

**THE BCS PROFESSIONAL EXAMINATION**  
**Professional Graduate Diploma**

April 2001

**EXAMINERS' REPORT**

**Computer Graphics**

This three-hour paper required candidates to attempt three questions from a choice of five i.e. candidates should allow one hour to answer each question.

**QUESTION ONE**

a) **Most colour graphics display devices permit only a small subset of the colours in a palette to be displayed.**

1. **Why is this?** (3 marks)
2. **Explain how a Colour Look-Up Table (CLUT) can be used to relate the displayable colour set to the palette colour set.** (6 marks)
3. **A device is required to display up to 100 colours from a palette of 4000 colours. How small can the CLUT be made?** (6 marks)

b) **Explain how colour is physically produced on a raster graphics screen. You should make reference to electron guns, triads and shadow masks in your answer.** (10 marks)

**Answer Pointers**

This question is designed to test the candidates' knowledge and understanding of how colour is handled in the hardware of a graphics device.

a) 1. A large set of displayable colours (e.g. true colour) takes a lot of memory and processing time. In practice a much reduced set can provide adequate colour.

2. A CLUT is **indexed by displayable colours**. It **contains intensities for each of the primaries**.

A diagram of a CLUT should be provided.

The diagram should be accompanied by a textual description ...

If the rows are numbered  $0 - (2^m)-1$  there will be  $2^m$  displayable colours (with  $m$  bits/per pixel).

If there are  $k$  bits/channel and 3 channels (1 for each colour) then the palette size will be  $2^{(3*k)}$ .

An explanation of what one example from the entries in the table means in terms of RGB should also be given.

3. Palette size = 4000, so need to find the smallest  $k$  such that  
 $2^{(3*k)} \geq 4000$

Therefore  $k=4$

Displayable colours = 100, so need to find the smallest  $m$  such that  
 $2^m \geq 100$

Therefore  $m=7$

Therefore the CLUT size is  $(2^m) * 3*k = 3*(2^9) = \underline{1.5 \text{ Kbits}}$

- b) There is 1 electron gun for each colour  
 Intensity of each colour determined by power of its gun  
 Required intensity derived from CLUT  
 (Diagram of an electron gun expected – highly simplified)

Colour is actually in the phosphor on the screen  
 Pixels are made up of triads  
 Triads consist of 3 coloured phosphor dots – 1 for each primary  
 (Diagram of a triad expected here)

For a given pixel each gun hits its appropriate triad member (R, G or B)  
 Guns prevented from interfering with nearby dots by shadow mask  
 Shadow mask is a grill located just behind the phosphor  
 (Diagram of a shadow mask expected here)

### Marks Breakdown

- |       |                                      |                   |
|-------|--------------------------------------|-------------------|
| a) 1. | Each point                           | (1 mark)          |
| 2.    | Each point                           | (1 mark)          |
|       | Diagram of CLUT                      | (2 marks)         |
|       | Each point in description of diagram | (1 mark)          |
| 3.    | Formula for palette size             | (2 marks)         |
|       | Formula for displayable colours      | (2 marks)         |
|       | Formula for CLUT size                | (2 marks)         |
| b)    | Each point                           | (1 mark up to 10) |

## QUESTION TWO

- a) Describe the three key characteristics which can be determined from a light energy spectrum. (6 marks)
- b) What is a colour gamut? From what limitation do all colour gamuts suffer in practice? (4 marks)
- c) A colour is defined as (0.9, 0.4, 0.4) in the RGB model. What are its components in the CMYK model? (4 marks)
- d) Briefly describe the three main components in Phong's illumination model. (6 marks)
- e) What additional light sources can be added to Phong's model to improve visual realism and how can they be taken into account? (5 marks)

## Answer Pointers

This question is designed to test the candidates' knowledge and understanding of colour and illumination models.

- a) The **dominant wavelength**  
This is the main colour, the wavelength with the greatest energy  
*Graph of energy spectrum (highest peak identified) expected here*

The **saturation**  
This is a measure of the purity of the dominant colour

$$\text{Saturation} = (1 - (e1/e2)) * 100\%$$

*e1* is the energy of the background spectrum  
*e2* is the energy of the wavelength whose purity is being measured

The **luminance**  
This is the total energy of the entire spectrum  
*Graph of energy spectrum (area under graph identified) expected here*

- b) A colour gamut is the **complete palette of colours** which can be **generated by a colour model** (e.g. RGB, CMY, etc.).

In practice **no gamut which is derived from 3 primaries is as extensive as the palette discernible by the human eye**. The **CIE gamut is derived from imaginary primaries**, not real ones, and so is not practical.

- c) CMY are the complements of RGB

$$\begin{array}{ll} R = 0.9 & G = 0.4 & B = 0.4 \\ \text{So} & C = 1 - R = 0.1 & M = 1 - G = 0.6 & Y = 1 - B = 0.6 \\ \text{Now} & K = \min(C, M, Y) = 0.1 \\ \text{Therefore} & C = C - K = 0.0 & M = M - K = 0.5 & Y = Y - K = 0.5 \end{array}$$

CMYK components are (0.0, 0.5, 0.5, 0.1)

- d) **Ambient illumination:**  
This is the **background luminance** that falls on all surfaces

**Diffuse reflection:**  
This is light which is reflected in **all directions** from a matt surface

**Specular reflection:**  
This is the **direction specific** reflection obtained from a glossy surface

- e) Light which emanates from other surfaces within a model can be added.  
This is global illumination as opposed to local illumination

Global illumination is modelled by one of two means

**Ray tracing:**  
Rays are fired from the viewpoint into the model  
Where a ray intersects an object the illumination at that point is determined

**Radiosity:**  
Radiosity is a measure of the energy leaving a surface  
There are two main components, reflected energy and emitted energy  
Each surface patch has its own iterative radiosity equation which combines the effects of other patches

### Marks Breakdown

- |    |  |           |
|----|--|-----------|
| a) | Dominant wavelength                              | (2 marks) |
|    | Saturation                                       | (2 marks) |
|    | Luminance  | (2 marks) |
| b) | Each point                                       | (1 mark)  |
| c) | Each component                                   | (1 mark)  |
| d) | Each component                                   | (2 marks) |
| e) | Difference between global and local illumination | (1 mark)  |
|    | Description of Ray Tracing                       | (2 marks) |
|    | Description of Radiosity                         | (2 marks) |

### QUESTION THREE

A drawing package requires the implementation of scan conversion algorithms to display primitive shapes. Your answers to each section of this question should identify any limitations of algorithms. Implementations can be given in your preferred programming language. Assume that you have been provided with a function write(x, y) for displaying a point on the screen.

- a) Describe the mid-point line algorithm for scan converting straight lines between the points (x1, y1) and (x2, y2), giving any restrictions which the algorithm places on the end points (13 marks)
- b) Describe how the algorithm can be extended to scan convert lines between all possible choices of end points (12 marks)

The solution to part (a) should have been in the form of a subroutine, coded in any programming language, which implements the mid-point algorithm. As the normally stated algorithm only works for lines in a certain range of gradient, the constraints on the end points should have been given in terms of x1, y1, m, x2 and y2.

The solution to part (b) should have addressed each of the 8 gradient ranges which need to be treated separately. For each range, state the required changes, such as change increment x to decrement x.

### Answer Pointers

- a) The mid-point line algorithm assumes that the gradient of the line is between 0 and 1. Hence  $x_1 < x_2$ ;  $y_1 < y_2$  and  $(x_2 - x_1) \leq (y_2 - y_1)$ .

The algorithm works by either moving one pixel in x or one pixel in both x and y depending on which is nearer the true line.

```
midpoint(x1, y1, x2, y2)
{
    dx = x2 - x1 ;
    dy = y2 - y1 ;
    d = 2 * dy - dx ; // the current error
    incE = 2 * dy ; // the horizontal (E) move error increment
    incNE = 2 * ( dy - dx ) ; // the NE move error increment

    x = x1 ; // start x
    y = y1 ; // start y

    write(x, y) ;

    while ( x < x2 )
    {
        if ( d <= 0 )
        {
            d += incE ;
            x++ ;
        }
    }
```

```

else
{
    d += incNE ;
    x++ ; y++ ;
}
write(x, y) ;
}
}

```

- b)** The algorithm can be extended for lines with gradients outside the range [0, 1] by suitable reflection about the major axes as follows:
1. Gradient in the range 0-45 degrees, use the standard mid-point line algorithm: increment x or increment both x and y.
  2. Gradient in the range 45-90 degrees, use the mid-point line algorithm: increment y or increment both x and y.
  3. Gradient in the range 90-135 degrees, use the mid-point line algorithm: increment y or decrement x and increment y.
  4. Gradient in the range 135-180 degrees, use the mid-point line algorithm: decrement x or decrement x and increment y.
  5. Gradient in the range 180-225 degrees, use the mid-point line algorithm: decrement x or decrement both x and y.
  6. Gradient in the range 225-270 degrees, use the mid-point line algorithm: decrement y or decrement both x and y.
  7. Gradient in the range 270-315 degrees, use the mid-point line algorithm: decrement y or increment x and decrement y.
  8. Gradient in the range 315-360 degrees, use the mid-point line algorithm: increment x or increment x and decrement y.

### Answer Pointers

- |           |   |             |
|-----------|---|-------------|
| <b>a)</b> | Formula for algorithm                       | (2 marks)   |
|           | Description of how algorithm works          | (3 marks)   |
|           | Sub-routine                                 | (8 marks)   |
| <b>b)</b> | Each of the above suggestions or equivalent | (1.5 marks) |

### QUESTION FOUR

**A three-dimensional graphics package makes use of a number of transformations to manipulate primitive shapes.**

**In each of the following, draw a diagram to show the effects of the transformations in question and derive the transformation matrix.**

- a) Give the matrices for a rotation of angle  $\theta$  about the x-axis and for a rotation of angle  $\phi$  about the y-axis.

Multiply the matrices together to give a composite rotation matrix

(9 marks)

- b) An arbitrary unit direction vector  $U = (u_1, u_2, u_3)$  can be rotated into the positive z-axis by applying a rotation about the y-axis followed by a rotation about the x-axis.

By multiplying the composite matrix derived in part a) by the vector  $U$ , and equating it with a unit vector along the positive z-axis, generate three equations in  $\theta$  and  $\phi$ , the solutions of which define the composite rotation matrix from  $U$  onto the positive z-axis

(5 marks)

- c) Solve the three equations to give the sines and cosines of  $\theta$  and  $\phi$  in terms of  $u_1, u_2$  and  $u_3$

(11 marks)

The answer to a) should have stated the two 3x3 rotation matrices. The matrices should have been multiplied to give a composite 3x3 matrix. Note that the multiplication order is important.

Following on from this, the composite 3x3 matrix from part a) should be multiplied by  $U$  to give a 3x1 matrix. Equating this with a unit vector along z gives three simultaneous equations in  $\theta, \phi, u_1, u_2$  and  $u_3$ .

Finally, the answer should have taken the three equations derived in part b) and solved them using the methods of trigonometry and simultaneous equations. This usually involves taking pairs of equations and eliminating one of the  $u$  terms.

### Answer Pointers

a)

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{pmatrix} \begin{pmatrix} \cos\phi & 0 & -\sin\phi \\ 0 & 1 & 0 \\ \sin\phi & 0 & \cos\phi \end{pmatrix} = \begin{pmatrix} \cos\theta \cos\phi & 0 & \sin\theta \cos\phi \\ -\sin\theta \sin\phi & \cos\theta & -\sin\theta \cos\phi \\ \cos\theta \sin\phi & \sin\theta & \cos\theta \cos\phi \end{pmatrix}$$

b)

$$\begin{aligned} u_1 \cos\phi - u_3 \sin\phi &= 0 \\ -u_1 \sin\theta \sin\phi + u_2 \cos\theta - u_3 \sin\theta \cos\phi &= 0 \\ u_1 \cos\theta \sin\phi + u_2 \sin\theta + u_3 \cos\theta \cos\phi &= 1 \end{aligned}$$

c)

$$\begin{aligned} u_1 &= \cos\theta \sin\phi \\ u_2 &= \sin\theta \\ u_3 &= \cos\theta \cos\phi \end{aligned}$$

### QUESTION FIVE

**A drawing package needs to be able to define and efficiently display Bézier curves. Your answers to each section should include a diagram where appropriate.**

- a) Describe the Bézier form of the cubic polynomial curve segment. (4 marks)**
- b) Describe the iterative evaluation method for displaying a parametric cubic and identify the number of multiplications and additions required for each point. (6 marks)**
- c) Show how the use of forward differences can be used to reduce the number of calculations and identify how many multiplications and additions are required for each point. (7 marks)**
- d) Describe the recursive sub-division method for displaying a parametric cubic. (6 marks)**
- e) Give a suitable flatness test for a Bézier curve segment undergoing recursive sub-division. (2 marks)**

The answer to part a) should have described how each of the four control points affect the shape of the curve. It should identify which control points the curve passes through and how the others work. A diagram should have been drawn to illustrate this.

For part b), the answer should have shown how the cubic polynomial can be rewritten using Horner's rule to reduce the number of calculations. The total number of calculations for iterating over the x, y and z co-ordinates should have been stated.

In part c), the answer should have shown how forward differences can be used to generate a curve by calculating the increments from one point to the next rather than calculating each point individually. Again, a calculation count should have been stated.

Next, rather than choosing an increment value for the spline parameter  $t$ , the line segments can be recursively split into two until a line segment meets a criterion for being considered a straight line. Segments are only displayed when they meet the criterion.

Finally, the answer should have specified suitable constraints on the Bezier curve's control points which would identify which curves meet the criterion for being considered a straight line for display purposes.

### **Answer Pointers**

- a)** A Bézier curve is defined by four control points  $P_1, P_2, P_3, P_4$ . The curve passes through the points  $P_1$  and  $P_4$  and in general does not pass through points  $P_2$  and  $P_3$ . The end point tangents are defined by  $(P_1, P_2)$  and  $(P_4, P_3)$  respectively.
- b)** The iterative approach involves evaluating the polynomial for a range of values of the control parameter  $t$ .  
Using Horner's rule the polynomial  $at^3 + bt^2 + ct + d$  can be more efficiently evaluated as  $((at + b)t + c)t + d$  this gives 3 multiplications and 3 additions.  
Three points and an increment of  $t$  gives 9 multiplications and 10 additions per point.



- c) Forward difference use the function  $\Delta f(t) = f(t+\delta) - f(t)$ ;  $\delta > 0$

$$f(t) = at^3 + bt^2 + ct + d$$

$$\Delta f(t) = 3at^2\delta + t(3a\delta^2 + 2b\delta) + a\delta^3 + b\delta^2 + c\delta = At^2 + Bt + C = (At+B)t+C$$

This involves some initial calculation and 2 multiplications and 3 additions per coordinate, 6 multiplications and 9 additions per point.

- d) Recursive subdivision works by recursively dividing a curve until the curve segments are effectively straight line segments based on some error threshold e.

```
drawcurve(curve, e)
{
  if (straightline(curve, e) )
  {
    draw(curve);
  }
  else
  {
    dividecurve(curve, left, right);
    drawcurve(left, e);
    drawcurve(right, e);
  }
}
```

- e) A Bézier curve can be considered for flatness by examining the convex hull of its control points.

Draw a line between P1 and P4. If the length of the perpendiculars from P2 to the line and from P3 to the line are both less than e, then the curve segment can be considered flat.