UNIVERSITY of HOUSTON

DEPARTMENT OF COMPUTER SCIENCE

COSC 4370 Fall 2023

Interactive Computer Graphics

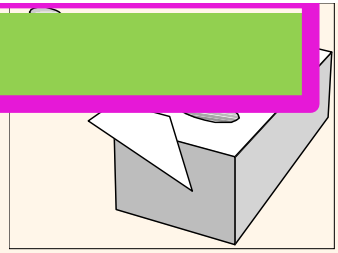
M & W 5:30 to 7:00 PM

Prof. Victoria Hilford

PLEASE TURN your webcam ON

NO CHATTING during LECTURE

From 5:30 to 5:45 PM – 15 minutes.



COSC 4370

5:30 to 7

PLEASE

LOG IN

CANVAS

Please close all other windows.

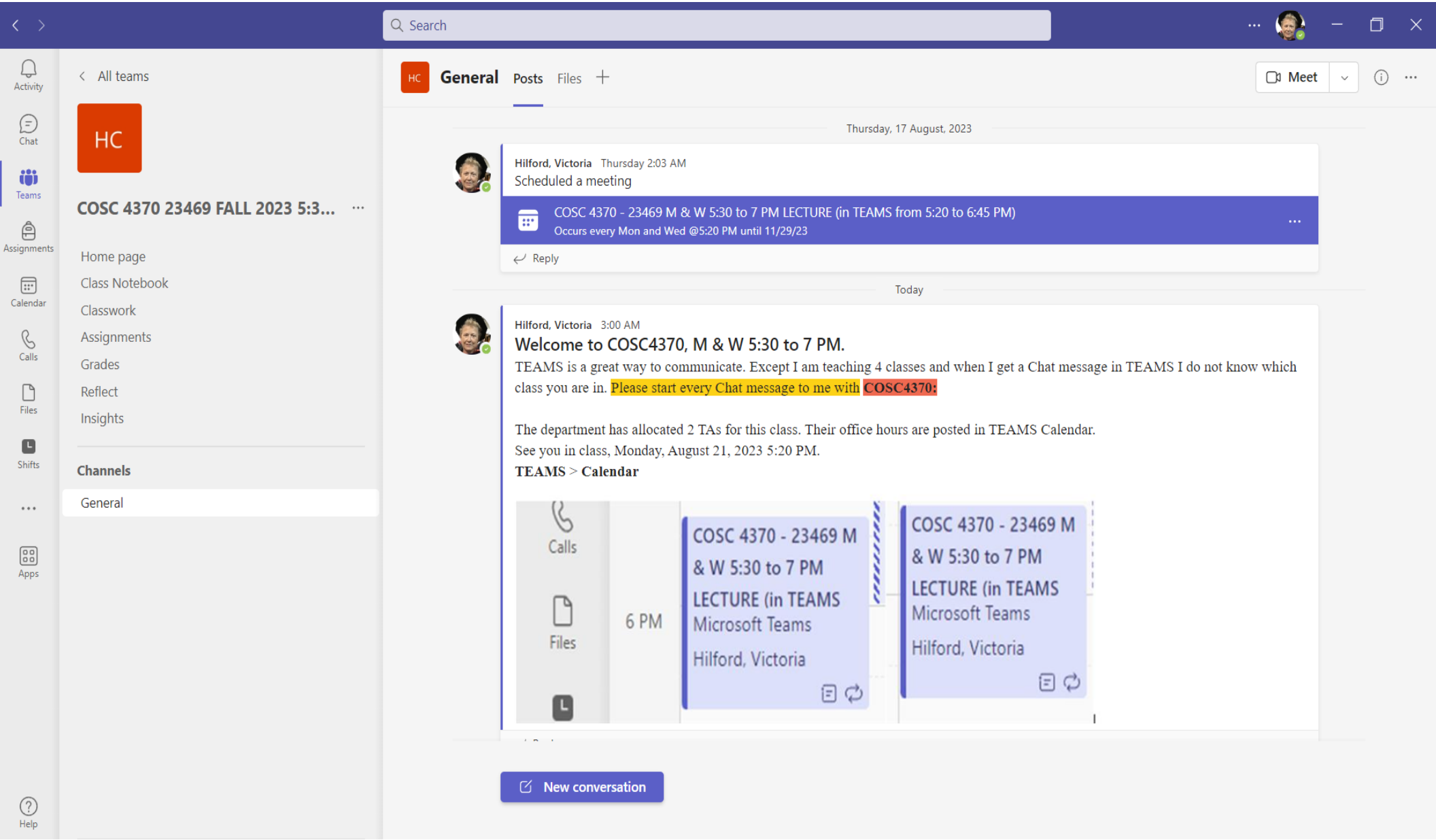
VH, Victoria Hilford.

Office Hours:
M & W 12 to 12:30 PM
TEAMS (online synchronous)

TA, Shu-Yuan Yang (A – L).
Office Hours:
M & W 10 to 11 AM
TEAMS (online synchronous)

TA, ??? (M – Z).
Office Hours:
M & W ? to ? PM
TEAMS (online synchronous)

TEAMS CHAT



You may not use TEAMS or CANVAS Class Roll to create outside groups

On the syllabus

UNIVERSITY of HOUSTON

NATURAL SCIENCES AND MATHEMATICS

COURSE TITLE/SECTION: COSC4370 – Interactive Computer Graphics (23469)

TIME: MW 5:30 – 7 PM

FACULTY: Dr. Victoria Hilford **OFFICE HOURS: MW (12–12:30 PM)**

Synchronous ONLINE (attendance required, no class recording)

E-mail: CANVAS Email, TEAMS **Phone: N/A** **FAX: N/A**

The information contained in this class syllabus is subject to change without notice. Students are expected to be aware of any additional course policies presented by the instructor during the course.
This course is using TEAMS and CANVAS.

LAPTOP REQUIREMENT

All students must have a laptop for the duration of the class.

I. **Course:** COSC4370 – Interactive Computer Graphics

A. **Catalog Description:** Introduction to graphics hardware and software; interactive systems to display three-dimensional objects; display and input devices; and alternative algorithms, system configuration, and design trade-offs.

B. **Prerequisites**: A grade of C- or better in MATH 2318, and COSC 2320 or COSC 2430.

II. **Course Contents:**
Graphics Systems and Models, Graphics Programming, Geometric Objects and Transformations, Viewing, Lighting and Shading, From Vertices to Fragments, Discrete Techniques, Modeling and Hierarchy, Procedural Methods, Curves and Surfaces, Advanced Rendering.

III. Course Structure:

We will be using CANVAS for all contents including lectures, homework assignments, project assessments, and exams.
We will be using Visual Studio 2019 and Open GL libraries (installation instructions in CANVAS). By the end of the course, we will attempt to convert from OpenGL to WebGL (based on OpenGL libraries).

IV. Textbook (optional):

Interactive Computer Graphics by Edward Angel eBook:

https://www.pearson.com/en-us/subject-catalog/p/interactive-computer-graphics-a-top-down-approach-with-webgl/P200000003526/9780135217160

OpenGL A Primer by Edward Angel 3E

V. Course Requirements

A. Exams:

- 4 Exams
- There are no makeup Exams.

B. Projects:

- 4 Projects (Visual Studio 2019 and OpenGL large programming assignments due at the beginning of the class)
- There are no makeup Projects.
- Projects Acceptance Testing files are self-graded. Incorrect scores are considered cheating, and it will result in a score of ZERO.

C. Homework Assignments:

- 12 Homework Assignments (Visual Studio 2019 and OpenGL programming assignments due at the beginning of the class)
- There are no makeup Homework Assignments.
- Homework Assignments are self-graded. Incorrect scores are considered cheating, and it will result in a score of ZERO.

D. Class Participation:

- 14 Class Participations (Visual Studio 2019 and OpenGL programming assignments created during class time due at the end of the class)
- There are no makeup Class Participations.
- Class Participations are self-graded. Incorrect scores are considered cheating, and it will result in a score of ZERO.

VI. Evaluation and Grading:

Must be in TEAMS to take any graded assignment. Any attempt to take a graded assignment not being in TEAMS will result in a grade of ZERO for that graded assignment.

Exams: 40%
Projects: 30%
Homework Assignments: 15%
Class Participation: 15%

A: [93+
A-: [90-93)

B+: [87-90)
B: [83-87)
B-: [80-83)

C+: [77-80)
C: [73-77)
C-: [70-73)

D+: [67-70)
D: [63-67)
D-: [60-63)

F: 0-60

Policy on grades of I (Incomplete): The grade of "I" (Incomplete) is a conditional and temporary grade given when a student, for reasons beyond his or her control, has not completed a relatively small portion of all requirements. Sufficiently serious, documented situations include illness, death in the family, etc.

Presence in Class
Your presence in class each class is required, class is synchronous online.
Classes will not be recorded by the instructor.

Reasonable Academic Adjustments/Auxiliary Aids
The University of Houston complies with Section 504 of the Rehabilitation Act of 1973 and the Americans with Disabilities Act of 1990, pertaining to the

Excused Absence Policy

Regular class attendance, participation, and engagement in coursework are important contributors to student success. Absences may be excused as provided in the University of Houston Undergraduate Excused Absence Policy and Graduate Excused Absence Policy for reasons including: medical illness of student or close relative, death of a close family member, legal or government proceeding that a student is obligated to attend, recognized professional and educational activities where the student is presenting, and University-sponsored activity or athletic competition. Under these policies, students with excused absences will be provided with an opportunity to make up any quiz, exam or other work that contributes to the course grade or a satisfactory alternative. Please read the full policy for details regarding reasons for excused absences, the approval process, and extended absences. Additional policies address absences related to military service, religious holy days, pregnancy and related conditions, and disability.

Please remember that this is a synchroneous online class. TEAMS and CANVAS can be accessed from any mobile device. You need to be available just the 1 hour and 15 minutes (from 5:30 to 6:45 PM).

Notification of all excused absences must be sent to the instructor before the excused class.

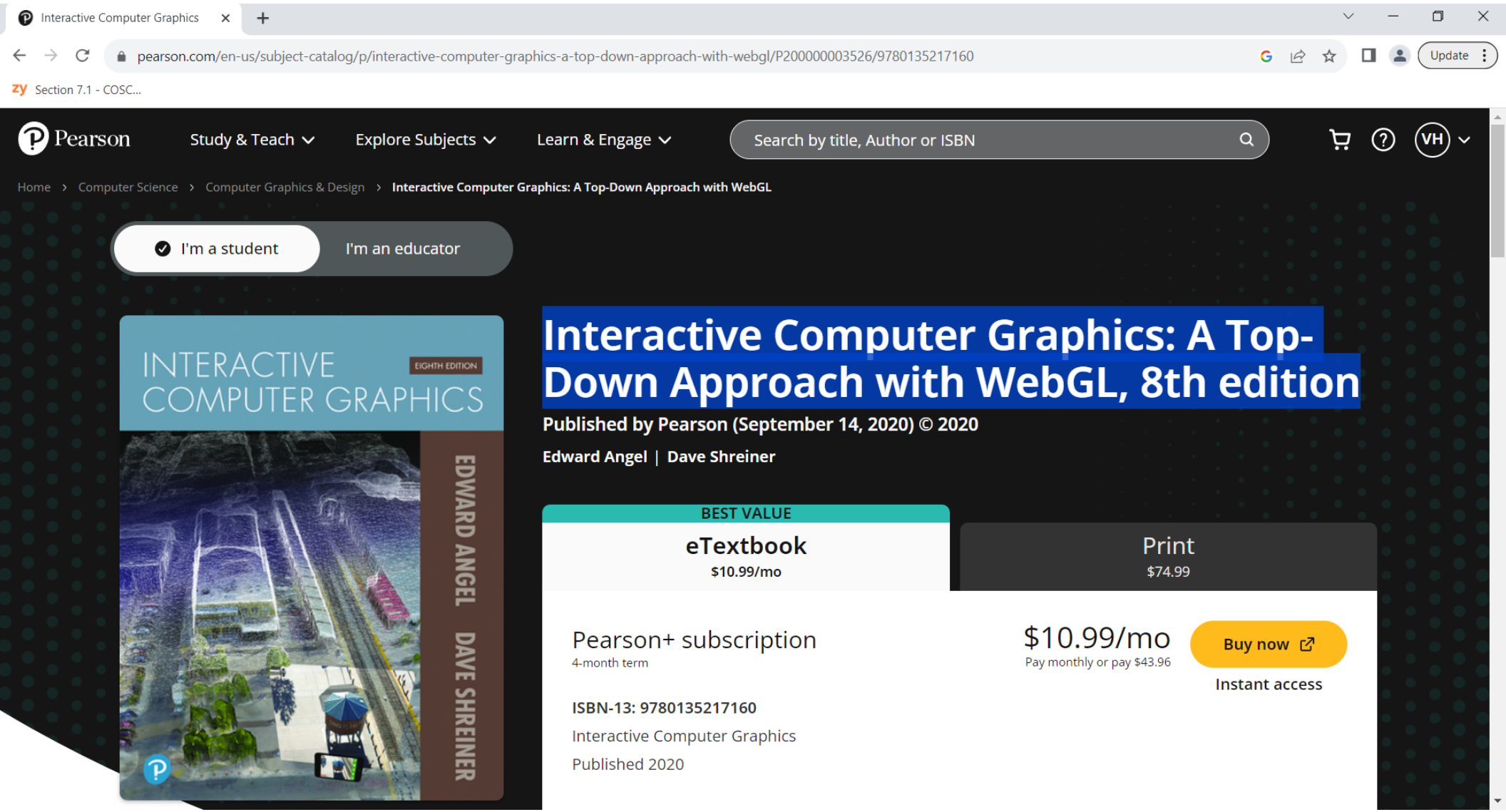
Recording of Class

Students may not record all or part of class, livestream all or part of class, or make/distribute screen captures, without advanced written consent of the instructor. If you have or think you may have a disability such that you need to record class-related activities, please contact the Justin Dart, Jr. Student Accessibility Center. If you have accommodation to record class-related activities, those recordings may not be shared with any other student, whether in this course or not, or with any other person or on any other platform. Failure to comply with requirements regarding recordings will result in a disciplinary referral to the Dean of Students Office and may result in disciplinary action.

Recording of Class

Students may not record all or part of class, livestream all or part of class, or make/distribute screen captures, without advanced written consent of the instructor. If you have or think you may have a disability such that you need to record class-related activities, please contact the Justin Dart, Jr. Student Accessibility Center. If you have accommodation to record class-related activities, those recordings may not be shared with any other student, whether in this course or not, or with any other person or on any other platform. Failure to comply with requirements regarding recordings will result in a disciplinary referral to the Dean of Students Office and may result in disciplinary action.

https://www.pearson.com/en-us/subject-catalog/p/interactive-computer-graphics-a-top-down-approach-with-webgl/P200000003526/9780135217160



COSC 4370 – Fall 2023 5:30 to7 PM. Detailed Syllabus

08.21.2023 (M 5:30 to 7) (1)		Lecture 0
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

09.27.2023 (W 5:30 to 7) (11)	Homework 4	Lecture 5
10.02.2023 (M 5:30 to 7) (12)	Homework 5	Lecture 6
10.04.2023 (W 5:30 to 7) (13)	Homework 6	Lecture 7
10.09.2023 (M 5:30 to 7) (14)		PROJECT 2
10.11.2023 (W 5:30 to 7) (15)		EXAM 2 REVIEW
10.16.2023 (M 5:30 to 7) (16)		EXAM 2

10.18.2023 (W 5:30 to 7) (17)	Homework 7	Lecture 8
10.23.2023 (M 5:30 to 7) (18)		Lecture 9
10.25.2023 (W 5:30 to 7) (19)		Lecture 10
10.30.2023 (M 5:30 to 7) (20)		PROJECT 3
11.01.2023 (W 5:30 to 7) (21)		EXAM 3 REVIEW
11.06.2023 (M 5:30 to 7) (22)		EXAM 3

11.08.2023 (W 5:30 to 7) (23)		Lecture 11
11.13.2023 (M 5:30 to 7) (24)	Homework 8	Lecture 12
11.15.2023 (W 5:30 to 7) (25)		Lecture 13
11.20.2023 (M 5:30 to 7) (26)		PROJECT 4
11.27.2023 (M 5:30 to 7) (27)		EXAM 4 REVIEW
11.29.2023 (M 5:30 to 7) (28) LAST CLASS		EXAM 4

From 5:45 to 6:35 PM – 50 minutes.

Objectives

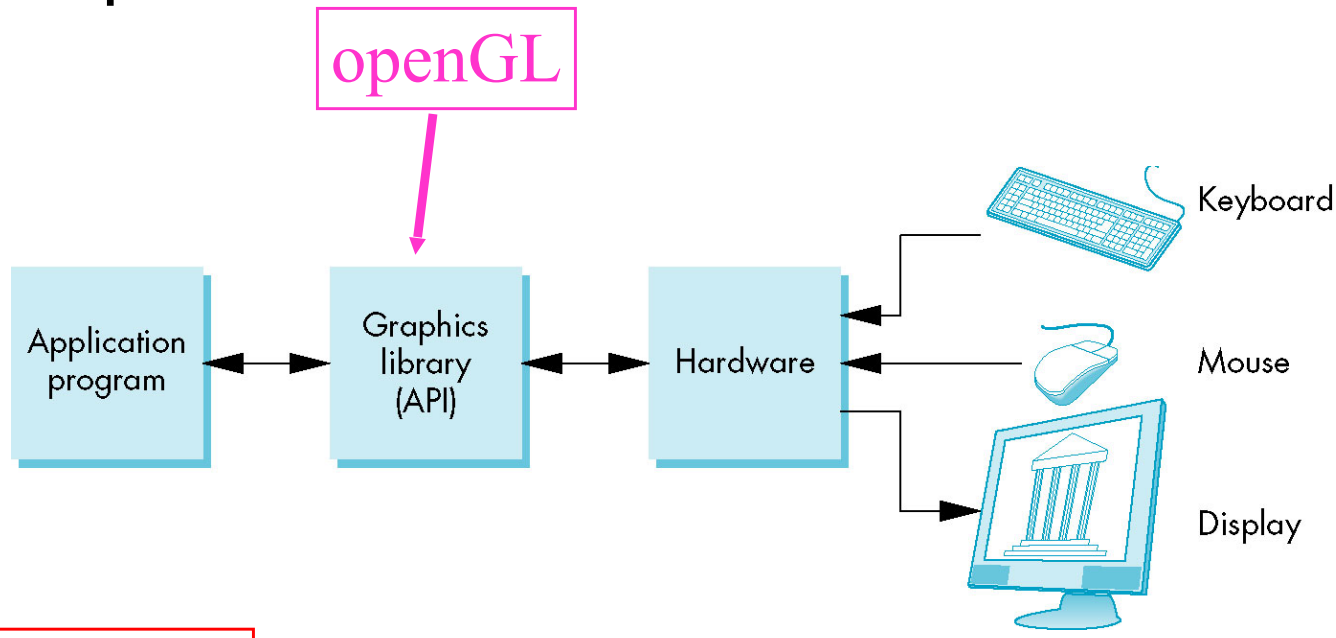
- Broad introduction to Computer Graphics
  - Software
  - Hardware
  - Applications
- Top-down approach
- OpenGL

Outline

Part 1: Introduction

Text: Chapter 1

- What is Computer Graphics?
- Applications Areas
- History
- Image formation
- Basic Architecture



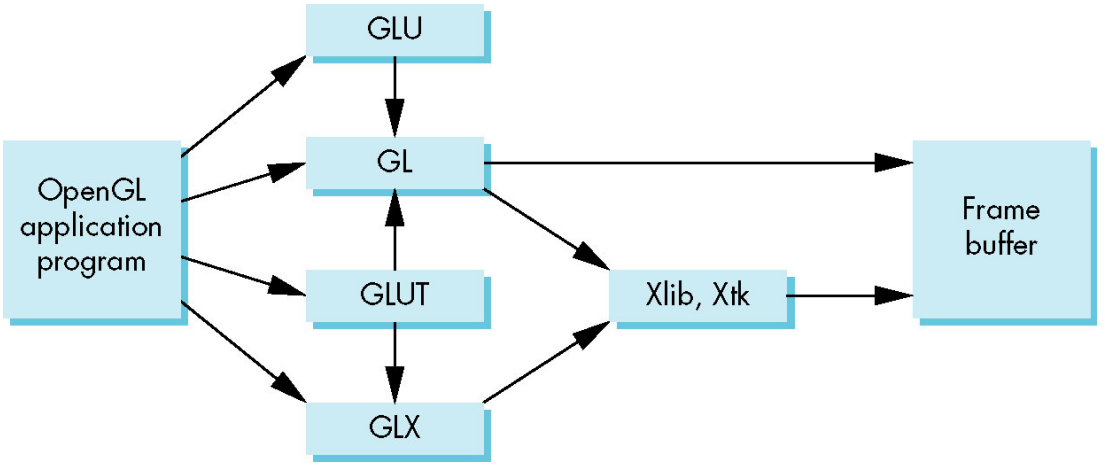
application program

display

Outline

Part 2: Basic OpenGL

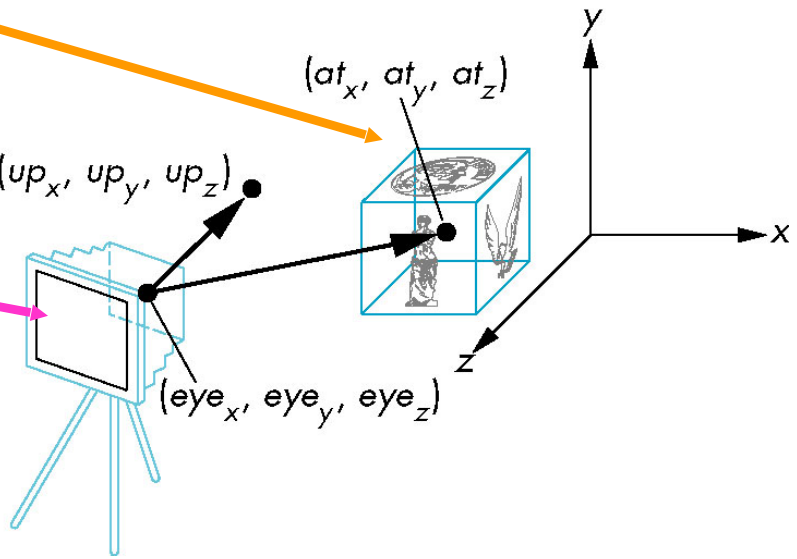
- Text: Chapters 2-3
  - Architecture
  - GLUT
  - Simple programs in two and three dimensions
  - Interaction



Outline

Part 3: Three-Dimensional Graphics

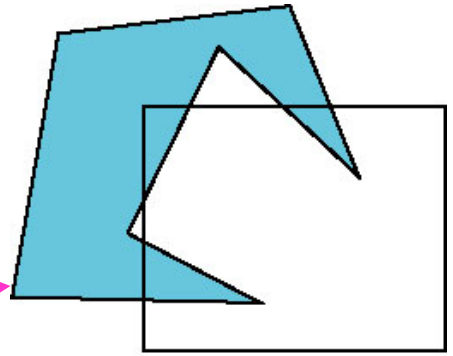
- Text: Chapters 4-6
  - Geometry
  - Transformations
  - Homogeneous Coordinates
  - Viewing
  - Shading

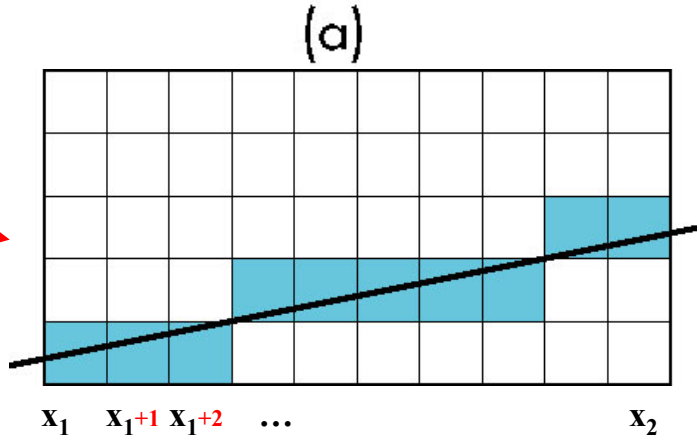


Outline

Part 5: Implementation

- Text: Chapter 7
  - Approaches (**object** vs image space)
  - Implementing the pipeline
  - Clipping
  - Line drawing (Explicit Representation)
  - Polygon Fill
  - Display issues (color)





$y_i = m * x_i + h$

Raster display will excite pixels at discrete x_i and y_i pixel addresses

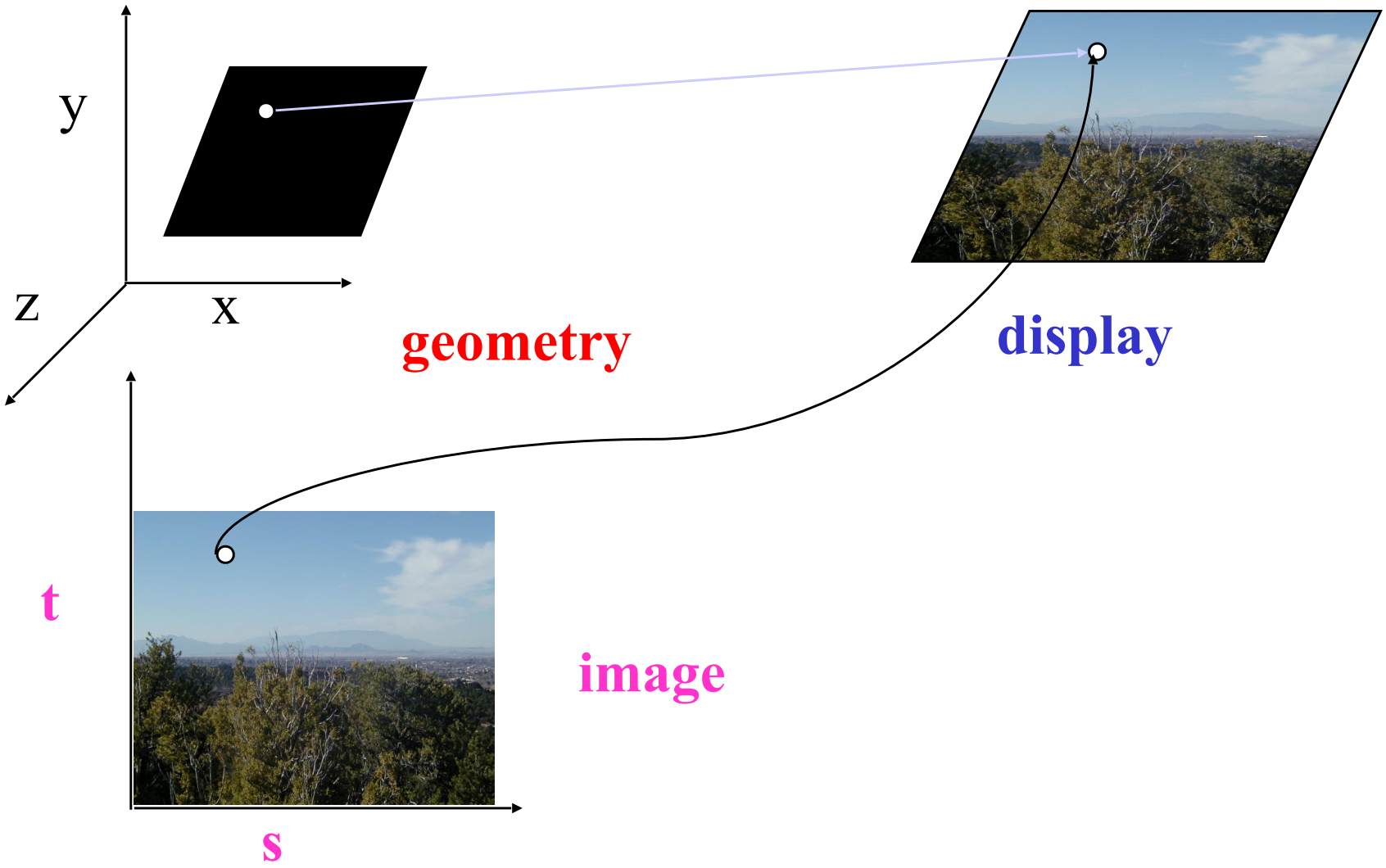
Outline

Part 4: Discrete Methods

- Text: Chapter 8
  - Buffers
  - Bitmaps and Pixel Maps
  - Texture Mapping
  - Compositing and Transparency

(a)

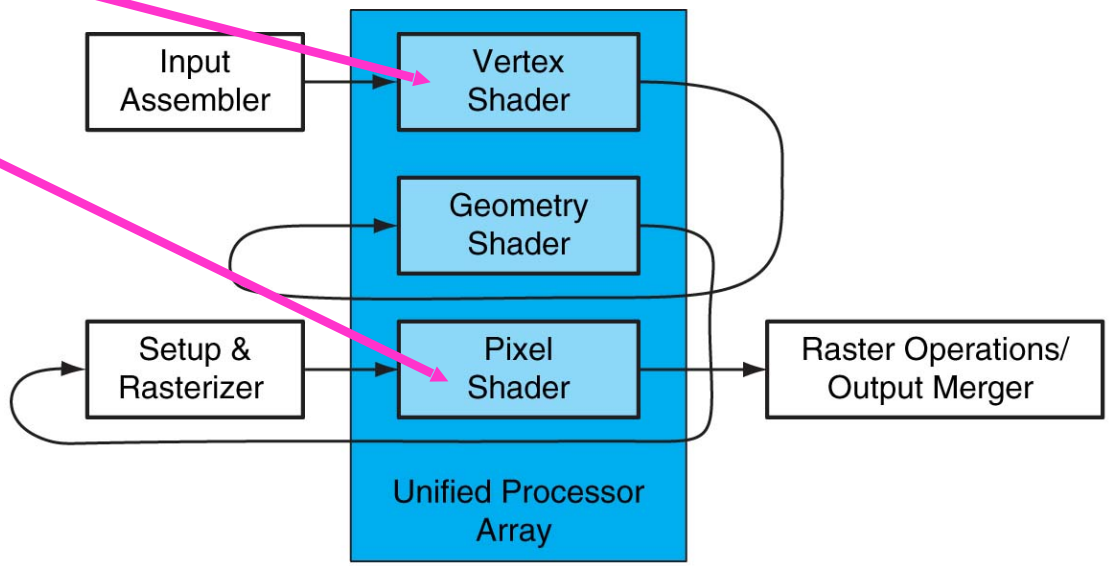
Texture Mapping



Outline

Part 6: Programmable Pipelines

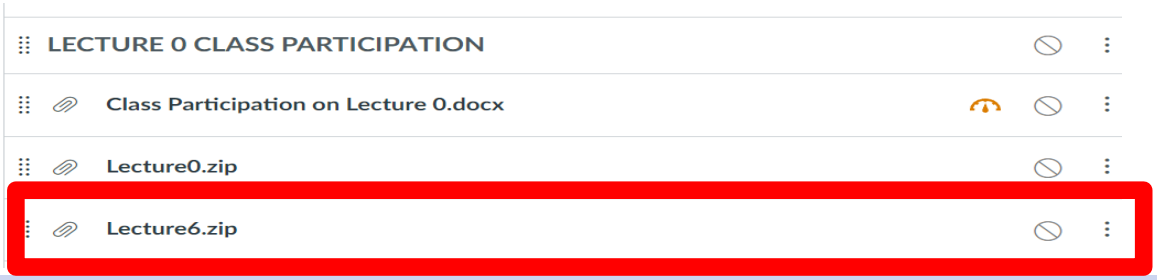
- Text: Chapter 9
  - Shading Languages
  - GLSL (OpenGL Shading Language)
  - Vertex Shaders
  - Fragment (Pixel) Shaders

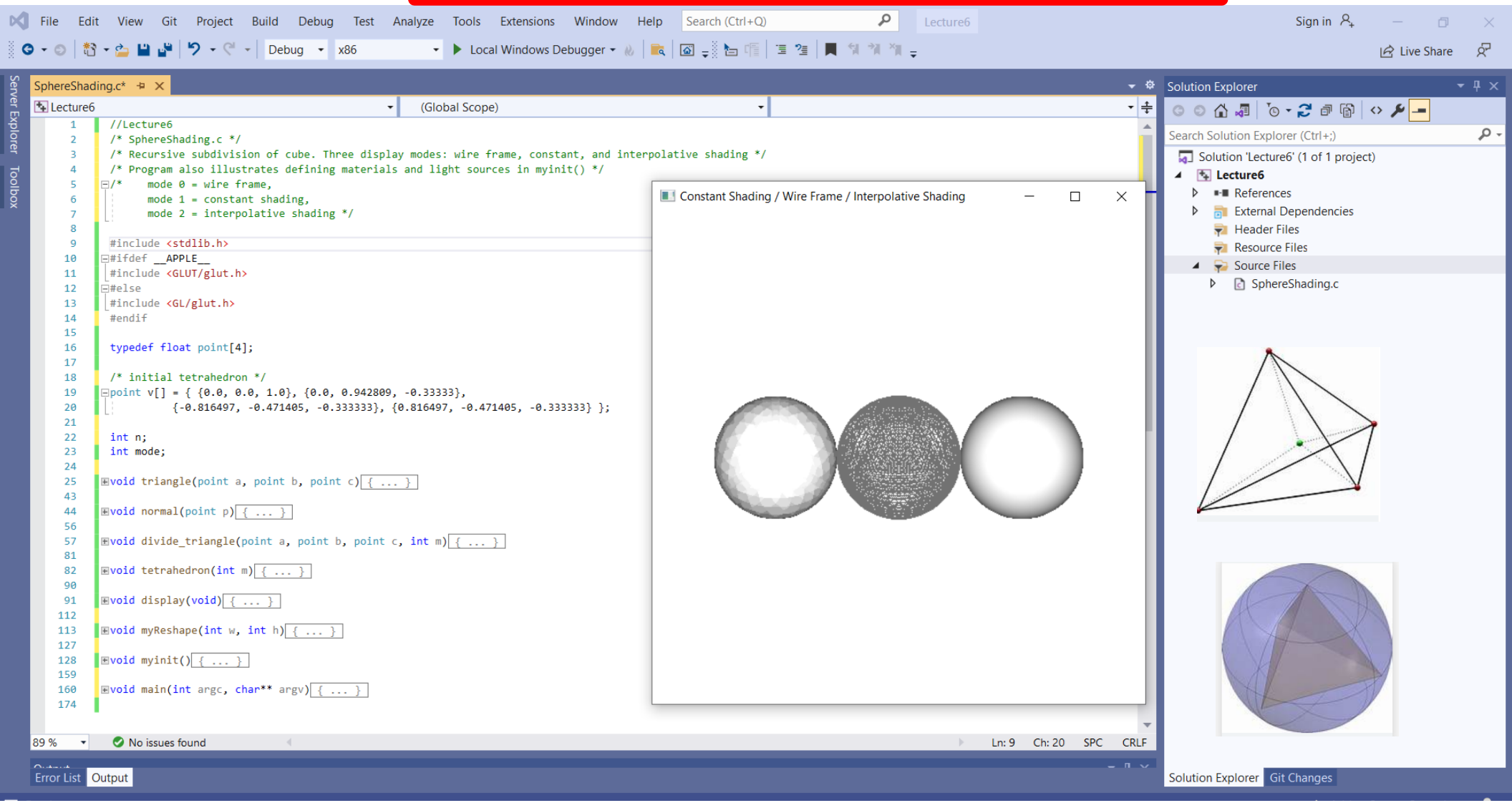


DEMO Shading

Download; UnZip; Click Solution

VH ONLY

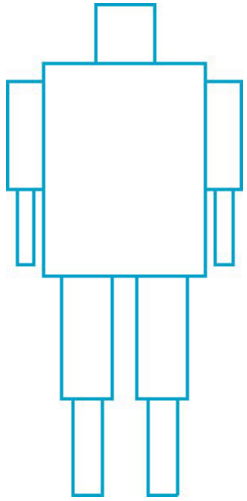


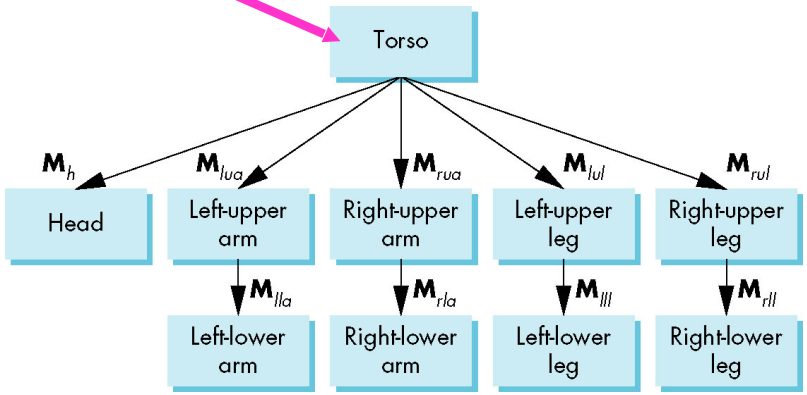


Outline

Part 7: Hierarchy and Procedural Methods

- Text: Chapters 10-11
- Tree Structured **Models**
  - Traversal Methods
  - Scene Graphs
  - Particle Systems

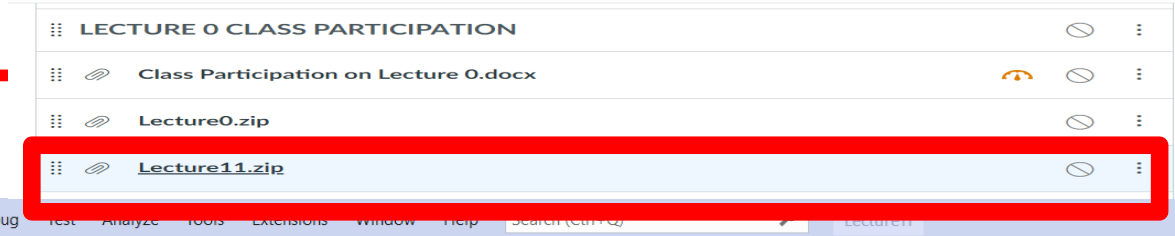


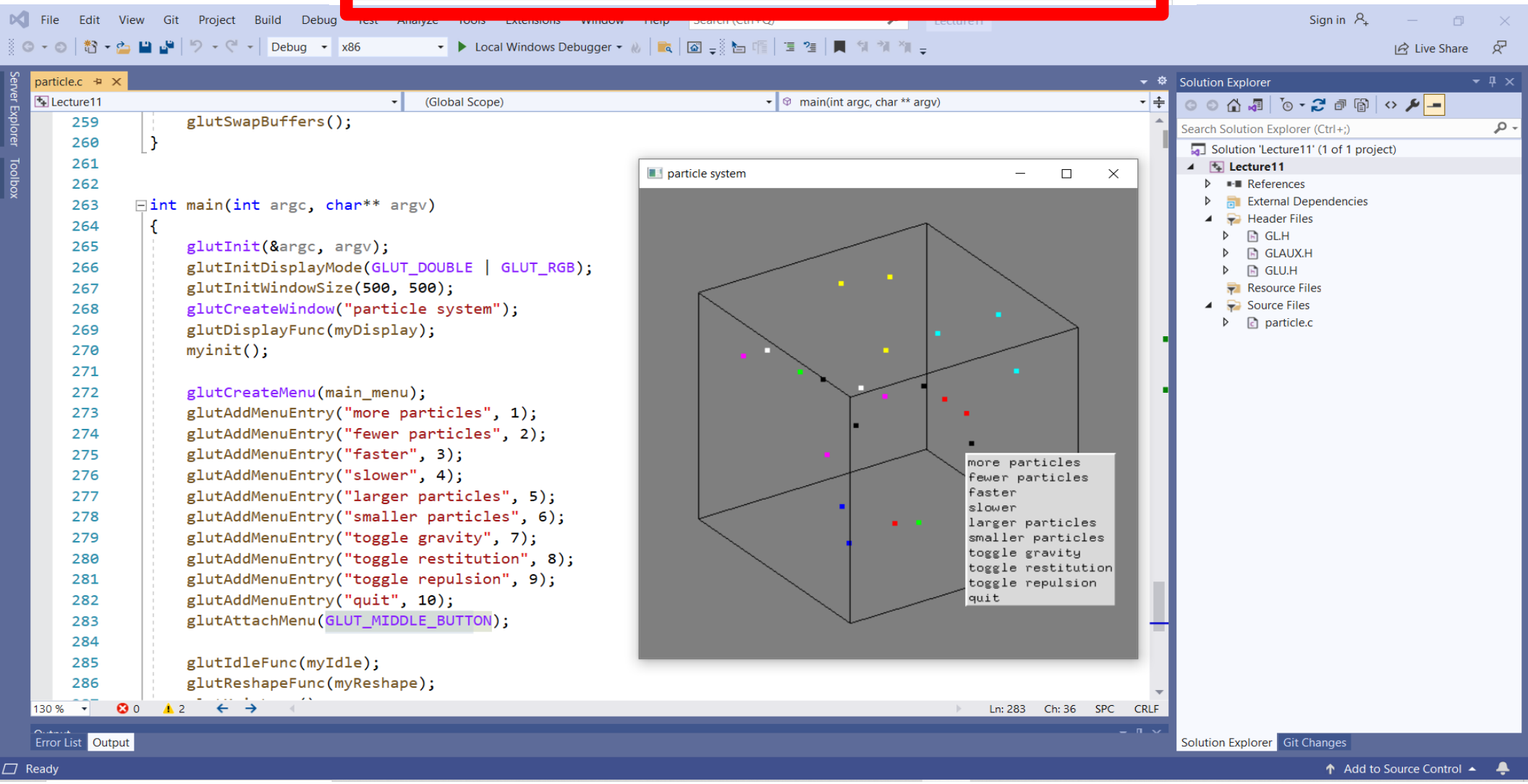


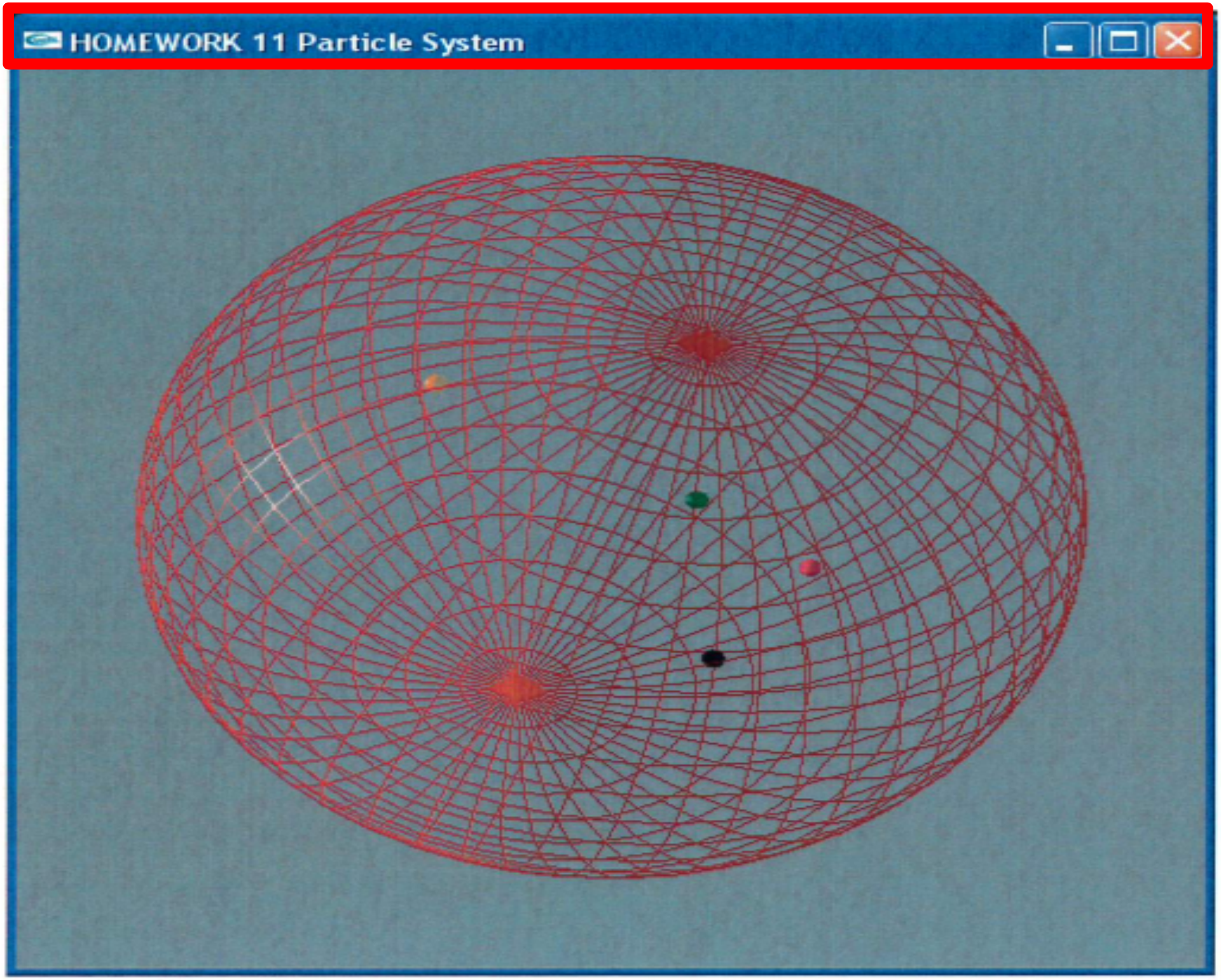
DEMO Particle Systems

Download; UnZip; Click Solution Middle Button

VH ONLY



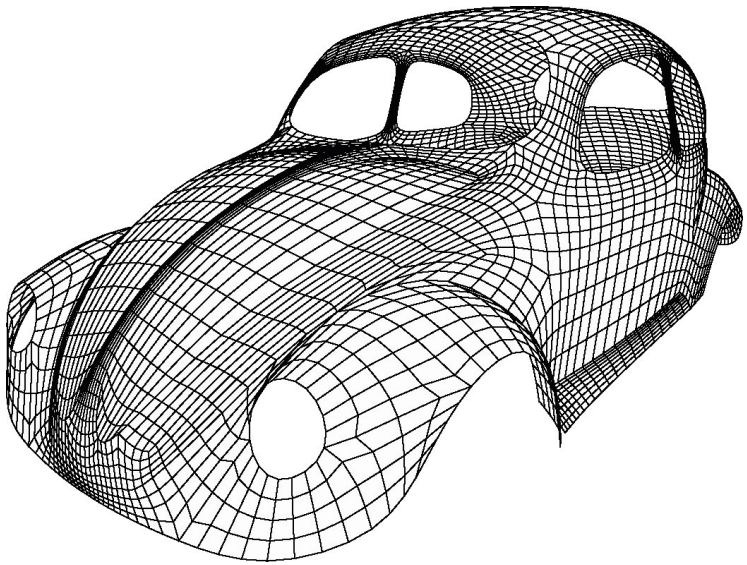



HOMEWORK 11 Particle System

Outline

Part 8: Curves and Surfaces

- Text: Chapter 12



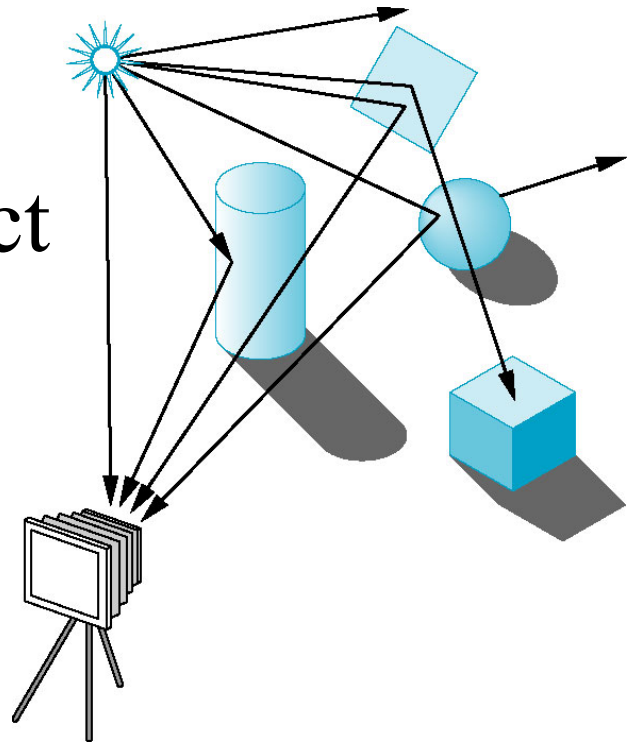
Outline

Part 9: Advanced Rendering

- Text: Chapter 13

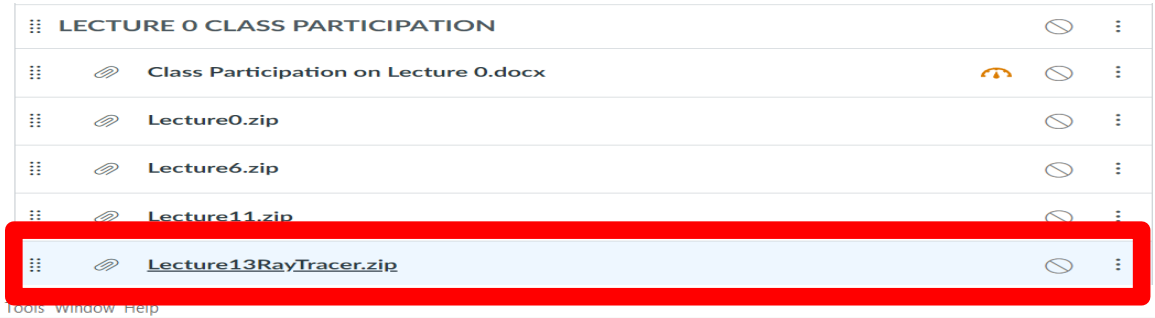
- Ray Tracing

Some objects are blocked from light
Light can reflect from object to object
Some objects might be translucent

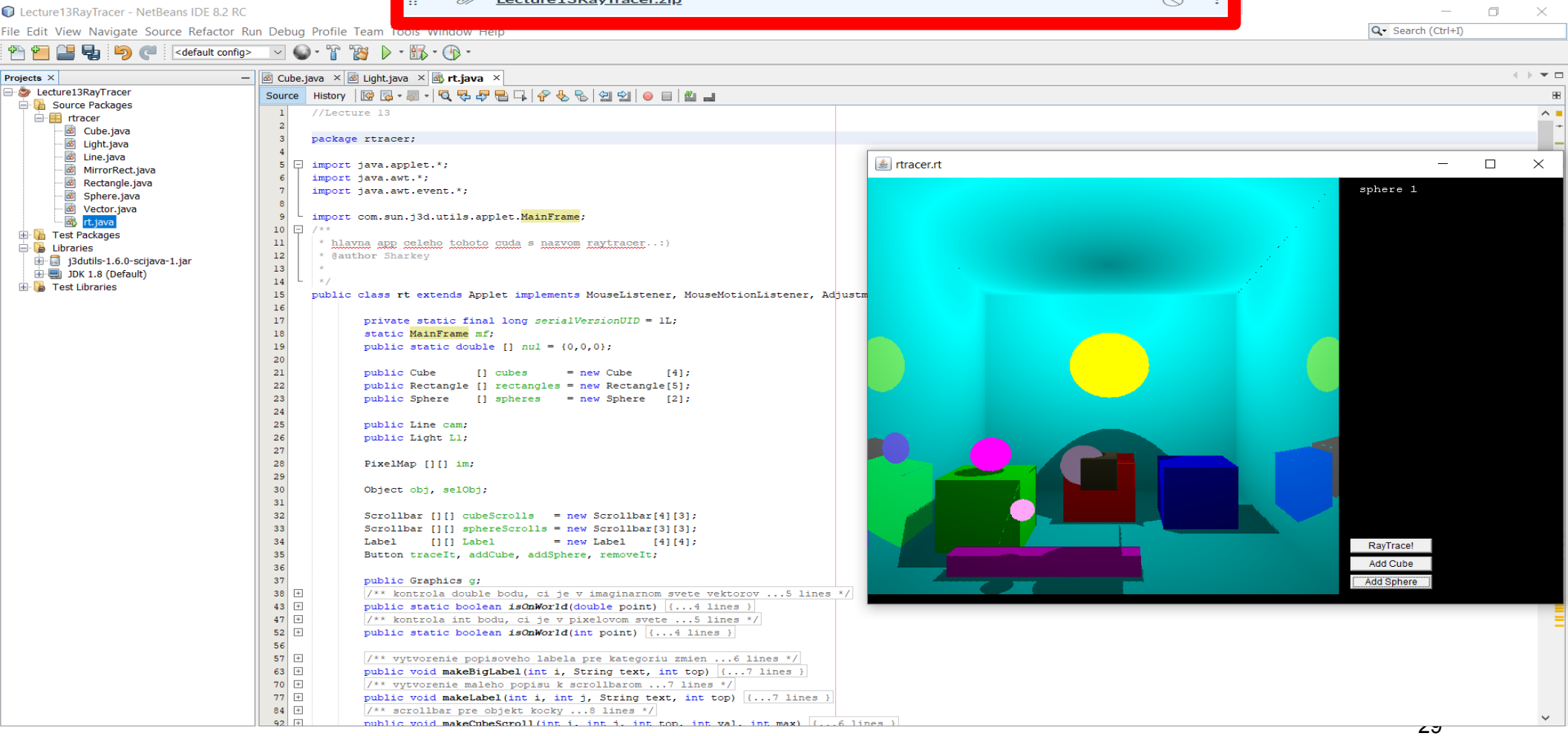


DEMO Ray Tracer

Download; UnZip; NetBeans Open Solution



VH ONLY



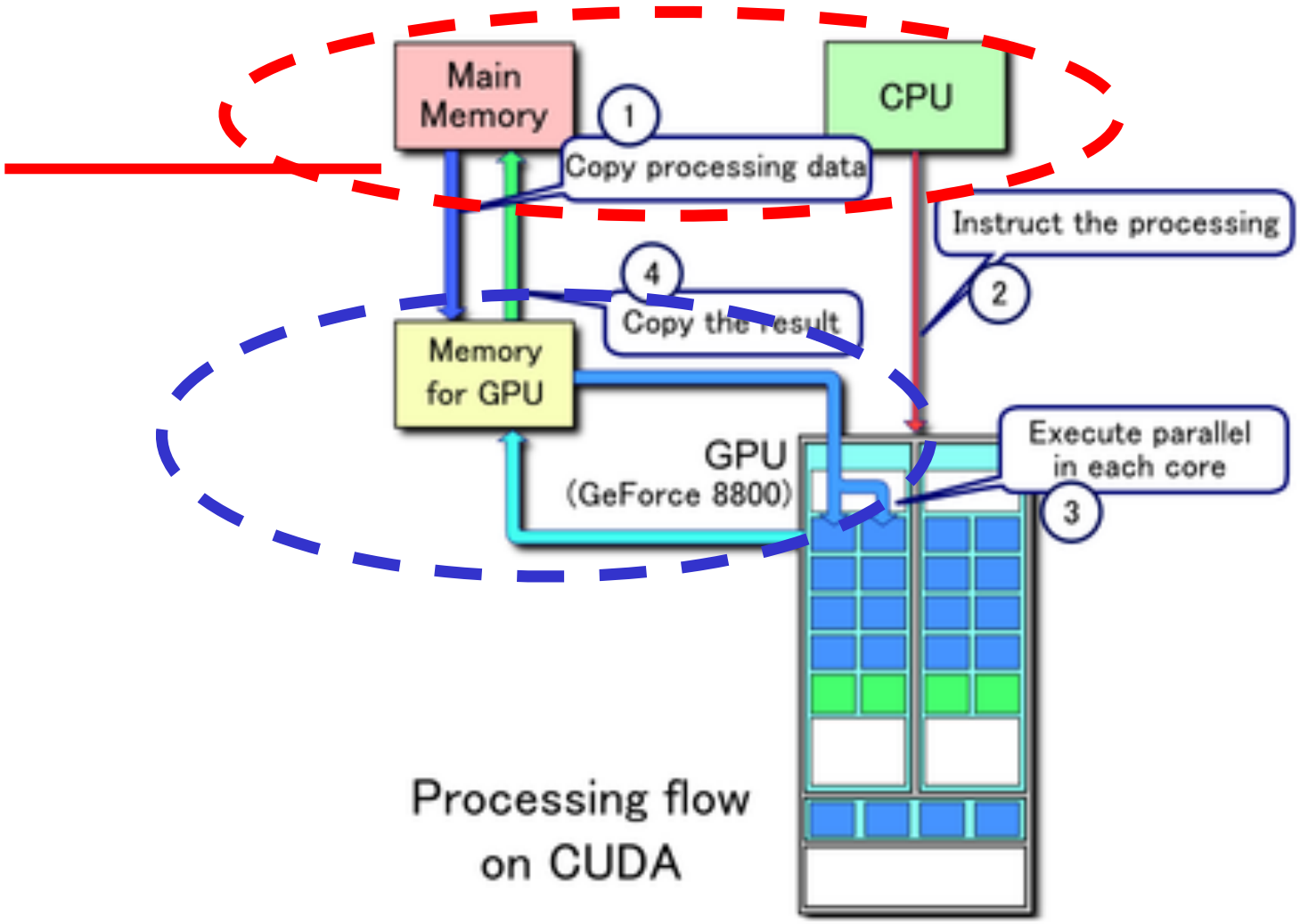
COSC 4370 – Computer Graphics

Lecture 0 - Computer Graphics is Cool

GPU - Hardware

Computer Graphics OpenGL Demo

GPGPU on Cuda



1. Copy data from **Main Memory** to **GPU Memory**
2. **CPU** instructs the process to **GPU**
3. **GPU** execute parallel in each core
4. Copy the result from **GPU Memory** to **Main Memory**

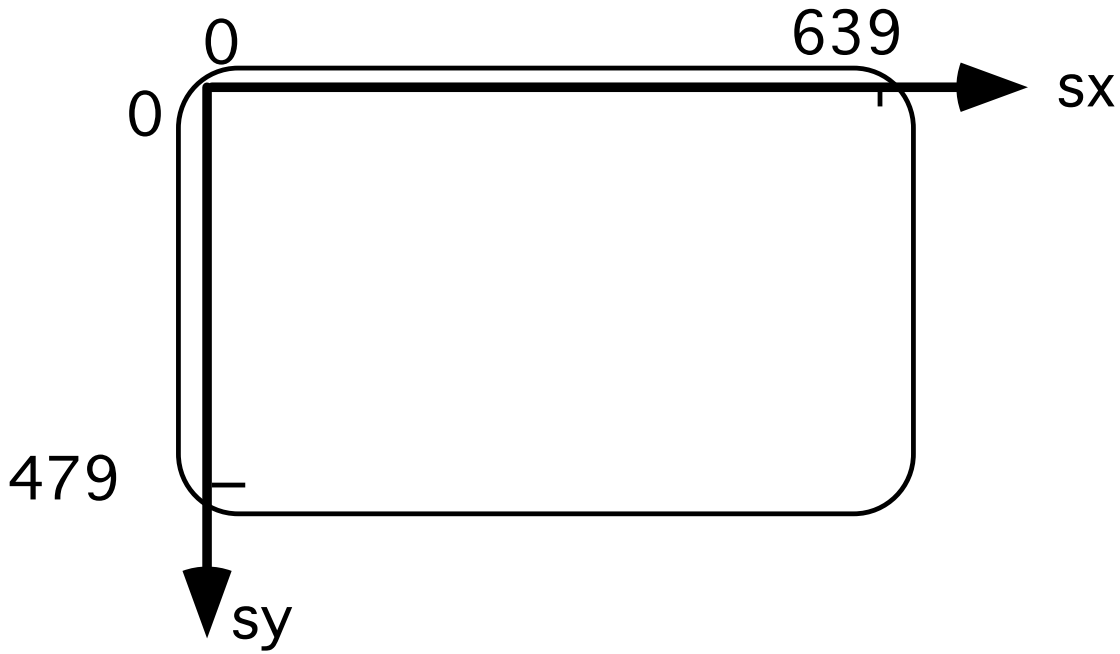
Graphics Display Devices

- Raster displays:

  Computer monitor: moves a beam of electrons across the screen from left to right and top to bottom.

  Printer: does the same thing with ink or toner.

  Coordinate system used:



Output Devices
CRT

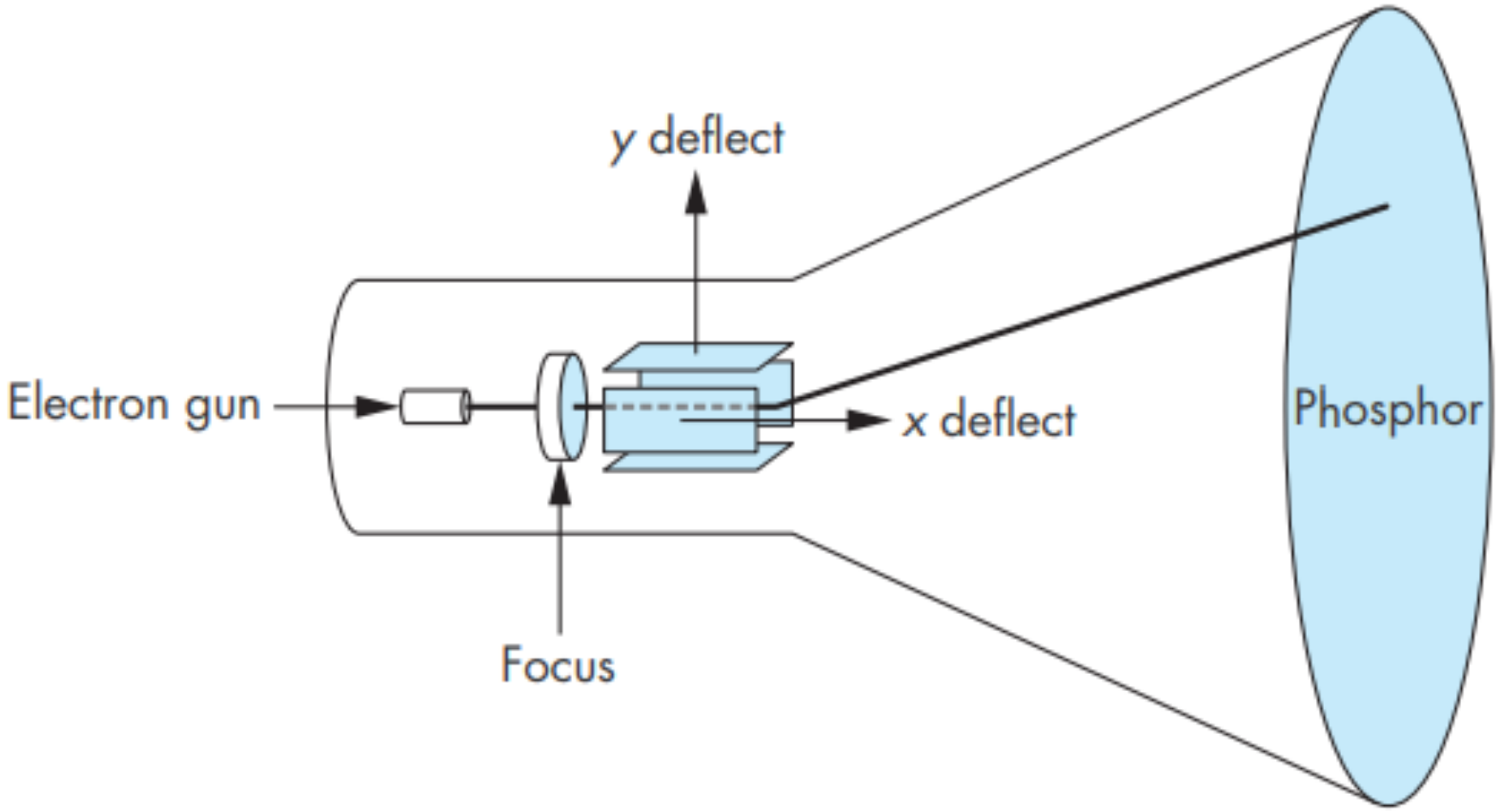
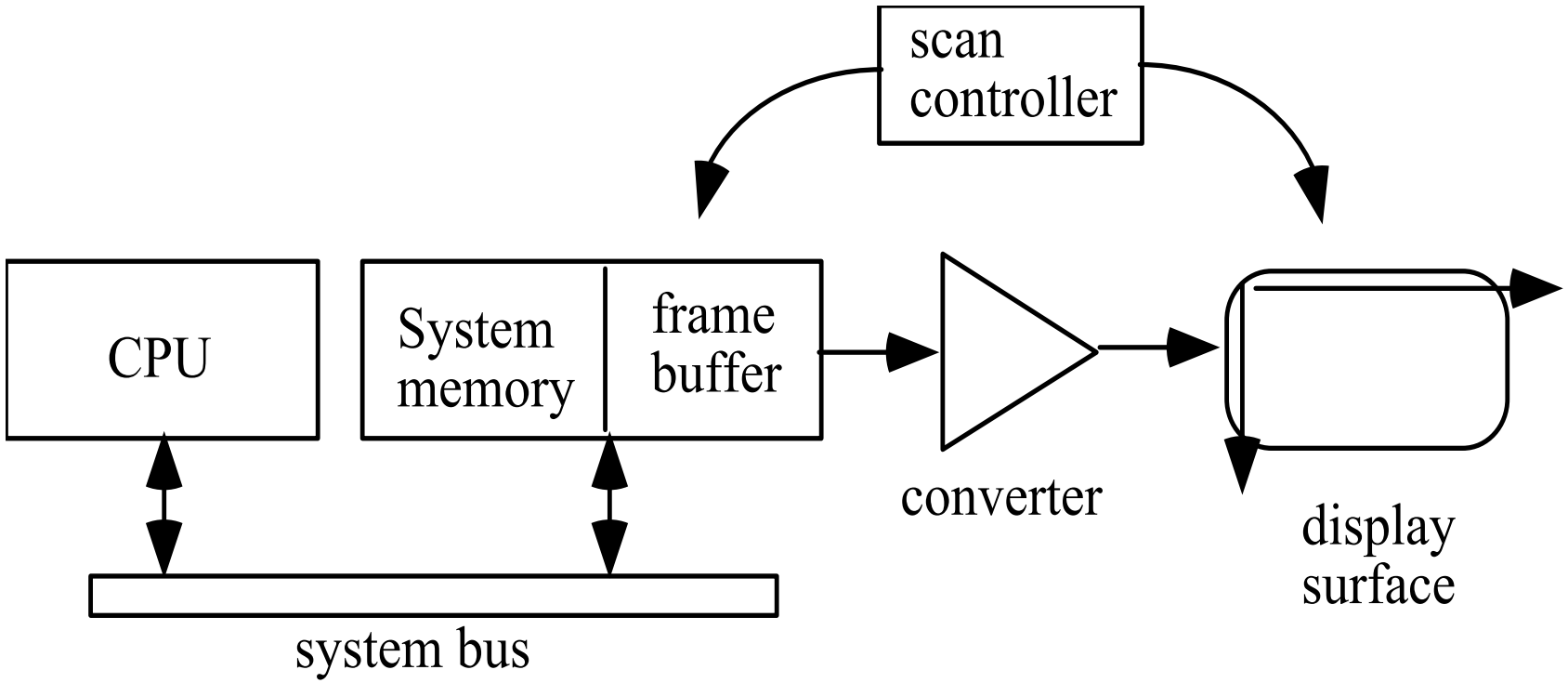
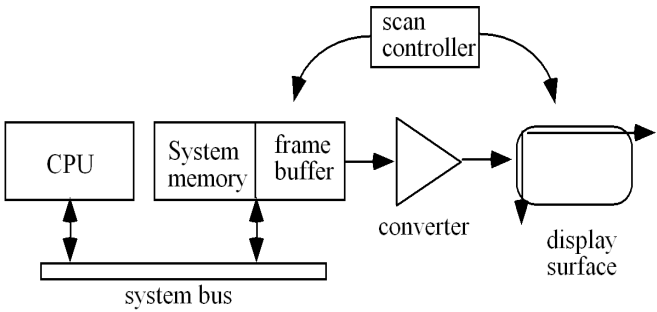


FIGURE 1.3 The cathode-ray tube (CRT).

Function of Scan Controller



Graphics Display Devices

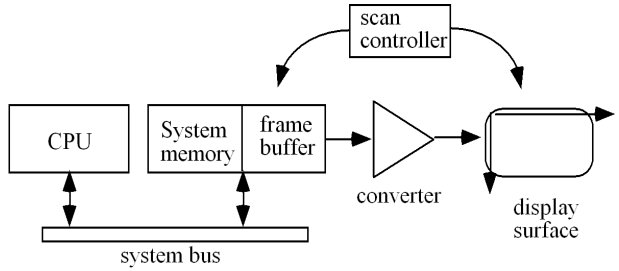


Raster displays are always connected to a frame buffer, a region of memory sufficiently large to hold all the pixel values for the display.

The frame buffer may be physical memory on-board the display or in the host computer.

Alternatively, a graphics card installed in a personal computer might house the frame buffer.

Graphics Display Devices

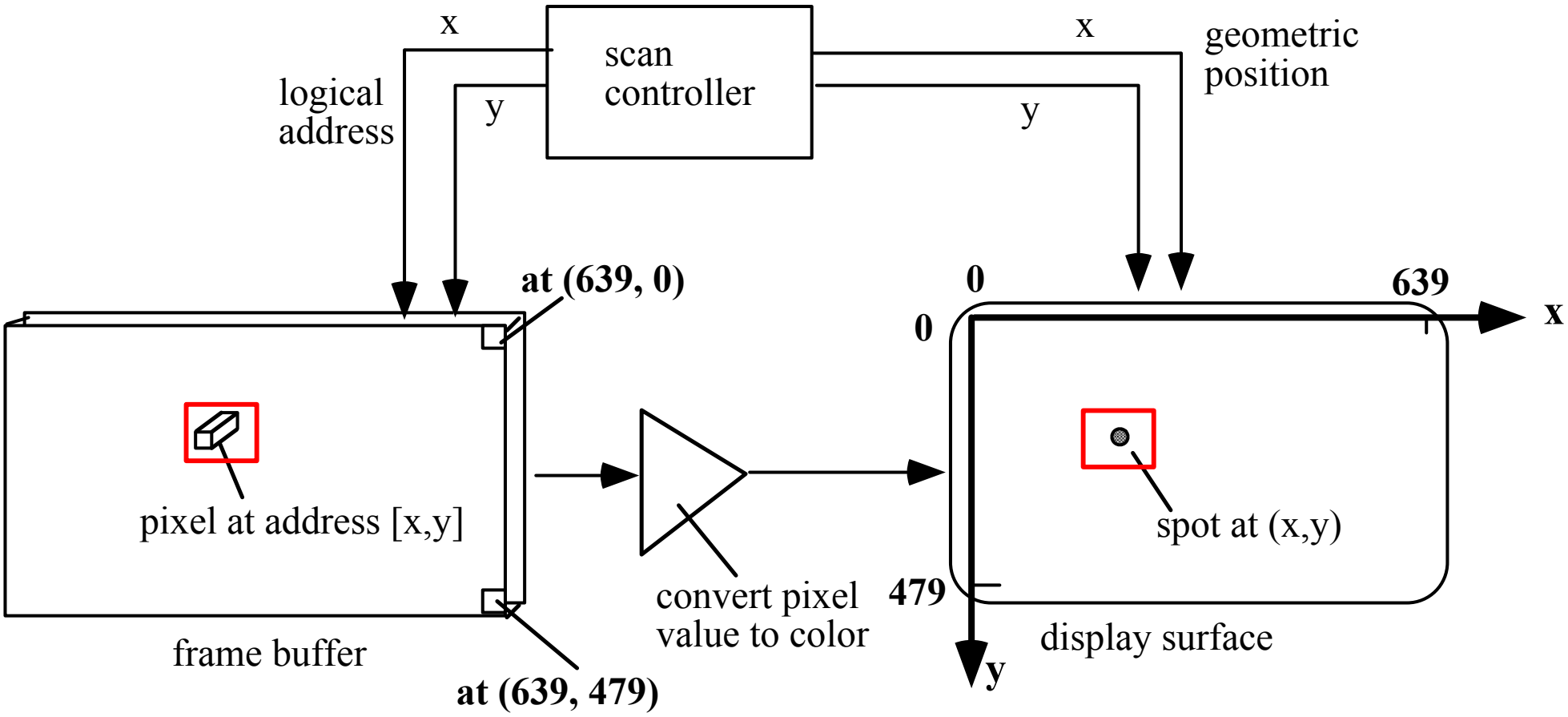


Each instruction of the graphics program (stored in system memory) is executed by the central processing unit (CPU), storing an appropriate value for each **pixel** into the frame buffer (refresh buffer).

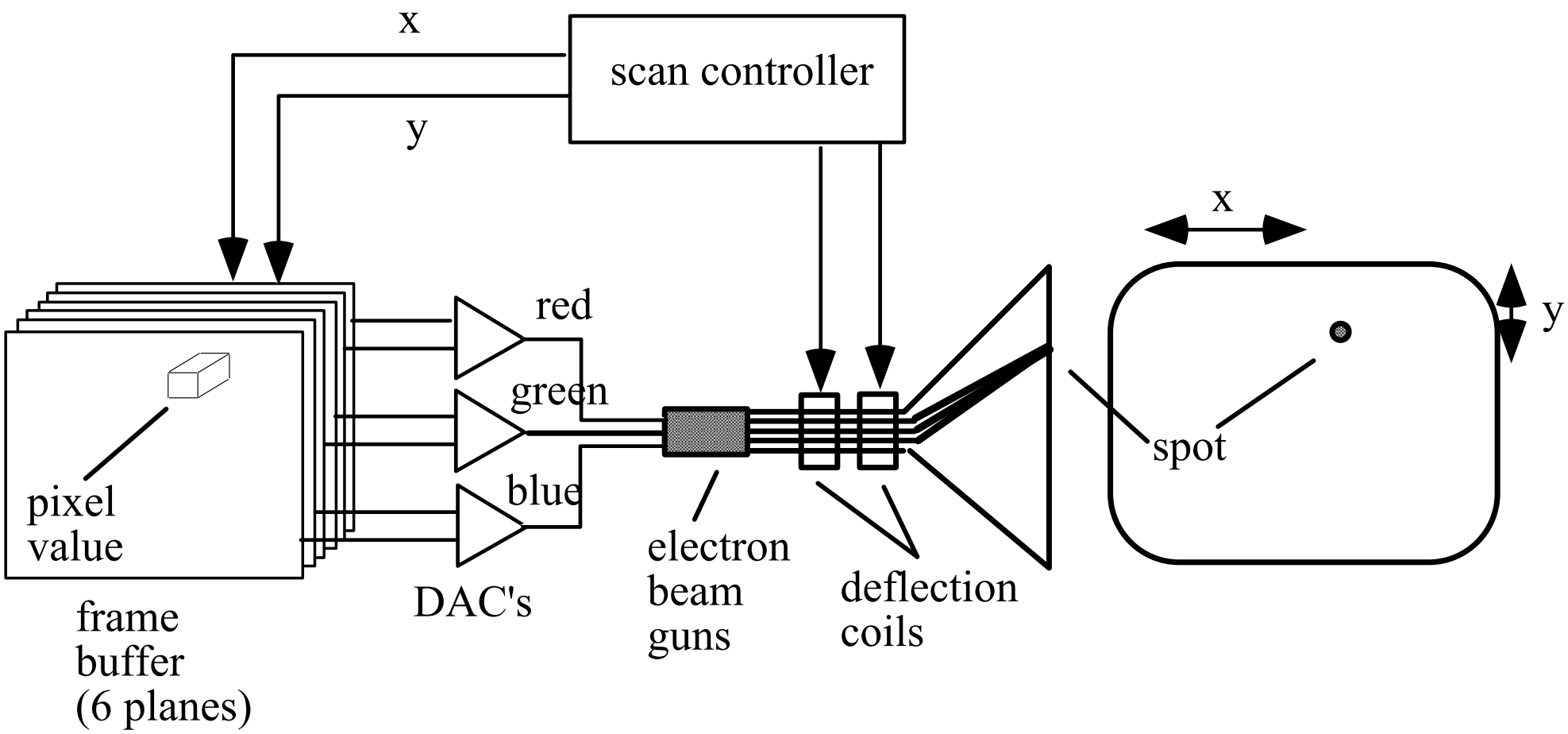
A **scan controller** (**video controller**) (not under program control) causes the frame buffer to send each **pixel** through a **converter** to the appropriate physical location on the display surface.

The converter takes a **pixel** value such as **01001011** and **converts** it to the corresponding color value quantity that produces a spot of color on the display.

Graphics Display Device Operation



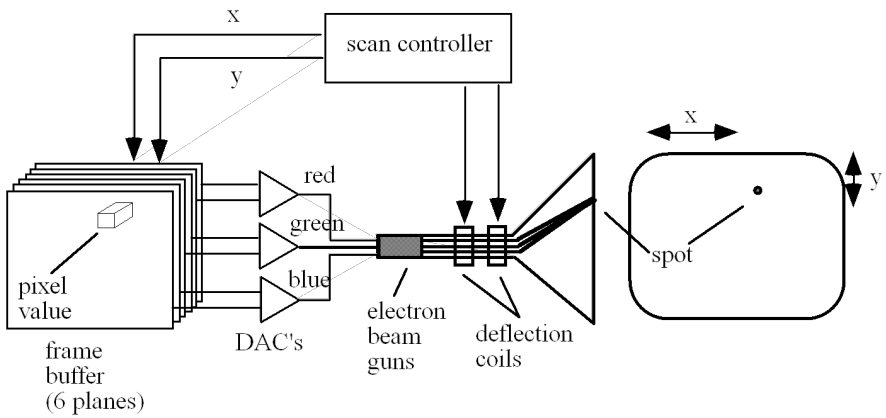
Color CRT


x
y
scan controller
x
y
red
green
blue
pixel
value
frame
buffer
(6 planes)
DAC's
electron
beam
guns
deflection
coils
spot

Video Monitor Operation

The digital frame buffer value is converted to an analog voltage for each of R, G, and B by the DAC. Electron guns for each color are deflected to the appropriate screen location.

The process is repeated 60 times each second to prevent flicker. (60 Hz)



Pipeline Architectures
Geometric Pipeline

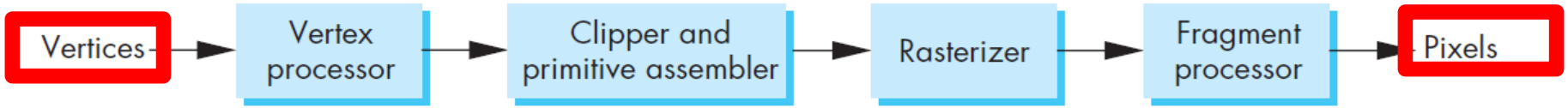


FIGURE 1.31 Geometric pipeline.

We start with a set of objects. Each object comprises a set of graphical primitives.

Each primitive comprises a set of vertices. We can think of the collection of primitive types and vertices as defining the **geometry** of the scene.

In a complex scene, there may be thousands—even millions—of **vertices** that define the objects.

We must process all these **vertices** in a similar manner to form an image in the **frame buffer**.

Pipeline Architectures
Geometric Pipeline

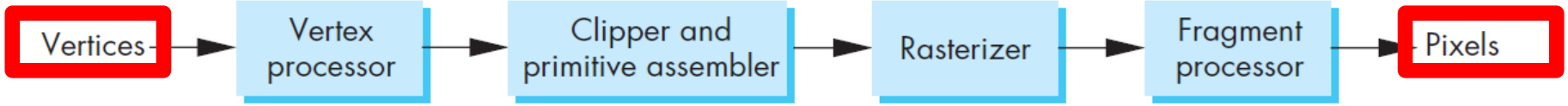


FIGURE 1.31 Geometric pipeline.

If we think in terms of processing the geometry of our **objects** to obtain an **image** these are the major steps in the imaging process:

1. Vertex processing
2. Clipping and primitive assembly
3. Rasterization
4. Fragment processing

Pipeline Architectures
Geometric Pipeline

Processing the geometry of our objects to obtain an image

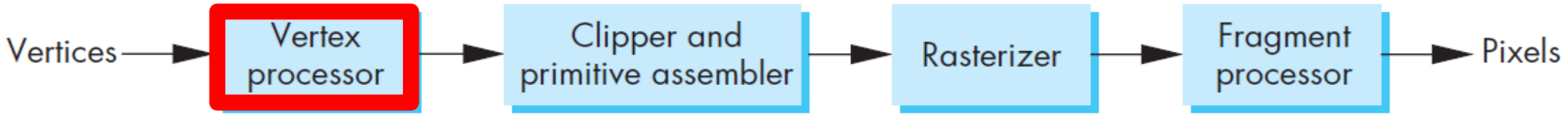


FIGURE 1.31 Geometric pipeline.

1. Vertex processing

   Each **object** comprises a set of graphical primitives.

   Each primitive comprises a set of **vertices**.

Pipeline Architectures
Geometric Pipeline

Processing the geometry of our objects to obtain an image

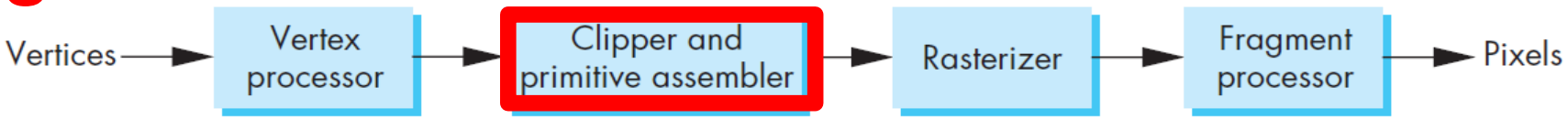


FIGURE 1.31 Geometric pipeline.

2. Clipping and **primitive assembly** (lines, polygons)

No imaging system can see the whole world at once.

Clipping must be done on a primitive by primitive basis rather than on a **vertex** by **vertex** basis.

Output of this stage is a set of primitives whose projections can appear in the image.

Pipeline Architectures
Geometric Pipeline

Processing the geometry of our objects to obtain an image

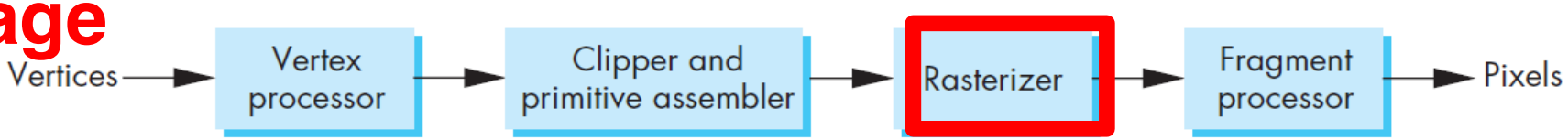


FIGURE 1.31 Geometric pipeline.

3. **Rasterization** (Scan-Conversion)

The primitives that emerge from the clipper are still represented in terms of their **vertices** and must be further processed to generate **pixels** in the **frame buffer** (for example, if three **vertices** specify a triangle filled with a solid color, the **rasterizer** must determine which **pixels** in the **frame buffer** are inside the polygon).

The output of the **rasterizer** is a set of **fragments** for each primitive.

Pipeline Architectures
Geometric Pipeline

Processing the geometry of our objects to obtain an image

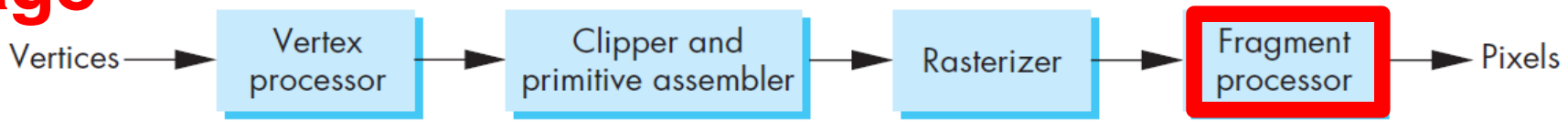
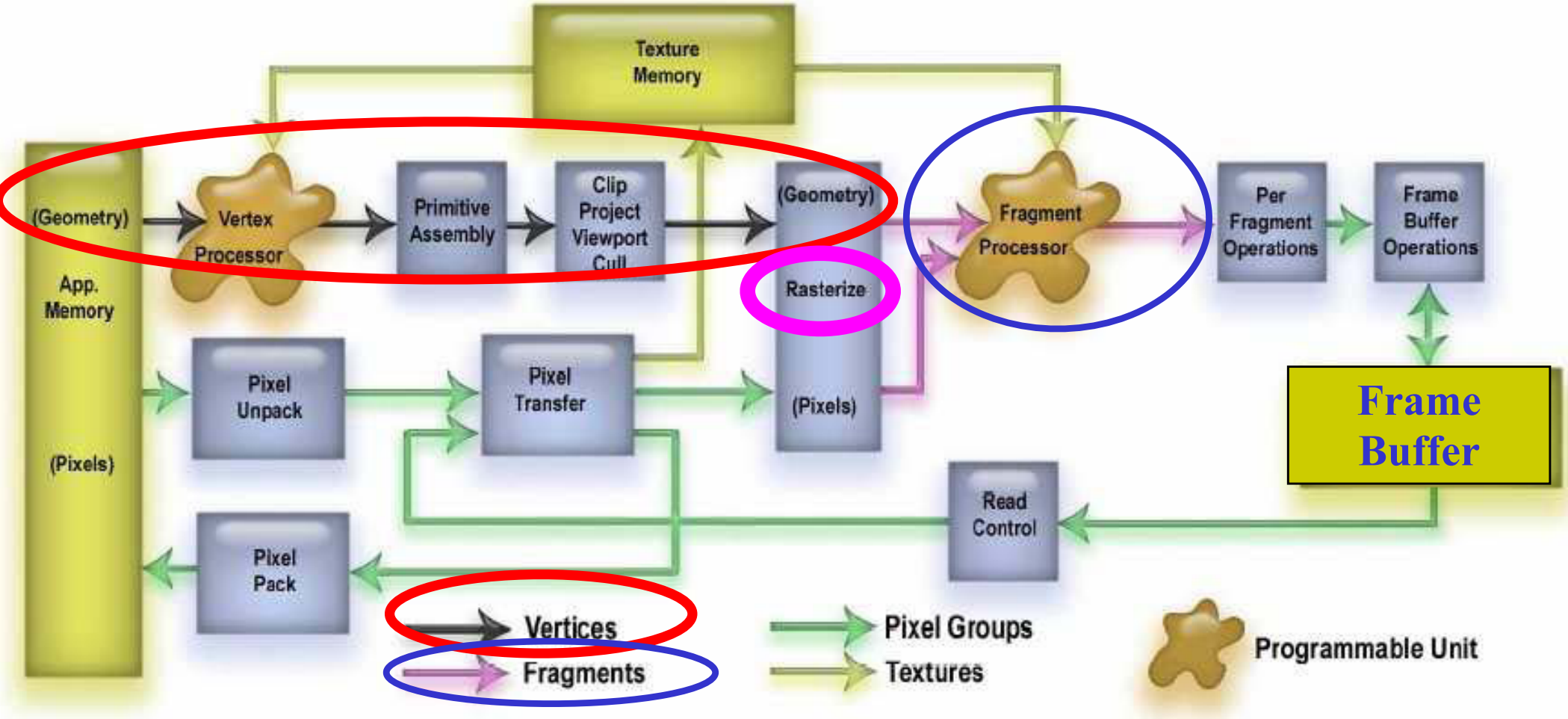


FIGURE 1.31 Geometric pipeline.

4. **Fragment** processing

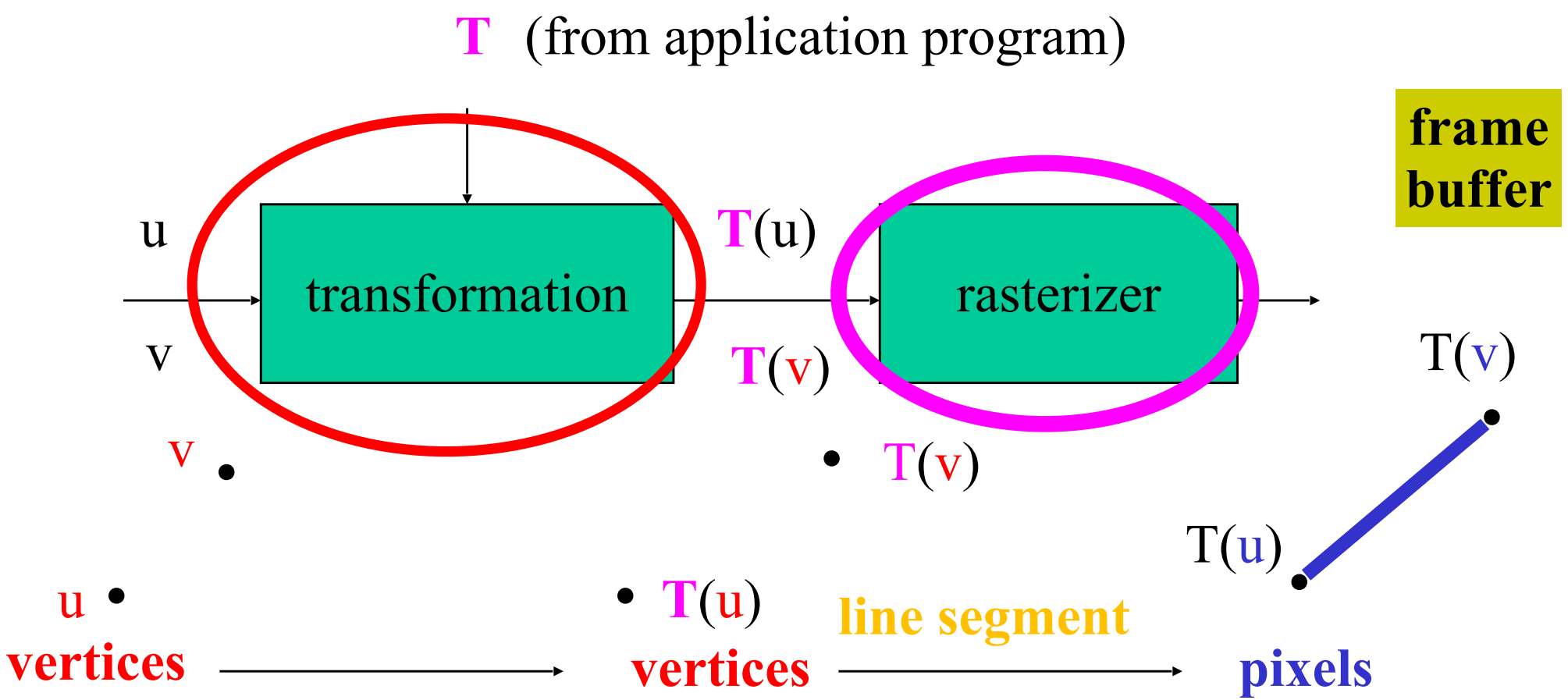
The final block in our pipeline takes in the **fragments** generated by the **rasterizer** and updates the **pixels** in the **frame buffer**.

OpenGL Graphics Pipeline

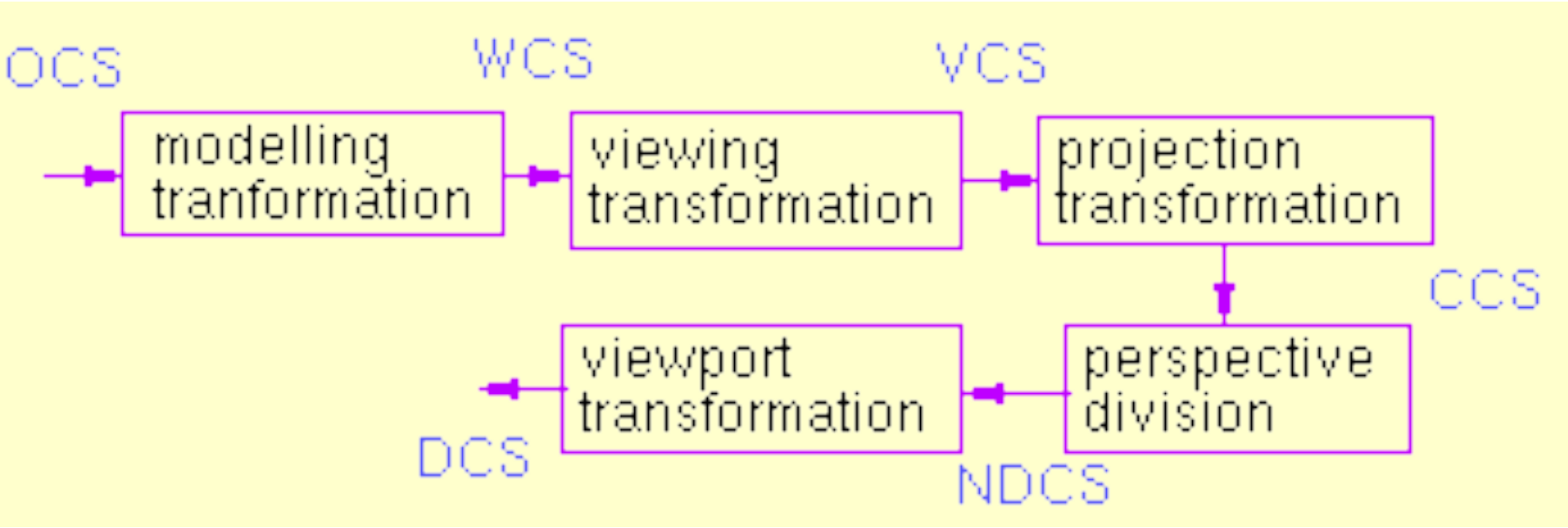


Pipeline Implementation

We need only to transform the homogeneous-coordinate representation of the endpoints of a line segment to determine completely a transformed line. Thus, we can implement our graphics systems as a pipeline that passes endpoints through affine transformation units and generates the interior points at the rasterization stage.

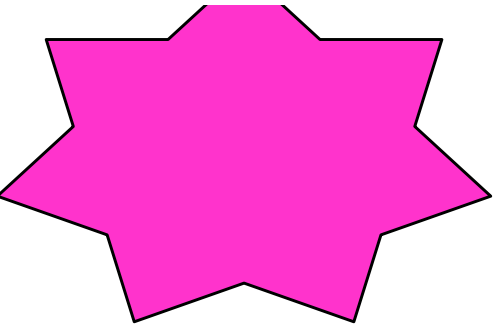


Stages of Vertex Transformations



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

Computer Graphics – Top-Down Example



LECTURE 0 CLASS PARTICIPATION

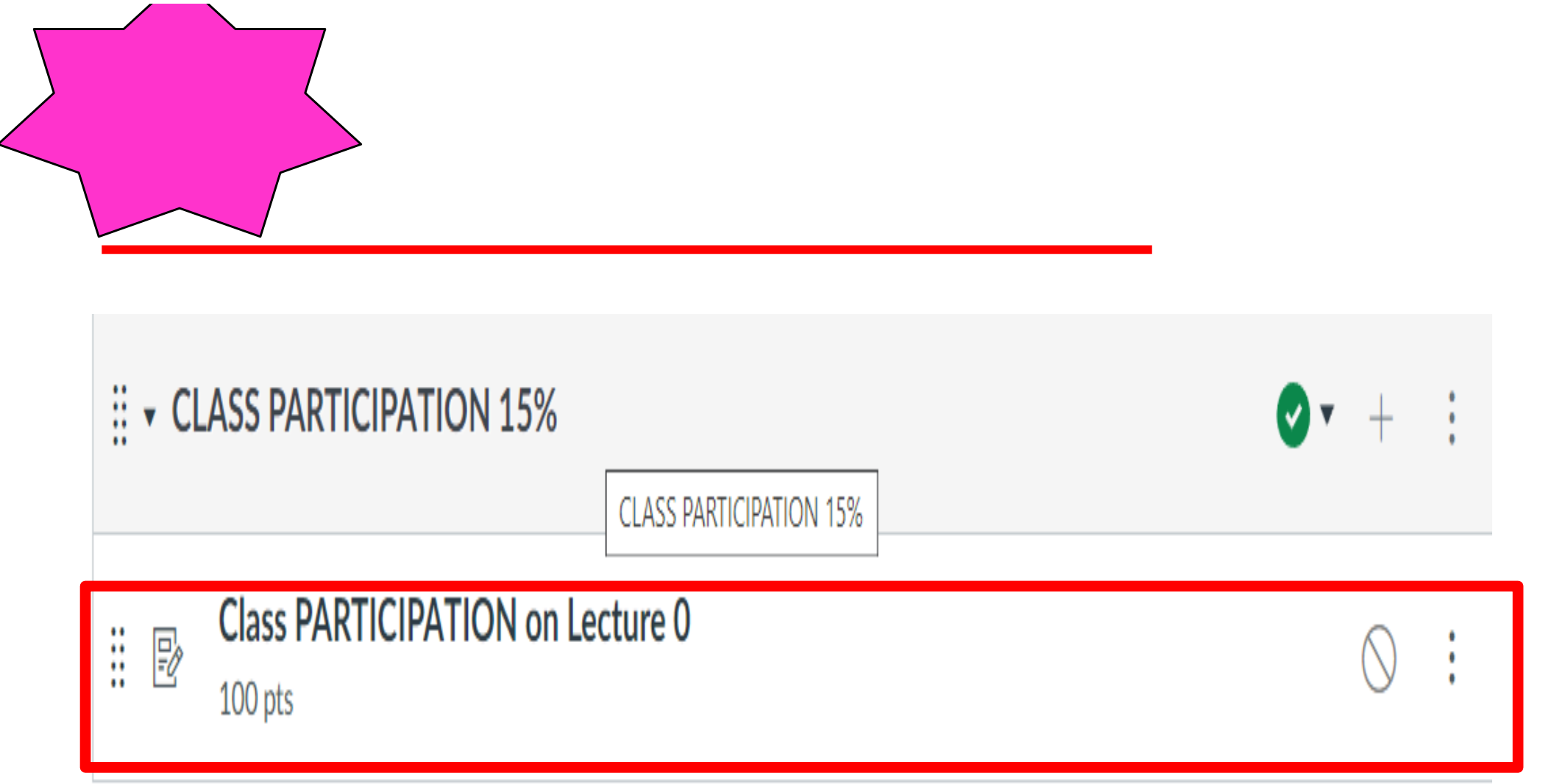
VH, publish folder LECTURE 0 CLASS PARTICIPATION

Class Participation on Lecture 0.doc

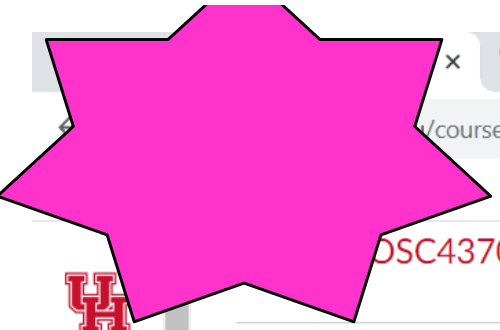
VH, publish document Class Participation on Lecture 0.doc

Lecture0.zip

VH, publish zip Lecture0.zip



VH, publish Assignment Class PARTICIPATION on Lecture 0



How to Log | Home | Mic | Mail - Hilfo | COSC 4370 | Mail - Hilfo | Links to ppt | Index of /~ | Int

/courses/4717/assignments/117124

OSC4370 23469 - Inter... > Assignments > Class PARTICIPATION on ... Stu

Account
Dashboard
Courses
Calendar
Inbox
History
Commons
Studio

H 2023 Fall 1

Home
Announcements
Assignments
Discussions
Grades
People
Pages
Files
Syllabus
Outcomes
Rubrics
Quizzes
Modules
Collaborations

Class PARTICIPATION on Lecture 0

Publish | Edit

Download and complete this word document.

Class Participation on Lecture 0.doc

Lecture0.zip

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.

Points 100

Submitting a file upload

Download Class Participation on Lecture 0.doc

Name: _____

Total score: ☐

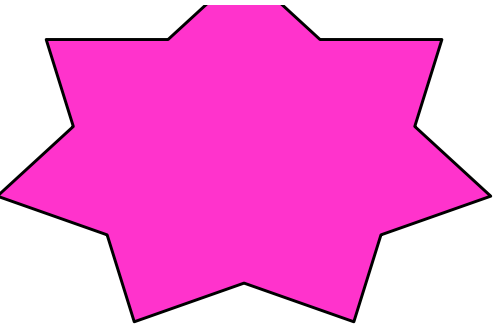
Class PARTICIPATION on Lecture 0.doc **ANSWER SHEET**

(Out of 100 points, 20 points each. Please record your own total score!)

(Attach as score.doc!)

1. Download **Lecture0.zip** from CANVAS. Unzip, open .sln with Visual Studio 2019. Run and Insert Print Screen **HERE** (20 points)

Class Participation 1!



Class PARTICIPATION on Lecture 0

Publish | Edit | ⋮

Download and complete this word document.

Class Participation on Lecture 0.doc

Lecture0.zip

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

Warning:

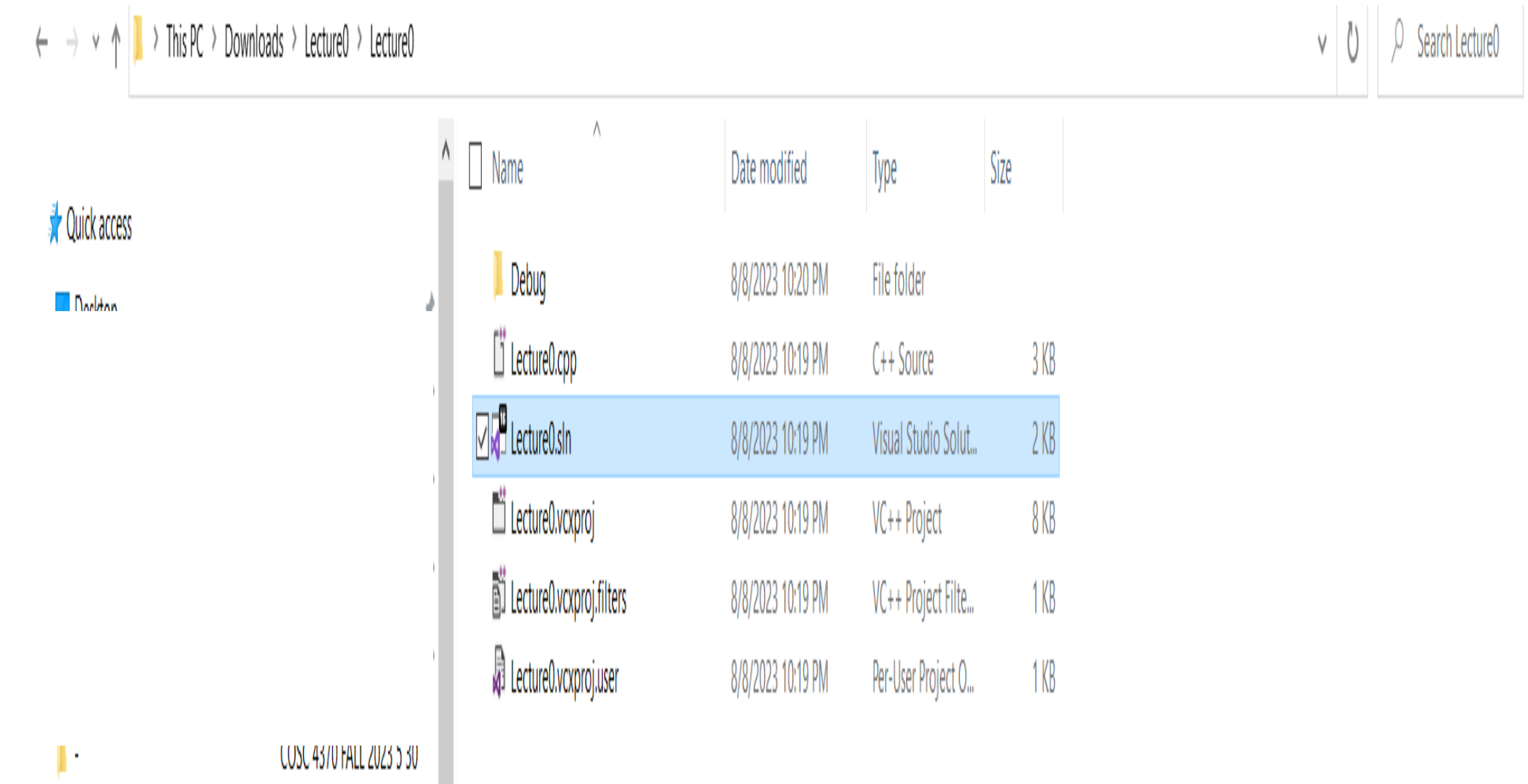
TA, at random, will inspect the Uploaded document.

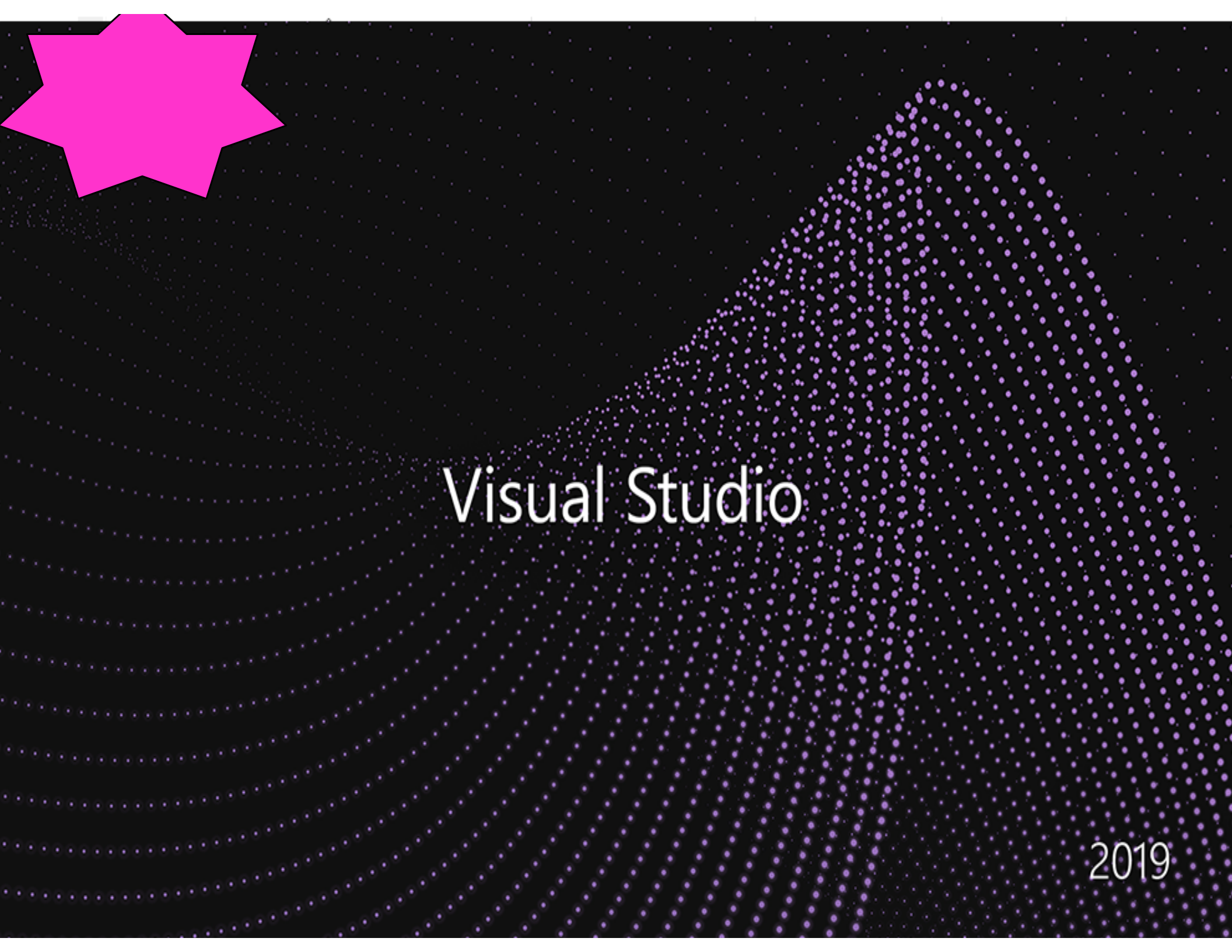
If you score is not honestly entered you will get a zero.

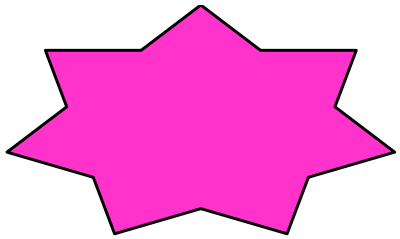
Points	100
Submitting	a file upload

Download Lecture0.zip from CANVAS (your Visual Studio folder)
Unzip and open solution

UnZip
Click on Solution



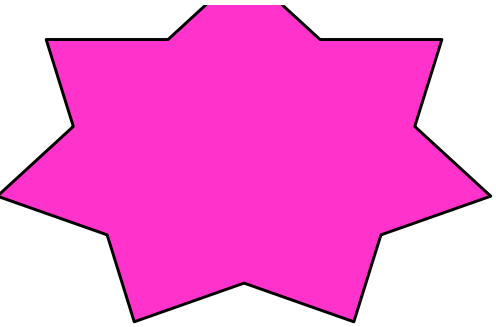

Visual Studio
2019



```
// Last Name, First Name
// Lecture0
// Fall 2023
#include <GL/glut.h>
#include<iostream>
using namespace std;
int rx = 100, ry = 125;
int xCenter = 250, yCenter = 250;

void myinit(void)
{
    glClearColor(1.0, 1.0, 1.0, 0.0);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluOrtho2D(0.0, 640.0, 0.0, 480.0);
}

void setPixel(GLint x, GLint y)
{
    glBegin(GL_POINTS);
    glVertex2i(x, y);
    glEnd();
}
void ellipseMidPoint()
{
    float x = 0;
    float y = ry;
    float p1 = ry * ry - (rx * rx) * ry + (rx * rx) * (0.25);
    float dx = 2 * (ry * ry) * x;
```



Name: _____

Total score: ☐

Class PARTICIPATION on Lecture 0.doc **ANSWER SHEET**

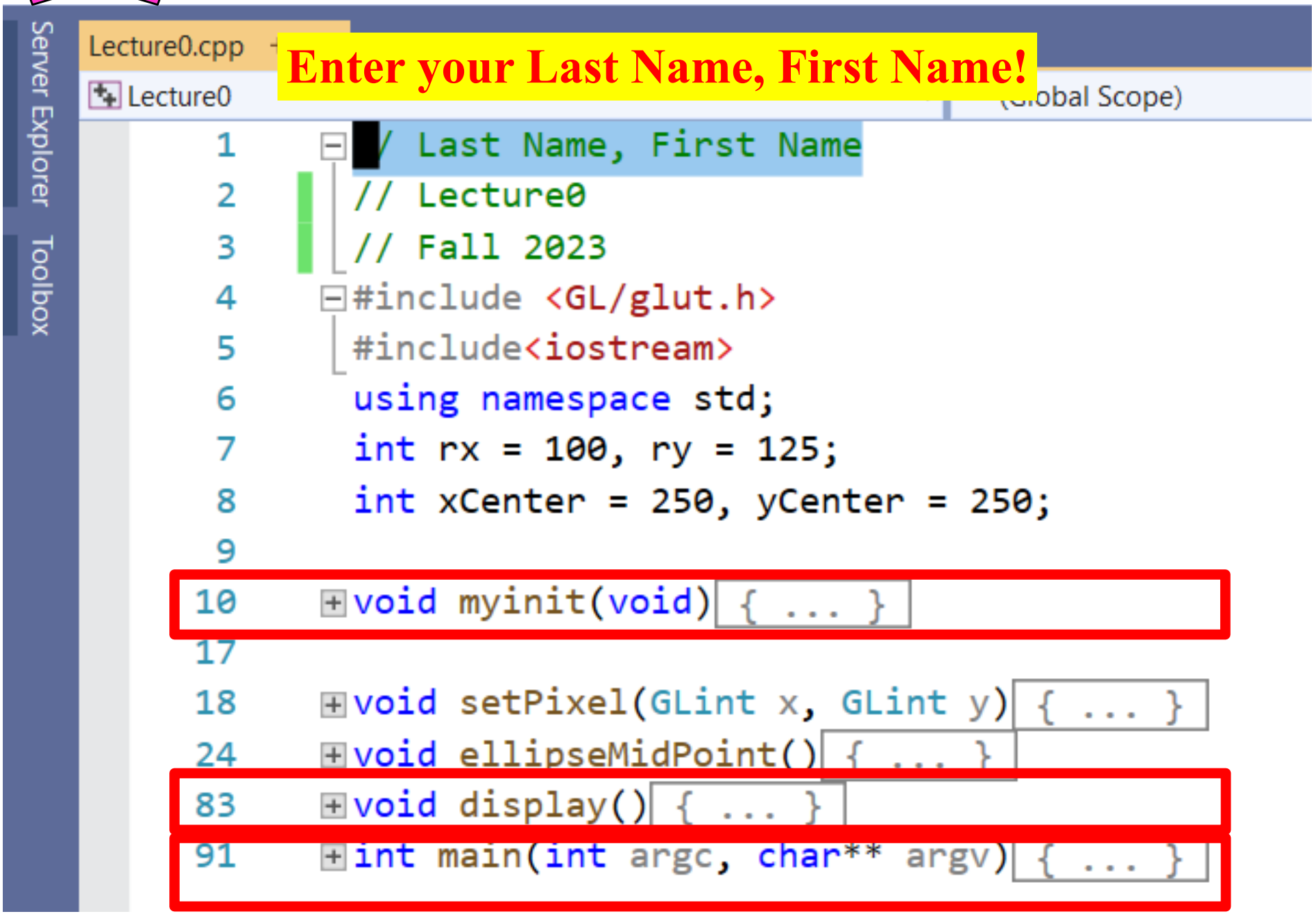
(Out of 100 points, 20 points each. Please record your own total score!)

(Attach as score.doc!)

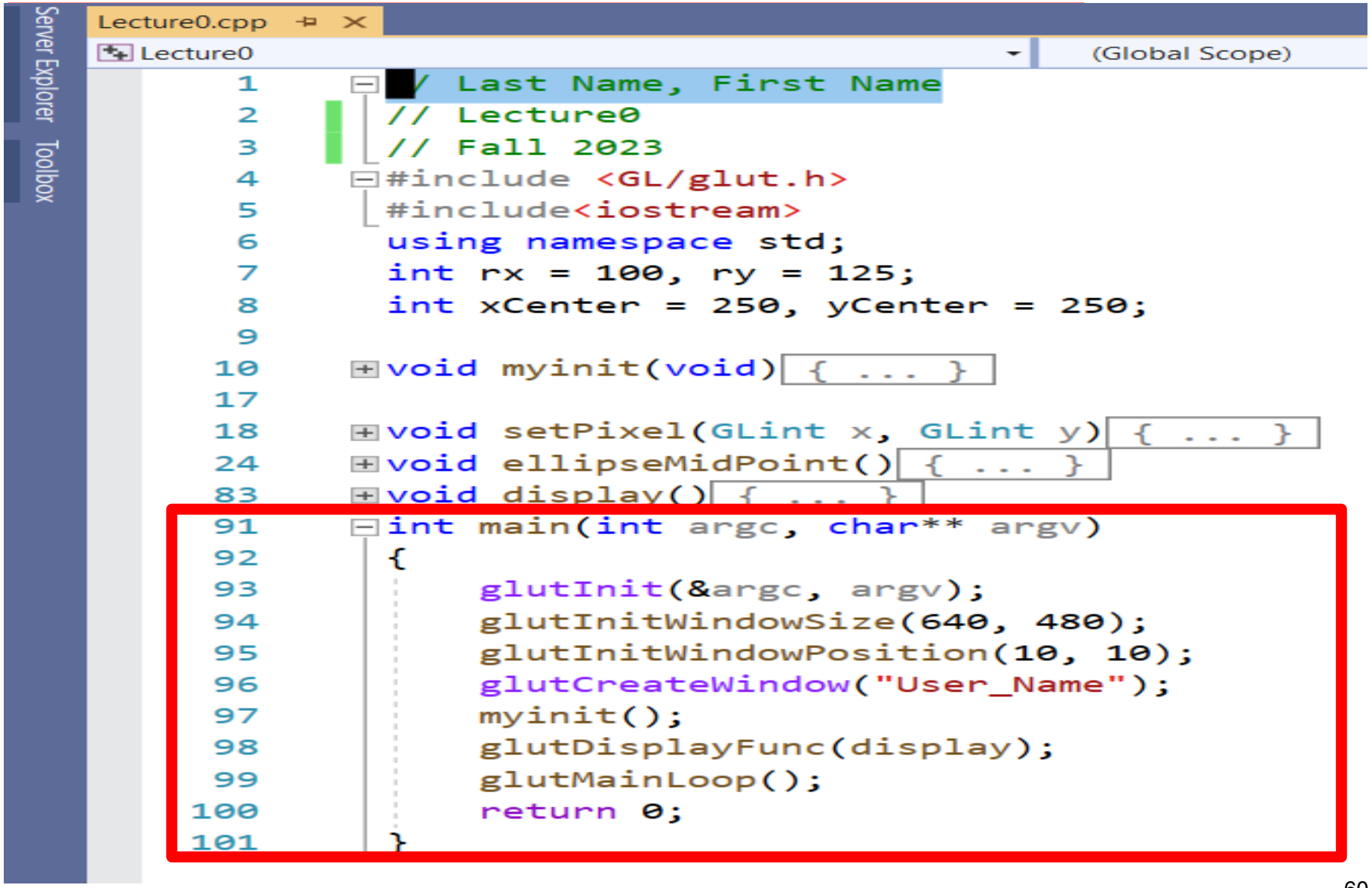
1. Download **Lecture0.zip** from CANVAS. Unzip, open .sln with Visual Studio 2019. Run and Insert Print Screen **HERE** (20 points)

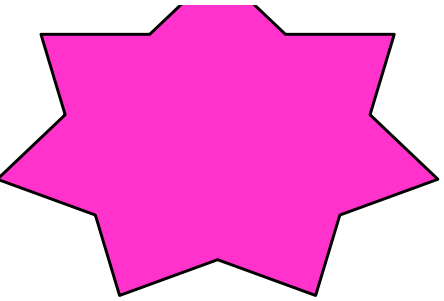
Take print screen of the desktop and insert here!

OpenGL


Server Explorer
Toolbox
Lecture0.cpp
Lecture0
(Global Scope)
Enter your Last Name, First Name!
1 // Last Name, First Name
2 // Lecture0
3 // Fall 2023
4 #include <GL/glut.h>
5 #include<iostream>
6 using namespace std;
7 int rx = 100, ry = 125;
8 int xCenter = 250, yCenter = 250;
9
10 void myinit(void) { ... }
17
18 void setPixel(GLint x, GLint y) { ... }
24 void ellipseMidPoint() { ... }
83 void display() { ... }
91 int main(int argc, char** argv) { ... }

OpenGL


Server Explorer
Toolbox
Lecture0.cpp
Lecture0
(Global Scope)
1 // Last Name, First Name
2 // Lecture0
3 // Fall 2023
4 #include <GL/glut.h>
5 #include<iostream>
6 using namespace std;
7 int rx = 100, ry = 125;
8 int xCenter = 250, yCenter = 250;
9
10 void myinit(void) { ... }
17
18 void setPixel(GLint x, GLint y) { ... }
24 void ellipseMidPoint() { ... }
83 void display() { ... }
91 int main(int argc, char** argv)
92 {
93 glutInit(&argc, argv);
94 glutInitWindowSize(640, 480);
95 glutInitWindowPosition(10, 10);
96 glutCreateWindow("User_Name");
97 myinit();
98 glutDisplayFunc(display);
99 glutMainLoop();
100 return 0;
101 }



OpenGL

Server Explorer Toolbox

Lecture0.cpp

Lecture0 (Global Scope)

```
1   // Last Name, First Name
2   // Lecture0
3   // Fall 2023
4   #include <GL/glut.h>
5   #include<iostream>
6   using namespace std;
7   int rx = 100, ry = 125;
8   int xCenter = 250, yCenter = 250;
9
10  void myinit(void)
11  {
12      glClearColor(1.0, 1.0, 1.0, 0.0);
13      glMatrixMode(GL_PROJECTION);
14      glLoadIdentity();
15      gluOrtho2D(0.0, 640.0, 0.0, 480.0);
16  }
17
18  void setPixel(GLint x, GLint y) { ... }
24  void ellipseMidPoint() { ... }
83  void display()
84  {
85      glClear(GL_COLOR_BUFFER_BIT);
86      glColor3f(1.0, 0.0, 0.0);
87      glPointSize(2.0);
88      ellipseMidPoint();
89      glFlush();
90  }
91  int main(int argc, char** argv) { ... }
```

Abstractions

GLUT

- Windowing toolkit (key, mouse handler, window events)

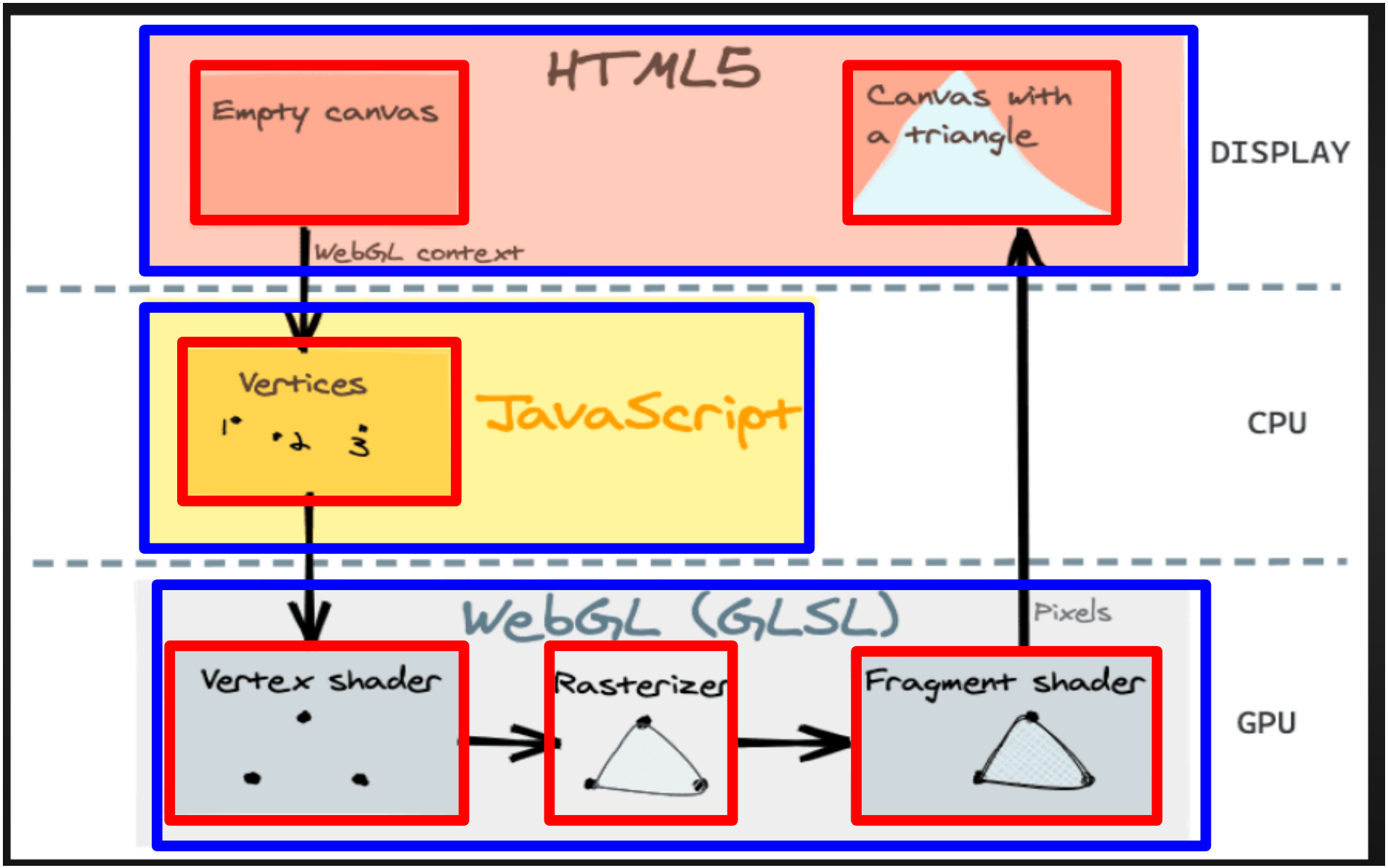
GLU

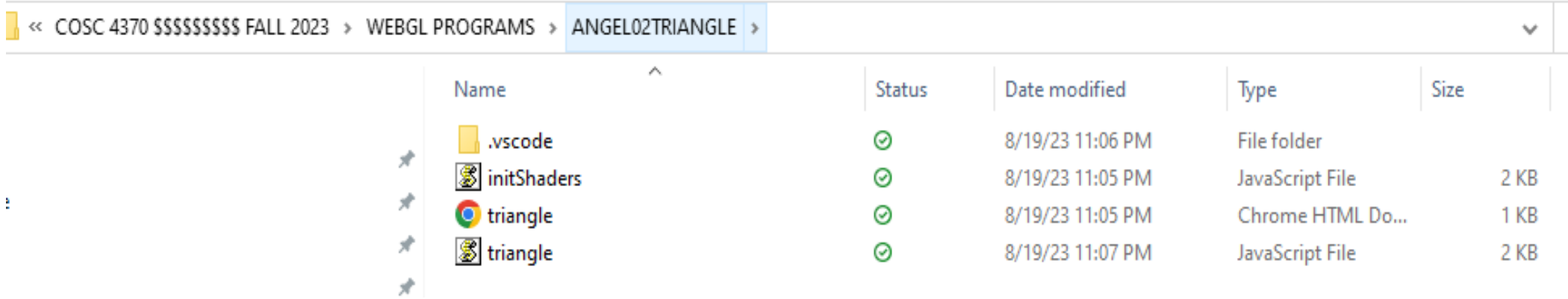
- Viewing –perspective/orthographic
- Image scaling, polygon tessellation
- Sphere, cylinders, quadratic surfaces

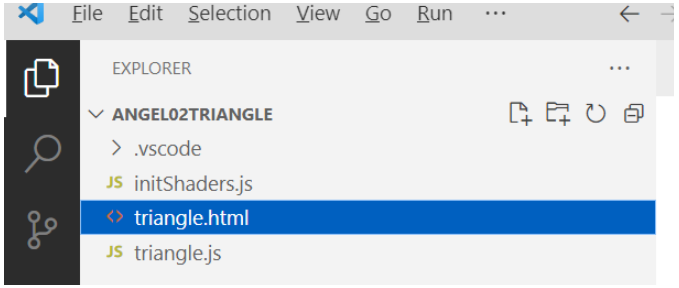
GL

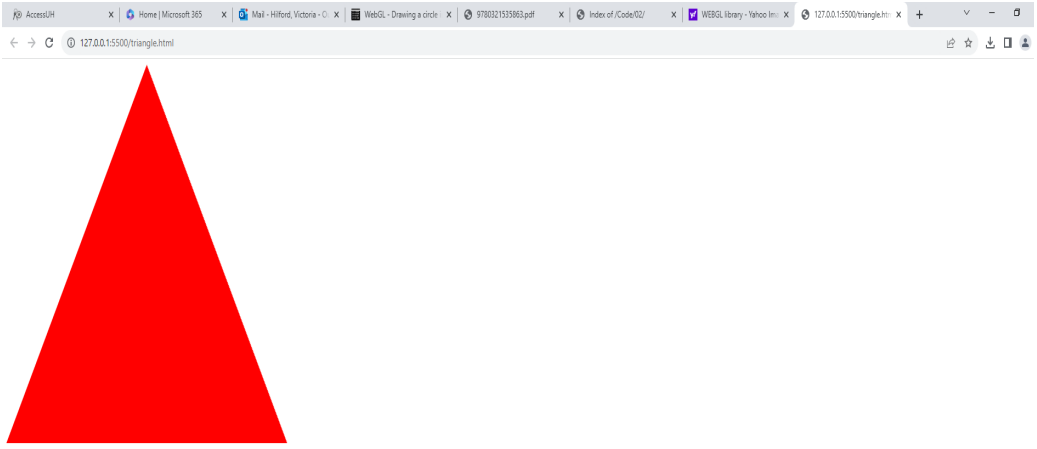
- Primitives - points, line, polygons
- Shading and Colour
- Translation, rotation, scaling
- Viewing, Clipping, Texture
- Hidden surface removal

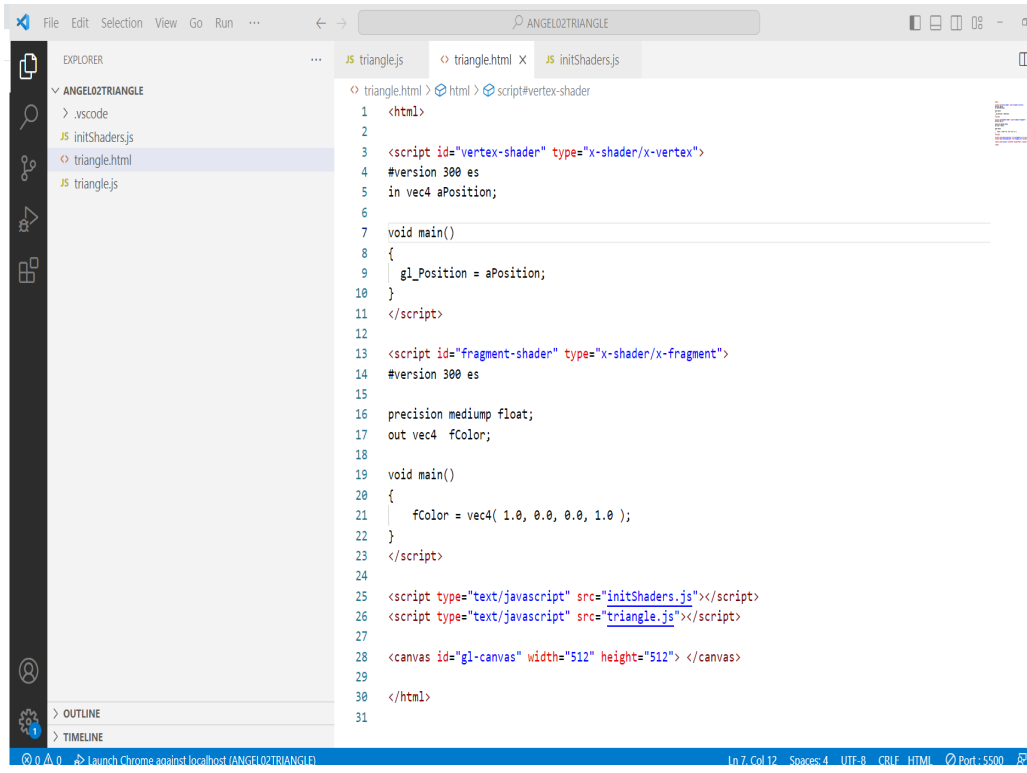
WebGL

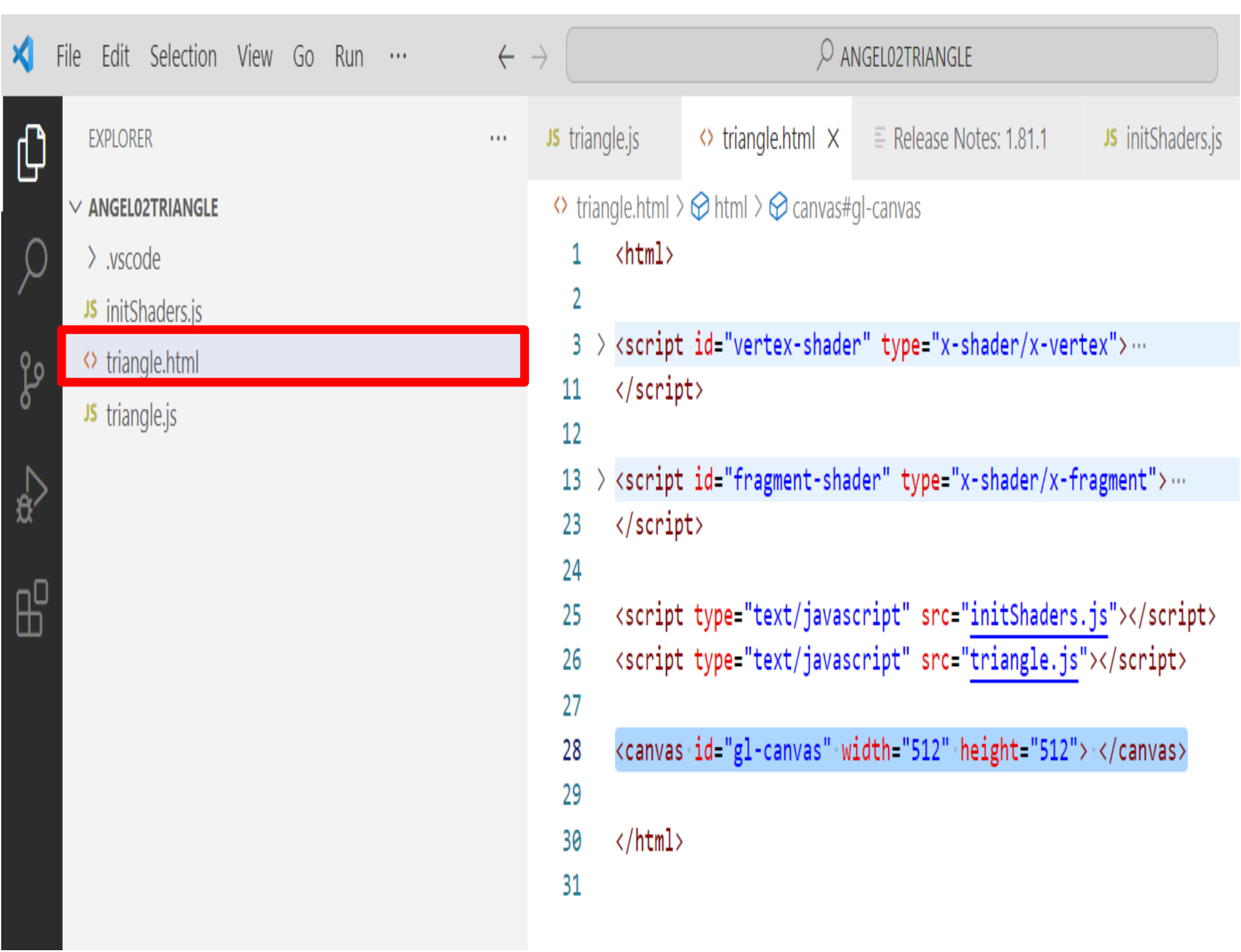

HTML5
Empty canvas
Canvas with a triangle
DISPLAY
WebGL context
Vertices
1
2
3
JavaScript
CPU
WebGL (GLSL)
Pixels
Vertex shader
Rasterizer
Fragment shader
GPU

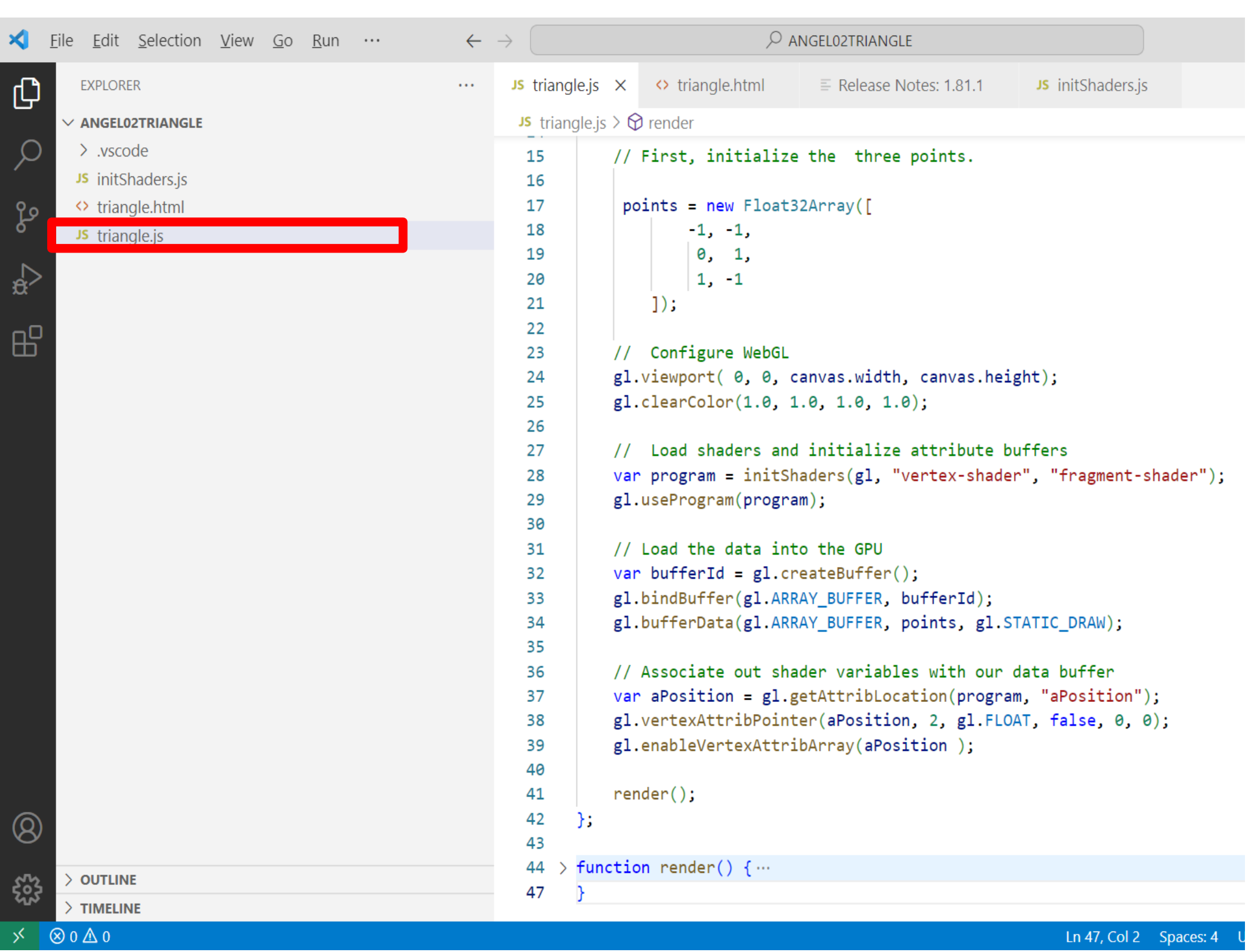

« COSC 4370 $$$$$$$$$ FALL 2023 › WEBGL PROGRAMS › ANGEL02TRIANGLE ›
Name
Status
Date modified
Type
Size
.vscode
8/19/23 11:06 PM
File folder
initShaders
8/19/23 11:05 PM
JavaScript File
2 KB
triangle
8/19/23 11:05 PM
Chrome HTML Do...
1 KB
triangle
8/19/23 11:07 PM
JavaScript File
2 KB

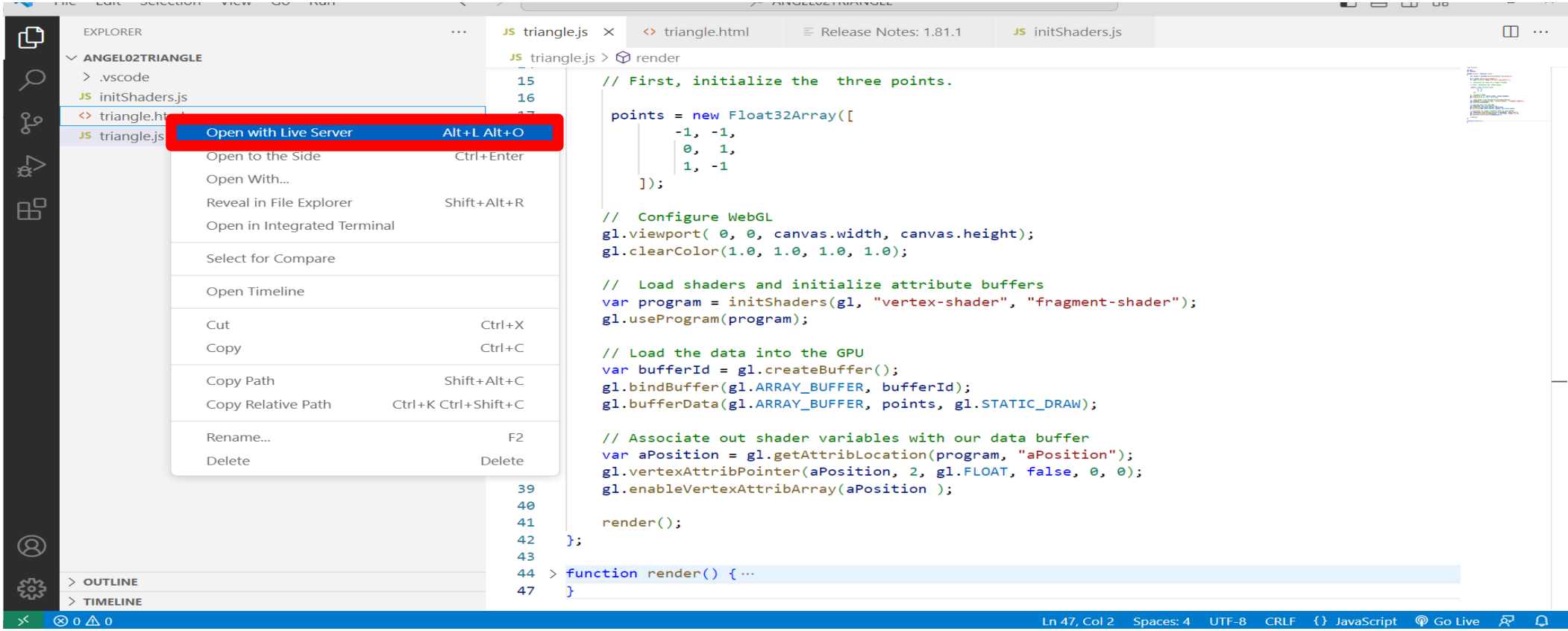

File Edit Selection View Go Run ···
EXPLORER
ANGEL02TRIANGLE
.vscode
initShaders.js
triangle.html
triangle.js

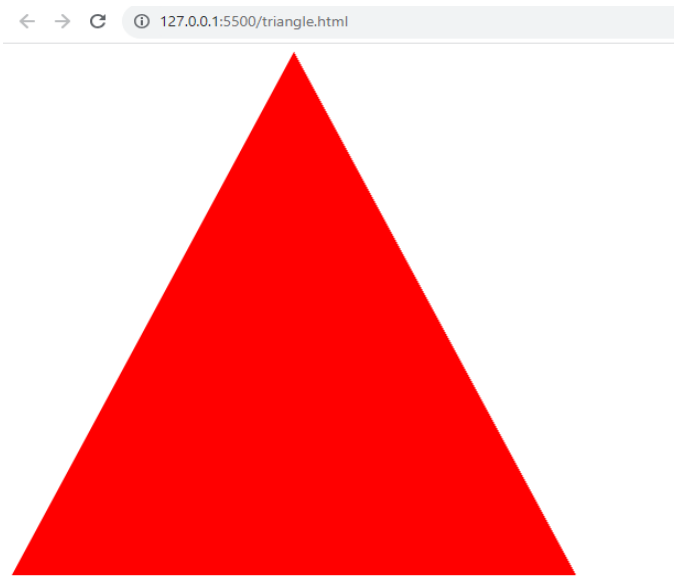

127.0.0.1:5500/triangle.html


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


File Edit Selection View Go Run ···
ANGEL02TRIANGLE
EXPLORER
triangle.js
triangle.html
Release Notes: 1.81.1
initShaders.js
ANGEL02TRIANGLE
.vscode
initShaders.js
triangle.html
triangle.js
triangle.html > html > canvas#gl-canvas
1 <html>
2
3 <script id="vertex-shader" type="x-shader/x-vertex">···
11 </script>
12
13 <script id="fragment-shader" type="x-shader/x-fragment">···
23 </script>
24
25 <script type="text/javascript" src="initShaders.js"></script>
26 <script type="text/javascript" src="triangle.js"></script>
27
28 <canvas id="gl-canvas" width="512" height="512"> </canvas>
29
30 </html>
31


File Edit Selection View Go Run ···
ANGEL02TRIANGLE
EXPLORER
ANGEL02TRIANGLE
.vscode
initShaders.js
triangle.html
triangle.js
triangle.js
triangle.html
Release Notes: 1.81.1
initShaders.js
triangle.js > render
// First, initialize the three points.
points = new Float32Array([
-1, -1,
0, 1,
1, -1
]);
// Configure WebGL
gl.viewport(0, 0, canvas.width, canvas.height);
gl.clearColor(1.0, 1.0, 1.0, 1.0);
// Load shaders and initialize attribute buffers
var program = initShaders(gl, "vertex-shader", "fragment-shader");
gl.useProgram(program);
// Load the data into the GPU
var bufferId = gl.createBuffer();
gl.bindBuffer(gl.ARRAY_BUFFER, bufferId);
gl.bufferData(gl.ARRAY_BUFFER, points, gl.STATIC_DRAW);
// Associate out shader variables with our data buffer
var aPosition = gl.getAttribLocation(program, "aPosition");
gl.vertexAttribPointer(aPosition, 2, gl.FLOAT, false, 0, 0);
gl.enableVertexAttribArray(aPosition);
render();
};
function render() {···
}
OUTLINE
TIMELINE
0 0
Ln 47, Col 2
Spaces: 4


EXPLORER
ANGEL02TRIANGLE
.vscode
initShaders.js
triangle.html
triangle.js
Open with Live Server Alt+L Alt+O
Open to the Side Ctrl+Enter
Open With...
Reveal in File Explorer Shift+Alt+R
Open in Integrated Terminal
Select for Compare
Open Timeline
Cut Ctrl+X
Copy Ctrl+C
Copy Path Shift+Alt+C
Copy Relative Path Ctrl+K Ctrl+Shift+C
Rename... F2
Delete Delete
OUTLINE
TIMELINE
triangle.js
triangle.html
Release Notes: 1.81.1
initShaders.js
triangle.js > render
// First, initialize the three points.
points = new Float32Array([
-1, -1,
0, 1,
1, -1
]);
// Configure WebGL
gl.viewport(0, 0, canvas.width, canvas.height);
gl.clearColor(1.0, 1.0, 1.0, 1.0);
// Load shaders and initialize attribute buffers
var program = initShaders(gl, "vertex-shader", "fragment-shader");
gl.useProgram(program);
// Load the data into the GPU
var bufferId = gl.createBuffer();
gl.bindBuffer(gl.ARRAY_BUFFER, bufferId);
gl.bufferData(gl.ARRAY_BUFFER, points, gl.STATIC_DRAW);
// Associate out shader variables with our data buffer
var aPosition = gl.getAttribLocation(program, "aPosition");
gl.vertexAttribPointer(aPosition, 2, gl.FLOAT, false, 0, 0);
gl.enableVertexAttribArray(aPosition);
render();
};
function render() {
}
Ln 47, Col 2
Spaces: 4
UTF-8
CRLF
JavaScript
Go Live


127.0.0.1:5500/triangle.html

COSC 4370 – Computer Graphics

Lecture 0 - REVIEW

Geometry

Chapter 4.1- 4.2

Representation

Chapter 4.3

Trigonometry

Linear Algebra - Matrices

Calculus - Derivatives

Geometry
Chapter 4.1- 4.2

Objectives

- Introduce the elements of geometry
  - Scalars
  - Points
  - Vectors
- **Develop mathematical operations among them in a coordinate-free manner**
- Define basic primitives
  - Line segments
  - Polygons

Basic Elements

- Geometry is the study of the relationships among **objects** in an n-dimensional space

In computer graphics, we are interested in **objects** that exist in 3-dimensions

Coordinate-Free Geometry

- When we learned simple geometry, most of us started with a **Cartesian approach**

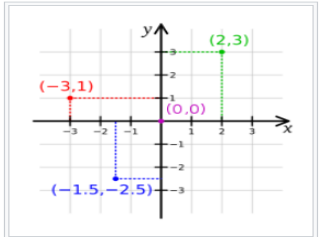


Illustration of a Cartesian coordinate plane. Four points are marked and labeled with their coordinates: (2, 3) in green, (−3, 1) in red, (−1.5, −2.5) in blue, and the origin (0, 0) in purple.

Points were at locations in space P = (x,y,z)

We derived results by algebraic manipulations involving these coordinates

- This approach was nonphysical

Physically, points exist regardless of the location of an arbitrary **coordinate system**

Most geometric results are independent of the **coordinate system**

Scalars

- Need three basic elements in geometry

  Scalars, Vectors, Points

- Scalars can be defined as members of **sets** which can be combined by two operations (addition and multiplication) obeying some fundamental **axioms** (*associativity, commutivity, inverses*)

- Examples include the **real** and **complex** number systems under the ordinary rules with which we are familiar

- Scalars alone **have no geometric properties**

Vectors

Physical definition: a **Vector** is a quantity with two attributes

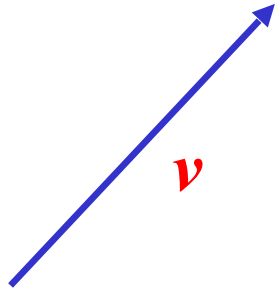
- **Direction**
- **Magnitude**

Examples include

- Force
- Velocity

Directed line segments

- Most important example for graphics
- Can map to other types

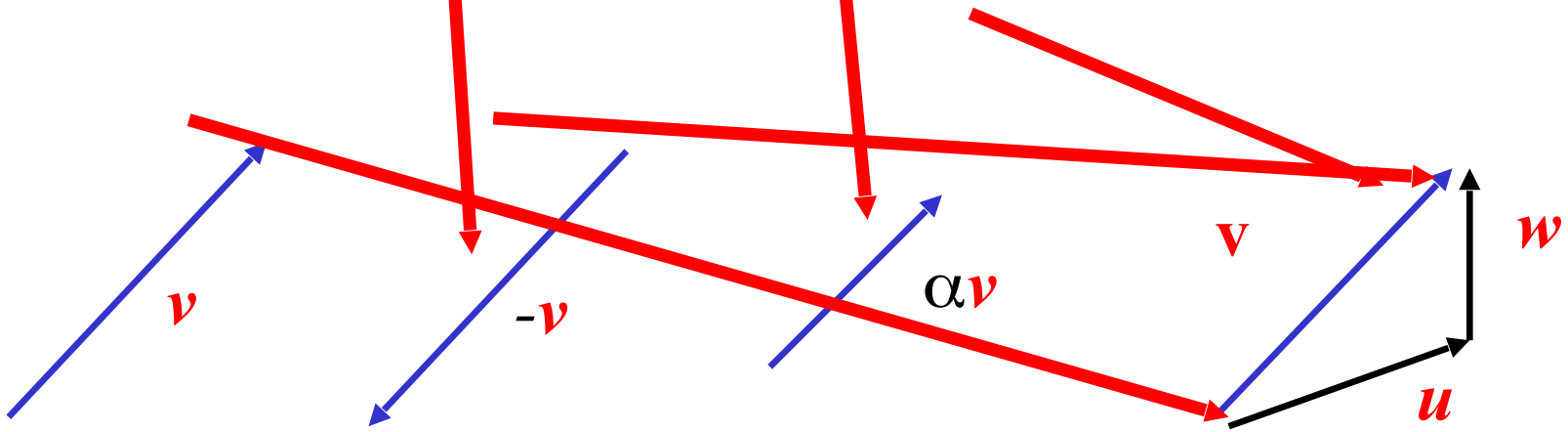


Vector Operations

- Every **Vector** has an **inverse**

  Same **magnitude** but points in opposite **direction**
- Every **Vector** can be **multiplied** by a Scalar
- There is a zero **Vector**

  Zero **magnitude**, undefined **orientation**
- The **sum** of any two **Vectors** is a **Vector**



Linear Vector Spaces

- Mathematical system for manipulating **Vectors**
- **Operations**

  Scalar **Vector** **multiplication** $u = \alpha v$

  Vector **Vector** **addition**: $v = u + w$

- Expressions such as

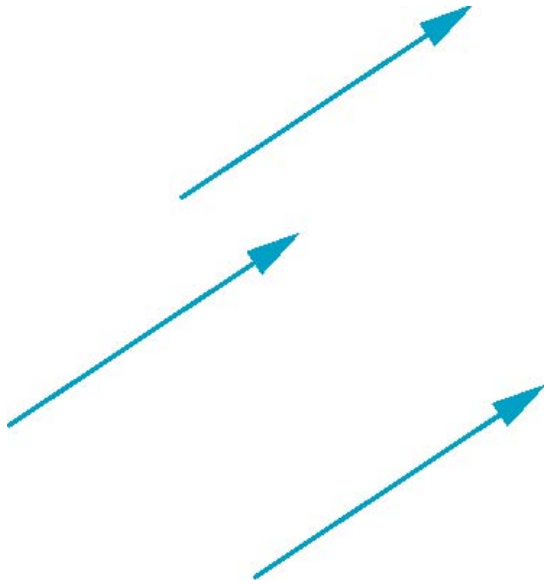
  $v = u + 2w - 3r$

Make sense in a **Vector Space**

Vectors Lack Position

- These **vectors** are identical

  Same **length** (**magnitude**) and **direction**



- **Vector** spaces insufficient for geometry

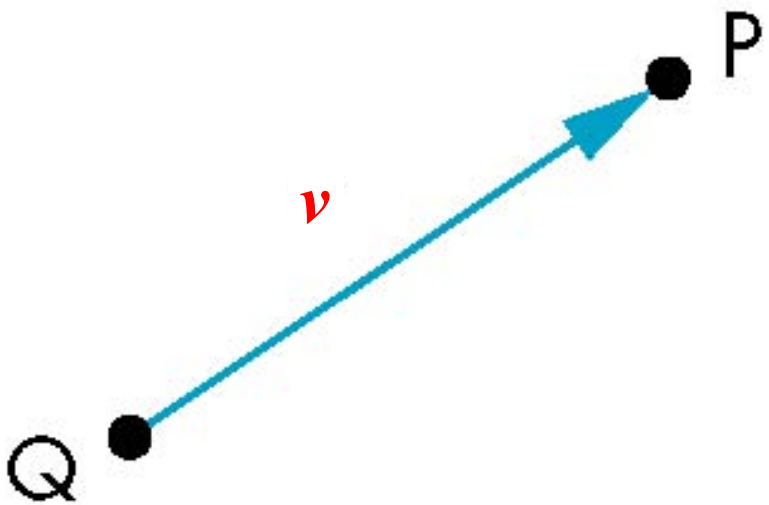
  Need **Points**

Points

- **Location** in space
- Operations allowed between **Points** and **Vectors**

  Point Point subtraction yields a **Vector**

  Equivalent to **Point Vector addition**



$$v = P - Q$$

$$P = Q + v$$

Affine Spaces

- **Point** + **Vector** Space
- **Operations**
  - **Vector Vector** addition
  - **Scalar Vector** multiplication
  - **Point Vector** addition
  - **Scalar Scalar** operations
- For any **Point** define
  - $1 \cdot P = P$
  - $0 \cdot P = \mathbf{0}$ (zero **Vector**)

Affine transformation is a linear mapping method that **preserves points**, **straight lines**, and **planes**.

Sets of parallel lines remain parallel after an **affine transformation**.

Objectives

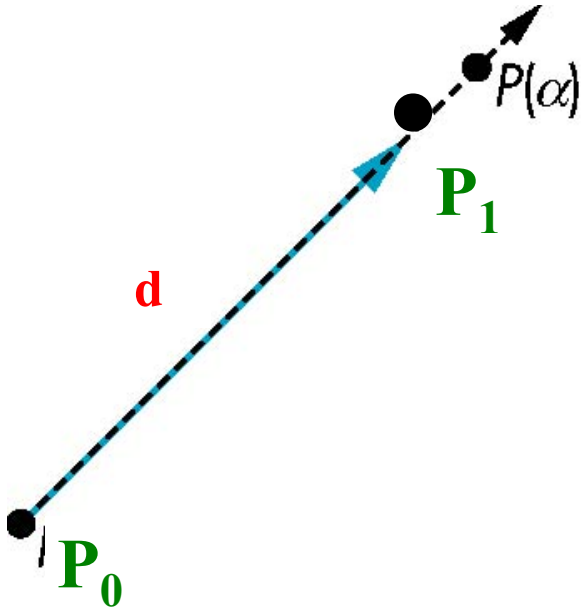
- Introduce the elements of geometry
  - Scalars
  - Vectors
  - Points
- Develop mathematical operations among them in a coordinate-free manner
- Define basic primitives
  - Line segments
  - Polygons

Lines

Consider all **Points** of the form (Parametric Form)

$$P(\alpha) = P_0 + \alpha\, d$$

Set of all Points that pass through P_0 in the direction of the vector d



$$P_0 = (x_0, y_0)$$

$$\begin{aligned} P(\alpha) &= P_0 + \alpha\, d \\ &= P_0 + \alpha\,(P_1 - P_0) \\ &= \alpha\, P_1 + (1 - \alpha)\, P_0 \end{aligned}$$

Line – Explicit, Implicit & Parametric Forms

- **Parametric** form of the line
  - More robust and general than other forms
  - Extends to curves and surfaces

- Two-dimensional forms
  - **Explicit**: $y = mx + b$
  - **Implicit**: $ax + by + c = 0$
  - **Parametric**:

$$x(\alpha) = \alpha x_1 + (1-\alpha)x_0$$
$$y(\alpha) = \alpha y_1 + (1-\alpha)y_0$$

$$P(\alpha) = \mathbf{P_0} + \alpha\, \mathbf{d}$$
$$= \alpha\, \mathbf{P_1} + (1 - \alpha)\, \mathbf{P_0}$$

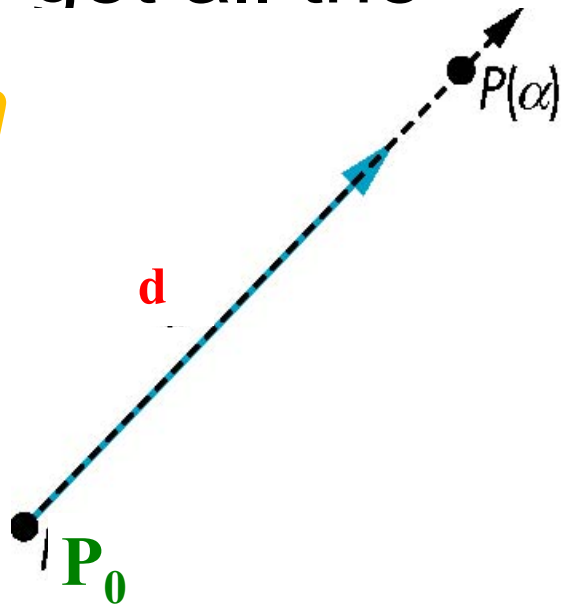
Rays and Line Segments

- If $\alpha >= 0$, then $P(\alpha)$ is the *ray* leaving P_0 in the direction **d**

If we use two **Points** to define **v**, then

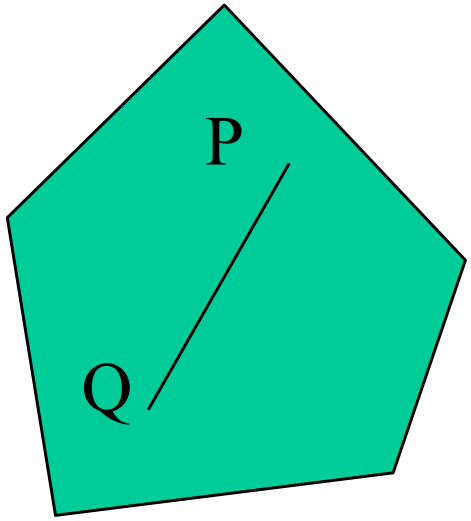
$$P(\alpha) = Q + \alpha (R-Q) = Q + \alpha v = \alpha R + (1-\alpha) Q$$

For $0<=\alpha<=1$ we get all the Points on the *lin* joining R and Q

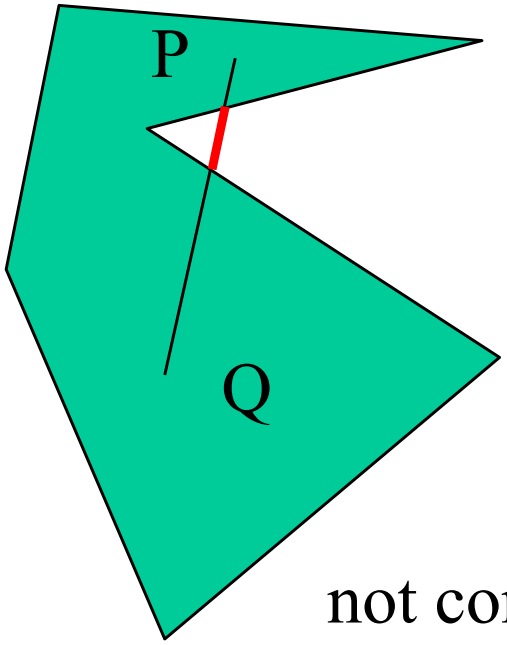


Convexity

An **object** is ***convex*** iff for any two **Points** in the **object** all **Points** on the line segment between these **Points** are also in the **object**



convex



not convex

Affine Sums

- Consider the "sum"

$P=\alpha_1 P_1+\alpha_2 P_2+\ldots..+\alpha_n P_n$

Can show by induction that this sum makes sense iff

$\alpha_1+\alpha_2+\ldots..\alpha_n=1$

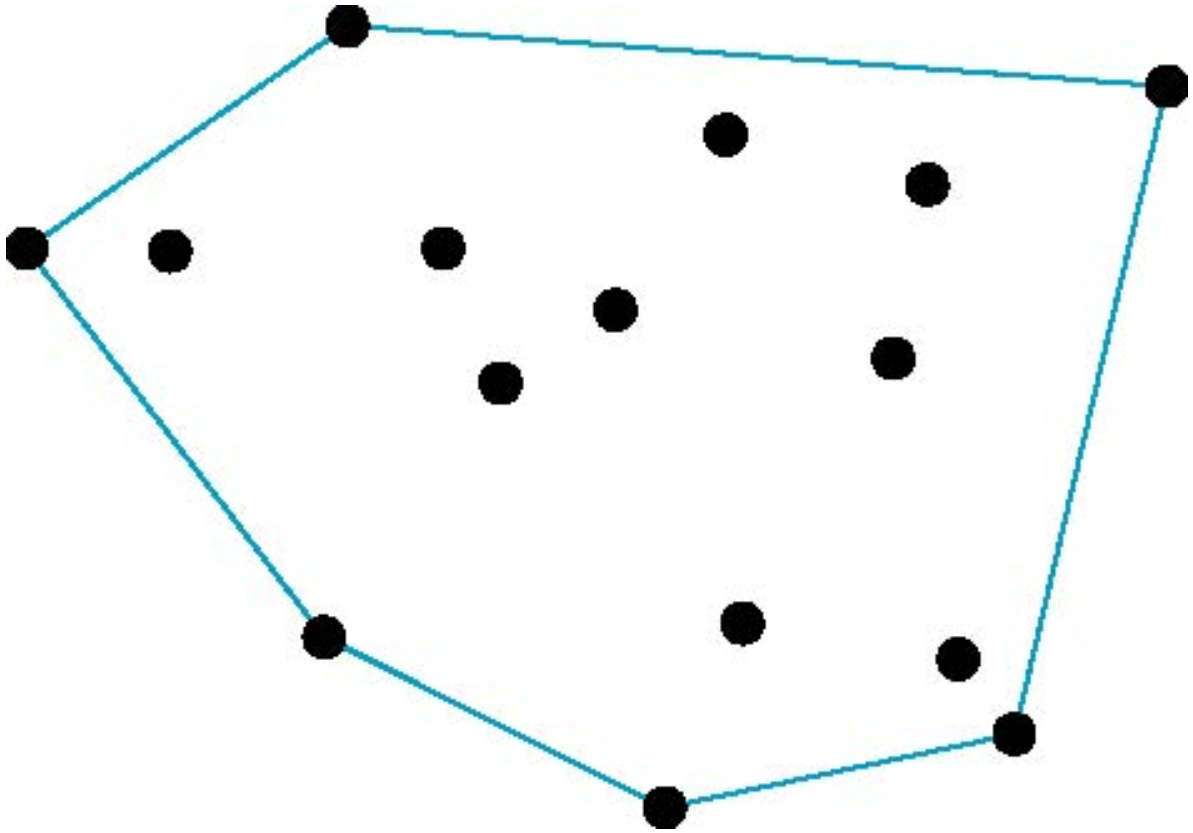
in which case we have the ***affine sum*** of the n Points

$P_1,P_2,\ldots..P_n$

- If, in addition, $\alpha_i>=0$, we have the ***convex hull*** of the set points $P_1,P_2,\ldots..P_n$

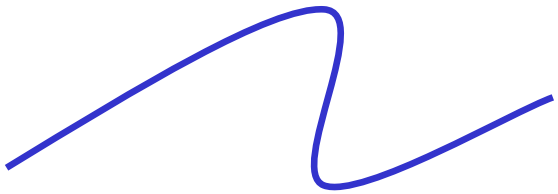
Convex Hull

- Smallest **convex object** containing $P_1, P_2, \ldots . P_n$
- Formed by “shrink wrapping” **Points**



Curves

- **Curves** are *one parameter entities* of the form P(α) where the *function is nonlinear*



P(α)

Explicit (may not): $y = mx + c$

Implicit (Rectangular) $ax + by + c = 0$

Parametric: $x = x(t)\ \ y = y(t)\ \ a <= t <= b$

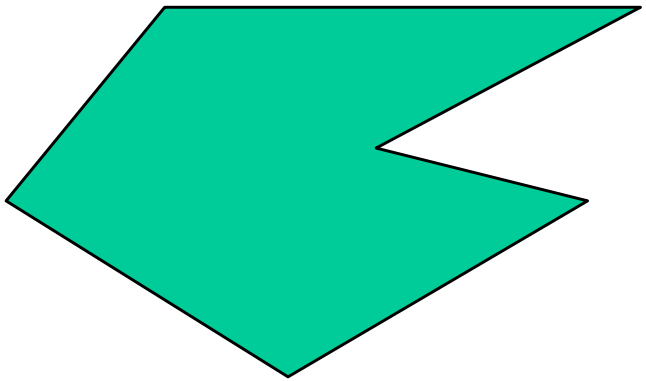
Surfaces

- **Surfaces** are formed from *two-parameter functions*

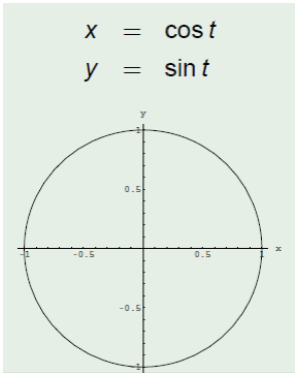
$P(\alpha, \beta)$

Linear functions give planes and polygons

Plot **curve** $x = cos(t)$ $y = sin(t)$ $0 <= t <= 2\Pi$



$P(\alpha, \beta)$

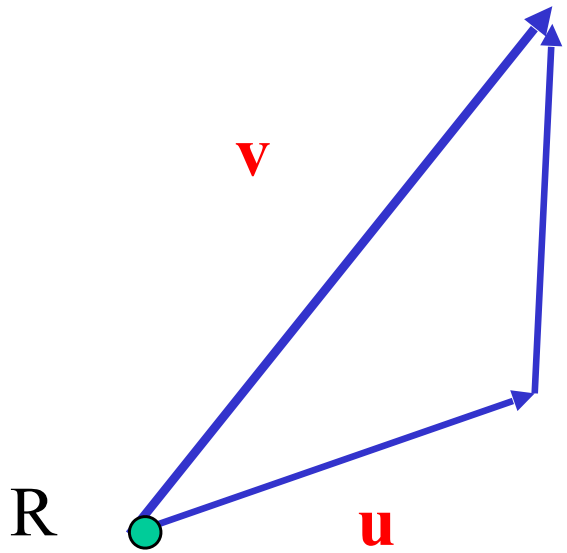


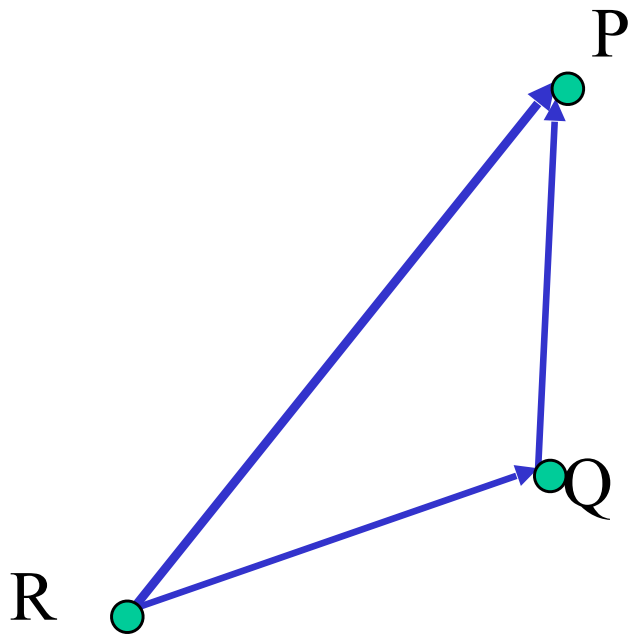
What is its **Implicit** form?

$x^2 + y^2 = 1$

Planes

A **Plane** can be defined by a **Point and two Vectors** **or** by **three Points**

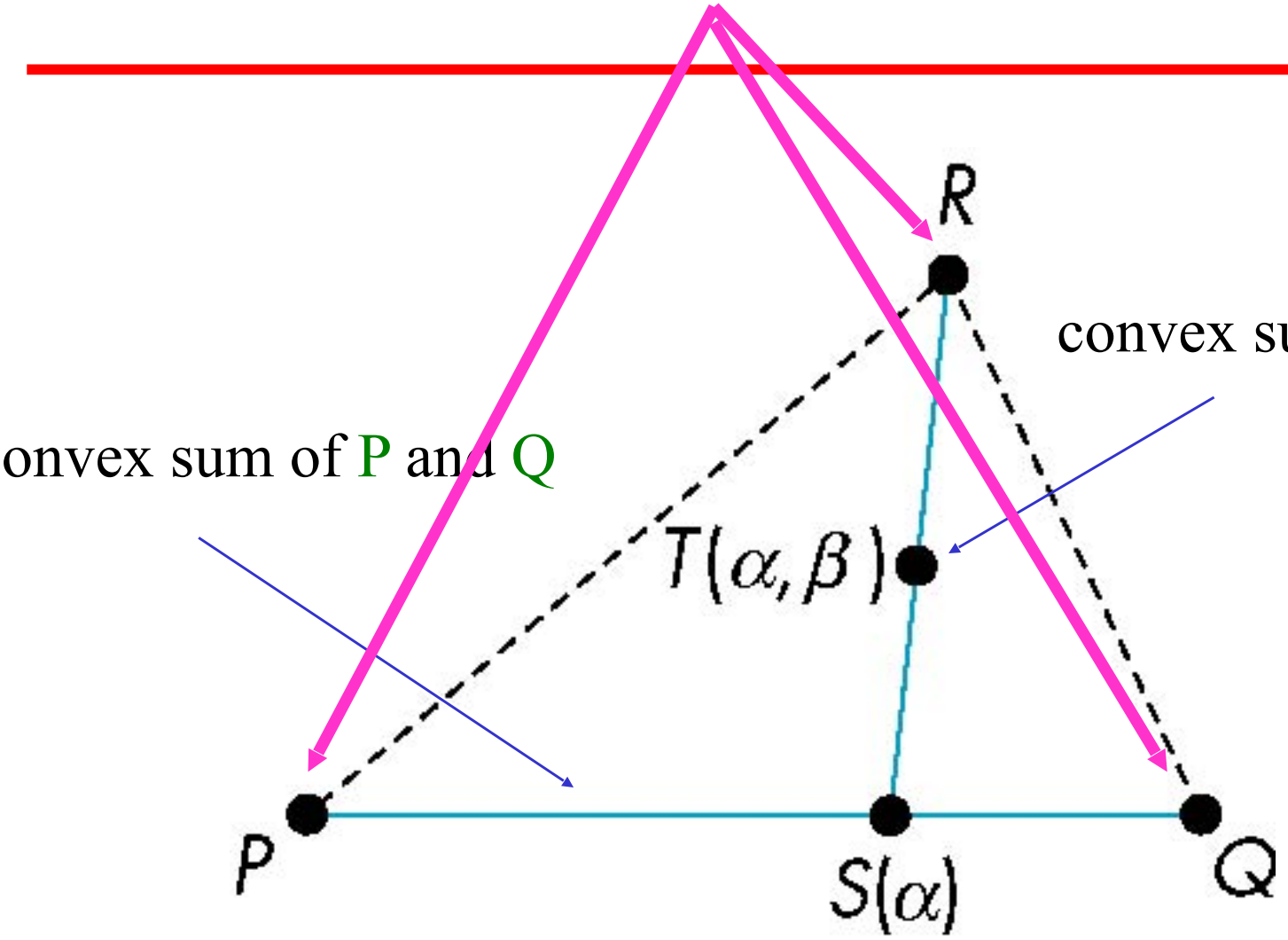




$P(\alpha,\beta)=R+\alpha u+\beta v$

$P(\alpha,\beta)=R+\alpha(Q-R)+\beta(P-R)$

Triangles



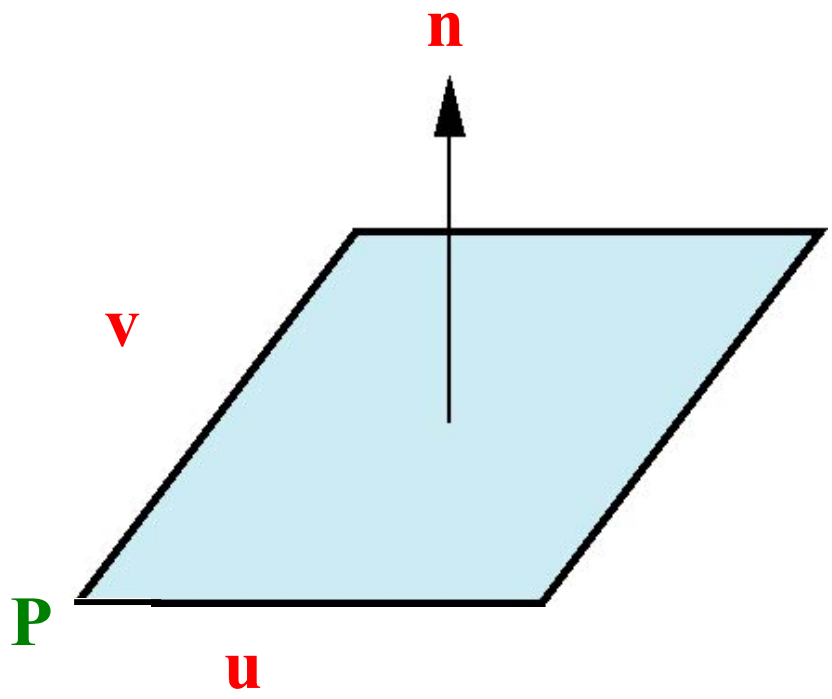
for 0<=α,β<=1, we get all points in triangle

Normals

Every Plane has a vector **n** normal (**perpendicular, orthogonal**) to it

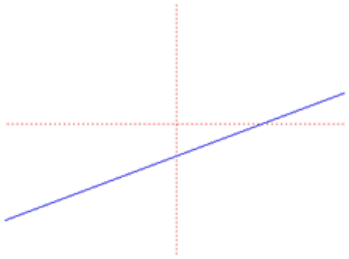
From **Point**-two **Vector** form $P(\alpha,\beta)=P+\alpha\mathbf{u}+\beta\mathbf{v}$, we know we can use the **cross product** $\times$ to find $\mathbf{n} = \mathbf{u} \times \mathbf{v}$ and the equivalent form using the **dot product** $\cdot$

$$(P(\alpha) - P) \cdot \mathbf{n} = 0$$



From 6:35 to 6:45 PM – 10 minutes.

2. Write the **explicit**, **implicit**, and **parametric** representations for a **Line.** (20 points)



Class Participation 2!

ANSWER:

3. Write the **explicit**, **implicit**, and **parametric** representations for a **Curve.** (20 points)

ANSWER:

Class Participation 3!

4. Plot Curve $x = cos(t)\ y = sin(t)\ 0 <= t <= 2\Pi$ (20 points)

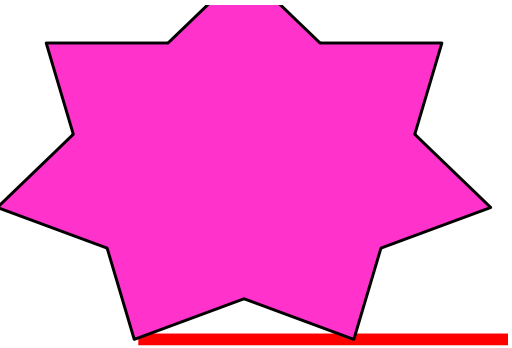
ANSWER:

Class Participation 4!

5. What is its **implicit** form? (20 points)

ANSWER:

Class Participation 5!



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.

Name: _____

Total score: ☐

Class PARTICIPATION on Lecture 0.doc **ANSWER SHEET**

(Out of 100 points, 20 points each. Please record your own total score!)

(Attach as score.doc!)

VH, close Assignment Class PARTICIPATION on Lecture 0

Next.

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

At 6:45 PM.

End Class 1

VH, Download Attendance Report
Rename it:
8.21.2023 Attendance Report FINAL

VH, upload Lecture 0 to CANVAS.