USN □□□□□□□□□□

CELEBRATING 25 YEARS
CMRIT
CMR INSTITUTE OF TECHNOLOGY, BENGALURU.
ACCREDITED WITH A+ GRADE BY NAAC

Internal Assessment Test 2 – July 2024

Sub:	Computer Graphics and Fundamentals of Image Processing	Sub Code:	21CS63	Branch:	CSE

Date:	09.07.2024	Duration:	90 mins	Max Marks:	50	Sem/Sec:	6 A, B, C	OBE

	Answer any FIVE FULL Questions	MARKS	CO	RBT
1 (a)	List the picking techniques used in computer graphics and explain **any one** in detail. **Solution:** **Cursor-positioning approach:** It is a pick procedure, which could map a selected screen position to a world-coordinate location using the inverse viewing and geometric transformations that were specified for the scene. Then, the world coordinate position can be compared to the coordinate extents of objects. If the pick position is within the coordinate extents of a single object, the pick object has been identified. The object name, coordinates, or other information about the object can then be used to apply the desired transformation or editing operations. But if the pick position is within the coordinate extents of two or more objects, further testing is necessary. Depending on the type of object to be selected and the complexity of a scene, several levels of search may be required to identify the picked object. For example, if we are attempting to pick a sphere whose coordinate extents overlap the coordinate extents of some other three-dimensional object, the pick position could be compared to the coordinate extents of the individual surface facets of the two objects. If this test fails, the coordinate extents of individual line segments can be tested. When coordinate-extent tests do not uniquely identify a pick object, the distances from the pick position to individual line segments could be computed. For a two-dimensional line segment with pixel endpoint coordinates (x1, y1) and (x2, y2), the perpendicular distance squared from a pick position (x, y) to the line is calculated as d2 = [{x(y − y1) − y(x − x1)}^2] / x^2 + y^2 **Pick window:** Another picking technique is to associate a pick window with a selected cursor position. The picking window is centered on the cursor position, as shown in Figure 2, and clipping procedures are used to determine which objects intersect the pick window. For line picking, we can set the pick-window dimensions w and h to very small values, so that only one line segment intersects the pick window. Some graphics packages implement three-dimensional picking by reconstructing a scene using the viewing and projection transformations with the pick window as the clipping window. Nothing is displayed from this reconstruction, but clipping procedures are applied to determine which objects are within the pick view volume. A list of information for each object in the pick view volume can then be returned for processing. This list can contain information such as object name and depth range, where the depth range could be used to select the nearest object in the pick view volume. **Highlighting:** It can also be used to facilitate picking. One way to do this is to successively highlight those objects whose coordinate extents overlap a pick position (or pick window). As each object is highlighted, a user could issue a “reject” or “accept” action using keyboard keys. The sequence stops when the user accepts a highlighted object as the pick object. Picking could also be accomplished simply by successively highlighting all objects in the scene	5 M	CO3	L1

	without selecting a cursor position. The highlighting sequence can be initiated with a button or function key, and a second button can be used to stop the process when the desired object is highlighted. If very many objects are to be searched in this way, additional buttons can be used to speed up the highlighting process. One button initiates a rapid successive highlighting of structures. A second button is activated to stop the process, and a third button is used to back up slowly through the highlighting process. Finally, a stop button could be pressed to complete the pick procedure.			
(b)	Discuss how consistency, backup, and error handling are maintained in GUI. **Solution:** **Consistency:** An important design consideration in an interface is consistency. An icon shape should always have a single meaning, rather than serving to represent different actions or objects depending on the context. Some other examples of consistency are always placing menus in the same relative positions so that a user does not have to hunt for a particular option, always using the same combination of keyboard keys for an action, and always using the same color encoding so that a color does not have different meanings in different situations. **Backup and Error Handling:** A mechanism for undoing a sequence of operations is another common feature of an interface, which allows a user to explore the capabilities of a system, knowing that the effects of a mistake can be corrected. Typically, systems can now undo several operations, thus allowing a user to reset the system to some specified action. For those actions that cannot be reversed, such as closing an application without saving changes, the system asks for a verification of the requested operation. In addition, good diagnostics and error messages help a user to determine the cause of an error. Interfaces can attempt to minimize errors by anticipating certain actions that could lead to an error; and users can be warned if they are requesting ambiguous or incorrect actions, such as attempting to apply a procedure to multiple application objects.	5M	CO3	L2
2 (a)	Explain OpenGL Keyboard function and Space Ball function with proper code snippets. **Solution:** **GLUT Keyboard Functions** With keyboard input, we use the following function to specify a procedure that is to be invoked when a key is pressed: glutKeyboardFunc (keyFcn); The specified procedure has three arguments: void keyFcn (GLubyte key, GLint xMouse, GLint yMouse) Parameter key is assigned a character value or the corresponding ASCII code. The display-window mouse location is returned as position (xMouse, yMouse) relative to the top-left corner of the display window. When a designated key is pressed, we can use the mouse location to initiate some action, independently of whether any mouse buttons are pressed. For function keys, arrow keys, and other special-purpose keys, we can use the command glutSpecialFunc (specialKeyFcn); The specified procedure has the same three arguments: void specialKeyFcn (GLint specialKey, GLint xMouse, GLint yMouse) **GLUT Spaceball Functions** We use the following function to specify an operation when a spaceball button is activated for a selected display window: glutSpaceballButtonFunc (spaceballFcn); The callback function has two parameters: void spaceballFcn (GLint spaceballButton, GLint action) Spaceball buttons are identified with the same integer values as a tablet, and parameter action is assigned either the value GLUT UP or the value GLUT DOWN.We can determine the number of available spaceball buttons with a call to glutDeviceGet using the argument GLUT NUM SPACEBALL BUTTONS.	4M	CO3	L3

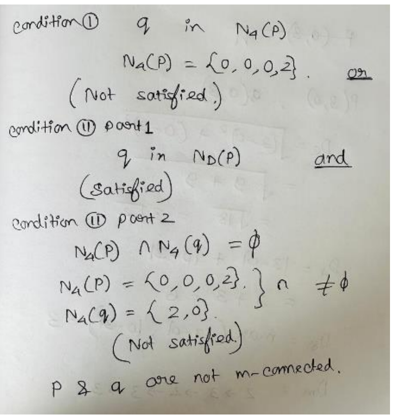
	Translational motion of a spaceball, when the mouse is in the display window, is recorded with the function call glutSpaceballMotionFunc (spaceballTranlFcn); The three-dimensional translation distances are passed to the invoked function as, for example: void spaceballTranslFcn (GLint tx, GLint ty, GLint tz) These translation distances are normalized within the range from −1000 to 1000. Similarly, a spaceball rotation is recorded with glutSpaceballRotateFunc (spaceballRotFcn); The three-dimensional rotation angles are then available to the callback function, as follows: void spaceballRotFcn (GLint thetaX, GLint thetaY, GLint thetaZ)			
(b)	Describe the openGL menu and submenu functions with a suitable example. **Solution:** **Creating a GLUT Menu** A pop-up menu is created with the statement glutCreateMenu (menuFcn); where parameter menuFcn is the name of a procedure that is to be invoked when a menu entry is selected. This procedure has one argument, which is the integer value corresponding to the position of a selected option. void menuFcn (GLint menuItemNumber) The integer value passed to parameter menuItemNumber is then used by menuFcn to perform an operation. When a menu is created, it is associated with the current display window. Once we have designated the menu function that is to be invoked when a menu item is selected, we must specify the options that are to be listed in the menu. We do this with a series of statements that list the name and position for each option. These statements have the general form glutAddMenuEntry (charString, menuItemNumber); Parameter charString specifies text that is to be displayed in the menu, and parameter menuItemNumber gives the location for that entry in the menu. For example, the following statements create a menu with two options: glutCreateMenu (menuFcn); glutAddMenuEntry ("First Menu Item", 1); glutAddMenuEntry ("Second Menu Item", 2); Next, we must specify a mouse button that is to be used to select a menu option. This is accomplished with glutAttachMenu (button); where parameter button is assigned one of the three GLUT symbolic constants referencing the left, middle, or right mouse button. **Creating GLUT Submenus** A submenu can be associated with a menu by first creating the submenu using glutCreateMenu, along with a list of suboptions, and then listing the submenu as an additional option in the main menu.We can add the submenu to the option list in a main menu (or other submenu) using a sequence of statements such as submenuID = glutCreateMenu (submenuFcn); glutAddMenuEntry ("First Submenu Item", 1); . . . glutCreateMenu (menuFcn); glutAddMenuEntry ("First Menu Item", 1); . . . glutAddSubMenu ("Submenu Option", submenuID); The glutAddSubMenu function can also be used to add the submenu to the current menu.	6M	CO3	L2

3 (a) Define m-connectivity. Consider the image segment shown. 5M CO4 L3

Check whether p and q are m-connected or not. $\begin{bmatrix} 0 & {}^{p}2 & \overset{q}{\textcircled{2}} \\ 0 & \textcircled{2} & 0 \\ 0 & 0 & 0 \end{bmatrix}$

Solution:

Mixed connectivity Mixed connectivity is also known as m-connectivity. Two pixels p and q are said to be in m-connectivity when q is in N4(p) or 2. q is in ND(p) and the intersection of N4(p) and N4(q) is empty.



condition ① q in $N_4(P)$

$N_4(P) = \{0,0,0,2\}$. or

(Not satisfied)

condition ⑪ point 1

q in $N_D(P)$ and

(Satisfied)

condition ⑪ point 2

$N_4(P) \cap N_4(q) = \phi$

$\left.\begin{array}{l} N_4(P) = \{0,0,0,2\}. \\ N_4(q) = \{2,0\}. \end{array}\right\} \cap \neq \phi$

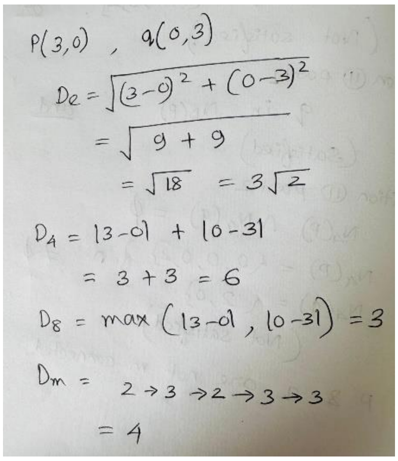
(Not satisfied)

p & q are not m-connected.

(b) Consider the image segment shown. 5M CO4 L3

$\begin{array}{cccc} 2 & 1 & 2 & \textcircled{3}\,q \\ 2 & 0 & 1 & 3 \\ 1 & 3 & 2 & 2 \\ \textcircled{2}\,p & 1 & 0 & 1 \end{array}$ Let V = {2, 3}. Compute De, D_4, D_8 and Dm distances between p and q.

Solution:



$P(3,0)$, $q(0,3)$

$De = \sqrt{(3-0)^2 + (0-3)^2}$

$= \sqrt{9 + 9}$

$= \sqrt{18} = 3\sqrt{2}$

$D_4 = |3-0| + |0-3|$

$= 3 + 3 = 6$

$D_8 = \max(|3-0|, |0-3|) = 3$

$D_m = 2 \rightarrow 3 \rightarrow 2 \rightarrow 3 \rightarrow 3$

$= 4$

4 Explain the fundamental steps of digital image processing with a neat diagram. 10M CO4 L1

Solution:

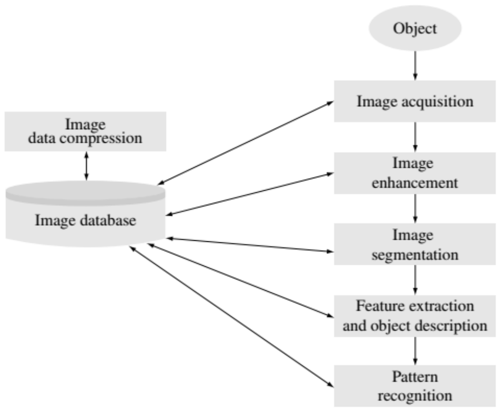
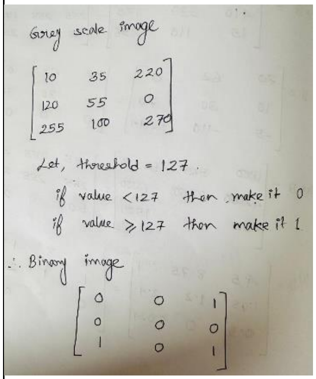


	Image acquisition This step aims to obtain the digital image of the object. **Image enhancement** This step aims to improve the quality of the image so that the analysis of the images is reliable. **Image segmentation** This step divides the image into many sub-regions and extracts the regions that are necessary for further analysis. The portions of the image that are not necessary, such as image backgrounds (dictated by the imaging requirement), are discarded. **Feature extraction and object description** Imaging applications use many routines for extraction of image features that are necessary for recognition. This is called image feature extraction step. The extracted object features are represented in meaningful data structures and the objects are described. **Pattern recognition** This step is for identifying and recognizing the object that is present in the image, using the features generated in the earlier step and pattern recognition algorithms such as classifications or clustering.			
5 (a)	List some practical applications of NAND, NOR, and XOR operations. **Solution:** NAND: 1. Computation of the intersection of images. 2. Design of filter masks. 3. Slicing of grey scale images; NOR: 1. Used to perform the union operation 2. Used as a merging operator XOR: 1. Change detection 2. Use as a subcomponent of a complex imaging operation.	6 M	CO4	L1
(b)	With the help of an example, explain how a gray-level image is converted to a binary image. **Solution:** The binary image is created from a grey scale image using a threshold process. The pixel value is compared with the threshold value. If the pixel value of the grey scale image is greater than the threshold value, the pixel value in the binary image is considered as 1. Otherwise, the pixel value is 0.  Grey scale image $\begin{bmatrix} 10 & 35 & 220 \\ 120 & 55 & 0 \\ 255 & 100 & 279 \end{bmatrix}$ Let, threshold = 127. if value <127 then make it 0 if value $\geq$127 then make it 1 ∴ Binary image $\begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ 1 & 0 & 1 \end{bmatrix}$	4 M	CO4	L2
6	Compare linear and non-linear operations. Consider the following 8-bit integer-type images. $A=\begin{bmatrix} 90 & 70 & 230 \\ 210 & 180 & 120 \\ 5 & 0 & 6 \end{bmatrix}$ $B=\begin{bmatrix} 20 & 8 & 230 \\ 200 & 150 & 50 \\ 10 & 110 & 250 \end{bmatrix}$ Perform addition, subtraction, multiplication, and division operations. **Solution:** An operator is called a linear operator if it obeys the following rules of additivity and homogeneity.	10 M	CO4	L4

1. Property of additivity

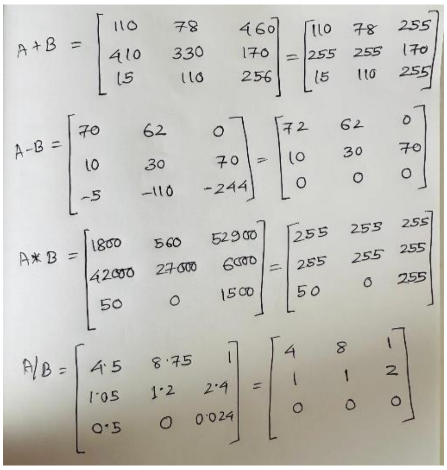
$$\begin{aligned} H(a_1 f_1(x, y) + a_2 f_2(x, y)) &= H(a_1 f_1(x, y)) + H(a_2 f_2(x, y)) \\ &= a_1 H(f_1(x, y)) + a_2 H(f_2(x, y)) \\ &= a_1 \times g_1(x, y) + a_2 \times g_2(x, y) \end{aligned}$$

2. Property of homogeneity

$$H(kf_1(x,y)) = kH(f_1(x,y)) = kg_1(x,y)$$

A non-linear operator, as the name suggests, does not follow these rules.

Since the images are 8-bit image, range of the color levels is 0-255.



$$A+B = \begin{bmatrix} 110 & 78 & 460 \\ 410 & 330 & 170 \\ 15 & 110 & 256 \end{bmatrix} = \begin{bmatrix} 110 & 78 & 255 \\ 255 & 255 & 170 \\ 15 & 110 & 255 \end{bmatrix}$$

$$A-B = \begin{bmatrix} 70 & 62 & 0 \\ 10 & 30 & 70 \\ -5 & -110 & -244 \end{bmatrix} = \begin{bmatrix} 72 & 62 & 0 \\ 10 & 30 & 70 \\ 0 & 0 & 0 \end{bmatrix}$$

$$A*B = \begin{bmatrix} 1800 & 560 & 52900 \\ 42000 & 27000 & 6000 \\ 50 & 0 & 1500 \end{bmatrix} = \begin{bmatrix} 255 & 255 & 255 \\ 255 & 255 & 255 \\ 50 & 0 & 255 \end{bmatrix}$$

$$A/B = \begin{bmatrix} 4.5 & 8.75 & 1 \\ 1.05 & 1.2 & 2.4 \\ 0.5 & 0 & 0.024 \end{bmatrix} = \begin{bmatrix} 4 & 8 & 1 \\ 1 & 1 & 2 \\ 0 & 0 & 0 \end{bmatrix}$$

CI **CCI** **HOD**