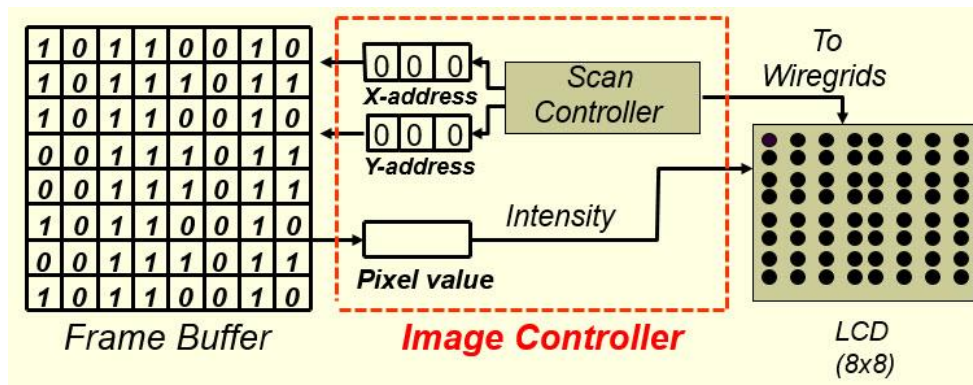


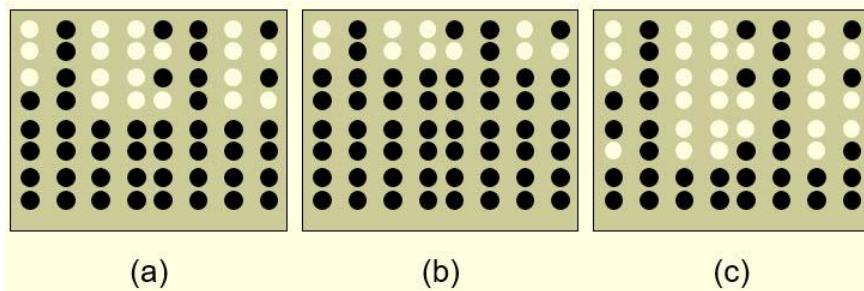
CS 535 Fall 2026
Midterm (100 points) October 22, 2026
Solution Set

1. (20 points)

(a) Given the following simple two-color raster scan system (a frame buffer with 8 x 8 entries, an image controller (also called a display processing unit) and a LCD display with a resolution of 8 x 8 pixels)



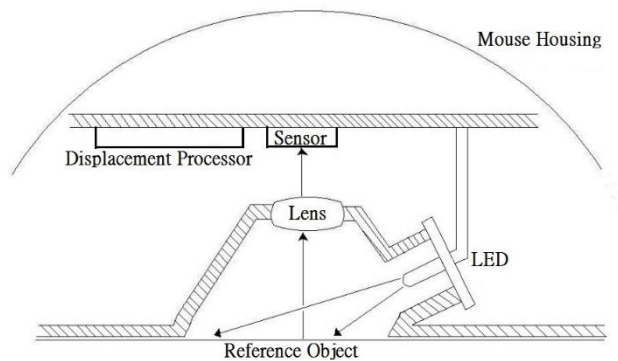
if the refresh rate is 120 Hz (meaning the screen will be refreshed 120 times per second), then after 1/96 seconds, how would the screen look like, the one shown in (a), (b), (c), or, none of these? (4 points)



Sol:

$1/96 = 1/120 + 1/4(1/120)$. After the first refresh cycle, the entire image will be shown on screen and will remain on screen (why?). so the answer is: None of these.

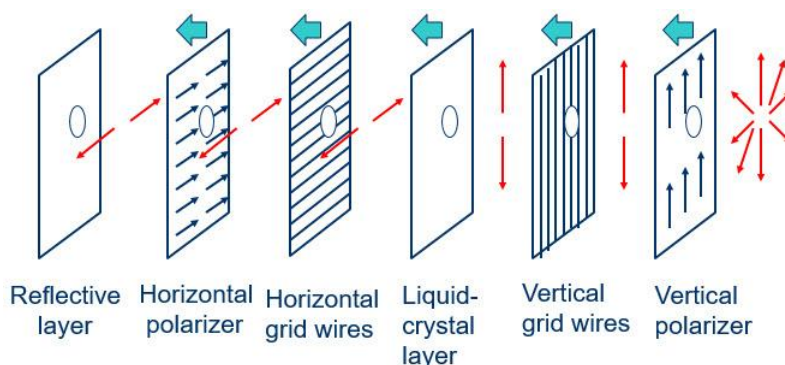
(b) A modern **optical mouse** (see below for an illustration of its internal structure) does not have to be operated on a special pad, it can be operated on any surface. However, the performance of a modern optical mouse on a piece of black glass is not good. Why?. (6 points)



Sol:

A piece of black glass suffers two problems. First, a significant amount of the light emitted by the LED will be absorbed by the glass, only a small portion will be reflected by the glass. Therefore, the images taken by the sensor of the mouse will not be bright enough for the computation process. Second, a more serious problem is that, since the surface of a glass is very smooth, shadows cast by the texture of the glass surface on itself are either too small or there will be no shadows at all. Combined, it would be difficult for the Displacement Processor to identify the locations of the LED at different moments based on differences of the shadows. So it would be difficult for the Displacement Processor to compute the distance the mouse has moved either.

- (c) In a flat panel display, if the third layer (the liquid-crystal layer) can only rotate the wave direction of incoming light by 30 degrees, instead of 90 degrees, when it is not in an electric field, then the flat panel display would not work anymore. Do you have a way to make it work? If you do, put your solution in the following box. (6 points)



SOL:

Add two more liquid-crystal layers before the Horizontal grid wires

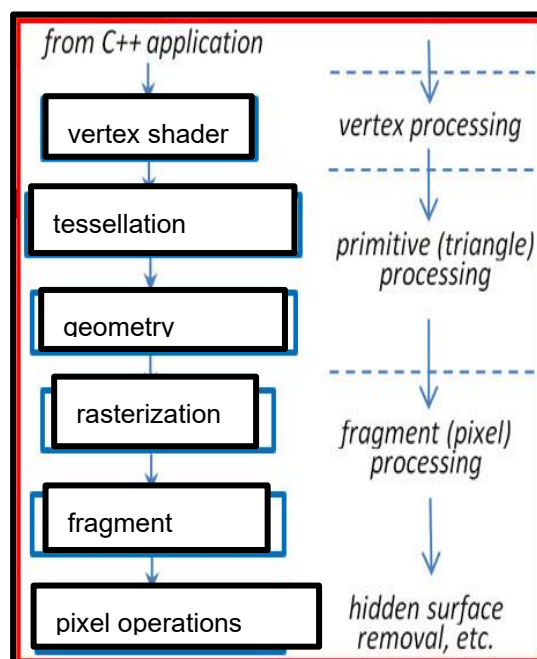
(d) In the notes "2D Raster Algorithms", an algorithm to efficiently scan convert a 2D polygon is introduced. For each active edge (an edge that intersects the current scanline), the algorithm gets the intersection point of the edge with the current scanline with only one floating-point addition, and there is no need for the algorithm to sort the intersection points. However, when building the bucket-sorted edge table (ET) for the given polygon (such as a triangle), certain edges should be ignored and certain edges should be shortened by one unit in y-direction. Which edges should be ignored and why? (4 points)

Sol:

Horizontal edges should be ignored. The purpose is to ensure the number of intersection points of a scan line with the bounding edges of a polygon is an even number.

2. (8 points)

Modern 3D graphics programming utilizes a pipeline. Each stage of the pipeline is done by a specific hardware. In the following chart, fill out the blanks for the names of those stages. (6 points)



Some of the stages are programmable and some are not. Which one(s) are

not programmable and why? (2 points)

The **rasterization** and **pixel operations** are not programmable. These two stages use pre-defined algorithms to perform primitive (lines, triangles) scan-conversion process and hidden line/surface elimination process.

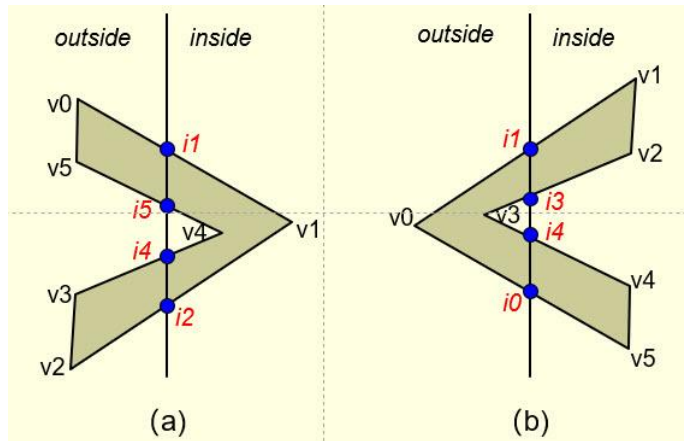
3. “Event handling” used to be done by the OpenGL command “glutMainLoop ()”. But now is handled by the command “glfwPollEvent()” in a while loop. The first instruction in that while loop is the “display()” function. What is the advantage of this kind arrangement? Put your answer in the following blank. (6 points)

With the “*display()*” function arranged as the first step in the while loop, animation is automatically supported by a modern C++/OpenGL application – simply render the entire scene each **time *display()* is called**.

4. Depth buffer and color buffer should both be cleared at the beginning of the “display” function when doing hidden surface elimination . Why? (6 points)

The color buffer is cleared each frame to avoid a moving object leaving a trail as it moves. This is mainly useful for moving objects. Note that if the depth buffer is not cleared, a moving object will still be shown in the non-overlapping area from the previous frame. Why?

5. If the two polygons shown in (a) and (b) are clipped against the left bounding edge of a 2D window using Sutherland-Hodgman and Weiler-Atherton algorithms, respectively, what would the outputs be in each case? In each case, the output by each algorithm is supposed to be a sequence of points (vertices of the original polygon or intersection points of the edges of the polygon with the left bounding edge of the window) or several sequences of points. For your convenience, intersection points of the polygon with the left bounding edge of the window are also shown below. (8 points)



Sol:

Case (a):

SH algorithm outputs one polygon: $P: i_1 v_1 i_2 i_4 v_4 i_5$

WA algorithm also outputs one polygon: $P: i_1 v_1 i_2 i_4 v_4 i_5$

Case (b):

SH algorithm outputs one polygon: $P: i_1 v_1 v_2 i_3 i_4 v_4 v_5 i_0$

WA algorithm outputs two polygons:

$P1: i_1 v_1 v_2 i_3$ and $P2: i_4 v_4 v_5 i_0$

6. (8 points)

(a) In perspective projection, if the projection plane is perpendicular to z axis at $z = -2$ and vanishing point of a line passing through $A=(-4, 0, 0)$ and $B=(0, 0, -6)$ is at $(2, 0, -2)$, then where is the center of projection if it is a point on the z-axis? (4 points)

Sol:

The point on the line

$$L(t) = (2, 0, -2) + t(4, 0, -6) = (2+4t, 0, -2-6t), \quad t \in \mathbb{R}$$

whose x-coordinate is zero is the center of projection.

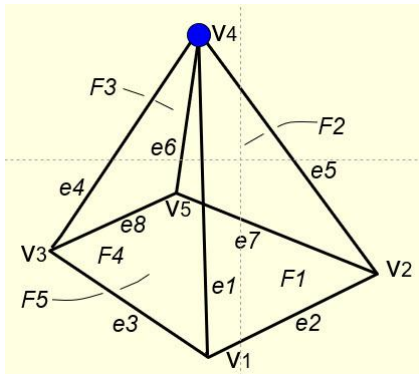
That point is $(0, 0, 1)$

(b) In 3D graphics, for perspective projection, we do clipping in homogeneous coordinated, i.e., before the perspective division step. Why? (4 points)

Sol:

To avoid getting incorrect clipping results.

7. To use the winged-edge data structure to represent the following pyramid, we need an Edge Table, a Vertex-Edge Table and a Face-Edge Table. To find all the adjacent edges of vertex v_4 , how many times do we have to access the Edge Table, the Vertex-Edge Table and the Face-Edge Table, respectively? Justify your answer. (4 points)



Sol:

Edge Table: 4 Vertex-Edge Table: 1
Face-Edge Table: 0

8. (8 points)

(a) OpenGL lists the vertices of each triangle in a fixed direction (counter-clockwise or clockwise) with respect to the outward normal of the triangle for a 3D object. Why? (4 points)

Sol:

OpenGL uses this information to determine if a triangle is a front triangle or a back triangle.

(b) Propose a practical use for `glCullFace(GL_FRONT_AND_BACK)`. (4 points)

Sol:

One can call `glCullFace(GL_FRONT_AND_BACK)` to make a specific object in a scene invisible for a specific purpose such as debugging.

9. In an OpenGL program, one can put computation of the perspective matrix in `display()` or in `init()`. Would there be any difference between these two choices?

(6 points)

Sol:

If the view point is fixed, computing the perspective matrix can be done either in *display()* or in *init()*. If the view point is not fixed, say is floating, then computing the perspective matrix should be done in *display()*.

10. (8 points)

(a) In the notes “3D Data Structures, 3D Data Management and 3D Models”, when we use a procedural approach to render a textured sphere, we divide the sphere into n slices and each slice into n segments. But the number of vertices (*numVertices*) in the vertex array is set to $(n + 1) * (n + 1)$. Why? (4 points)

Sol:

Each sliced sphere region is a closed band and the texture map is supposed to wrap around the entire sphere, this means the last point of each row of vertices should be connected to the start point of each row so two triangles can be formed between the last two points and the first two points of each two adjacent rows of vertices as well. Therefore each sliced sphere region actually is represented by $(n+1)$ vertices in the vertex array.

On the other hand, the number of rows of vertices that should be generated is 1 plus the number of sliced sphere regions because we generate two rows of vertices for each sliced region, one for each boundary of the region. By generating two rows of vertices for n slices, we get $2n$ rows of vertices. But $(n-1)$ pairs of the vertex rows coincide, $(n-1)$ of them should be ignored. So totally we have $2n - (n-1) = n+1$ distinct rows of vertices.

Since we have $n+1$ rows of vertices and each row has $n+1$ vertices, so totally we have $(n+1)*(n+1)$ vertices in the vertex array.

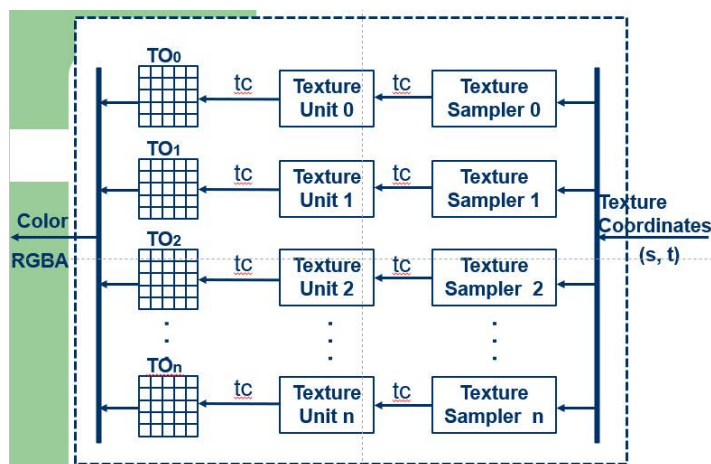
(b) In case (a), how many VAOs and VBOs are needed for the sphere to be rendered? Remember the sphere is textured and shaded (that is, a normal vector is needed for each vertex) ? (4 points)

Sol:

One option is to use one VAO and three VBO's, one VBO for vertex list, one for texture coordinates list, and one for normal list. Another option is to use one VAO and one VBO for a single vertex list. In this case, each vertex in the vertex list contains all the information about the vertex: vertex coordinates, texture coordinates, normal vector (or RGB colors of that vertex).

11. (4 points)

The purpose of texture mapping is to create an illusion of detail without increasing the amount of geometry to draw. The work is mainly performed by OpenGL. The following figure shows the process of sampling colors for object surfaces to be textured.



(a) Which stage of the OpenGL pipeline does this color sampling work shown in the above figure? (2 points)

Fragment shader

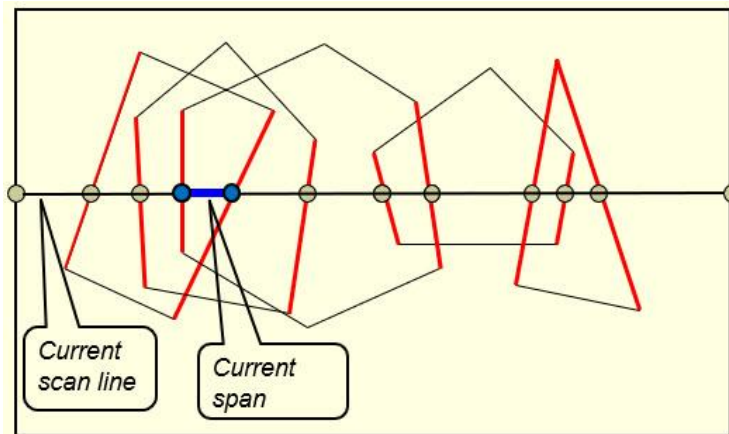
(b) What is a "texture unit" and what is its function? (2 points)

A texture unit is a piece of hardware on the graphics card. It is used to sample a texture object to get color for the pixel/ fragment with the specified texture coordinates.

12. (12 points)

(a) The scan-line hidden surface elimination method is an extension of the 2D polygon scan conversion method. The 2D scan conversion method processes one polygon at a time while the scan-line hidden surface elimination method can process multiple polygons simultaneously. Using the scan-line method to eliminate hidden surfaces, one needs to build two tables: a *bucket-sorted edge table* (ET) and a *polygon table* (PT), and maintains two lists: an *active edge list* (AEL) and an *active polygon list* (APL). Given the following projected polygons, the current scan line and the current span, please answer the following questions:

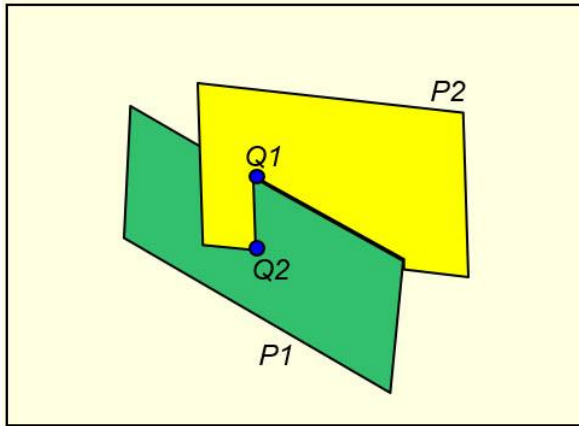
- (i) how many edges are contained in the ET?
- (ii) how many polygons are contained in the PT?
- (iii) how many edges are contained in the current AEL?
- (iv) how many polygons are contained in the current APL? (4 points)



Sol:

(i) 22 (ii) 5 (iii) 10 (iv) 3

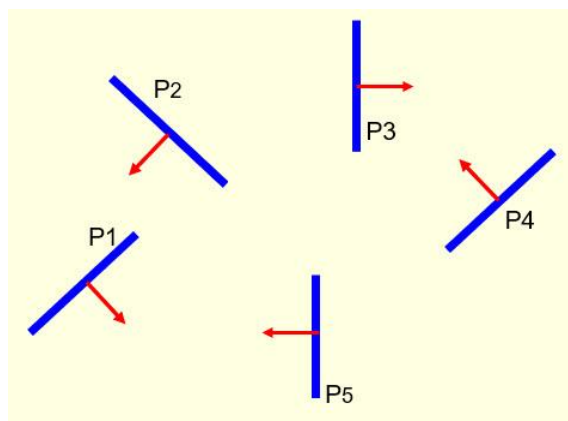
(b) Can Z-buffer method handle intersecting 3D polygons like the case shown below? Justify your answer. Note that part of polygon P2 is blocked by polygon P1 and part of polygon P1 is blocked by polygon P2. (4 points)



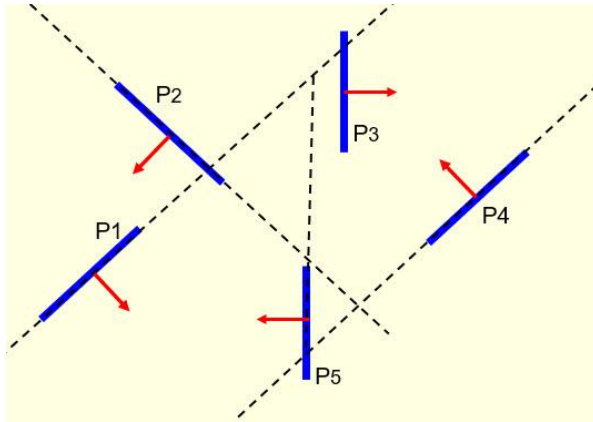
Sol:

YES. If polygon P1 is processed first, then by the time polygon P2 is processed, the portion of P2 that blocks P1 will overwrite the portion of P1 that is blocked by P1 in the depth array during the depth comparison process, and the portion of P2 that is blocked by P1 will not make any changes to the depth array. Therefore, we would see exactly the above scene shown on screen after the rendering process is done. The case that P2 is processed before P1 would generate exactly the same result too.

(c) The BSP tree method is basically a polygon sorting method. By sorting polygons using a BSP tree and then rendering polygons in the sorted order, one can eliminate hidden surfaces through the "overwriting" process. Given the following 5 polygons, what is the minimum number of polygons that could be added to a BSP tree? What is the maximum number?



To help you figure out all possible BSP trees that you could get, the following figure is provided. (4 points)



Sol:

Minimum number: 5

Maximum number: 8