Name: ____________________

Student Number: ____________________

Signature: ____________________

The University Of New South Wales

Sample Exam - Written

SAMPLE SOLUTIONS

COMP3421 & COMP9415

Computer Graphics

Time allowed: **1 hours**
Total number of questions: **14**
Total number of marks: **28**
Number of pages: 4
Note: Actual exam will be 2 hours and worth 60 marks
Examination Materials: Rulers, Textbooks, print-outs and hand written notes **permitted**.
UNSW Approved Calculators **may** be used.
Questions are **NOT** worth equal marks.
Answer **all** questions.
This paper may **not** be retained by the candidate.

There are 3 parts. Part A, B and C. Answer each part in a separate booklet.
Answers must be written in ink. Except where they are expressly required, pencils may be used only for drawing, sketching or graphical work.

Examiner's Use Only:

1	2	3	4	5	6	7	8	9	10	11	12	13	14	Total

Part A:

Question 1

(5 marks)

The normal at a vertex $(0, 1, 2)$ on a surface is $(0, 4, 5)$. The light source is at $(0, 1, 4)$. The intensity of the light is $(0.9, 0, 0.2)$. The diffuse co-efficient of the surface is $(0.4, 1, 0)$. What will the RGB colour of the vertex be? Assume there is no specular, ambient or any other light reflected by the surface.

Answer

We need to use the equation $I_d = I_s \rho_d (\hat{s} \cdot \hat{m})$

Where ρ_d is the diffuse co-efficents of the material (0.4,1,0) and I_s is the intensity of the light source (0.9,0,0.2).

s is the normalized vector TO the light source so $\mathbf{s} = (0, 1, 4) - (0, 1, 2) = (0, 0, 2)$

Normalised we get:

$\hat{\mathbf{s}} = (0, 0, 1)$

$\hat{\mathbf{m}}$ is the normal which we also need to normalize so

$\hat{\mathbf{m}} = (0, 4/\sqrt{41}, 5/\sqrt{41}) = (0, 0.625, 0.781)$

$$\begin{aligned} I_d &= (0.4, 1, 0) * (0.9, 0, 0.2) * ((0, 0, 1) \cdot (0, 0.625, 0.781)) \\ &= (0.4, 1, 0) * (0.9, 0, 0.2) * 0.781 \\ &= (0.281, 0, 0) \end{aligned}$$

Question 2

(6 marks) Suppose you want a camera positioned at point (3,2,1) in world co-ordinates looking towards point (1,0,-1) without any rotation around the cameras z-axis.

(a) What would the camera's local coordinate frame be (expressed as a matrix)? *(3 marks)*

(b) What would the view matrix be for such a camera? *(2 marks)*

(c) Give the camera co-ordinates of a vertex with world co-ordinates of (-1,1,3). *(1 mark)*

Answer

(a) First we need to find our **k** vector.

This will be the position of the camera – the position it is looking at.

$(3, 2, 1) - (1, 0, -1) = (2, 2, 2)$

To get the **i** vector, we need to consider that the camera is not rotated around its z-axis. That means the i axis will be perpendicular to both k and the global y-axis which is (0,1,0).

$$\begin{aligned} \mathbf{i} &= (0, 1, 0) \times k \\ &= (0, 1, 0) \times (2, 2, 2) \\ &= (2, 0, -2) \\ \mathbf{j} &= k \times i \\ &= (2, 2, 2) \times (2, 0, -2) = (-4, 8, -4) \end{aligned}$$

We need to normalise these:

$$\begin{aligned}
\mathbf{i} &= (2/\sqrt{8}, 0, -2/\sqrt{8}) \\
&= (1/\sqrt{2}, 0, -1/\sqrt{2}) \\
\mathbf{j} &= (-4/\sqrt{96}, 8/\sqrt{96}, -4/\sqrt{96}) \\
&= (-1/\sqrt{6}, 2/\sqrt{6}, -1/\sqrt{6}) \\
\mathbf{k} &= (2/\sqrt{12}, 2/\sqrt{12}, 2/\sqrt{12}) \\
&= (1/\sqrt{3}, 1/\sqrt{3}, 1/\sqrt{3})
\end{aligned}$$

Putting these directly in a matrix:

$$\begin{pmatrix} 1/\sqrt{2} & -1/\sqrt{6} & 1/\sqrt{3} & 3 \\ 0 & 2/\sqrt{6} & 1/\sqrt{3} & 2 \\ -1/\sqrt{2} & -1/\sqrt{6} & 1/\sqrt{3} & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

(b) For the view matrix, we need the inverse of the above matrix. We could calculate that directly, but it is easier to just think of it as TR where

$$T = \begin{pmatrix} 1 & 0 & 0 & 3 \\ 0 & 1 & 0 & 2 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

And

$$R = \begin{pmatrix} 1/\sqrt{2} & -1/\sqrt{6} & 1/\sqrt{3} & 0 \\ 0 & 2/\sqrt{6} & 1/\sqrt{3} & 0 \\ -1/\sqrt{2} & -1/\sqrt{6} & 1/\sqrt{3} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Then we need to find the inverse which would be $R^{-1}T^{-1}$

The inverse of a rotation matrix is just the transpose.

$$R^{-1} = \begin{pmatrix} 1/sqrt(2) & 0 & -1/sqrt(2) & 0 \\ -1/sqrt(6) & 2/sqrt(6) & -1/sqrt(6) & 0 \\ 1/sqrt(3) & 1/sqrt(3) & 1/sqrt(3) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

And the inverse of the translation is just the negation.

$$T^{-1} = \begin{pmatrix} 1 & 0 & 0 & -3 \\ 0 & 1 & 0 & -2 \\ 0 & 0 & 1 & -1 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Which gives us

$$\begin{pmatrix} 1/sqrt(2) & 0 & -1/sqrt(2) & -2/\sqrt{2} \\ -1/sqrt(6) & 2/sqrt(6) & -1/sqrt(6) & 0 \\ 1/sqrt(3) & 1/sqrt(3) & 1/sqrt(3) & -6\sqrt{3} \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

(c)

$$\begin{pmatrix} 1/sqrt(2) & 0 & -1/sqrt(2) & -2/\sqrt{2} \\ -1/sqrt(6) & 2/sqrt(6) & -1/sqrt(6) & 0 \\ 1/sqrt(3) & 1/sqrt(3) & 1/sqrt(3) & -6\sqrt{3} \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} -1 \\ 1 \\ 3 \\ 1 \end{pmatrix} = \begin{pmatrix} -4.242 \\ 0 \\ -1.733 \\ 1 \end{pmatrix}$$

Part B: Short answer questions

Provide short 3-4 sentence answers to the following.

Question 3

(3 marks) What are BSP trees? Give one application for which they can be used? What problem do they solve in that application?

Answer

BSP trees are data structures that are used to partition objects within a space. One area where they can be used is for ray tracing. When used in ray tracing, they allow us to avoid testing for ray-intersection with every object in the scene. By dividing the scene up, we are able to only test against the objects in the subspaces that the ray travels through.

Question 4

(3 marks) What is the difference between a fragment shader and vertex shader? How do they relate?

Answer

Both vertex and fragment shaders are pieces of code that run on the GPU as part of the rendering process. The vertex shader is executed once for every vertex in the geometry being drawn. It compute the position of the vertex in CVV coordinates as well as any additional information that is passed on to the fragment shader. The fragment shader operates on every fragment that is produced by the rasterisation process. It must compute the color of the fragment. Values that are output by the vertex shader are interpolated before being passed into the fragment shader.

Question 5

(3 marks) What is trilinear filtering?

Answer

Trilinear filtering is a form of texture filtering. It performs a texture lookup and bilinear filtering on the two closest mipmap levels and then linearly interpolates the results. This helps to prevent abrupt changes in quality at boundaries where one mip-map level is switched to the next.

Part C: Design problems

Provide one paragraph answers to the following.

Question 6

(4 marks) You are applying for a job as a computer graphics expert. In the techincal interview they ask you what kind of modelling techniques you would use to model the shape and surface of a shiny metal teapot for a real-time game. Give reasons for your choices.

Answer

I would model the shape of the teapot using Bezier patches. Alternatively, the base and lid could be modeled as surfaces of revolution. The spout could possibly be modeled using extrusion. Assuming we want the teapot to be a metallic surface we would model it as being highly specular. We could possibly use environment mapping to get reflections on the teapot. Since we are modeling this for a real-time game, we probably would not use ray-tracing techniques that would make the teapot look more realistic, as we would be concerned about efficiency.

Question 7

(4 marks) You want to render a scene with soft shadows and realistic diffuse lighting. What technique/s would give the most realistic outcome? What are the pros and cons of this/these techniques?

Answer

I would use the global illumination approach called radiosity. The advantages of the approach is that it models indirect diffuse lighting and as a result can render soft shadows and realistic diffuse lighting. The disadvantage of radiosity is that it does not handle translucent, transparent or specular surfaces in the calculations. It is calculation intensive and too slow for real-time applications. It can be used in real-time applications to pre-compute lighting for portions of geometry that are static and rendered offline into textures. However, pre-computed lighting is not suitable for dynamic lights or moving objects.

— **End of exam** —