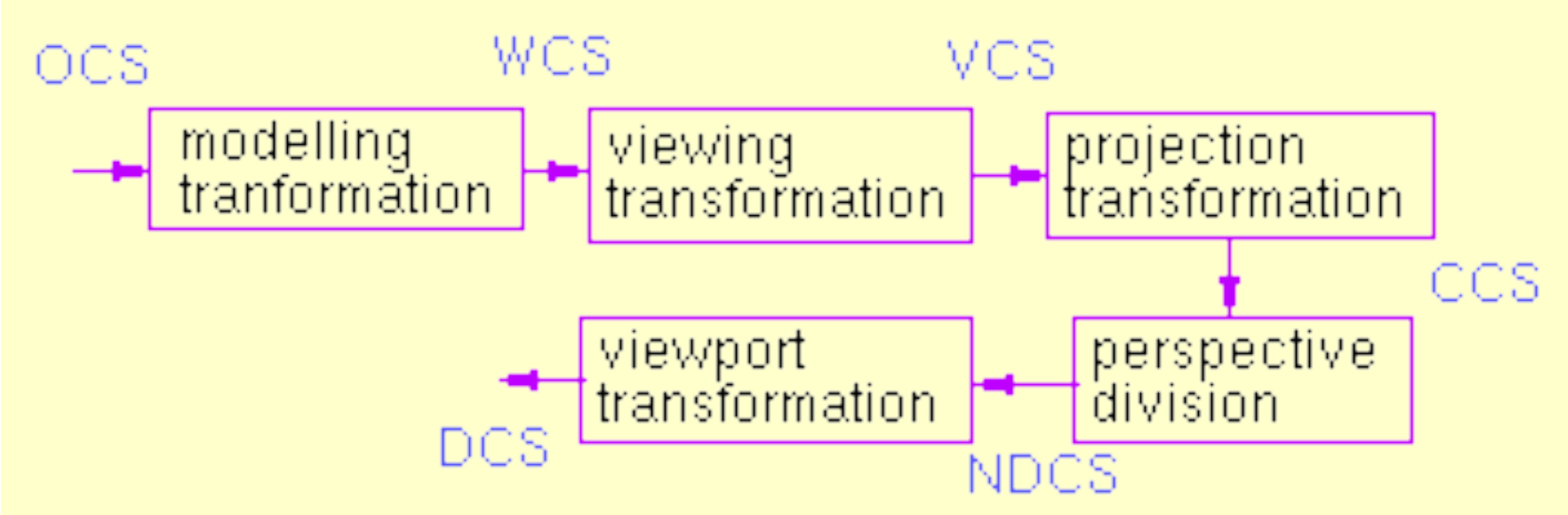
illustration of rendering capabilities

Tron Legacy 2010

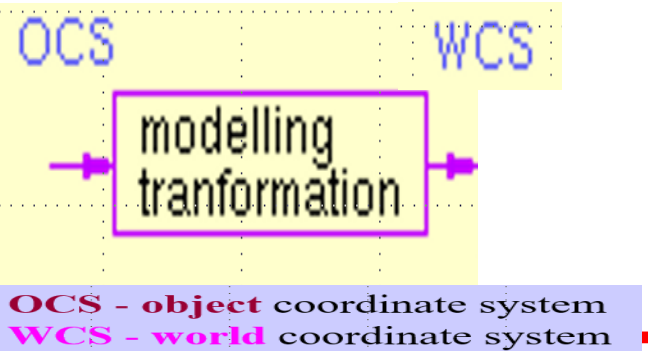
http://www.youtube.com/watch?v=a1IpPpB3iWI

http://design.osu.edu/carlson/history/tron.html

Stages of Vertex Transformations

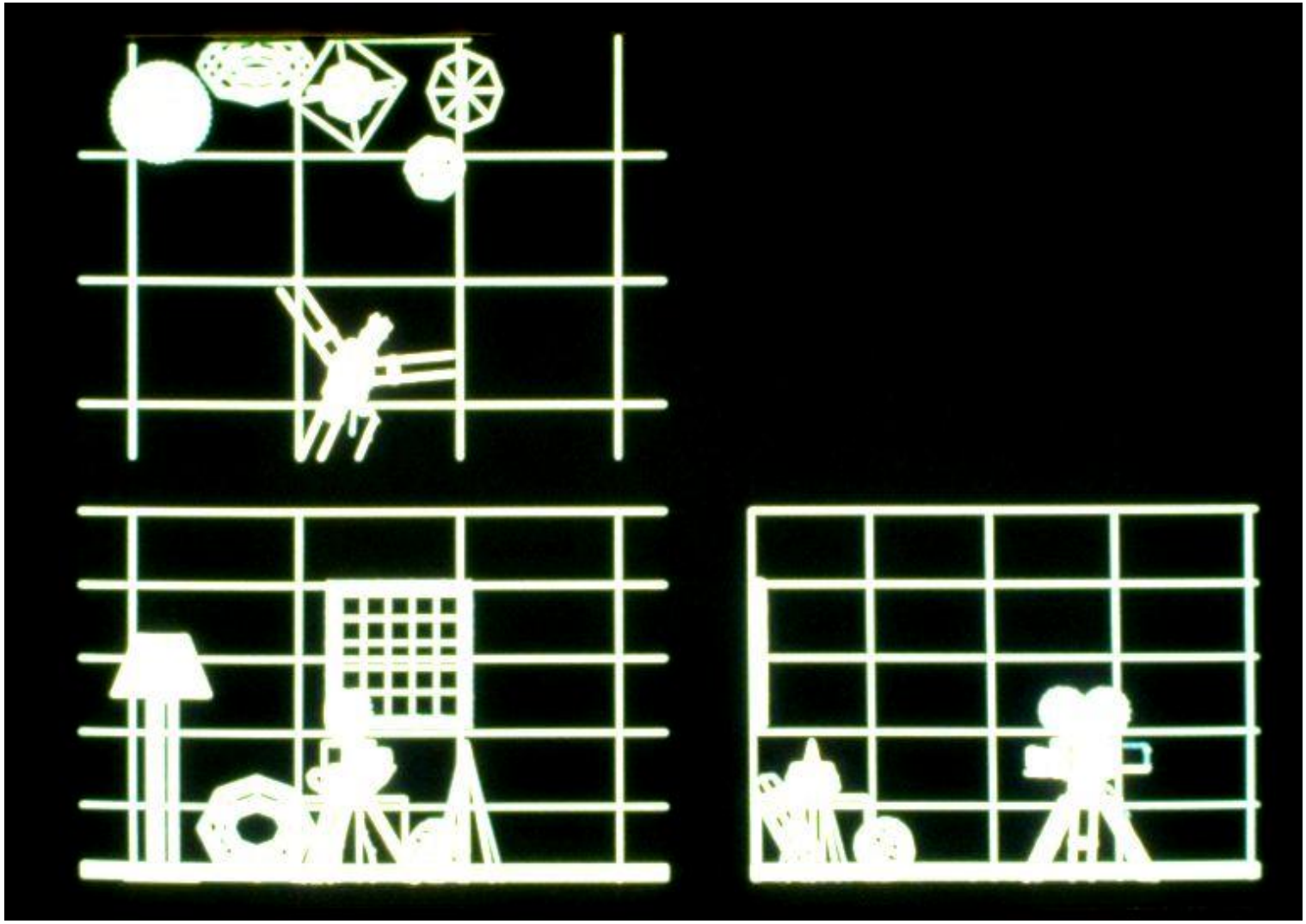


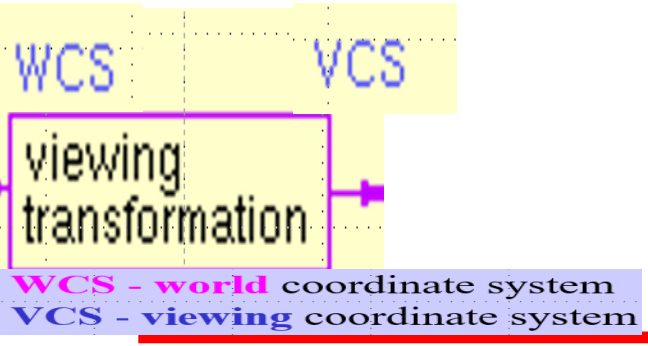
OCS - object coordinate system
WCS - world coordinate system
VCS - viewing coordinate system
CCS - clipping coordinate system
NDCS - normalized device coordinate system
DCS - device coordinate system



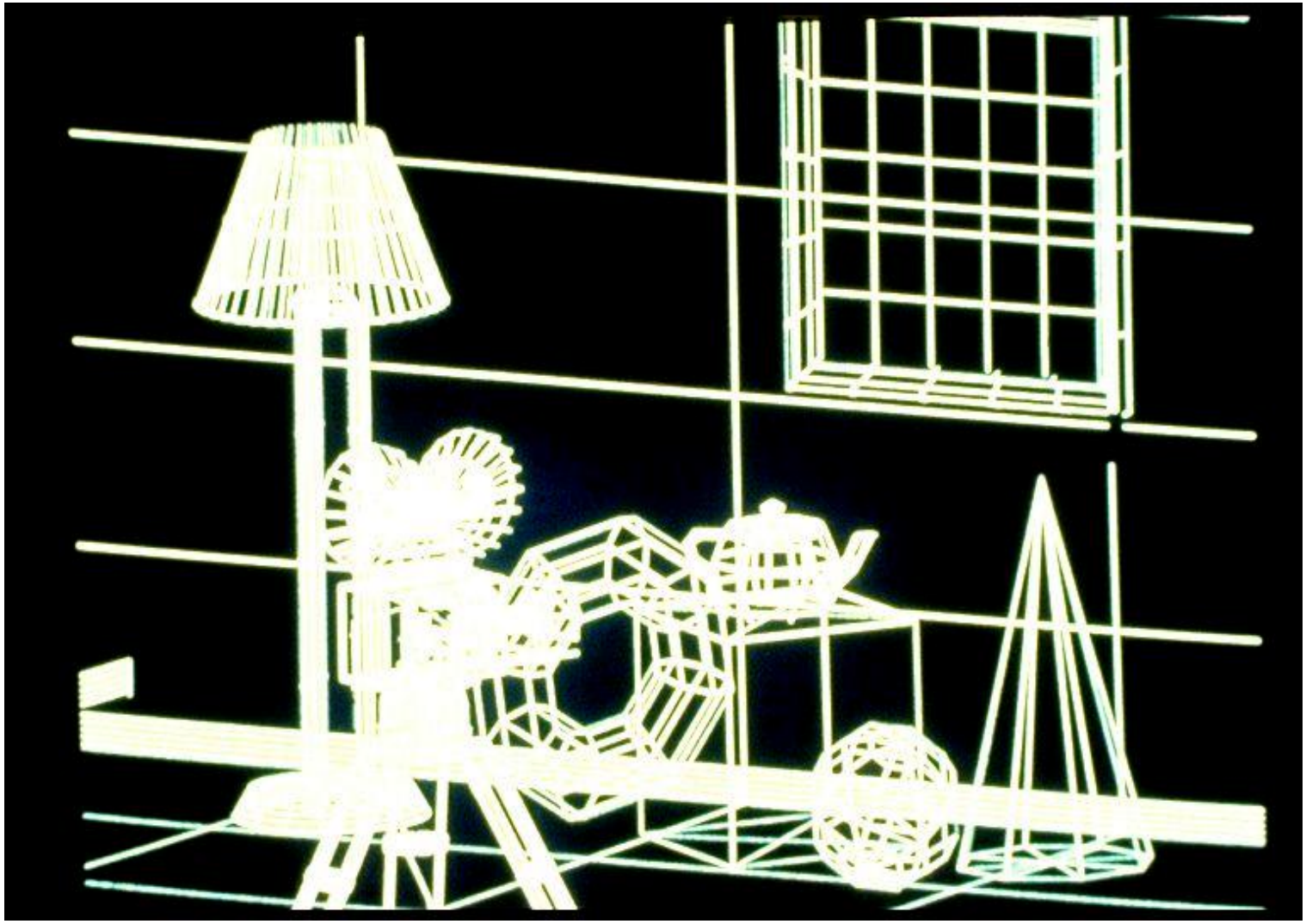
OCS - **object** coordinate system
WCS - **world** coordinate system

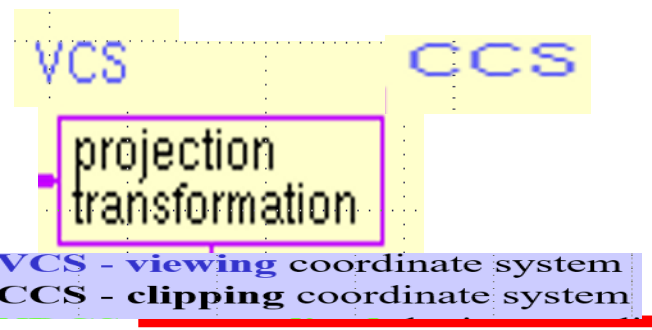
Modeling Transformation: Object Placement



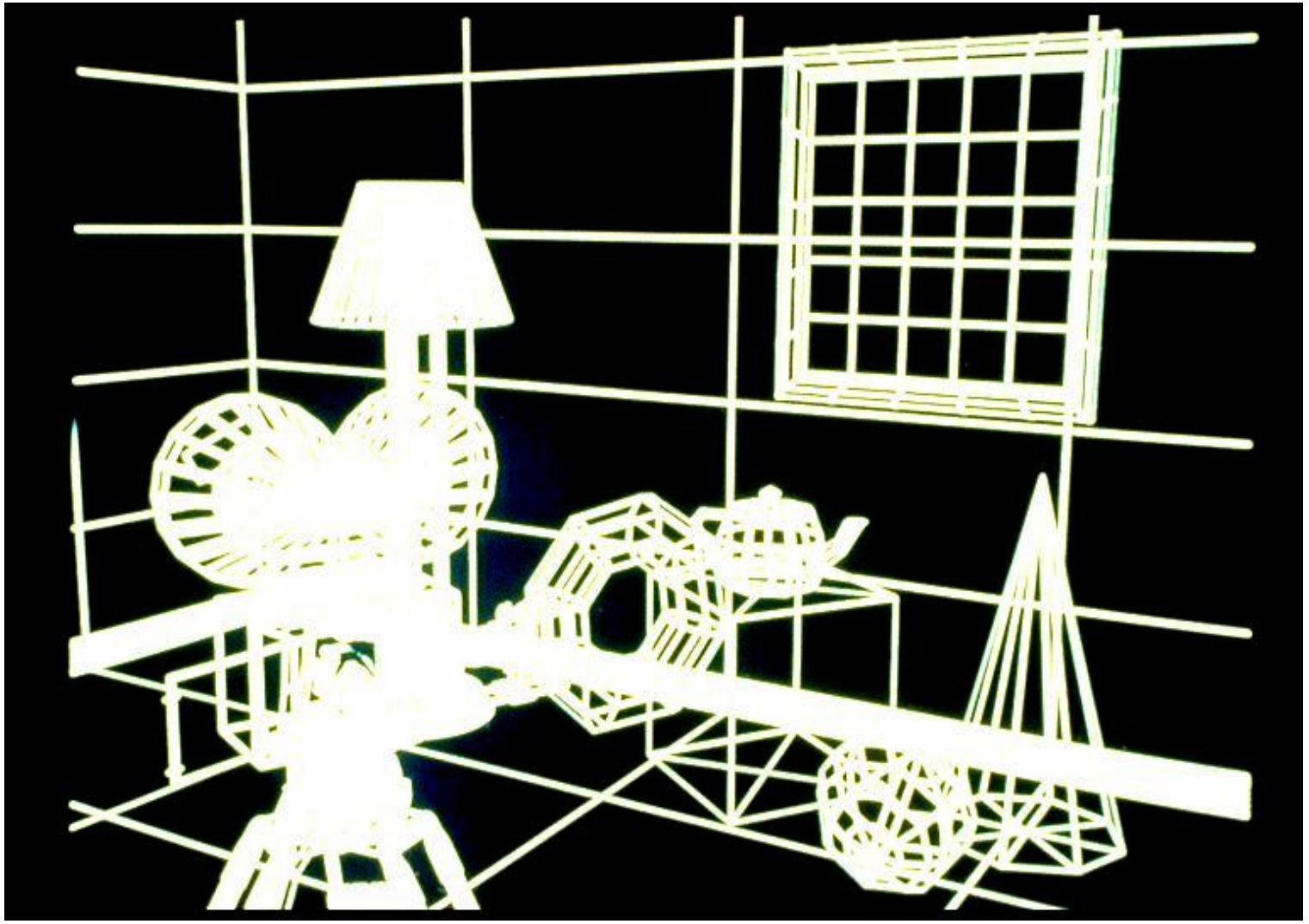


Viewing Transformation: Camera Placement

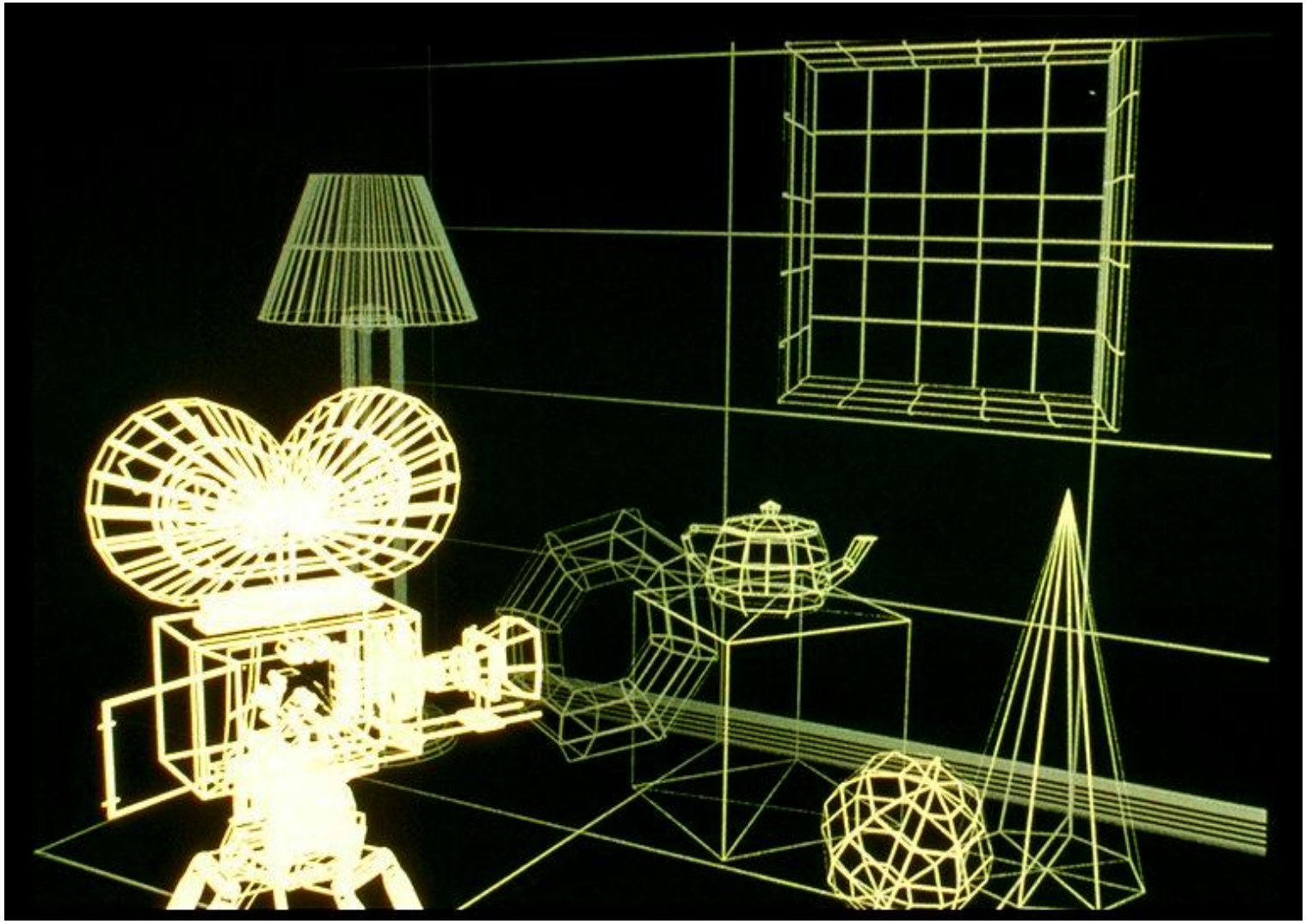




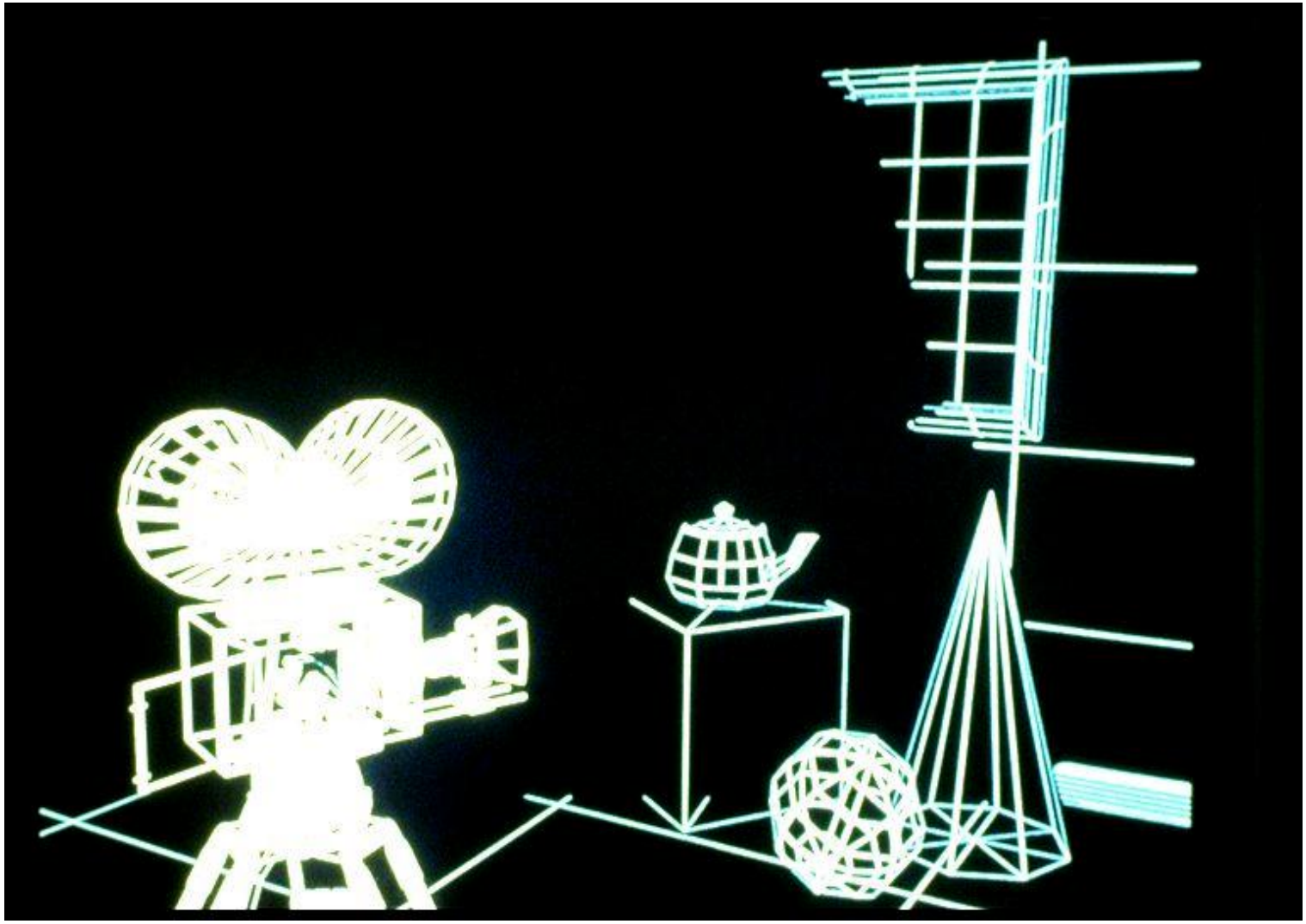
Perspective Projection



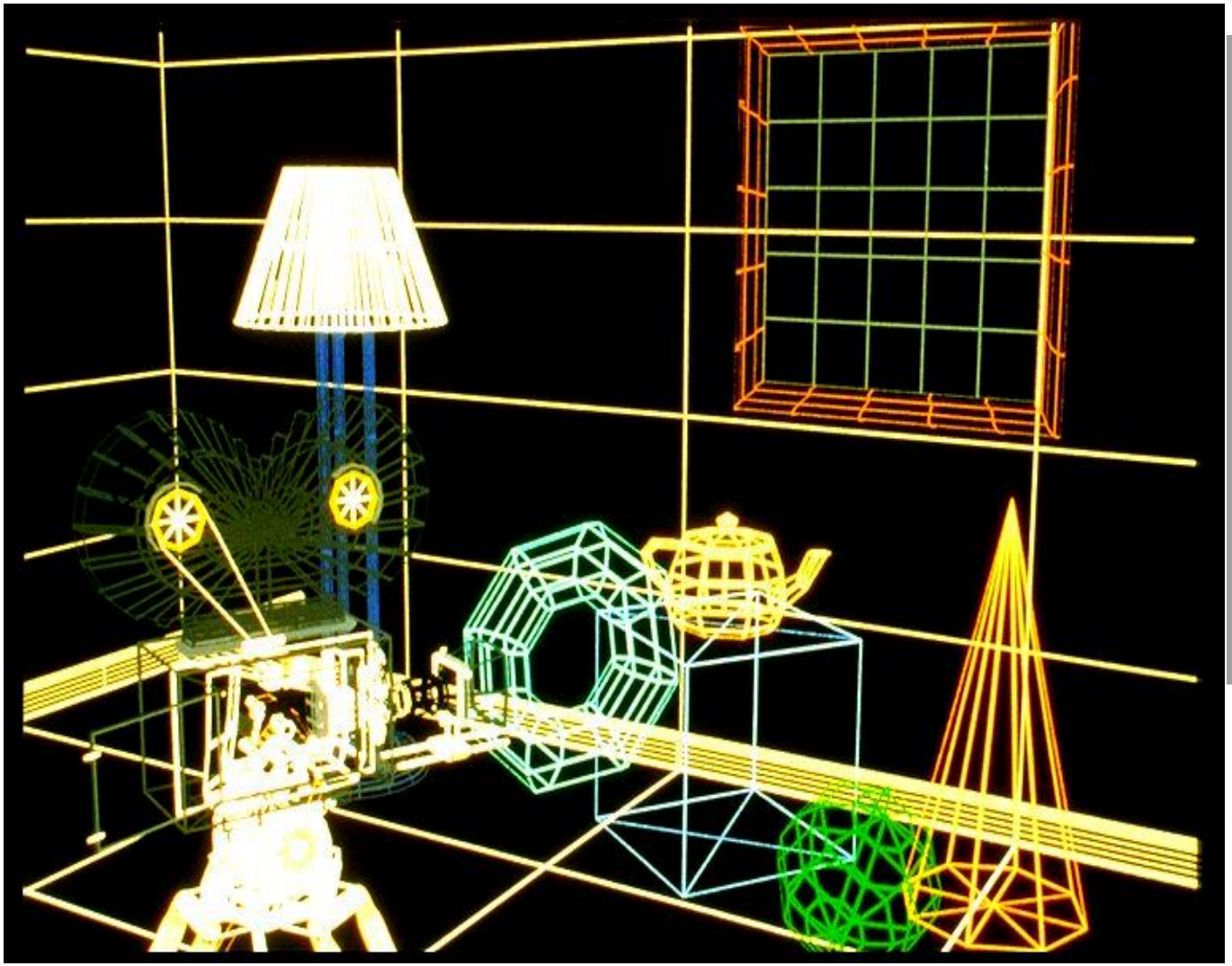
Depth Cueing

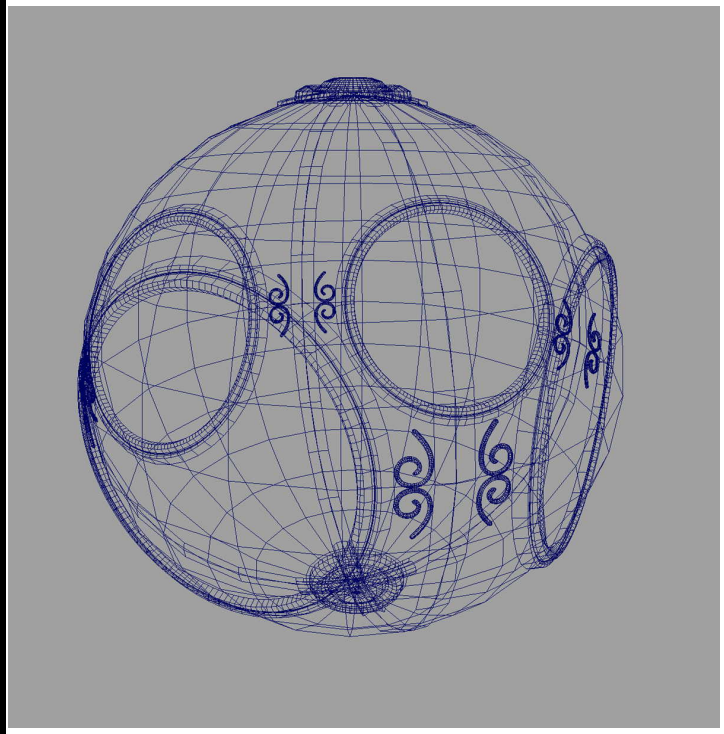


Depth Clipping



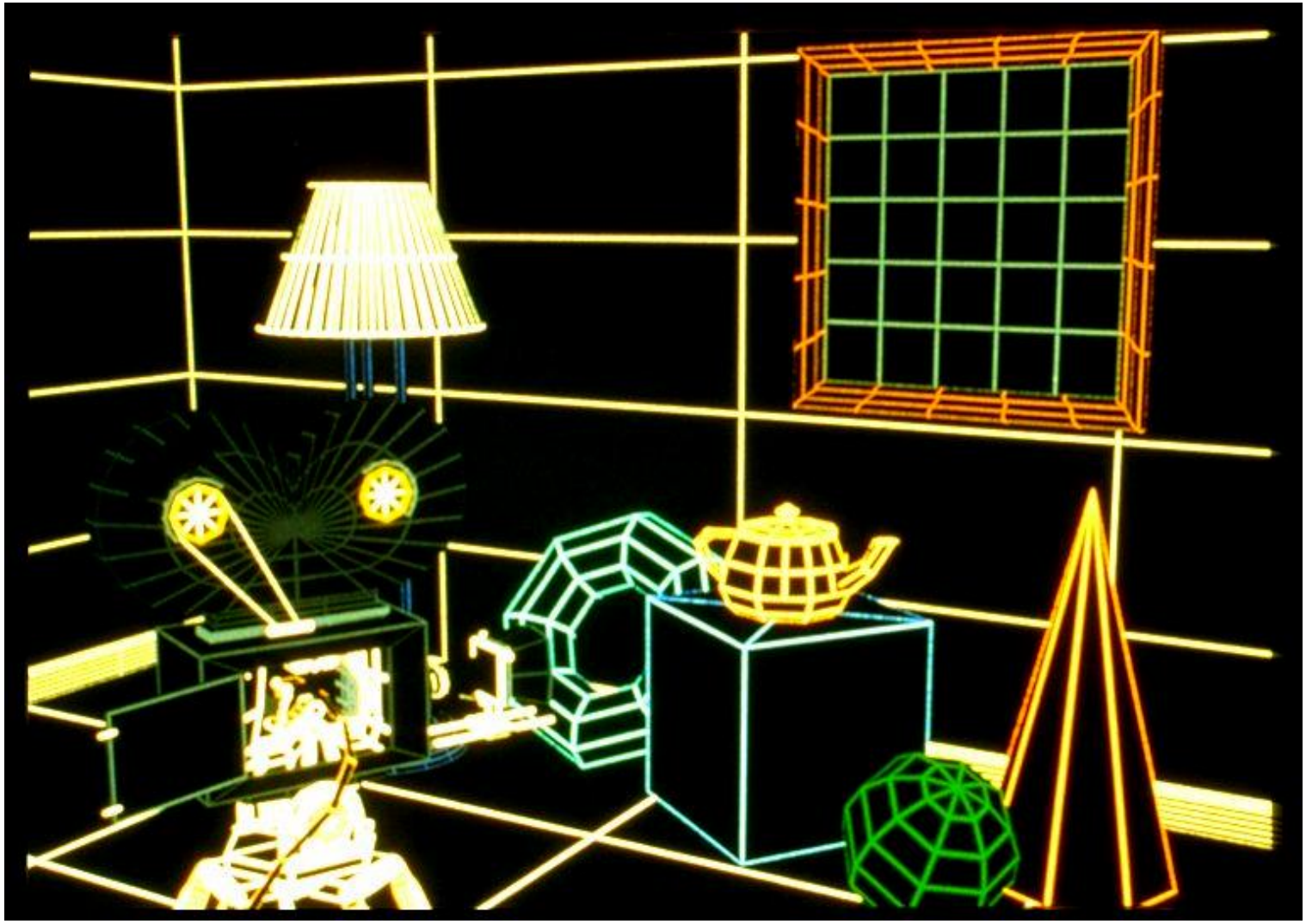
Colored Wireframes



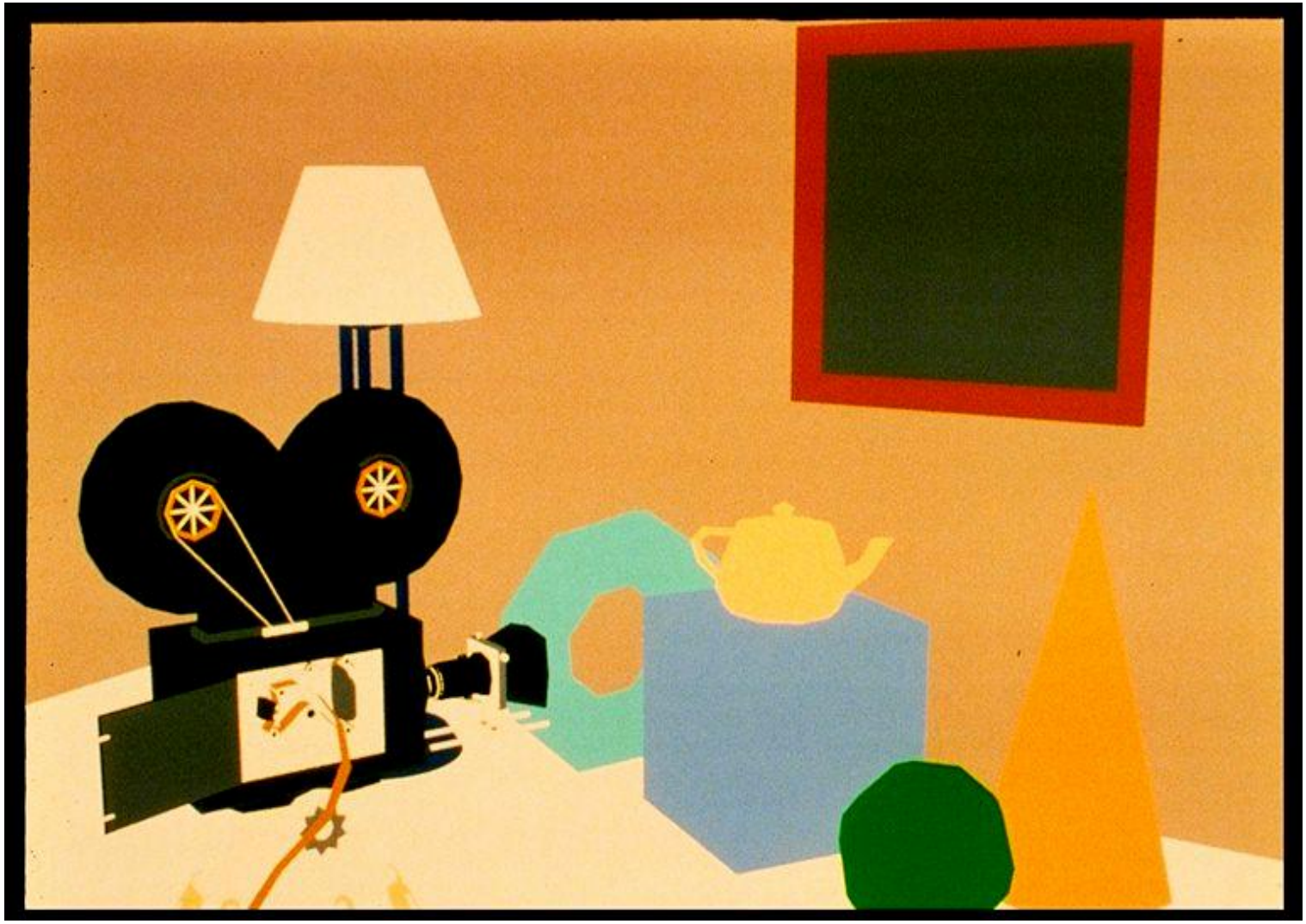


wireframe representation of sun object

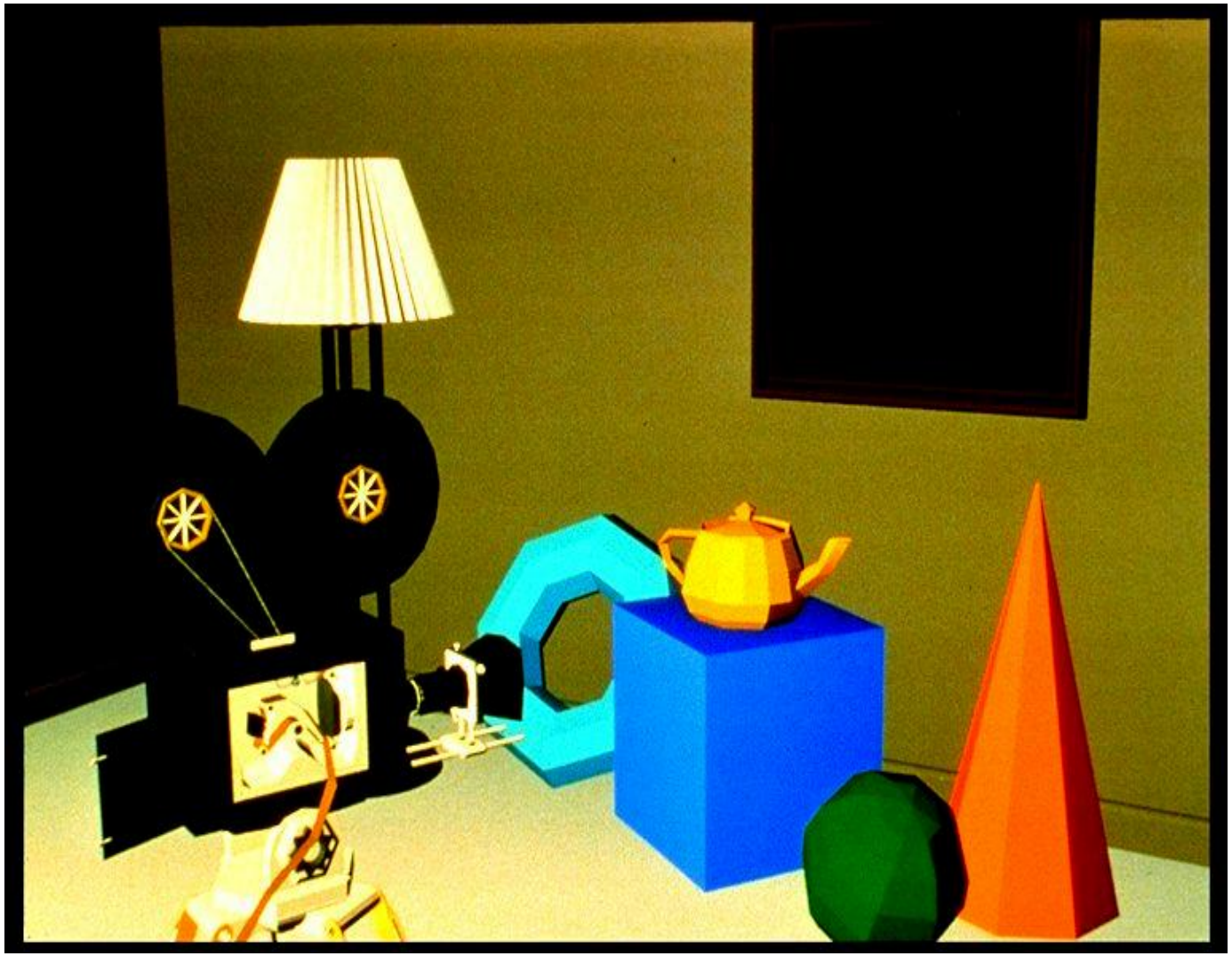
Hidden Line Removal

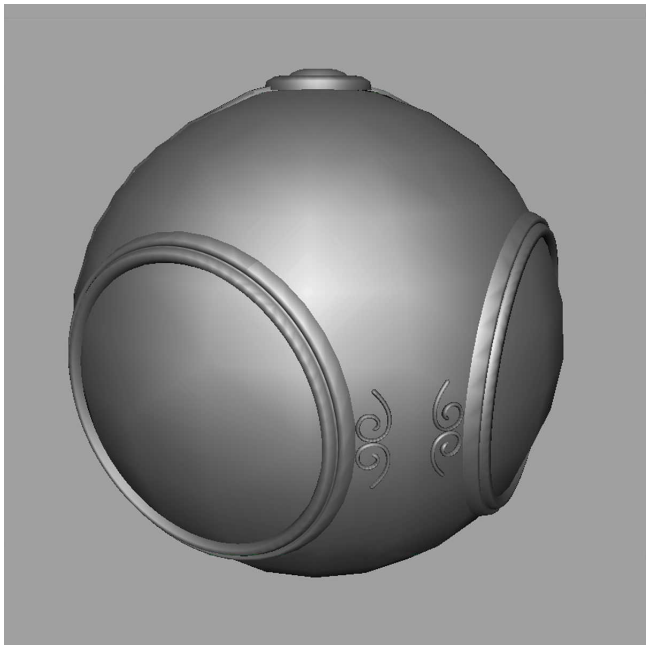


Hidden Surface Removal



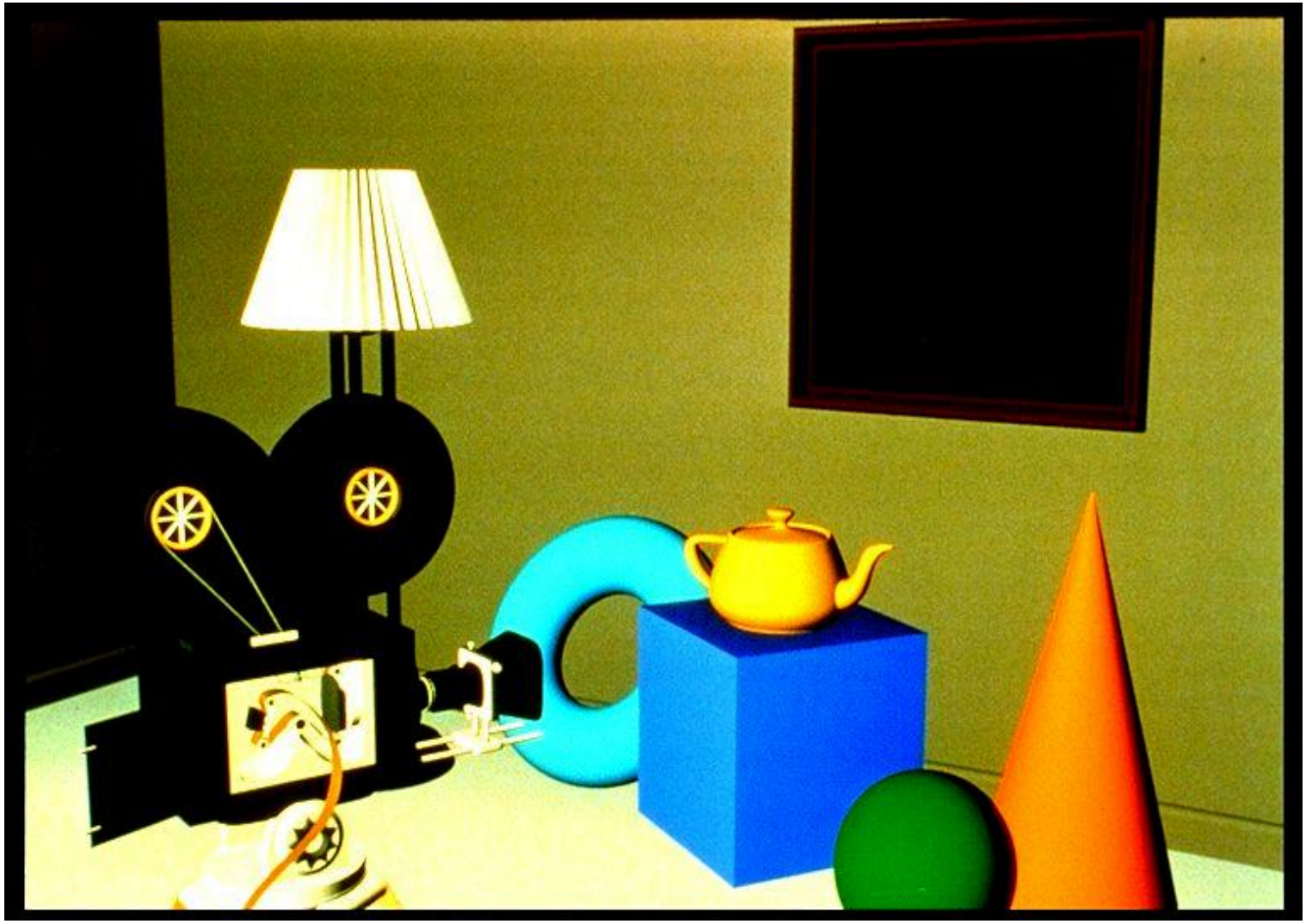
Per-Polygon Flat Shading



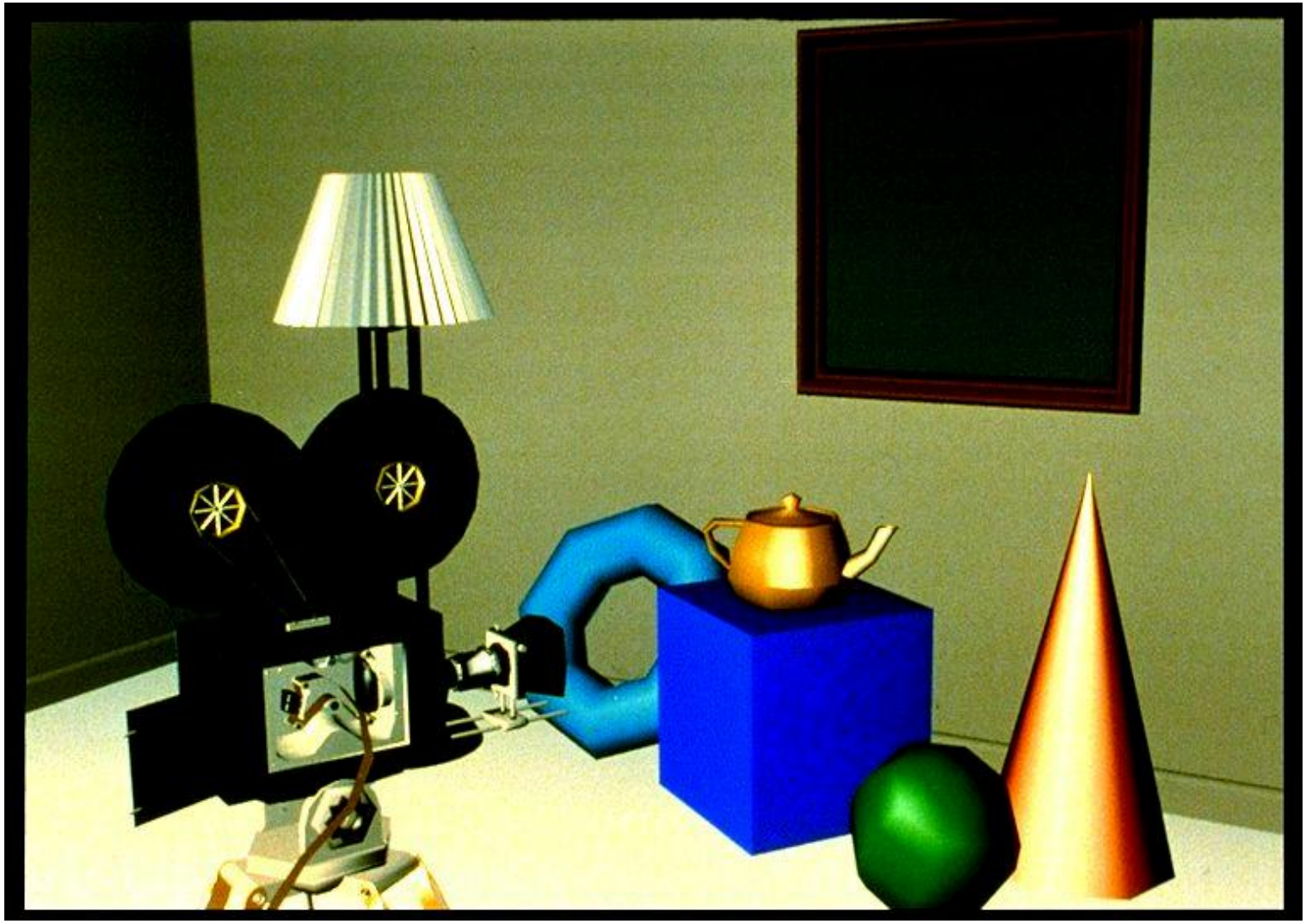


smooth shading

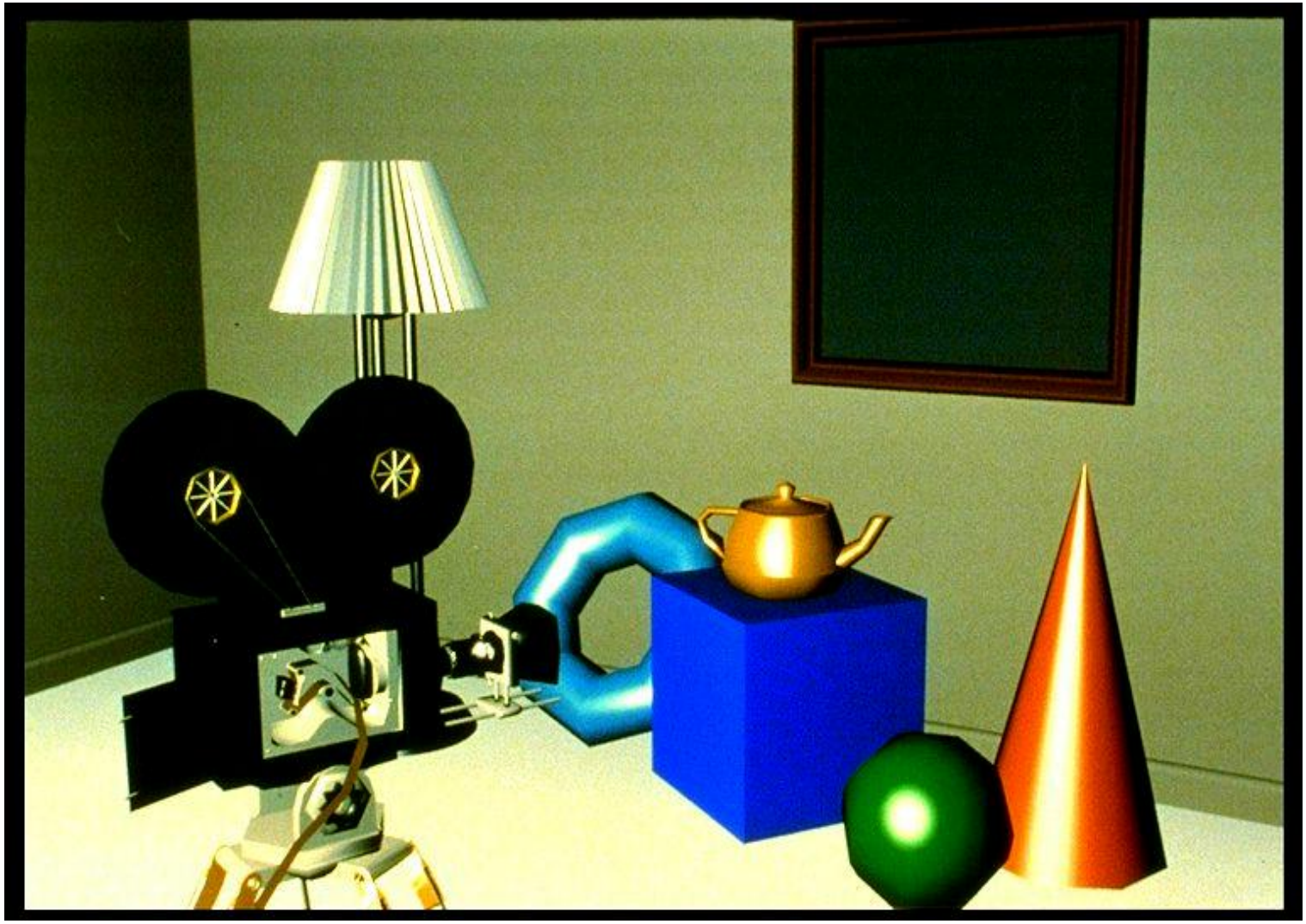
Gouraud Shading



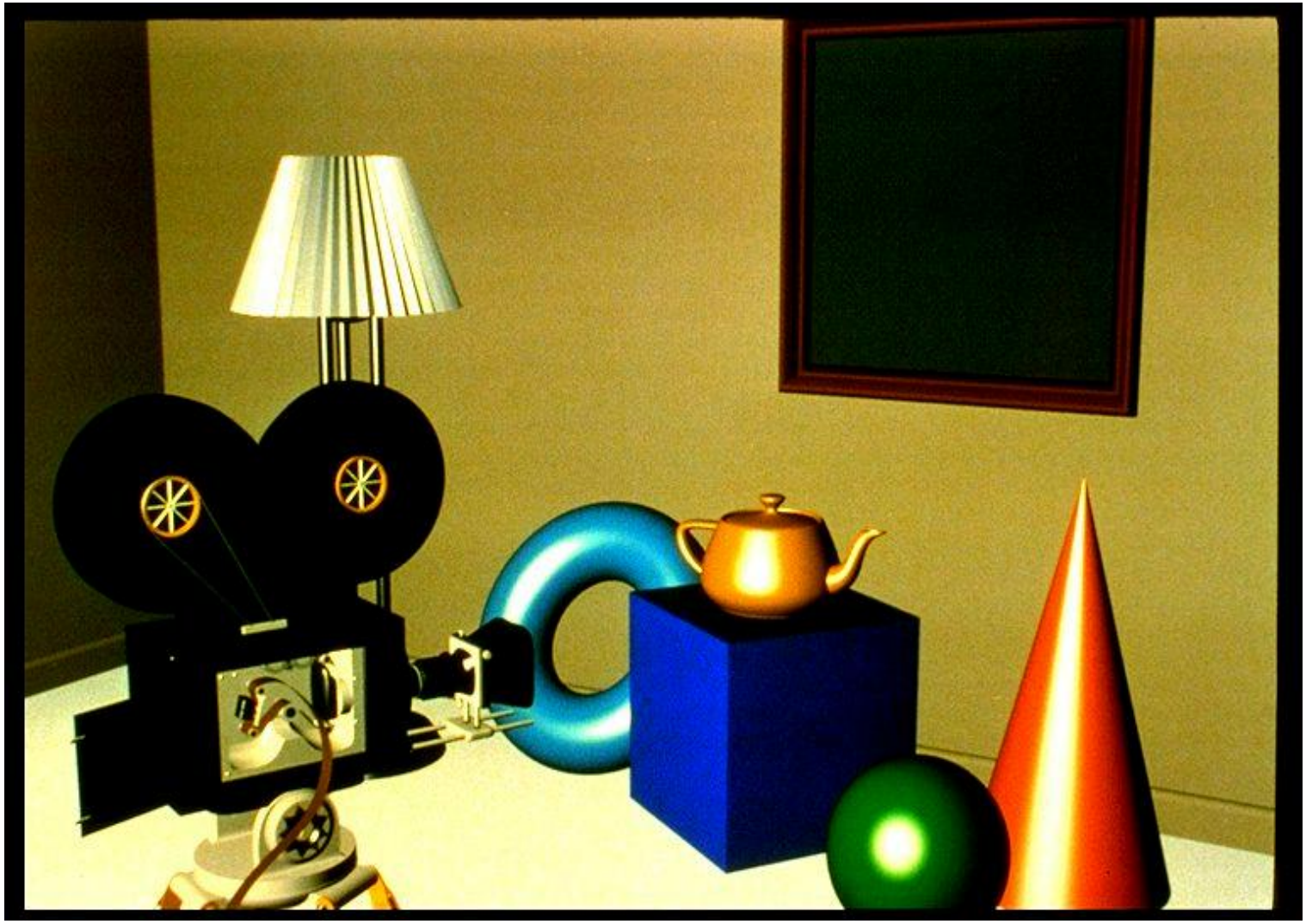
Specular Reflection



Phong Shading

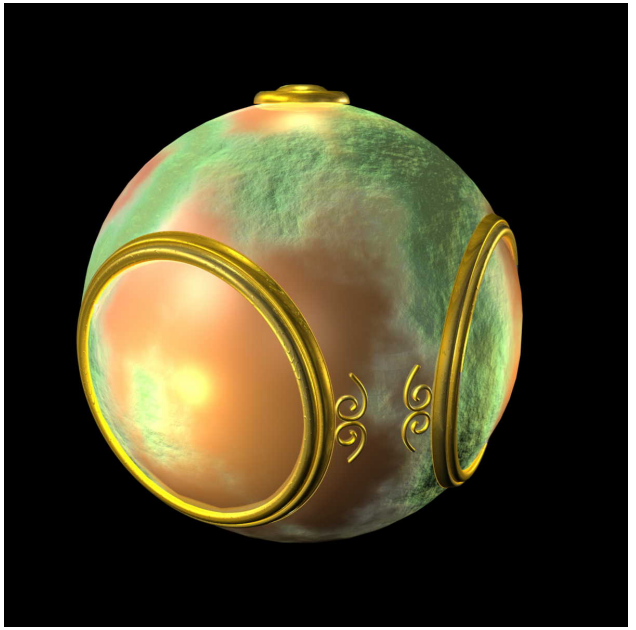


Curved Surfaces



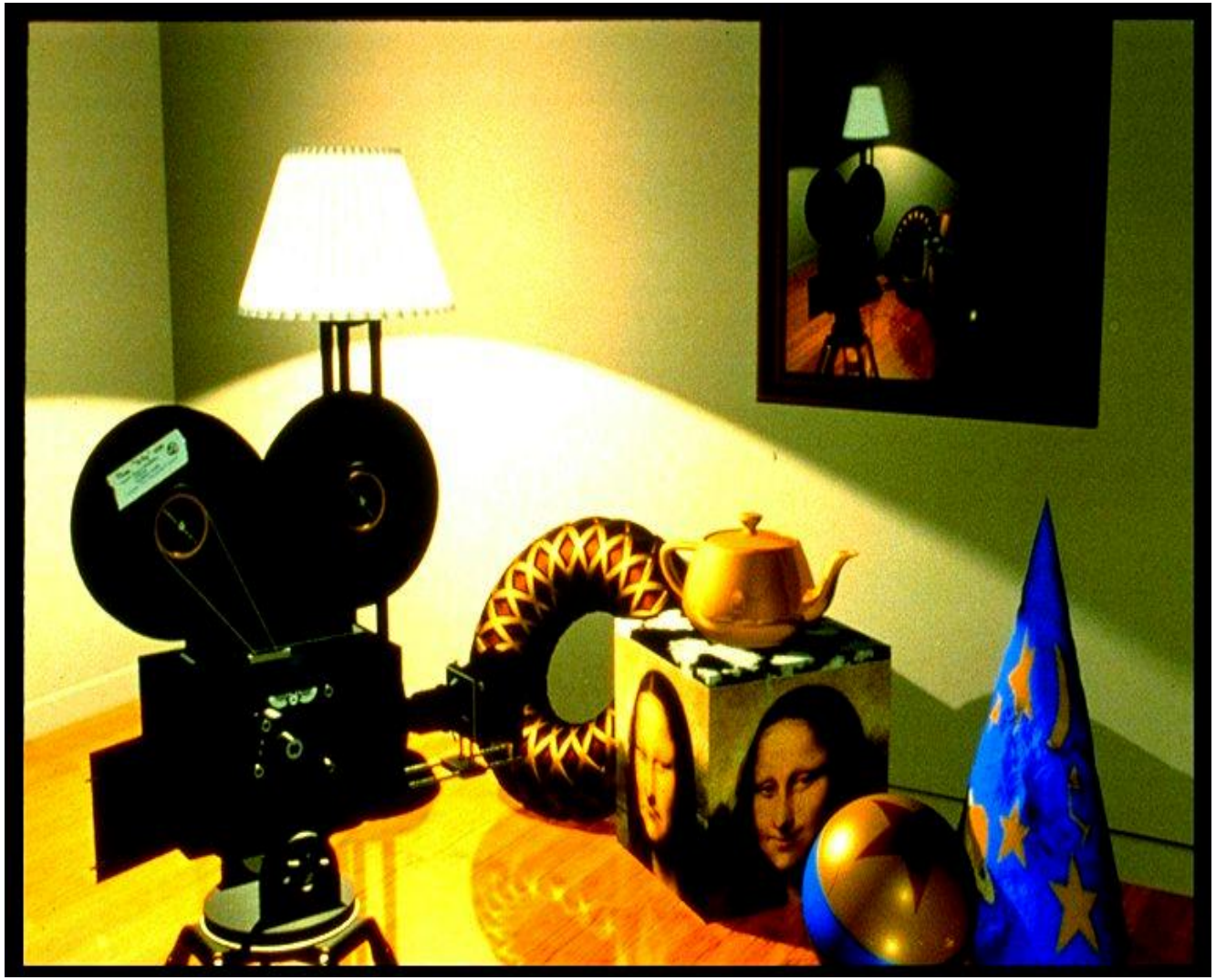
Complex Lighting and Shading

Texture Mapping



bump mapping

Reflection Mapping



environment mapping

Image Formation
Chapter 1.3

Imaging systems

- In **computer graphics, we form images which are generally two dimensional** using a process analogous to how images are formed by physical imaging systems

  Pinhole Cameras

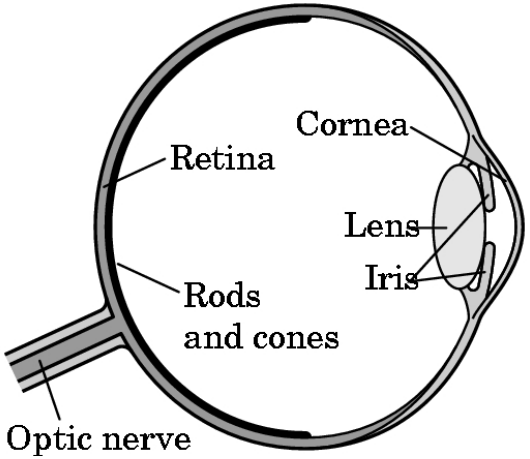
  Microscopes

  Telescopes

  Human visual system

Image formation with the Human Visual System

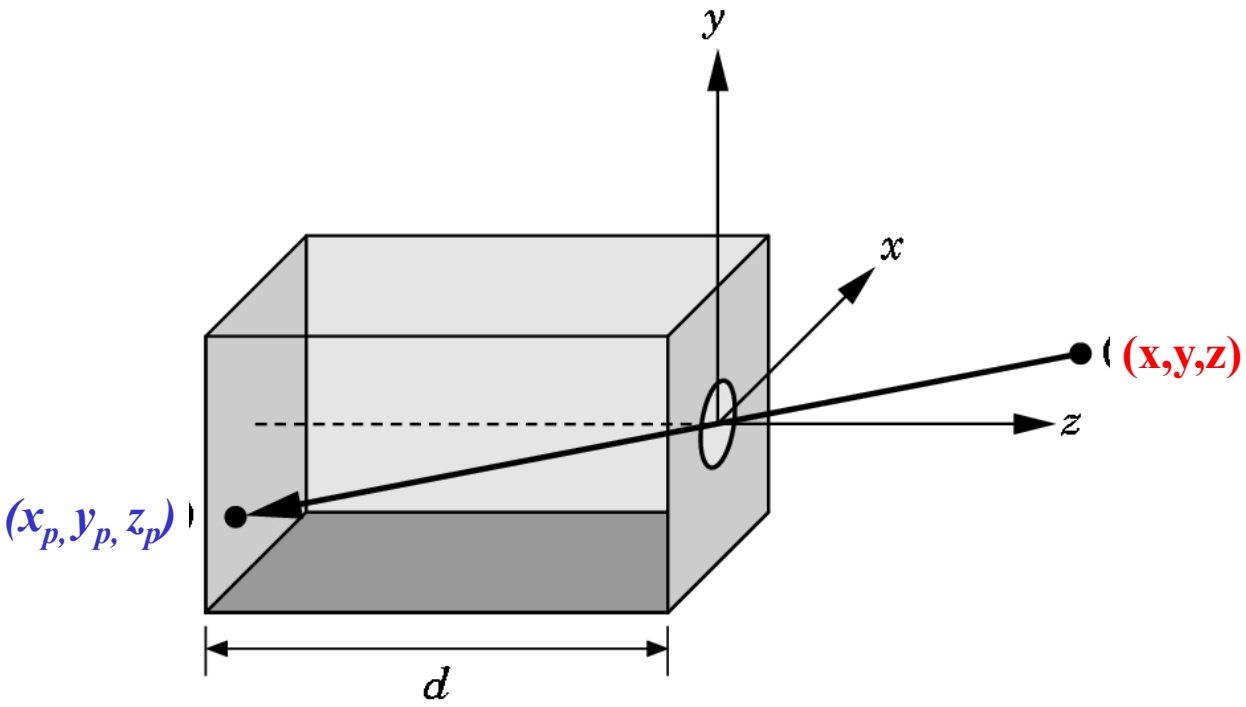
- Human visual system has two types of sensors
  - Rods: monochromatic, night vision
  - Cones
    - Color sensitive
    - Three types of cones



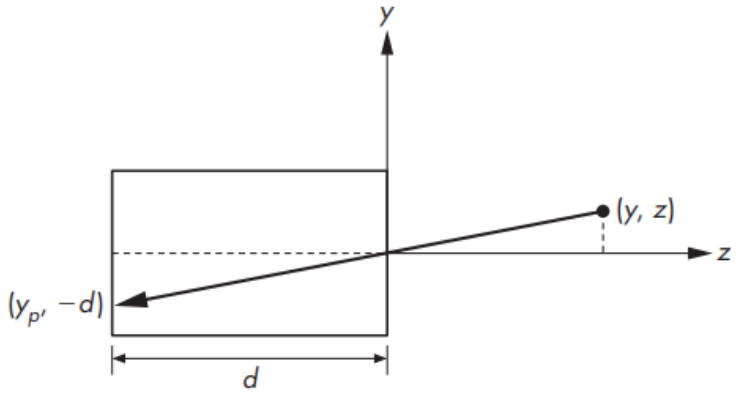
    - Only three values (the *tristimulus* values) are sent to the brain

- Need only match these three values
  - Need only three *primary* colors

Image formation with the Pinhole Camera



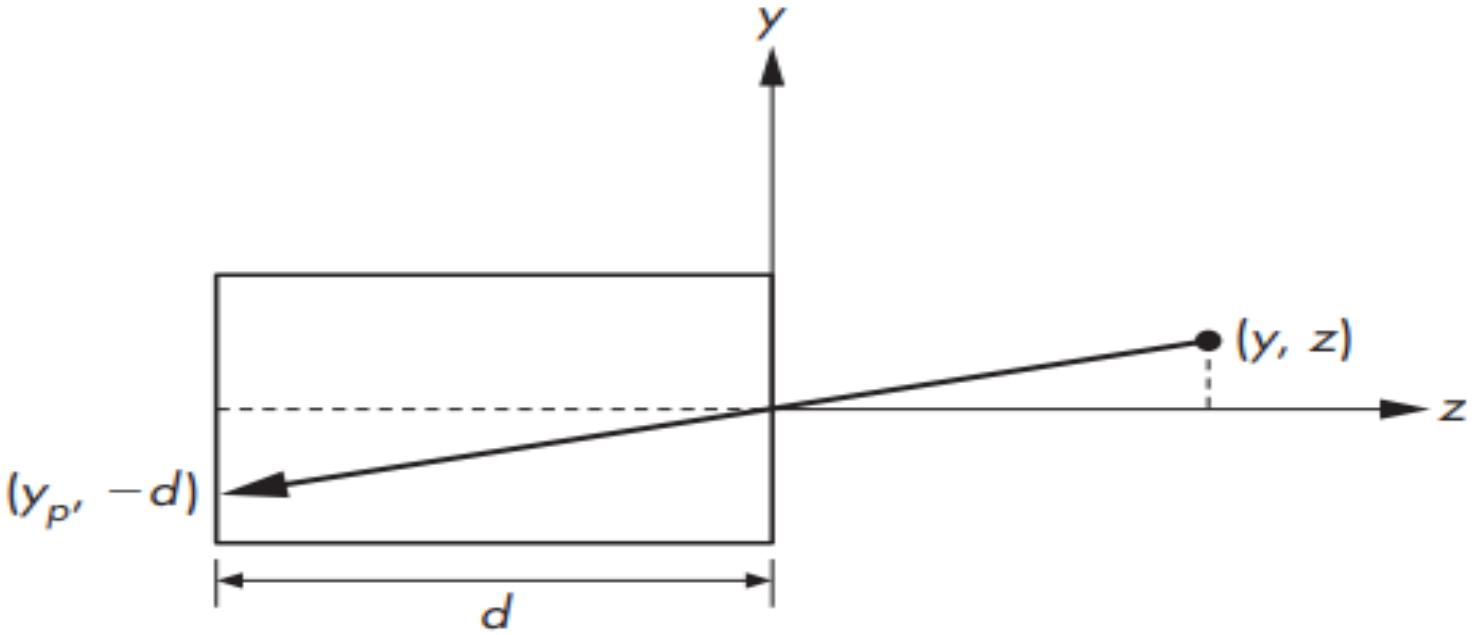
Side view of pinhole camera.



The Uni

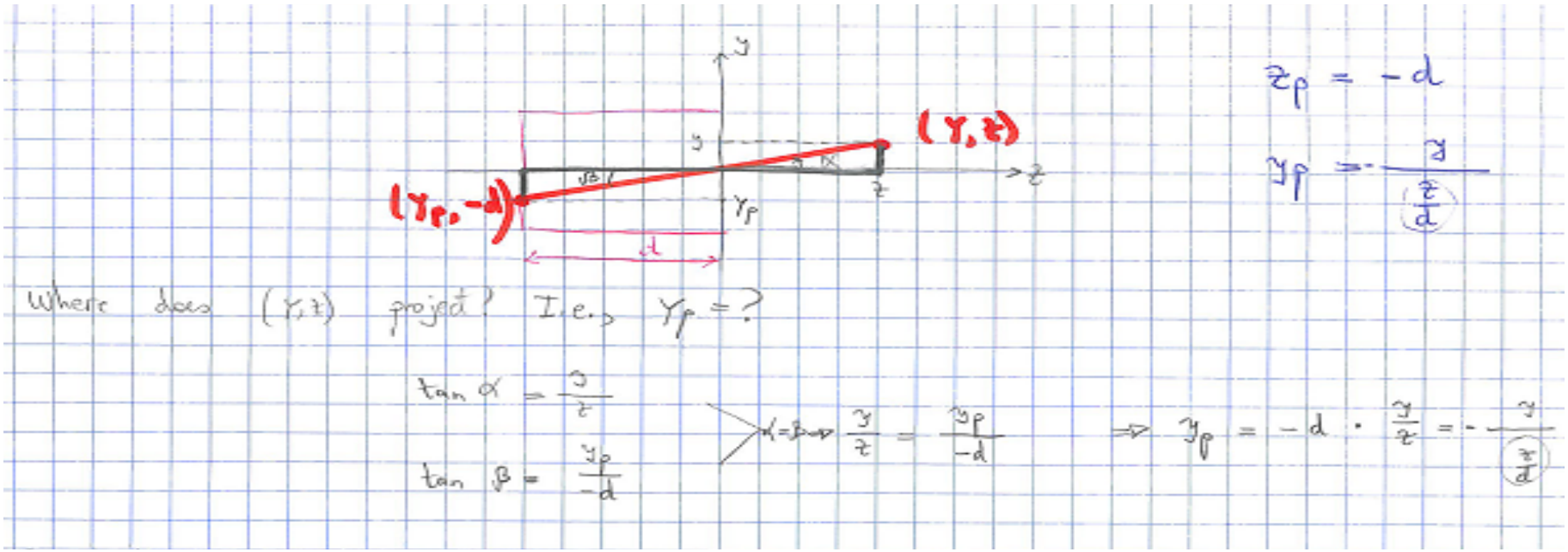
Pinhole Camera

These are equations of simple perspective



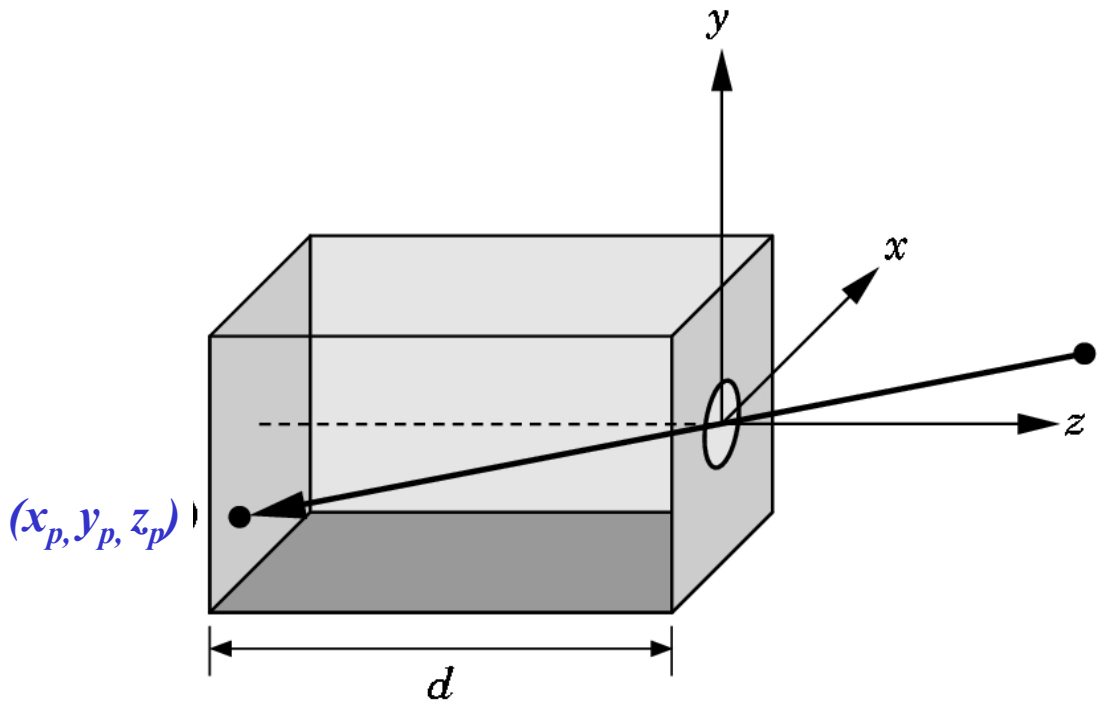
What is y_p?

$$y_p = -\frac{y}{z/d}.$$



Pinhole Camera

These are equations of simple perspective



What is x_p? (top view)

$$x_p = -\frac{x}{z/d}.$$

Similarly $y_p = -\frac{y}{z/d}$

Pinhole Camera

These are equations of simple perspective

What is θ/2?
(field/angle of view)

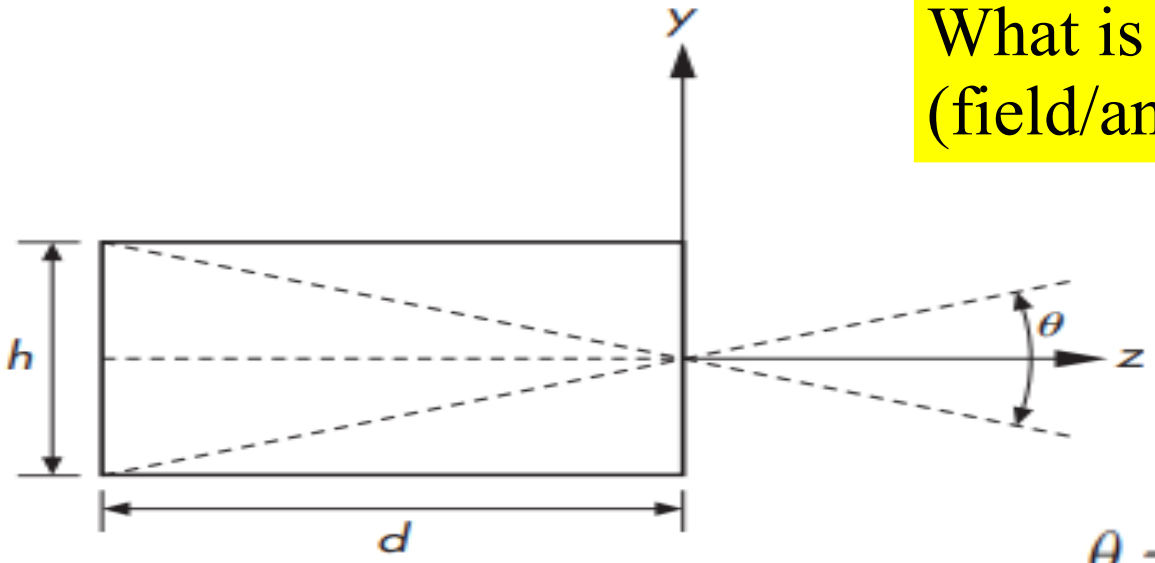
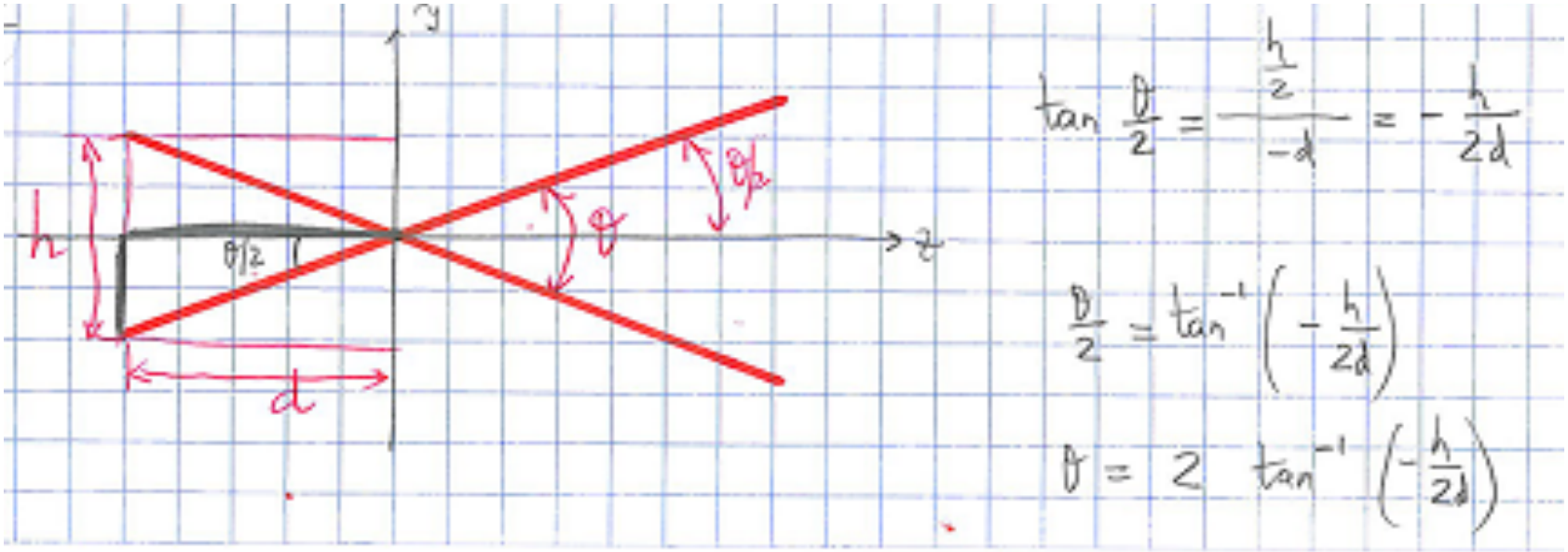
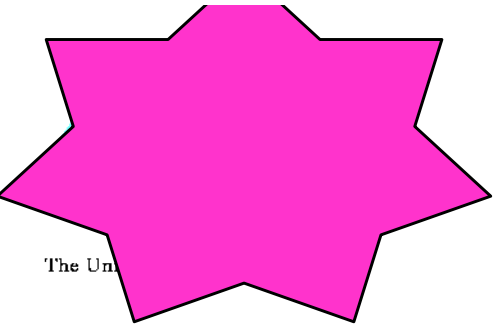


FIGURE 1.14 Angle of view.

$$\theta = 2\tan^{-1}\frac{h}{2d}.$$





LECTURE 1 CLASS PARTICIPATION

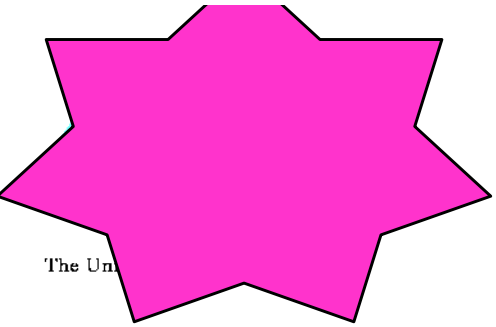
VH, publish LECTURE 1 CLASS PARTICIPATION

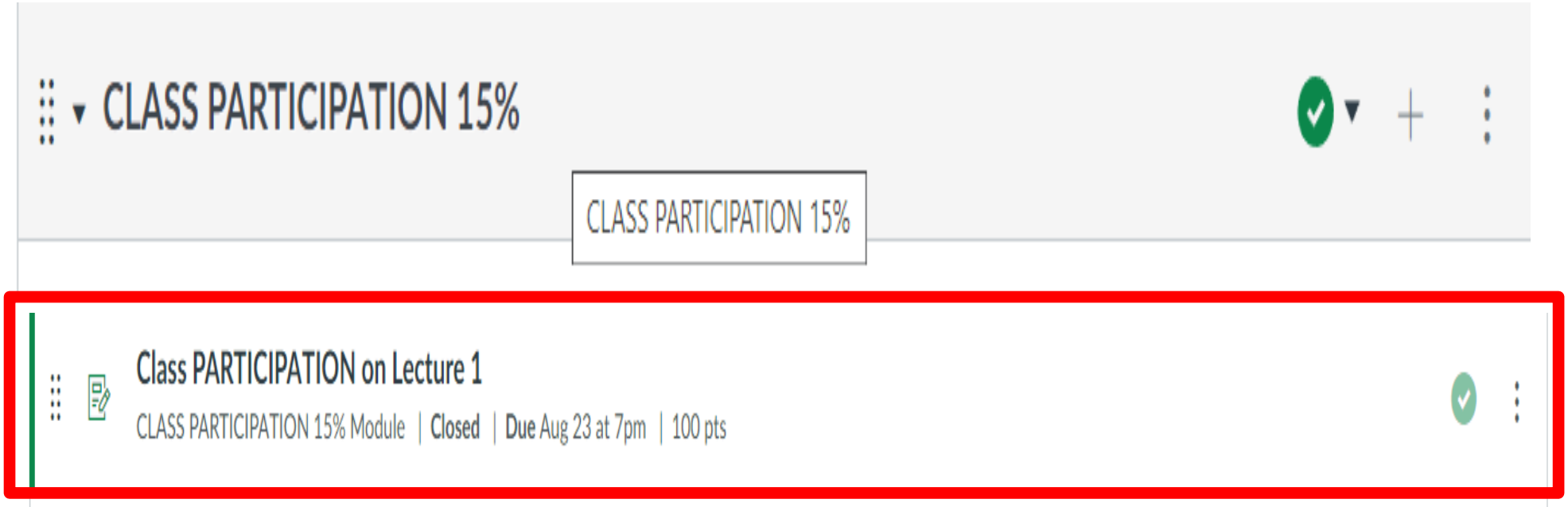
Class Participation on Lecture 1.doc

VH, publish Class Participation on Lecture 1

simple.c

VH, publish simple.c





VH, publish Assignment Class PARTICIPATION on Lecture 1

Download Class PARTICIPATION on Lecture 1.doc

The Un

Class PARTICIPATION on Lecture 1

Published | Edit

Download and complete this word document.

Class Participation on Lecture 1.doc

simple.c

You will be prompted when to Upload completed document to CANVAS as **score.doc** (example 100.doc).

Warning:

TA, at random, will inspect the Uploaded document.

If you score is not honestly entered you will get a zero.

VH, publish Class Participation on Lecture 1

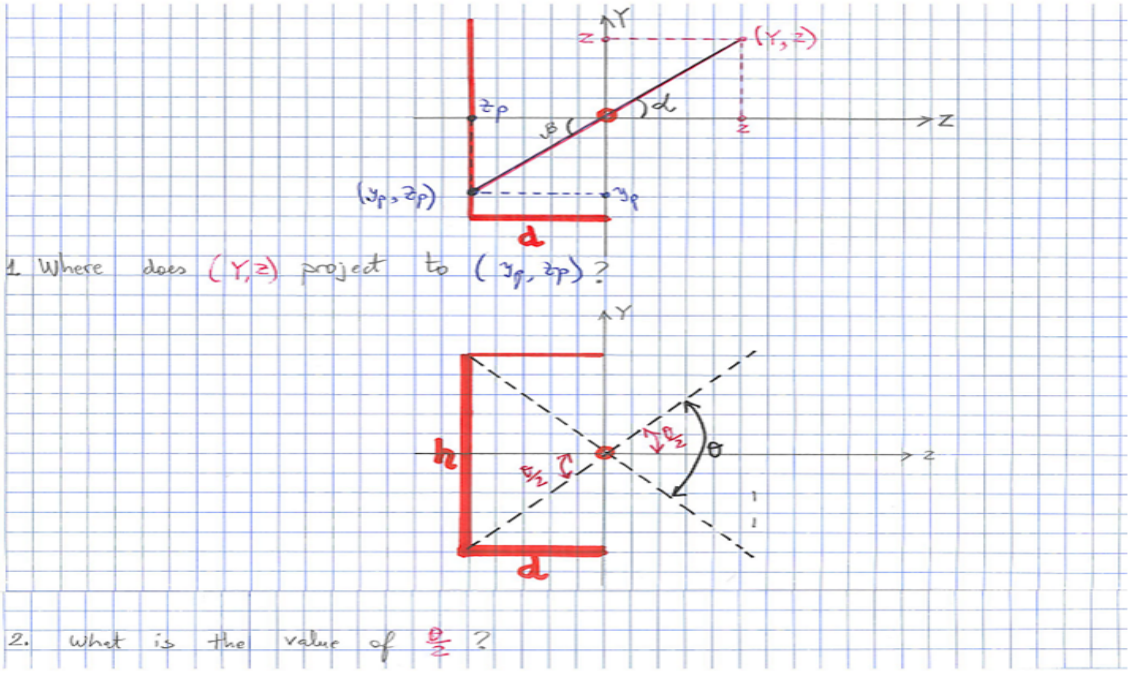
The Un

Class PARTICIPATION on Lecture 1.doc ANSWER SHEET

(Out of 100 points. Please record your own total score!)

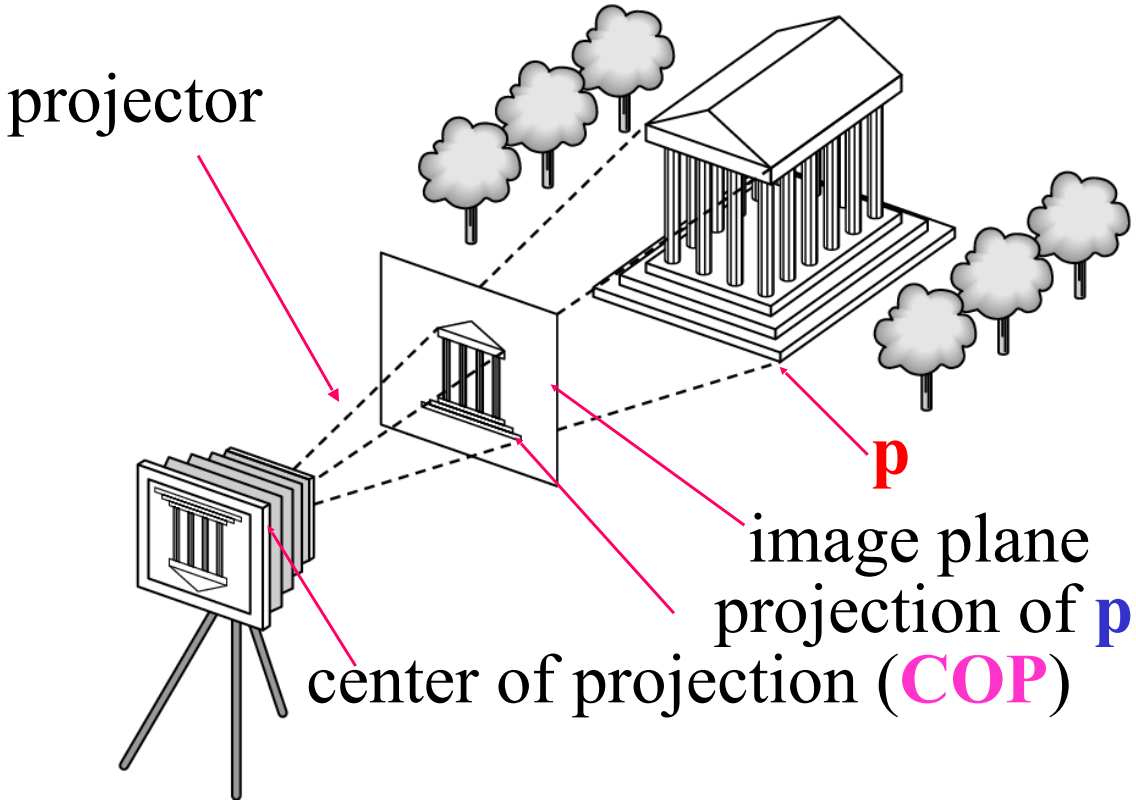
(Attach as score.doc!)

1. z_p? (10 points)
   y_p? (10 points)
   x_p? (10 points)
   $\theta/2$? (10 points)

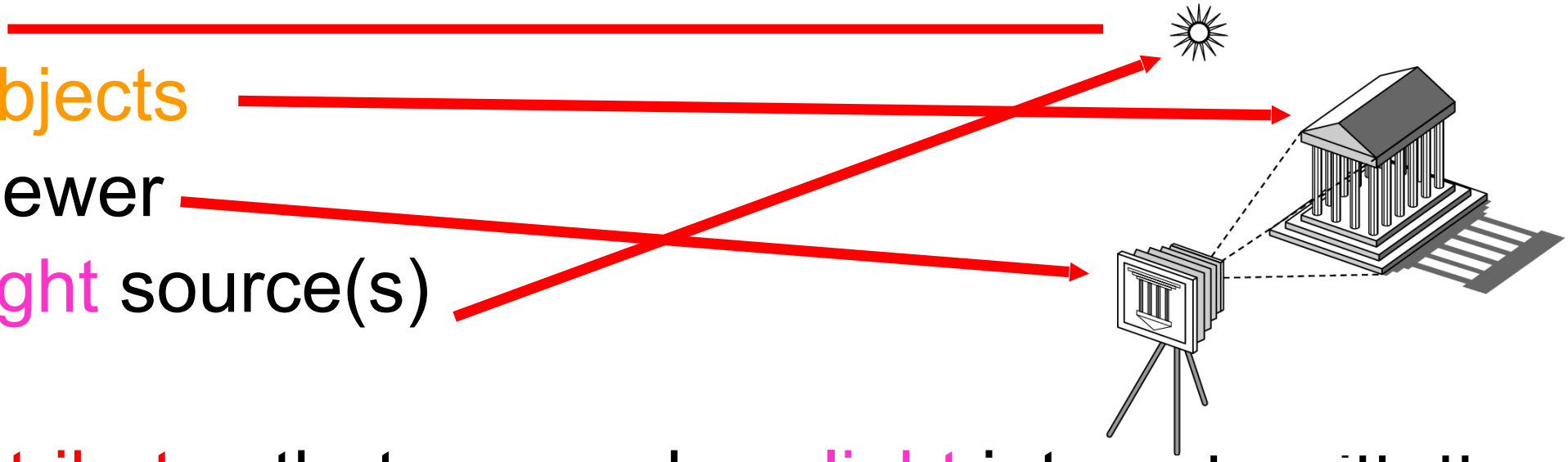


Class Participation 1!

Image formation with the Synthetic Camera Model



Elements of Image Formation



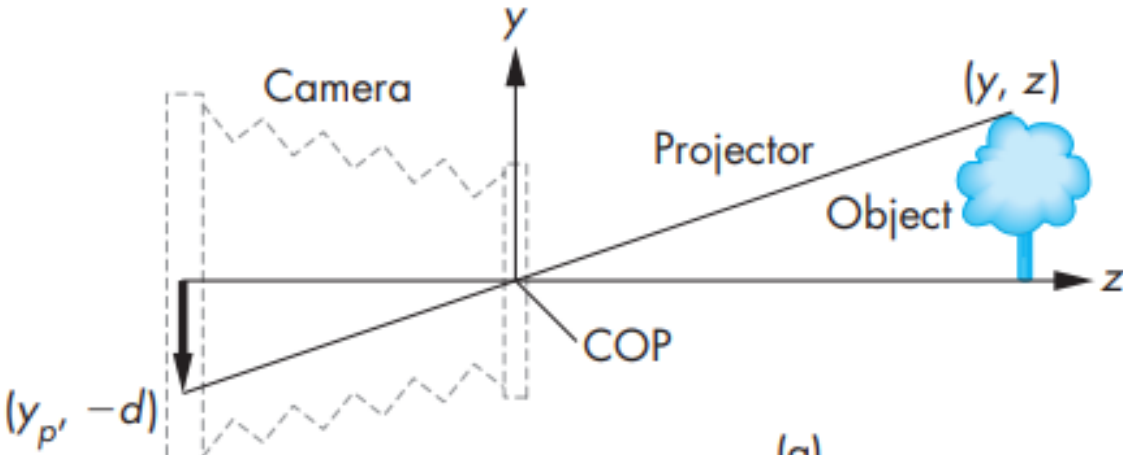
- Objects
- Viewer
- Light source(s)

- Attributes that govern how light interacts with the materials in the scene

- Note the independence of the objects, the viewer, and the light source(s)

The University of New Mexico

Equivalent Views of Image Formation



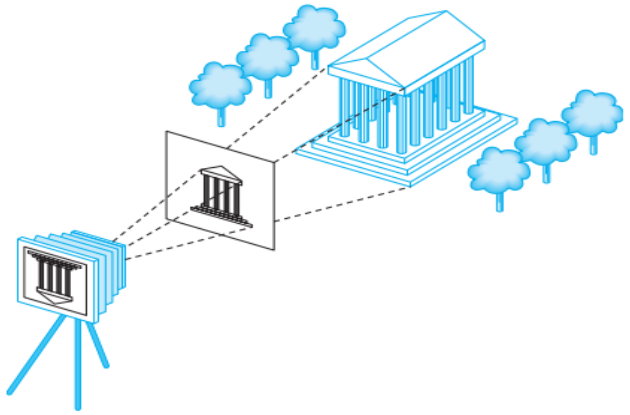


FIGURE 1.18 Imaging with the synthetic camera.

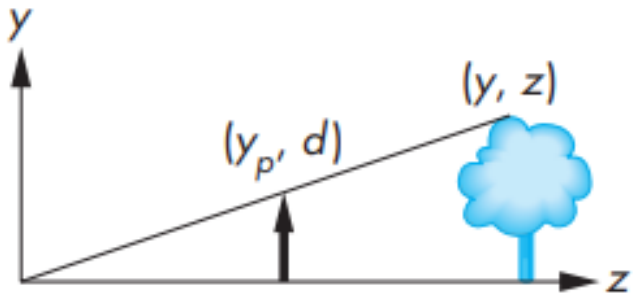
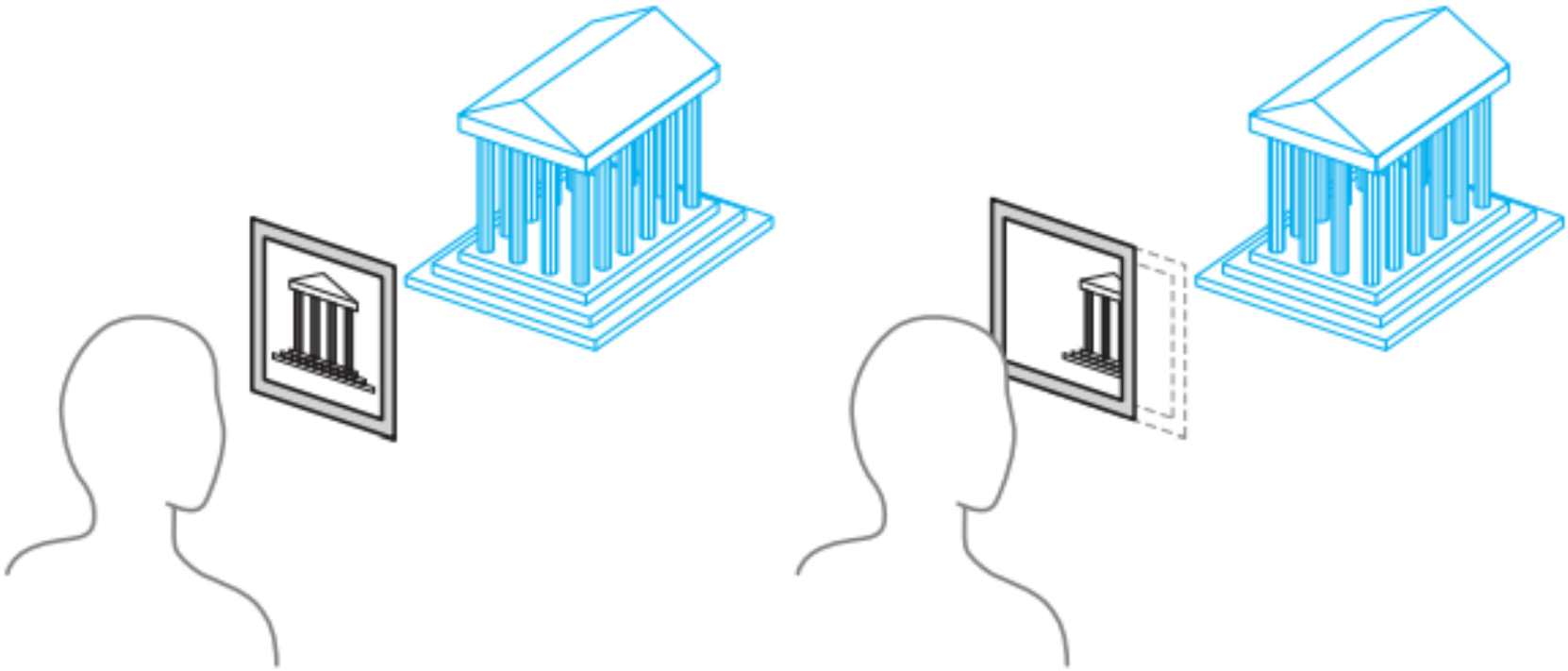


FIGURE 1.17 Equivalent views of image formation. (a) Image formed on the back of the camera. (b) Image plane moved in front of the camera.

Clipping Window/Rectangle



(a) (b)

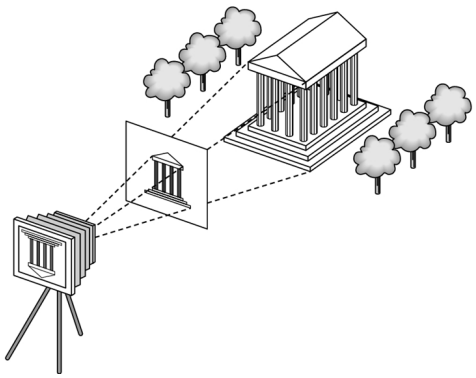
FIGURE 1.19 Clipping. (a) Window in initial position. (b) Window shifted.

Advantages

- Separation of objects, viewer, light sources

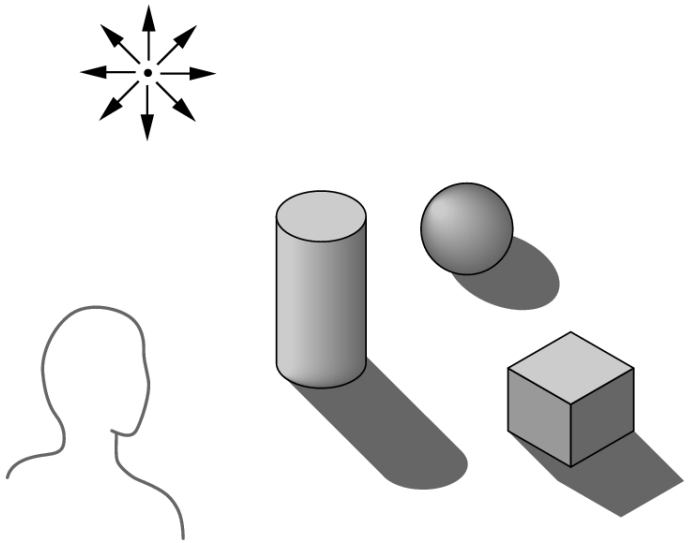
- Leads to simple software **API**
  - Specify objects, lights, camera, attributes
  - Let implementation determine image

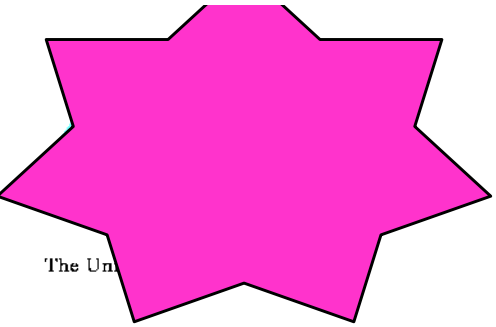
- Leads to fast hardware implementation



Global vs Local Lighting

- Cannot compute color or shade of each object independently
  - Some objects are blocked from light
  - Light can reflect from object to object
  - Some objects might be translucent





Class PARTICIPATION on Lecture 1

Published | Edit

Download and complete this word document.

Class Participation on Lecture 1.doc

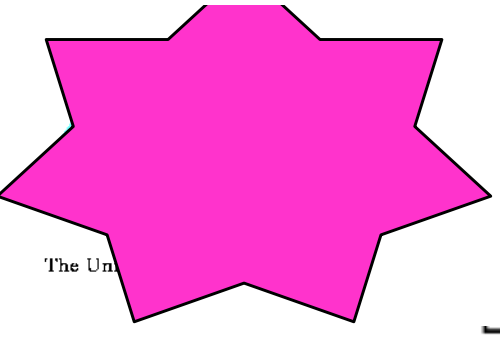
simple.c

You will be prompted when to Upload completed document to CANVAS as **score.doc** (example 100.doc).

Warning:

TA, at random, will inspect the Uploaded document.

If you score is not honestly entered you will get a zero.



OpenGL

Class PARTICIPATION on Lecture 1.doc **ANSWER SHEET**

(Out of 100 points. Please record your own total score!)

(Attach as score.doc!)

2. Download the simple.c from CANVAS, build a Lecture1 C++ Empty Project and add to Lecture1 Project.

a. **Build and run the project.**
(Take print screen under 2. a) (30 points)

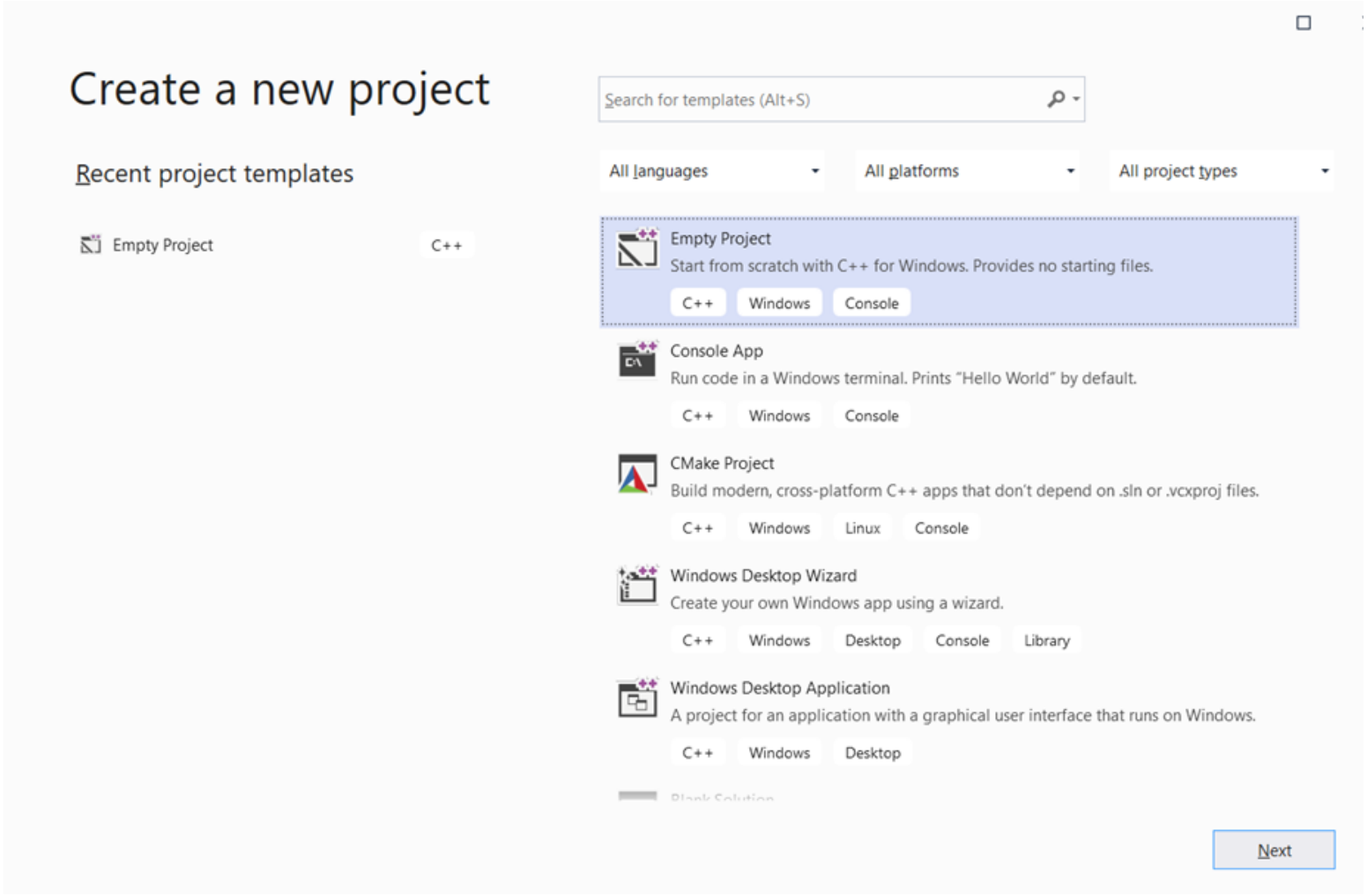
b. **Reposition the triangle as specified in the class. Rebuild and run the project.**
(Take print screen under 2. b) (30 points)

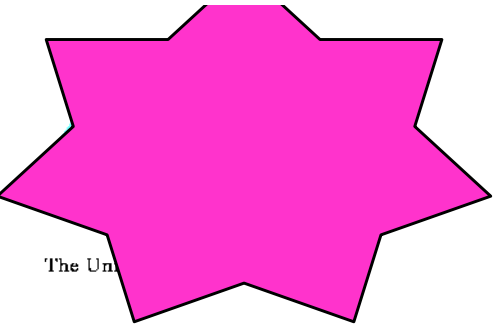
Class Participation 2!

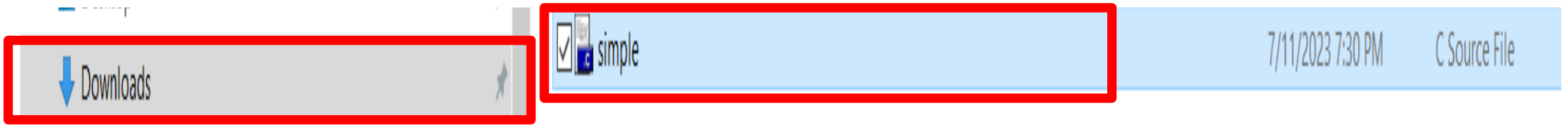
Take print screen of the desktop and insert here!

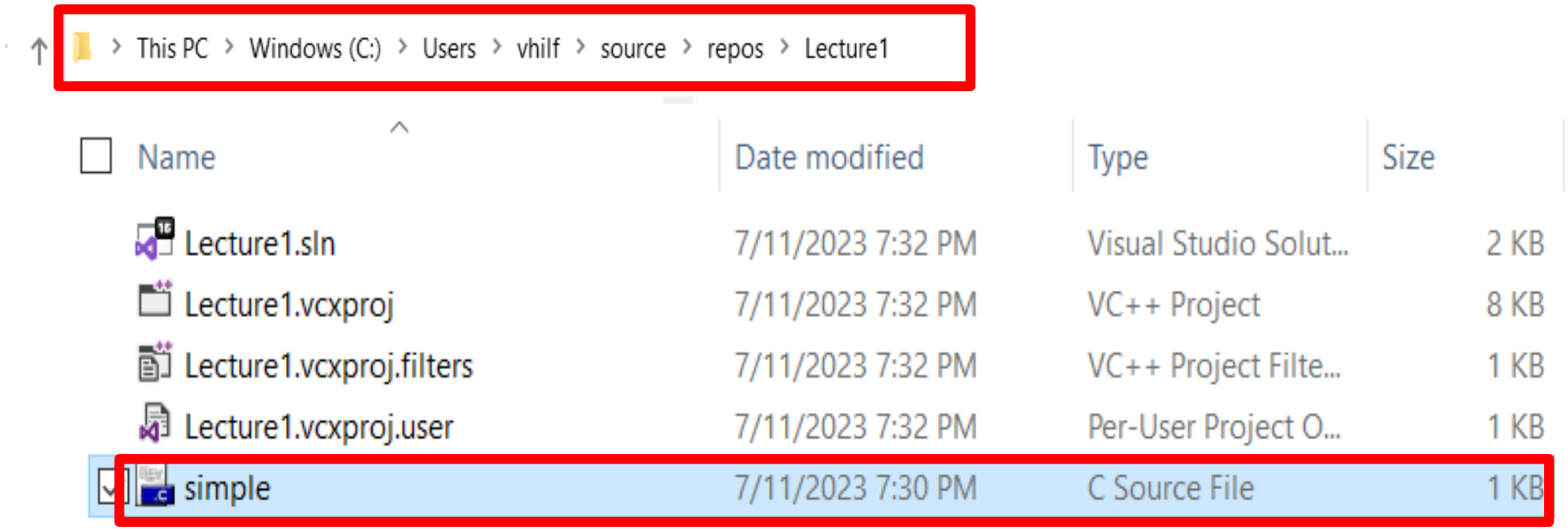
2.

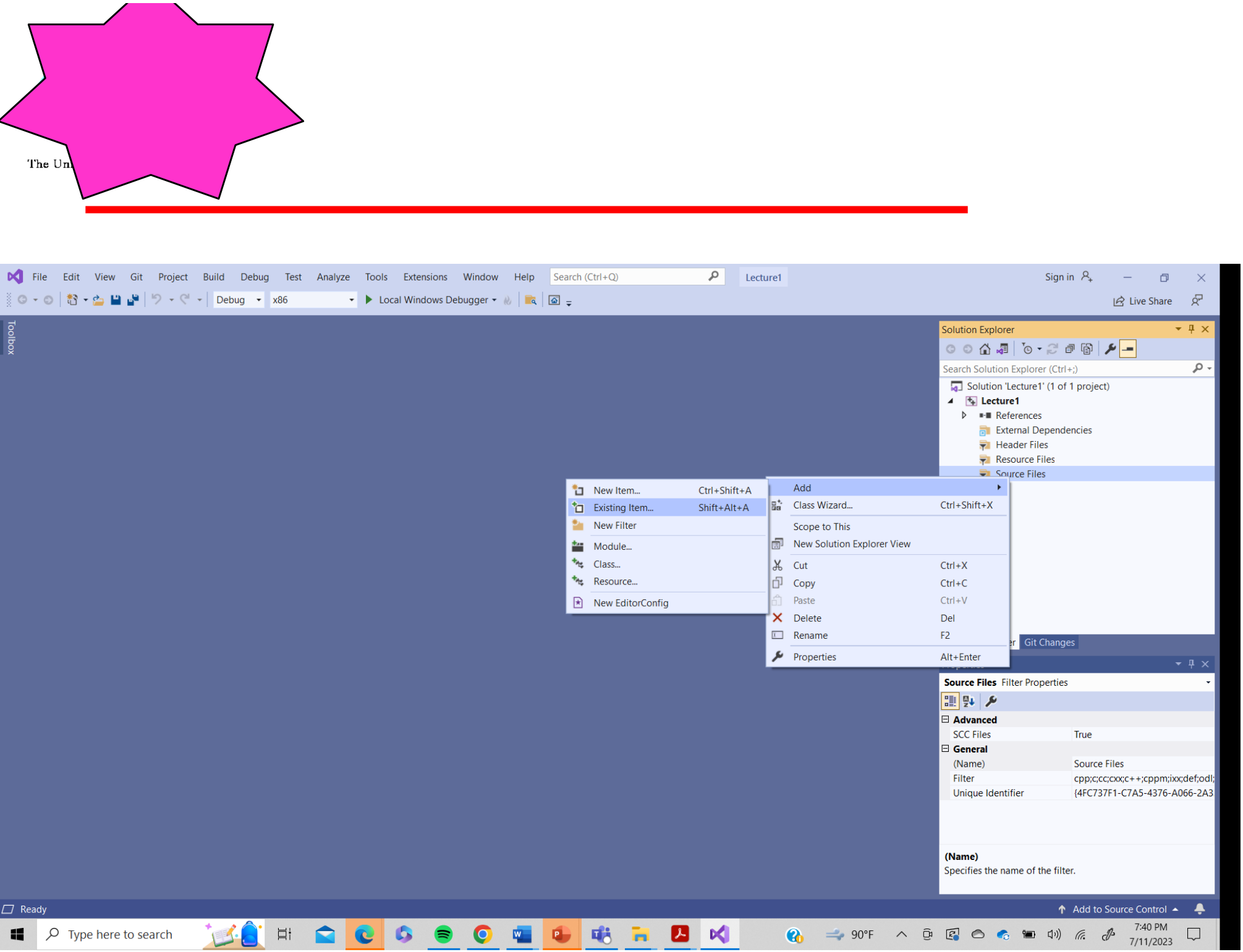
Create **Lecture1** Empty Project:

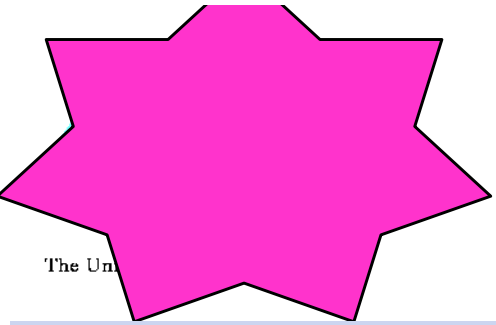





Downloads
simple
7/11/2023 7:30 PM
C Source File


This PC
Windows (C:)
Users
vhilf
source
repos
Lecture1
Name
Date modified
Type
Size
Lecture1.sln
7/11/2023 7:32 PM
Visual Studio Solut...
2 KB
Lecture1.vcxproj
7/11/2023 7:32 PM
VC++ Project
8 KB
Lecture1.vcxproj.filters
7/11/2023 7:32 PM
VC++ Project Filte...
1 KB
Lecture1.vcxproj.user
7/11/2023 7:32 PM
Per-User Project O...
1 KB
simple
7/11/2023 7:30 PM
C Source File
1 KB


File Edit View Git Project Build Debug Test Analyze Tools Extensions Window Help
Search (Ctrl+Q)
Lecture1
Sign in
Debug
x86
Local Windows Debugger
Live Share
Toolbox
Solution Explorer
Search Solution Explorer (Ctrl+;)
Solution 'Lecture1' (1 of 1 project)
Lecture1
References
External Dependencies
Header Files
Resource Files
Source Files
New Item... Ctrl+Shift+A
Existing Item... Shift+Alt+A
New Filter
Module...
Class...
Resource...
New EditorConfig
Add
Class Wizard... Ctrl+Shift+X
Scope to This
New Solution Explorer View
Cut Ctrl+X
Copy Ctrl+C
Paste Ctrl+V
Delete Del
Rename F2
Properties Alt+Enter
Git Changes
Source Files Filter Properties
Advanced
SCC Files True
General
(Name) Source Files
Filter cpp;c;cc;cxx;c++;cppm;ixx;def;odl;
Unique Identifier {4FC737F1-C7A5-4376-A066-2A3
(Name)
Specifies the name of the filter.
Ready
Add to Source Control
Type here to search
90°F
7:40 PM
7/11/2023



The Un

File Edit View Git Project Build Debug Test Analyze Tools Extensions Window Help Search (Ctrl+Q) Lecture1 Sign in

Live Share

Enter your Last Name, First Name!

Server Explorer Toolbox

simple.c

Lecture1 (Global Scope)

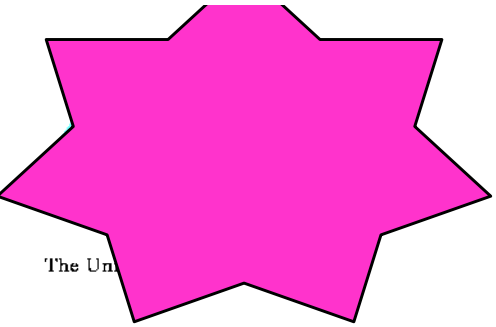
```
1   // Last Name, First Name
2   // Lecture1
3   // Fall 2023
4
5   /* simple.c  This program draws a white rectangle on a black background. */
6
7   #include <GL/glut.h>        /* glut.h includes gl.h and glu.h*/
8
9   void display(void)
10  {
11      glClear(GL_COLOR_BUFFER_BIT);
12
13      /* default viewing volume is a cube of side 2 centered at the origin*/
14      /* thus, this square is centered at the origin*/
15
16     glBegin(GL_TRIANGLES);
17         glVertex2f(0.0, 0.0);
18         glVertex2f(1.0, 1.0);
19         glVertex2f(1.0, 0.0);
20     glEnd();
21
22     glFlush();
23  }
24
25  int main(int argc, char** argv)
26  {
27     glutInit(&argc,argv);
28     glutInitDisplayMode (GLUT_SINGLE | GLUT_RGB);
29     glutInitWindowSize(500,500);
30     glutInitWindowPosition(0,0);
31     glutCreateWindow("simple");
32     glutDisplayFunc(display);
33     glutMainLoop();
34  }
```

Solution Explorer

Search Solution Explorer (Ctrl+;)

Solution 'Lecture1' (1 of 1 project)

- Lecture1
  - References
  - External Dependencies
  - Header Files
  - Resource Files
  - Source Files
    - simple.c

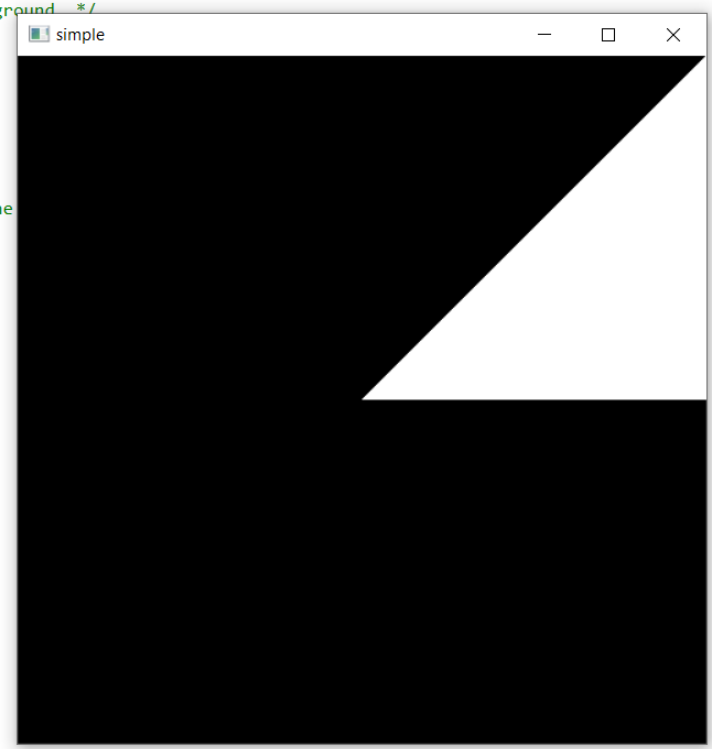


Run

simple.c

Lecture1 | (Global Scope)

```
1   // Last Name, First Name
2   // Lecture1
3   // Fall 2023
4
5   /* simple.c  This program draws a white rectangle on a black background  */
6
7   #include <GL/glut.h>        /* glut.h includes gl.h and glu.h*/
8
9   void display(void)
10  {
11      glClear(GL_COLOR_BUFFER_BIT);
12
13       /* default viewing volume is a cube of side 2 centered at the
14       /* thus, this square is centered at the origin*/
15
16     glBegin(GL_TRIANGLES);
17         glVertex2f(0.0, 0.0);
18         glVertex2f(1.0, 1.0);
19         glVertex2f(1.0, 0.0);
20      glEnd();
21
22      glFlush();
23  }
24
25  int main(int argc, char** argv)
26  {
27      glutInit(&argc,argv);
28      glutInitDisplayMode (GLUT_SINGLE | GLUT_RGB);
29      glutInitWindowSize(500,500);
30      glutInitWindowPosition(0,0);
31      glutCreateWindow("simple");
32      glutDisplayFunc(display);
33      glutMainLoop();
34  }
35
```



106 % | No issues found | Ln: 1 | Ch: 1 | MIXED | CRLF

Output

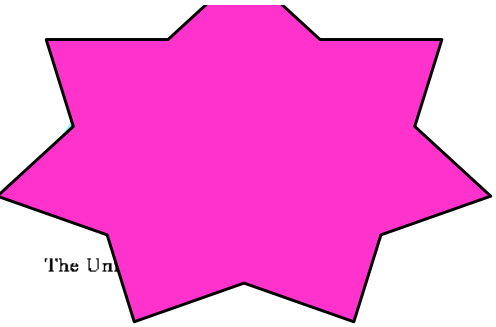
Solution Explorer

Search Solution Explorer (Ctrl+;)

Solution 'Lecture1' (1 of 1 project)
- Lecture1
  - References
  - External Dependencies
  - Header Files
  - Resource Files
  - Source Files
    - simple.c

Solution Explorer | Git Changes

Build succeeded | Add to Source Control



Places triangle with corner at the origin (0.0,0.0)

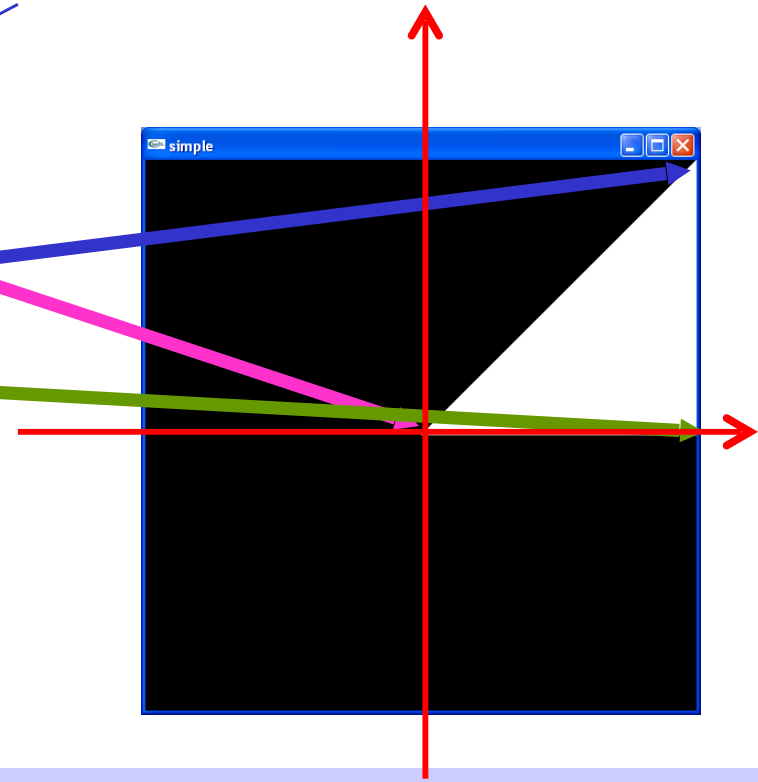
Example

type of object

location of vertex

```
glBegin(GL_TRIANGLES);
 glVertex2f(0.0, 0.0);
 glVertex2f(1.0, 1.0);
 glVertex2f(1.0, 0.0);
glEnd( );
```

end of object definition

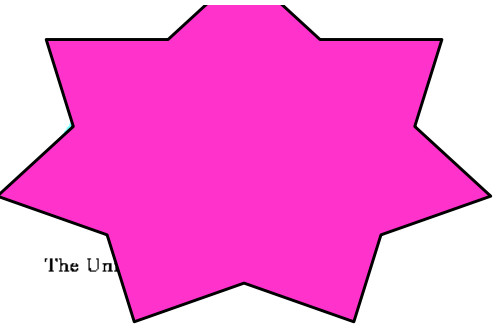


2. Download the simple.c from CANVAS, build a Lecture1 C++ Empty Project and add to Lecture1 Project.

a. Build and run the project.
(Take print screen under 2. a) (30 points)

simple.c
Lecture 1 Class Participation 2

Take print screen of the desktop and insert here!



Reposition triangle as shown below

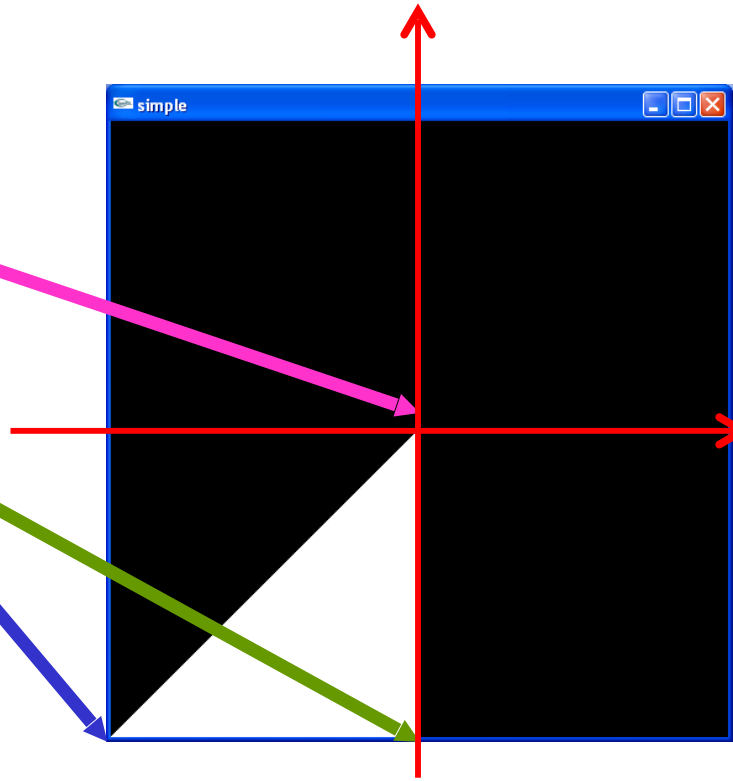
Got it?

type of object

location of vertex

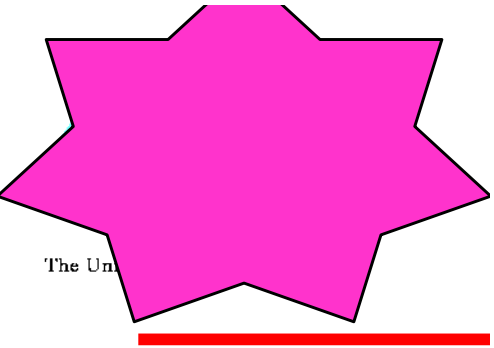
```
glBegin(GL_TRIANGLES);
   glVertex2f(0.0 ,   0.0);
   glVertex2f(-1.0, -1.0);
   glVertex2f(0.0 , -1.0);
glEnd( );
```

end of object definition



b. Reposition the triangle as specified in the class. Rebuild and run the project.
(Take print screen under 2. b) (30 points)

Take print screen of the desktop and insert here!



You will be prompted when to Upload completed document to CANVAS as **score.doc** (example 100.doc).

Warning:

TA, at random, will inspect the Uploaded document.

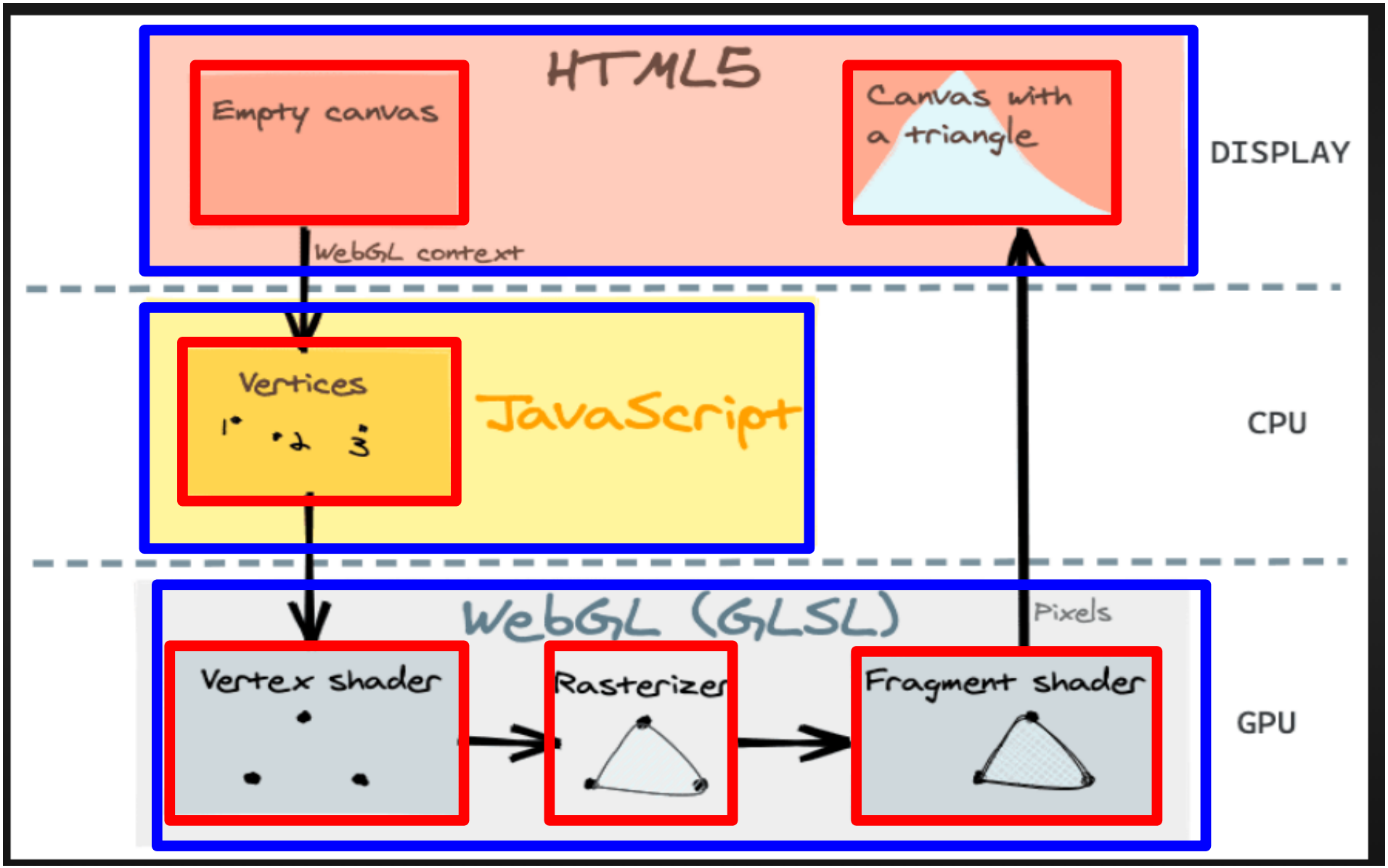
If you score is not honestly entered you will get a zero.

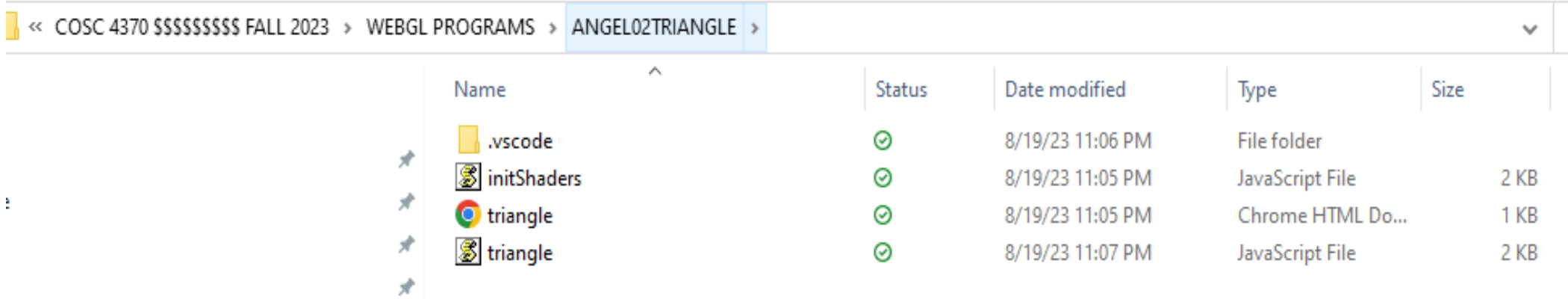
Please rename document to score.doc (example 100.doc)
Warning: if you score is not honestly honest you will get a zero.

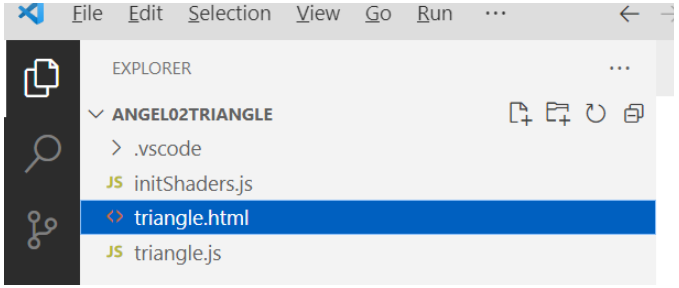
Class PARTICIPATION on Lecture 1.doc **ANSWER SHEET**
(Out of 100 points. Please record your own total score!)
(Attach as score.doc!)

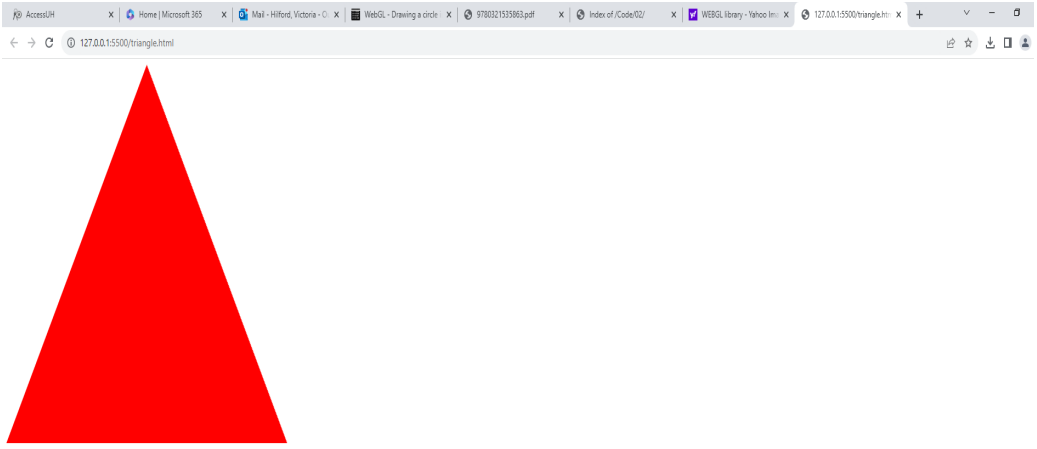
WebGL

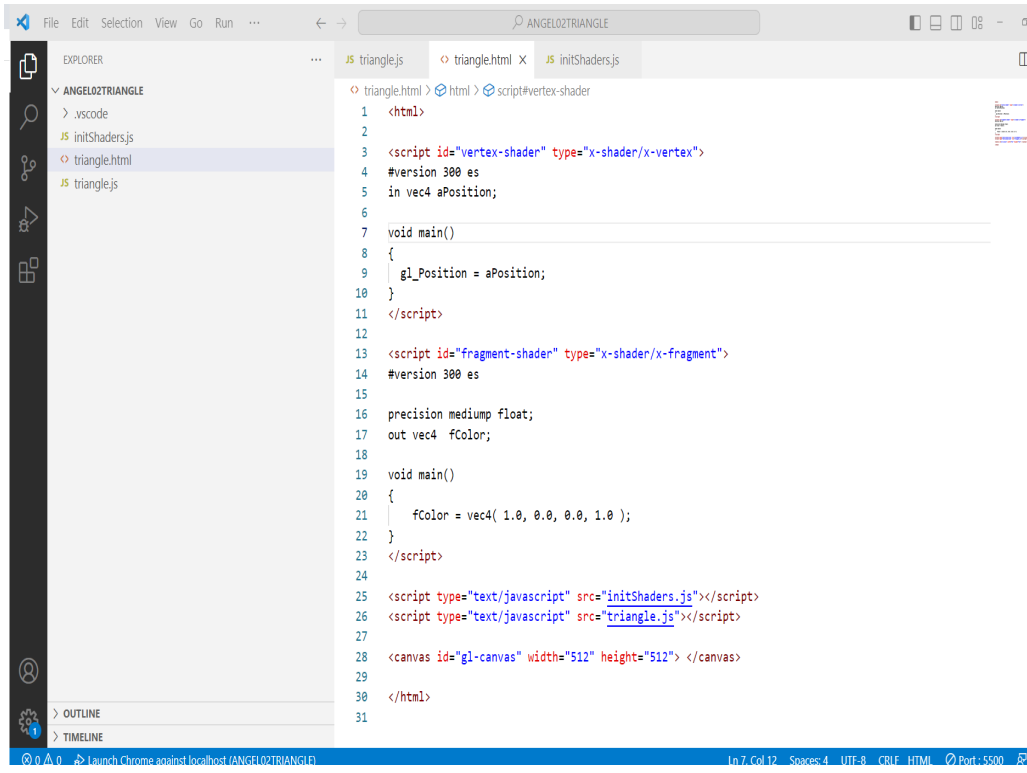
The University of New Mexico




« COSC 4370 $$$$$$$$$ FALL 2023 › WEBGL PROGRAMS › ANGEL02TRIANGLE ›
Name
Status
Date modified
Type
Size
.vscode
8/19/23 11:06 PM
File folder
initShaders
8/19/23 11:05 PM
JavaScript File
2 KB
triangle
8/19/23 11:05 PM
Chrome HTML Do...
1 KB
triangle
8/19/23 11:07 PM
JavaScript File
2 KB


File Edit Selection View Go Run ···
EXPLORER
ANGEL02TRIANGLE
.vscode
initShaders.js
triangle.html
triangle.js


127.0.0.1:5500/triangle.html


File Edit Selection View Go Run ···
ANGEL02TRIANGLE
EXPLORER
triangle.js
triangle.html
initShaders.js
ANGEL02TRIANGLE
.vscode
initShaders.js
triangle.html
triangle.js
triangle.html > html > script#vertex-shader
1 <html>
3 <script id="vertex-shader" type="x-shader/x-vertex">
4 #version 300 es
5 in vec4 aPosition;
7 void main()
8 {
9 gl_Position = aPosition;
10 }
11 </script>
13 <script id="fragment-shader" type="x-shader/x-fragment">
14 #version 300 es
16 precision mediump float;
17 out vec4 fColor;
19 void main()
20 {
21 fColor = vec4(1.0, 0.0, 0.0, 1.0);
22 }
23 </script>
25 <script type="text/javascript" src="initShaders.js"></script>
26 <script type="text/javascript" src="triangle.js"></script>
28 <canvas id="gl-canvas" width="512" height="512"> </canvas>
30 </html>
OUTLINE
TIMELINE
Launch Chrome against localhost (ANGEL02TRIANGLE)
Ln 7, Col 12 Spaces: 4 UTF-8 CRLF HTML Port : 5500

WebGL

Clear the Window

- `glClear(GL_COLOR_BUFFER_BIT)`
  - clears the frame buffer by overwriting it with the background color.
  - Background color is a state set by `glClearColor(GLfloat r, GLfloat g, GLfloat b, GLfloat a)` in the `init()`.

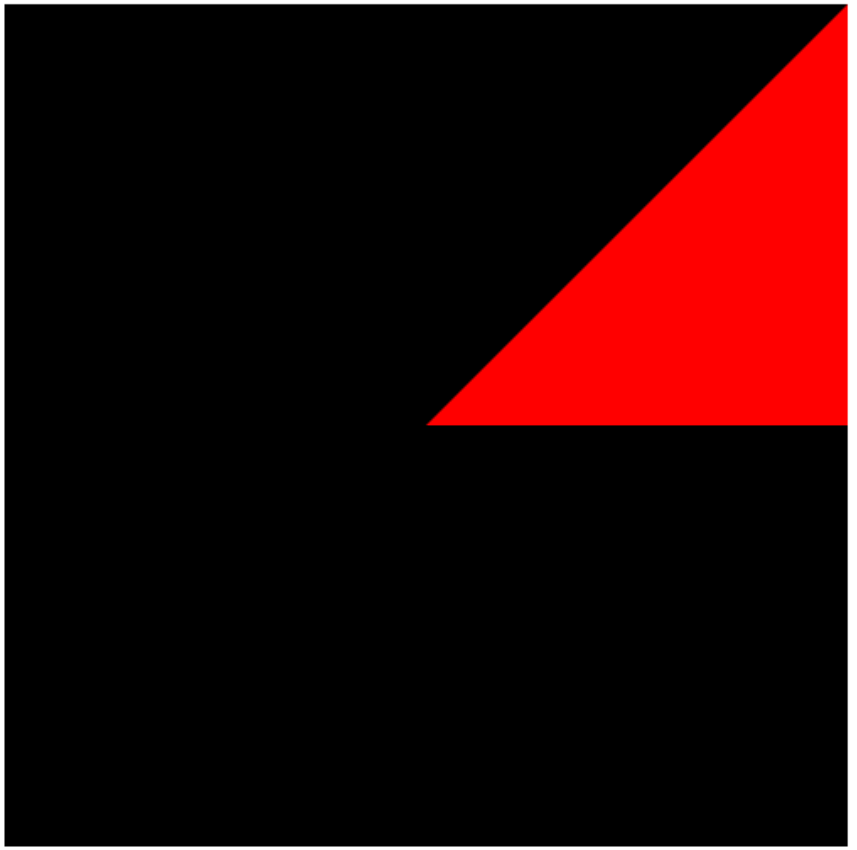
simple.c

Lecture1

```
1   //
2   //
3   //
4
5   /*                                          black background  */
6
7   #ir                                         ıd glu.h*/
8
9   void display(void)
10  {
11      glClear(GL_COLOR_BUFFER_BIT);
12
13      /* default viewing volume is
14      /* thus, this square is cent
15
16      glBegin(GL_TRIANGLES);
17          glVertex2f(0.0, 0.0);
18          glVertex2f(1.0, 1.0);
19          glVertex2f(1.0, 0.0);
20      glEnd();
21
22      glFlush();
23  }
24
25  int main(int argc, char** argv)
26  {
27      glutInit(&argc,argv);
28      glutInitDisplayMode (GLUT_SING
29      glutInitWindowSize(500,500);
30      glutInitWindowPosition(0,0);
31      glutCreateWindow("simple");
32      glutDisplayFunc(display);
33      glutMainLoop();
34  }
35
```

```
15      // First, initialize the  three points
16
17      points = new Float32Array([
18                  -1, -1,
19                  0,  1,
20                  1, -1
21              ]);
22
23      //  Configure WebGL
24      gl.viewport(  0, 0, canvas.width, canva
25      gl.clearColor(1.0, 1.0, 1.0, 1.0);
26
27      //  Load shaders and initialize attrib
28      var program = initShaders(gl, "vertex
29      gl.useProgram(program);
30
31      // Load the data into the GPU
32      var bufferId = gl.createBuffer();
33      gl.bindBuffer(gl.ARRAY_BUFFER, buffer
34      gl.bufferData(gl.ARRAY_BUFFER, points,
35
36      // Associate out shader variables with
37      var aPosition = gl.getAttribLocation(p
38      gl.vertexAttribPointer(aPosition, 2, g
39      gl.enableVertexAttribArray(aPosition );
40
41      render();
42  };
43
44  function render() {···
47  }
```

127.0.0.1:5500/triangle.html



WebGL

The Un

simple.c

Lecture1

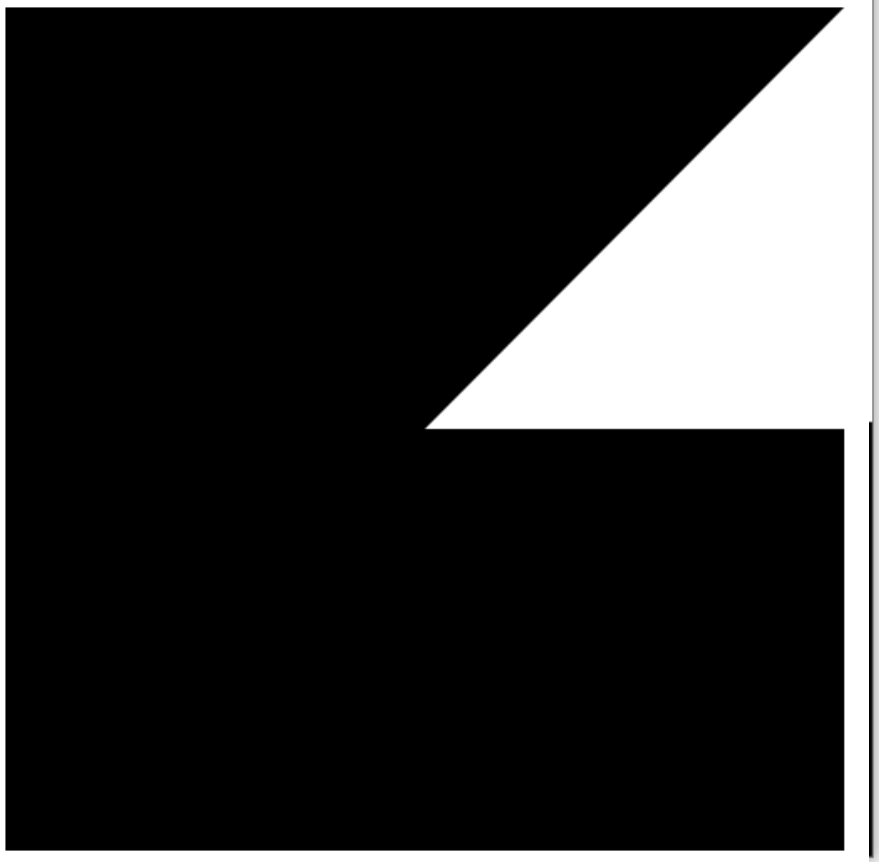
(Global Scope)

```
<html>

<script id="vertex-shader" type="x-shader/x-vertex">…
</script>

<script id="fragment-shader" type="x-shader/x-fragment">
#version 300 es

precision mediump float;
out vec4  fColor;

void main()
{
    fColor = vec4( 1.0, 0.0, 0.0, 1.0 );
}
</script>

<script type="text/javascript" src="initShaders.js"></script>
<script type="text/javascript" src="triangle.js"></script>

<canvas id="gl-canvas" width="500" height="500"> </canvas>

</html>
```

```
    glBegin(GL_TRIANGLES);
        glVertex2f(0.0, 0.0);
        glVertex2f(1.0, 1.0);
        glVertex2f(1.0, 0.0);
    glEnd();

    glFlush();
}

int main(int argc, char** argv)
{
    glutInit(&argc,argv);
    glutInitDisplayMode (GLUT_SING
    glutInitWindowSize(500,500);
    glutInitWindowPosition(0,0);
    glutCreateWindow("simple");
    glutDisplayFunc(display);
    glutMainLoop();
}
```

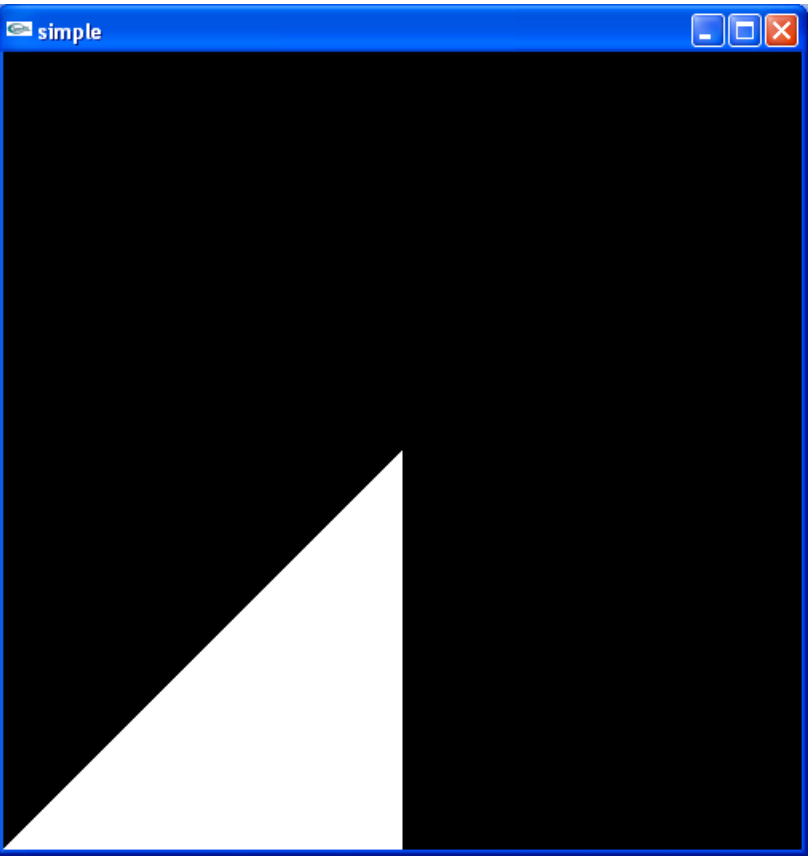
```
te rectangle on a black backg

h includes gl.h and glu.h*/

    // First, initialize the  three point
    points = new Float32Array([
                -1, -1,
                0,  1,
                1, -1
            ]);

    //  Configure WebGL
    gl.viewport( 0, 0, canvas.width, canv
    gl.clearColor(1.0, 1.0, 1.0, 1.0);

    //  Load shaders and initialize attri
    var program = initShaders(gl, "vertex
    gl.useProgram(program);

    // Load the data into the GPU
    var bufferId = gl.createBuffer();
    gl.bindBuffer(gl.ARRAY_BUFFER, buffer
    gl.bufferData(gl.ARRAY_BUFFER, points

    // Associate out shader variables wit
    var aPosition = gl.getAttribLocation(
    gl.vertexAttribPointer(aPosition, 2,
    gl.enableVertexAttribArray(aPosition

    render();
};

function render() {…
}
```

127.0.0.1:5500/triangle.html

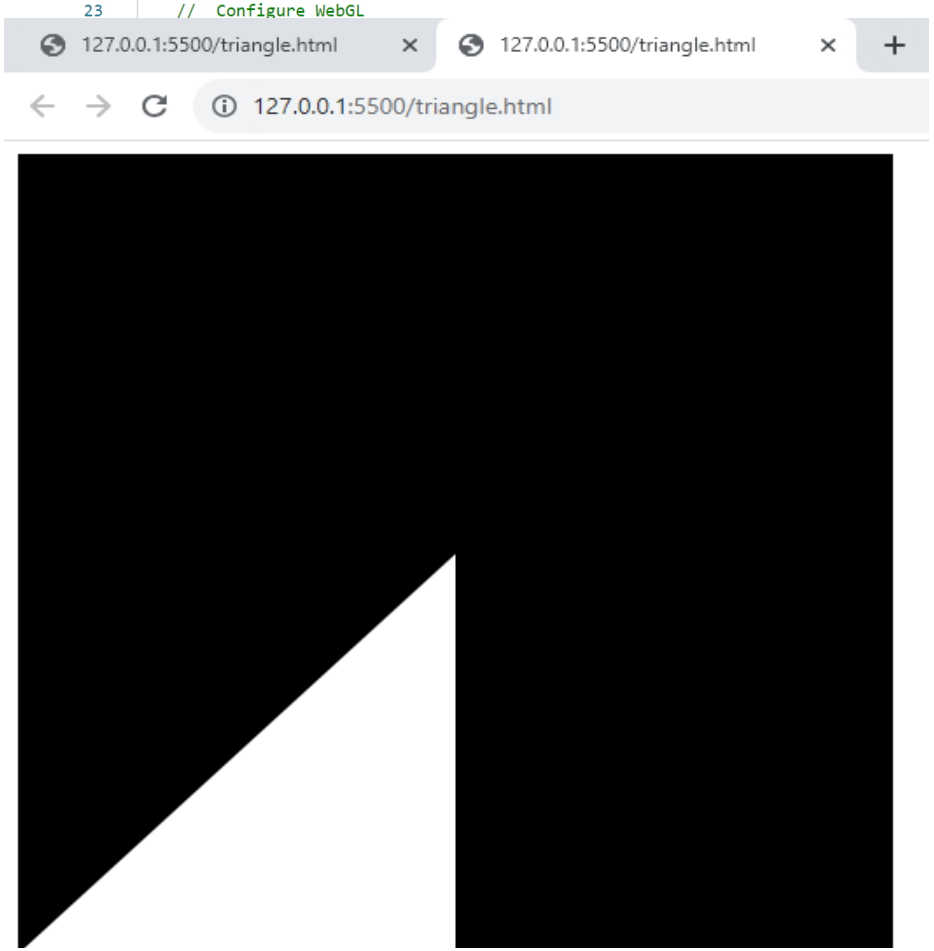


OpenGL / WebGL

```
glBegin(GL_TRIANGLES);
  glVertex2f(0.0 ,  0.0);
  glVertex2f(-1.0, -1.0);
  glVertex2f(0.0 , -1.0);
glEnd( );
```

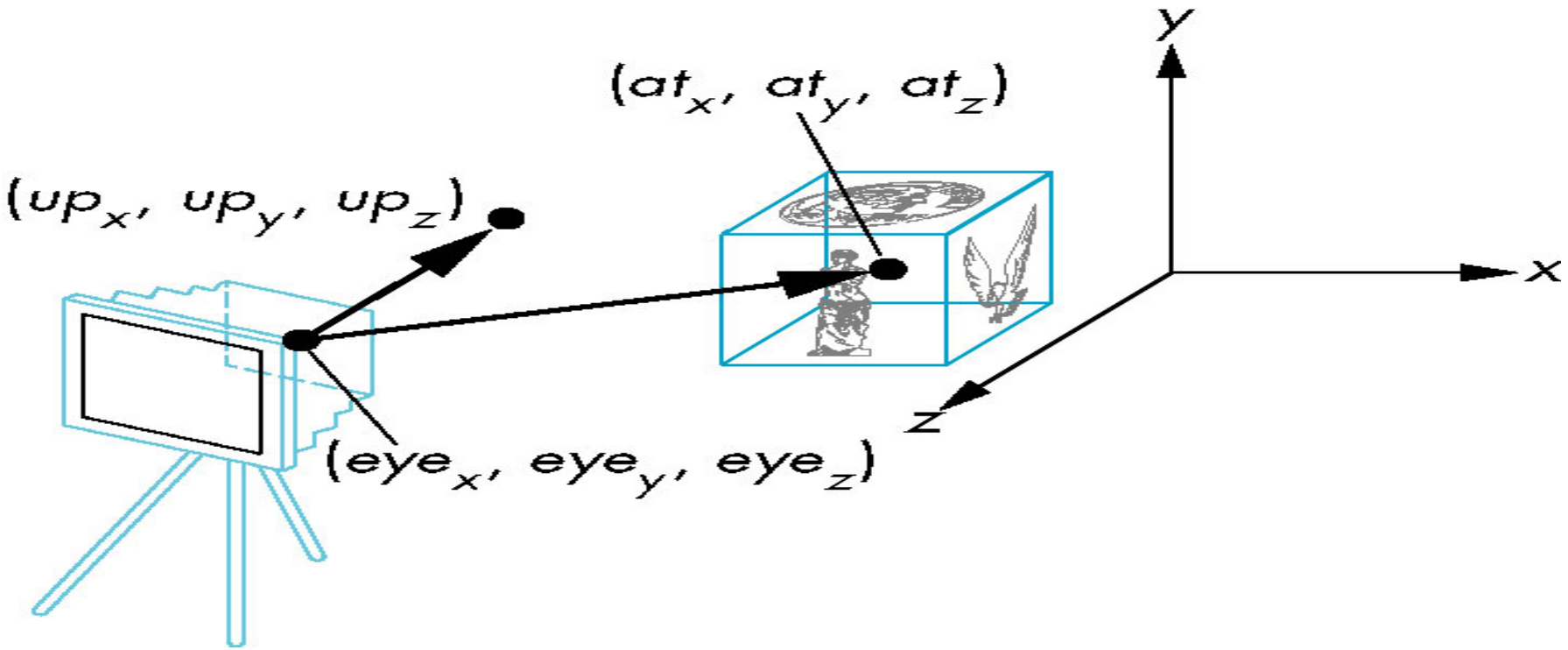



```
17      points = new Float32Array([
18          0, 0,
19         -1, -1,
20          0, -1
21      ]);
22
23      // Configure WebGL
```



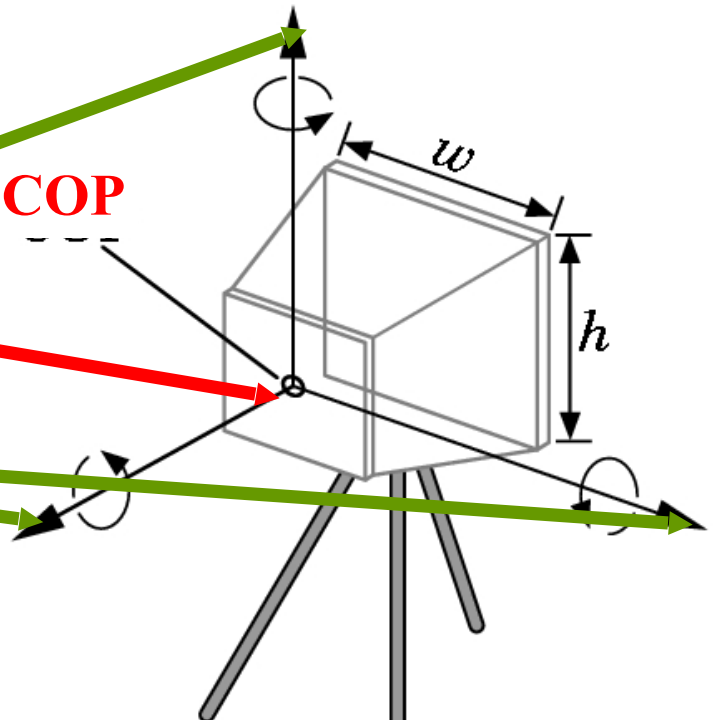
Camera & Objects

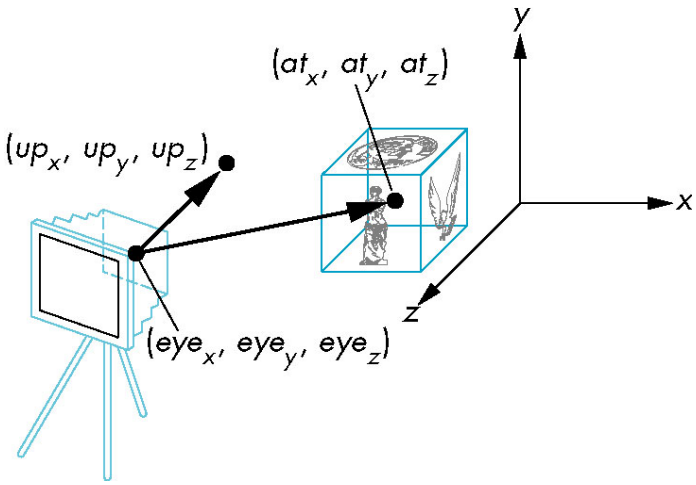
NONE of the **APIs** provide functions for directly specifying a desired relationship between the **camera** and an **object**.



Camera Specification

- Six degrees of freedom
  - Position of center of lens
  - Orientation
- Lens (focal length)
- Film size (width & height)
- Orientation of film plane

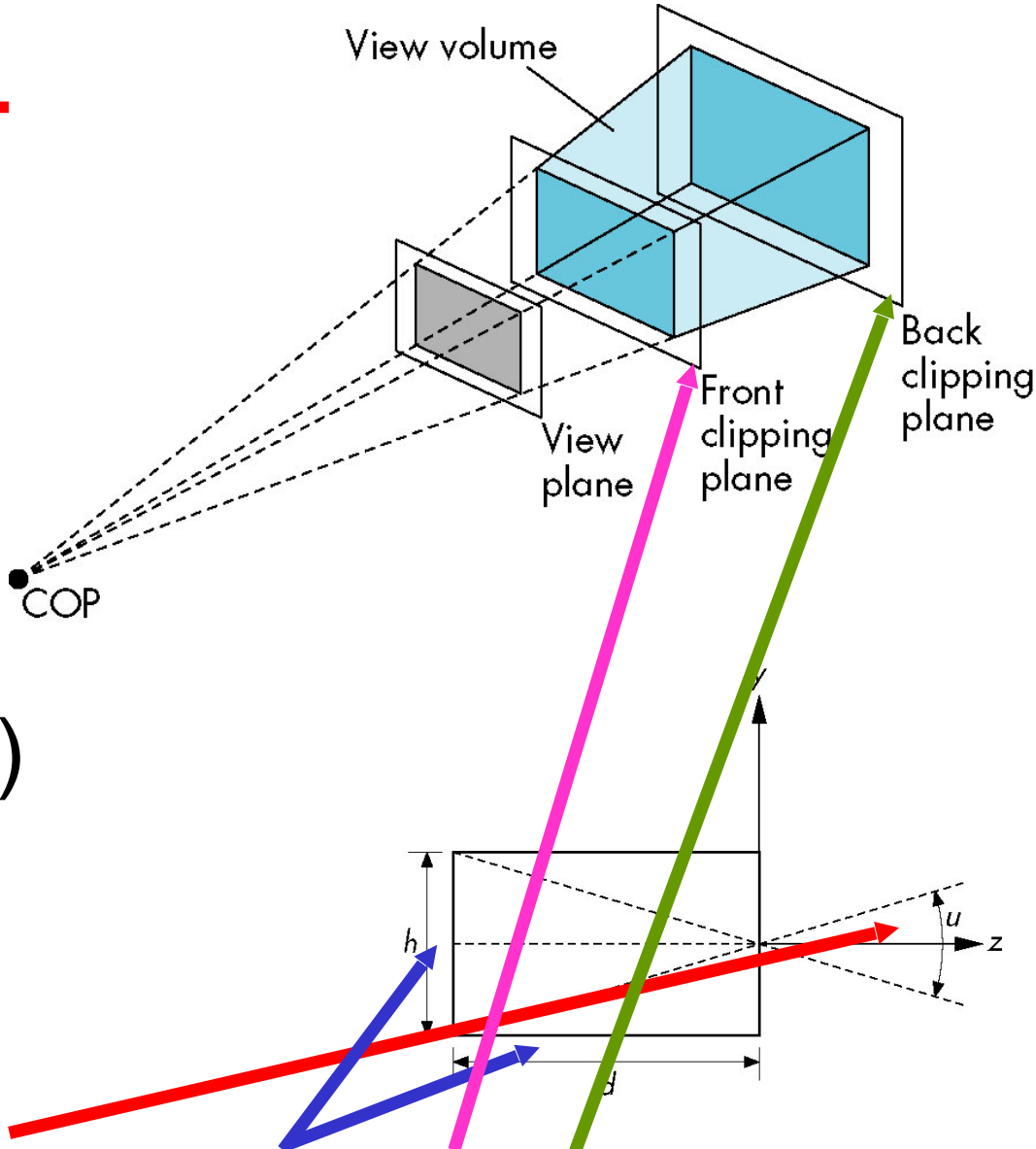




gluLookAt(**COP**_ x, **COP** _ y, **COP** _ z, **at**_ **x, at**_ **y, at**_ **z**, **up**_ x, **up**_ y, **up**_ z);

Projection Specification

- Six degrees of freedom
  - Position of center of lens
  - Orientation
- Lens (focal length)
- Film size (height & width)
- Orientation of film plane

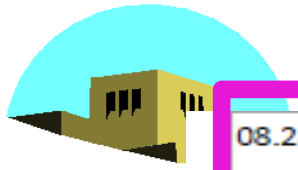


glPerspective(field_of_view, aspect_ratio, near, far);

Lights and Materials

- Types of lights
  - Point sources vs distributed sources
  - Spot lights
  - Near and far sources
  - Color properties

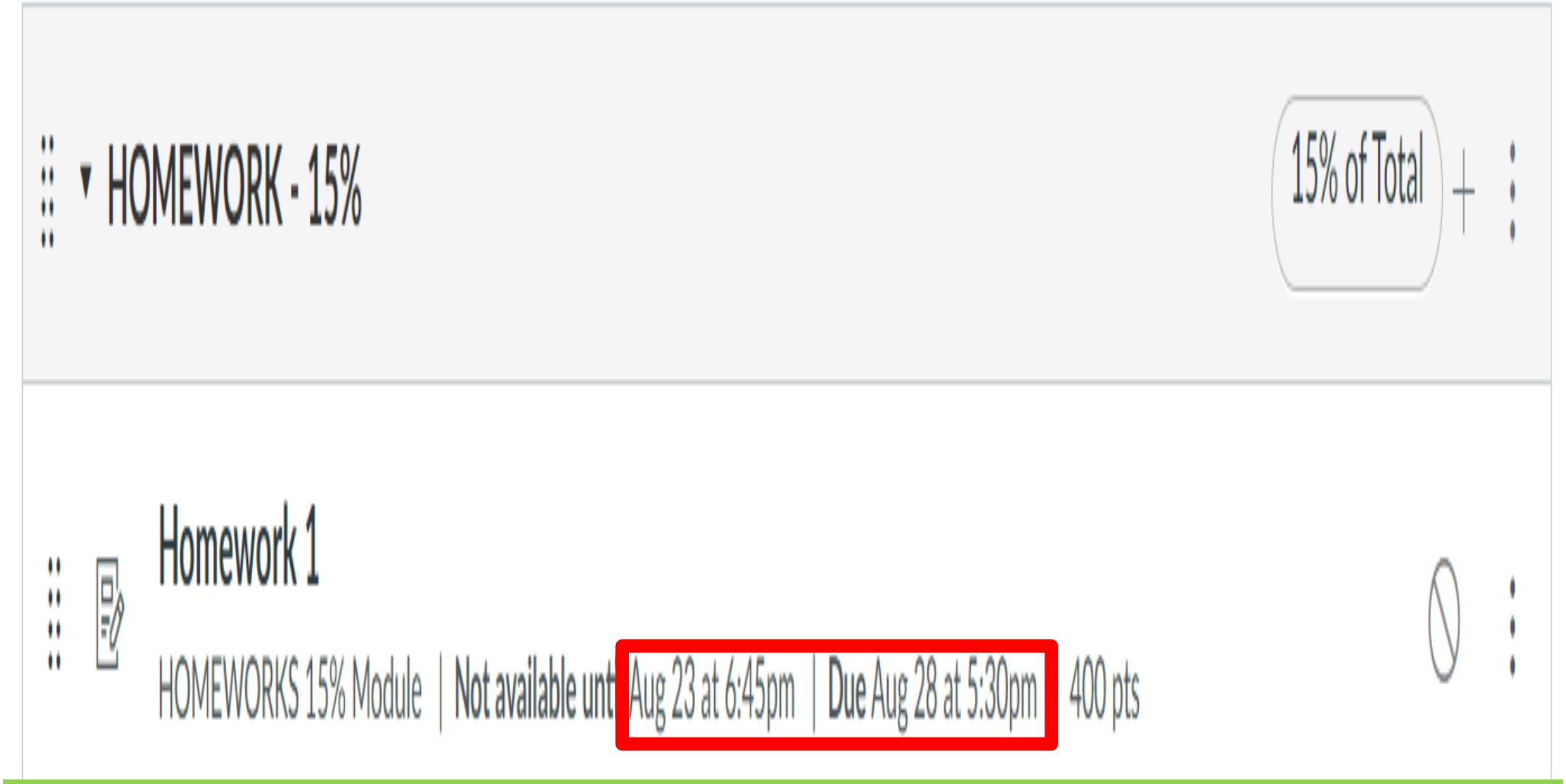
- **Material** properties
  - Absorption: color properties
  - Scattering
    - Diffuse
    - Specular



08.28.2023 (M 5:30 to 7) (3)	Homework 1	Lecture 2
08.30.2023 (W 5:30 to 7) (4)		Math Review 1
09.06.2023 (W 5:30 to 7) (5)	Homework 2	Lecture 3
09.11.2023 (M 5:30 to 7) (6)		Math Review 2
09.13.2023 (W 5:30 to 7) (7)	Homework 3	Lecture 4
09.18.2023 (M 5:30 to 7) (8)		PROJECT 1
09.20.2023 (W 5:30 to 7) (9)		EXAM 1 REVIEW
09.25.2023 (M 5:30 to 7) (10)		EXAM 1

At 6:30 PM NEXT.

The University of New Mexico



VH Publish.

H 2023 Fall 1

Home

Announcements

Assignments

Discussions

Grades

People

Pages

Files

Syllabus

Outcomes

Rubrics

Quizzes

Modules

Collaborations

Homework 1

Publish | Edit

Homework 1.doc

The HOMEWORK is due by 5:30 PM of the due date class.

The HOMEWORK will have a theoretical and a programming part.

You also are to submit the zip of the programming part - HOMEWORK1 (Visual Studio 2019 PROJECT (top FOLDER)) to CANVAS.

Rename Homework 1.doc to score.OPENGL.doc

You can do the programming part using WEBGL. You are to submit the zip of the programming part - HOMEWORK1 (WEBGL PROJECT (top FOLDER)) to CANVAS.

Rename Homework 1.doc to score.WEBGL.doc.

(MUST BE A .DOCX DOCUMENT)

Points 400

Name: _____

Homework 1

(400 points)

Hours spent:

The homework is to be turned in before 5:30 PM of the due date class.

Also, an implementation in Visual Studio 2019 or WEBGL is required, thus you are to submit the ZIPPED project (the top Homework1 folder) to CANVAS.

I UNDERSTAND THAT TURNING ANOTHER's WORK IN is CHEATING.
I UNDERSTAND THAT ANY KIND OF DISSEMINATION of this WORK is CHEATING.

I CERTIFY THAT THE HOMEWORKs SOLUTIONs ARE MY OWN WORK!

SIGNATURE: V X ?

HOMEWORK CHECKLIST (YOU MUST GRADE YOURSELF!):

1. A. 300 points – please count! ☐ 300 points
2. B. 100 points – please count! ☐ 100 points

3. Homework1.zip NOT submitted to CANVAS? ☐ -100 points

PLEASE ENTER YOUR GRADE IN THIS BOX:

Please rename Homework 1.doc to either
score.OPENGL.doc or score.WEBGL.doc

(Example 100.OPENGLdoc)

Warning: if your score is not honest you will get a zero.

At 6:45 PM.

End Class 2

VH, Download Attendance Report
Rename it:
8.23.2023 Attendance Report FINAL

VH, upload Lecture 1 to CANVAS.