NATIONAL UNIVERSITY OF SINGAPORE

CS3241 — COMPUTER GRAPHICS

(AY2020/2021 SEMESTER 1)

MIDTERM ASSESSMENT

Time Allowed: **1 Hour 30 Minutes**

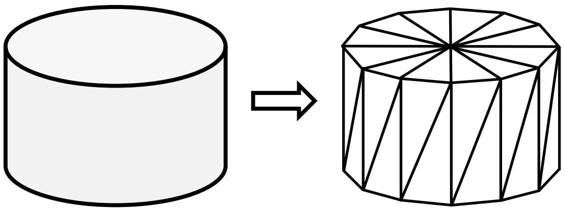
INSTRUCTIONS

1. This assessment contains **27** multiple-choice questions.
2. Each question has one correct answer. **2 marks** are awarded for each correct answer and there is no penalty for a wrong answer.
3. The full score of this assessment is **54 marks**.
4. Answer **all questions**.
5. This is an **Open-Book** assessment.
6. **Follow the instructions of your invigilator or the module coordinator to submit your answers**.

Section A

A closed cylinder, which is made of a flat top surface, a curved surface, and a flat bottom surface, is approximated and represented as a set of triangles as shown in the following diagram. The flat bottom surface, which is not visible in the diagram, is represented the same way as the top surface.

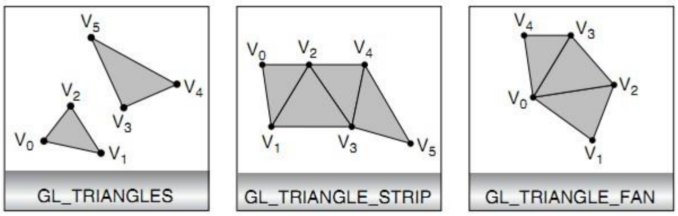
For the next 3 questions, consider this polygonal representation of the closed cylinder.



(1) How many ***distinct* vertices** are there in the representation of the cylinder?

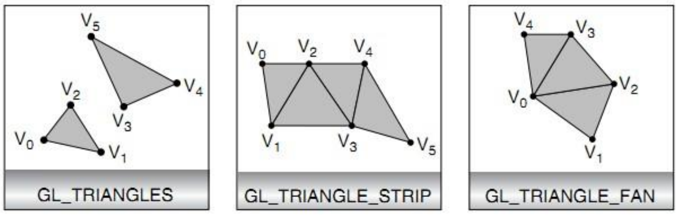
A. 12
B. 13
C. 24
D. 26
E. 48
F. 144

(2) We want to draw the **top and bottom flat surfaces** of the cylinder as `GL_TRIANGLE_FAN` primitives. In total, how many times do we need to call the `glVertex*()` function (just for drawing the top and bottom flat surfaces)?



A. 22
B. 24
C. 26
D. 28
E. 30
F. 32

(3) We want to draw the **curved surface** of the cylinder as a `GL_TRIANGLE_STRIP` primitive. How many times do we need to call the `glVertex*()` function (for drawing just the curved surface of the cylinder)?



A. 22
B. 24
C. 26
D. 28
E. 36
F. 72

Section B

(4) (This question has been removed.)

(5) Suppose an inkjet printer has used up its yellow ink, and it is left with only cyan and magenta ink. Which of the following colors cannot be produced by the printer on a piece of white paper?

- **A.** Red
- **B.** Blue
- **C.** Cyan
- **D.** Magenta
- **E.** White

(6) Suppose there is a real **pinhole camera** whose flat projection plane is made of some translucent material (e.g. tracing paper), which allows the projected image to be viewed by a person standing behind the camera. Which of the following statements is/are correct about the pinhole camera?

(i) The image is formed by a parallel projection.
(ii) The projected image seen by the person is a *rotated* version of what the person would see the scene directly.
(iii) The bigger the pinhole, the sharper the projected image.

- **A.** Only (i).
- **B.** Only (ii).
- **C.** Only (iii).
- **D.** Only (i) and (ii).
- **E.** Only (ii) and (iii).

Section C

(7) What does the homogeneous coordinates $[6\ \ 10\ \ 8\ \ 2]^T$ represent?

- **A.** The 3D point (3, 5, 4).
- **B.** The 3D point (6, 10, 8).
- **C.** The 3D point (12, 20, 16).
- **D.** The 3D vector (3, 5, 4).
- **E.** The 3D vector (6, 10, 8).
- **F.** The 3D vector (12, 20, 16).

(8) Given homogeneous coordinates $[16\ \ 12\ \ 12\ \ 4]^T$ and $[6\ \ 2\ \ 0\ \ 2]^T$, what does $[16\ \ 12\ \ 12\ \ 4]^T - [6\ \ 2\ \ 0\ \ 2]^T$ represent?

- **A.** The 3D point (10, 10, 12).
- **B.** The 3D point (5, 5, 6).
- **C.** The 3D point (1, 2, 3).
- **D.** The 3D vector (1, 2, 3).
- **E.** The 3D vector (5, 5, 6).
- **F.** The 3D vector (10, 10, 12).

(9) Which of the followings is the matrix that rotates objects about the point (9, 6, 4), where the rotation axis is the vector (0, 1, 0), and the rotation angle is θ?

- **A.** $\mathbf{T}(9, 6, 4) \cdot \mathbf{R}_y(\theta) \cdot \mathbf{T}(-9, 0, -4)$
- **B.** $\mathbf{T}(-9, -6, -4) \cdot \mathbf{R}_y(\theta) \cdot \mathbf{T}(9, 6, 4)$
- **C.** $\mathbf{T}(0, 6, 0) \cdot \mathbf{R}_y(\theta) \cdot \mathbf{T}(0, -6, 0)$
- **D.** $\mathbf{T}(0, -6, 0) \cdot \mathbf{R}_y(\theta) \cdot \mathbf{T}(0, 6, 0)$
- **E.** $\mathbf{T}(9, -3, 4) \cdot \mathbf{R}_y(\theta) \cdot \mathbf{T}(-9, 3, -4)$
- **F.** $\mathbf{T}(9, -3, 4) \cdot \mathbf{R}_y(\theta) \cdot \mathbf{T}(-9, -3, -4)$

(10) Given that $\boldsymbol{S}_1$ and $\boldsymbol{S}_2$ are any two possibly different 4x4 **scaling** matrices; $\boldsymbol{R}_1$ and $\boldsymbol{R}_2$ are any two possibly different 4x4 **rotation** matrices; and $\boldsymbol{T}_1$ and $\boldsymbol{T}_2$ are any two possibly different 4x4 **translation** matrices. Which of the following statements is correct? (The expression $A \equiv B$ means that A and B are *always equal*.)

- **A.** $\boldsymbol{S}_1\boldsymbol{S}_2 \equiv \boldsymbol{S}_2\boldsymbol{S}_1$
- **B.** $\boldsymbol{R}_1\boldsymbol{R}_2 \equiv \boldsymbol{R}_2\boldsymbol{R}_1$
- **C.** $\boldsymbol{T}_1\boldsymbol{R}_1 \equiv \boldsymbol{R}_1\boldsymbol{T}_1$
- **D.** $\boldsymbol{T}_1\boldsymbol{S}_1 \equiv \boldsymbol{S}_1\boldsymbol{T}_1$
- **E.** $\boldsymbol{S}_1\boldsymbol{R}_1 \equiv \boldsymbol{R}_1\boldsymbol{S}_1$

Section D

(11) Suppose we want to position the camera at the world-space location (`ex`, `ey`, `ez`), with the camera's *z*-axis points in the –*x* direction, and the camera's *y*-axis points in the –*z* direction. Which of the following `gluLookAt()` function calls achieves that?

A. `gluLookAt( ex, ey, ez, ex+1, ey, ez, 0, 0, 1 );`
B. `gluLookAt( ex, ey, ez, ex-1, ey, ez, 0, 0, -1 );`
C. `gluLookAt( ex, ey, ez, ex+2, ey, ez, 0, 1, -2 );`
D. `gluLookAt( ex, ey, ez, ex-2, ey, ez, 0, 1, 2 );`
E. `gluLookAt( ex, ey, ez, ex+2, ey, ez, 1, 0, -2 );`
F. `gluLookAt( ex, ey, ez, ex-2, ey, ez, 1, 0, 2 );`

(12) Which of the following is equivalent to this `gluLookAt()` function call:

```
gluLookAt( ex, ey, ez, ex, ey-1, ez, 0, 1, 1 );
```

A.
```
glRotated(90, 1, 0, 0); glRotated(180, 0, 1, 0);
glTranslated(-ex,-ey,-ez);
```

B.
```
glRotated(90, 0, 1, 0); glRotated(180, 0, 0, 1);
glTranslated(-ex,-ey,-ez);
```

C.
```
glTranslated(-ex,-ey,-ez);
glRotated(180, 1, 0, 0); glRotated(90, 0, 1, 0);
```

D.
```
glRotated(180, 0, 1, 0); glTranslated(-ex, -ey, -ez);
```

E.
```
glTranslated(-ex, -ey, -ez); glRotated(180, 0, 1, 0);
```

(13) A vertex, whose **camera coordinates** are (5, 5, 7.5), is being projected using the following OpenGL orthographic projection:

```
glOrtho( 0, 10, 0, 10, -10, 0 );
```

What will be the vertex's Normalized Device Coordinates (NDC)?

- **A.** (0, 0, –0.75)
- **B.** (0, 0, 0.75)
- **C.** (0, 0, –0.5)
- **D.** (0, 0, 0.5)
- **E.** (0, 0, –0.25)
- **F.** (0, 0, 0.25)

(14) Suppose the **viewport** is set up as `glViewport(30, 20, 200, 100)`, and the entire viewport is within the rendering window, what is the **2D window coordinates** of the NDC point (0.5, 0.5, 0.5)?

- **A.** (80, 45)
- **B.** (100, 50)
- **C.** (130, 70)
- **D.** (150, 75)
- **E.** (180, 95)
- **F.** (230, 120)

Section E

For the next 5 questions, consider the following C/C++ program, where the function `display()` is a GLUT display callback function.

```
...

void drawPoint(float v[3]) {
    glBegin(GL_POINTS);  glVertex3fv(v);  glEnd();
}

void display( void ) {
    ...

    glviewport( 0, 0, 800, 600 );
    glClear( GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT );

    glMatrixMode( GL_PROJECTION );
    glLoadIdentity();
    drawPoint(pointA);
    gluPerspective( fovY, aspect, near, far );

    glMatrixMode( GL_MODELVIEW );
    glLoadIdentity();
    drawPoint(pointB);
    gluLookAt( eyeX, eyeY, eyeZ, atX, atY, atZ, upX, upY, upZ);

    drawPoint(pointC);

    glPushMatrix();
      glTranslate3d( 10.0, 20.0, 30.0 );
      drawPoint(pointD);
    glPopMatrix();

    drawPoint(pointE);

    glPushMatrix();
      glTranslate3d( 40.0, 20.0, 20.0 );
      drawPoint(pointF);
    glPopMatrix();

    glutSwapBuffers();
}
...
```

(15) In which space are the coordinates of **`pointA`** specified?

- **A.** An object space
- **B.** World space
- **C.** Camera space
- **D.** Clip space
- **E.** NDC space
- **F.** Window space
- **G.** Unknown space

(16) In which space are the coordinates of **`pointB`** specified?

- **A.** An object space
- **B.** World space
- **C.** Camera space
- **D.** Clip space
- **E.** NDC space
- **F.** Window space
- **G.** Unknown space

(17) In which space are the coordinates of **`pointC`** specified?

- **A.** An object space
- **B.** World space
- **C.** Camera space
- **D.** Clip space
- **E.** NDC space
- **F.** Window space
- **G.** Unknown space

(18) In which space are the coordinates of **`pointD`** specified?

- **A.** An object space
- **B.** World space
- **C.** Camera space
- **D.** Clip space
- **E.** NDC space
- **F.** Window space
- **G.** Unknown space

(19) In which space are the coordinates of **`pointE`** specified?

- **A.** An object space
- **B.** World space
- **C.** Camera space
- **D.** Clip space
- **E.** NDC space
- **F.** Window space
- **G.** Unknown space

Section F

(20) A 3D scene was rendered to a **viewport** of size 300x200 pixels. If we now render the same scene, using the same view setup as before, to a **viewport** of size 600x400 pixels, which of the following OpenGL pipeline operations will have the most significant increase in **workload**? We assume that the viewports are entirely within the rendering window.

- **A.** Rasterization
- **B.** Back-face culling
- **C.** Viewport transformation
- **D.** Clipping
- **E.** View transformation
- **F.** Projection transformation

(21) Pixel p in the framebuffer is covered by triangles t_1, t_2, t_3, t_4, t_5, t_6, t_7, and t_8 only. The triangles are rendered in the order $t_1, t_2, \ldots, t_8$. Suppose the **depth values** of the fragments of $t_1, t_2, \ldots, t_8$ at the location of p are 0.9, 0.7, 0.6, 0.2, 0.1, 0.8, 0.4, 0.3, respectively, how many times will the z-value in the **z-buffer** for pixel p be modified *after* it was initialized?

- **A.** 1
- **B.** 2
- **C.** 4
- **D.** 5
- **E.** 6
- **F.** 7

(22) Which of the following statements is correct about **z-fighting**?

- **A.** When rendered with an **orthographic** projection, polygons **nearer** to the viewpoint are more prone to z-fighting than those polygons **further** away from the viewpoint.
- **B.** When rendered with an **orthographic** projection, polygons **further** away from the viewpoint are more prone to z-fighting than those polygons **nearer** to the viewpoint.
- **C.** When rendered with a **perspective** projection, polygons **nearer** to the viewpoint are more prone to z-fighting than those polygons **further** away from the viewpoint.
- **D.** When rendered with a **perspective** projection, polygons **further** away from the viewpoint are more prone to z-fighting than those polygons **nearer** to the viewpoint.
- **E.** None of the other statements is correct.

(23) If we render a **cube** (which has 6 faces) using **perspective projection**, what is the **maximum** possible number of faces that can be eliminated by **back-face culling**?

- **A.** 6
- **B.** 5
- **C.** 4
- **D.** 3
- **E.** 2
- **F.** 1

(24) Which of the following statements is/are correct about **back-face culling**?

(i) Back-face culling could be performed in **window space**.
(ii) Back-face culling could be performed in **camera space**.
(iii) Back-face culling **cannot** be performed in the **vertex processing stage**.

- **A.** None of them.
- **B.** Only (iii).
- **C.** Only (i) and (ii).
- **D.** Only (i) and (iii).
- **E.** Only (ii) and (iii).
- **F.** All (i), (ii) and (iii).

(25) When using the Cohen-Sutherland Algorithm (i.e. the one that uses *outcodes*) to clip a **2D line segment** against a rectangular clipping window, what is the **maximum** number of **intersections** that may need to be computed?

- **A.** 1
- **B.** 2
- **C.** 3
- **D.** 4
- **E.** 6
- **F.** 8

(26) When a **2D triangle** is clipped against a rectangular clipping window, what is/are the **most complex shape(s)** that can be formed *inside* the clipping window?

- **A.** One 8-sided polygon
- **B.** One 7-sided polygon
- **C.** One 5-sided polygon
- **D.** One quadrilateral
- **E.** Two separate quadrilaterals
- **F.** Two separate triangles

(27) **Two** straight **line segments** are being **scan-converted**. The first line segment has endpoints at pixel locations $\boldsymbol{A}_1(1, 1)$ and $\boldsymbol{A}_2(110, 50)$, and the second line segment has endpoints at pixel locations $\boldsymbol{B}_1(12, 12)$ and $\boldsymbol{B}_2(32, 72)$. Assuming the line segments are drawn as thin as possible and not broken, what is the **total number of fragments** (pixels that are turned on) that are produced for these two line segments? This number includes the four fragments at the four end points of the two line segments.

- **A.** 21 + 61
- **B.** 50 + 110
- **C.** 50 + 21
- **D.** 50 + 61
- **E.** 110 + 21
- **F.** 110 + 61

END OF QUESTIONS