

EXAMINATIONS – 2019

TRIMESTER 2

<p>CGRA 151 INTRODUCTION TO COMPUTER GRAPHICS</p>

Time Allowed: TWO HOURS

CLOSED BOOK

Permitted materials: Silent non-programmable calculators or silent programmable calculators with their memories cleared are permitted in this examination.

Printed foreign language–English dictionaries are permitted.

No other material is permitted.

Instructions: Attempt ALL questions.

The examination will be marked out of 120 marks.

Brief *Processing* documentation is provided on page 19, which may be detached from the question paper.

Answer in the appropriate boxes if possible — if you write your answer elsewhere, make it clear where your answer can be found.

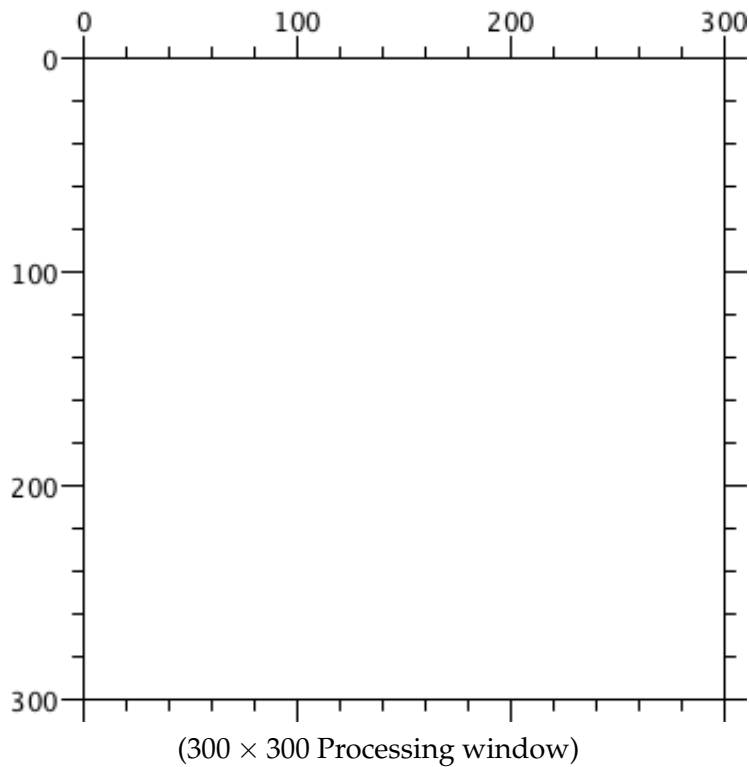
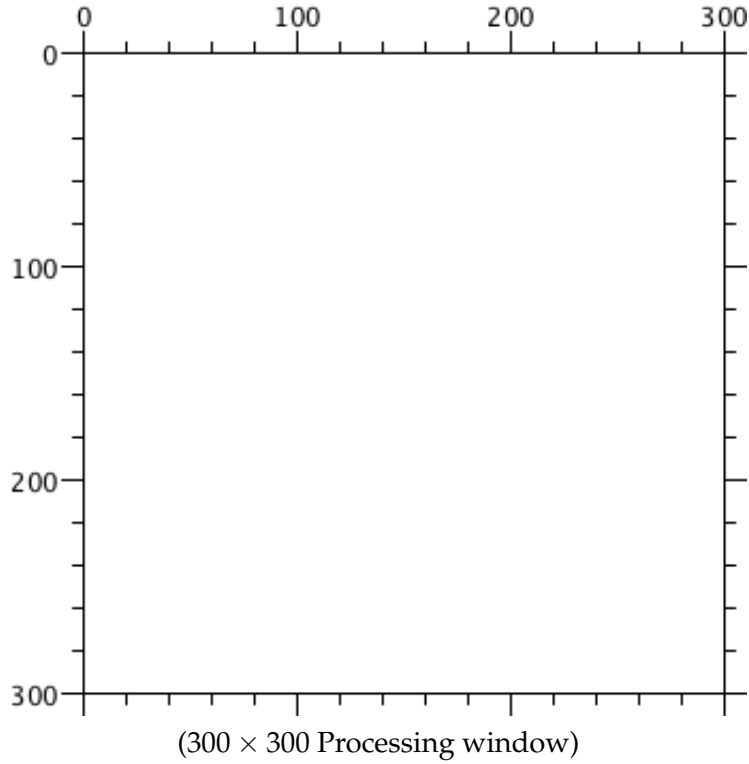
There are spare pages for your working and your answers in this question paper. However, you may ask for additional paper if you need it.

Questions

	Marks
1. Programming in Processing	30
2. Graphics algorithms	30
3. Colour, displays, and human vision	30
4. Mathematics for graphics	30
Total	<hr/> 120 <hr/>

SPARE PAGE FOR EXTRA ANSWERS

Cross out rough working that you do not want marked.
Specify the question number for work that you do want marked.



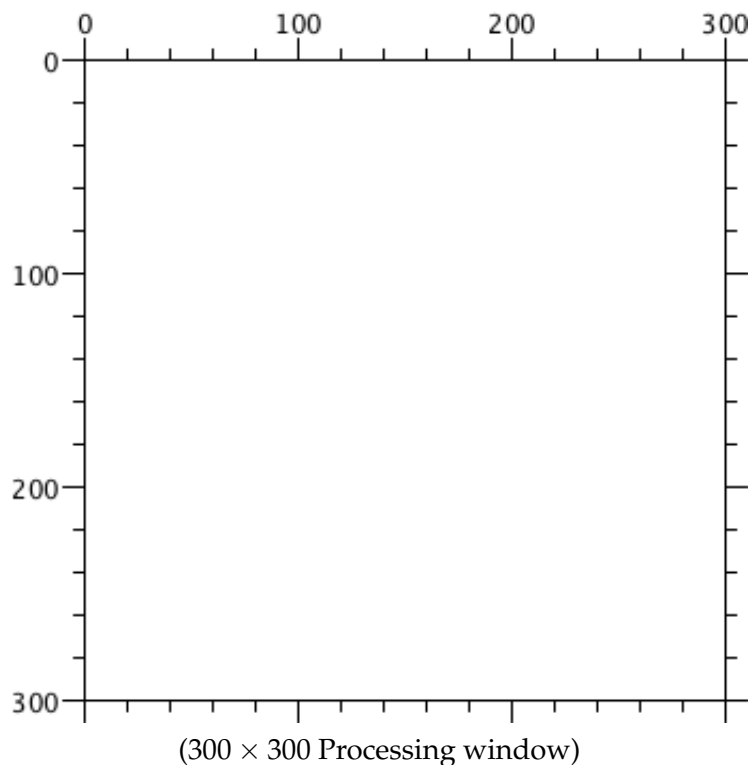
Question 1. Programming in Processing**[30 marks]**(a) **[15 marks]** Interpreting Processing Code

Sketch what the following code will draw in the Processing window.

```

size(300, 300);
background(255);
stroke(0);
noFill();
ellipse(140, 40, 40, 40);
line(120, 100, 160, 100);
for (int i = 0; i < 40; i += 20) {
    rect(80 + i, 220 - i, 120 - (i * 2), 20);
}
beginShape();
    vertex(100, 200);
    vertex(120, 100);
    vertex(100, 60);
    vertex(180, 60);
    vertex(160, 100);
    vertex(180, 200);
endShape();
rect(20, 20, 40, 40);
translate(200, 0);
rect(20, 20, 40, 40);
translate(0, 200);
rect(20, 20, 40, 40);

```

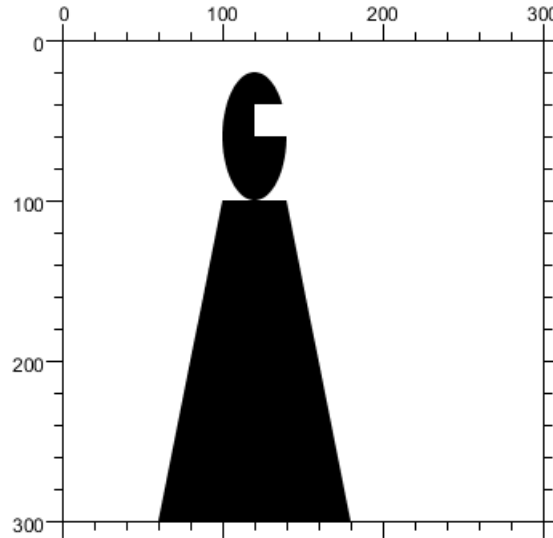


(Question 1 continued on next page)

(Question 1 continued)

(b) [15 marks] Writing Processing Code

(i) (8 marks) Write processing code to draw the following in the Processing window.



(300 × 300 Processing window)

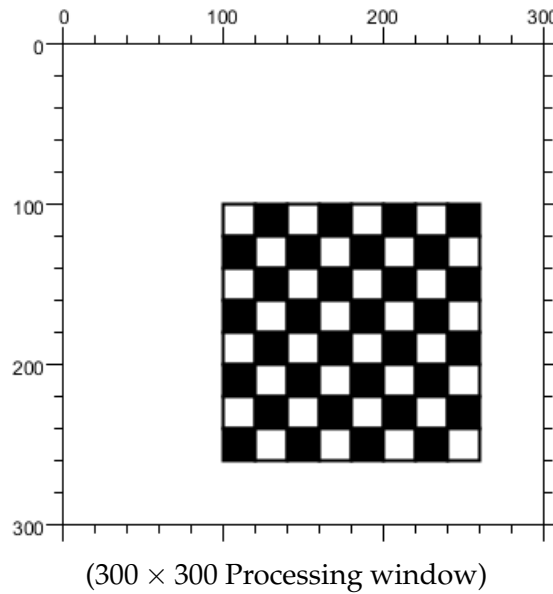
```
size(300, 300);  
background(255);
```

```
//
```

(Question 1 continued on next page)

(Question 1 continued)

(ii) (7 marks) Write processing code to draw the following in the Processing window. For full credit you must use loops appropriately.



```
size(300, 300);  
background(255);
```

```
//
```

Student ID:

SPARE PAGE FOR EXTRA ANSWERS

Cross out rough working that you do not want marked.
Specify the question number for work that you do want marked.

Question 2. Graphics Algorithms**[30 marks]**

(a) **[10 marks]** Please complete the following algorithm for drawing a straight line that uses integer end points under the following assumptions:

- the line starts at (x_0, y_0) , ends at (x_1, y_1) , and $x_0 < x_1$.
- the slope m of the line is between 0° and 45° .
- the function `drawPixel(x,y)` turns on the pixel at integer location (x, y)
- the function `ROUND(f)` returns the closest integer of a floating point f .

```

void drawLine( int x0, int y0, int x1, int y1 ) {

    // Initialize the slope value

    float m = _____;

    // Get the x and y coordinate of the starting pixel

    int x = _____;

    float yi = y0;

    int y = _____;

    // Draw the line , one pixel at a time

    while (x < x1) {

        DrawPixel(_____);

        x = _____;

        yi = _____;

        y = _____;

    }

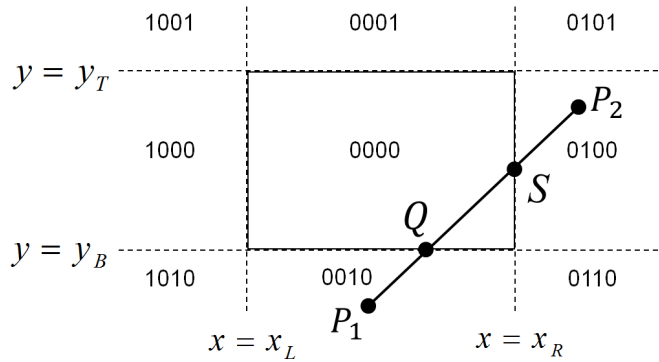
}

```

(Question 2 continued on next page)

(Question 2 continued)

(b) [10 marks] In Cohen-Sutherland's line clipping algorithm, every line endpoint is assigned a 4 bit region code, as shown in the diagram below. The infinite edges of the window to clip against are $x = x_L, x = x_R, y = y_T, y = y_B$.

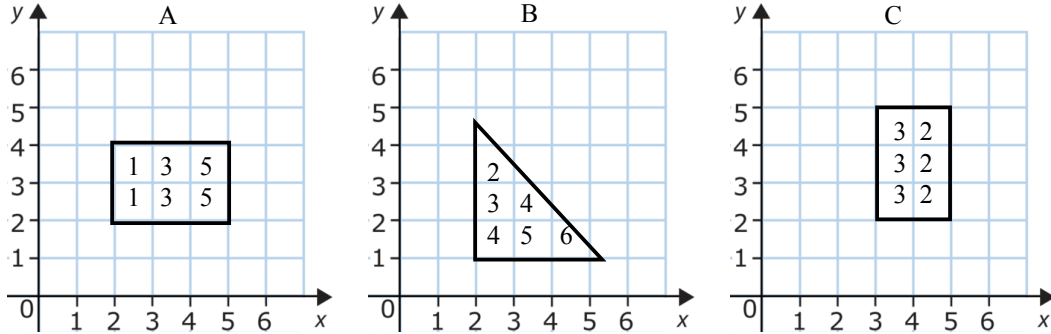


(i) (6 marks) Explain how the region codes are used to decide whether clipping is needed for straight line P_1P_2 and, if so, explain the whole procedure of using the region codes to choose which infinite edge to clip against until no further clipping is needed. Note that the point lying on the edges will be treated as INSIDE, where the corresponding code will be 0.

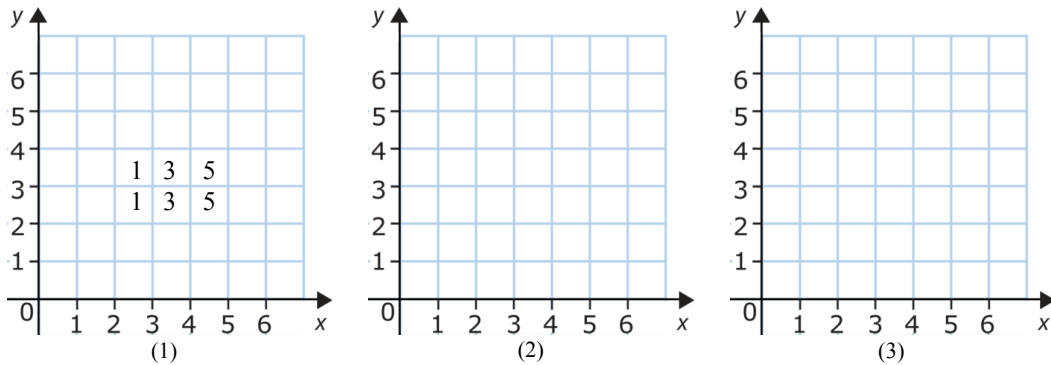
(ii) (4 marks) If $P_1 = (x_1, y_1)$ and $P_2 = (x_2, y_2)$, please show how to calculate the coordinates of the intersection point S between P_1P_2 and $x = x_R$.

(Question 2 continued)

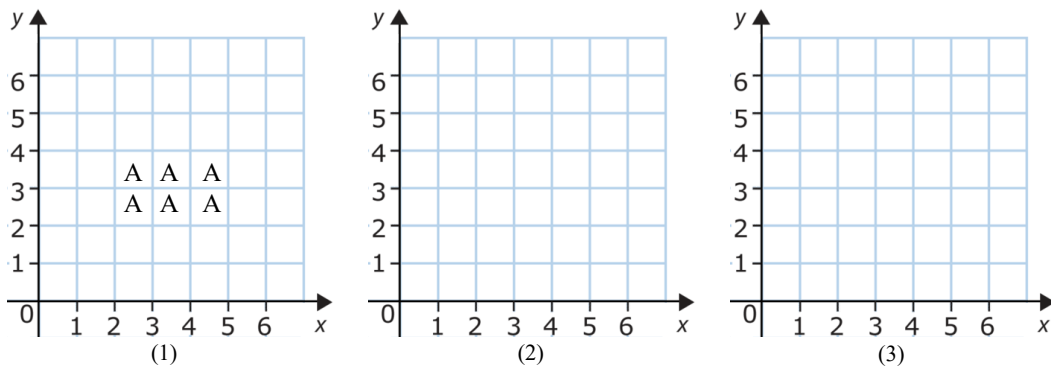
(c) [10 marks] In Z-buffer algorithm, suppose you are given 3 polygons *A*, *B* and *C* and their individual depth values are as follows:



During the process of Z-buffer algorithm, we convert one polygon at a time and update the depth and colour value for each pixel of the output. In the following graphs, the depth values and which polygon's colour should be used for each pixel after scan converting have been filled for polygon *A* (1). Please fill in the depth value and which polygon's colour should be used for each pixel grid after converting *B* (2) and *C* (3).



(i) (4 marks) Final depth map

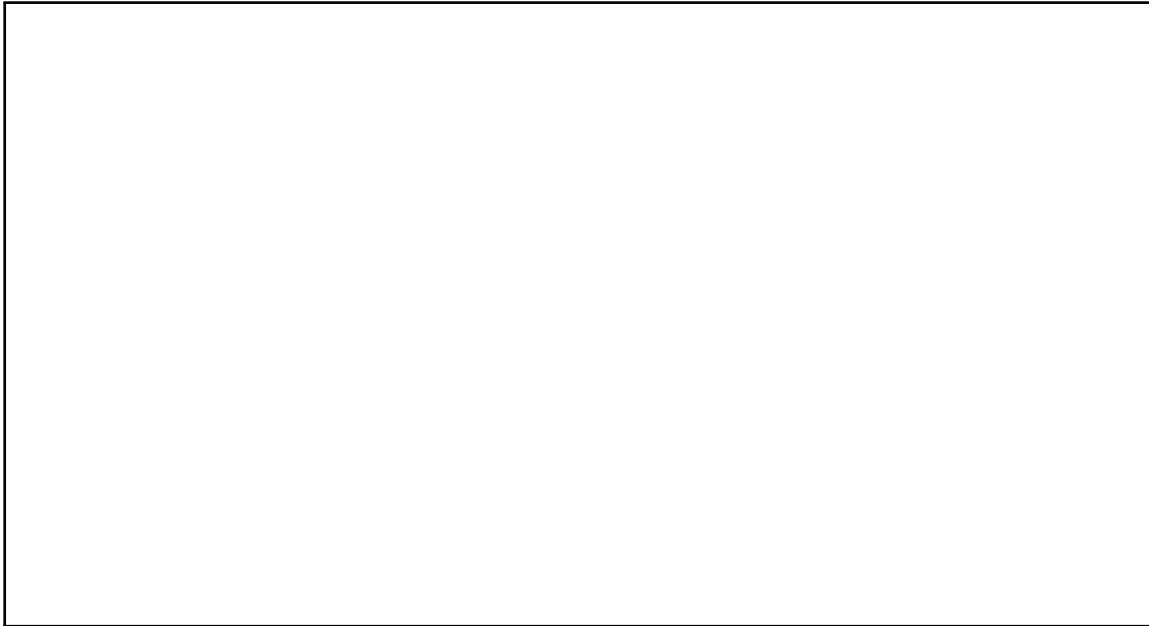


(ii) (2 marks) Final colour map

(Question 2 continued on next page)

(Question 2 continued)

(iii) (4 marks) What are the advantages of using the Z-buffer algorithm compared with the BSP tree algorithm.




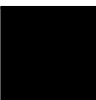
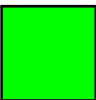
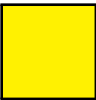
Question 3. Colour, Displays, and Human Vision

[30 marks]

(a) [9 marks] Colour spaces

The human eye has three types of colour receptors and therefore there are a number of three-dimensional co-ordinate systems that can be used to represent colour. Estimate the co-ordinates of colours for the following colour systems (value ranges are given in square brackets [minimum, maximum]):

- **RGB:** Red [0, 255], Green [0, 255], Blue [0, 255]
- **HSV:** Hue [0, 360], Saturation [0, 100], Value [0, 100]
- **CMY:** Cyan [0, 100], Magenta [0, 100], Yellow [0, 100]

	<i>RGB</i>	<i>HSV</i>	<i>CMY</i>
 White	$R = 255$ $G = 255$ $B = 255$	$H = \text{any } [0, 360]$ $S = 0$ $V = 100$	$C = 0$ $M = 0$ $Y = 0$
 Black	$R =$ $G =$ $B =$	$H =$ $S =$ $V =$	$C =$ $M =$ $Y =$
 Green	$R =$ $G =$ $B =$	$H =$ $S =$ $V =$	$C =$ $M =$ $Y =$
 Yellow	$R =$ $G =$ $B =$	$H =$ $S =$ $V =$	$C =$ $M =$ $Y =$

(Question 3 continued on next page)

(Question 3 continued)

(b) **[6 marks]** Explain why Cyan, Magenta, and Yellow are the three primary colours for printing.

(c) **[3 marks]** Explain why printers might use Cyan, Magenta, and Yellow (CMY) as primary colours instead of the Red, Yellow, Blue (RYB) primaries traditionally taught in art schools.

(Question 3 continued)

(d) **[6 marks]** Assume you have a phone/tablet with a screen size of 25.6×14.4 cm with a pixel density of 50 pixels per cm. Use your knowledge of the limitations of the human vision to calculate whether the phone's resolution exceeds the maximum resolving power of the human eye (which is $\frac{1}{60}$ of a degree), if it is held 30 cm away from your eyes. You can assume that the pixels are square. Show your working.

(e) **[6 marks]** Explain why colour discrimination becomes worse the further away an image is from the centre of your vision.

Student ID:

SPARE PAGE FOR EXTRA ANSWERS

Cross out rough working that you do not want marked.
Specify the question number for work that you do want marked.

Question 4. Mathematics for graphics**[30 marks]**

(a) [10 marks] Vector and matrix algebra

Given three vectors, $\mathbf{p} = \begin{bmatrix} 1 \\ 2 \\ -2 \end{bmatrix}$, $\mathbf{q} = \begin{bmatrix} 3 \\ 0 \\ 4 \end{bmatrix}$ and $\mathbf{r} = \begin{bmatrix} 4 \\ 1 \\ 3 \end{bmatrix}$
 and two matrices, $\mathbf{M}_1 = \begin{bmatrix} 0 & -1 & 3 \\ 1 & 0 & 5 \\ 0 & 0 & 1 \end{bmatrix}$ and $\mathbf{M}_2 = \begin{bmatrix} 1 & 0 & -1 \\ 0 & 1 & -2 \\ 0 & 0 & 1 \end{bmatrix}$,

evaluate the following expressions.

Magnitude

$|\mathbf{p}| =$

Dot product

$\mathbf{p} \cdot \mathbf{r} =$

Angle between \mathbf{p} and \mathbf{r}

$\theta =$

Matrix addition

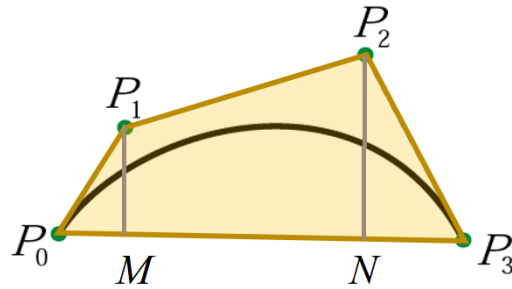
$\mathbf{M}_1 + \mathbf{M}_2 =$

Matrix multiplication

$\mathbf{M}_1\mathbf{M}_2 =$

(Question 4 continued)

(b) [10 marks] The diagram below is a piece of Bezier curve, which is defined by 4 known control points: $P_0(x_0, y_0)$, $P_1(x_1, y_1)$, $P_2(x_2, y_2)$, and $P_3(x_3, y_3)$. M , and N are the projections of P_1 and P_2 on the straight line P_0P_3 . Explain, through words, or diagrams, or otherwise, how to compute the coordinates of M and N based on the known coordinates, and how they are used to check the flatness of this Bezier curve.

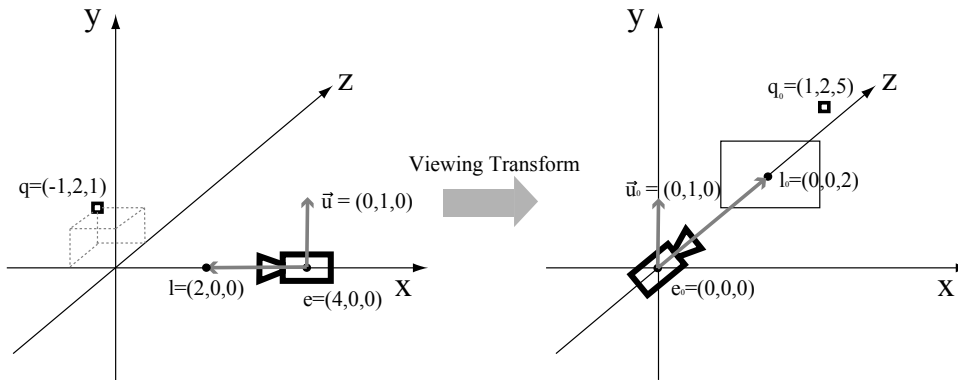


(Question 4 continued on next page)

(Question 4 continued)

(c) [10 marks] Given such a 3D point $q = (-1, 2, 1)$ in the world coordinate system, and a camera of which the centre point is $e = (4, 0, 0)$ and the up vector is $u = (0, 1, 0)$. The camera is looking at $l = (2, 0, 0)$. After viewing transform, the coordinates are transformed to a coordinate system where the camera is a standard camera, with an image plane centred at $l_0 = (0, 0, 2)$, as shown in the right hand graph.

The transformed coordinates of q in the viewing coordinates are given: $q_0 = (1, 2, 5)$. Please 1) calculate the projected 2D coordinates of q_0 on the 2D image plane; 2) describe the transformation steps needed to convert the world coordinates to the viewing coordinates for the given camera, and show the single matrix for the viewing transform.



Student ID:

SPARE PAGE FOR EXTRA ANSWERS

Cross out rough working that you do not want marked.
Specify the question number for work that you do want marked.

Brief Processing documentation

You may detach this page from the question paper.

```

void point( float x, float y )
// draw a point at location (x,y)

void line( float x1, float y1, float x2, float y2 )
// draw a line segment from location (x1,y1) to location (x2,y2)

void rect( float left, float top, float width, float height )
// draw a rectangle with its top left corner at the indicated position
// of size width times height

void ellipse( float centerX, float centerY, float width, float height )
// draw an ellipse centred at the indicated position
// of size width times height

void triangle( float x1, float y1, float x2, float y2, float x3, float y3 )
// draw a triangle with vertices at the three points
// (x1,y1), (x2,y2), (x3, y3)

void beginShape()
// start a shape which comprises a sequence of polygon vertices

void vertex( float x, float y )
// specify a vertex in the current shape

void endShape( mode )
// end and draw a shape. If the mode is "CLOSE" then the final point
// will be connected to the initial point. If the mode is missing then
// the shape will be left open.

void bezier( float x1, float y1, float x2, float y2,
             float x3, float y3, float x4, float y4 )
// draw a Bezier cubic curve with the four control points
// (x1,y1), (x2,y2), (x3, y3), (x4, y4)

void background( float grey )
void stroke( float grey )
void fill( float grey )
// set the colour of the background, stroke or fill to be
// a grey value between 0 (black) and 255 (white)

void noStroke()
void noFill()
// turn off drawing strokes around objects and filling objects , respectively

```
