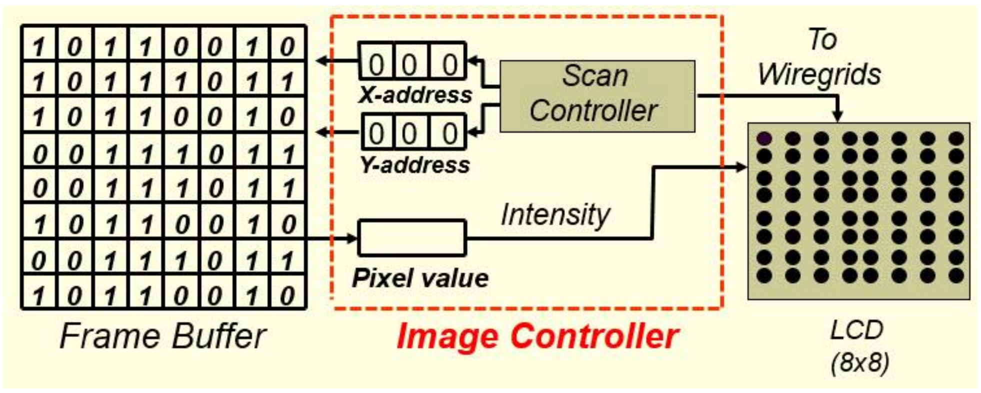
CS 535 Computer Graphics

Midterm Exam Solution Set (40 points)

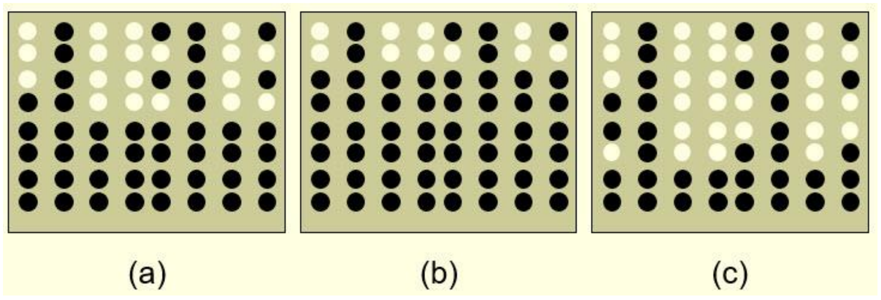
Due: 10/17/2024

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)

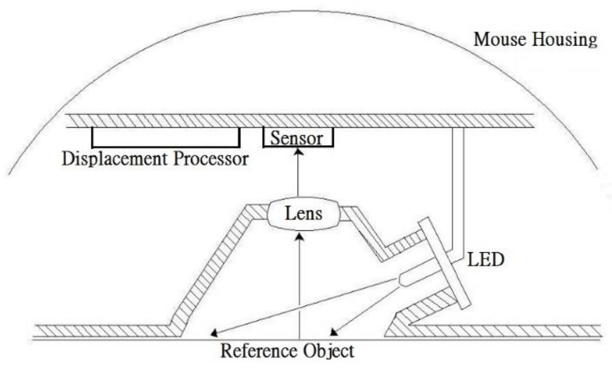


   if the refresh rate is 60 Hz (meaning the screen will be refreshed 60 times per second), then after the first 3/80 seconds, how would the screen look like, the one shown in (a), (b), (c), or, none of these? (4 points)



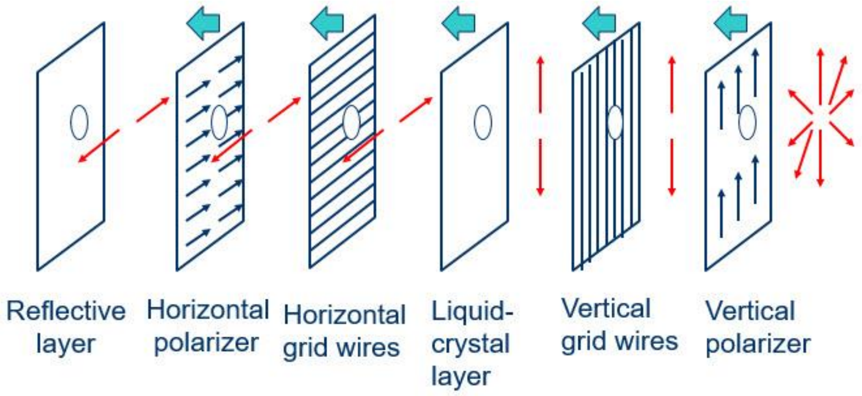
   None of these.

   (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, a modern optical mouse works better on a rough surface than on a smooth surface. Why? Put your answer in the box below the following figure. (6 points)



When lit at a grazing angle by the LED of the mouse, the texture of a rough surface casts bigger shadows on itself and the difference between the shadows is clear when the location of the LED is changed. Hence, it is easier for the software to identify the locations of the LED at different moments and consequently the distance the mouse has moved.

(c) In a flat panel display, if the third layer (the liquid-crystal layer) can only rotate the wave direction of incoming light by 45 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)

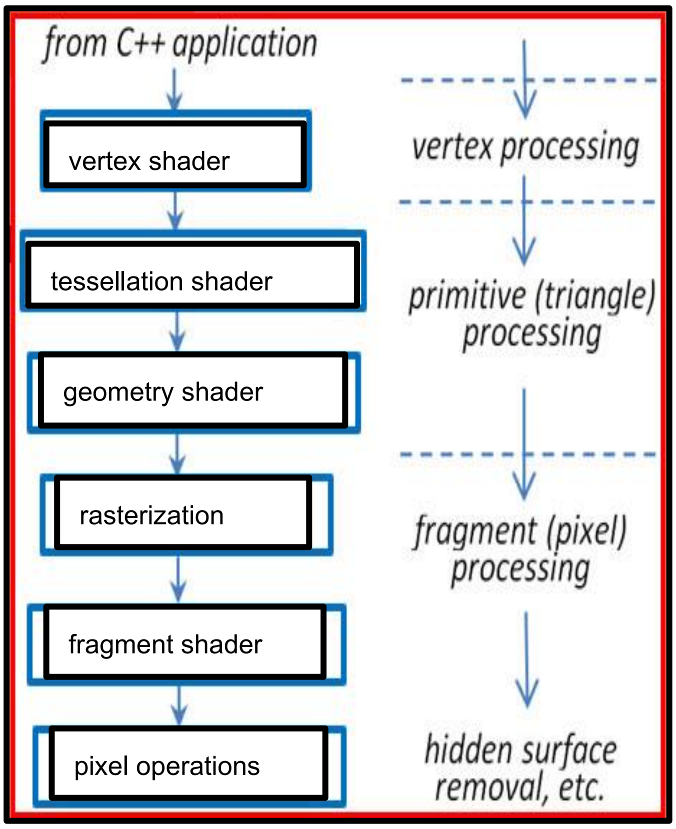


Add another liquid-crystal layer before the Horizontal grid wires

(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 shortened by one unit in y-direction? (4 points)

Each edge of the polygon which is not horizontal and whose lower vertex (the vertex with a smaller y-coordinate than the other vertex of the edge) is not a local maximum nor a local minimum of the polygon's boundary.

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)

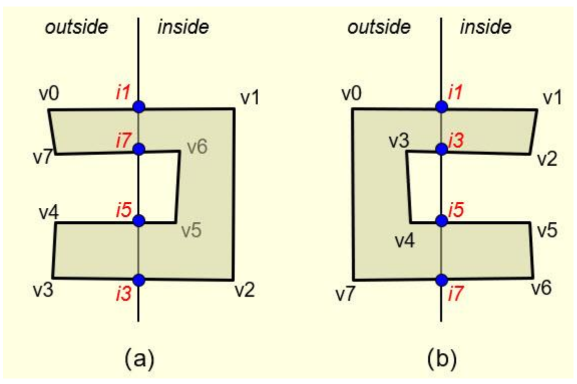
Some of the stages are programmable and some are not. Which one(s) are not programmable and why? (2 points)

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 ( )". Now it 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. 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)



Case (a):

SH algorithm outputs one polygon: P: $i_1v_1v_2i_3i_5v_5v_6i_7$

WA algorithm also outputs one polygon: P: $i_1v_1v_2i_3i_5v_5v_6i_7$

Case (b):

SH algorithm outputs one polygon: P: $i_1v_1v_2i_3i_5v_5v_6i_7$

WA algorithm outputs two polygons:

P1: $i_1v_1v_2i_3$ and P2: $i_5v_5v_6i_7$

5 (8 points)

(a) In perspective projection, if the projection plane is perpendicular to *z* axis at z = -3 and **vanishing point** of a line passing through A=(-5, 0, 0) and B=(0, 0, -4) is at (5, 0, -3), then where is the **center of projection**? (4 points)

Any point on the line

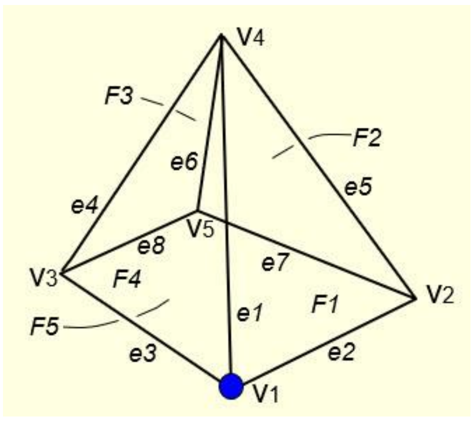
$L_1(s)$ = (5, 0, -3) + s(5, 0, -4) = (5+5s, 0, -3-4s), s ∊ R

with a negative parameter (s < 0) can be a center of projection, such as (0, 0, 1) when s = -1.

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

To avoid getting incorrect clipping results.

6. 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 v1, how many times do we have to access the Edge Table, the Vertex-Edge Table and the Face-Edge Table, respectively? (6 points)



Edge Table: 3
Vertex-Edge Table: 1
Face-Edge Table: 0

7. (14 points)

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

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

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

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

(c) What is the purpose of 'clearing the *depth buffer'* each time through *display*() in a 3D case and what is the purpose of 'clearing the *color buffer'* each time through *display*() in a 3D case? (6 points)

'*Clearing the depth buffer'* is to ensure hidden surfaces (performed by openGL using Z-Buffer method) are eliminated properly; '*clearing the color buffer*' is to ensure that we wouldn't get a trail of an moving object on screen.

8. In an OpenGL program, one can put computation of the perspective matrix in *display()* or in *init().* For a floating view point, which option is a better choice and why? (4 points)

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()*.

9. (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 $(\boldsymbol{n}+\mathbf{1}) * (\boldsymbol{n}+\mathbf{1})$. Why? (4 points)

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)

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).

10. (12 points)

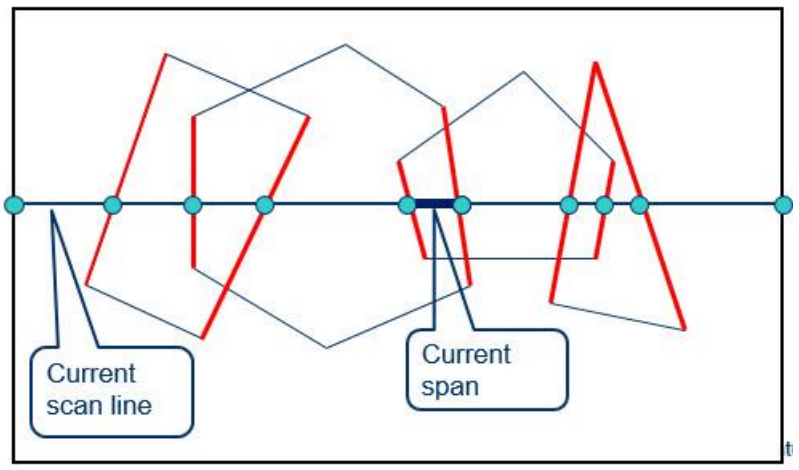
(a) The scan-line hidden surface elimination method is an extension of the 2D polygon scan conversion method. The 2D scan conversion method processes one polygon at a time while the scan-line hidden surface elimination method can process multiple polygons simultaneously. Using the scan-line method to 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 and put your answers in the following box:

(i) how many edges are contained in the ET?

(ii) how many polygons are contained in the PT?

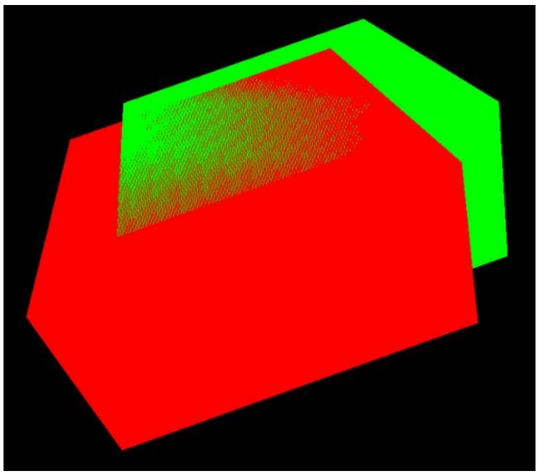
(iii) how many edges are contained in the current AEL?

(iv) how many polygons are contained in the current APL? (4 points)



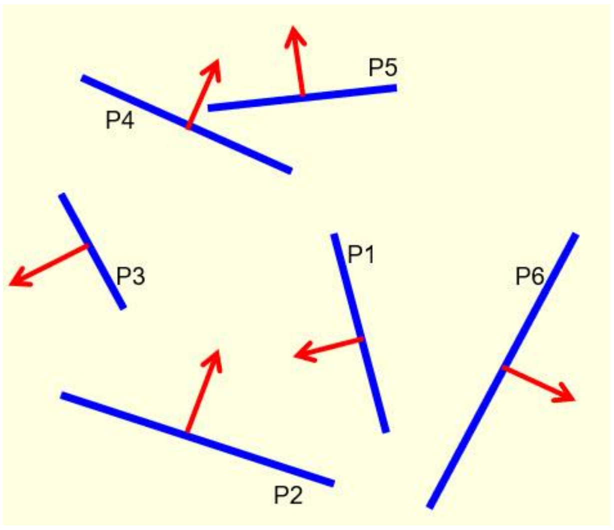
(i): 17
(ii): 4
(iii): 8
(iv): 2

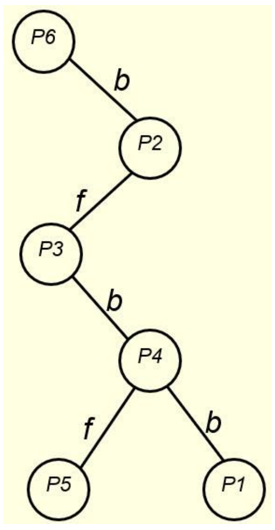
(b) OpenGL supports *Z-buffer method* in the "*Pixel Operation*" phase. Z-buffer method has the "*Z-fighting*" problem when two faces of two objects overlap and lie in coincident planes (see the following figure for an example of the "*Z-fighting*" problem). Provide a solution for the "*Z-fighting*" problem in the following blank. (4 points)



There is no real solution to this problem except to move one object slightly so the faces are no longer coplanar.

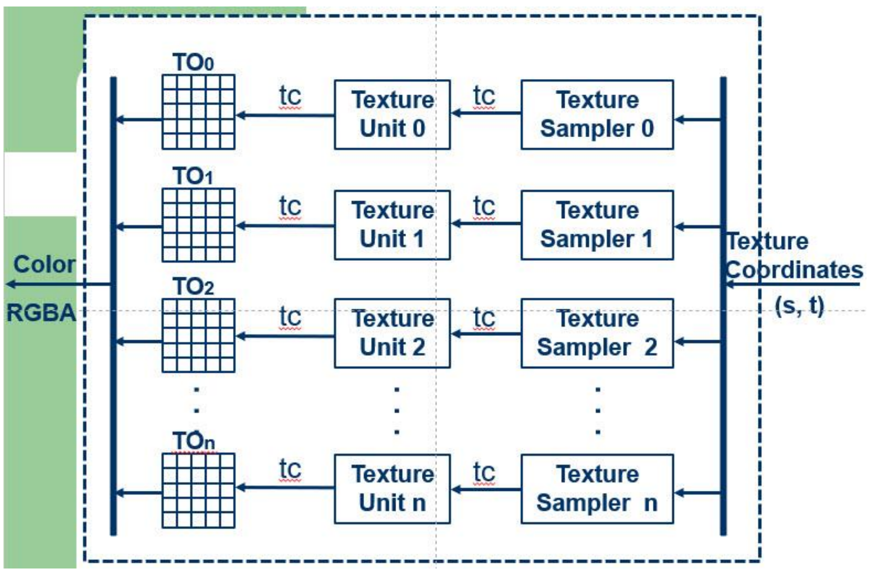
(c) The BSP tree method is basically a polygon sorting method. By sorting polygons using a BSP tree and then rendering polygons in the sorted order, one can eliminate hidden surfaces through the "overwriting" process. Given the following 6 polygons, what is the minimum number of polygons that could be added to a BSP tree? Show the corresponding BSP tree below too. (4 points)





Minimum number: 6
The corresponding BSP tree is shown above to the right of the given polygons.

11. (6 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? (4 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.