University of Canterbury

Mid-Year Examinations 2016

Prescription number: COSC363

Paper title: Computer Graphics

Time allowed: 3 hours

Number of pages: 9

Maximum marks: 100. Contribution to the final grade: 50%

Number of questions: 10

- **Answer *all* questions**.
- This exam is worth a total of 100 marks.
- This is a **closed book** exam. No written or printed material is allowed**.**
- **UC approved calculators** are allowed.
- Please answer all questions carefully and to the point. Check carefully the number of marks allocated to each question. This suggests the degree of detail required in each answer, and the amount of time you should spend on the question.
- Use the separate answer booklet provided for answering all questions.

Important Formulae

If $\boldsymbol{v}_1 = (x_1, y_1, z_1)$ and $\boldsymbol{v}_2 = (x_2, y_2, z_2)$, then the cross-product of the two vectors is given by

$$\boldsymbol{v}_1 \times \boldsymbol{v}_2 = (y_1 z_2 - y_2 z_1,\ z_1 x_2 - z_2 x_1,\ x_1 y_2 - x_2 y_1).$$

Rotation matrices:

$$R_x(\theta) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$R_y(\theta) = \begin{bmatrix} \cos\theta & 0 & \sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$R_z(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Equation of a ray through $\boldsymbol{p}_0$ and having a unit direction $\boldsymbol{d}$: $\boldsymbol{p} = \boldsymbol{p}_0 + t\boldsymbol{d}$, $t > 0$.

Equation of a plane passing through $\boldsymbol{a}$ and with a normal direction $\boldsymbol{n}$: $(\boldsymbol{p}-\boldsymbol{a}) \bullet \boldsymbol{n} = 0$

Second order Bezier curve defined using three control points P_0, P_1, P_2:

$$P(t) = (1-t)^2 P_0 + 2t(1-t)\, P_1 + t^2 P_2\ ,\ \text{ where } P(t) = (x(t), y(t)), \quad 0 \le t \le 1.$$

Question 1. [10 marks for the whole question] *Transformations*

Assume that you are given three functions: (i) `drawCube()` that draws a cube of side 1 unit, centred at the origin, (ii) `drawSail()` that draws a single sail/blade of a windmill at the origin, and (iii) `drawCone()` that draws the base of a windmill (a cone with a predefined radius and height) at the origin, with the cone's axis along the *z*-axis. Figures (1a), (1b), (1c) show the models generated by the functions.

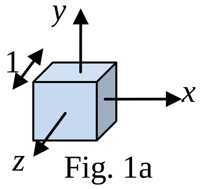


Fig. 1a

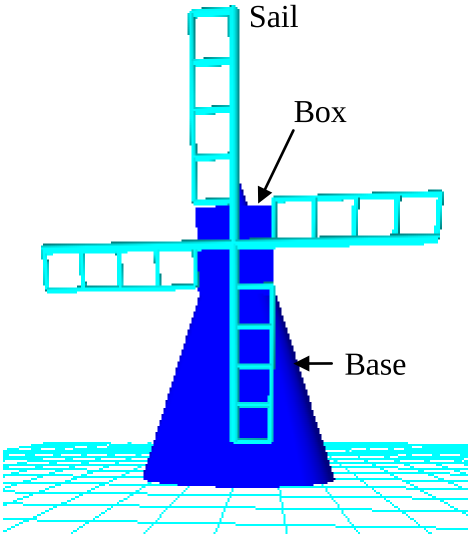


Fig. 1d

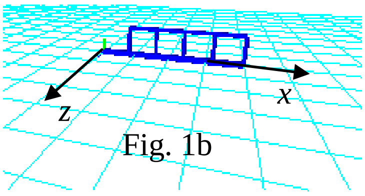


Fig. 1b

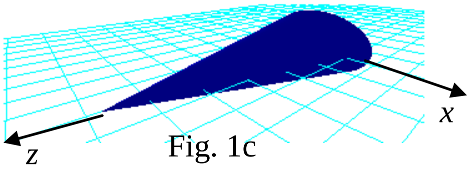


Fig. 1c

You are also given the following set of instructions for creating the model of a windmill:

Step 1: Create the base of the windmill using `drawCone()` and rotate it to make the axis vertical. You need to find the angle and axis of rotation.

Step 2: Create a cube using `drawCube()` at the origin and scale it by 8 times along all three axes.

Step 3: Move the scaled cube along the *y*-axis to position (0, 25, 0).

Step 4: Using a for-loop and the function `drawSail()`, create four blades at the origin and rotate them 90 degrees apart.

Step 5: Translate the set of four blades to position (0, 25, 5).

Step 6: Steps 1-5 above create the windmill model. Translate the whole windmill to position (100, 0, 50)

Using the given functions, OpenGL transformation functions and glPushMatrix-glPopMatrix blocks, write a code segment for defining only the transformations as outlined above. The transformed model of the windmill at position (100, 0, 50) is shown in Fig. 1d.

Question 2. [12 marks for the whole question] *Illumination Model*

Sam wrote an OpenGL program to render a three-dimensional scene, and included the following lines of code in the `initialize()` function.

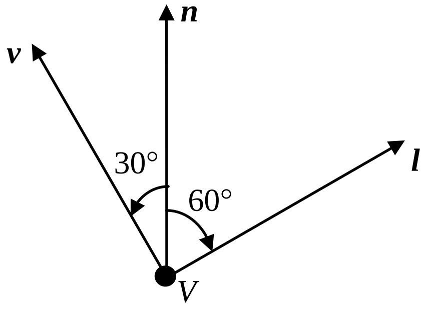
```
glColorMaterial(GL_FRONT, GL_AMBIENT_AND_DIFFUSE);
glEnable(GL_COLOR_MATERIAL);
```

(a) [3 Marks] Briefly describe the usefulness of the above statements in a program, and how it affects lighting calculations in a scene.

(b) [3 Marks] Sam noticed that his program did not produce specular highlights on polygonal elements even when lighting was enabled and the light's specular colour was set to white. Give a possible cause for this problem, and suggest a method to generate specular reflections on all polygonal objects. You are not required to write any OpenGL functions in your answer.

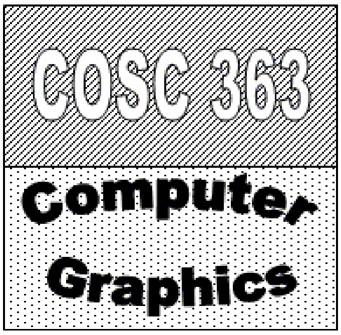
(c) [3 Marks] Sam also noticed a problem with the implementation of a spotlight that was directed towards the centre of a floor plane defined using a large quad. The spot light was not visible even though the light's parameters were set correctly. Explain the cause of this problem by describing the way lighting computations are performed in the OpenGL fixed function pipeline, and suggest a method to rectify this problem.

(d) [3 Marks] The figure given below shows the normal vector ***n***, light source vector ***l***, and the view vector ***v*** at a vertex *V*. All vectors are unit vectors on the *xy*-plane. Redraw this diagram in your answer booklet, and show the reflection vector ***r*** and the half-way vector ***h***. Also write the values of the angle between ***n*** and ***h***, and the angle between ***r*** and ***v***.

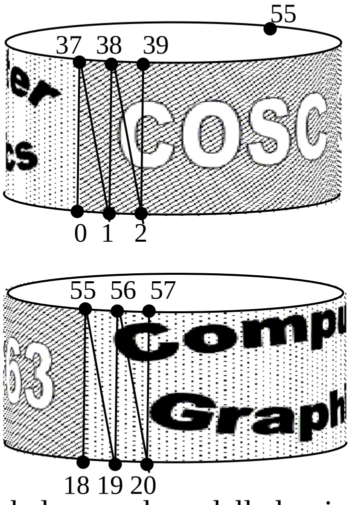


Question 3. [8 marks for the whole question] *Texture Mapping.*

The curved surface of a cylinder is required to be texture mapped using a single image as shown in the figure below. One half of the cylinder uses the top portion of the image, while the other half of the cylinder is textured using the bottom half of the image.



Texture



The cylindrical surface is divided into two halves and modelled using two triangle strips. The first few triangles of each triangle strip are shown in the figure.

	Verts on lower edge	Verts on upper edge
Triangle strip-1	0, 1, …18	37, 38, … 55
Triangle strip-2	18, 19, …36	55, 56, …73

The vertex coordinates are stored in arrays as given below.

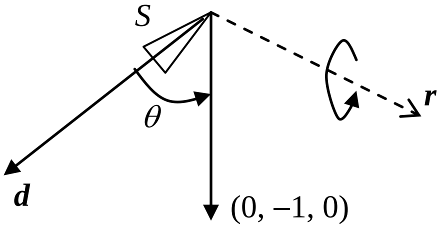
```
float vx[74], vy[74], vz[74];
```

Assume that the coordinates of all vertices have been computed. Use glBegin()-glEnd() blocks and two separate for-loops to define the two triangle strips. Assign texture coordinates to vertices to generate a texture mapping as shown in the figure above. Use glVertex3f() function for vertex definition, and glTexCoord2f() for specifying texture coordinates. No other OpenGL functions should be used.

Question 4. [10 marks for the whole question] *Mathematical Aspects*

(a) [2 Marks] Compute the magnitude (length) of the vector (2, 3, –6) and normalize the vector.

(b) [4 Marks] A spotlight S is defined with a direction vector $\boldsymbol{d}$ = (–1, –2, 2) as shown in the figure below. An animation sequence requires a single rotation of this spotlight from its initial direction $\boldsymbol{d}$ towards the direction (0, –1, 0). Compute the angle of rotation θ, and the vector $\boldsymbol{r}$ about which this rotation must be performed. It is not necessary to normalize the vector $\boldsymbol{r}$.



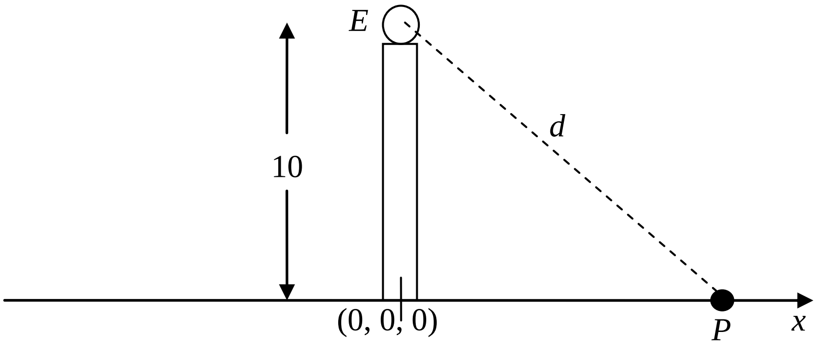
(c) [2 Marks] Write the transformation defined in the code segment below as a product of two 4x4 matrices. You are not required to multiply the matrices.

```
glPushMatrix();
glTranslatef(-2, 3, 8);
glScalef(2, 0.5, 4);
glutSolidTeapot(1);
glPopMatrix();
```

(d) [2 Marks] What transformation does the following matrix represent?

$$\begin{bmatrix} 0.866 & 0 & 0.5 & 0 \\ 0 & 1 & 0 & 0 \\ -0.5 & 0 & 0.866 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Question 5. [10 marks for the whole question] *Viewing*



In an OpenGL program, a camera E is required to be positioned at the point (0, 10, 0) as shown in the figure above. The camera looks at a point P on the x-axis, and the distance EP is d units.

(a) [2 Marks] For the camera configuration given above, write the values of all nine parameters of the following function:

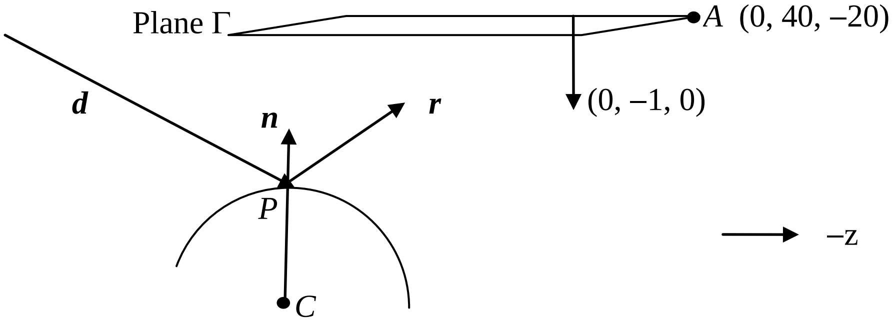
```
gluLookAt(ex, ey, ez, lx, ly, lz, upx, upy, upz);
```

(b) [3 Marks] Draw a figure in the answer booklet showing the points *E*, *P*, the *x*-axis, and the three axes *u*, *v*, *w* of the eye coordinate frame. Indicate an axis perpendicular to the plane of the paper with a '+' sign (eg. +*u*) if it is directed towards the reader, or a '–' sign if it is oriented away from the reader.

(c) [2 Marks] Write the eye-coordinates of the points *E* and *P*.

(d) [3 Marks] Briefly describe in words how the *u*-axis of the eye coordinate frame is computed.

Question 6. [8 marks for the whole question] *Ray tracing*.



A primary ray in the direction ***d*** hits a reflective sphere at *P* and bounces off in the direction ***r***. The ray ***r*** then hits a ceiling plane Γ as shown in the figure above.

(a). [1 Mark] If the centre of the sphere is at *C* = (2, 6, –10), and the point of intersection is given by *P* = (2, 10, –10), compute the **unit** normal vector ***n*** at *P*.

(b). [3 Marks] The direction of the reflected ray is given by ***r*** **=** –2 (***d*** •***n***) ***n*** + ***d***. If ***d*** = (0, –0.6, –0.8), find the components of the vector ***r***.

(c). [3 Marks] The plane Γ passes through the point *A*, and has a normal vector $\boldsymbol{n}_1$ = (0, –1, 0). Compute the ray parameter *t* at the point of intersection of the ray ***r*** and the plane, using the equation

$$t = \frac{(A - P) \bullet n_1}{r \bullet n_1}$$

(d) [1 Mark] What does the above value of *t* represent?

Question 7. [15 marks for the whole question] *Vertex and Fragment Shaders*

The source codes for a vertex shader and a fragment shader are given below.

Vertex shader:

```
layout (location = 0) in vec4 position;
layout (location = 1) in vec4 color;
uniform mat4 mvpMatrix;
out vec4 theColor;
void main()  {
     gl_Position = mvpMatrix * position;
     theColor = color;

}
```

Fragment shader

```
in vec4 theColor;
void main() {
     if (theColor.r < 0.5) discard;
     gl_FragColor = theColor;
}
```

Give a brief description of each of the 6 lines of code (excluding "void main()") in the vertex shader, and each of the 3 lines of code in fragment shader. For declaration statements, explain the role of storage qualifiers in determining how values are received from the application or passed to other shaders. For statements within the main() function, explain what computations are done, or why a variable is assigned to another.

Question 8. [10 marks for the whole question] *Bezier Curves.*

A quadratic Bezier curve is generated by combining three control points using second degree Bernstein polynomials ($(1-t)^2$, $2t(1-t)$, t^2).

(a) [4 Marks] Given three control points $P_0 = (3, 0, 0)$, $P_1 = (1, 0, 1)$, $P_2 = (-1, 4, 2)$, write the parametric equations for $x(t)$, $y(t)$, $z(t)$ for points on the Bezier curve.

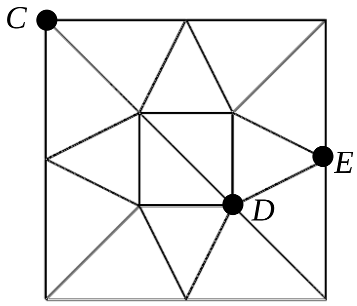
(b) [4 Marks] Show that every point on the above Bezier curve satisfies the equations $y = z^2$ and $x + 2z = 3$.

(c) [2 Marks] A general n^{th} degree Bezier curve is formed using n+1 control points P_0, ..P_n. Outline two geometrical properties satisfied by the Bezier curve.

Question 9. [10 marks for the whole question] *Tessellation Shaders*

(a) [5 Marks] Consider a surface design application where a 4x4 control polygonal grid is used to generate a bi-cubic Bezier surface. Briefly describe how the tessellation stage of the OpenGL-4 pipeline could be effectively used for this application, clearly outlining the main computations performed inside the control and evaluation shaders. You are not required to write any equations or code as part of your answer.

(b) [5 Marks] A quad domain is tessellated as shown in the figure below. Write the inner and outer tessellation levels used in generating the figure, and give the tessellation coordinates of the points *C*, *D* and *E*.



Question 10. [7 marks for the whole question] *Geometry Shader*

A geometry shader provides the following useful features:

(i) The shader can process a primitive as a whole.

(ii) The shader can generate new primitives.

(iii) The shader can discard a primitive.

Give short descriptions of three applications, each using one of the above features.

END OF PAPER