Past Paper Questions

(with answers !)

A note about solutions

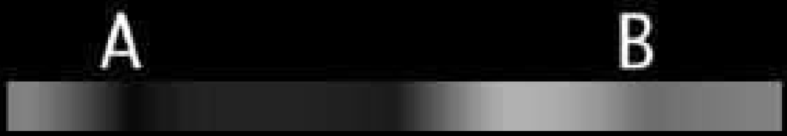
In the solutions to the past paper questions
We demonstrate the use of various different "tools"
(spreadsheets, programming languages, maths libs)

Aim is to illustrate the range of approaches possible
Don't feel constrained to using what we propose

Make use of whatever you feel comfortable with !

Colour Interpolation

There are two points `A` and `B` on a 1D linear colour spectrum, as shown in the diagram below:



The x position of `A` is 92 and it has an RGB colour of `[90, 90, 213]`
The x position of `B` is 479 and it has an RGB colour of `[180, 204, 22]`

What is the colour at x position 340, providing your answer in the form of separate Red, Green and Blue colour channels integers (0-255) each rounded to the nearest whole number.

Spreadsheet !

	A	B	diff	x=340
x position	92	479	387	340
Proportion	0	1	-	0.64
Red	90	180	90	148
Green	90	204	114	163
Blue	213	22	-191	91

Projection onto the Image Plane

Given the following situation:

- a vertex in 3D space at position `[22, -29, 31]`
- a camera positioned at `[0, 0, 160]` and oriented towards the world origin
- a focal length of 80
- an image plane of size 160 x 120 (width x height)

Calculate the [x,y] coordinates on the image plane at which the vertex should be drawn. Assume an SDL style image plane (i.e. origin at the top left corner) with no image plane scaling in use. Providing your answer rounded to the nearest 2 decimal places.

Using Pen and Paper !

distZ = | vertexZ - cameraZ | = | 31 - 160 | = 129

proportion = focalLength / distZ = 0.62

x = imageWidth/2 + (vertexX * proportion)

x = 160/2 + (22 * 0.62) = 80 + 13.64 = 93.64

y = imageHeight/2 - (vertexY * proportion)

y = 120/2 – (–29 * 0.62) = 60 + 17.98 = 77.98

Camera Orientation

The following is a camera orientation matrix (with elements rounded to 2 decimal places), where the first column is *right* the second column is *up* and the third column is *forward* aspects of the camera:

$$\begin{bmatrix} 0.67 & -0.02 & 0.75 \\ 0.0 & -1.01 & -0.02 \\ 0.75 & 0.01 & -0.67 \end{bmatrix}$$

What is the camera orientation matrix after a rotation of 44 degrees in the y axis, about the centre of the camera ?

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

Using C++

```
glm::mat3 original(0.67,0.0,0.75,
                   -0.02,-1.01,0.01,
                   0.75,-0.02,-0.67);
// C++ trig functions expect angle in radians !
float angle = 44.0 * (PI/180.0);
glm::mat3 rotation(cos(angle),0,-sin(angle),
                   0,1,0,
                   sin(angle),0,cos(angle));
glm::mat3 final = rotation * original;
std::cout << glm::to_string(final) << std::endl;
```

Output to 2DP (remember, GLM is Column Major !)

(1.0, 0.0, 0.07),(-0.01, -1.01, 0.02),(0.07, -0.02, -1.0)

Expected Exam Answer Format

Remember that the issue of column/row major
Relates to programming languages / maths libs
(it is the order you pass parameters into functions)
(when converting a 2D matrix into a 1D list)

In exam provide your answers as "normal" matrices:

```
1.0   -0.01   0.07
0.0   -1.01  -0.02
0.07   0.02  -1.0
```

(a column is a column and a row is a row !)

Reflection

Consider a 3D model containing a single triangle with vertices:
`[-446, -209, -102] [441, -152, -116] [-22, 404, -87]`

A single light point is positioned at `[11, -4, -18]`
Assuming that the surface of the triangle is a perfect mirror:

Give the normalised reflection vector of light bouncing off a point on the surface at `[-76, 0, -101]` rounding your answer to the nearest two decimal places.

Using Python !

```
e0 = v1-v0;
e1 = v2-v0;
normal = numpy.cross(e0, e1)
normal = normalise(normal)

incidence = normalise(pointOnSurface - positionOfLight);
reflection = incidence - (2 * normal * numpy.dot(incidence, normal))
reflection = normalise(reflection)
```

Get vectors for the two edges (e0 and e1)
Cross product to find normal (be sure to normalise !)

Incidence vector is from light to point-on-surface
Calculate reflection using formula from lecture

Solution vector should be: [-0.7, -0.02, 0.72]

Specular Lighting

Consider a 3D model containing a single triangle with vertices:
[-446, -209, -102] [441, -152, -116] [-22, 404, -87]

A single light point is positioned at [11, -4, -18]
Assuming that the surface of the triangle is a perfect mirror:

Give the normalised reflection vector of light bouncing off a point on the surface at [-76, 0, -101] rounding your answer to the nearest two decimal places.

Assuming that the light source has an intensity (brightness) of 1.0 and that the surface has a specular exponent of 141:

What would be the specular illumination brightness of the surface point when viewed from a camera at [-163, 4, -18] rounding your answer to the nearest two decimal places.

In Python

```
vectorToCamera = normalise(positionOfCamera - pointOnSurface);
brightness = pow(vectorToCamera.dot(reflectionVector), specularExponent)
```

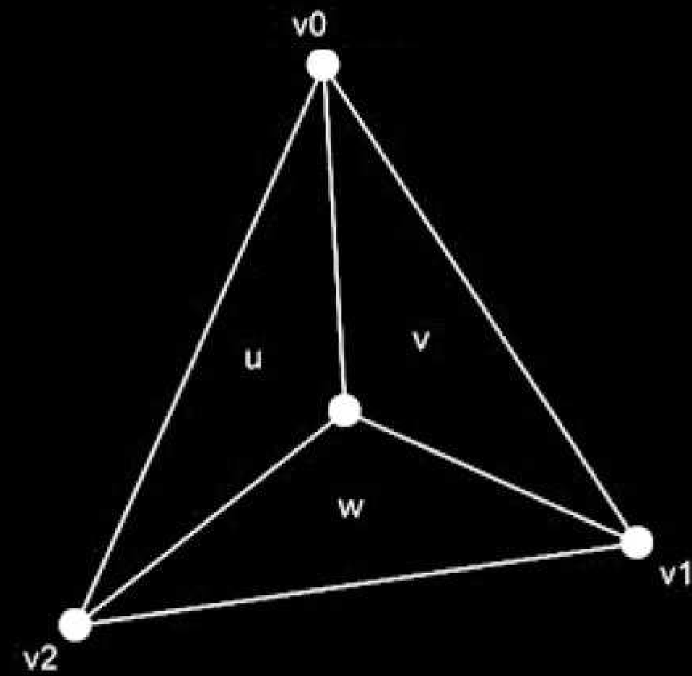
Calculate vector from point-on-surface to camera

Dot product with reflection vector (from previous)

To power of specular exponent (to control spot size)

brightness = 0.75

2D Colour Interpolation



The colours of the three vertices, expressed as [Red, Green, Blue] triples with a range of (0-255) are as follows:

- Colour of v0 is [201, 32, 39]
- Colour of v1 is [51, 227, 53]
- Colour of v2 is [33, 43, 200]

If these three colours were interpolated across the surface of the triangle, what would be the colour of the pixel given by: u=0.28 v=0.29 w=0.43 ? Provide your answer as a [Red, Green, Blue] triple with each channel rounded to the nearest whole number.

Spreadsheet !

	v0		v1		v2		Solution
	Value	Weight	Value	Weight	Value	Weight	
Red	201	0.43	51	0.28	33	0.29	110
Green	32	0.43	227	0.28	43	0.29	90
Blue	39	0.43	53	0.28	200	0.29	90