TABLE OF CONTENTS

Unit I

Unit II

Unit III

Unit IV

Unit V

Unit VI

1 Graphics Primitives

1.1 : Introduction

Q.1 What is pixel ? ☞[SPPU : Dec.-09,11,13, May-13, Marks 2]

Ans. : • In computer graphics, pictures or graphics objects are presented as a collection of discrete picture elements called **pixels.**

• The pixel is the smallest addressable screen element.

Q.2 What is computer graphics ? ☞[SPPU : Dec.-14,19, Marks 2]

Ans. : • The computer graphics is one of the most effective and commonly used way to communicate the processed information to the user.

• It displays the information in the form of **graphics objects** such as pictures, charts, graphs and diagrams instead of simple text. Thus we can say that computer graphics makes it possible to **express data in pictorial form.**

• The picture or graphics object may be an engineering drawing, business graphs, architectural structures, a single frame from an animated movie or a machine parts illustrated for a service manual.

Q.3 Why computer graphics is emerging as an important field in computer science. ☞[SPPU : May-05, Dec.-07,08, Marks 4]

Ans. : • Computer graphics permits extensive, high-bandwidth user-computer interaction. It significantly enhances the ability to understand information, to perceive trends and to visualize real or imaginary objects either moving or stationary in a realistic environment. It also makes it possible to get high quality and more precise results and products with lower analysis and design cost.

Because of these reasons, computer graphics is emergining as an important field in computer science.

Q.4 State the applications of computer graphics.

☞[SPPU : Dec.-14,19, Marks 6]

Ans. : Applications of computer graphics are :

• **User Interfaces :** User friendliness is one of the main factors underlying the success and popularity of any system. The built-in graphics provided with user interfaces use visual control items such as buttons, menus, icons, scroll bar etc, which allows user to interact with computer only by mouse-click. Typing is necessary only to input text to be stored and manipulated.

• **Plotting of Graphics and Chart :** In industry, business, government and educational organizations, computer graphics is most commonly used to create 2D and 3D graphs of mathematical, physical and economic functions in form of histograms, bars and pie-charts. These graphs and charts are very useful for decision making.

• **Office Automation and Desktop Publishing :** The desktop publishing on personal computers allow the use of graphics for the creation and dissemination of information.

• **Computer-aided Drafting and Design :** The computer-aided drafting uses graphics to design components and systems electrical, mechanical, electro-mechanical and electronic devices such as automobile bodies, structures of building, airplane, ships, Very Large-Scale Integrated (VLSI) chips, optical systems and computer networks.

• **Simulation and Animation :** Use of graphics in simulation makes mathematic models and mechanical systems more realistic and easy to study. The interactive graphics supported by animation software proved their use in production of animated movies and cartoons films.

• **Art and Commerce :** There is a lot of development in the tools provided by computer graphics. This allows user to create artistic pictures which express messages and attract attentions. Such pictures are very useful in advertising.

• **Process Control :** Process systems and processing parameters are shown on the computer with graphic symbols and identifications. This makes it easy for operator to monitor and control various processing parameters at a time.

• **Cartography :** Computer graphics is also used to represent geographic maps, weather maps, oceanographic charts, contour maps, population density maps and so on.

- **Education and Training :** Computer graphics can be used to generate models of physical, financial and economic systems. These models can be used as educational aids. Models of physical systems, physiological systems, population trends, or equipment, such as colour-coded diagram can help trainees to understand the operation of the system.
- **Image Processing :** In computer graphics, a computer is used to create pictures. Image processing, on the other hands, applies techniques to modify or interpret existing pictures such as photographs and scanned images.

Q.5 With the help of block diagram explain raster scan displays.

☞[SPPU : Dec.-06,10,13,15, May-09,10,14,15, Marks 8]

Ans. : • The Fig. Q.5.1 shows the architecture of a raster display.

- It consists of display controller, Central Processing Unit (CPU), video controller, refresh buffer, keyboard, mouse and the CRT.
- As shown in the Fig. Q.5.1, the display image is stored in the form of 1s and 0s in the refresh buffer.

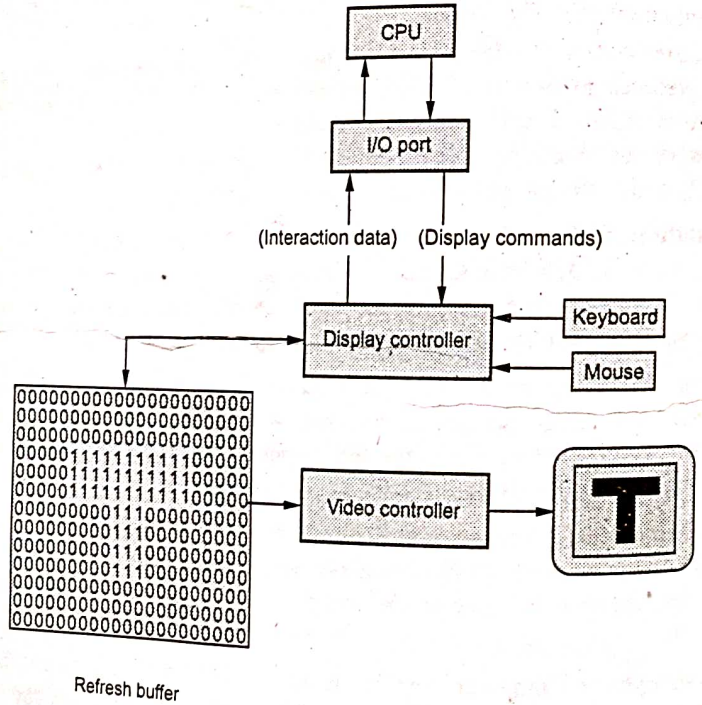


Fig. Q.5.1 Architecture of a raster display

- The video controller reads this refresh buffer and produces the actual image on the screen. It does this by scanning one scan line at a time, from top to bottom and then back to the top, as shown in the Fig. Q.5.1.
- Raster scan is the most common method of displaying images on the CRT screen.
- In this method, the horizontal and vertical deflection signals are generated to move the beam all over the screen in a pattern shown in the Fig. Q.5.2.
- Here, the beam is swept back and forth from the left to the right across the screen.
- When the beam is moved from the left to the right, it is ON. The beam is OFF, when it is moved from the right to the left as shown by dotted line in Fig. Q.5.2.

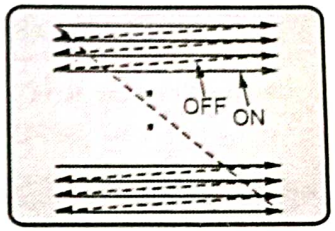


Fig. Q.5.2 Raster scan CRT

- When the beam reaches the bottom of the screen, it is made OFF and rapidly retraced back to the top left to start again. A display produced in this way is called **raster scan display.**

Q.6 Define random scan display. **☞ [SPPU : May-14, Marks 2]**

Ans. : In displays where the beam is moved between the end points of the graphics primitives is called random (vector) scan displays.

1.2 : Persistence, Resolution and Aspect Ratio

Q.7 Define following terms :

i) Persistence **☞[SPPU : May-14, Mark 1]**

ii) Resolution iii) Aspect ratio

☞ [SPPU : Dec.-06,10,13,14,15, May-09,15,19, Marks 4]

OR Write the properties of video display devices.

☞ [SPPU : May-15, Marks 4]

Ans. : Persistence : Persistence is defined as the time it takes the emitted light from the screen to decay to one-tenth of its original intensity.

Resolution : Resolution indicates the maximum number of points that can be displayed without overlap on the CRT. It is defined as the number of points per centimeter that can be plotted horizontally and vertically.

Aspect Ratio : It is the ratio of vertical points to horizontal points to produce equal length lines in both directions on the screen.

1.3 : Frame Buffers

Q.8 Define the following term : Frame buffer.

☞[SPPU : Dec.-11,12,15, May-13, Marks 2]

OR Describe frame buffer display in computer graphics.

☞[SPPU : Dec.-13, 16, May-15, Marks 4]

Ans. : In raster scan displays a special area of memory is dedicated to graphics only. This memory area is called **frame buffer**. It holds the set of intensity values for all the screen points.

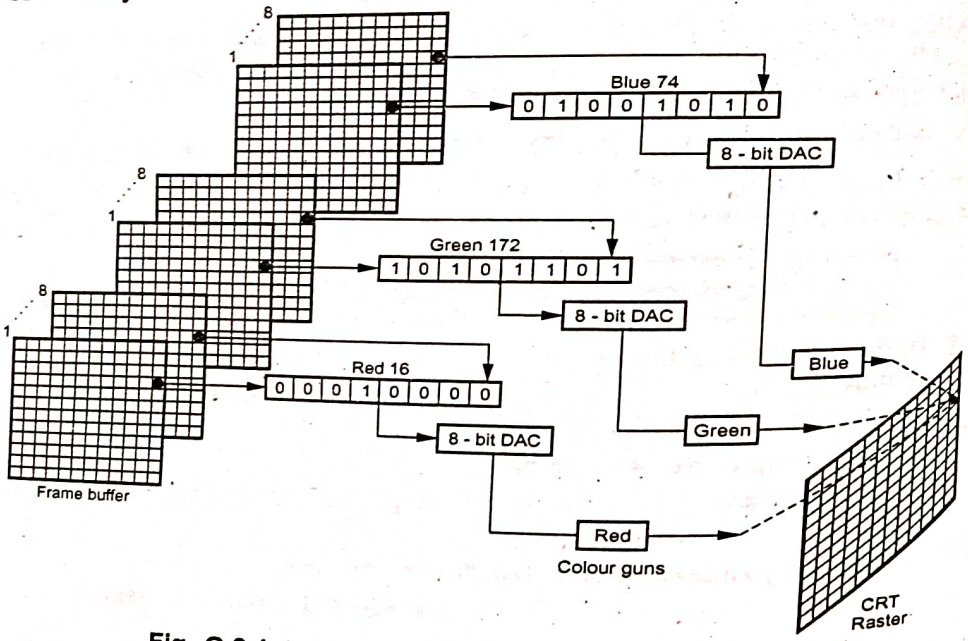


Fig. Q.8.1 A 24-bit plane colour frame buffer

- The video controller retrieves the stored intensity values from frame buffer and displays them on the screen one row (scan line) at a time, typically 50 times per second.
- The Fig. Q.8.1 shows 24-bit plane colour frame buffer. In 24-bit plane, 8-bits are used to represent each colour, and hence there are 8-bit planes per colour. Each group of bit planes drives an 8-bit DAC. Each group generates 256 (2^8) shades or intensities of red, green or blue. These are combined into 16, 777, 216 $\left[(2^8)^3 = 2^{24}\right]$ possible colours. This is a **full colour frame buffer.**

Q.9 Find the refresh rate of a 512 × 512 frame buffer, if the access time for each pixel is 200 nanoseconds.

☞[SPPU : Dec.-10, Marks 4]

Ans. :

Refresh rate

$$= \frac{1}{\text{Access time for each pixel} \times \text{Number of pixels information in frame buffer}}$$

$$= \frac{1}{200 \times 10^{-9} \times 512 \times 512} = 19.0734 \text{ frames /sec.}$$

Q.10 Find the amount of memory required by an 8 plane frame buffer each of red, green, blue having resolution of 1024 × 768.

☞[SPPU : Dec.-12, Marks 4]

Ans. : For 8 plane frame buffer each of red, green and blue requires 24-bits to represent per pixel.

∴ Amount of memory required = 24 × 1024 × 768

= 18874368 bits

= 2359296 bytes

1.4 : Display Devices

Q.11 With a neat cross sectional view explain the functioning of CRT.

Ans. : • A CRT is an evacuated glass tube. An **electron gun** at the rear of the tube produces a beam of electrons which is directed towards the front of the tube (screen) by a high voltage typically 15000 to

20,000 volts. The negatively charged electrons are then accelerated toward the phosphor coating at the inner side of the screen by a high positive voltage or by using **accelerating anode**. The phosphor substance gives off light when it is stroked by electrons.

- One way to keep the phosphor glowing is to redraw the picture repeatedly by quickly directing the electron beam back over the same points. This type of display is called a **refresh CRT**.
- The **control grid** voltage determines the velocity achieved by the electrons before they hit the phosphor. The control grid voltage determines how many electrons are actually in the electron beam.
- A more negative voltage applied to the control grid will shut off the beam. A smaller negative voltage on the control grid simply decreases the number of electrons passing through. Since the amount of light emitted by the phosphor coating depends on the number of electrons striking the screen, we control the brightness of a display by varying the voltage on the control grid.
- The **focusing system** concentrates the electron beam so that the beam converges to a small point when it hits the phosphor coating. It is possible to control the point at which the electron beam strikes the screen, and therefore the position of the dot upon the screen, by deflecting the electron beam. Fig. Q.11.1 shows the electrostatic deflection of the electron beam in a CRT.

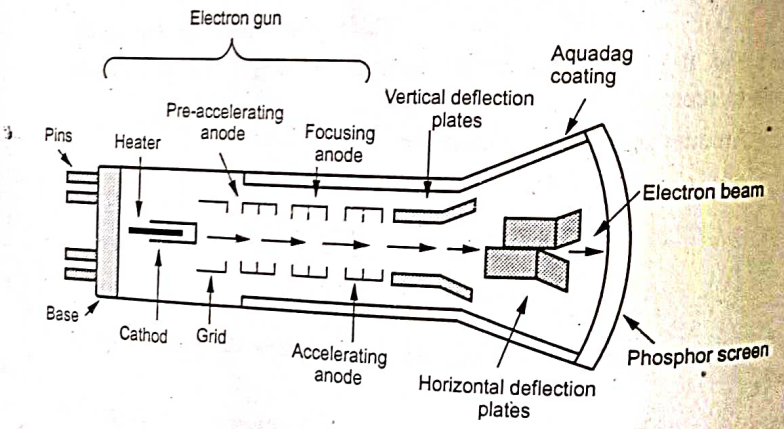


Fig. Q.11.1 Cathode ray tube

- The **deflection system** of the cathode-ray-tube consists of two pairs of parallel plates, referred to as the **vertical** and **horizontal deflection** plates. The voltage applied to vertical plates controls the vertical deflection of the electron beam and voltage applied to the horizontal deflection plates controls the horizontal deflection of the electron beam.
- The electrons that strike the screen, release secondary emission electrons. Aqueous graphite solution known as **Aquadag** collects these secondary electrons. To maintain the CRT in electrical equilibrium, it is necessary to collect the secondary electrons.

Q.12 List the methods used for colour display.

Ans. : There are two basic techniques used for producing colour displays :

- Beam-penetration technique and
- Shadow-mask technique

Q.13 Explain the beam - penetration technique.

Ans. : This technique is used with random-scan monitors. In this technique, the inside of CRT screen is coated with two layers of phosphor, usually red and green. The displayed colour depends on how far the electron beam penetrates into the phosphor layers. The outer layer is of red phosphor and inner layer is of green phosphor. A beam of slow electrons excites only the outer red layer. A beam of very fast electrons penetrates through the red layer and excites the inner green layer. At intermediate beam speeds, combinations of red and green light are emitted and two additional colours, orange and yellow displayed. The beam acceleration voltage controls the speed of the electrons and hence the screen colour at any point on the screen.

Merits and Demerits

- It is an inexpensive technique to produce colour in random scan monitors.
- It can display only four colours.
- The quality of picture produced by this technique is not as good as compared to other techniques.

Q.14 Explain shadow mask method.

OR Explain shadow mask technique and explain how does it differ from beam penetration technique ?

Ans. : The shadow mask technique produces a much wider range of colours than the beam penetration technique. Hence this technique is commonly used in raster-scan displays including colour TV. In a shadow mask technique, CRT has three phosphor colour dots at each pixel position. One phosphor dot emits a red light, another emits a green light, and the third emits a blue light. The Fig. Q.14.1 shows the shadow mask CRT. It has three electron guns, one for each colour dot, and a shadow mask grid just behind the phosphor coated screen.

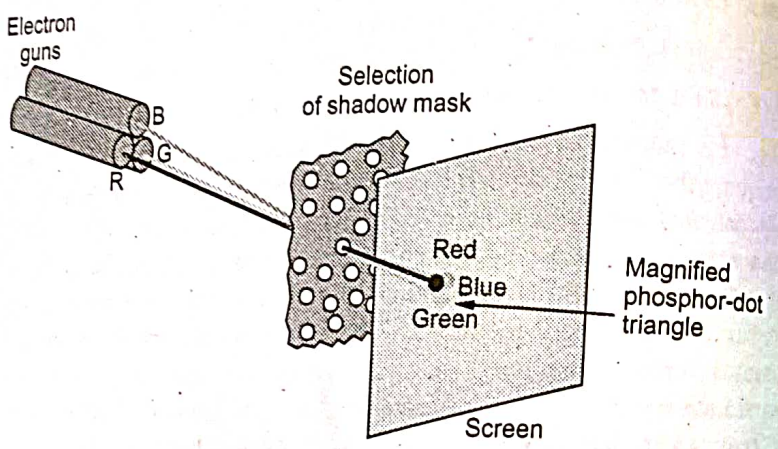


Fig. Q.14.1

- The shadow mask grid consists of series of holes aligned with the phosphor dot pattern. As shown in the Fig. Q.14.1, three electron beams are deflected and focused as a group onto the shadow mask and when they pass through a hole in the shadow mask, they excite a dot triangle. A dot triangle consists of three small phosphor dots of red, green and blue colour. These phosphor dots are arranged so that each electron beam can activate only its corresponding colour dot when it passes through the shadow mask. A dot triangle when activated appears as a small dot on the screen which has colour of combination of three small dots in the dot triangle. By varying the intensity of the three electron beams we can obtain different colours in the shadow mask CRT.

Q.15 Explain DVST.

Ans. : A Direct-View Storage Tube (DVST) uses the storage grid which stores the picture information as a charge distribution just behind the phosphor-coated screen.

- The Fig. Q.15.1 shows the general arrangement of the DVST. It consists of two electron guns : a primary gun and a flood gun. A primary gun stores the picture pattern and the flood gun maintains the picture display. A primary gun produces high speed electrons which strike on the storage grid to draw the picture pattern. As electron beam strikes on the storage grid with high speed, it knocks out electrons from the storage grid keeping the net positive charge. The knocked out electrons are attracted towards the collector. The net positive charge on the storage grid is nothing but the picture pattern. The continuous low speed electrons from flood gun pass through the control grid and are attracted to the positive charged areas of the storage grid. The low speed electrons then penetrate the storage grid and strike the phosphor coating without affecting the positive charge pattern on the storage grid. During this process the collector just behind the storage grid smooths out the flow of flood electrons.

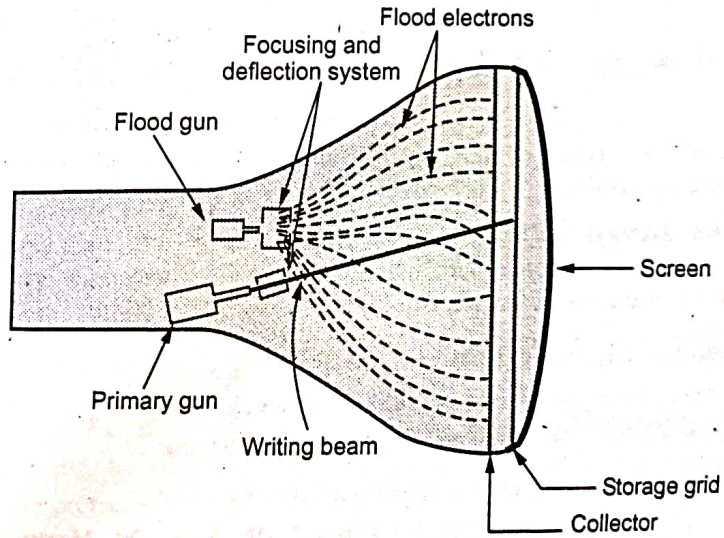


Fig. Q.15.1 Arrangement of the DVST

Q.14 Explain shadow mask method.

OR Explain shadow mask technique and explain how does it differ from beam penetration technique ?

Ans. : The shadow mask technique produces a much wider range of colours than the beam penetration technique. Hence this technique is commonly used in raster-scan displays including colour TV. In a shadow mask technique, CRT has three phosphor colour dots at each pixel position. One phosphor dot emits a red light, another emits a green light, and the third emits a blue light. The Fig. Q.14.1 shows the shadow mask CRT. It has three electron guns, one for each colour dot, and a shadow mask grid just behind the phosphor coated screen.

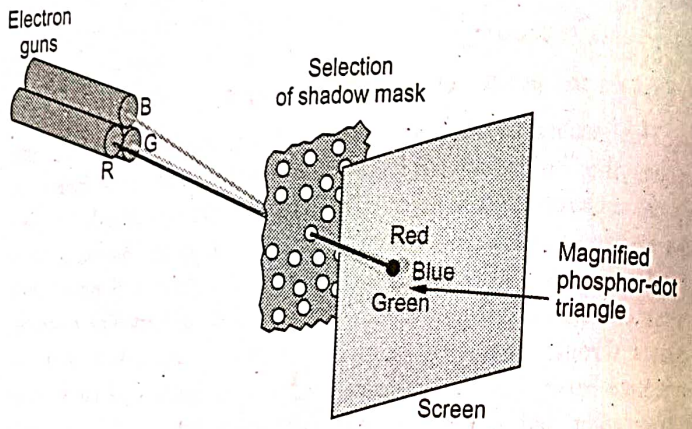


Fig. Q.14.1

- The shadow mask grid consists of series of holes aligned with the phosphor dot pattern. As shown in the Fig. Q.14.1, three electron beams are deflected and focused as a group onto the shadow mask and when they pass through a hole in the shadow mask, they excite a dot triangle. A dot triangle consists of three small phosphor dots of red, green and blue colour. These phosphor dots are arranged so that each electron beam can activate only its corresponding colour dot when it passes through the shadow mask. A dot triangle when activated appears as a small dot on the screen which has colour of combination of three small dots in the dot triangle. By varying the intensity of the three electron beams we can obtain different colours in the shadow mask CRT.

Q.15 Explain DVST.

Ans. : A Direct-View Storage Tube (DVST) uses the storage grid which stores the picture information as a charge distribution just behind the phosphor-coated screen.

- The Fig. Q.15.1 shows the general arrangement of the DVST. It consists of two electron guns : a primary gun and a flood gun. A primary gun stores the picture pattern and the flood gun maintains the picture display. A primary gun produces high speed electrons which strike on the storage grid to draw the picture pattern. As electron beam strikes on the storage grid with high speed, it knocks out electrons from the storage grid keeping the net positive charge. The knocked out electrons are attracted towards the collector. The net positive charge on the storage grid is nothing but the picture pattern. The continuous low speed electrons from flood gun pass through the control grid and are attracted to the positive charged areas of the storage grid. The low speed electrons then penetrate the storage grid and strike the phosphor coating without affecting the positive charge pattern on the storage grid. During this process the collector just behind the storage grid smooths out the flow of flood electrons.

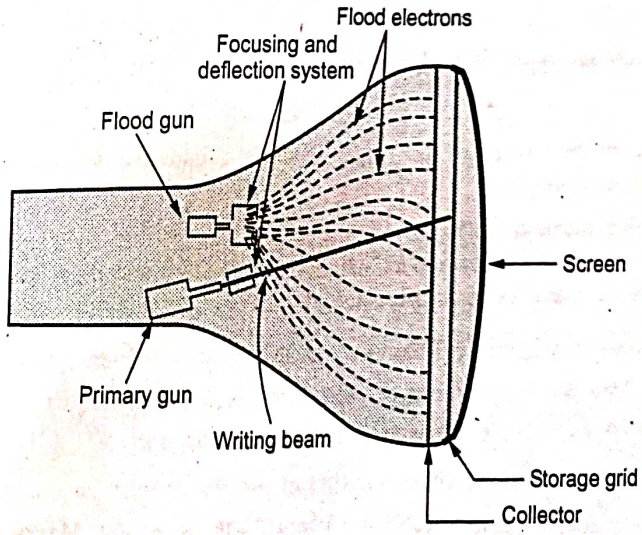


Fig. Q.15.1 Arrangement of the DVST

Q.16 List merit and demerit of DVST.

Ans. : Advantages of DVST

1. Refreshing of CRT is not required.
2. Because no refreshing is required, very complex pictures can be displayed at very high resolution without flicker.
3. It has flat screen.

Disadvantages of DVST

1. They do not display colours and are available with single level of line intensity.
2. Erasing requires removal of charge on the storage grid. Thus erasing and redrawing process takes several seconds.
3. Selective or part erasing of screen is not possible.
4. Erasing of screen produces unpleasant flash over the entire screen surface which prevents its use of dynamic graphics applications.
5. It has poor contrast as a result of the comparatively low accelerating potential applied to the flood electrons.
6. The performance of DVST is some what inferior to the refresh CRT.

Q.17 State the types of flat panel displays.

Ans. :

- There are two types of flat panel displays : **emissive displays** and **nonemissive displays.**
- **Emissive displays** : They convert electrical energy into light energy. Plasma panels, thin-film electro luminescent displays, and light emitting diodes are examples of emissive displays.
- **Nonemissive displays** : They use optical effects to convert sunlight or light from some other source into graphics patterns. Liquid crystal display is an example of nonemissive flat panel display.

Q.18 List the operating characteristics of plasma panel.

☞ [JNTU : May-07, 08, Sept.-08, Marks 4]

Ans.: • Fig. Q.18.1 shows the construction of plasma panel display. It consists of two plates of glass with thin, closely spaced gold electrodes. The gold electrodes are attached to the inner faces and covered with a

dielectric material. These are attached as a series of vertical conducting ribbons on one glass plate, and a set of horizontal ribbons to the other glass plate. The space between two glass plates is filled with neon-based gas and sealed. By applying voltages between the electrodes the gas within the panel is made to behave as if it were divided into tiny cells, each one independent of its neighbours. These independent cells are made to glow by placing a firing voltage of about 120 volts across it by means of the electrodes. The glow can be sustained by maintaining a high frequency alternating voltage of about 90 volts across the cell. Due to this refreshing is not required.

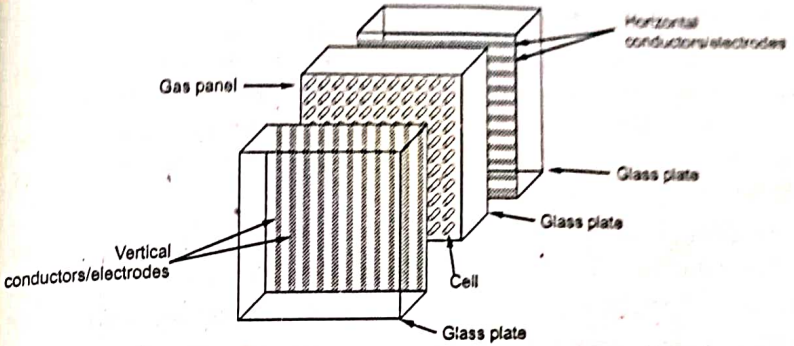


Fig. Q.18.1 Construction of plasma panel display

Q.19 State the advantages and disadvantages of plasma display.

Ans. : Advantages

1. Refreshing is not required.
2. Produces a very steady image, totally free of flicker.
3. Less bulky than a CRT.
4. Allows selective writing and selective erasing, at speed of about 20 μsec per cell.
5. It has the flat screen and is transparent, so the displayed image can be superimposed with pictures from slides or other media projected through the rear panel.

Disadvantages

1. Strictly monochromatic devices.
2. Relatively poor resolution of about 60 dots per inch.

2. Requires complex addressing and wiring.
3. Costlier than the CRTs.

Q.20 State the difference between thin-film electroluminescent displays and plasma displays.

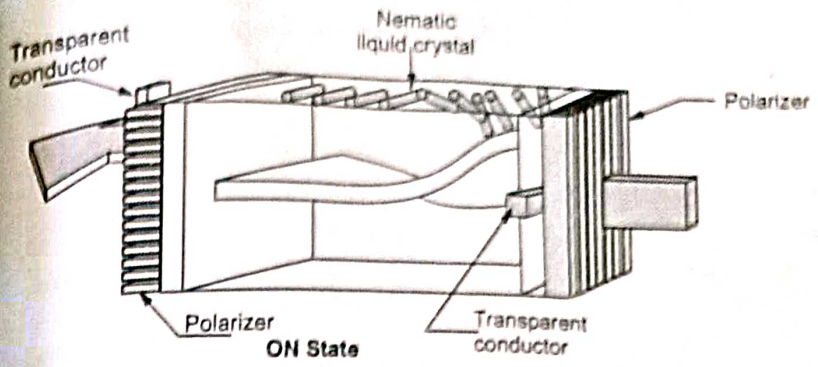
Ans. : • Thin-film electroluminescent displays are similar in construction to a plasma panel. However, the region between the glass plates is filled with a phosphor, such as zinc sulfide doped with manganese, instead of neon gas.

Q.21 Write a note on LCDs.

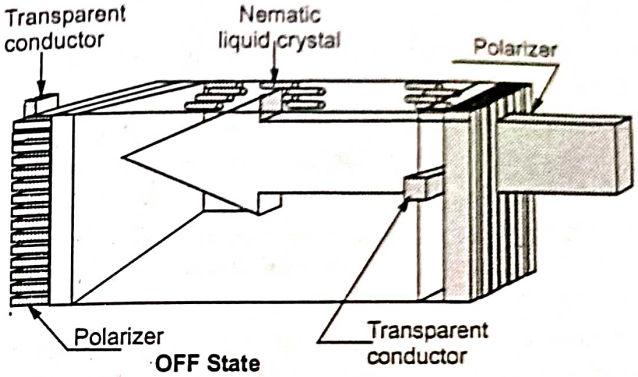
Ans. • The term liquid crystal refers to the fact that these compounds have a crystalline arrangement of molecules, yet they flow like a liquid. Flat panel displays commonly use nematic (thread like) liquid-crystal compounds that tend to keep the long axes of the rod- shaped molecules aligned.

• Two glass plates, each containing a light polarizer at right angles to the other plate sandwich the liquid-crystal material. Rows of horizontal transparent conductors are built into one glass plate, and columns of vertical conductors are put into the other plate. The intersection of two conductors defines a pixel position. In the ON state, polarized light passing through material is twisted so that it will pass through the opposite polarizer. It is then reflected back to the viewer. To turn OFF the pixel, we apply a voltage to the two intersecting conductors to align the molecules so that the light is not twisted as shown in the Fig. Q.21.1. This type of flat panel device is referred to as a passive matrix LCD. In **active matrix LCD**, transistors are used at each (x, y) grid point. Use of transistors cause the crystal to change their state quickly and also to control the degree to which the state has been changed. These two properties allow LCDs to be used in miniature television sets with continuous-tone images. (See Fig. Q.21.1 on next page).

• The transistor can also serve as a memory for the state of a cell and can hold the cell in that state until it is changed. Thus use of transistor make cell on for all the time giving brighter display than it would be if it had to be refreshed periodically.



(a) Field effect display 'ON State'



(b) Field effect display 'OFF State'

Fig. Q.21.1

Advantages of LCD Displays

- Low cost
- Low weight
- Small size
- Low power consumption

Q.22 What are LED displays ?

Ans. : • The Light-Emitting Diode (LED) displays are emissive type display. In these displays, a matrix of diodes is arranged to form the pixel positions in the display, and picture definition is stored in a refresh buffer.

As in scan-line refreshing of a CRT, information is read from the refresh buffer and converted to voltage levels that are applied to the diodes to produce the light patterns in the display.

Q.23 Explain the applications for large-screen displays. What graphical output devices support it ?

Ans. : Applications of large screen displays are :

1. **Education :** Many institutes use large screen displays to give effective audio-visual training in the class-room.
2. **Entertainment :** Large screen displays are used to watch live matches in the hotels or in the community halls.
3. **Sports ground :** Large screen displays are used to show current scores in the stadiums.
4. **Advertisement :** Large screen displays are used for advertising purpose in many public places.
5. **Railway, airway and roadway transport :** Large screen displays are used at the railway stations, airports and bus stands to provide the schedule information to the travelers.
6. **Seminars, presentations and video-conferencing :** Large screen displays are used during presentations, seminars and video-conferencing.
7. **Space stations :** Large screen displays are used in space stations to display the satellite images of the planets. They are used as one of the visual communication medium for communicating with the astronauts.

Plasma-panel and flat panel displays support large screen displays.

END...

Unit I

2 Introduction to OpenGL

2.1 : OpenGL Architecture

Q.1 What is openGL ? ☞ **[SPPU : Dec.-17, Marks 3]**

Ans. : • OpenGL (Open Graphics Library) is a cross-language, multi-platform Application Programming Interface (API) for rendering 2D and 3D computer graphics.

- The API is typically used to interact with a Graphics Processing Unit (GPU), to achieve hardware-accelerated rendering.
- A basic library of functions is provided in OpenGL for specifying graphics primitives, attributes, geometric transformations, viewing transformations and many other operations.
- OpenGL is designed to be hardware independent, therefore many operations, such as input and output routines, are not included in the basic library. However, input and output routines and many additional functions are available in auxiliary libraries that have been developed for OpenGL programs.

Q.2 State the naming conventions of openGL.

Ans. : • Function names in the **OpenGL basic library** (also called the **OpenGL core library**) are prefixed with **gl** and each component word within a function name has its first letter capitalized. The following examples illustrate this naming convention.

glBegin, glClear, glCopyPixels, glPolygonMode

- Some functions in OpenGl require that one (or more) of their arguments be assigned a symbolic constant specifying, for instance, a parameter name, a value for a parameter or a particular mode. All such constants begin with the uppercase letters GL. In addition, component words within a constant name are written in capital letters and the underscore ( ) is used as a separator between all component words in the name.

Following are a few examples of the several hundred symbolic constants available for use with OpenGL functions.

- GL_2D, GL_RGB, GL_CCW, GL_POLYGON, GL_AMBIENT_AND_DIFFUSE

Q.3 Explain the display window management using GLUT.

Ans. : • We perform the GLUT initialization with the statement

```
glutInit (&argc, argv);
```

- Next, we can state that a display window is to be created on the screen with a given caption for the title bar. This is accomplished with the function

```
glutCreateWindow ("An Example OpenGL Program");
```

- Then we need to specify what the display window is to contain. For this, we create a picture using OpenGL functions and pass the picture definition to the GLUT routine **glutDisplayFunc**, which assigns our picture to the display window.
- As an example, suppose we have the OpenGL code for describing a line segment in a procedure called lineSegment. Then the following function call passes the line-segment description to the display window.

```
glutDisplayFunc (lineSegment);
```

- But the display window is not yet on the screen. We need one more GLUT function to complete the window-processing operations. After execution of the following statement, all display windows that we have created, including their graphic content, are now activated.

```
glutMainLoop ( );
```

- This function must be the last one in the program. It displays the initial graphics and puts the program into an infinite loop that checks for input from devices such as a mouse or keyboard.
- We use the **glutInitWindowPosition** function to give an initial location for the top left corner of the display window.

```
glutInitWindowPosition (50, 80);
```

- Similarly, the glutInitWindowSize function is used to set the initial pixel width and height of the display window.

```
glutInitWindowSize (300, 200);
```

- After the display window is on the screen, we can reposition and resize it.

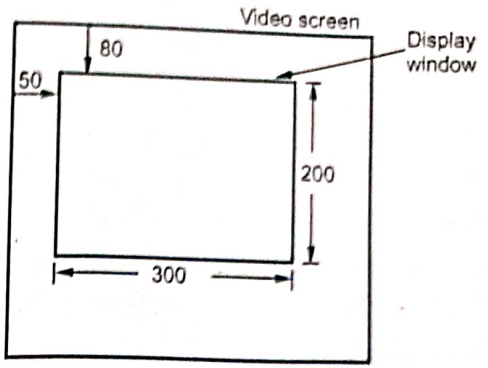


Fig. Q.3.1 A 300 by 200 display window at position (50, 80) relative to the top-left corner of the video display

- We can also set a number of other options for the display window, such as buffering and a choice of color modes, with the **glutInitDisplayMode** function. Arguments for this routine are assigned symbolic GLUT constants.
- For example,

```
glutInitDisplayMode (GLUT_SINGLE | GLUT_RGB);
```

- The above command specifies that a single refresh buffer is to be used for the display window and that the RGB (red, green, blue) color mode is to be used for selecting color values.

Q.4 Give the format of OpenGL command.

Ans. : The Fig. Q.4.1 shows the format of OpenGL command.

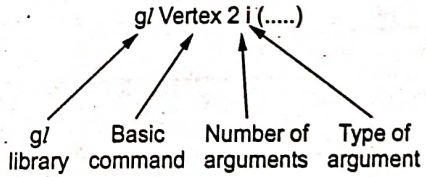


Fig. Q.4.1 Format of OpenGL command

Q.5 Explain the vertex function in OpenGL.

Ans. : • In OpenGL, the general syntax for vertex function is

```
glvertex * ( )
```

where the * can be interpreted as either two or three characters of the form nt or ntv, where

- n : The number of dimensions (2, 3 or 4)
- t : Denotes data type
  - i : integer
  - f : float
  - d : double
  - v : indicates that the variables are specified through a pointer to an array, rather than through an argument list.

Q.6 State various forms of vertex function in OpenGL.

Ans. : glvertex2i (GLint xi, GLint yi) : Function works in two dimension with integers.

glvertex3f (GLfloat x, GLfloat y, GLfloat z) : Function works in three dimension with floating point numbers.

glvertex3fv (vertex) : Function works in three dimension with floating point numbers.

Q.7 State the data types supported by OpenGL.

Ans. : Table Q.7.1 shows the data types supported by OpenGL.

Suffix	Data type	C Type	OpenGL Type
b	8-bit integer	signed char	GLbyte
s	16-bit integer	short	GLshort
i	32-bit integer	int or long	GLint, GLsizei
f	32-bit floating point	float	GLfloat, GLclampf
d	64-bit floating point	double	GLdouble, GLclampd
ub	8-bit unsigned integer	unsigned char	GLubyte, GLboolean
us	16-bit unsigned integer	unsigned short	GLushort
ui	32-bit unsigned integer	unsigned int	GLuint, GLenum, GLbitfield
	Nothing	void	GLvoid

Table Q.7.1 OpenGL data types

Q.8 Explain various types of functions supported by OpenGL.

Ans. : • An API for interfacing with graphics system can contain hundreds of individual functions. These functions are divided into seven major groups :

1. Primitive functions
2. Attribute functions
3. Viewing functions
4. Transformation functions
5. Input functions
6. Control functions
7. Query functions

- **Primitive functions** : These functions define the low-level objects or atomic entities that our system can display. Depending on the API, the primitives can include points, line segments, polygons, pixels, text and various types of curves and surfaces.
- **Attribute functions** : These functions allow us to perform operations ranging from choosing the color with which we display a line segment, to picking a pattern with which to fill the inside of a polygon, to selecting a typeface for the titles on a graph.
- **Viewing functions** : These functions allow us to specify various views, although APIs differ in the degree of flexibility they provide in choosing a view.
- **Transformation functions** : These functions allow us to carry out transformations of objects, such as rotation, translation and scaling. Providing the user with a set of transformation functions is one of the characteristics of a good API.
- **Input functions** : These functions allow us to deal with the diverse forms of input that characterize modern graphics systems. These functions include the functions to deal with devices such as keyboards, mice and data tablets.
- **Control functions** : These functions enable us to communicate with the window system, to initialize our programs and to deal with any errors that take place during the execution of our programs.
- **Query functions** : Within our applications we can often use other information within the API, including camera parameters or values in the frame buffer. A good API provides this information through a set of query functions.

2.2 : Primitives and Attributes

Q.9 List out different OpenGL primitives.

Ans. : OpenGL provides ten different primitive types for drawing points, lines, and polygons, as shown in Fig. Q.9.1.

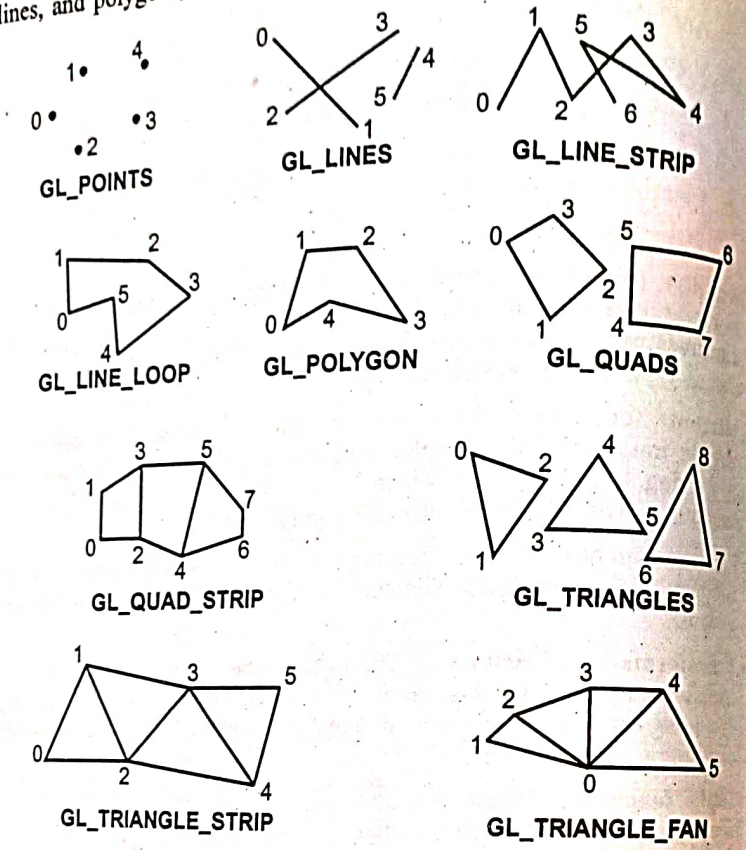


Fig. Q.9.1 OpenGL primitive types

GL_POINTS

- We use this primitive type to render mathematical points.
- **Example :**

```
glBegin(GL_POINTS);
glVertex2f(x1, y1);
glEnd();
```


GL_LINES

- We use this primitive to draw unconnected line segments.
- **Example :**

```
glBegin(GL_LINES);
glVertex2f(x1, y1);
glVertex2f(x2, y2);
glEnd();
```

GL_LINE_STRIP

- We use this primitive to draw a sequence of connected line segments.
- **Example :**

```
glBegin(GL_LINE_STRIP);
glVertex2f(x1, y1);
glVertex2f(x2, y2);
glVertex2f(x3, y3);
glEnd();
```

GL_LINE_LOOP

- We use this primitive to close a line strip.
- **Example :**

```
glBegin(GL_LINE_LOOP);
glVertex2f(x1, y1);
glVertex2f(x2, y2);
glVertex2f(x3,y3);
glEnd();
```

GL_TRIANGLES

- We use this primitive to draw individual triangles.
- **Example :**

```
glBegin(GL_TRIANGLES);
glVertex2f(x1, y1);
glVertex2f(x2, y2);
glVertex2f(x3, y3);
glEnd();
```

GL_TRIANGLE_STRIP

- We use this primitive to draw a sequence of triangles that share edges.
- **Example :**

```
glBegin(GL_TRIANGLE_STRIP);
glVertex2f(x1, y1);
glVertex2f(x2, y2);
glVertex2f(x3, y3);
glVertex2f(x4, y4);
glEnd();
```

GL_TRIANGLE_FAN

- We use this primitive to draw a fan of triangles that share edges and also share a vertex. Each triangle shares the first vertex specified.
- **Example :**

```
glBegin(GL_TRIANGLE_FAN);
glVertex2f(x1, y1);
glVertex2f(x2, y2);
glVertex2f(x3, y3);
glVertex2f(x4, y4);
glEnd();
```

GL_QUADS

- We use this primitive to draw individual convex quadrilaterals.
- **Example :**

```
glBegin(GL_QUADS);
glVertex2f(x1, y1);
glVertex2f(x2, y2);
glVertex2f(x3, y3);
glVertex2f(x4, y4);
glEnd();
```

GL_QUAD_STRIP

- We use this primitive to draw a sequence of quadrilaterals that share edges.

- **Example :**

```
glBegin(GL_QUADS);
glVertex2f(x1, y1);
glVertex2f(x2, y2);
glVertex2f(x3,y3);
glVertex2f(x4,y4);
glEnd();
```

GL_POLYGON

- We use GL_POLYGON to draw a single filled convex n-gon primitive. OpenGL renders an n-sided polygon, where n is the number of vertices specified by the application. If n is less than 3, OpenGL renders nothing.
- **Example :**

```
glBegin(GL_POLYGON);
glVertex2f(x1, y1);
glVertex2f(x2, y2);
glVertex2f(x3, y3);
glVertex2f(x4, y4);
glVertex2f(x5, y5);
glEnd();
```

Q.10 List OpenGL point attributes.

Ans. : • Besides changing the color of a point, we can also specify the point size. We set the size for an OpenGL point with :

```
glPointSize (size);
```

- Parameter size is assigned a positive floating point value, which is rounded to an integer (unless the point is to be anti-aliased). A point size of 1.0 (the default value) displays a single pixel and a point size of 2.0 displays a 2 by 2 pixel array.

Q.11 List OpenGL line attributes.

Ans. : • Line width : It can be changed using the following OpenGL function :

```
glLineWidth (width);
```

- We assign a floating-point value to parameter width and this value is rounded to the nearest non-negative integer.

• We set a current display style for lines with this OpenGL function :

```
glLineStipple (repeatFactor, pattern);
```

• Parameter pattern is used to reference a 16-bit integer that describes how the line should be displayed. A 1 in the bit-pattern denotes an "on" pixel position. The pattern is applied to the pixels along the line path starting with the low-order bits in the pattern. The default pattern is 0xFFFF which produces a solid line.

• As an example of specifying a line style, the following function call results in dashed lines :

```
glLineStipple (1, 0x00FF);
```

• Integer parameter **repeatFactor** specifies how many times each bit in the pattern is to be repeated before the next bit in the pattern is applied. The default repeat value is 1.

• Before a line can be displayed in the current line-style pattern, we must activate the line-style feature of OpenGL. We accomplish this with the following function :

```
glEnable (GL_LINE_STIPPLE);
```

• Without enabling this feature, lines would still appear as solid lines, even though a pattern was provided.

• To disable the use of a pattern we can issue the following function call :

```
glDisable (GL_LINE_STIPPLE);
```

Q.12 List OpenGL fill attributes.

Ans. : • By default, a polygon is displayed as a solid-color region, using the current color setting.

• To fill the polygon with a pattern in OpenGL, a 32-bit by 32-bit bit mask has to be specified similar to defining a line-style :

```
GLubyte pattern [] = { 0xff, 0x00, 0xff, 0x00, ...};
```

• Once we have set the mask, we can establish it as the current fill pattern :

```
glPolygonStipple (pattern);
```

• We need to enable the fill routines before we specify the vertices for the polygons that are to be filled with the current pattern. We do this with the statement :

```
glEnable (GL_POLYGON_STIPPLE);
```


• Similarly, we turn off the pattern filling with :

```
glDisable (GL_POLYGON_STIPPLE);
```

Q.13 List OpenGL character attributes.

Ans. : • We have two methods for displaying characters with the OpenGL. Either we can design a font set using the bitmap functions in the core library or we can invoke the GLUT character-generation routines.

• The GLUT library contains functions for displaying predefined bitmap and stroke character sets. Therefore, the character attributes we can set are those that apply to either bitmaps or line segments.

• For either bitmap or outline fonts, the display color is determined by the current color state.

• In general, the spacing and size of characters is determined by the font designation, such as GLUT BITMAP 9 BY 15 and GLUT STROKE MONO ROMAN. However, we can also set the line width and line type for the outline fonts.

• We specify the width for a line with the glLineWidth function and we select a line type with the glLineStipple function. The GLUT stroke fonts will then be displayed using the current values we specified for the OpenGL line-width and line-type attributes.

2.3 : Simple Modelling and Rendering of 2D and 3D Geometric Objects

Q.14 Write a short note on modelling and rendering.

Ans. : • Two principal tasks are required to create an image of a two or three-dimensional scene : **Modeling** and **rendering.**

• The modeling task generates a model, which is the description of an object that is going to be used by the graphics system. Models must be created for every object in a scene; they should accurately capture the geometric shape and appearance of the object.

• The second task, rendering, takes models as input and generates pixel values for the final image.

• OpenGL supports a handful of primitive types for modeling two-dimensional (2D) and three-dimensional (3D) objects : Points, lines, triangles, quadrilaterals and (convex) polygons.

- In addition, OpenGL includes support for rendering higher-order surface patches using evaluators. A simple object, such as a box, can be represented using a polygon for each face in the object.
- Part of the modeling task consists of determining the 3D coordinates of each vertex in each polygon that makes up a model.
- To provide accurate rendering of a model's appearance or surface shading, the modeler may also have to determine color values, shading normals and texture coordinates for the model's vertices and faces.
- Complex objects with curved surfaces can also be modeled using polygons. A curved surface is represented by a gridwork or mesh of polygons in which each polygon vertex is placed on a location on the surface.

Q.15 Explain the modelling of a coloured cube.

Ans. : A cube is a simple 3D object.

- Here, we use surface based model. It represents cube either as the intersection of six planes or as the six polygons, called **facets**. Facets define the faces of cube.

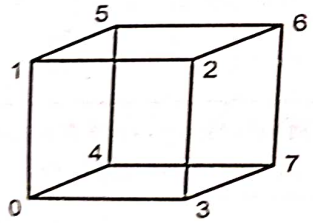


Fig. Q.15.1

- The following code statement initializes an array with the vertices of a cube.

```
GLFloat vertices [8] [3] = {   {– 1.0, – 1.0, – 1.0},
                               { 1.0, – 1.0, – 1.0},
                               {1.0, 1.0, – 1.0},
                               {– 1.0, 1.0, – 1.0},
                               {– 1.0, – 1.0, 1.0},
                               { 1.0, – 1.0, 1.0},
                               { 1.0, 1.0, 1.0},
                               {– 1.0, 1.0, 1.0 }
                           };
```


- Now we can specify the faces of the cube. An example, code to specify one face is as given below :

```
glBegin (GL_POLYGON) ;
    glVertex3fv (vertices [0]) ;
    glVertex3fv (vertices [3]) ;
    glVertex3fv (vertices [2]) ;
    glVertex3fv (vertices [1]) ;
glEnd ( ) ;
```

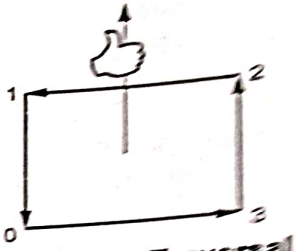


Fig. Q.15.2 Traversal of the edges of a polygon

- Similarly the other five faces can be defined. The other five faces are - (2, 3, 7, 6), (0, 4, 7, 3), (1, 2, 6, 5), (4, 5, 6, 7) and (0, 1, 5, 4). Here, the order of traversal of the edges of cube faces is determined by **right hand rule.**

2.4 : Interaction

Q.16 What is interaction ?

Ans. : Interaction in the field of computer graphics refers to the process of enabling the users to interact with computer displays. The user can observe the image change in response to the input from the user and act accordingly.

Q.17 Write a short note on OpenGL events.

Ans. : • The OpenGL supports basic window events : The window has been resized, the window needs to draw its contents, a key on the keyboard has been pressed or released, a mouse has moved or mouse button was pressed or released. All such event processing is based on callbacks. It is necessary to write functions to process specific events and set them as callbacks for a specific window. When event occurs, a corresponding callback function is called.

- GLUT handles events with **callback functions.** If you want to handle an event, such as a keystroke or mouse movement, you write a function that performs the desired action. Then you **register** your function by passing its name as an argument to a function with a name of the form glut...Func(), in which "..." indicates which callback you are registering.

Q.18 Explain the keyboard callback function supported by OpenGL.

Ans. : • If the program has registered a keyboard callback function, the keyboard callback function is called whenever the user presses a key.

1. Register a keyboard callback function.

```
glutKeyboardFunc(keyboard);
```

2. A keyboard callback function. It receives three arguments : The key that was pressed and the current X and Y coordinates of the mouse.

- The callback function registered by **glutKeyboardFunc()** recognizes only keys corresponding to ASCII graphic characters and esc, backspace and delete. The keyboard callback function in Fig. Q.18.1 is a useful default keyboard function : It allows the user to quit the program by pressing the escape key.

```
#define ESCAPE 27
void keyboard (unsigned char key, int x, int y)
{
    if (key == ESCAPE)
exit(0);
}
```

Fig. Q.18.1 Quitting with the escape key

- To make your program respond to other keys, such as the arrow keys and the function ("F") keys :
  1. Register a special key callback function.

     glutSpecialFunc(special);
  2. Declare the special key function as follows :

GLUT_KEY_F1	GLUT_KEY_F8	GLUT_KEY_UP
GLUT_KEY_F2	GLUT_KEY_F9	GLUT_KEY_DOWN
GLUT_KEY_F3	GLUT_KEY_F10	GLUT_KEY_PAGE_UP
GLUT_KEY_F4	GLUT_KEY_F11	GLUT_KEY_PAGE_DOWN
GLUT_KEY_F5	GLUT_KEY_F12	GLUT_KEY_HOME
GLUT_KEY_F6	GLUT_KEY_LEFT	GLUT_KEY_END
GLUT_KEY_F7	GLUT_KEY_RIGHT	GLUT_KEY_INSERT

Table Q.18.1 Constants for special keys


```
void special (int key, int x, int y)
{
switch (key)
{
case GLUT_KEY_F1:
      // code to handle F1 key
      break;
      ....
}
}
```

- The arguments that OpenGL passes to special() are : The key code, as defined in Table Q.18.1, the X co-ordinate of the mouse and the Y co-ordinate of the mouse.

Q.19 Explain the mouse event callback function.

Ans. : • The mouse event callback function responds to pressing or releasing one of the mouse buttons.

1. Register a mouse callback function.

   glutMouseFunc(mouse_button);
2. The mouse callback function is passed four arguments.

- The first argument specifies the button. Its value is one of
  - GLUT_LEFT_BUTTON,
  - GLUT_MIDDLE_BUTTON, or
  - GLUT_RIGHT_BUTTON
- The second argument specifies the button state. Its value is one of
  - GLUT_DOWN (the button has been pressed) or
  - GLUT_UP (the button has been released).
- The remaining two arguments are the X and Y co-ordinates of the mouse.

Q.20 Write a note on window event.

Ans. : • Most window systems allow a user to resize the window interactively, usually by using the mouse to drag a corner of the window to a new location.

- Whenever the user moves or resizes the graphics window, OpenGL informs your program, provided that you have registered the appropriate **callback** function. The main problem with reshaping is that the user is likely to change the shape of the window as well as its size; you do not want the change to distort your image. Register a callback function that will respond to window reshaping :

```
glutReshapeFunc(reshape);
```

- The above function sets the reshape callback for the current window.
- The reshape callback is triggered when a window is reshaped. A reshape callback is also triggered immediately before a window's first display callback after a window is created.

Reshape Callback Function

```
void reshape (GLsize w, GLsize h)
{
    glMatrixMode (GL_PROJECTION);
    glLoadIdentity( );
    gluOrtho2D (0.0, (GLdouble)w, 0.0, (GLdouble)h);
    glMatrixMode (GL_MODELVIEW);
    glLoadIdentify ( );
    glViewport (0, 0, w, h);
}
```

- The width and height parameters of the callback specify the new window size in pixels.
- We use these values to create a new OpenGL clipping window using gluOrtho2D, as well as a new viewport with the same aspect ratio.
- Another event such as a window movement without resizing are handled using following function call.

```
glutMotionFunc( );
```

- This function sets the motion callback for the current window.

2.5 : Picking

Q.21 What is picking ?

Ans. : • In science and engineering 3D visualization computer applications, it is useful for the user to point to something and have the program figure out what is being pointed to. This is called **picking**.

Q.22 State the steps involved in picking process when back buffer approach is used.

Ans. : Steps involved are as follows :

- Draw the objects into the back buffer with different colours.
- Get the position of mouse using callback function.
- Find the colour at the position in the frame buffer corresponding to the mouse position using **glReadPixel( )** function.
- Identify the object using pixel colour.

Q.23 Explain the selection mode approach to implement picking in OpenGL.

Ans. : • OpenGL uses selection mode approach to implement picking.

- When we plan to use OpenGL's selection mode, we have to first draw our scene into the frame buffer and then we enter selection mode and redraw the scene. However, once we are in selection mode, the contents of the frame buffer don't change until we exit selection mode.
- When we exit selection mode, OpenGL returns a list of the primitives that intersect the viewing volume (remember that the viewing volume is defined by the current model view and projection matrices and any additional clipping planes).
- Each primitive that intersects the viewing volume causes a selection **hit**. The list of primitives is actually returned as an array of integer-valued **names** and related data - the **hit records** - that correspond to the current contents of the **name stack**.
- We construct the name stack by loading names onto it as we issue primitive drawing commands while in selection mode. Thus, when the list of names is returned, we can use it to determine which primitives might have been selected on the screen by the user.
- In addition to this selection mode, OpenGL provides a utility routine designed to simplify selection in some cases by restricting drawing to a small region of the viewport. Typically, we use this routine to determine which objects are drawn near the cursor, so that we can identify which object the user is picking.

Q.24 Explain the steps to be performed in the selection mode.

Ans. : • To use the selection mode, we need to perform the following steps :

1. Specify the array to be used for the returned hit records with **glSelectBuffer()**.
2. Enter selection mode by specifying GL_SELECT with glRenderMode().
3. Initialize the name stack using **glInitNames()** and **glPushName()**.
4. Define the viewing volume we want to use for selection. Usually this is different from the viewing volume we originally used to draw the scene, so we probably want to save and then restore the current transformation state with **glPushMatrix()** and **glPopMatrix()**.
5. Alternately issue primitive drawing commands and commands to manipulate the name stack so that each primitive of interest has an appropriate name assigned.
6. Exit selection mode and process the returned selection data (the hit records).

Q.25 How to use selection mode for picking ?

Ans. : • We can use selection mode to determine if objects are picked. To do this, we use a special picking matrix in conjunction with the projection matrix to restrict drawing to a small region of the viewport, typically near the cursor.

- Then we allow some form of input, such as clicking a mouse button, to initiate selection mode.
- With selection mode established and with the special picking matrix used, objects that are drawn near the cursor cause selection hits. Thus, during picking we are typically determining which objects are drawn near the cursor.
- Picking is set up almost exactly like regular selection mode is, with the following major differences.
  - Picking is usually triggered by an input device.
  - We use the utility routine **gluPickMatrix()** to multiply a special picking matrix onto the current projection matrix. This routine should be called prior to multiplying a standard projection matrix (such as **gluPerspective()** or **glOrtho()**). We

will probably want to save the contents of the projection matrix first, so the sequence of operations may look like this :

```
glMatrixMode (GL_PROJECTION);
glPushMatrix ();
glLoadIdentity ();
gluPickMatrix (...);
gluPerspective, glOrtho, gluOrtho2D, or glFrustum
 /* ... draw scene for picking ; perform picking ... */
glPopMatrix();
```

END...

Unit I

3 Scan Conversion

3.1 : Lines

Q.1 Define line and line segment. ☞ **[SPPU : Dec.-07,11, Marks 2]**

Ans. : Line is a straight object with no curves, no thickness and it extends in both directions without end. If it does have ends it is called a line segment.

Q.2 Define vectors. ☞ **[SPPU : Dec.-07,11, Marks 2]**

Ans. : A vector is an quantity having direction as well as magnitude, especially as determing the position of one point in space relative to another.

3.2 : Line Drawing Algorithms

Q.3 What is vector generation ? Explain the problem of vector generation. ☞ **[SPPU : Dec.-12, Marks 6]**

Ans. : The process of 'turning on' the pixels for a line segment is called vector generation or line generation and the algorithms for them are known as vector generation algorithms or line drawing algorithms.

Problems of vector generation :

- The 45° line is straight but its width is not constant.
- The line with any other orientation is neither straight nor has same width. Such cases are due to the finite resolution of display and we have to accept approximate pixels in such situations.
- The brightness of the line is dependent on the orientation of the line.
- We can observe that the effective spacing between pixels for the 45° line is greater than for the vertical and horizontal lines. This will make the vertical and horizontal lines appear brighter than the 45° line.

- Complex calculations are required to provide equal brightness along lines of varying length and orientation. Therefore, to draw line rapidly some compromises are made such as
  - Calculate only an approximate line length
  - Reduce the calculations using simple integer arithmetic
  - Implement the result in hardware or firmware

Q.4 Explain DDA algorithm for line. Discuss its advantages and disadvantages.

☞ **[SPPU : May-05, 10, 11, 17, 19, Dec.-06,07,08, Marks 10]**

Ans. : DDA Line Algorithm

1. Read the line end points (x_1, y_1) and (x_2, y_2) such that they are not equal.

   [if equal then plot that point and exit]
2. $\Delta x = |x_2 - x_1|$ and $\Delta y = |y_2 - y_1|$
3. if $(\Delta x \geq \Delta y)$ then

   length = Δx

   else

   length = Δy

   end if
4. $\Delta x = (x_2 - x_1) / \text{length}$

   $\Delta y = (y_2 - y_1) / \text{length}$

   [This makes either Δx or Δy equal to 1 because length is either $|x_2 - x_1|$ or $|y_2 - y_1|$. Therefore, the incremental value for either x or y is one.]
5. $x = x_1 + 0.5 * \text{Sign}(\Delta x)$

   $y = y_1 + 0.5 * \text{Sign}(\Delta y)$

   Here, Sign function makes the algorithm work in all quadrant. It returns – 1, 0, 1 depending on whether its argument is < 0, = 0, > 0 respectively. The factor 0.5 makes it possible to round the values in the integer function rather than truncating them.

   plot (Integer (x), Integer (y))
6. i = 1

   [Begins the loop, in this loop points are plotted]

   While (i ≤ length)

   {

```
        x = x + Δx
        y = y + Δy
        plot (Integer (x), Integer (y))
        i = i + 1
    }
7. Stop
```

Advantages of DDA Algorithm

1. It is the simplest algorithm and it does not require special skills for implementation.
2. It is a faster method for calculating pixel positions than the direct use of equation $y = mx + b$. It eliminates the multiplication in the equation by making use of raster characteristics, so that appropriate increments are applied in the x or y direction to find the pixel positions along the line path.

Disadvantages of DDA Algorithm

1. Floating point arithmetic in DDA algorithm is still time-consuming.
2. The algorithm is orientation dependent. Hence end point accuracy is poor.

Q.5 Consider the line from (0, 0) to (– 6, – 6). Use the simple DDA algorithm to rasterize this line.

☞ **[SPPU : June-12, May-17, Marks 10]**

Ans. : Evaluating steps 1 to 5 in the DDA algorithm we have,

$x_1 = 0$

$y_1 = 0$

$x_2 = -6$

$y_2 = -6$

$\therefore$ Length $= |x_2 - x_1| = |y_2 - y_1| = 6$

$\therefore$ $\Delta x = \Delta y = -1$

Initial values for

$x = 0 + 0.5 * \text{Sign}(-1)$

$= -0.5$

$y = 0 + 0.5 * \text{Sign}(-1)$

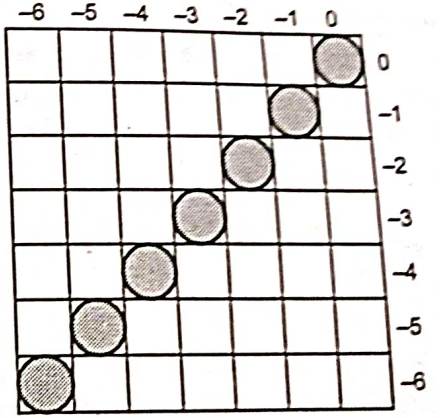
$= -0.5$

Tabulating the results of each iteration in the step 7 we get,

i	x	y	Plot
1	– 0.50	– 0.50	(0, 0)
2	– 1.50	– 1.50	(– 1, – 1)
3	– 2.50	– 2.50	(– 2, – 2)
4	– 3.50	– 3.50	(– 3, – 3)
5	– 4.50	– 4.50	(– 4, – 4)
6	– 5.50	– 5.50	(– 5, – 5)
7	– 6.50	– 6.50	(– 6, – 6)

Table Q.5.1

The results are plotted as shown in the Fig. Q.5.1. It shows that the rasterized line lies on the actual line and it is 45° line.



* –ve pixel values are with reference to pixel at the center of screen

Fig. Q.5.1 Result for a simple DDA

Q.6 Using DDA algorithm find out which pixels would be turned on for the line with end points (1, 1) to (5, 3).

☞ **[SPPU : May-15, Marks 4]**

Ans. : $x_1 = 1,\ y_1 = 1,\ x_2 = 5, y_2 = 3$

$\therefore$ Length $= |x_2 - x_1| = 5 - 1 = 4$

$\therefore \quad \Delta x = |x_2 - x_1| / \text{length} = \frac{4}{4} = 1$

$\therefore \quad \Delta y = |y_2 - y_1| / \text{length} = \frac{2}{4} = 0.5$

Initial value for

$x = 1 + 0.5 * \text{sign}(1) = 1.5$

$y = 1 + 0.5 * \text{sign}(0.5) = 1.5$

i	x	y	Plot
1	1.5	1.5	(1, 1)
2	2.5	2.0	(2, 2)
3	3.5	2.5	(3, 2)
4	4.5	3.0	(4, 3)
5	5.5	3.5	(5, 3)

Q.7 Consider line segment from A(– 2, – 1) to B(6, 3) use DDA line drawing algorithm to rasterize this line.

☞ [SPPU : May-19, Marks 6]

Ans. : *Evaluating steps 1 to 5 in the DDA algorithm we have,*

$x_1 = -2, y_1 = -1, x_2 = 6$ and $y_2 = 3$

$\Delta x = |x_2 - x_1| = |6 - (-2)| = 8$

$\Delta y = |y_2 - y_1| = |3 - (-1)| = 4$

$\therefore$ Length = 8

$\Delta x = (x_2 - x_1)/\text{Length} = 8/8 = 1$

$\Delta y = (y_2 - y_1)/\text{Length} = 4/8 = 0.5$

i	x	y	plot
1	– 2	– 1	plot(– 2, – 1)
2	– 1	– 0.5	plot(– 1, – 1)
3	0	0	plot(0, 0)
4	1	0.5	plot(1, 0)
5	2	1	plot(2, 1)
6	3	1.5	plot(3, 1)

7	4	2	plot(4, 2)
8	5	2.5	plot(5, 2)
9	6	3	plot(6, 3)

Q.8 Scan convert a line with end points (10, 5) and (16, 10) using DDA line drawing algorithm. **☞ [SPPU : May-18, Marks 4]**

Ans. : $x_1 = 10$, $x_2 = 16$, $y_1 = 5$ and $y_2 = 10$

$\therefore$ Length $= |x_2 - x_1| = 16 - 10 = 6$

$\Delta y = |y_2 - y_1| / \text{length} = \frac{10-5}{6} = \frac{5}{6}$

$\Delta x = |x_2 - x_1| / \text{length} = \frac{6}{6} = 1$

Initial values for,

$x = 10 + 0.5 * \text{sign}(1) = 10.5$

$y = 5 + 0.5 * \text{sign}(5/6) = 5.5$

i	x	y	Plot
1	10.50	5.50	(10, 5)
2	11.50	6.33	(11, 6)
3	12.50	7.16	(12, 7)
4	13.50	8.99	(13, 8)
5	14.50	8.83	(14, 8)
6	15.50	9.66	(15, 9)
7	16.50	10.49	(16, 10)

Q.9 Explain Bresenham's line drawing algorithm.

☞[SPPU : Dec.-07,12,17,18, May-14,15,17,19, Marks 3]

Ans. : Bresenham's line drawing algorithms for $0 < m < 1$

1. Read the line end points (x_1, y_1) and (x_2, y_2) such that they are not equal.

   [if equal then plot that point and exit]

2. $\Delta x = |x_2 - x_1|$ and $\Delta y = |y_2 - y_1|$

3. [Initialize starting point]

   $x = x_1$

   $y = y_1$

   Plot (x, y) ;

4. $e = 2 * \Delta y - \Delta x$

   [Initialize value of decision variable or error to compensate for nonzero intercepts]

5. $i = 1$

   [Initialize counter]

6. while ($e \geq 0$)

   {

   $y = y + 1$

   $e = e - 2 * \Delta x$

   }

   $x = x + 1$

   $e = e + 2 * \Delta y$

7. Plot (x, y)
8. $i = i + 1$
9. if ($i \leq \Delta x$) then go to step 6.
10. Stop

Q.10 Using Bresenham's line algorithm, find out which pixel would be turned on for the line with end points (4, 4) to (12, 9).

☞ [SPPU : Dec.-10, Marks 8]

Ans. : Evaluating step 1 through 4 in the Bresenham's algorithm we have,

$\Delta x = |12 - 4| = 8, \quad \Delta y = |9 - 4| = 5$

$x = 4, \quad y = 4$

$e = 2 * \Delta y - \Delta x = 2 * 5 - 8 = 2$

Tabulating the results of each iteration in the step 5 through 10.

i	e	x	y	Plot
1	2	4	4	(4, 4)
2	– 4	5	5	(5, 5)
3	6	6	5	(6, 5)
4	0	7	6	(7, 6)
5	– 6	8	7	(8, 7)
6	4	9	7	(9, 7)
7	– 2	10	8	(10, 8)
8	8	11	8	(11, 8)
9	2	12	9	(12, 9)

Q.11 Consider the line from (1, 1) to (6, 4). Use Bresenham's line drawing algorithm to rasterize this line and give output pixels.

☞[SPPU : Dec.-11, Marks 10]

Ans. : Evaluating step 1 through 4 in Bresenham's algorithm we have,

$\Delta x = |6 - 1| = 5, \quad \Delta y = |4 - 1| = 3$

$x = 1, \quad y = 1$

$e = 2 * \Delta y - \Delta x = 2 * 3 - 5 = 1$

Tabulating the results of each iteration in the step 5 through 10.

i	e	x	y	Plot
1	1	1	1	(1, 1)
2	– 3	2	2	(2, 2)
3	3	3	2	(3, 2)
4	– 1	4	3	(4, 3)
5	5	5	3	(5, 3)
6	1	6	4	(6, 4)

Q.12 Consider a line from (4, 9) to (7, 7). Use Bresenham's line drawing algorithm to rasterize this line.

☞[SPPU : June-12, Marks 10]

Ans. : Evaluating step 1 through 4 in Bresenham's algorithm we have,

$\Delta x = |7 - 4| = 3, \quad \Delta y = |7 - 9| = 2$

$x = 4, \quad y = 9$

$e = 2 * \Delta y - \Delta x = 2 * 2 - 3 = 1$

Tabulating the results of each iteration in the step 5 through 10.

i	e	x	y	Plot
1	1	4	9	(4, 9)
2	-1	5	8	(5, 8)
3	3	6	8	(6, 8)
4	1	7	7	(7, 7)

Q.13 Consider the line from (0, 0) to (6, 6). Use Bresenham's algorithm to rasterize this line.

☞[SPPU : May-13, Dec.-13, May-16, Marks 6, Dec.-17, Marks 6]

Ans. : Evaluating step 1 through 4 in the Bresenham's algorithm we have,

$\Delta x = |6 - 0| = 6, \quad \Delta y = |6 - 0| = 6$

$x = 0, \quad y = 0$

$e = 2 * \Delta y - \Delta x = 2 * 6 - 6 = 6$

Tabulating the results of each iteration in the step 5 through 10.

i	e	x	y	Plot
1	6	0	0	(0, 0)
2	6	1	1	(1, 1)
3	6	2	2	(2, 2)
4	6	3	3	(3, 3)
5	6	4	4	(4, 4)
6	6	5	5	(5, 5)
7	6	6	6	(6, 6)

Q.14 Differentiate Bresenham and Vector generation algorithm for line. **☞[SPPU : May-11, Marks 8; Dec.-17, Marks 6]**

Ans. :

Sr. No.	Vector generation algorithm	Bresenham's line algorithm
1.	It uses floating point arithmetic.	It uses only integer addition, subtraction and multiplication by 2.
2.	Due to floating point arithmetic it takes more time.	It is quicker than vector generation algorithm.
3.	Less efficient.	More efficient.
4.	Where speed is important this algorithm needs to be implemented in hardware.	Hardware implementation is not required.

Q.15 Find out which pixel would be turned on for the line with end points (2, 2) to (6, 5) using the same.

☞ [SPPU : May-14, Marks 3]

Ans. : Evaluating step 1 through 4 in the Bresenham's algorithm.

$\Delta x = |6 - 2| = 4, \quad \Delta y = |5 - 2| = 3$

$x = 2, \quad y = 2$

$e = (2 \times \Delta y) - \Delta x = (2 \times 3) - 4 = 2$

Tabulating the result of each iteration in the step 5 through 10.

i	e	x	y
1	2	2	2
2	0	3	3
3	-2	4	4
4	4	5	4
5	2	6	5

Q.16 Explain Bresenham's line algorithm and find out which pixels would be turned on for the line with end points (5, 2) to (8, 4) using the same. **☞[SPPU : Dec.-14, Marks 6]**

Ans. : $\Delta x = |8 - 5| = 3$, $\Delta y = |4 - 2| = 2$, $x = 5, y = 2$

$e = 2 * \Delta y - \Delta x = 2 * 2 - 3 = 1$

i	e	x	y	Plot
1	1	5	2	(5, 2)
2	– 1	6	3	(6, 3)
3	3	7	3	(7, 3)
4	1	8	4	(8, 4)

Pixels would be turned on for the line : (5, 2), (6, 3), (7, 3), (8, 4).

Q.17 Consider a line from (2, 5) to (8, 8). Use Bresenham's line drawing algorithm rasterize this line. ☞ **[SPPU Dec.-19, Marks 6]**

Ans. : Evaluating step 1 through step 4 in the Bresenham's algorithm we have,

$$\Delta x = |8 - 2| = 6, \Delta y = |8 - 5| = 3$$

$$x = 2, y = 5$$

$$e = 2 * \Delta y - \Delta x = 2 * 3 - 6 = 0$$

Tabulating the result of each iteration in the step 5 through 10

i	e	x	y	Plot
1	0	2	5	(2, 5)
2	– 6	3	6	(3, 6)
3	0	4	6	(4, 6)
4	– 6	5	7	(5, 7)
5	0	6	7	(6, 7)
6	– 6	7	8	(7, 8)
7	0	8	8	(8, 8)

Q.18 Explain how DDA line algorithm can be extended to generate a thick line of thickness 'w'. ☞ **[SPPU : Dec.-10, Marks 8]**

Ans. : • To produce a thick line, we have to run two line drawing algorithms in parallel to find the pixels along the line edges, and while

stepping along the line we have to turn on all the pixels which lie between the boundaries. This is illustrated in Fig. Q.18.1.

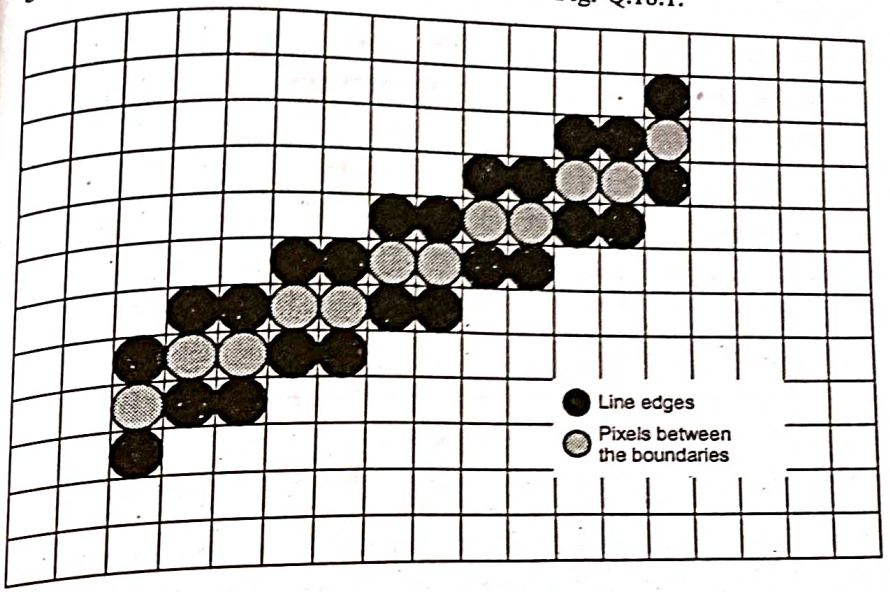


Fig. Q.18.1 Thick line

- Let us consider line from point (x_1, y_1) to (x_2, y_2) having thickness w, then we have a top boundary between the points $(x_1, y_1 + w_y)$ and $(x_2, y_2 + w_y)$ and a lower boundary between $(x_1, y_1 - w_y)$ and $(x_2, y_2 - w_y)$ where w_y is given by

$$w_y = \frac{(w-1)}{2} \frac{\left[(x_2 - x_1)^2 + (y_2 - y_1)^2\right]^{1/2}}{|x_2 - x_1|}$$

- Here, w_y is the amount by which the boundary lines are moved from the line center, as shown in the Fig. Q.18.2. The factor $(w - 1)$ in the above equation exist because the line boundary itself has a thickness of one pixel. We further divide the factor $(w - 1)$ by 2 because half the thickness will be used

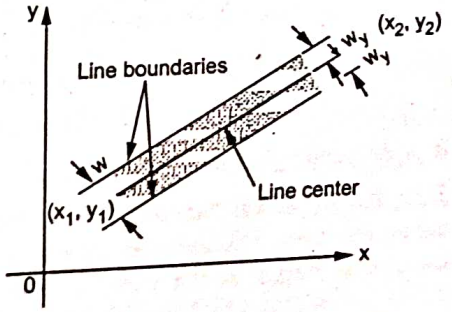


Fig. Q.18.2 Thick line details

to offset the top boundary, and the other half to move the bottom boundary.

- We can use equation for w_y for lines having slope less than 1. For sharp slope lines, i.e. lines having slope greater than 1 lines are handled similarly with the x and y roles reversed. In this case w_x is given as

$$w_x = \frac{(w-1)}{2} \cdot \frac{\left[(x_2 - x_1)^2 + (y_2 - y_1)^2\right]^{1/2}}{|y_2 - y_1|}$$

- Thus, left and right boundaries are $(x_1 - w_x, y_1)$ to $(x_2 - w_x, y_2)$ and $(x_1 + w_x, y_1)$ to $(x_2 + w_x, y_2)$, respectively.

3.3 : Antialiasing and Antialiasing Techniques

Q.19 What is aliasing and antialiasing ?

Ans. : • In the line drawing algorithms, we have seen that all rasterized locations do not match with the true line and we have to select the optimum raster locations to represent a straight line. This problem is severe in low resolution screens. In such screens line appears like a stair-step, as shown in the Fig. Q.19.1. This effect is known as **aliasing**.

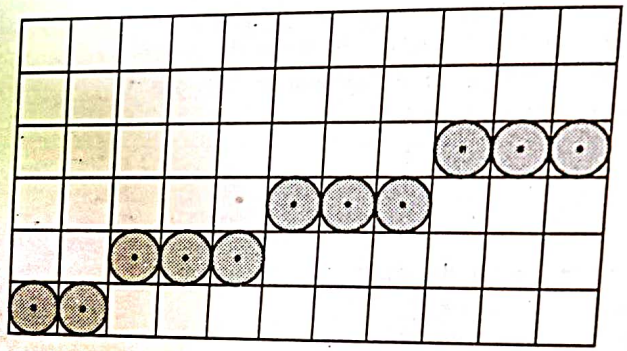


Fig. Q.19.1 Aliasing effect

- It is dominant for lines having gentle and sharp slopes.
- The aliasing effect can be reduced by adjusting intensities of the pixels along the line. The process of adjusting intensities of the pixels along the line to minimize the effect of aliasing is called **antialiasing**.

Q.20 List and explain any two antialiasing methods.

[SPPU : Dec.-10,13,14, May-13, Marks 4]

Ans. : Anti-aliasing methods are basically classified as

- **Supersampling or Postfiltering :** In this antialiasing method the sampling rate is increased by treating the screen as if it were covered with a finer grid than is actually available. We can then use multiple sample points across this finer grid to determine an appropriate intensity level for each screen pixel. This technique of sampling object characteristics at a high resolution and displaying the results at a lower resolution is called **supersampling** or **postfiltering**.
- **Area sampling or Prefiltering :** In this antialiasing method pixel intensity is determined by calculating the areas of overlap of each pixel with the objects to be displayed. Antialiasing by computing overlap area is referred to as area sampling or prefiltering.

Antialiasing methods are further classified as shown in the Fig. Q.20.1.

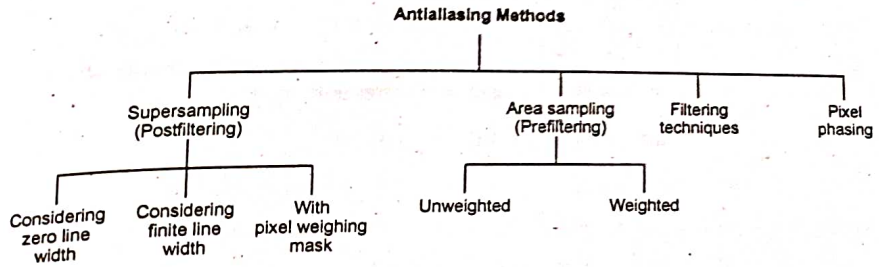


Fig. Q.20.1 Classification of antialiasing methods

Unweighted Area Sampling

- In this antialiasing, instead of picking closest pixel, both pixels are highlighted. However, their intensity values may differ.
- In unweighted area sampling, the intensity of pixel is proportional to the amount of line area occupied by the pixel.
- This technique produces noticeably better results than does setting pixels either to full intensity or to zero intensity.

Weighted Area Sampling

- In weighted area sampling equal areas contribute unequally i.e. a small area closer to the pixel center has greater intensity than does one at a greater distance. Thus, in weighted area sampling the intensity of the pixel is dependent on the line area occupied and the distance of area from the pixel's center.

3.4 : Circle Drawing Algorithms

Q.21 Explain vector generation/DDA algorithm for circle.

☞[SPPU : May-05, Marks 6]

Ans. : Algorithm

1. Read the radius (r), of the circle and calculate value of ε
2. start_x = 0

   start_y = r
3. x_1 = start_x

   y_1 = start_y
4. do

   { $x_2 = x_1 + \varepsilon y_1$

   $y_2 = y_1 - \varepsilon x_2$

   [x_2 represents x_{n+1} and x_1 represents x_n]

   Plot (int (x_2) , int (y_2))

   $x_1 = x_2$;

   $y_1 = y_2$;

   [Reinitialize the current point]

   } while (y_1 – start_y) < ε or (start_x – x_1) > ε

   [check if the current point is the starting point or not. If current point is not starting point repeat step 4 ; otherwise stop]
5. Stop.

Q.22 Derive the expression for decision parameter used in Bresenham's circle algorithm. Also explain the Bresenham's circle algorithm.

☞[SPPU : May-06,07,13,14,16, Dec.-07,12,13, Marks 10

Dec.-17, Marks 6]

Ans. : • Let us assume point P (x_i , y_i) in Fig. Q.22.1 as a last scan converted pixel. Now we have two options either to choose pixel A or pixel B. The closer pixel amongst these two can be determined as follows

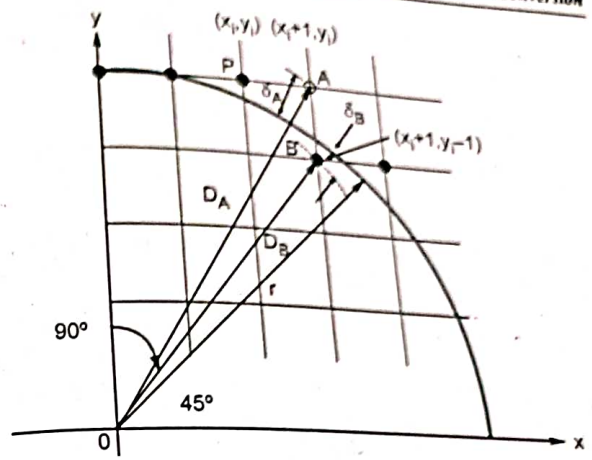


Fig. Q.22.1 Scan conversion with Bresenham's algorithm

• The distances of pixels A and B from the origin are given as

$$D_A = \sqrt{(x_{i+1})^2 + (y_i)^2} \quad \text{and}$$

$$D_B = \sqrt{(x_{i+1})^2 + (y_i - 1)^2}$$

Now, the distances of pixels A and B from the true circle are given as

$$\delta_A = D_A - r \text{ and } \delta_B = D_B - r$$

However, to avoid square root term in derivation of decision variable, i.e. to simplify the computation and to make algorithm more efficient the δ_A and δ_B are defined as

$$\delta_A = D_A^2 - r^2 \quad \text{and}$$

$$\delta_B = D_B^2 - r^2$$

That is

$$\delta_A = (x_i + 1)^2 + y_i^2 - r^2 \qquad \ldots (1)$$

$$\delta_B = (x_i + 1)^2 + (y_i - 1)^2 - r^2 \qquad \ldots (2)$$

This function D provides a relative measurement of the distance from the centre of a pixel to the true circle. Since D_A will always be positive

(A is outside of the true circle) and D_B will always be negative (B is inside of the true circle), a decision variable d_i can be defined as

$$d_i = \delta_A + \delta_B \quad \ldots (3)$$

From equations (1), (2) and (3) we have

$$d_i = (x_i + 1)^2 + y_i^2 - r^2 + (x_i + 1)^2 + (y_i - 1)^2 - r^2$$

$$= 2(x_i + 1)^2 + y_i^2 + (y_i - 1)^2 - 2r^2 \quad \ldots (4)$$

When $d_i < 0$, we have $|\delta_A| < |\delta_B|$ and pixel A is chosen. When $d_i \geq 0$, we have $|\delta_A| \geq |\delta_B|$ and pixel B is selected. In other words we can write,

For $d_i < 0$, $x_{i+1} = x_i + 1$ and

For $d_i \geq 0$, $x_{i+1} = x_i + 1$ and $y_{i+1} = y_i - 1$

Derivation of Decision Variable ($d_i + 1$)

$$d_{i+1} = 2(x_{i+1} + 1)^2 + y_{i+1}^2 + (y_{i+1} - 1)^2 - 2r^2$$

$$\therefore d_{i+1} - d_i = 2(x_{i+1} + 1)^2 + y_{i+1}^2 + (y_{i+1} - 1)^2 - 2(x_i + 1)^2 - y_i^2 - (y_i - 1)^2$$

Since $x_{i+1} = x_i + 1$ We have,

$$d_{i+1} = d_i + 2(x_i + 1 + 1)^2 + y_{i+1}^2 + (y_{i+1} - 1)^2 - 2(x_i + 1)^2 - y_i^2 - (y_i - 1)^2$$

$$= d_i + 2(x_i^2 + 4x_i + 4) + y_{i+1}^2 + y_{i+1}^2 - 2y_{i+1} + 1 - 2(x_i^2 + 2x_i + 1) - y_i^2 - (y_i^2 - 2y_i + 1)$$

$$= d_i + 2x_i^2 + 8x_i + 8 + 2y_{i+1}^2 - 2y_{i+1} + 1 - 2x_i^2 - 4x_i - 2 - y_i^2 - y_i^2 + 2y_i - 1$$

$$= d_i + 4x_i + 2(y_{i+1}^2 - y_i^2) - 2(y_{i+1} - y_i) + 6$$

If A is chosen pixel (meaning that the $d_i < 0$) then $y_{i+1} = y_i$ and so $d_{i+1} = d_i + 4x_i + 6$. On the other hand, if B is the chosen pixel (meaning that the $d_i \geq 0$) then $y_{i+1} = y_i - 1$ and so

$$d_{i+1} = d_i + 4x_i + 2[(y_i - 1)^2 - y_i^2] - 2[y_i - 1 - y_i] + 6$$

$$= d_i + 4x_i + 2[y_i^2 - 2y_i + 1 - y_i^2] - 2(-1) + 6$$

$$= d_i + 4x_i - 4y_i + 2 + 2 + 6 = d_i + 4(x_i - y_i) + 10$$

Finally, the equation for d_i at starting point, i.e. at $x = 0$ and $y = r$ can be simplified as follows

$$d_i = \delta_A + \delta_B$$

$$= (x_i + 1)^2 + (y_i)^2 - r^2 + (x_i + 1)^2 + (y_i - 1)^2 - r^2$$

$$= (0 + 1)^2 + (r)^2 - r^2 + (0 + 1)^2 + (r - 1)^2 - r^2$$

$$= 1 + r^2 - r^2 + 1 + r^2 - 2r + 1 - r^2 = 3 - 2r$$

Q.23 Write and explain Bresenham's circle drawing algorithm with related mathematics. ☞ **[SPPU : Dec.-17, May-18, Marks 6]**

Ans. : Bresenham's circle drawing algorithm to plot 1/8 of the circle :

1. Read the radius (r) of the circle.
2. $d = 3 - 2r$

   [Initialize the decision variable]
3. $x = 0, y = r$

   [Initialize starting point]
4. do

```
{
    plot (x, y)
    if (d < 0) then
    {
        d = d + 4x + 6
    }
    else
    { d = d + 4 (x - y) + 10
      y = y - 1
    }
    x = x + 1
} while (x < y)
```

5. Stop

Q.24 Explain the midpoint circle generating algorithm.

Ans. :

- Fig. Q.24.1 shows the two possible y positions (y_i and $y_i + 1$) at sampling position $x_i + 1$. Therefore, we have to determine whether the pixel at position $(x_i + 1, y_i)$ or at position $(x_i + 1, y_i - 1)$ is closer to the circle. For this purpose decision parameter is used. It uses the circle function ($f_{circle}(x, y) = x^2 + y^2 - r^2$) evaluated at the midpoint between these two pixels.

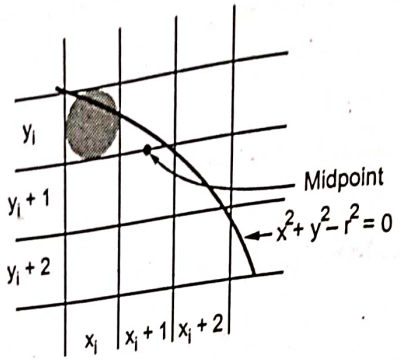


Fig. Q.24.1 Decision parameter to select correct pixel in circle generation algorithm

$$d_i = f_{circle}(x_i + 1, y_i - \frac{1}{2}) = (x_i + 1)^2 + \left(y_i - \frac{1}{2}\right)^2 - r^2$$

$$= (x_i + 1)^2 + y_i^2 - y_i + \frac{1}{4} - r^2 \quad \ldots (1)$$

- If $d_i < 0$, this midpoint is inside the circle and the pixel on the scan line y_i is closer to the circle boundary.
- If $d_i \geq 0$, the midposition is outside or on the circle boundary, and $y_i - 1$ is closer to the circle boundary.
- The incremental calculation can be determined to obtain the successive decision parameters.
- We can obtain a recursive expression for the next decision parameter by evaluating the circle function at sampling position $x_{i+1} + 1 = x_i + 2$.

$$d_{i+1} = f_{circle}(x_{i+1} + 1, y_{i+1} - \frac{1}{2})$$

$$= [(x_i + 1) + 1]^2 + \left(y_{i+1} - \frac{1}{2}\right)^2 - r^2$$

$$= (x_i + 1)^2 + 2(x_i + 1) + 1 + y_{i+1}^2 - (y_{i+1}) + \frac{1}{4} - r^2 \quad \ldots (2)$$

Looking at equations (1) and (2) we can write

$$d_{i+1} = d_i + 2(x_i + 1) + (y_{i+1}^2 - y_i^2) - (y_{i+1} - y_i) + 1$$

where y_{i+1} is either y_i or y_{i-1}, depending on the sign of d_i.

If d_i is negative, $y_{i+1} = y_i$

$$\therefore \quad d_{i+1} = d_i + 2(x_i + 1) + 1$$

$$= d_i + 2x_{i+1} + 1 \quad \ldots (3)$$

If d_i is positive, $y_{i+1} = y_{i-1}$

$$\therefore \quad d_{i+1} = d_i + 2(x_i + 1) + 1 - 2y_{i+1} \quad \ldots (4)$$

The terms $2x_{i+1}$ and $-2y_{i+1}$ in equations (3) and (4) can be incrementally calculated as

$$2x_{i+1} = 2x_i + 2$$

$$2y_{i+1} = 2y_i - 2$$

The initial value of decision parameter can be obtained by evaluating circle function at the start position $(x_0, y_0) = (0, r)$.

$$d_0 = f_{circle}\left((0 + 1)^2 + \left(r - \frac{1}{2}\right)^2 - r^2\right) = 1 + \left(r - \frac{1}{2}\right)^2 - r^2 = 1.25 - r$$

Algorithm

1. Read the radius (r) of the circle
2. Initialize starting position as
   $x = 0$
   $y = r$
3. Calculate initial value of decision parameter as
   $P = 1.25 - r$

4. do

```
{ plot (x, y)
  if (d < 0)
  { x = x + 1
    y = y
    d = d + 2x + 1
  }
  else
{ x = x + 1
  y = y – 1
  d = d + 2x + 2y + 1
}
while (x < y)
```

5. Determine symmetry points
6. Stop.

Q.25 Calculate the pixel position along the circle path with radius r = 10 centered on the origin using Bresenham's circle drawing algorithm from point (0,10) to point x = y.

Ans. : The value of d is given as,

$$d = 3 - 2r = 3 - 2 * 10 = -17$$

Tabulating the results of step 4 we get,

x	y	d
0	10	– 17
1	10	– 11
2	10	– 1
3	10	13
4	9	– 5
5	9	17
6	8	11
7	7	13

Q.26 Calculate the pixel position along the circle path with radius r = 10 centered on the origin using midpoint circle drawing algorithm from point (0,10) to point x = y.

Ans. : Initial value of decision parameter

$$d = 1.25 - r = -8.75$$

i	d	x	y
1	– 8.75	0	10
2	– 5.75	1	10
3	– 0.75	2	10
4	6.25	3	10
5	– 2.75	4	9
6	8.25	5	9
7	5.25	6	8
8	6.25	7	7

END...

Unit II

4 Polygons and Polygon Filling

4.1 : Introduction to Polygon

Q.1 What is polygon ? ☞[SPPU : Dec.-06, May-12, Marks 2]

Ans. : When starting point and terminal point of any polyline is same, i.e. when polyline is closed then it is called **polygon.**

Q.2 Is circle a polygon ? Justify. ☞[SPPU : May-11, Marks 4]

Ans. : Circle is not a polygon. Because circle don't have any line segments.

Q.3 What are different types of polygons ? Explain with example.

☞[SPPU : Dec.-09,18, May-10, 11, 12, Marks 4]

Ans. : There are two types of polygons :

- Convex and
- Concave

- A convex polygon is a polygon in which the line segment joining any two points within the polygon lies completely inside the polygon. Fig. Q.3.1 shows the examples of convex polygons.

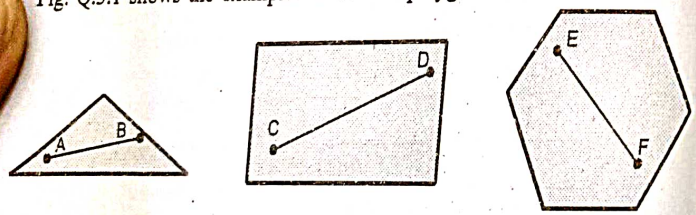


Fig. Q.3.1 Convex polygons

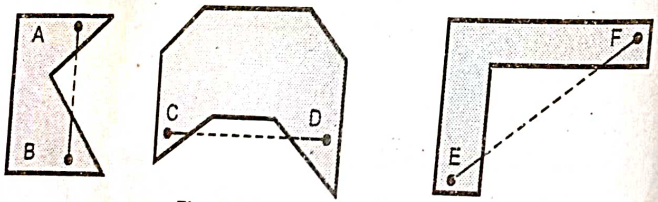


Fig. Q.3.2 Concave polygons

- A concave polygon is a polygon in which the line segment joining any two points within the polygon may not lie completely inside the polygon. Fig. Q.3.2 shows the examples of concave polygons.
- A polygon is simple if it is described by a single, non-intersecting boundary; otherwise it is said to be **complex.**

Fig. Q.3.3 Complex polygon

4.2 : Inside Test

Q.4 Explain the different methods for testing a pixel inside of polygon. ☞[SPPU : May-05,07,10,13,16,19, Dec.-09,10,12,13,14,15,16,17,18, Marks 6]

Ans. : Even-Odd Method :

- One simple method of doing this is to construct a line segment between the point in question and a point known to be outside the polygon, as shown in the Fig. Q.4.1. Now count how many intersections of the line segment with the polygon boundary occur. If there are an odd number of intersections, then the point in question is inside; otherwise it is outside.

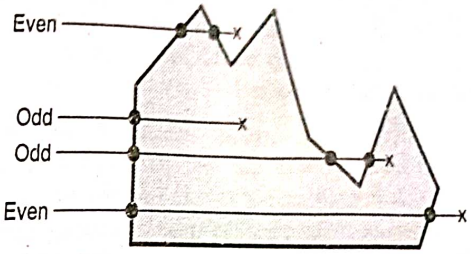


Fig. Q.4.1

- If the intersection point is vertex of the polygon then we have to look at the other endpoints of the two segments which meet at this vertex. If these points lie on the same side of the constructed line, then the point in question counts as an even number of intersections. If they lie on opposite sides of the constructed line, then the point is counted as a single intersection. This is illustrated in Fig. Q.4.2.

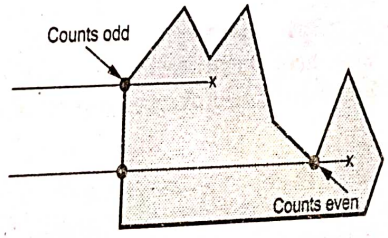


Fig. Q.4.2

Winding-Number Method :

- In winding number method we have to picturise a line segment running from outside the polygon to the point in question and consider the polygon sides which it crosses.
- As shown in the Fig. Q.4.3, point (x_a, y_a) is a test point and line $y = y_a$ is a line runs from outside the polygon to point (x_a, y_a). It is important to note that the line we select must not pass through any vertices.

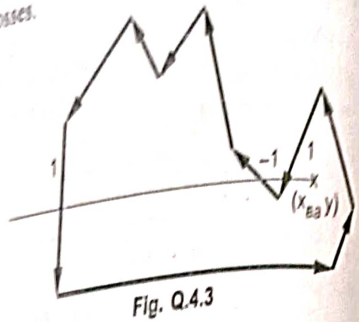


Fig. Q.4.3

The polygon edges crossed by this line could be drawn in two ways. The edge could be drawn starting below the line, cross it, and end above the line or starting above the line, cross it, and end below the line. In first case we have to give direction number as –1 and in the second case we have give direction number as 1.

- The sum of the direction numbers for the sides that cross the constructed horizontal line segment is called the **winding number** for the point in question. For polygons or two dimensional objects, the point is said to be inside when the value of winding number is nonzero.

4.3 : Polygon Filling

Q.5 Enlist any three polygon filling algorithms.

☞[SPPU : May-11,13, Dec.-12,13, Marks 4]

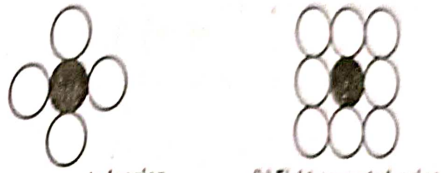
Ans. : Polygon filling algorithms are classified as : Seed fill algorithm and scan line algorithm.

- The seed fill algorithm is further classified as flood fill algorithm and boundary fill algorithm.
- Algorithms that fill interior-defined regions are called **flood-fill algorithms**; those that fill boundary-defined regions are called **boundary-fill algorithms** or **edge-fill algorithms.**

Q.6 Explain boundary-fill/edge-fill algorithm for polygon.

☞[SPPU : May-05,09,18, Dec.-16,19, Marks 6]

Ans. : • In this method, edges of the polygons are drawn. Then starting with some seed, any point inside the polygon we examine the neighbouring pixels to check whether the boundary pixel is reached. If boundary pixels are not reached, pixels are highlighted and the process is continued until boundary pixels are reached.



(a) Four connected region (b) Eight connected region

Fig. Q.6.1

- Boundary defined regions may be either 4-connected or 8-connected as shown in the Fig. Q.6.1.
- If a region is 4-connected, then every pixel in the region may be reached by a combination of moves in only four directions : left, right, up and down.
- For an 8-connected region every pixel in the region may be reached by a combination of moves in the two horizontal, two vertical and four diagonal directions.

Q.7 Write algorithm to fill polygon area using flood fill method.

☞[SPPU : May-15, Dec.-15, Marks 4]

Ans. : • Sometimes it is required to fill in an area that is not defined within a single colour boundary. In such cases we can fill areas by replacing a specified interior colour instead of searching for a boundary colour. This approach is called a flood-fill algorithm.

- Like boundary fill algorithm, here we start with some seed and examine the neighbouring pixels. However, here pixels are checked for a specified interior colour instead of boundary colour and they are replaced by new colour.
- Using either a 4-connected or 8-connected approach, we can step through pixel positions until all interior point have been filled. The

following procedure illustrates the recursive method for filling 8-connected region using flood-fill algorithm.

Q.8 Explain scan line algorithm with example.

☞[SPPU : Dec.-05,06,08, May-06,07,08,11,12,14,17,19, Marks 6]

OR What is meant by coherence and how it can increase the efficiency of scan line polygon filling.

☞[SPPU : May-15,18, Marks 4]

Ans. : • Fig. Q.8.1 illustrates the scan line algorithm for filling of polygon.

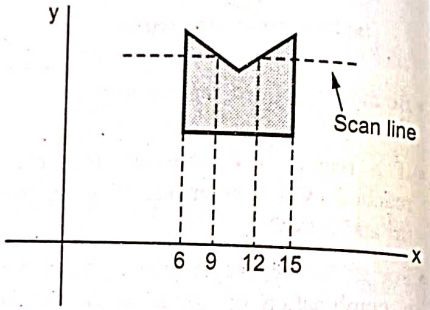


Fig. Q.8.1

- For each scan line crossing a polygon, this algorithm locates the intersection points of the scan line with the polygon edges. These intersection points are then sorted from left to right and the corresponding positions between each intersection pair are set to the specified fill colour.
- In Fig. Q.8.1, we can see that there are two stretches of interior pixels from x = 6 to x = 9 and x = 12 to x = 15.
- The scan line algorithm first finds the largest and smallest y values of the polygon. It then starts with the largest y value and works its way down, scanning from left to right, in the manner of a raster display.
- The important task in the scan line algorithm is to find the intersection points of the scan line with the polygon boundary.
- When intersection points are even, they are sorted from left to right, paired and pixels between paired points are set to the fill colour.
- In some cases intersection point is a vertex. When scan line intersects polygon vertex a special handling is required to find the exact intersection points. To handle such cases, we must look at the other endpoints of the two line segments of the polygon which meet at this

vertex. If these points lie on the same (up or down) side of the scan line, then the point in question counts as an even number of intersections. If they lie on opposite sides of the scan line, then the point is counted as single intersection. This is illustrated in Fig. Q.8.2.

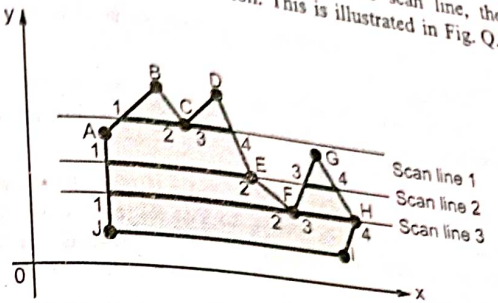


Fig. Q.8.2 Intersection points along the scan line that intersect polygon vertices

- It is necessary to calculate x intersection points for scan line with every polygon side.
- We can simplify these calculations by using **coherence properties.**
- A coherence property of a scene is a property of a scene by which we can relate one part of a scene with the other parts of a scene. Here, we can use a slope of an edge as a coherence property. By using this property we can determine the x intersection value on the lower scan line if the x intersection value for current scan line is known. This is given as

$$x_{i+1} = x_i - \frac{1}{m}$$

where m is the slope of the edge

- As we scan from top to bottom value of y coordinates between the two scan line changes by 1.

$$y_{i+1} = y_i - 1$$

- Many times it is not necessary to compute the x intersections for scan line with every polygon side.
- Thus we can say that coherence properties can increase the efficiency of scan line polygon filling algorithm.

Q.9 What are the steps involved in filling polygon in scan line method ? [SPPU : Dec.-05, 06, 08, May-06, 07, 08, Marks 8]

Ans. :

1. Read n, the number of vertices of polygon
2. Read x and y coordinates of all vertices in array x[n] and y[n].
3. Find y_{min} and y_{max}.
4. Store the initial x value (x_1) y values y_1 and y_2 for two endpoints and x increment Δx from scan line to scan line for each edge in the array edges [n] [4].

   While doing this check that $y_1 > y_2$, if not interchange y_1 and y_2 and corresponding x_1 and x_2 so that for each edge, y_1 represents its maximum y coordinate and y_2 represents its minimum y coordinate.
5. Sort the rows of array, edges [n] [4] in descending order of y_1, descending order of y_2 and ascending order of x_2.
6. Set $y = y_{max}$
7. Find the active edges and update active edge list :

   if ($y > y_2$ and $y \leq y_1$)

   { edge is active }

   else

   { edge is not active }
8. Compute the x intersects for all active edges for current y value [initially x-intersect is x_1 and x intersects for successive y values can be given as

   $x_{i+1} \leftarrow x_i + \Delta x$

   where $\Delta x = -\frac{1}{m}$ and $m = \frac{y_2 - y_1}{x_2 - x_1}$ i.e. slope of a line segment
9. If x intersect is vertex i.e. x-intersect = x_1 and $y = y_1$ then apply vertex test to check whether to consider one intersect or two intersects. Store all x intersects in the x-intersect [ ] array.
10. Sort x-intersect [ ] array in the ascending order,
11. Extract pairs of intersects from the sorted x-intersect [ ] array.
12. Pass pairs of x values to line drawing routine to draw corresponding line segments

13. Set $y = y - 1$
14. Repeat steps 7 through 13 until $y \geq y_{min}$.
15. Stop

In step 7, we have checked for $y \leq y_1$ and not simply $y < y_1$. Hence step 9 a becomes redundant. Following program takes care of that.

END...

Unit II

5 Windowing and Clipping

5.1 : Introduction

Q.1 What is windowing and clipping ?

☞[SPPU : May-13, Dec.-05,12,13, Marks 4]

Ans. : The process of selecting and viewing the picture with different views is called **windowing**, and a process which divides each element of the picture into its visible and invisible portions, allowing the invisible portion to be discarded is called **clipping**.

5.2 : Viewing Transformations

Q.2 Describe viewing transformation.

☞[SPPU : Dec.-10, 11, 19, May-12, Marks 8]

Ans. :

- The picture is stored in the computer memory using any convenient Cartesian co-ordinate system, referred to as **World Co-ordinate System** (WCS).
- When picture is displayed on the display device it is measured in **Physical Device Co-ordinate System** (PDCS) corresponding to the display device. Therefore, displaying an image of a picture involves mapping the co-ordinates of the points and lines that form the picture into the appropriate physical device co-ordinate where the image is to be displayed. This mapping of co-ordinates is achieved with the use of co-ordinate transformation known as **viewing transformation**.
- Sometimes the two dimensional viewing transformation is simply referred to as the **window to view port transformation** or the **windowing transformation**.
- The viewing transformation which maps picture co-ordinates in the WCS to display co-ordinates in PDCS is performed by the following transformations.

- Converting world co-ordinates to viewing co-ordinates.

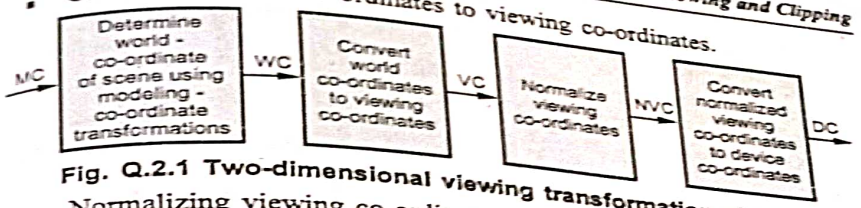


Fig. Q.2.1 Two-dimensional viewing transformation pipeline

- Normalizing viewing co-ordinates.
- Converting normalized viewing co-ordinates to device co-ordinates.

- World Co-ordinate System (WCS) is infinite in extent and the device display area is finite.
- To perform a viewing transformation we select a finite world co-ordinate area for display called a **window**.
- An area on a device to which a window is mapped is called a **viewport**.

5.3 : 2D Clipping

Q.3 What is line clipping ?

☞[SPPU : May-12, Marks 2]

Ans. :

- The lines are said to be interior to the clipping window and hence visible if both end points are interior to the window.
- If both end points of a line are exterior to the window, the line is not necessarily completely exterior to the window.
- If both end points of a line are completely to the right of, completely to the left of, completely above, or completely below the window, then the line is completely exterior to the window and hence invisible.
- The lines which across one or more clipping boundaries require calculation of multiple intersection points to decide the visible portion of them.
- Thus deciding visible portion of the line is known as line clipping.

5.4 : Cohen-Sutherland Line Clipping Algorithm

Q.4 Explain Cohen-Sutherland algorithm with the help of suitable example.

☞[SPPU : May-06,07,08,10,11,12,13,15,16, Dec.-10,13,15,16,18, Marks 8; May-19, Marks 4]

Ans. :

1. Read two end points of the line say P_1 (x_1, y_1) and P_2 (x_2, y_2).
2. Read two corners (left-top and right-bottom) of the window, say $(Wx_1, Wy_1$ and $Wx_2, Wy_2)$.
3. Assign the region codes for two endpoints P_1 and P_2 using following steps :

   Initialize code with bits 0000
   - Set Bit 1 - if $(x < Wx_1)$
   - Set Bit 2 - if $(x > Wx_2)$
   - Set Bit 3 - if $(y < Wy_2)$
   - Set Bit 4 - if $(y > Wy_1)$

4. Check for visibility of line P_1 P_2
   a) If region codes for both endpoints P_1 and P_2 are zero then the line is completely visible. Hence draw the line and go to step 9.
   b) If region codes for endpoints are not zero and the logical ANDing of them is also non-zero then the line is completely invisible, so reject the line and go to step 9.
   c) If region codes for two endpoints do not satisfy the conditions in 4a) and 4b)the line is partially visible.
5. Determine the intersecting edge of the clipping window by inspecting the region codes of two endpoints.
   a) If region codes for both the end points are non-zero, find intersection points P_1' and P_2' with boundary edges of clipping window with respect to point P_1 and point P_2, respectively
   b) If region code for any one end point is non-zero then find intersection point P_1' or P_2' with the boundary edge of the clipping window with respect to it.
6. Divide the line segments considering intersection points.
7. Reject the line segment if any one end point of it appears outsides the clipping window.
8. Draw the remaining line segments.
9. Stop.

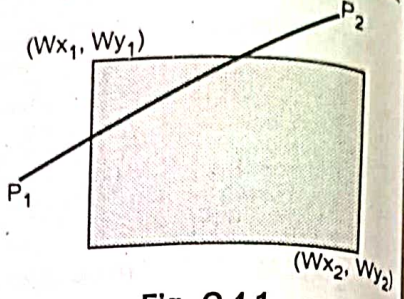


Fig. Q.4.1

Q.5 Use the Cohen-Sutherland outcode algorithm to clip two lines P_1 (40, 15) – P_2 (75, 45) and P_3 (70, 20) – P_4 (100, 10) against a window A (50, 10), B (80, 10), C (80, 40), D(50,40).

☞[Dec.-12, Marks 8]

Solution : Line 1 : P_1 (40, 15) P_2 (75, 45) $x_L = 50$ $y_B = 10$ $x_R = 80$ $y_T = 40$

Point	Endcode	ANDing	Position
P_1	0 0 0 1	0 0 0 0	Partially visible
P_2	0 0 0 0		

$$x_L, y = m(x_L - x) + y_1 = \frac{6}{7}(50 - 40) + 15 \quad m = \frac{45-15}{75-40} = \frac{6}{7}$$

$$= \mathbf{23.57}$$

$$y_T, x = x_1 + \left(\frac{1}{m}\right)(y_T - y_1) = 40 + \left(\frac{7}{6}\right)(40 - 15)$$

$$= \mathbf{69.16}$$

$I_1 = (50, 23.57)$ $I_2 = (69.06, 40)$

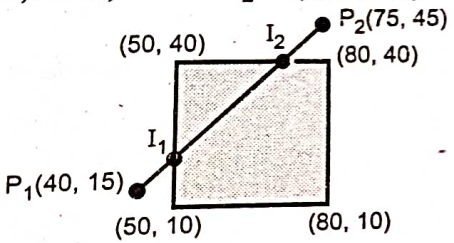


Fig. Q.5.1

Line 2 : P_3 (70, 20) P_4 (100, 10)

Point	End code	ANDing	Position
P_3	0 0 0 0	0 0 0 0	Partially visible
P_4	0 0 1 0		

$$\text{Slope } m' = \frac{10-20}{100-70} = \frac{-10}{30} = \frac{-1}{3}$$

$$x_R, y = m(x_R - x_1) + y_1 = \frac{-1}{3}(80 - 70) + 20 = \mathbf{16.66}$$

$$I = (80, 16.66)$$

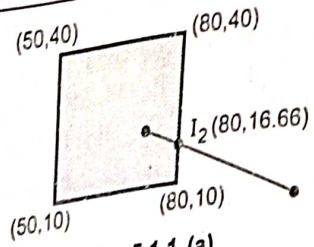
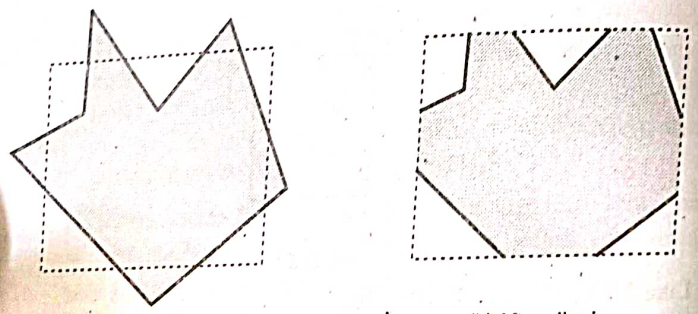


Fig. 5.1.1 (a)

5.5 : Polygon Clipping

Q.6 Can line clipping algorithm be used for polygon clipping ? Justify. **☞[SPPU : May-10, Marks 8]**

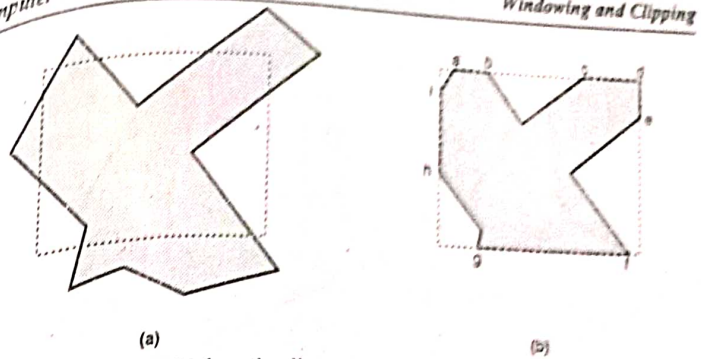
Ans. : • A polygon is nothing but the collection of lines. Therefore, we might think that line clipping algorithm can be used directly for polygon clipping. However, when a closed polygon is clipped as a collection of lines with line clipping algorithm, the original closed polygon becomes one or more open polygon or discrete lines as shown in the Fig. Q.6.1. Thus, we need to modify the line clipping algorithm to clip polygons.



(a) Before clipping (b) After clipping

Fig. Q.6.1 Polygon clipping done by line clipping algorithm

- We consider a polygon as a closed solid area. Hence after clipping it should remain closed. To achieve this we require an algorithm that will generate additional line segment which make the polygon as a closed area.
- For example, in Fig. Q.6.2 the lines a - b, c - d, d - e, f - g, and h - i are added to polygon description to make it closed.



(a) (b)

Fig. Q.6.2 Modifying the line clipping algorithm for polygon

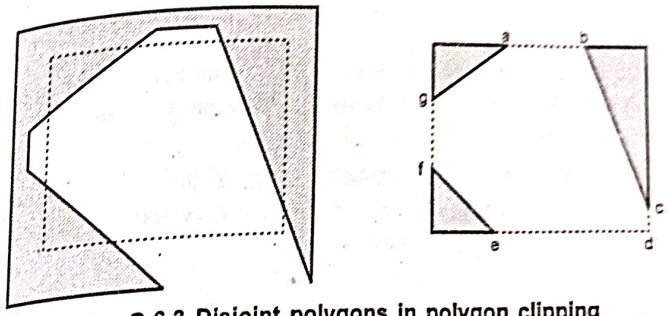


Fig. Q.6.3 Disjoint polygons in polygon clipping

- Adding lines c - d and d - e is particularly difficult. Considerable difficulty also occurs when clipping a polygon results in several disjoint smaller polygons as shown in the Fig. Q.6.3. For example, the lines a - b, c - d, d - e and g - f are frequently included in the clipped polygon description which is not desired.

Q.7 Describe Sutherland - Hodgeman polygon clipping algorithm with example. **☞[SPPU : Dec.-06,07,08,11,14,17,18, Marks 6, May-10,14, Marks 8]**

Ans. :

1. Read co-ordinates of all vertices of the polygon.
2. Read co-ordinates of the clipping window.
3. Consider the left edge of the window.
4. Compare the vertices of each edge of the polygon, individually with the clipping plane.

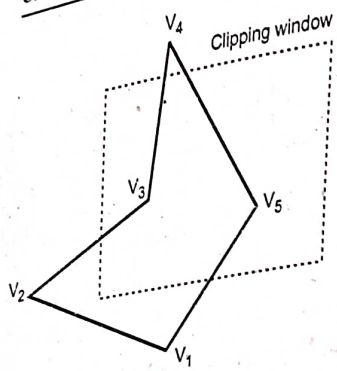


Fig. Q.7.1

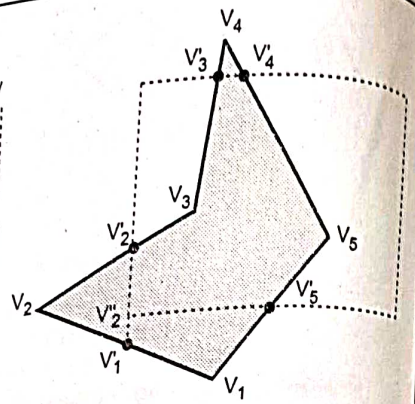


Fig. Q.7.1 (a)

5. Save the resulting intersections and vertices in the new list of vertices according to four possible relationships between the edge and the clipping boundary.
6. Repeat the steps 4 and 5 for remaining edges of the clipping window. Each time the resultant list of vertices is successively passed to process the next edge of the clipping window.
7. Stop.

Let us consider, a polygon and clipping window shown in Fig. Q.7.1

Original polygon vertices are V_1, V_2, V_3, V_4, V_5. After clipping each boundary the new vertices are given in Fig. Q.7.1 (a).

After left clipping : V_1', V_2', V_3, V_4, V_5, V_1
After right clipping : V_1', V_2', V_3, V_4, V_5, V_1
After top clipping : V_1', V_2', V_3, V_3', V_4', V_5, V_1
After bottom clipping : V_2'', V_2', V_3, V_3', V_4', V_5, V_5'

5.6 : Weiler Atherton Polygon Clipping Algorithm

Q.8 Explain Weiler Atherton Polygon Clipping Algorithm.

[SPPU : Dec.-17, Marks 2]

Ans. : The clipping algorithms previously discussed require a convex polygon. In context of many applications, e.g. hidden surface removal, the ability to clip to concave polygon is required. A powerful but somewhat more complex clipping algorithm developed by Weiler and Atherton

meets this requirement. This algorithm defines the polygon to be clipped as a **subject polygon** and the clipping region is the clip polygon.

The algorithm describes both the subject and the clip polygon by a circular list of vertices. The boundaries of the subject polygon and the clip polygon may or may not intersect. If they intersect, then the intersections occur in pairs. One of the intersections occurs when a subject polygon edge enters the inside of the clip polygon and one when it leaves. As shown in the Fig. Q.8.1, there are four intersection vertices I_1, I_2 , I_3 and I_4. In these intersections I_1 and I_3 are entering intersections, and I_2 and I_3 are leaving intersections. The clip polygon vertices are marked as C_1, C_2, C_3 and C_4.

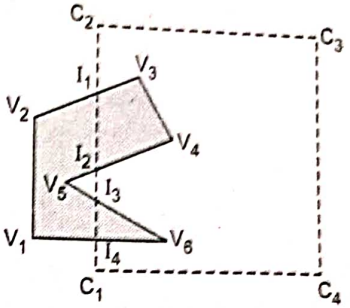


Fig. Q.8.1

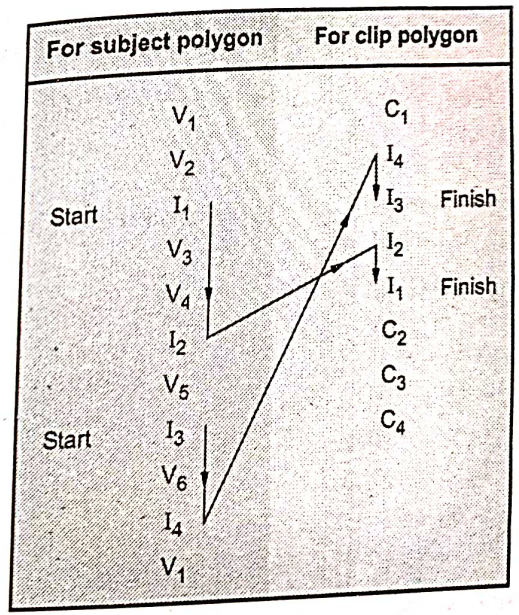


Table Q.8.1 List of polygon vertices

In this algorithm two separate vertices lists are made one for clip polygon and one for subject polygon including intersection points. The Table Q.8.1 shows these two lists for polygons shown in Fig. Q.8.2.

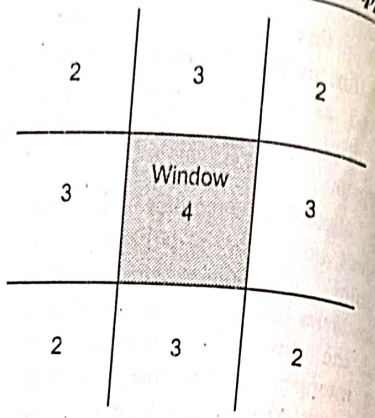


Fig. Q.8.2 Nine regions of clipping plane

The algorithm starts at an entering intersection (I_1) and follows the subject polygon vertex list in the downward direction (i.e. I_1, V_3, V_4, I_2). At the occurrence of leaving intersection the algorithm follows the clip polygon vertex list from the leaving intersection vertex in the downward direction (i.e. I_2, I_1). At the occurrence of the entering intersection the algorithm follows the subject polygon vertex list from the entering intersection vertex. This process is repeated until we get the starting vertex. This process we have to repeat for all remaining entering intersections which are not included in the previous traversing of vertex list. In our example, entering vertex I_3 was not included in the first traversing of vertex list,. Therefore, we have to go for another vertex traversal from vertex I_3.

above two vertex traversals gives two clipped inside polygons. There

$_3$, V_4, I_2, I_1 and I_3, V_6, I_4, I_3

END...

Unit - III

6 2D Transformations

6.1 : Introduction

Q.1 Explain image transformation with example.

[SPPU : May-15, Marks 3]

Ans. : • Almost all graphics systems allow the programmer to define picture that include a variety of transformations.

• For example, the programmer is able to magnify a picture so that detail appears more clearly, or reduce it so that more of the picture is visible. The programmer is also able to rotate the picture so that he can see it in different angles.

6.2 : Two - dimensional Transformations

Q.2 Scale the polygon with co-ordinates A (4,5), B (8,10) and C (8,2) by 2 units in x-direction and 3 units in y-direction. Find the transformed A, B and C points. **[SPPU : Dec.-14, Marks 6]**

Ans. :

$$\begin{bmatrix} A' \\ B' \\ C' \end{bmatrix} = \begin{bmatrix} 4 & 5 & 1 \\ 8 & 10 & 1 \\ 8 & 2 & 1 \end{bmatrix} \begin{bmatrix} 2 & 0 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 8 & 15 & 1 \\ 16 & 30 & 1 \\ 16 & 6 & 1 \end{bmatrix}$$

The transformed points are $A' = (8, 15)$, $B' = (16, 30)$ and $C' = (16, 6)$.

Q.3 Write 2D transformation matrices of translation, scaling. Give the derivation of 2D rotation matrix.

[SPPU : May-14, Dec.-14, Marks 4]

OR Describe w.r.t. 2D transformation :
i) Scaling ii) Rotation iii) Translation

[SPPU : Dec.-07,09,18, May-12,15, Marks 8]

Ans. : Translation :

- We can translate a two dimensional point by adding translation distances, t_x and t_y, to the original co-ordinate position (x, y) to move the point to a new position (x′, y′), as shown in the Fig. Q.3.1

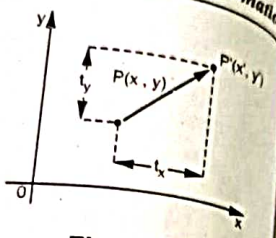


Fig. Q.3.1

$$x' = x + t_x \quad \ldots(1)$$

$$y' = y + t_y \quad \ldots(2)$$

- The translation distance pair (t_x, t_y) is called a **translation vector** or **shift vector.**
- It is possible to express the translation equations (1) and (2) as a single matrix equation by using column vectors to represent co-ordinate positions and the translation vector :

$$P = \begin{bmatrix} x \\ y \end{bmatrix} \quad P' = \begin{bmatrix} x' \\ y' \end{bmatrix} \quad T = \begin{bmatrix} t_x \\ t_y \end{bmatrix}$$

This allows us to write the two dimensional translation equations in the matrix form :

$$P' = P + T \quad \ldots(3)$$

Rotation

- Let us consider the rotation of the object about the origin, as shown in the Fig. Q.3.2.
- Here, r is the constant distance of the point from the origin, angle ϕ is the original angular position of the point from the horizontal and θ is the rotation angle.

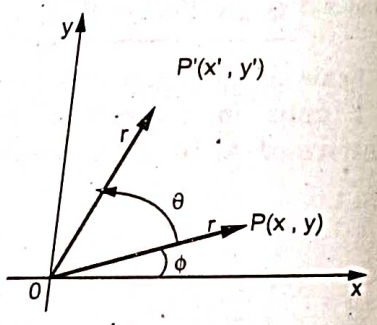


Fig. Q.3.2

- Using standard trigonometric equations, we can express the transformed co-ordinates in terms of angles θ and ϕ as,

$$\left.\begin{aligned} x' &= r\cos(\phi+\theta) = r\cos\phi\cos\theta - r\sin\phi\sin\theta \\ y' &= r\sin(\phi+\theta) = r\cos\phi\sin\theta + r\sin\phi\cos\theta \end{aligned}\right\} \quad \ldots(4)$$

The original co-ordinates of the point in polar co-ordinates are given as,

$$\left.\begin{aligned} x &= r\cos\phi \\ y &= r\sin\phi \end{aligned}\right\} \quad \ldots(5)$$

- Substituting equations (5) into (4), we get the transformation equations for rotating a point (x, y) through an angle θ about the origin as :

$$\left.\begin{aligned} x' &= x\cos\theta - y\sin\theta \\ y' &= x\sin\theta + y\cos\theta \end{aligned}\right\} \quad \ldots(6)$$

- The above equations can be represented in the matrix form as given below

$$[x' \;\; y'] = [x \;\; y]\begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix}$$

$$\therefore \quad P' = P \cdot R \quad \ldots(7)$$

where R is rotation matrix and it is given as

$$R = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \quad \ldots(8)$$

- It is important to note that positive values for the rotation angle define counterclockwise rotations about the rotation point and negative values rotate objects in the clockwise sense.
- For negative values of θ i.e. for clockwise rotation, the rotation matrix becomes

$$R = \begin{bmatrix} \cos(-\theta) & \sin(-\theta) \\ -\sin(-\theta) & \cos(-\theta) \end{bmatrix}$$

$$= \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \quad \because \cos(-\theta) = \cos\theta \text{ and } \sin(-\theta) = -\sin\theta \quad \ldots(9)$$

Scaling

- This operation can be carried out for polygons by multiplying the co-ordinate values (x, y) of each vertex by scaling factors S_x and S_y to produce the transformed co-ordinates (x′, y′).

$$x' = x \cdot S_x \quad \ldots(10)$$

and $$y' = y \cdot S_y$$

- Scaling factor S_x scales object in the x direction and scaling factor S_y scales object in the y direction. The equation (10) can be written in the matrix form as given below :

$$[x' \;\; y'] = [x \;\; y]\begin{bmatrix} S_x & 0 \\ 0 & S_y \end{bmatrix}$$

$$= [x \cdot S_x \quad y \cdot S_y] \qquad \ldots (11)$$

$$= P \cdot S$$

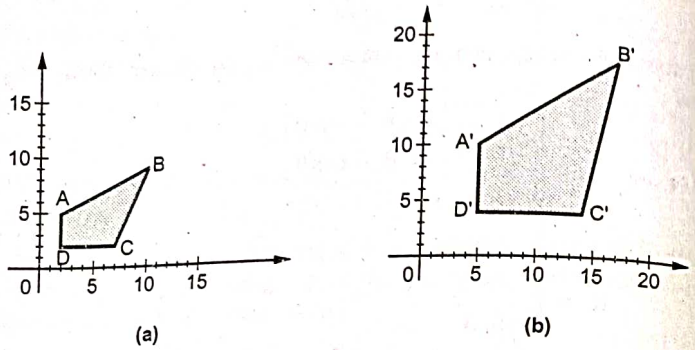


Fig. Q.3.3

6.3 : Homogeneous Co-ordinates

Q.4 What is homogeneous co-ordinate system ? Explain need of homogeneous co-ordinates. Compare homogeneous and normalized co-ordinates. ☞**[SPPU : Dec.-05,16,18, May-07,12, Marks 8]**

OR What is the need of homogenous co-ordinates ? Give the homogenous co-ordinates for translation, rotation and scaling.

☞**[SPPU : Dec.-10,12, Marks 10]**

Ans. :. • In order to combine sequence of transformations we have to eliminate the matrix addition associated with the translation terms in M_2. To achieve this we have to represent matrix M_1 as 3×3 matrix instead of 2×2 introducing an additional dummy co-ordinate W. Here, points are specified by three numbers instead of two. This co-ordinate system is called **homogeneous co-ordinate system** and it allows us to express all transformation equations as matrix multiplication.

• The homogeneous co-ordinate is represented by a triplet (X_W, Y_W, W), where

$$x = \frac{X_W}{W} \quad \text{and} \quad y = \frac{Y_W}{W}$$

• For two dimensional transformations, we can have the homogeneous parameter W to be any non zero value. But it is convenient to have $W = 1$. Therefore, each two dimensional position can be represented with homogeneous co-ordinate as $(x, y, 1)$.

• Summarizing it all up, we can say that the homogeneous co-ordinates allow combined transformation, eliminating the calculation of intermediate co-ordinate values and thus save required time for transformation and memory required to store the intermediate co-ordinate values.

• The homogeneous co-ordinates for translation are given as,

$$T = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ t_x & t_y & 1 \end{bmatrix} \qquad \ldots (1)$$

• The homogeneous co-ordinates for rotation are given as,

$$R = \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad \ldots (2)$$

• The homogeneous co-ordinate for scaling are given as,

$$S = \begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad \ldots (3)$$

6.4 : Composite Transformation

Q.5 Explain the concept of 2D rotation about an arbitrary point with matrix representation.

☞**[SPPU : Dec.-13,15,16,17,18, May-15,16, June-22, Marks 6]**

Ans. : To rotate an object about an arbitrary point, (x_p, y_p) we have to carry out three steps :

1. Translate point (x_p, y_p) to the origin
2. Rotate it about the origin and
3. Finally, translate the center of rotation back where it belongs Fig. Q.5.1.

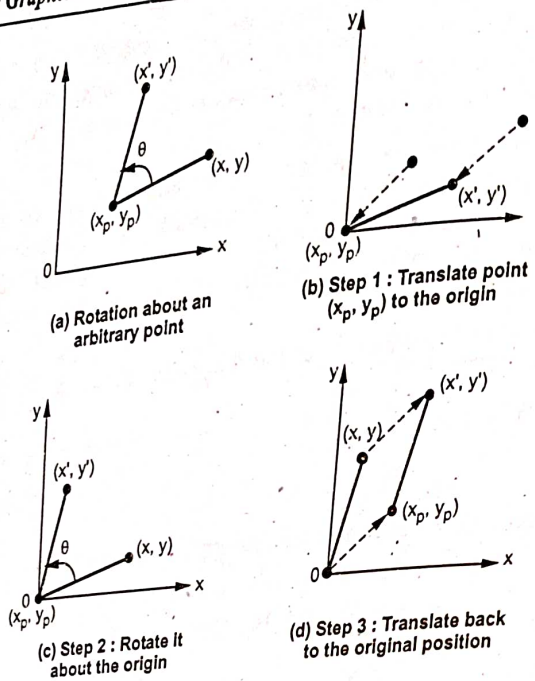


Fig. Q.5.1

- Matrix multiplication is not commutative, i.e. multiplying matrix A by matrix B will not always yield the same result as multiplying matrix B by matrix A. Therefore, in obtaining composite transformation matrix, we must be careful to order the matrices so that they correspond to the order of the transformations on the object.
- The translation matrix to move point (x_p, y_p) to the origin is given as,

$$T_1 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -x_p & -y_p & 1 \end{bmatrix}$$

- The rotation matrix for counterclockwise rotation of point about the origin is given as,

$$R = \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

- The translation matrix to move the center point back to its original position is given as,

$$T_2 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ x_p & y_p & 1 \end{bmatrix}$$

- Therefore, the overall transformation matrix for a counterclockwise rotation by an angle θ about the point (x_p, y_p) is given as,

$$T_1 \cdot R \cdot T_2 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -x_p & -y_p & 1 \end{bmatrix} \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ x_p & y_p & 1 \end{bmatrix}$$

$$= \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ -x_p\cos\theta + y_p\sin\theta & -x_p\sin\theta - y_p\cos\theta & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ x_p & y_p & 1 \end{bmatrix}$$

$$= \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ -x_p\cos\theta + y_p\sin\theta + x_p & -x_p\sin\theta - y_p\cos\theta + y_p & 1 \end{bmatrix} \quad \ldots (1)$$

Q.6 Prove that two 2D rotations about the origin commute; that is $R_1R_2 = R_2R_1$. **[SPPU : Dec.-10, Marks 8]**

Ans. : We have

$$R_1R_2 = \begin{bmatrix} \cos\theta_1 & \sin\theta_1 \\ -\sin\theta_1 & \cos\theta_1 \end{bmatrix} \begin{bmatrix} \cos\theta_2 & \sin\theta_2 \\ -\sin\theta_2 & \cos\theta_2 \end{bmatrix} \text{ and}$$

$$R_2R_1 = \begin{bmatrix} \cos\theta_2 & \sin\theta_2 \\ -\sin\theta_2 & \cos\theta_2 \end{bmatrix} \begin{bmatrix} \cos\theta_1 & \sin\theta_1 \\ -\sin\theta_1 & \cos\theta_1 \end{bmatrix}$$

$$\therefore R_1R_2 = \begin{bmatrix} \cos\theta_1 \cdot \cos\theta_2 - \sin\theta_1\sin\theta_2 & \cos\theta_1 \cdot \sin\theta_2 - \sin\theta_1\cos\theta_2 \\ -\sin\theta_1 \cdot \cos\theta_2 - \cos\theta_1\sin\theta_2 & -\sin\theta_1 \cdot \sin\theta_2 + \cos\theta_1\cos\theta_2 \end{bmatrix}$$

and

$$R_2R_1 = \begin{bmatrix} \cos\theta_2 \cdot \cos\theta_1 - \sin\theta_2\sin\theta_1 & \cos\theta_2 \cdot \sin\theta_1 + \sin\theta_2 \cdot \cos\theta_1 \\ -\sin\theta_2 \cdot \cos\theta_1 - \cos\theta_2\sin\theta_1 & -\sin\theta_2 \cdot \sin\theta_1 + \cos\theta_2 \cdot \cos\theta_1 \end{bmatrix}.$$

Since, multiplication is commutative $\cos\theta_1 \cdot \cos\theta_2 = \cos\theta_2 \cdot \cos\theta_1$. Therefore, $R_1R_2 = R_2R_1$.

Q.7 A 2D rectangular block with 1 unit height and 2 units width has one vertex "A" at origin. The block is shifted by 1 unit in X-direction and scaled by 2 units along Y-direction. Draw initial state of the rectangle and transformed final state of given rectangle. Give complete mathematical formulation.

☞[SPPU : May-11, Marks 18]

Ans. :

$$T = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix} \qquad S = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\therefore \quad TS = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 1 & 0 & 1 \end{bmatrix}.$$

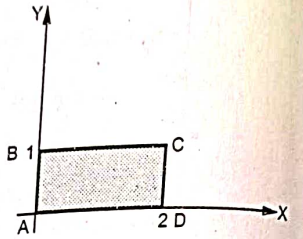


Fig. Q.7.1 Initial state

Co-ordinates of rectangle are :

A(0,0), B(0,1), C(2,1), D(2,0)

$$\begin{bmatrix} A' \\ B' \\ C' \\ D' \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 1 \\ 2 & 1 & 1 \\ 2 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 1 & 0 & 1 \end{bmatrix}$$

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

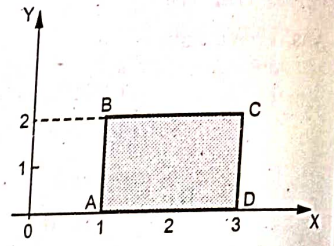


Fig. Q.7.2 Final state

Q.8 Fig. Q.8.1 and Fig. Q.8.2 show basic 2D blocks. Apply translation and scaling transformations to get the Fig. Q.8.3. Draw diagrams of all intermediate steps.

☞[SPPU : May-10, Marks 18]

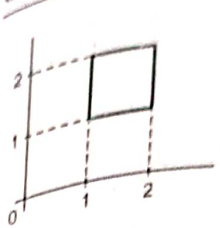


Fig. Q.8.1

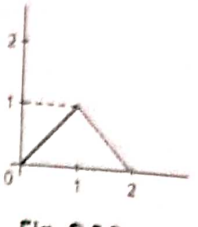


Fig. Q.8.2

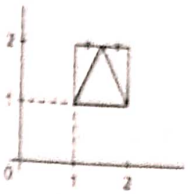


Fig. Q.8.3

Ans. : The scaling matrix is

$$S = \begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Here $S_x = 0.5$, $S_y = 1$

$$S = \begin{bmatrix} 0.5 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

The translation matrix is

$$T = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ t_x & t_y & 1 \end{bmatrix}$$

Here, $t_x = 1$, $t_y = 1$

$$\therefore \quad T = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

$\therefore$ The overall transformation matrix is

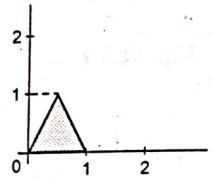
$$S.T. = \begin{bmatrix} 0.5 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 0.5 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

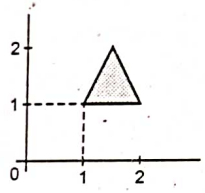
By applying this transformation to a triangle (0, 0), (1, 1), (2, 0), we get

$$\begin{bmatrix} 0 & 0 & 1 \\ 1 & 1 & 1 \\ 2 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0.5 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 \\ 1.5 & 2 & 1 \\ 2 & 1 & 1 \end{bmatrix}$$

After applying scaling transformation to Fig. Q.8.2,

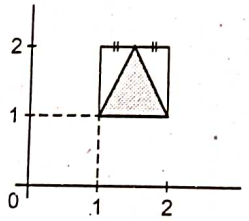


After applying translation transformation to above figure,



No transformation is applied to Fig. Q.8.1.

∴ By applying overall transformation we get,



Q.9 Consider the square A(1, 0), B(0, 0), C(0, 1) and D(1, 1). Show the steps to rotate the given square by 45° clockwise about point A(1, 0). **☞[SPPU : Dec.-10, Marks 10, June-22, Marks 6]**

Ans. : The translation matrix to move point (x_p, y_p) to the origin is given as,

$$T_1 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -x_p & -y_p & 1 \end{bmatrix}$$

The rotation matrix for clockwise rotation of point about the origin is given as,

$$R = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

The translation matrix to move the center point back to its original position is give as,

$$T_2 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ x_p & y_p & 1 \end{bmatrix}$$

Therefore, the overall transformation matrix for a clockwise rotation by an angle θ about the point (x_p, y_p) is given as -

$$T_1 \cdot R \cdot T_2 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -x_p & -y_p & 1 \end{bmatrix} \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ x_p & y_p & 1 \end{bmatrix}$$

$$\therefore T_1 \cdot R \cdot T_2 = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ -x_p\cos\theta - y_p\sin\theta & x_p\sin\theta - y_p\cos\theta & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ x_p & y_p & 1 \end{bmatrix}$$

$$= \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ -x_p\cos\theta - y_p\sin\theta + x_p & x_p\sin\theta - y_p\cos\theta + y_p & 1 \end{bmatrix}$$

Here, $\theta = 45°$, $x_p = 1$ and $y_p = 0$.

Substituting values we get,

$$T_1 \cdot R \cdot T_2 = \begin{bmatrix} \cos 45 & -\sin 45 & 0 \\ \sin 45 & \cos 45 & 0 \\ -\cos 45 + 1 & \sin 45 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & 0 \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 \\ -\frac{1}{\sqrt{2}} + 1 & \frac{1}{\sqrt{2}} & 1 \end{bmatrix}$$

Q.10 Consider a square P(0, 0), Q(0, 10), R(10, 10), S(10, 0). Rotate the square anticlockwise about fixed point R(10, 10) by an angle 45°.

[SPPU : May-19, Marks 4]

Ans. :

$$T = \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ -x_p\cos\theta + y_p\sin\theta + x_p & -x_p\sin\theta - y_p\cos\theta + y_p & 1 \end{bmatrix}$$

Note : Here, rotation is anticlockwise

Given : $\theta = 45$, $x_p = 10$ and $y_p = 10$

$$T = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 \\ -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 \\ 10 & -10(\sqrt{2})+10 & 1 \end{bmatrix}$$

$$\begin{bmatrix} P' \\ Q' \\ R' \\ S' \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 10 & 1 \\ 10 & 10 & 1 \\ 10 & 0 & 1 \end{bmatrix} \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 \\ -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 \\ 10 & \frac{-20}{\sqrt{2}}+10 & 1 \end{bmatrix} = \begin{bmatrix} 10 & \frac{-20}{\sqrt{2}}+10 & 1 \\ \frac{-10}{\sqrt{2}}+10 & \frac{-10}{\sqrt{2}}+10 & 1 \\ 10 & 10 & 1 \\ \frac{10}{\sqrt{2}}+10 & \frac{-10}{\sqrt{2}}+10 & 1 \end{bmatrix}$$

Q.11 Magnify the triangle with vertices A(0, 0), B(1, 1), C(5, 2) to twice its size as well as rotate it by 45°. Derive the translation matrices.

[SPPU : Dec.-11,13, May-13, Marks 8]

Ans. :

Scaling matrix is given by $S = \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix}$ and

Rotation matrix is given by $R = \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$

$$\therefore SR = \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} s_x\cos\theta & s_x\sin\theta & 0 \\ -s_y\sin\theta & s_y\cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Where $s_x = 2$, $s_y = 2$ and $\theta = 45°$

$$\therefore SR = \begin{bmatrix} 2\cos 45 & 2\sin 45 & 0 \\ -2\sin 45 & 2\cos 45 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} A' \\ B' \\ C' \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 1 & 1 \\ 5 & 2 & 1 \end{bmatrix} \begin{bmatrix} 2\cos 45 & 2\sin 45 & 0 \\ -2\sin 45 & 2\cos 45 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 2.828 & 1 \\ 4.2426 & 9.898 & 1 \end{bmatrix}$$

Q.12 Perform scaling on a triangle (1, 1), (8, 1) and (1, 9) with scaling factor of 2 in both x and y directions. Find the final co-ordinates of triangle.

[SPPU : May-17, Marks 2]

Ans. :

$$S = \begin{bmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} A' \\ B' \\ C' \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 \\ 8 & 1 & 1 \\ 1 & 9 & 1 \end{bmatrix} \begin{bmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 2 & 2 & 1 \\ 16 & 2 & 1 \\ 2 & 18 & 1 \end{bmatrix}$$

The co-ordinates of triangle are (2, 2), (16, 2), (2, 18).

Q.13 For origin centered unit square, rotate 45° clockwise, scale by a factor 2 in x - direction. Find resultant co-ordinates of square (write required matrices).

[SPPU : Dec.-17, Marks 4]

Ans. :

$$R = \begin{bmatrix} \cos 45 & -\sin 45 & 0 \\ \sin 45 & \cos 45 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$S = \begin{bmatrix} 2 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$R.S. = \begin{bmatrix} 1/\sqrt{2} & -1/\sqrt{2} & 0 \\ 1/\sqrt{2} & 1/\sqrt{2} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 2 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} \sqrt{2} & -1/\sqrt{2} & 0 \\ \sqrt{2} & 1/\sqrt{2} & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} A' \\ B' \\ C' \\ D' \end{bmatrix} = \begin{bmatrix} -0.5 & -0.5 & 1 \\ -0.5 & 0.5 & 1 \\ 0.5 & 0.5 & 1 \\ 0.5 & -0.5 & 1 \end{bmatrix} \begin{bmatrix} \sqrt{2} & -1/\sqrt{2} & 0 \\ \sqrt{2} & 1/\sqrt{2} & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} -\sqrt{2} & 0 & 0 \\ 0 & 1/\sqrt{2} & 0 \\ \sqrt{2} & 0 & 0 \\ 0 & -1/\sqrt{2} & 0 \end{bmatrix}$$

Q.14 Rotate origin centered square with 2 unit length of each side, in clockwise direction with rotation angle of 90°.

☞[SPPU : May-18, Marks 3]

Ans. :

$$p' = \begin{bmatrix} 1 & 1 \\ -1 & 1 \\ -1 & -1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$$

$$= \begin{bmatrix} 1 & 1 \\ -1 & 1 \\ -1 & -1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} = \begin{bmatrix} 1 & -1 \\ 1 & 1 \\ -1 & 1 \\ -1 & -1 \end{bmatrix}$$

Q.15 Find a transformation of a triangle A(1, 0) B(0, 1) C(1, 1) by translating one unit in x and y directions and then rotating 45° about the origin.

☞[SPPU : Dec.-19, Marks 6]

Ans. : The rotation matrix is,

$$R = \begin{bmatrix} \cos 45 & \sin 45 & 0 \\ -\sin 45 & \cos 45 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1/\sqrt{2} & 1/\sqrt{2} & 0 \\ -1/\sqrt{2} & 1/\sqrt{2} & 0 \\ 0 & 0 & 1 \end{bmatrix}$$ and

The translation matrix is

$$T = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

$$T \cdot R = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} 1/\sqrt{2} & 1/\sqrt{2} & 0 \\ -1/\sqrt{2} & 1/\sqrt{2} & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1/\sqrt{2} & 1/\sqrt{2} & 0 \\ -1/\sqrt{2} & 1/\sqrt{2} & 0 \\ 0 & \sqrt{2} & 1 \end{bmatrix}$$

$$\therefore \begin{bmatrix} A' \\ B' \\ C' \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} 1/\sqrt{2} & 1/\sqrt{2} & 0 \\ -1/\sqrt{2} & 1/\sqrt{2} & 0 \\ 0 & \sqrt{2} & 1 \end{bmatrix} = \begin{bmatrix} 1/\sqrt{2} & 3/\sqrt{2} & 1 \\ -1/\sqrt{2} & 3/\sqrt{2} & 1 \\ 0 & 2\sqrt{2} & 1 \end{bmatrix}$$

6.5 : Reflection and Shear Transformations

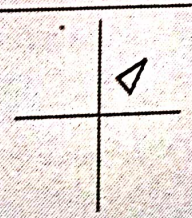
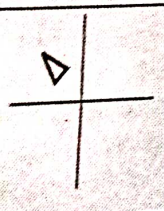
Q.16 Explain the term reflection.

☞[SPPU : May-12,19, Dec.-18, Marks 3]

Ans. : • A reflection is a transformation that produces a mirror image of an object relative to an axis of reflection.

• We can choose an axis of reflection in the xy plane or perpendicular to the xy plane.

Point to Remember

Reflection	Transformation matrix	Original image	Reflected image
Reflection about Y-axis	$\begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$		

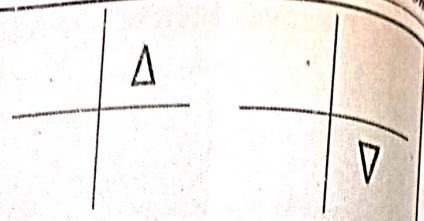
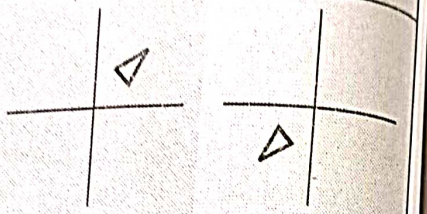
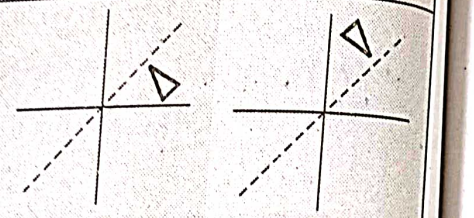
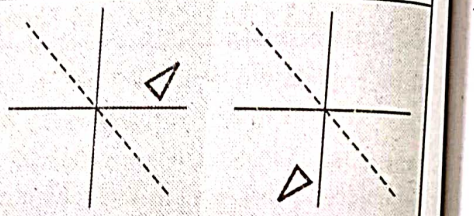
Reflection about X axis	$\begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$	
Reflection about origin	$\begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$	
Reflection about line y = x	$\begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$	
Reflection about line y = – x	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$	

Table Q.16.1 Common reflections

Q.17 What is shear transformation ? Explain X-shear and Y-shear.

☞[SPPU : May-08,07,12,14,16, Dec.-12,16, Marks 5]

Ans. : • A transformation that slants the shape of an object is called the shear transformation.

• Two common shearing transformations are used. One shifts x co-ordinate values and other shifts y co-ordinate values. However, in both the cases only one co-ordinate (x or y) changes its co-ordinates and other preserves its values.

X-shear

• The x shear preserves the y co-ordinates, but changes the x values which causes vertical lines to tilt right or left as shown in the Fig. Q.17.1.

• The transformation matrix for x shear is given as

$$X\_sh = \begin{bmatrix} 1 & 0 & 0 \\ Sh_x & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$x' = x + Sh_x \cdot y \quad \text{and}$$

$$\therefore \quad y' = y \qquad \ldots (1)$$

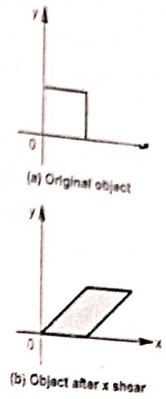


Fig. Q.17.1

Y Shear

• The y shear preserves the x co-ordinates, but changes the y values which causes horizontal lines to transform into lines which slope up or down, as shown in the Fig. Q.17.2.

• The transformation matrix for y shear is given as

$$Y\_sh = \begin{bmatrix} 1 & Sh_y & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\therefore \quad x' = x \quad \text{and}$$

$$y' = y + Shy \cdot x \qquad \ldots (2)$$

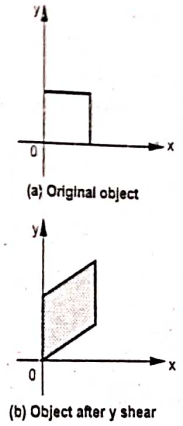


Fig. Q.17.2

6.6 : Inverse Transformation

Q.18 Explain inverse transformation and derive the matrix for inverse transformation.

☞[SPPU : Dec.-06, 18, May-07, 08, Marks 8]

Ans. : • When we apply any transformation to point (x, y) we get a new point (x′, y′). Sometimes it may require to undo the applied transformation. In such a case we have to get original point (x, y) from the point(x′, y′). This can be achieved by **inverse transformation.**

- The inverse transformation uses the matrix inverse of the transformation matrix to get the original point (x, y). The inverse of a matrix is another matrix such that when the two are multiplied together, we get the identity matrix.
- If the inverse of matrix T is T^{-1}, then

$$T\,T^{-1} = T^{-1}T = I \quad \ldots (1)$$

where I is the identity matrix with all elements along the major diagonal having value 1 and all other elements having value zero.

$$I = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

- The elements for the inverse matrix T^{-1} can be calculated from the elements of T as

$$t_{ij}^{-1} = \frac{(-1)^{i+j} \det M_{ji}}{\det T} \quad \ldots (2)$$

where t_{ij}^{-1} is the element in the i^{th} row and j^{th} column of T^{-1} and M_{ji} is the (n – 1) by (n – 1) submatrix obtained by deleting the j^{th} row and i^{th} column of the matrix A. The det M_{ji} and det T is the determinant of the M_{ji} and T matrices.

- The determinant of a 2 × 2 matrix is

$$\det \begin{vmatrix} t_{11} & t_{12} \\ t_{21} & t_{22} \end{vmatrix} = t_{11} \cdot t_{22} - t_{12} \cdot t_{21} \quad \ldots (3)$$

- The determinant of a 3 × 3 matrix is

$$\det T = t_{11} \cdot (t_{22}\, t_{33} - t_{23}\, t_{32}) - t_{12} \cdot (t_{21}\, t_{33} - t_{23}\, t_{31}) + t_{13} \cdot (t_{21}\, t_{32} - t_{22}\, t_{31}) \quad \ldots (4)$$

- In general form, the determinant of T is given by

$$\det T_j = \sum t_{ij}\, (-1)^{i+j} \det M_{ij} \quad \ldots (5)$$

where M_{ij} is the submatrix formed by deleting row i and column j from matrix T.

- The inverse of the homogeneous co-ordinate transformation matrix can be given as

$$\begin{bmatrix} a & d & 0 \\ b & e & 0 \\ c & f & 1 \end{bmatrix}^{-1} = \frac{1}{ae-bd} \begin{bmatrix} e & -d & 0 \\ -b & a & 0 \\ bf-ce & cd-af & ae-bd \end{bmatrix}$$

- It is important to note that the elements of inverse matrix T^{-1} can be calculated from the element of T as

$$t_{ij}^{-1} = \frac{(-1)^{i+j} \det M_{ji}}{\det T} \quad \ldots (6)$$

In the above equation the term det T is in the denominator. Hence, we can obtain an inverse matrix if and only if the determinant of the matrix is nonzero.

END...

Unit III

7 3D Transformations and Projections

7.1 : 3D Translation

Q.1 Obtain the 3D transformation matrices for translation.

☞ [SPPU : Dec.-05,07,08, May-05,07,08,18, Marks 2]

Ans. : • Three dimensional transformation matrix for translation with homogeneous coordinates is as given below. It specifies three coordinates with their own translation factor.

$$T = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ t_x & t_y & t_z & 1 \end{bmatrix}$$

$$\therefore \quad P' = P \cdot T$$

$$[x' \ y' \ z' \ 1] = [x \ y \ z \ 1] \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ t_x & t_y & t_z & 1 \end{bmatrix}$$

$$= [x + t_x \quad y + t_y \quad z + t_z \quad 1] \qquad \ldots (1)$$

7.2 : 3D Scaling

Q.2 Obtain 3D transformation matrix for scaling.

☞[SPPU : May-05, 07, 08, 16, Dec.-05, 07, 08, 16,18, Marks 2]

Ans. : • Three dimensional transformation matrix for scaling with homogeneous co-ordinates is as given below.

• It specifies three co-ordinates with their own scaling factor.

$$S = \begin{bmatrix} S_x & 0 & 0 & 0 \\ 0 & S_y & 0 & 0 \\ 0 & 0 & S_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$P' = P \cdot S$$

$$\therefore \quad [x' \ y' \ z' 1] = [x \ y \ z \ 1] \begin{bmatrix} S_x & 0 & 0 & 0 \\ 0 & S_y & 0 & 0 \\ 0 & 0 & S_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$= [x \cdot S_x \quad y \cdot S_y \quad z \cdot S_z \quad 1] \qquad \ldots (1)$$

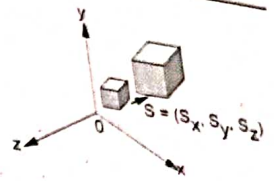


Fig. Q.2.1 3 D Scaling

7.3 : 3D Rotation

Q.3 Obtain the 3D transformation matrix for rotation about z-axis.

☞[SPPU : May-11,19, Marks 3]

OR Explain the rotation about all co-ordinate axis.

☞[SPPU : May-06, 07, Dec.-16,18, Marks 8]

OR Give homogenous transformation matrix for 3D rotation with respect to y axis.

☞ [SPPU : May-16, Marks 2]

Ans. : • Three dimensional transformation matrix for each co-ordinate axes rotations with homogeneous co-ordinate are as given below.

$$R_z = \begin{bmatrix} \cos\theta & \sin\theta & 0 & 0 \\ -\sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

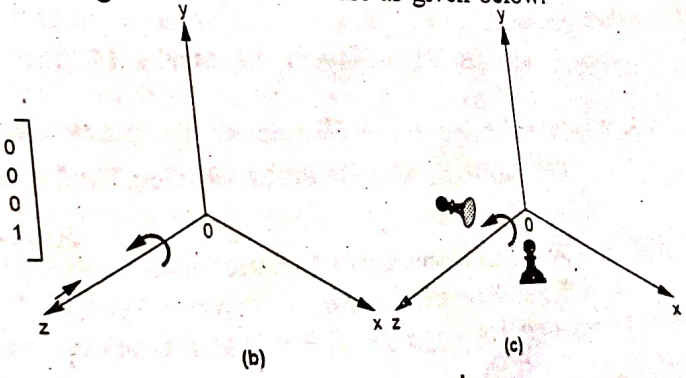


Fig. Q.3.1 Rotation about z axis

Unit III

7 3D Transformations and Projections

7.1 : 3D Translation

Q.1 Obtain the 3D transformation matrices for translation.

☞ [SPPU : Dec.-05,07,08, May-05,07,08,18, Marks 2]

Ans. : • Three dimensional transformation matrix for translation with homogeneous coordinates is as given below. It specifies three coordinates with their own translation factor.

$$T = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ t_x & t_y & t_z & 1 \end{bmatrix}$$

$$\therefore \quad P' = P \cdot T$$

$$[x' \ y' \ z' \ 1] = [x \ y \ z \ 1] \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ t_x & t_y & t_z & 1 \end{bmatrix}$$

$$= [x + t_x \quad y + t_y \quad z + t_z \quad 1] \qquad \ldots (1)$$

7.2 : 3D Scaling

Q.2 Obtain 3D transformation matrix for scaling.

☞[SPPU : May-05, 07, 08, 16, Dec.-05, 07, 08, 16,18, Marks 2]

Ans. : • Three dimensional transformation matrix for scaling with homogeneous co-ordinates is as given below.

• It specifies three co-ordinates with their own scaling factor.

$$S = \begin{bmatrix} S_x & 0 & 0 & 0 \\ 0 & S_y & 0 & 0 \\ 0 & 0 & S_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$P' = P \cdot S$$

$$\therefore \quad [x' \ y' \ z' 1] = [x \ y \ z \ 1] \begin{bmatrix} S_x & 0 & 0 & 0 \\ 0 & S_y & 0 & 0 \\ 0 & 0 & S_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$= [x \cdot S_x \quad y \cdot S_y \quad z \cdot S_z \quad 1] \qquad \ldots (1)$$

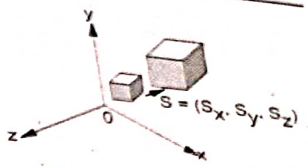


Fig. Q.2.1 3 D Scaling

7.3 : 3D Rotation

Q.3 Obtain the 3D transformation matrix for rotation about z-axis.

☞[SPPU : May-11,19, Marks 3]

OR Explain the rotation about all co-ordinate axis.

☞[SPPU : May-06, 07, Dec.-16,18, Marks 8]

OR Give homogenous transformation matrix for 3D rotation with respect to y axis. **☞ [SPPU : May-16, Marks 2]**

Ans. : • Three dimensional transformation matrix for each co-ordinate axes rotations with homogeneous co-ordinate are as given below.

$$R_z = \begin{bmatrix} \cos\theta & \sin\theta & 0 & 0 \\ -\sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

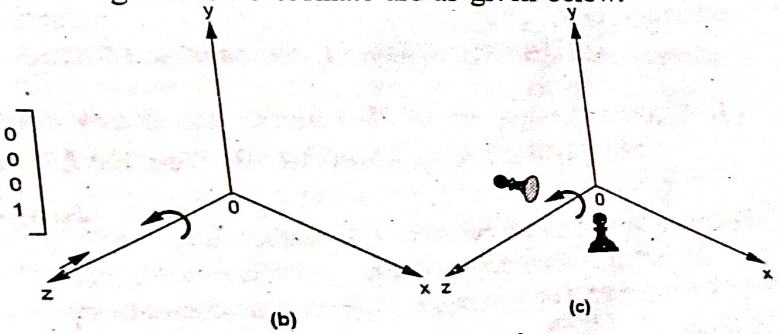


Fig. Q.3.1 Rotation about z axis

- The positive value of angle θ indicates counterclockwise rotation. For clockwise rotation value of angle θ is negative.

$$R_x = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & \sin\theta & 0 \\ 0 & -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

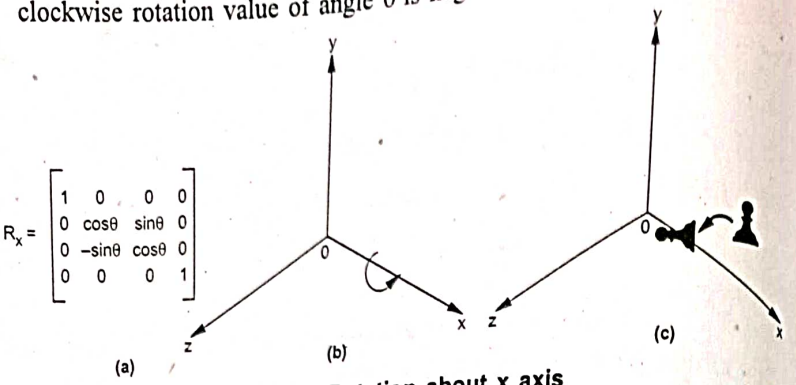


Fig. Q.3.2 Rotation about x axis

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

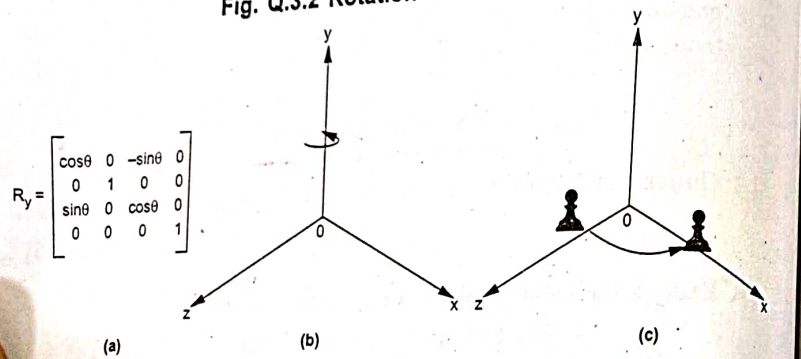


Fig. Q.3.3 Rotation about y axis

7.4 : Rotation about Arbitrary Axis

Q.4 Derive the transformation matrix for rotation about any arbitrary axis.

☞ **[SPPU : May-11, 12, Marks 10 Dec.-11, Marks 12]**

OR Explain the concept of 3D rotation about any arbitrary axis.

☞ **[SPPU : May-06,08,12,14, Dec.-06,07,08,14, Marks 8 June-22, Marks 6]**

Ans. : • When an object is to be rotated about an axis that is not parallel to one of the coordinate axes, we have to perform some additional transformations. The sequence of these transformations is given below.

1. Translate the object so that rotation axis specified by unit vector u passes through the coordinate origin. (See Fig. Q.4.1 (a) and (b))

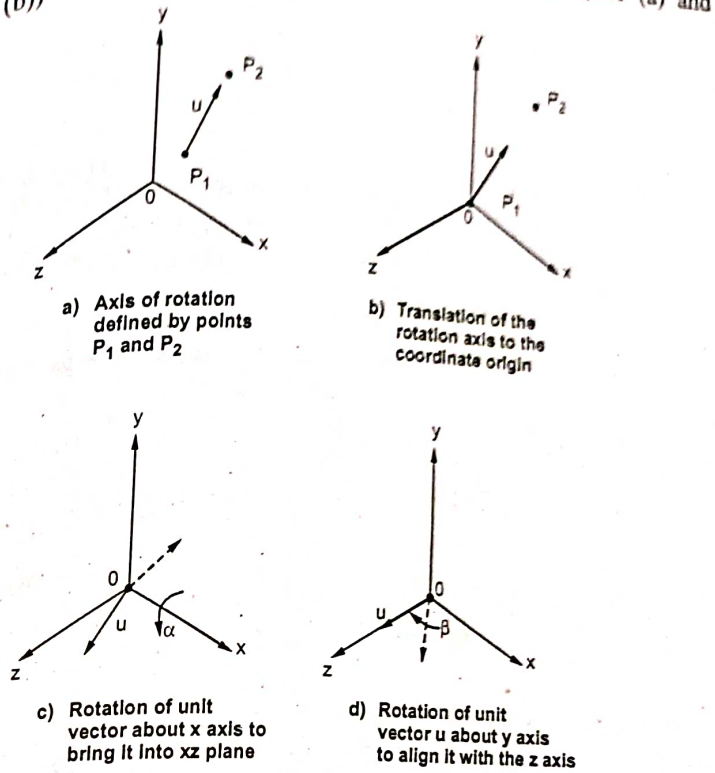


Fig. Q.4.1

2. Rotate the object so that the axis of rotation coincides with one of the coordinate axes. Usually the z axis is preferred. To coincide the axis of rotation to z axis we have to first perform rotation of unit vector u about x axis to bring it into xz plane and then perform rotation about y axis to coincide it with z axis. (See Fig. Q.4.1 (c) and (d))
3. Perform the desired rotation θ about the z axis.
4. Apply the inverse rotation about y axis and then about x axis to bring the rotation axis back to its original orientation.
5. Apply the inverse translation to move the rotation axis back to its original position.

Derivation of transformation matrix

The translation matrix is given as

$$T = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -x_1 & -y_1 & -z_1 & 1 \end{bmatrix}$$

- To perform the rotation of unit vector u about x axis. The rotation of u around the x axis into the xz plane is accomplished by rotating u′ (0, b, c) through angle α into the z axis and the cosine of the rotation angle α can be determined from the dot product of u′ and the unit vector u_z (0, 0, 1) along the z axis.

$$\cos\alpha = \frac{u' \cdot u_z}{|u'||u_z|} = \frac{c}{d}$$ where u′ (0, b, c) = bJ + cK and

u_z (0, 0, 1) = K

$$= \frac{c}{|u'||u_z|}$$

$$= \frac{c}{|u'|}$$ Since $|u_z| = 1$

$$= \frac{c}{d}$$

where d is the magnitude of u′.

$$d = \sqrt{b^2 + c^2}$$

- Similarly, we can determine the sine of α from the cross product of u′ and u_z.

$$u' \times u_z = u_x |u'| |u_z| \sin\alpha \quad \ldots (1)$$

and the Cartesian form for the cross product gives us

$$u' \times u_z = u_x \cdot b \quad \ldots (2)$$

- Equating the right sides of equations (1) and (2) we get

$$u_x |u'| |u_z| \sin\alpha = u_x \cdot b$$

$$\therefore \quad |u'| |u_z| \sin\alpha = b$$

$$\therefore \quad \sin\alpha = \frac{b}{|u'||u_z|}$$

$$= \frac{b}{d}$$ since $|u_z| = 1$ and $|u'| = d$

- By substituting values of cos α and sin α the rotation matrix R_x can be given as

$$R_x = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & c/d & b/d & 0 \\ 0 & -b/d & c/d & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- To perform the rotation of unit vector about y axis. This can be achieved by rotating u″ (a, 0, d) through angle β onto the z axis. Using similar equations we can determine cos β and sin β as follows.
- We have angle of rotation = $-\beta$

$$\therefore \quad \cos(-\beta) = \cos\beta = \frac{u'' \cdot u_z}{|u''||u_z|}$$ where u″ = aI + dK and

$$u_z = K$$

$$= \frac{d}{|u''||u_z|}$$

$$= \frac{d}{|u''|}$$ ∵ $|u_z| = 1$

$$= \frac{d}{\sqrt{a^2 + d^2}}$$

- Consider cross product of u″ and u_z

$$u'' \times u_z = u_y |u''| |u_z| \sin(-\beta)$$

$$= -u_y |u''| |u_z| \sin\beta$$ ∵ $\sin(-\theta) = -\sin\theta$

- Cartesian form of cross product gives us

$$u'' \times u_z = u_y (+a)$$

- Equating above equations,

$$-|u''| |u_z| \sin\beta = a$$

$$\therefore \quad \sin\beta = \frac{-a}{|u''||u_z|}$$

$$= \frac{-a}{|u''|}$$ ∵ $|u_z| = 1$

$$= \frac{-a}{\sqrt{a^2 + d^2}}$$

but we have, $d = \sqrt{b^2 + c^2}$

$$\therefore \quad \cos\beta = \frac{d}{\sqrt{a^2 + d^2}} = \frac{\sqrt{b^2 + c^2}}{\sqrt{a^2 + b^2 + c^2}}$$

and $$\sin\beta = \frac{-a}{\sqrt{a^2 + d^2}} = \frac{-a}{\sqrt{a^2 + b^2 + c^2}}$$

By substituting values of $\cos\beta$ and $\sin\beta$ in the rotation matrix R_y and by substituting $x = \sqrt{b^2 + c^2}$ and $|V| = \sqrt{a^2 + b^2 + c^2}$ we have R_{xy},

$\therefore$ Resultant rotation matrix $R_{xy} = R_x \cdot R_y$

$$R_{xy} = \begin{bmatrix} \frac{\lambda}{|V|} & 0 & \frac{a}{|V|} & 0 \\ \frac{-ab}{|V|\lambda} & \frac{c}{\lambda} & \frac{b}{|V|} & 0 \\ \frac{-ac}{|V|\lambda} & \frac{-b}{\lambda} & \frac{c}{|V|} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$R_z = \begin{bmatrix} \cos\theta & \sin\theta & 0 & 0 \\ -\sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The overall transformation matrix for rotation about an arbitrary axis then can be expressed as the concatenation of five individual transformations.

$$R(\theta) = T \cdot R_{xy} \cdot R_z \cdot R_{xy}^{-1} \cdot T^{-1}$$

7.5 : Concept of Parallel and Perspective Projections

Q.5 Define projection.

Ans. : The process of displaying 3D into 2D display unit is known as projection. The projection transforms 3D objects into a 2D projection plane.

Q.6 What do you mean by parallel projection ?

Ans. : Parallel projection is one in which z co-ordinate is discarded and parallel lines from each vertex on the object are extended until they intersect the view plane.

Q.7 What do you mean by perspective projection ?

Ans. : Perspective projection is one in which the lines of projection are not parallel. Instead, they all converge at a single point called the center of projection.

Q.8 What is projection reference point ?

Ans. : In perspective projection, the lines of projection are not parallel. Instead, they all converge at a single point called projection reference

Q.9 Explain parallel and perspective projection ?

[SPPU : Dec.-08,10,15, May-07,12,13,19, June-22, Marks 5]

Ans. : Parallel Projection

- In parallel projection, z co-ordinate is discarded and parallel lines from each vertex on the object are extended until they intersect the view plane.
- The point of intersection is the **projection of the vertex.**
- We connect the projected vertices by line segments which correspond to connections on the original object.
- As shown in the Fig. Q.9.1, a parallel projection preserves relative proportions of objects but does not produce the realistic views.

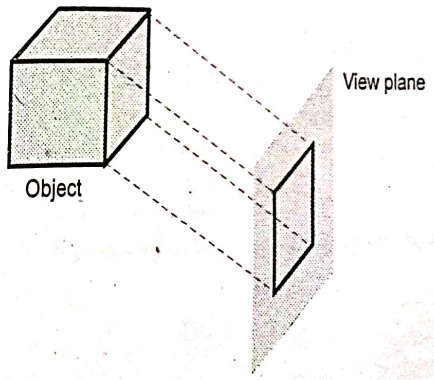


Fig. Q.9.1 Parallel projection of an object to the view plane

- Types of parallel projections are : Orthographic projection and oblique projection.

Perspective Projection

- The perspective projection, on the other hand, **produces realistic views** but **does not preserve relative proportions.**
- In perspective projection, the lines of projection are not parallel. Instead, they all converge at a single point called the **center of projection** or **projection reference point.**
- The object positions are transformed to the view plane along these converged projection lines and the projected view of an object is determined by calculating the intersection of the converged projection lines with the view plane, as shown in the Fig. Q.9.2.

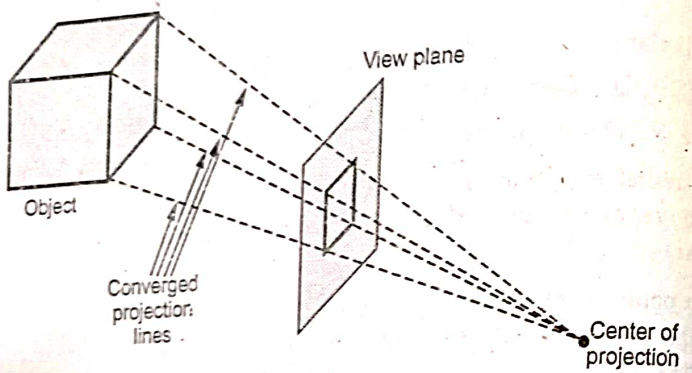


Fig. Q.9.2 Perspective projection of an object to the view plane

7.6 : Types of Parallel Projections

Q.10 Explain in detail types of parallel projection.

☞ [SPPU : June-22, Marks 6]

Ans. : • Parallel projections are basically categorized into two types, depending on the relation between the direction of projection and the normal to the view plane.

- When the direction of the projection is normal (perpendicular) to the view plane, we have an **orthographic parallel projection.** Otherwise, we have an oblique parallel projection. *Fig. Q.10.1* illustrates the two types of parallel projection.
- Orthographic Projections are further classified as axonometric orthographic projection and multiview orthographic projection.

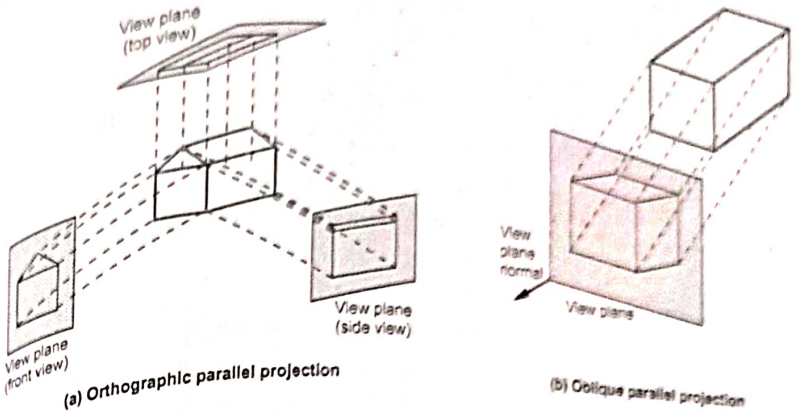


Fig. Q.10.1

- The orthographic projection can display more than one face of an object. Such an orthographic projection is called **axonometric** orthographic projection.
- **Axonometric projections are of three types :**
  - **Isometric :** All three principle axes are foreshortened equally.
  - **Dimetric :** Two principle axes are foreshortened equally.
  - **Trimetric :** All three principle axes are foreshortened unequally.
- The oblique projections are further classified as the **cavalier** and **cabinet projections.**
- For the cavalier projection, the direction of projection makes a 45° angle with the view plane.
- When the direction of projection makes an angle of arctan (2) = 63.4° with the view plane, the resulting view is called a **cabinet projection.**

7.7 : Types of Perspective Projections

Q.11 Give examples one for each case of 3D objects having i) Never a vanishing point ii) At most one vanishing point iii) At most two vanishing point iv) At most three vanishing points.

☞ [SPPU : May-13, Marks 8; June-22, Marks 6]

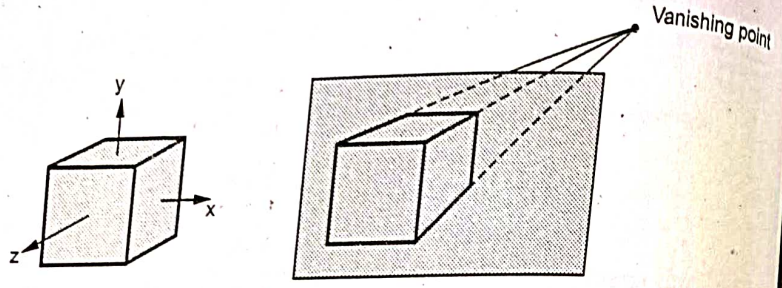
Ans. :

i) Never a vanishing point :

It is a parallel project (Refer Q.9)

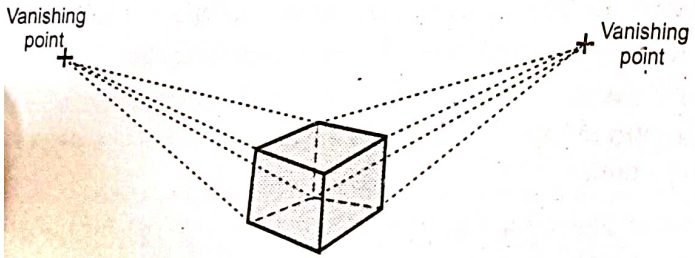
ii) At most one vanishing point

- Here, the object is placed so that one of its surfaces is parallel to the plane of projection. This is shown in the Fig. Q.11.1 (b).
- One point perspective projections are simple to produce and find many practical uses in engineering, architecture, and in computer graphics.



(a) Co-ordinate description

(b) One-point perspective projection of a cube



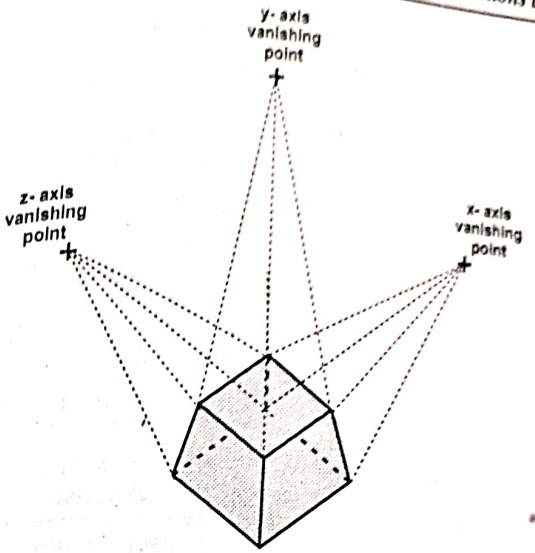
(c) Two-points perspective projection of a cube

Q.11.1

At most two vanishing point

this projection, two surfaces of the object have vanishing points. In case of a cube this is achieved if the object is rotated along its xis, so that lines along that axis remain parallel to the viewing plane, those along the two other axes have vanishing points.

Q.11.1 (c) shows a two-point perspective of a cube.



(d) Three-point perspective of a cube

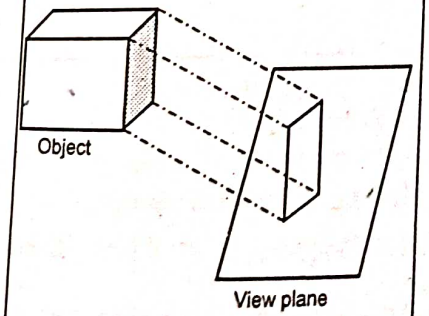
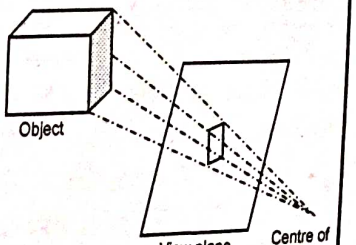
Fig. Q.11.1 Perspective projections

iv) At most three vanishing points

- A three-point perspective is achieved by positioning the object so that none of its axes are parallel to the plane of projection. Although the visual depth cues in a three-point perspective are stronger than in the two-point perspective, the resulting geometrical deformations are sometimes disturbing to the viewer.
- Fig. Q.11.1 (d) is a three-point perspective projection of a cube.

Q.12 Give comparison between parallel and perspective projections.

Ans. :

Sr. No.	Parallel Projection	Perspective Projection
1.		

2.	It is a projection where co-ordinates of the given object are transformed by forming parallel lines to a view plane.	It is a projection where co-ordinates of the given object are transformed by forming converging lines also known as a center of projection to a view plane.
3.	Distance of the object from the center of projection is infinite.	Distance of the object from the center of projection is finite.
4.	It produces accurate view of various sides of the object to the view plane.	It does not produce accurate view of various sides of the object to the view plane; object size varies based on the distance from the view plane after transformation.
5.	It does not provide a realistic representation of an object.	It gives a realistic representation of an object.
6.	It preserves relative proportion of an object.	It does not preserve relative proportion of an object.
7.	Types : 1. Orthographic projection 2. Oblique projection.	Types : 1. One point perspective projection. 2. Two point perspective projection. 3. Three point perspective projection.
8.	Parallel projections are best suitable for architectural drawings, where measurements are necessary.	Perspective projections are suitable for creating realistic views.

7.8 : 3D Viewing

Q.13 Explain with example, 3D viewing transformation.

☞[SPPU : May-06,13, Dec.-05,06,07,11,17, Marks 8]

OR Explain 3D viewing process with various 3D viewing parameters.

☞[SPPU : May-08,15, Dec.-13, Marks 8]

Ans. : • The 3D viewing process is inherently more complex than the 2D viewing process. In two dimensional viewing we have 2D window and 2D viewport and objects in the world co-ordinates are clipped against the window and are then transformed into the viewport for display.

• The complexity added in the three dimensional viewing is because of the added dimension and the fact that even though objects are three dimensional the display devices are only 2D.

• The mismatch between 3D objects and 2D displays is compensated by introducing **projections.**

• The projections transform 3D objects into a 2D projection plane. The Fig. Q.13.1 shows the conceptual model of the 3D transformation process.

• In 3D viewing, we specify a view volume in the world co-ordinates using modeling transformation. The world co-ordinate positions of the objects are then converted into viewing co-ordinates by **viewing transformation.**

• The **projection transformation** is then used to convert 3D description of objects in viewing co-ordinates to the 2D projection co-ordinates.

• Finally, the **workstation transformation** transforms the projection co-ordinates into the device co-ordinates.

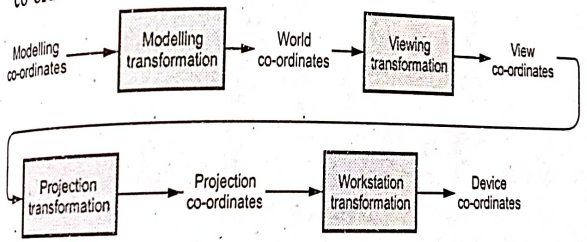


Fig. Q.13.1 Conceptual model of 3D transformation process

7.9 : View Volumes and General Projection Transformations

Q.14 Derive the transformation matrix for parallel projection.

☞[SPPU : May-06, Dec.-07,12, Marks 5]

Ans. : • The general equation of parallel projection onto a given view plane instead of xy plane in the direction of a given vector $V(x_p, y_p, z_p)$ as follows :

• Let us consider that R_0 is the view reference point, P_1 is the object point and P_2 is the projected point. Now perform the following steps :

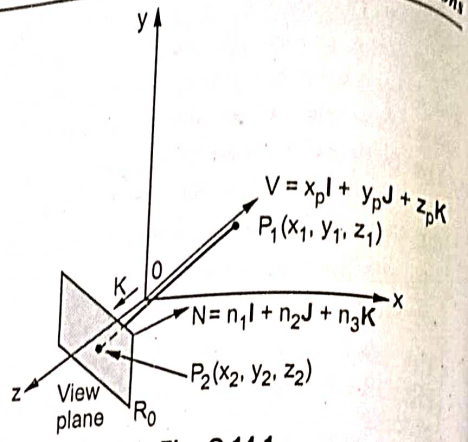


Fig. Q.14.1

1. Translate the view reference point R_0 of the view plane to the origin using the translation matrix T.
2. Perform an alignment transformation R_{xy} so that the view normal vector N of the view plane points in the direction K, the normal to the xy plane.
3. Project point P_1 on to the xy plane.
4. Perform the inverse of steps 2 and 1.

$$\therefore \quad Par_{v,N,R_o} = T \cdot R_{xy} \cdot Par_v \cdot R_{xy}^{-1} \cdot T^{-1}$$

$$= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -x_o & -y_o & -z_o & 1 \end{bmatrix} \begin{bmatrix} \frac{\lambda}{|N|} & 0 & \frac{n_1}{|N|} & 0 \\ \frac{-n_1 n_2}{\lambda |N|} & \frac{n_3}{\lambda} & \frac{n_2}{|N|} & 0 \\ \frac{-n_1 n_3}{\lambda |N|} & \frac{-n_2}{\lambda} & \frac{n_3}{|N|} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ \frac{-x_p}{z_p} & \frac{-y_p}{z_p} & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} \frac{\lambda}{|N|} & \frac{-n_1 n_2}{\lambda |N|} & \frac{-n_1 n_3}{\lambda |N|} & 0 \\ 0 & \frac{n_3}{\lambda} & \frac{-n_2}{\lambda} & 0 \\ \frac{n_1}{|N|} & \frac{n_2}{|N|} & \frac{n_3}{|N|} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ x_o & y_o & z_o & 1 \end{bmatrix}$$

This is the general equation of parallel projection on the given view ane in matrix form.

Q.15 Derive the transformation matrix for perspective projection.

Ans. : • The general perspective-projection transformations can be obtained by performing following two operations.

• Make the center line of the frustum perpendicular to the view plane by shearing the view volume.

• Scale the view volume with a scaling factor that depends on 1/z.

• These two steps are illustrated in Fig. Q.15.1.

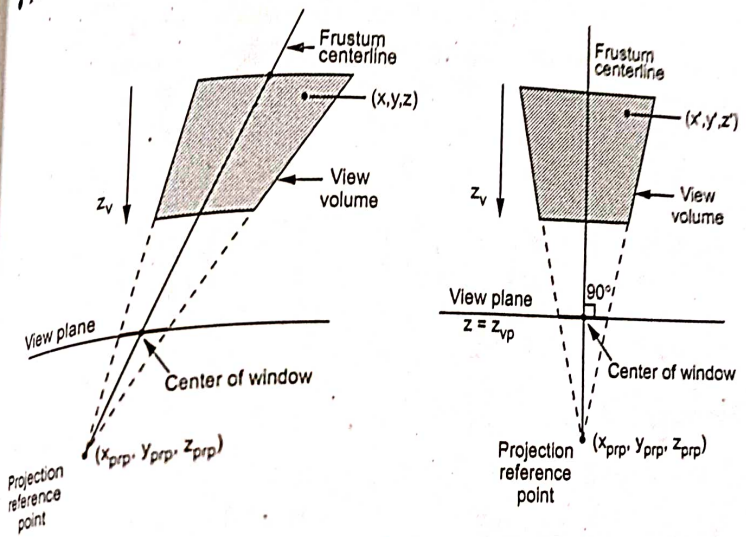


Fig. Q.15.1 Shearing the view volume to make the centerline of the frustum perpendicular to the view plane

• During shearing operation position (x, y, z) becomes position (x′, y′, z′) and its transformation matrix is given as

$$[x'y'z1] = [x \;\; y \;\; z \;\; 1] \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ a & b & 1 & 0 \\ -az_{prp} & -bz_{prp} & 0 & 1 \end{bmatrix}$$

where $$a = -\frac{x_{prp} - (x_{w\,min} + x_{w\,max})/2}{z_{prp}}$$

$$b = -\frac{y_{prp} - (y_{w\,min} + y_{w\,max})/2}{z_{prp}}$$

- By matrix multiplication we get,

$$x' = x + a\,(z - z_{prp})$$
$$y' = y + b\,(z - z_{prp})$$
$$z' = z$$

- Once we have converted a position (x, y, z) in the original view volume to position (x', y', z) in the sheared view volume, we have to apply scaling transformation. During scaling operation position (x', y', z) becomes (x'', y'', z) and its transformation matrix is given as

$$[x''\ y''\ z\ 1] = [x'\ y'\ z\ 1]\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ \frac{-x_{prp}}{z_{prp}-z_{vp}} & \frac{-y_{prp}}{z_{prp}-z_{vp}} & 1 & \frac{-1}{z_{prp}-z_{vp}} \\ \frac{x_{prp}z_{vp}}{z_{prp}-z_{vp}} & \frac{y_{prp}z_{vp}}{z_{prp}-z_{vp}} & 0 & \frac{z_{prp}}{z_{prp}-z_{vp}} \end{bmatrix}$$

- By matrix multiplication we get,

$$x'' = x'\left(\frac{z_{prp}-z_{vp}}{z_{prp}-z}\right) + x_{prp}\left(\frac{z_{vp}-z}{z_{prp}-z}\right)$$

$$y'' = y'\left(\frac{z_{prp}-z_{vp}}{z_{prp}-z}\right) + y_{prp}\left(\frac{z_{vp}-z}{z_{prp}-z}\right)$$

- Therefore, the general perspective-projection transformation can be expressed in matrix form as

$$M_{perspective} = M_{shear} \cdot M_{scale}$$

Q.16 A 3D cube dimensions (length, breadth, and height) 2 units each is placed in a 3D anti-clockwise axis system such that one of its vertex "A" is at the origin. (i.e. (0, 0, 0)) and vertex "F" in 3D space. Apply necessary transformations such that vertex F becomes the origin. Give complete mathematical formulation. Draw initial and final state of the cube. ☞**[SPPU : May-11, Marks 18]**

Ans. : To bring vertex F to origin we need translation,

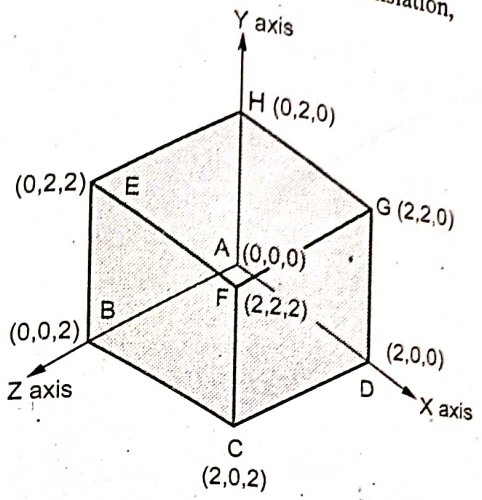


Fig. Q.16.1 Initial state of cube

$$T = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -2 & -2 & -2 & 1 \end{bmatrix}$$

$$\begin{bmatrix} A' \\ B' \\ C' \\ D' \\ E' \\ F' \\ G' \\ H' \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 2 & 1 \\ 2 & 0 & 2 & 1 \\ 2 & 0 & 0 & 1 \\ 0 & 2 & 2 & 1 \\ 2 & 2 & 2 & 1 \\ 2 & 2 & 0 & 1 \\ 0 & 2 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -2 & -2 & -2 & 1 \end{bmatrix}$$

$$\begin{bmatrix} A' \\ B' \\ C' \\ D' \\ E' \\ F' \\ G' \\ H' \end{bmatrix} = \begin{bmatrix} -2 & -2 & -2 & 1 \\ -2 & -2 & 0 & 1 \\ 0 & -2 & 0 & 1 \\ 0 & -2 & -2 & 1 \\ -2 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & -2 & 1 \\ -2 & 0 & -2 & 1 \end{bmatrix}$$

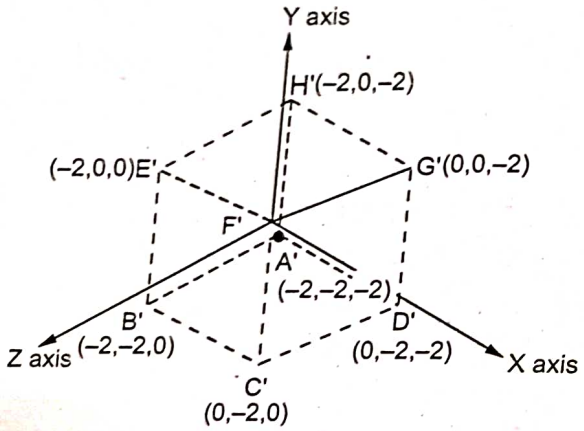


Fig. Q.16.2 Final state

END...

Unit IV

8 Colour Models

8.1 : Properties of Light

Q.1 Discuss the properties of light.

[SPPU : Dec.-17, 19, May-18, Marks 2, June-22, Marks 5]

Ans. :

- When this light is incident upon an object, some frequencies are absorbed and some are reflected by the object.
- The combination of reflected frequencies decides the colour of the object.
- The dominant frequency decides the colour of the object. Due to this reason, dominant frequency is also called the **hue** or simply the **colour.**
- The brightness refers to the intensity of the perceived light.
- The saturation describes the purity of the colour.
- Pastels and pale colours are described as less pure or less saturated.
- When the two properties purity and dominant frequency are used collectively to describe the colour characteristics, are referred to as **chromaticity.**
- When two colour sources are combined to produce white colour, they are referred to as **complementary colours.**
- Red and cyan, green and magenta, blue and yellow are complementary colour pairs.
- Usually, the colour model use combination of three colours to produce wide range of colours, called the **colour gamut** for that model.
- The basic colours used to produce colour gamut in particular model are called **primary colours.**

8.2 : CIE Chromaticity Diagram

Q.2 Explain CIE chromaticity diagram.

☞ [SPPU : May-09, 12, 13, 17, 18, Dec.-18, Marks 4]

Ans. : • Matching and therefore defining a coloured light with a combination of three fixed primary colours is desirable approach to specify colour.

- In 1931, the Commission Internationale de I´Eclairage (CIE) defined three standard primaries, called X, Y and Z to replace red, green and blue.
- Here, X, Y and Z represent vectors in a three-dimensional, additive colour space.
- The three standard primaries are imaginary colours. They are defined mathematically with positive colour-matching functions, as shown in Fig. Q.2.1. They specify the amount of each primary needed to describe any spectral colour.

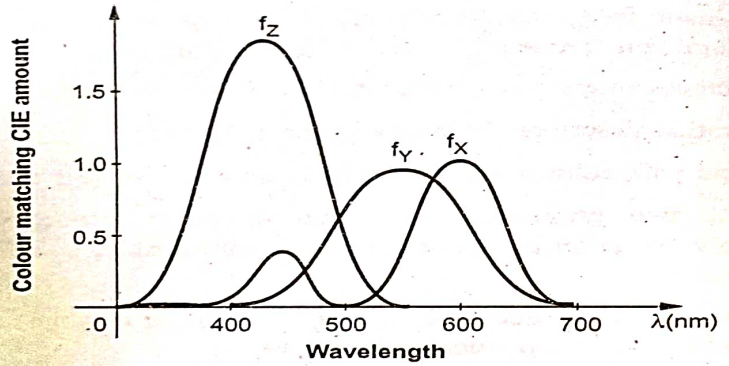


Fig. Q.2.1 Amounts of CIE primaries needed to display spectral colours

- The advantage of using CIE primaries is that they eliminate matching of negative colour values and other problems associated with selecting a set of real primaries.
- Any colour (C_λ) using CIE primaries can be expressed as,

$$C_\lambda = X\,X + Y\,Y + Z\,Z$$

where X, Y and Z are the amounts of the standard primaries needed to match C_λ and X, Y and Z represent vectors in a three-dimensional, additive colour space.

- With above expression we can define chromaticity values by normalizing against luminance (X + Y + Z). The normalizing amounts can be given as,

$$x = \frac{X}{X+Y+Z}, \quad y = \frac{Y}{X+Y+Z}, \quad z = \frac{Z}{X+Y+Z}$$

- Notice that x + y + z = 1. That is, x, y and z are on the (X + Y + Z = 1) plane. The complete description of colour is typically given with the three values x, y and z. The remaining values can be calculated as follows :

$$z = 1 - x - y, \qquad X = \frac{x}{y}Y, \qquad Z = \frac{z}{y}Y$$

- Chromaticity values depend only on dominant wavelength and saturation and are independent of the amount of luminous energy.
- By plotting x and y for all visible colours, we obtain the CIE chromaticity diagram shown in Fig. Q.2.2, which is the projection onto the (X, Y) plane of the (X + Y + Z = 1) plane.

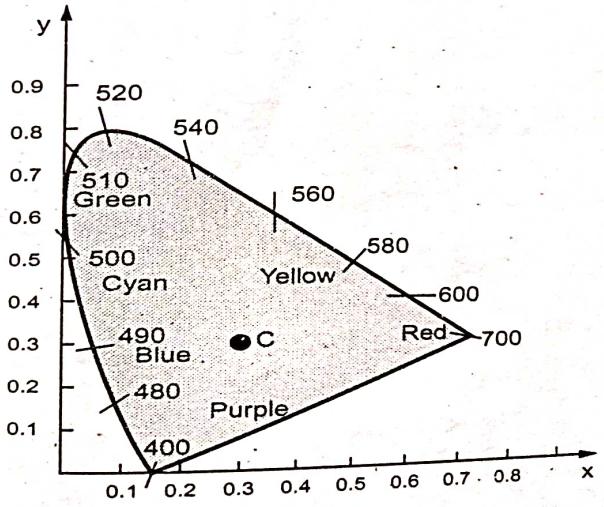


Fig. Q.2.2

- The interior and boundary of the tongue-shaped region represent all visible chromaticity values.
- The points on the boundary are the pure colours in the electromagnetic spectrum, labelled according to wavelength in nanometre from the red end to the violet end of the spectrum.
- A standard white light, is formally defined by a light source illuminant C, marked by the center dot.
- The line joining the red and violet spectral points is called the purple line, which is not the part of the spectrum.

8.3 : RGB Colour Model

Q.3 Explain RGB colour model.

☞[SPPU : Dec.-12,13,15,16, May-16,17,19, Marks 8]

Ans. : • In this model, the individual contribution of red, green and blue are added together to get the resultant colour.

• We can represent this colour model with the unit cube defined on R, G and B axes, as shown in the Fig. Q.3.1.

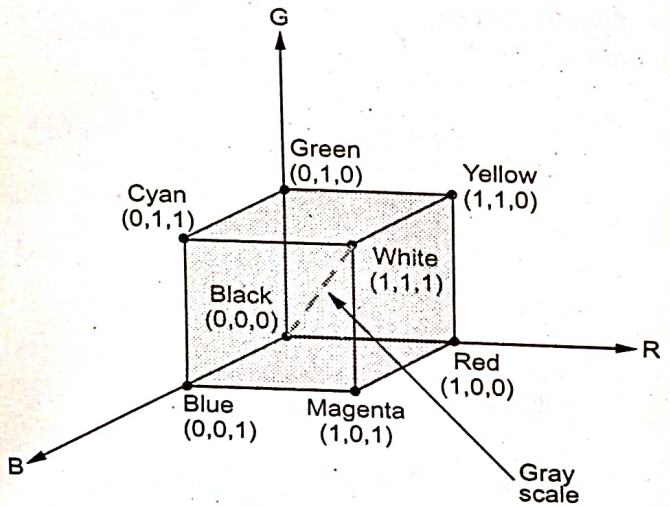


Fig. Q.3.1 The RGB cube

• The vertex of the cube on the axes represent the primary colours and the remaining vertices represent the complementary colour for each of the primary colours.

- The main diagonal of the cube, with equal amounts of each primary represents the gray levels. The end at the origin of the diagonal represents black (0, 0, 0) and other end represents white (1,1,1) .
- Each colour point within the bounds of the cube is represented as the triple (R, G, B), where value for R, G, B are assigned in the range from 0 to 1.
- As mentioned earlier, it is an additive model.
- Intensities of the primary colours are added to get the resultant colour. Thus, the resultant colour C_λ is expressed in RGB component as,

$$C_\lambda = RR + GG + BB$$

8.4 : HSV Colour Model

Q.4 Explain HSV colour model.

☞[SPPU : May-05,06,17,18,19, Dec.-05,08,10,13, Marks 4]

Ans. : • RGB and CMY models are hardware oriented model. In contrast, HSV colour model is user oriented. It uses colour descriptions that have a more intuitive appeal to a user.

- The colour specification in HSV model can be given by selecting a spectral colour and the amounts of white and black that are to be added to obtain different shades tints and tones.
- This model uses three colour parameter : Hue (H), saturation (S) and value (V).
- **Hue** distinguishes among colours such as red, green, purple and yellow.
- **Saturation** refers to how far colour is from a gray of equal intensity. For example, red is highly saturated whereas pink is relatively unsaturated. The value V indicates the level of brightness.
- This model is also known as **HSL** or **HSI.**
- HSL stands for hue, saturation and lightness.

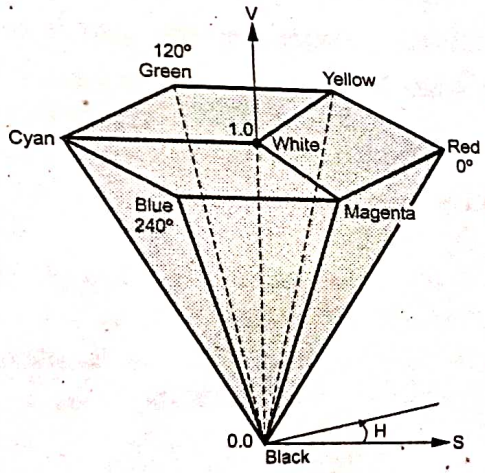


Fig. Q.4.1 (a) The HSV hexcone

- HSI stands for hue, saturation and intensity.
- The model uses cylindrical co-ordinate system and the subset of the space within which model is defined is a hexcone, or six-sided pyramid, as shown in the Fig. Q.4.1 (a).
- The top of the hexcone is derived from the RGB cube.
- If we imagine viewing the cube along the main diagonal from the white vertex to the origin (black), we see an outline of the cube that has the hexagon shape shown in Fig. Q.4.1 (b). This boundary of cube is used as a top if hexcone and it represents various hues.

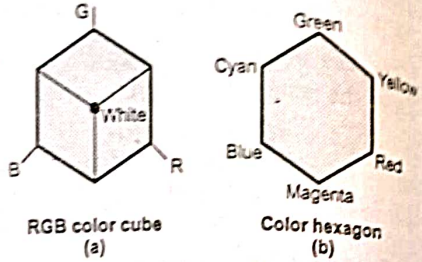


Fig. Q.4.1 (b) Top of hexcone

- Hue or H, is measured by the angle around the vertical axis, with red at 0 °, green at 120 ° and so on as shown in the Fig. Q.4.1 (a).
- Complementary colours in the HSV hexcone are 180 ° apart saturation parameter varies from 0 to 1. Its value is the ratio ranging from 0 on the centre line (V axis) to 1 on the triangular sides of the hexcone.
- The value V varies from 0 at the apex of the hexcone to 1 at the top. The apex represents black.
- At the top of the hexcone, colours have their maximum intensity.
- When V = 1 and S = 1, we have the pure hues. For example, pure red is at H = 0, V = 1 and S = 1, pure green is at H = 120, V = 1 and S = 1, pure blue is at H = 240, V = 1 and S = 1 and so on.
- The required colour can be obtained by adding either white or black to the pure hue.
- Black can be added to the selected hue by decreasing the setting for V while S is held constant.
- On the other hand, white can be added to the selected hue by decreasing S while keeping V constant.

8.5 : CMY Colour Model

Q.5 Explain CMY colour model.

☞ **[SPPU : May-06, Dec.-10,13,19, Marks 4]**

Ans. :

- In this model cyan, magenta and yellow colours are used as a primary colours. This model is used for describing colour output to hard-copy devices.
- Unlike video monitor, which produce a colour pattern by combining light from the screen phosphors, hard-copy devices such as plotters produce a colour picture by coating a paper with colour pigments.
- The subset of the cartesian co-ordinate system for the CMY model is the same as that for RGB except that white (full light) instead of black (no light) is at the origin. Colours are specified by what is removed or subtracted from white light, rather than by what is added to blackness.
- Cyan can be formed by adding green and blue light. Therefore, when white light is reflected from cyan coloured ink, the reflected light does not have red component. That is, red light is absorbed or subtracted, by the ink.
- Magenta ink subtracts the green component from incident light and yellow subtracts the blue component. Therefore, cyan, magenta and yellow are said to be complements of red, green and blue respectively.
- Fig. Q.5.1 shows the cube representation for CMY model.

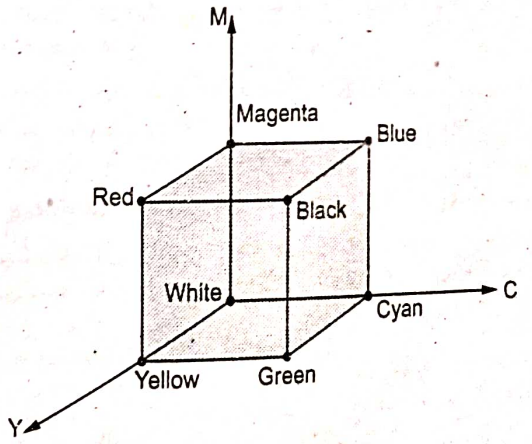


Fig. Q.5.1 The CMY cube

- As shown in the Fig. Q.5.1, point (1,1,1) represents black, because all components of the incident light are subtracted.
- The point (0, 0, 0), the origin represents white light.
- The main diagonal represents equal amount of primary colours, thus the gray colours.
- A combination of cyan and yellow produces green light, because the red and blue components of the incident light are absorbed.
- Other colour combinations are obtained by a similar subtractive process.
- It is possible to get CMY representation from RGB representation as follows,

$$\begin{bmatrix} C \\ M \\ Y \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} - \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

- The unit column vector is the RGB representation for white and the CMY representation for black. The conversion from RGB to CMY is then can be given as,

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} - \begin{bmatrix} C \\ M \\ Y \end{bmatrix}$$

END...

Unit IV

9 Illumination Models and Shading Algorithms

9.1 : Light Sources

Q.1 Explain light. **[SPPU : Dec.-14, Mark 1]**

Ans. : Light is an electromagnetic energy, reaches the eye after interacting with the physical environment.

Q.2 Describe point source illumination.

[SPPU : Dec.-07,10,15,16,17,18, May-16, Marks 4, June-22, Marks 3]

Ans. :
- A point source is a direction source, whose all the rays come from the same direction, therefore, it can be used to represent the distant sun by approximating it as an infinitely distant point source.
- The modelling of point sources requires additional work because their effect depends on the surface's orientation. If the surface is normal (perpendicular) to the incident light rays, it is brightly illuminated. The surfaces turned away from the light source (oblique surfaces) are less brightly illuminated.
- For oblique surfaces, the illumination decreases by a factor of cos I, where I is the angle between the direction of the light and the direction normal to the surface plane. The angle I is know as angle of incidence.

Q.3 What is diffuse reflection ? **[SPPU : June-22, Marks 3]**

Ans. : We know that shiny materials reflect more of the incident light, and dull surfaces absorb more of the incident light. Whereas rough or grainy surfaces tend to scatter the reflected light in all directions. This scattered light is called **diffuse reflection.**

9.2 : Ambient Light

Q.4 Write a note on ambient light.

Ans. : A surface of the object that is not exposed directly to a light source still will be visible if nearby objects are illuminated. A simple way to model the combination of light reflections from various surfaces to produce a uniform illumination called the **ambient light**, or **background light** is to set a general level of brightness for a scene. This level is defined by the parameter I_a and each surface is then illuminated with this constant value. With such illumination, the reflected light is a constant for each surface, independent of the viewing direction and the spatial orientation of the surface. However, the intensity of the reflected light for each surface, depends on the optical properties of the surface, that is; how much of the incident light energy is to be reflected and how much absorbed.

9.3 : Diffuse Illumination and Reflection

Q.5 Describe diffuse illumination.

☞[SPPU : Dec.-07,10,15,16,17,18, May-11,12,13,16, Marks 4, June-22, Marks 3]

Ans. : • An object illumination is as important as its surface properties in computing its intensity. The object may be illuminated by light which does not come from any particular source but which comes from all directions. When such illumination is uniform from all directions, the illumination is called **diffuse illumination.**

- Usually, diffuse illumination is a background light which is reflected from walls, floor and ceiling.

Q.6 Explain reflectivity. **☞[SPPU : Dec.-14, Mark 1]**

Ans. : • In practice, when object is illuminated, some part of light energy is absorbed by the surface of the object, while the rest is reflected.

- The ratio of the light reflected from the surface to the total incoming light to the surface is called **coefficient of reflection** or the **reflectivity.** It is denoted by R.

Q.7 What is Lambert's cosine law ? What is its significance ?

☞[SPPU : May-19, Marks 3]

Ans. : • The diffuse reflections from the surface are scattered with equal intensity in all directions, independent of the viewing direction. Such surfaces are sometimes referred to as **ideal diffuse reflectors.** They are also called **Lambertian reflector,** since radiated light energy from any point on the surface is governed by **Lambert's cosine law.**

- This law states that the reflection of light from a perfectly diffusing surface varies as the cosine of the angle between the normal to the surface and the direction of the reflected ray. This is illustrated in Fig. Q.7.1.

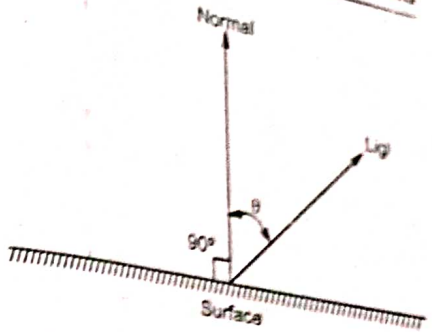


Fig. Q.7.1 The direction of light is measured from the surface normal

- Thus if the incident light from the source is perpendicular to the surface at a perpendicular point, that point is fully illuminated.
- On the other hand, as the angle of illumination moves away from the surface normal, the brightness of the point drops off; but the points appear to be squeezed closer together and the net effect is that the brightness of the surface is unchanged. This is illustrated in Fig. Q.7.2.

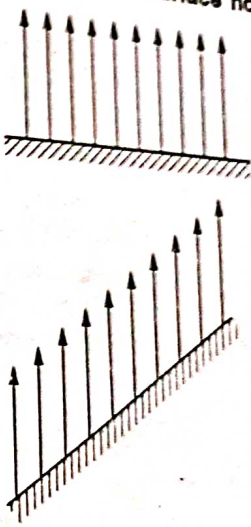


Fig. Q.7.2 Surface brightness

9.4 : Specular Reflection

Q.8 Explain the following in detail : Specular reflection.

☞[SPPU : May-13, Marks 3, June-22, Marks 6]

Ans. : • When we illuminate a shiny surface such as polished metal or an apple with a bright light, we observe highlight or bright spot on the shiny surface. This phenomenon of reflection of incident light in a concentrated region around the specular-reflection angle is called **specular-reflection.**

- Due to specular-reflection, at the highlight, the surface appears to be not in its original colour, but white, the colour of incident light.
- The Fig. Q.8.1 shows the specular-reflection direction at a point on the illuminated surface. The specular-reflection angle equals the angle of the incident light, with the two angles measured on opposite sides of the unit normal surface vector N.

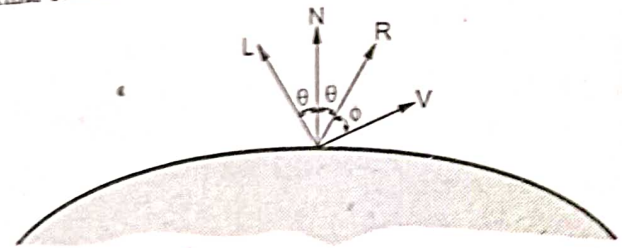


Fig. Q.8.1 Specular-reflection

- As shown in the Fig. Q.8.1, R is the unit vector in the direction of ideal specular-reflection, L is the unit vector directed towards the point light source and V is the unit vector pointing to the viewer from the surface position.
- The angle ϕ between vector R and vector V is called **viewing angle**. For an ideal reflector (perfect mirror), incident light is reflected only in the specular-reflection direction. In such case, we can see reflected light only when vector V and R coincide, i.e., $\phi = 0$.

9.5 : Shading Algorithms

Q.9 Explain shading. **[SPPU : Dec.-14, Mark 1]**

Ans. : According to light source, representing the polygon surface is known as shading.

Q.10 Explain Gouraud shading.

[SPPU : May-07,12,18,19, Dec.-08,11,12,17,18, Marks 4]

Ans. :
- In this method, the intensity interpolation technique developed by Gouraud is used, hence the name.
- The polygon surface is displayed by linearly interpolating intensity values across the surface. Here, intensity values for each polygon are matched with the values of adjacent polygons along the common edges. This eliminates the intensity discontinuities that can occur in flat shading.

- By performing following calculations we can display polygon surface with Gouraud shading.
  1. Determine the average unit normal vector at each polygon vertex.
  2. Apply an illumination model to each polygon vertex to determine the vertex intensity.
  3. Linearly interpolate the vertex intensities over the surface of the polygon.
- We can obtain a normal vector at each polygon vertex by averaging the surface normals of all polygons sharing that vertex. This is illustrated in Fig. Q.10.1.

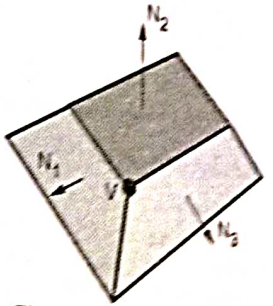


Fig. Q.10.1 Calculation of normal

- As shown in the Fig. Q.10.1, there are three surface normals N_1, N_2 and N_3 of polygon sharing vertex V. Therefore, normal vector at vertex V is given as

$$N_V = \frac{N_1 + N_2 + N_3}{|N_1 + N_2 + N_3|}$$

- In general, for any vertex position V, we can obtain the unit vertex normal by equation,

$$N_V = \frac{\sum_{i=1}^{n} N_i}{\left|\sum_{i=1}^{n} N_i\right|}$$

where n is the number of surface normals of polygons sharing that vertex.

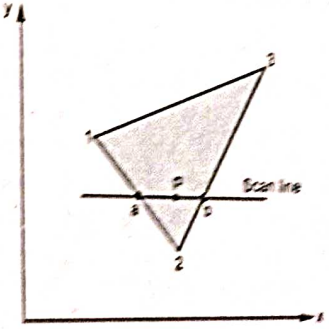


Fig. Q.10.2

- The next step in Gouraud shading is to find vertex intensities. Once we have the vertex normals, their vertex intensities can be determined by applying illumination model to each polygon vertex.
- Finally, each polygon is shaded by linear interpolating of vertex intensities along each edge and then between edges along each scan line. This is illustrated in Fig. Q.10.2.

Q.11 Explain Phong's shading.

☞[SPPU : May-07,10,15,18,19, Dec.-17,18, Marks 4]

Ans. : • Phong shading, also known as **normal-vector interpolation** shading, interpolates the surface normal vector N, instead of the intensity.

• By performing following steps we can display polygon surface using Phong shading.

1. Determine the average unit normal vector at each polygon vertex.
2. Linearly interpolate the vertex normals over the surface of the polygon.
3. Apply an illumination model along each scan line to determine projected pixel intensities for the surface points.

• The first steps in the Phong shading is same as first step in the Gouraud shading.

• In the second step the vertex normals are linearly interpolated over the surface of the polygon. This is illustrated in Fig. Q.11.1. As shown in the Fig. Q.11.1, the normal vector N for the scan line intersection point along the edge between vertices 1 and 2 can be obtained by vertically interpolating between edge endpoint normals :

$$N = \frac{y - y_2}{y_1 - y_2} N_1 + \frac{y_1 - y}{y_1 - y_2} N_2$$

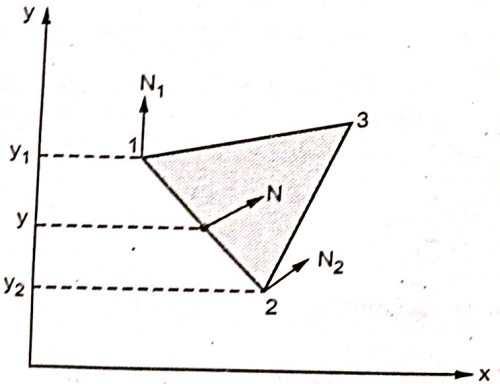


Fig. Q.11.1 Calculation of interpolation of surface normals along a polygon edge

• Like, Gouraud shading, here also we can use incremental methods to evaluate normals between scan lines and along each individual scan line.

• Once the surface normals are evaluated, the surface intensity at that point is determined by applying the illumination model.

Q.12 Explain the difference between Gouraud shading and phong shading.

☞[SPPU : Dec.-19, Marks 6]

Ans. :

Sr. No.	Gouraud shading	Phong's shading
1.	In this shading model the polygon surface is displayed by linearly interpolating intensity values across the surface.	In this shading model, the surface normal vector N is interpolated, instead of intensity.
2.	It suffers from mach - band effect.	It greatly reduces the mach - band effect.
3.	It requires less calculations than Phong's shading.	It requires more calculations, increasing the cost of shading.

Q.13 Explain halftone shading. **☞[SPPU : May-17, Marks 4]**

Ans. : • Many displays and hardcopy devices are bilevel. They can only produce two intensity levels. In such displays or hardcopy devices, we can create an apparent increase in the number of available intensities. This is achieved by incorporating multiple pixels positions into the display of each intensity value.

• When we view a very small area from a sufficiently large viewing distance, our eyes average fine details within the small area and record only the overall intensity of the area. This phenomenon of apparent increase in the number of available intensities by considering combine intensity of multiple pixels is known as **halftoning**.

• The halftoning is commonly used in printing black and white photographs in newspapers, magazines and books. The pictures produced by halftoning process are called **halftones**.

• In computer graphics, halftone reproductions are approximated using rectangular pixel regions, say 2×2 pixels or 3×3 pixels. These regions are called **halftone patterns** or **pixel patterns**. Fig. Q.13.1 shows the halftone patterns to create number of intensity levels.

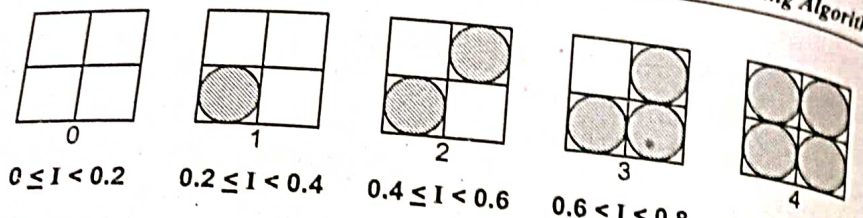
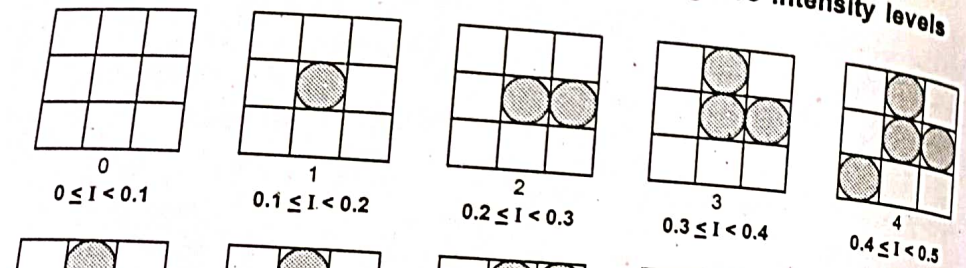


Fig. Q.13.1 (a) 2×2 pixel patterns for creating five intensity levels



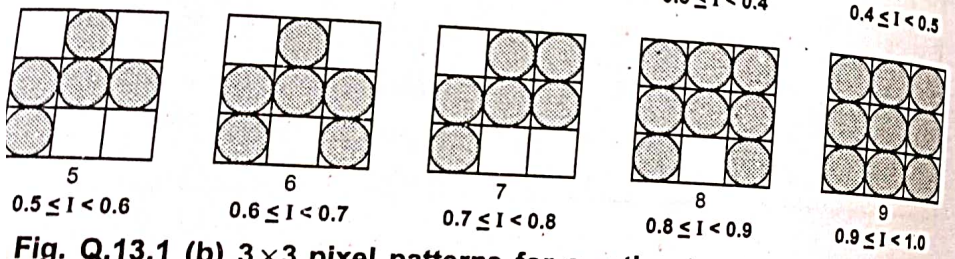


Fig. Q.13.1 (b) 3×3 pixel patterns for creating ten intensity levels

END...

Unit IV

10 Hidden Surfaces

10.1 : Introduction

Q.1 Why are hidden surface algorithms needed ?

☞ **[SPPU : Dec.-06,08,11,13, May-07,08,12, June-22, Marks 2]**

Ans. : Hidden surface algorithms are needed to determine which lines or surfaces of the objects are visible, so that we can display only the visible lines or surfaces.

Q.2 Explain, how hidden lines and surfaces are removed ?

☞ **[SPPU : May-11, Marks 8]**

Ans. : There are two approaches to remove hidden lines and surfaces are called **object-space methods** and **image-space methods**, respectively.

Object-Space Method

- Object-space method is implemented in the **physical co-ordinate system** in which objects are described.
- It compares objects and parts of objects to each other within the scene definition to determine which surfaces, as a whole, we should label as visible. Object-space methods are generally used in **line-display algorithms.**

Image-Space Method

- Image space method is implemented in the **screen co-ordinate system** in which the objects are viewed.
- In an image-space algorithm, visibility is decided point by point at each pixel position on the view plane.
- Most **hidden line/surface algorithms** use image-space methods.

10.2 : Back Face Detection and Removal Algorithm

Q.3 Explain backface removal algorithm.

[SPPU : May-05,06,08,12,13,18, Dec.-07,11,12,13,16,19, Marks 8]

Ans. : • If a polygon is visible, the light surface should face towards us and the dark surface should face away from us.

• The direction of the light face can be identified by examining the result

$$N \cdot V > 0$$

where

N : Normal vector to the polygon surface with Cartesian components (A, B, C).

V : A vector in the viewing direction from the eye (or "camera") position (Refer Fig. Q.3.1)

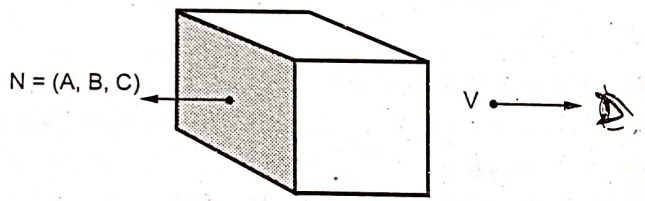


Fig. Q.3.1

• We know that, the dot product of two vectors, gives the product of the lengths of the two vectors times the cosine of the angle between them. This cosine factor is important to us because if the vectors are in the same direction ($0 \le \theta < \pi/2$), then the cosine is positive and the overall dot product is positive. However, if the directions are opposite ($\pi/2 < \theta \le \pi$), then the cosine and the overall dot product is negative.

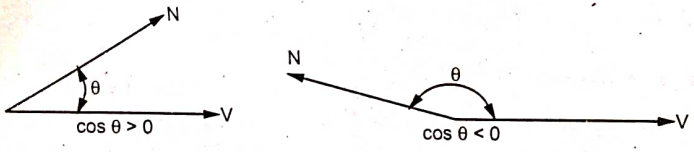


Fig. Q.3.2 Cosine angles between two vectors

• If the dot product is positive, we can say that the polygon faces towards the viewer; otherwise it faces away and should be removed.

• In case, if object description has been converted to projection co-ordinates and our viewing direction is parallel to the viewing z_v axis, then $V = (0, 0, V_z)$ and

$$V \cdot N = V_z C$$

• So that we only have to consider the sign of C, the z component of the normal vector N. Now, if the z component is positive, then the polygon faces towards the viewer, if negative, it faces away.

10.3 : Z-Buffer Algorithm

Q.4 Explain Z-buffer algorithm.

[SPPU : May-10,11,14,16,17,18,19, Dec.-11,12,13,16,18, Marks 8; June-22, Marks 6]

Ans. : Z-buffer Algorithm

1. Initialize the Z-buffer and frame buffer so that for all buffer positions
   Z-buffer (x, y) = 0 and frame-buffer (x, y) = $I_{background}$
2. During scan conversion process, for each position on each polygon surface, compare depth values to previously stored values in the depth buffer to determine visibility.
   Calculate z-value for each (x, y) position on the polygon.
   If z > Z-buffer (x, y), then set
   Z-buffer (x, y) = z, frame-buffer (x, y) = $I_{surface}$ (x, y)
3. Stop

• Note that, $I_{background}$ is the value for the background intensity and $I_{surface}$ is the projected intensity value for the surface at pixel position (x, y). After processing of all surfaces, the Z-buffer contains depth values for the visible surfaces and the frame buffer contains the corresponding intensity values for those surfaces.

• To calculate z-values, the plane equation

$$Ax + By + Cz + D = 0$$

is used where (x, y, z) is any point on the plane and the coefficient A, B, C and D are constants describing the spatial properties of the plane. (Refer Appendix A for details)

• Therefore, we can write

$$z = \frac{-Ax - By - D}{C}$$

- Note, if at (x, y) the above equation evaluates to z_1, then at (x + Δx, y) the value of z, is

$$z_1 = \frac{A}{C} (\Delta x)$$

- Only one subtraction is needed to calculate z(x + 1, y), given z(x, y), since the quotient A/C is constant and Δx = 1. A similar incremental calculation can be performed to determine the first value of z on the next scan line, decrementing by B/C for each Δy.

10.4 : Painter's (Depth Sort) Algorithm

Q.5 Explain Painter's algorithm.

☞ **[SPPU : Dec.-07,08,11,14,15,16,18,19, May-09,10,11,12,16,17,18,19, Marks 8, June-22, Marks 6]**

Ans. : Painter's Algorithm :

1. Sort all polygons in order of decreasing depth.
2. Determine all polygons Q (preceding P) in the polygon list whose z-extents overlap that of P.
3. Perform test 2 through 6 for each Q

a) If every Q passes the tests, scan convert the polygon P.

b) If test fails for some Q, swap P and Q in the list and make the indication that Q is swapped. If Q has already been swapped, use the plane containing polygon P to divide polygon Q into two polygons, Q_1 and Q_2. Replace Q with Q_1 and Q_2. Repeat step 3.

10.5 : Warnock's (Area Subdivision) Algorithm

Q.6 Explain Warnock's algorithm.

☞ **[SPPU : Dec.-05,07,10,14,16,19, May-06,07,17,18,19, Marks 8]**

Ans. : Algorithm

1. Initialize the area to be the whole screen.
2. Create the list of polygons by sorting them with their z-values of vertices. Don't include disjoint polygons in the list because they are not visible.
3. Find the relationship of each polygon.

4. Perform the visibility decision test.

a) If all the polygons are disjoint from the area, then fill area with background colour.

b) If there is only one intersecting or only one contained polygon then first fill entire area with background colour and then fill the part of the polygon contained in the area with the colour of polygon.

c) If there is a single surrounding polygon, but no intersecting or contained polygons, then fill the area with the colour of the surrounding polygon.

d) If surrounding polygon is closer to the viewpoint than all other polygons, so that all other polygons are hidden by it, fill the area with the colour of the surrounding polygon.

e) If the **area is the pixel** (x, y) and neither a, b, c, nor d applies, compute the z co-ordinate at pixel (x, y) of all polygons in the list. The pixel is then set to colour of the polygon which is closer to the viewpoint.

5. If none of the above tests are true then subdivide the area and go to step 2.

Q.7 What are the advantages of Warnock's algorithm.

☞ **[SPPU : May-15,18, Marks 3]**

Ans. : Advantages

1. It follows the divide-and-conquer strategy, therefore, parallel computers can be used to speed up the process.
2. Extra memory buffer is not required.

Q.8 Why Warnock's algorithm is also called as area subdivision algorithm ? ☞ **[SPPU : Dec.-10, Marks 4]**

Ans. : Warnock's algorithm sub divides each area into four equal squares and at each stage in the recursive-subdivision process, the relationship between projection of each polygon and the area of interest is checked for four possible relationships. Because of this Warnock's algorithm is also called as area subdivision algorithm.

END...

Unit V

11 Curves and Fractals

11.1 : Curve Generation

Q.1 What is true curve generation ? Write a Pseudo code to implement DDA arc generation.

☞ **[SPPU : May-06, Dec.-07,10,11, Marks 10]**

OR Write short note on True curve generation.

☞ **[SPPU : Dec.-10, Marks 2]**

Ans. : In a true curve generation algorithm, rather than approximating a curve by small line segments, we use digital differential analyzer algorithm to determine points those lie on the curve.

Algorithm / Pseudo Code

Algorithm

1. Read the centre of a curvature, say (x_0, y_0).
2. Read the arc angle, say θ.
3. Read the starting point of the arc, say (x, y).
4. Calculate $d\theta$.

   $d\theta = \text{Min}\ (0.01,\ 1 / (3.2 \times (|x - x_0| + |y - y_0|)))$
5. Initialize Angle = 0.
6. While (Angle < θ)

   do

   { Plot (x, y)

   $x = x - (y - y_0) \times d\theta$

   $y = y + (x - x_0) \times d\theta$

   Angle = Angle + $d\theta$

   }
7. Stop.

Problems in True-Curve Generation Approach

1. To specify a curve, we need more information than just its end points.
2. It is difficult to apply transformations. For example, a circle when scaled in only one direction becomes an ellipse. If our algorithm supports only circular arc generation then ability to scale pictures is limited.
3. New clipping algorithm is required to clip arcs.
4. The curve generation algorithms for curves other than circular or elliptical such as airplane wings or cars or human faces, are complex.

11.2 : Interpolation

Q.2 Write a short note on blending functions.

☞ **[SPPU : Dec.-13, Marks 4]**

Ans. :

- Suppose we want a polynomial curve that will pass through n sample points.

  $(x_1, y_1, z_1), (x_2, y_2, z_2), \ldots, (x_n, y_n, z_n)$

- We will construct the function as the sum of terms, one term for each sample point. These functions can be given as

$$f_x(u) = \sum_{i=1}^{n} x_i B_i(u)$$

$$f_y(u) = \sum_{i=1}^{n} y_i B_i(u)$$

$$f_z(u) = \sum_{i=1}^{n} z_i B_i(u)$$

- The function $B_i(u)$ is called **'blending function'**. For each value of parameter u, the blending function determines how much the i^{th} sample point affects the position of the curve.
- In other words we can say that each sample points tries to pull the curve in its direction and the function $B_i(u)$ gives the strength of the pull.

- If for some value of u, $B_i(u) = 1$ for unique value of i (i.e. $B_i(u) = 0$ for other values of i) then i^{th} sample point has complete control of the curve and the curve will pass through i^{th} sample point.
- For different value of u, some other sample point may have complete control of the curve. In such case the curve will pass through that point as well.
- In general, the blending functions give control of the curve to each of the sample points inturn for different values of u.

Q.3 What is interpolation ? Explain Lagrange interpolation method.

☞[SPPU : May-07,08,09,16, Dec.-05,10,11,12,13,16,18, Marks 8, June-22, Marks 6]

Ans. : In a curve generation, process of determining the intermediate points between the known sample points is acheived using interpolation.

The blending functions give control of the curve to each of the sample points inturn for different values of u.

- Let us assume that the first sample point (x_1, y_1, z_1) has complete control when $u = -1$, the second when $u = 0$, the third when $u = 1$, and so on. i.e.
  - When $u = -1 \Rightarrow B_1(u) = 1$ and 0 for $u = 0, 1, 2, \ldots, n-2$
  - When $u = 0 \Rightarrow B_2(u) = 1$ and 0 for $u = -1, 1, \ldots, n-2$
  - : : :
  - : : :
  - When $u = (n-2) \Rightarrow B_n(u) = 1$ and 0 for $u = -1, 0, \ldots, (n-1)$
- To get $B_1(u) = 1$ at $u = -1$ and 0 for $u = 0, 1, 2, \ldots, n-2$, the expression for $B_1(u)$ can be given as

$$B_1(u) = \frac{u\,(u-1)\,(u-2)\,\ldots\,[u-(n-2)]}{(-1)\,(-2)\,\ldots\,(1-n)}$$

where denominator term is a constant used. In general form i^{th} blending function which is 1 at $u = i-2$ and 0 for other integers can be given as :

$$B_i(u) = \frac{(u+1)\,(u)\,(u-1)\ldots[u-(i-3)]\,[u-(i-1)]\,\ldots\,[u-(i-2)]}{(i-1)\,(i-2)\,(i-3)\,\ldots\,(1)\,(-1)\ldots(i-n)}$$

- The approximation of the curve using above expression is called **Lagrange interpolation.**

11.3 : Spline Representation

Q.4 Explain the term control points and order of continuity in curve drawing.

☞[SPPU : May-11, 12, Marks 8]

Ans. :

- We can specify a spline curve by giving a set of co-ordinate positions, called **control points**, which indicates the general shape of the curve.
- When polynomial sections are fitted so that the curve passes through all control points, as shown in the Fig. Q.4.1, the resulting curve is said to **interpolate** the set of control points.

Interpolation spline

Fig. Q.4.1

- To ensure a smooth transition from one section of a piecewise parametric curve to the next, we can impose various continuity conditions at the connection points. We see **parametric continuity** and **geometric continuity** conditions.
- In geometric continuity we require parametric derivatives of two sections to be proportional to each other at their common boundary instead of equal to each other.
- Parametric continuity is set by matching the parametric derivatives of adjoining two curve sections at their common boundary.
- In zero order parametric continuity, given as C^0, it means simply the curve meet and same is for zero order geometric continuity.
- In first order parametric continuity called as C^1 means that first parametric derivatives of the coordinate functions for two successive curve sections are equal at the joining point and geometric first order continuity means the parametric first derivative are proportional at the intersection of two successive sections.

- Second order parametric continuity or C^2 continuity means that both the first and second parametric derivatives of the two curve sections are same at the intersection and for second order geometric continuity or C^2 continuity means that both the first and second parametric derivatives of the two curve sections are proportional at their boundary. Under C^2 continuity curvature of the two curve sections match at the joining positions.

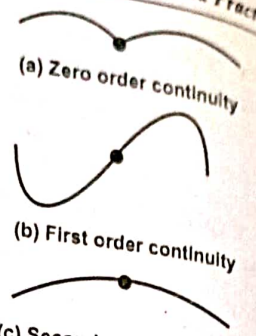


Fig. Q.4.2

Q.5 What is convex hull ?

Ans. :

- The convex polygon boundary that encloses a set of control points is called the **convex hull**. One way to envision the shape of a convex hull is to imagine a rubber band stretched around the positions of the control points so that each control point is either on the perimeter of the hull or inside it. Convex hulls provide a measure for the deviation of a curve or surface from the region bounding the control points. Some splines are bounded by the convex hull, thus ensuring that the polynomials smoothly follow the control points without erratic oscillations. Also, the polygon region inside the convex hull is useful in some algorithms as a clipping region.

Q.6 Why is cubic form chosen for representing curve ?

☞[SPPU : Dec.-10, Marks 6]

Ans. :

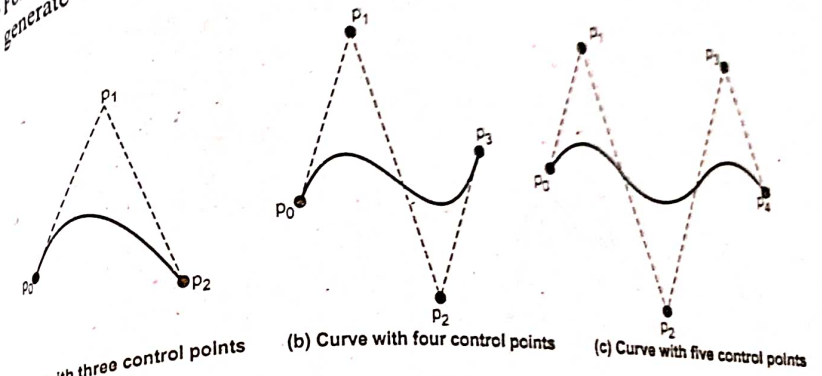
- Polylines and polygons are first-degree, piecewise linear approximation to curves and surfaces, respectively. But this lower degree polynomials give too little flexibility in controlling the shape of the curve.
- The higher-degree polynomials give reasonable design flexibility, but introduce unwanted wiggles and also require more computation.
- For this reason the third-degree polynomials are most often used for representation of curves. These polynomials are commonly known as **cubic polynomials.**

11.4 : Bezier Curves

Q.7 Explain the Bezier curve. ☞[SPPU : Dec.-05,12,13,14,15,18, May-07,10,11,13,14,18,19, June-22, Marks 6]

Ans. :

- In general, a Bezier curve section can be fitted to any number of control points. However, as number of control points increases, the degree of the Bezier polynomial also increases. Because in a Bezier curve degree of a polynomial is one less than the number of control points used.
- For example, three control points generate a parabola, four points generate cubic curve and so on. This is illustrated in Fig. Q.7.1.



(a) Curve with three control points (b) Curve with four control points (c) Curve with five control points

Fig. Q.7.1

- The Bezier curves can be specified with boundary conditions, with a characterizing matrix or with blending functions. Out of these, blending function specification is the most convenient way for general Bezier curves.
- Consider that the curve has n + 1 control points : $p_k = (x_k, y_k, z_k)$ ... where k varies from 0 to n. The co-ordinates of these control points can be blended to produce position vector P(u), which gives the path of an approximating Bezier polynomial function between p_0 and p_n. The position vector can be given by,

$$P(u) = \sum_{k=0}^{n} p_k \, BEZ_{k,n}(u), \quad 0 \le u \le 1 \qquad \ldots (1)$$

- The Bezier blending functions $BEZ_{k,n}(u)$ are the Bernstein polynomials. They are specified as,

$BEZ_{k,n}(u) = C(n,k)\,u^k(1-u)^{n-k}$ ... (2)

where the C (n ,k) are the binomial coefficients. Binomial coefficients are given by,

$$C(n,k) = \frac{n!}{k!\,(n-k)!}$$

- Equivalently, we can define Bezier blending functions with the recursive calculation as,

$$BEZ_{k,n}(u) = (1-u)\,BEZ_{k,n-1}(u) + uBEZ_{k-1,n-1}(u), \quad n > k \geq 1 \quad \ldots (3)$$

with $BEZ_{k,k} = u^k$ and $BEZ_{0,k} = (1-u)^k$. The position vector equation represents a set of three parametric equations for the individual curve co-ordinates :

$$x(u) = \sum_{k=0}^{n} x_k BEZ_{k,n}(u)$$

$$y(u) = \sum_{k=0}^{n} y_k BEZ_{k,n}(u)$$

$$z(u) = \sum_{k=0}^{n} z_k BEZ_{k,n}(u)$$

- The successive binomial coefficients can be calculated as

$$C(n,k) = \frac{n-k+1}{k} C(n,k-1) \quad \ldots (4)$$

$$\ldots (5)$$

Q.8 What are the properties of Bezier curve ?

[SPPU : May-05,06,14,17,19, Dec.-08,14,15]

Ans. : Properties of Bezier curve

1. The basis functions are real.
2. Bezier curve always passes through the first and last control points i.e. curve has same end points as the guiding polygon.
3. The degree of the polynomial defining the curve segment is one less than the number of defining polygon point. Therefore, for 4 control points, the degree of the polynomial is three, i.e. cubic polynomial.
4. The curve generally follows the shape of the defining polygon.

5. The direction of the tangent vector at the end points is the same as that of the vector determined by first and last segments.
6. The curve lies entirely within the convex hull formed by four control points.
7. The convex hull property for a Bezier curve ensures that the polynomial smoothly follows the control points.
8. The curve exhibits the variation diminishing property. This means that the curve does not oscillate about any straight line more often than the defining polygon.
9. The curve is invariant under an affine transformation.

11.5 : B-Spline Curve

Q.9 Write short note on B-splines.

[SPPU : May-10,11,13,15,17,18,19, Dec.-12,13,17, Marks 8]

OR Explain B-splines for curve generation.

[SPPU : May-16, Dec.-16, Marks 4]

Ans. : • There is another basis function, called the **B-spline** basis, which contains the Bernstein basis as a special case.

- The B-spline basis is nonglobal. It is nonglobal because each vertex B_i is associated with a unique basis function. Thus, each vertex affects the shape of the curve only over a range of parameter values where its associated basis function is nonzero.
- The B-spline basis also allows the order of the basis function and hence the degree of the resulting curve is independent on the number of vertices.
- It is possible to change the degree of the resulting curve without changing the number of vertices (control points) of the defining polygon.
- If P(u) be the position vectors along the curve as a function of the parameter u, a B-spline curve is given by

$$P(u) = \sum_{i=1}^{n+1} B_i N_{i,k}(u) \quad u_{min} \leq u < u_{max}, \; 2 \leq k \leq n+1 \quad \ldots (1)$$

where the B_i are the position vectors of the $n + 1$ defining polygon vertices and the $N_{i,k}$ are the normalized B-spline basis functions. For the i^{th} normalized B-spline basis function of order k, the basis function $N_{i,k}(u)$ are defined as

$$N_{i,1}(u) = \begin{cases} 1 & \text{if } x_i \le u < x_{i+1} \\ 0 & \text{Otherwise} \end{cases}$$

and $$N_{i,k}(u) = \frac{(u - x_1) N_{i,k-1}(u)}{x_{i+k-1} - x_i} + \frac{(x_{i+k} - u) N_{i+1,k-1}(u)}{x_{i+k} - x_{i+1}} \quad ...(2)$$

- The values of x_i are the elements of a knot vector satisfying the relation $x_i \le x_{i+1}$. The parameter u varies from u_{min} to u_{max} along the curve P(u).
- The choice of knot vector has a significant influence on the B-spline basis functions $N_{i,k}(u)$ and hence on the resulting B-spline curve.

Q.10 Discuss the properties of B-spline curve.

☞ **[SPPU : May-06, Marks 5]**

Ans. : Properties of B-spline curve

- The sum of the B-spline basis functions for any parameter value u is 1.

$$i.e. \sum_{i=1}^{n+1} N_{i,k}(u) \equiv 1$$

- Each basis function is positive or zero for all parameter values, i.e., $N_{i,k} \ge 0$.
- Except for $k = 1$ each basis function has precisely one maximum value.
- The maximum order of the curve is equal to the number of control points of defining polygon.
- The curve exhibits the variation diminishing property. Thus the curve does not oscillate about any straight line move often than its defining polygon.

The curve generally follows the shape of defining polygon.

Any affine transformation can be applied to the curve by applying it to he vertices of defining polygon.

he curve line within the convex hull of its defining polygon.

2.11 State advantages of B-spline curve over Bezier for generating curve.

☞ **[SPPU : Dec.-06, Marks 4, May-18, Marks 3]**

Ans. : Advantages of B-splines over Bezier curves

The degree of B-spline polynomial is independent on the number of control points of defining polygon (with certain limitations).

B-spline allows local control over the curve surface because each control point affects the shape of a curve only over a range of parameter values where its associated basis function is nonzero.

2.12 Give comparison between Bezier and B-spline curves.

☞ **[SPPU : Dec.-19, Marks 6]**

Ans. :

Sr. No.	Bezier curve	B-spline curve
1.	The degree of the polynomial defining the curve segment is one less that the number of defining polygon point (Control points). Therefore, for 4 control points, the degree of the polynomial is 3, i.e. cubic polynomial.	The degree of B-spline polynomial is independent on the number of vertices of defining polygon.
2.	Bezier curve can't be forced to interpolate any of its n control points without repeating it.	B-splines can be forced to interpolate any of its n control points without repeating it.
3.	Bezier curve requires less computation.	B-spline curve requires more computation.
4.	Bezier curves are less flexible.	B-Spline curve offers more control and flexibility.
5.	Bezier curves require less information.	B-Spline curves require more information such as degree of the curve and a knot vector.

6.	In Bezier curves, it is not possible to use lower degree curves and still maintain a large number of control points.	In B-Spline curves, it is possible to use lower degree curves and still maintain a large number of control points.

11.6 : Fractals

Q.13 Define Fractals.

OR Write short note on Fractals.

☞[SPPU : Dec.-12,13,15,16,18, May-13,14,17,19, Marks 6]

Ans. : Rough, jagged and random surfaces are called **fractals.**

Classification of Fractals

- The fractals can be classified as
  - Self similar
  - Self affine and
  - Invariant

Self similar fractals

- These fractals have parts those are scaled-down versions of the entire object.
- In these fractals object subparts are constructed by applying a scaling parameter s to the overall initial shape. It is a choice of user to use the same scaling factor s for all subparts, or use different scaling factors for different scaled-down parts of the object.
- Another sub class of self similar fractals is a statistically self-similar fractals, in which user can also apply random variations to the scaled-down subparts. These fractals are commonly used to model trees, shrubs, and other plants.

Self-affine fractals

- These fractals have parts those are formed with different scaling parameters, s_x, s_y, s_z in different co-ordinate directions.
- In these fractals, we can also apply random variations to obtain statistically self-affine fractals.
- These fractals are commonly used to model water, clouds and terrain.

Invarient fractals

- In these fractals, **nonlinear transformation** is used.
- It includes **self squaring fractals** such as the Mandelbrot set, which are formed with squaring functions in complex space, and self inverse fractals, form with inversion procedures.

Q.14 What do you mean by topological and fractal dimension ?

☞[SPPU : May-05, 06, 08, 09, Dec.-05, 07, 10, Marks 6]

Ans. : Consider an object composed of elastic or clay. If the object can be deformed into a line or line segment we assign its dimension $D_t = 1$. If object deforms into a plane or half plane or disk we assign its dimension $D_t = 2$. If object deforms into all space or half space or sphere, we assign its dimension $D_t = 3$. The dimension D_t is referred to as the **topological dimension.**

Fractal Dimension

- It is the second measure of an object dimension. Imagine that a line segment of length L is divided into N identical pieces. The length of each line segment l can be given as

$$l = \frac{L}{N}$$

- The ratio of length of original line segment and the length of each part of the line segment is referred to as scaling factor and is given as

$$s = \frac{L}{l}$$

- From above two equations we can write

$$N = s$$

i.e. $$N = s^1$$

- In other words we can say that if we scale a line segment by a factor 1/s then we have to add N pieces together to get the original line segment. If we scale square object by a factor 1/s we will get a small square. In case of s = 2, we require 4 pieces of square to get original square. In general we can write

$$N = s^2$$

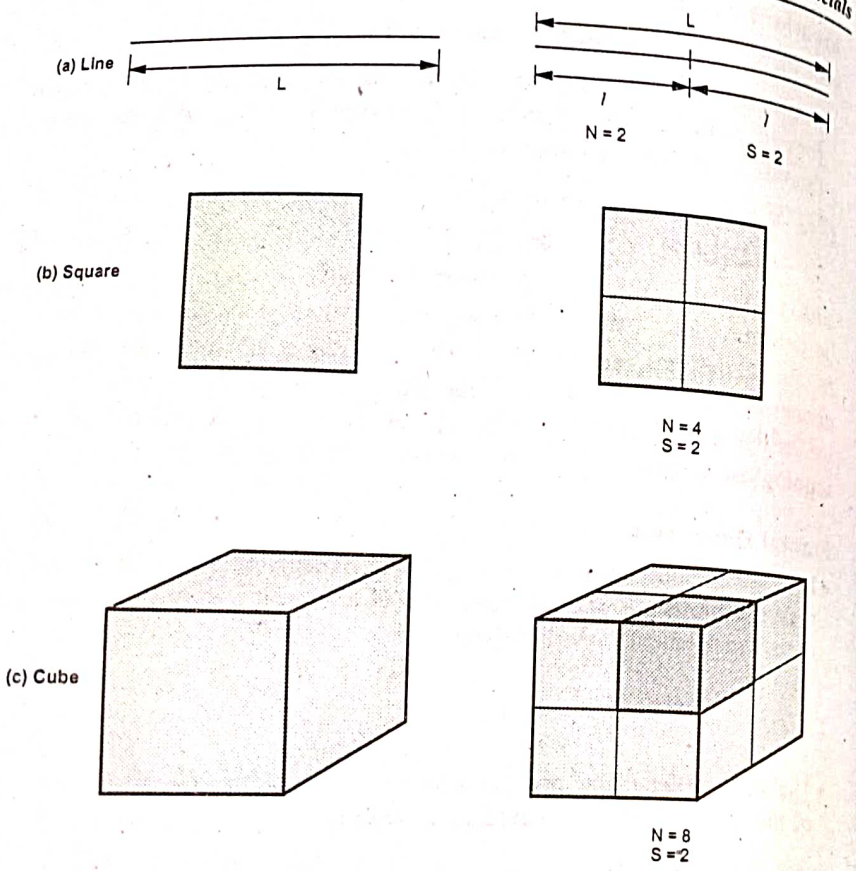


Fig. Q.14.1 Scaling of objects in various dimensions

- Similarly for cubical object we have, (Refer Fig. Q.14.1)

$$N = s^3$$

- We have seen that we can specify the dimension of the object by variable D. Here the exponent of s is a measure of object dimension. Thus we can write

$$N = s^D$$

Solving for D we get

$$D = \log N / \log s$$

- This D is called **fractal dimension.**

Q.15 Explain Hilbert's curve and give its fractal dimension.

[SPPU : May-06,15,17,18,19, June-22, Dec.-08,10,15,17,18, Marks 6]

Ans. : • The Hilbert's curve can be constructed by following successive approximations. If a square is divided into four quadrants we can draw the first approximation to the Hilbert's curve by connecting centre points of each quadrant as shown in Fig. Q.15.1.

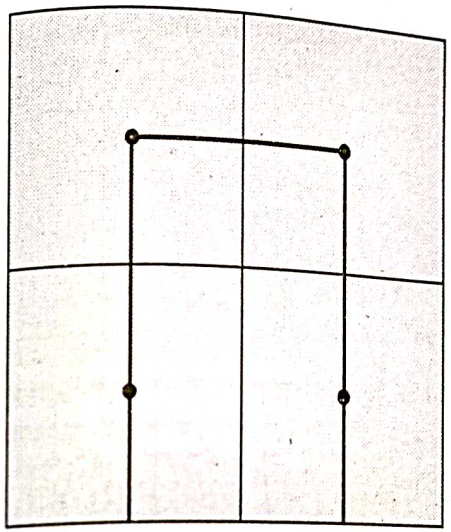


Fig. Q.15.1 The first approximation to Hilbert's curve

- The second approximation to the Hilbert's curve can be drawn by further subdividing each of the quadrants and connecting their centres before moving to next major quadrant (see Fig. Q.15.2).
- The third approximation subdivides the quadrants again. We can draw third approximation to Hilbert's curve by connecting the centres of finest level of quadrants before stepping to the next level of the quadrant (See Fig. Q.15.3).

From the above three figures we can easily note following points about Hilbert's curve.

1. If we infinitely extend the approximations to the Hilbert's curve, the curve fills the smaller quadrants but never crosses itself.
2. The curve is arbitrarily close to every point in the square.

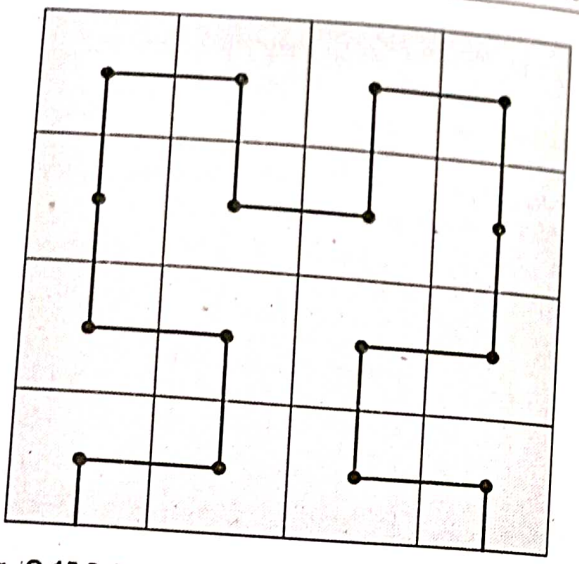


Fig. Q.15.2 Second approximation to Hilbert's curve

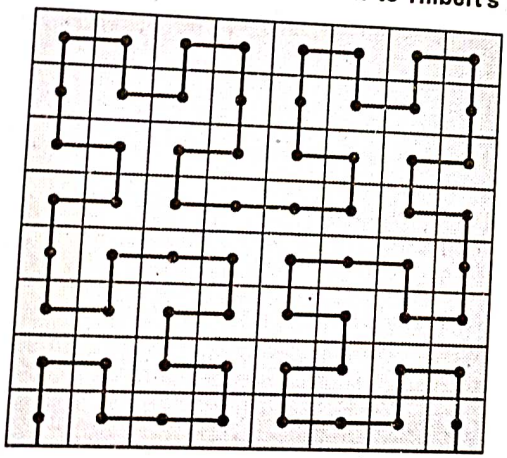


Fig. Q.15.3 Third approximation to Hilbert's curve

3. The curve passes through a point on a grid, which becomes twice as fine with each subdivision.
4. There is no limit to subdivisions and therefore length of curve is infinite.

5. With each subdivision length of curve increases by factor of 4.
6. At each subdivision the scale changes by 2 but length changes by 4 therefore for Hilbert's curve topological dimension is one but the fractal dimension is 2.

Q.16 State the application of Hilbert curve.

☞ **[SPPU : May-18, Marks 2]**

Ans. : Because of this locality property, the Hilbert curve is widely used in computer science. For example, the range of IP addresses used by computers can be mapped into a picture using the Hilbert curve. Code to generate the image would map from 2D to 1D to find the color of each pixel and the Hilbert curve is sometimes used because it keeps nearby IP addresses close to each other in the picture.

Q.17 Explain koch curve in detail giving the fractal dimension.

☞ **[SPPU : Dec.-07,15,17,18, May-08,09,17,19, June-22, Marks 6]**

Ans. : • The Koch curve can be drawn by dividing line into 4 equal segments with scaling factor 1/3 and middle two segments are so adjusted that they form adjacent sides of an equilateral triangle as shown in the Fig. Q.17.1. This is the first approximation to the koch curve.

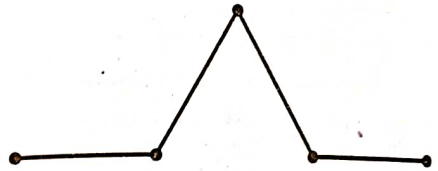


Fig. Q.17.1 First approximation of the Koch curve

• To apply the second approximation to the Koch curve we have to repeat the above process for each of the four segments. The resultant curve is shown in Fig. Q.17.2.

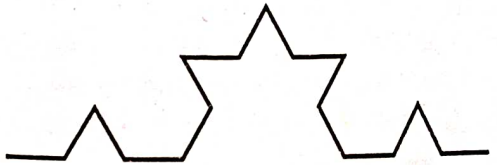


Fig. Q.17.2 The second approximation to Koch curve

- The resultant curve has more wiggles and its length is 16/9 times the original length.
- From the above figures we can easily note following points about the koch curve :
  1. Each repetition increases the length of the curve by factor 4/3.
  2. Length of curve is infinite.
  3. Unlike Hilbert's curve, it doesn't fill an area.
  4. It doesn't deviate much from its original shape.
  5. If we reduce the scale of the curve by 3, we find the curve that looks just like the original one; but we must assemble 4 such curves to make the original, so we have

$$4 = 3^D$$

Solving for D we get

$$D = \log_3 4 = \log 4 / \log 3 \cong 1.2618$$

- Therefore for koch curve topological dimension is 1 but fractal dimension is 1.2618.
- From the above discussion we can say that point sets, curves and surfaces which give a fractal dimension greater than the topological dimension are called **fractals**. The Hilbert's curve and koch curves are fractals, because their fractal dimensions (respectively, 2 and 1.2618) are greater than their topological dimension which is 1.

11.7 : Fractal Lines

Q.18 Write short note on fractal lines.

☞[SPPU : May-10, 11, 12, Dec.-12,13, Marks 4]

OR Explain fractal lines with example.

☞[SPPU : Dec.-15, May-16, Marks 3]

Ans. :

- Fractal line can be generated by performing following steps :
  1. If the straight line segment is specified by points (x_1, y_1, z_1) and (x_2, y_2, z_2), find the midpoint of line by using following expression :

$$\left(\frac{x_1+x_2}{2}, \frac{y_1+y_2}{2}, \frac{z_1+z_2}{2}\right)$$

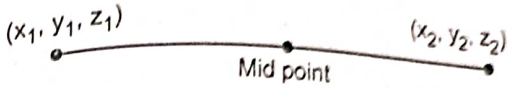


Fig. Q.18.1

2. Add an offset to the midpoint such that the resultant midpoint should not lie on the line itself. This can be achieved by adding offset term to each co-ordinate as follows :

$$\left(\frac{x_1+x_2}{2}+dx, \frac{y_1+y_2}{2}+dy, \frac{z_1+z_2}{2}+dz\right)$$

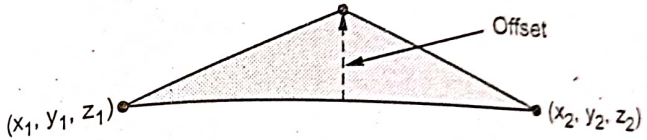


Fig. Q.18.2

In order to get random effect calculate the offset as shown below.

$$dx = L \times W \times GAUSS$$

$$dy = L \times W \times GAUSS$$

$$dz = L \times W \times GAUSS$$

where L : Length of the segment

W : Waiting function governing the curve roughness (i.e. fractal dimension)

GAUSS : Gaussian variable which returns random values between –1 and 1 with 0 mean.

i.e. returned values consist of equal amount of positive and negative values.

3. The shifted midpoint divides the original line into two parts. Repeat the same process for each part separately.
4. Repeat the process 1, 2, 3 until the segments become small enough. By following the above procedure we can get fractal line as shown in the Fig. Q.18.3.

• The above implementation can easily be achieved using a recursive procedure.

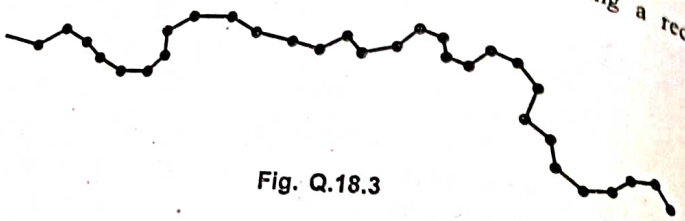


Fig. Q.18.3

END...

Unit VI

12 Segment

12.1 : Introduction

Q.1 What is segment ?

☞[SPPU : May-05,06,07,08,16, Dec.-05,11,13,14,15,17, Marks 2]

OR

Why do we need segments ? ☞[SPPU : Dec.11,13, Marks 2]

Ans. : • In practice, the image on the display screen is often composed of several pictures or items of information.

• An image may contain several views of an object and related information.

• It may also contain close up view of a particular component. For example, we may wish to display an internal plan of a living room. The plan may contain various objects such as sofa-set, T.V., show-case, teapot, show-pieces, wall hangings and so on.

• Each object has a set of attributes such as size, colour and its position in the room.

• We might wish to see all these objects simultaneously or a single object at a time.

• To view the entire image or a part of the image with various attributes we need to organize the image information in a particular manner.

• The image information is stored in the display file.

• Existing structure of the display file does not satisfy our requirements of viewing the image. Hence the display structure is modified to reflect the subpicture structure. To achieve this display file is divided into **segments**.

• Each segment corresponds to a component or an object of the overall display and is associated with a set of attributes.

- Along with the attributes the segment is also associated with the image transformation parameters such as scaling along X and/or Y-direction, rotation and shearing.
- Therefore, presence of segment allows :
  - Subdivision of the picture.
  - Visualization of a particular part of the picture.
  - Scaling, rotation and translation of a particular part of the picture.

12.2 : Segment Table

Q.2 Explain a segment table with example.

☞[SPPU : May-05,06,07,08,12,13,16, Dec.-05,14,15,18, Marks 4]

OR Give the structure of segment table.

Ans. :
- To access a particular segment and the information associated with it we must have a unique name assigned to each segment.
- Along with the name we must have its display file position and its attribute information.
- The structure used to organize all this information related to segments is called **segment table.**

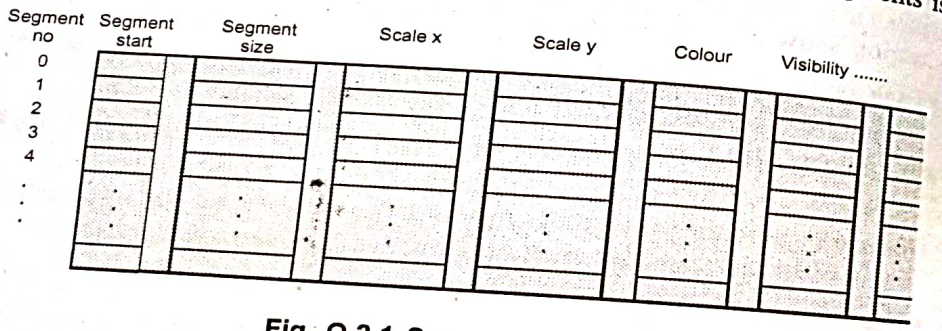


Fig. Q.2.1 Segment table

- First array holds the display file starting location for that segment, the second array holds the segment size information, while the third indicates the scaling and so on. This is illustrated in Fig. Q.2.1.
- Each row in the segment table represents information of one segment including its name, position, size, attributes and the image information parameters.

- If we wish to make segment 4 visible then this is achieved by setting the corresponding entry in the array to 'ON'.
- The display file interpreter initially checks the start, size and visible attribute of the segments and it interprets only those segments which are to be made visible.

12.3 : Functions for Segmenting the Display File

Q.3 Write the algorithm for change of visibility attribute of segments.

☞[SPPU : Dec.-11,13, Marks 3]

Ans. : Algorithm to change visibility of segment

```
Arguments SEGMENT-NAME
          ON-OFF
Global    VISIBILITY                  ... visibility setting
Constant  NUMBER-OF-SEGMENTS          ... array of visibility falgs
                                      ... the size of segment table
BEGIN
  IF SEGMENT-NAME<1 OR SEGMENT-NAME>
                                  NUMER-OF-SEGMENTS
  THEN
     RETURN ERROR 'INVNUD SEGMENT NAME';
     VISIBILITY [SEGMENT-NAME] ← ON-OFF;
     IF NOT ON-OFF THEN NEW-FRAME;
     RETURN;
END;
```

Q.4 Write algorithm to create a segment.

☞ [SPPU : Dec.-19, Marks 6]

Ans. : Algorithm : Create Segment

1. Check whether any segment is open; if so display error message : "Segment is still open" and go to step 9.
2. Read the name of new segment.
3. Check whether the new segment name is valid; if not display error message : "Not a valid segment name"and go to step 9.
4. Check whether the new segment name is already existing in the same-name list; if so display error message : "Segment name already exists" and go to step 9.

5. Initialize the start of the segment at the next free storage area in the display file.
6. Initialize size of this segment equal to zero.
7. Initialize all attributes of segment to their default values.
8. Indicate that the new segment is now open.
9. Stop.

Q.5 Write the algorithm for the following :
i) Delete a segment ii) Delete all segments.

☞[SPPU : Dec.-11,13,17, Marks 5]

Ans. : Algorithm : Delete A Segment

1. Read the name of segment which is to be deleted.
2. Check whether that segment name is valid; if not, display error message "Segment not valid" and go to step 8.
3. Check whether the segment is open, if yes, display error message "Can't delete the open segment" and go to step 8.
4. Check whether the size of segment is greater than 0; if no, no processing is required, as segment contains no instructions. Therefore, go to step 8.
5. Shift the display file elements which follow the segment which is to be deleted by its size.
6. Recover the deleted space by resetting the index of the next free instruction.
7. Adjust the starting positions of the shifted segments by subtracting the size of the deleted segment from it.
8. Stop.

Algorithm : Delete All Segments

```
For I = 0 To NUMBER-OF SEGMENTS DO
BEGIN
   SEGMENT-START[I] ← 1;
   SEGMENT-SIZE[I] ← 0;
END;
NOW-OPEN ← 0;
FREE ← 1;
```



```
NEW-FRAME;
RETURN;
END;
```

Q.6 Write an algorithm to rename a segment.

☞[SPPU : Dec.-17, May-19, Marks 4]

Ans. : Algorithm : Rename Segment

1. Check whether both old and new segment names are valid; if not, display error message "Not valid segment names" and go to step 6.
2. Check whether any of the two segments are open. If open, display error message "Segments still open" and go to step 6.
3. Check whether the new name we are going to give to the old segment is not already existing in the display file. If yes, display error message "Segment already exists" and go to step 6.
4. Copy the old segment table entry into the new position.
5. Delete the old segment.
6. Stop.

Q.7 Explain operations performed on segment table.

☞[SPPU : May-06,07,08,12,13,14,15,16, Dec.-11,13,14, Marks 6]

OR How do we get creat segment. **☞[SPPU : Dec.-13, Marks 2]**

Ans. : Basic operations performed on segments are :

- Create segment
- Close segment
- Delete segment
- Rename segment

Segment Creation

- In a segment creation process, initially, we have to check whether some other segment is still open.
- It is not allowed to open two segments at a same time because it is then difficult to assign the drawing instructions to particular segment. Hence, segment must be created or opened when no other segment is currently open.
- We must give the segment a name so that we can identify it. While doing this it is important to check whether a given segment name is

valid or not and whether there already exists a segment with the same name. If so, we have to assign other valid name to the segment.

- Once a valid segment name is assigned, we have to initialize the items in the segment table under our segment name to indicate that this is a fresh new segment.
- The first instruction of this segment will be located at the next free storage area in the display file.
- As we have not entered any instructions into the segment yet, its size is initialized with value zero.
- The attributes of the segment are initialized as a default attribute values.

Segment Close

- Once a segment is open we can enter the display file instructions in it. The entered commands are then associated with the open segment.
- After completion of entering all display file instructions, the segment must be closed.
- To close a segment it is necessary to change the name of the currently open segment. It can be achieved by changing the name of currently open segment as 0. Now the segment with name 0 is open i.e. unnamed segment is open.
- If there are two unnamed segments in the display file one has to be deleted.

Delete Segment

- When we want to delete a particular segment from the display file then we must recover the storage space occupied by its instructions and make this space free for some other segment. To do this we must not destroy and re-form the entire display file, but we must delete just one segment, while preserving the rest of the display file. The method to achieve this depends upon the data structure used to represent the display file.
- Here, we have used arrays to store the display file information.
- Use of arrays makes the recovery of storage space occupied by the segment very easy and straight forward. However using arrays is not as efficient as some other storage techniques such as link list.
- In case of arrays the gap left by deleted block is filled by shifting up all segments which are following the deleted segment.

Rename Segment

- We can display animated character on the display by presenting sequence of images, each with a slightly different drawing of the character.
- Let us assume that we have a segment in a display file for animated character. Then to display a new image in the sequence we have to delete the current segment and re-create it with the altered character. The problem in this process is that during the time after the first image is deleted and time before the second image is completely entered, only a partially completed character is displayed on the screen. We can avoid this problem by keeping the next image ready in the display file before deleting the current segment.
- This means that both segments, the segment which is to be deleted and the segment which is to be replaced with must exist in the display file at the same time.
- This can be achieved by creating a new invisible image under some temporary segment name.
- Now, when the current segment is deleted we can make the new image visible and rename it with the name of deleted segment.
- These steps can be repeated to achieve the animation.
- The idea of storing two images, one to show and other to create or alter, is called **double buffering**.

12.4 : More about Segments

Q.8 What are advantages of segments.

☞[SPPU : May-15, Marks 4]

Ans. : Advantages

1. Segmentation allows to organize display files in subpicture structure.
2. It allows to apply different set of attributes to the different portions of the image.
3. Due to segmentation selective portion of the image can be displayed.
4. Segmentation makes it easier to modify the picture by changing segment attributes or by replacing segments.
5. Segmentation allows application of transformation on selective portion of the image.

12.5 : Display File Structures

Q.9 Explain various data structures used to implement the segment table. **[SPPU : Dec.-08, May-09,12, Marks 6, May-10,13, Marks 8]**

OR Explain display file and its structure.

[SPPU : May-15, Dec.-17, Marks 4]

Ans. :

Array Structure

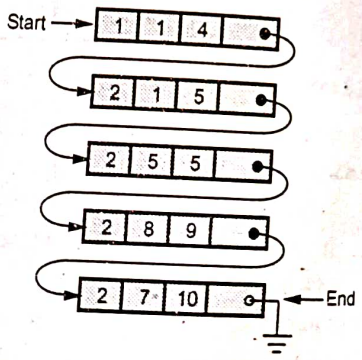
- The display file we have used is a simple array data structure. But array data structure is not much efficient.
- If we wish to remove an instruction at the beginning of the display file, we have to move all succeeding instructions.
- If the display file is large, a lot of processing is required to recover only a small amount of storage.

Linked List

- In a linked list the instructions are not stored in a sequence; rather a new field is added to the instruction. This field, called the **link** or **pointer**, gives the location of the next instruction.
- Here, we can step through the instructions by following the chain of links. This is illustrated in the Fig. Q.9.1.

	Df-op	Df-x	Df-y	Link
1				
2	2	1	5	8
3	1	1	4	2
4				
5				
6	2	7	10	0
7				
8	2	5	5	9
9	2	8	9	6

Start = 3



Note : Df : Display file

Fig. Q.9.1 Display file as linked list

- When we want to add an instruction to a display file, a free cell is obtained from the list, the correct instruction operation code and operands are stored and the cell is linked to the display file list.
- Deletion of cells from a linked list is very easy. To delete a cell, we need only change the pointer which points to that cell so that it points to the succeeding cell. This is illustrated in Fig. Q.9.2.

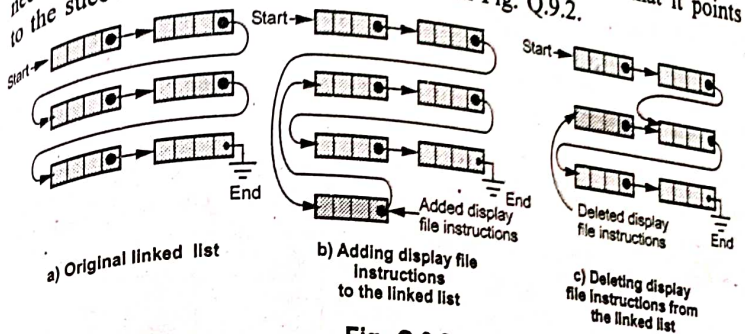


Fig. Q.9.2

Paging Scheme

- The paging scheme is a combination of array and linked list structure.
- In this method the display file is organized into a number of small arrays called **pages.**
- The pages are linked to form a linked list of pages.
- Each segment starts at the beginning of a page.
- If segment ends before the page boundary, the remaining page is not used.
- In this method, display file instructions can be accessed within a page just as they were accessed in an array.
- When the end of a page is reached, a link is followed to find the next page. This is illustrated in Fig. Q.9.3.

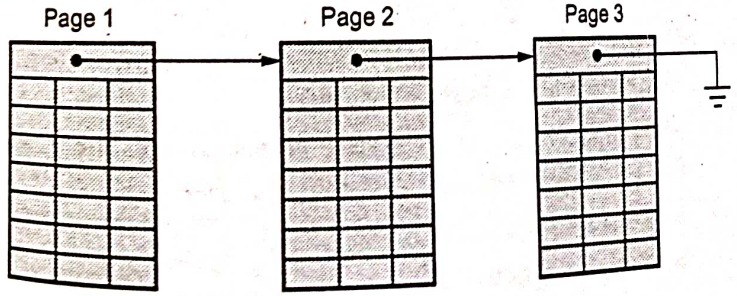


Fig. Q.9.3 Paging scheme for display file

12.6 : Image Transformation

Q.10 Explain, how segment operations play major role in image presentation and processing. **[SPPU : May-11, Marks 8]**

Ans. : Image Transformation

- One of the major advantage of computer graphics is that we can display pictures with certain alterations if required. For example, we can see image from a different angle, we can zoom particular part of the image, we can change the size of the image, we can change the position of particular part of the image.
- These changes are easy to perform because the graphics image information is stored in different segments and we can apply changes to different segments independently.

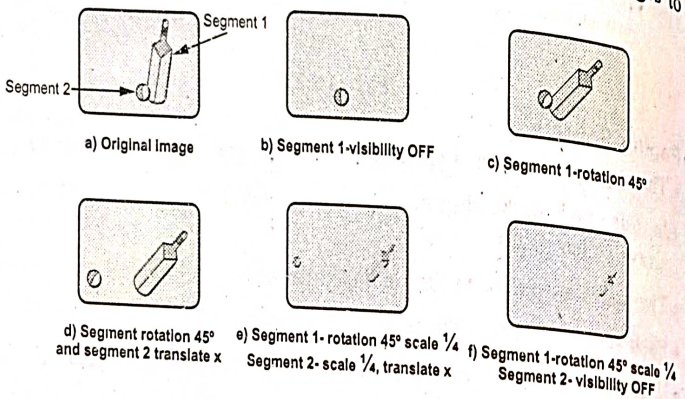


Fig. Q.10.1

- The changes in the original graphics information can be done by performing mathematical operations which are called **transformations.**
- The Fig. Q.10.1 shows the original image and the images after applying some form of transformation.

Algorithm : Setting Image Transformation

1. Read the name of segment, say s_name.
2. Check whether the name is valid or not. If not display error message " Not a valid segment " and go to step 8.
3. Read translation factor, say t_x and t_y.
4. Read the scaling factors, say s_x and s_y.

5. Read the angle of rotation, say θ.
6. Set various transformation parameters for the segment s_name.
   Translate_x [s_name] = t_x.
   Translate_y [s_name] = t_y.
   Scale_x [s_name] = s_x.
   Scale_y [s_name] = s_y.
   Rotate [s_name] = θ.
7. Check the visibility of the segment s_name. If visible then create a new frame for it.
8. Stop.

12.7 : Raster Technique

Q.11 Explain BITBLT operation of raster technique.

[SPPU : May-14, Marks 4]

Ans. : • The operation called a **RasterOp** or **Bit Block-Transfer** (BITBLT) can be used with raster displays.

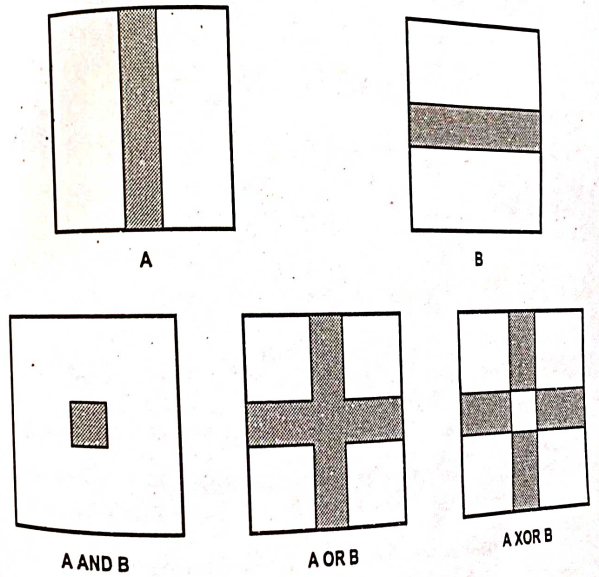


Fig. Q.11.1 Logical operation

- Here, logical operations are performed on bit-addressed blocks of memory.
- The BITBLT works on subarrays of the frame buffer.
- It performs simple operations on subarrays such as turning all pixels on or off, shifting all values by a row or a column, and copying values from another subarray.
- For one-bit-per-pixel frame buffer, it forms the logical AND, OR, or EXCLUSIVE_OR of pixel values in two subarrays, as shown in the Fig. Q.11.1.

END...

Unit VI

13 Animation

13.1 : Introduction

Q.1 What is computer animation ?

☞ **[SPPU : Dec.-10,13, Marks 4]**

OR Define animation. ☞ **[SPPU : Dec.-11, 16, May-14, Marks 2]**

Ans. : • Computer animation is the art of creating moving images via the use of computers. It is a subfield of computer graphics and animation.

• In computer animation to create the illusion of movement, an image is displayed on the computer screen, then quickly replaced by a new image that is similar to the previous image, but slightly shifted. This technique is identical to how the illusion of movement is achieved with television and motion pictures.

13.2 : Conventional and Computer Based Animation

Q.2 Define storyboard, keyframes, inbetween, pencil test and cels.

Ans. : • **Storyboard :** With the help of sequence of sketches the ideas and of structure animation is created. This outline form of animation is called a 'storyboard'.

• **Keyframes :** These are the frames in animation in which the entities being animated are at extreme position.

• **Inbetween :** Once the key-frames of the animation are drawn, the intermediate positions can be inferred. Filling the intermediate frames in key-frames is called 'inbetween'.

• **Pencil test :** With key-frames and inbetween a trial film is made, which is called a 'pencil test'.

• **Cels :** A common technique is to decompose an animation picture into several parts that can move more or less independently. These parts are

drawn individually on a clear plastic material that are called **cels** (Short for cellulose).

Q.3 Define route-sheet and exposure sheet.

Ans. : • **Route-sheet :** The sheet which describes each scene and the people responsible for the various aspects of producing the scene is called 'route-sheet'.

• **Exposure sheet :** The sheet which is an immensely detailed description of animation, is called an 'exposure sheet'. The exposure sheet gives details for each frame such as the order of all the figures in the frame, dialogue description, choice of background and the camera position within the frame.

Q.4 List the steps needed for creating conventional animation.

Ans. : • The sequence of steps needed for creating a conventional animation are :

1. The story of animation is a written and a storyboard is created.
2. If any sound track is to be recorded, according to the drawing for every scene in the animation, a detailed layout is produced.
3. The instants at which significant sounds occur are recorded in order. Then the detailed layout and the sound track are correlated.
4. Certain key-frames of the animation are drawn.
5. With key-frames and inbetween, pencil test frames are obtained.
6. The pencil test frames are then transferred to cels. This is done either by hand copying in ink or by photocopying directly onto the cels.
7. The cels are coloured in or painted and are arranged in a correct sequence. Then they are filmed.

Q.5 What is computer based animation ?

Ans. : • Computer animation can be created with a computer and animation software. Some examples of animation software are : Amorphium, Art of illusion, 3D studio max, Adobe flash and many more. These software provide basic functions to create an animation and to process individual objects. The functions are also available to store and manipulate object database, motion generation and object rendering. In object data base object shapes and associated parameters are stored and they are updated as we proceed animation.

Q.6 Explain the steps needed in computer generated animation.

Ans. : The basic steps needed in computer generated animation are :

1. **Digitization :** Before using the computer for animation, the drawings, decomposed parts must be digitized. This can be done using methods such as, producing original drawings by animation paint systems using optical scanning by tracking the drawings with a data tablet.

   The resulting drawings may need further processing such as filtering to clean up the glitches (if any). It also smooths the contours.

2. **Composition :** In this stage, the individual key frames are generated with proper combination of cels. The foreground and background figures are also combined using image-composition techniques to prepare final key-frames.

3. **In-between frames :** Animation software allows user to generate in-between frames. These are the intermediate frames between the keyframes. Usually there are 3 to 5 in-between frames between two keyframes. Suppose we want to design an animation sequence for 10 seconds. For this we need to have 24 × 10 frames because film requires 24 frames per second. Out of these 240 frames we can have 48 keyframes and remaining 192 in-between frames. In this case there are four in-between frames between two keyframes.

4. **Equivalent of pencil test :** A rectangular array is taken and several small low-resolution frames of an animation are placed in it. In a frame buffer, a particular portion of such an low resolution image, typically one twenty-fifth or one thirty-sixth is taken. This portion is moved to the center of the screen which is called **panning**. Then using **zooming** technique this portion is enlarged so as to fill the entire screen. The same procedure is repeated on the several films of the animation which are stored in an array for the single image. If these steps are carried out fast enough, the effect of continuity is observed. Since each frame of the animation is reduced to a very small part of the total image (typically one twenty-fifth or one thirty-sixth), and is then expanded to fill the screen, this process effectively lowers the display device's resolution.

13.3 : Design for Animation Sequences

Q.7 Explain the steps for designing animation sequences.

☞[SPPU : May-18, Marks 2]

Ans. : The steps for designing animation sequences are as follows :

- Storyboard layout
- Object definitions
- Keyframe specifications
- Generation of in-between frames

- **Storyboard layout :** The storyboard is an outline of the action. It defines a set of basic events that are to take place in a specific order.

13.4 : Animation Languages

Q.8 Write a note on animation languages.

☞ [SPPU : Dec.-12,13, Marks 8]

OR Give different types of animation languages.

☞ [SPPU : Dec.-11, Marks 2]

Ans. : • There are many different languages for describing computer animation, and new ones are constantly being developed. These animation languages can be categorized as :

- Linear list notations
- General purpose languages with embedded animation directives.
- Graphical languages

Linear list notations

- In linear-list notations, each event in the animation is described by a starting and ending frame number and an action that is to take place.

General purpose languages

- General purpose languages such as C, C++, Pascal or Lisp can be used to design and control the animation sequences.

Graphical languages

- There are several specialized animation languages developed called graphical languages.
- These languages provide various animation functions which make it easy to design and control the animation.

13.5 : Keyframes and Morphing

Q.9 What is morphing ?

☞[SPPU : Dec.-12, May-13,18,19, Marks 2; June-22, Marks 3]

Ans. : • Morphing is a special effect in motion pictures and animations that changes one image into another through a seamless transition.

- It is a transformation of object shapes from one form to another.

Q.10 What are the applications of morphing ?

☞[SPPU : Dec.-12, May-15, Marks 3]

Ans. : Morphing is used to produce special effects in motion pictures and animations that changes one image into another through a seamless transition. It is used as a transition technique between one scene and another in television shows, even if the contents of the two images are entirely unrelated. It is commonly used in entertainment industry to produce special effects.

Q.11 What is tweening ?

Ans. : • Tweening is a short for in-betweening. It is the process of generating intermediate frames between two images to give the appearance that the first image evolves smoothly into the second image.

- It is an interpolation technique where an animation program generates extra frames between the key frames that the user has created. This gives smoother animation without the user having to draw every frame.
- In tweening a scene is described by a mathematical model - a set of two or three-dimensional objects whose positions are given by sets of coordinates. It uses mathematical formulae to generate these coordinates at a sequence of discrete times.

13.6 : Motion Specification

Q.12 Explain various methods to specify the motion of the objects.

☞ [SPPU : June-22, Marks 6]

Ans. : • The various ways in which the motions of objects can be specified. These are :

- Direct motion specification
- Goal-directed systems

- **Direct motion specification**
  - It is most straightforward method for defining a motion sequence.
  - In this method, the rotation angles and the translation vectors are specified so that the geometrical transformations can be applied to the objects in the scene to generate animation sequences.
- **Goal-directed systems**
  - In these systems, instead of specifying motion parameters, action goal specific instructions are specified. Thus these systems are known as goal directed systems.
  - For example, we could specify that we want an object to walk or to run to a particular destination.
- **Kinematics and Dynamics**
  - In case of kinematic descriptions, motion parameters such as position, velocity and acceleration are specified without reference to the forces that causes the motion to generate animation sequences.

END...

Unit VI

14 Gaming

14.1 : Gaming Platforms

Q.1 Explain need of NVIDIA workstation in Gaming.

[SPPU : Dec.-15, 17, Marks 5]

Ans. : In gaming, we need a processor to perform many graphics objects simultaneously and in real time. A conventional CPU does not have special hardware support to perform such task efficiently. However, NVIDIA workstation uses NVIDIA GPU which provides high floating point execution bandwidth, hardware multithreading, several streaming multiprocessors and so on. All these contribute to make NVIDIA GPU workstation most suitable for high performance graphics and computer animation involved in gaming.

Q.2 Write curve four important features of NVIDIA gaming platform. **[SPPU : May-19, Marks 4]**

Ans. : Features of NVIDIA gaming platform

1. Highest level performance and the smoothest experience possible from the moment you start playing.
2. Enables developers to add amazing graphics effects.
3. Dedicated ray tracing hardware enables fast real-time ray tracing with physical accurate shadows, reflection, refractions and global elimination.
4. Provides variable rate shading and faster frame rates.

Q.3 Explain the architecture of any NVIDIA processor.

[SPPU : Dec.-15, 17, 18, May-18, Marks 4]

Ans. : • Fig. Q.3.1 illustrates the major components of a general-purpose graphics processor.

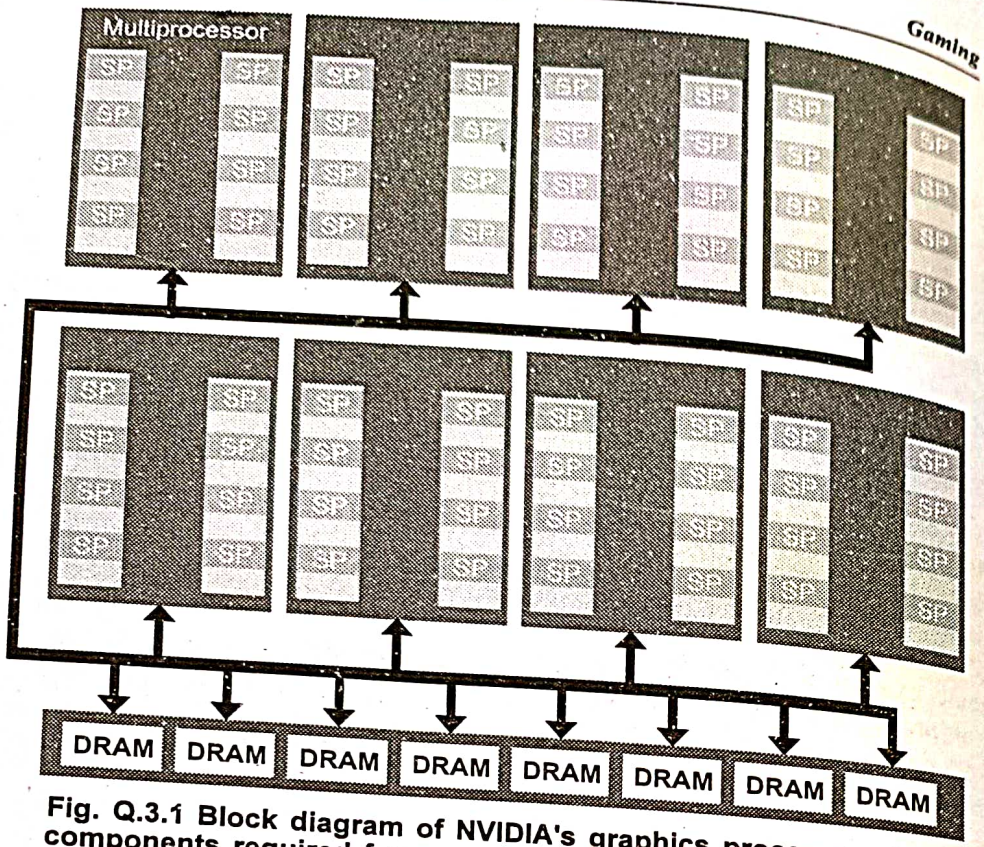


Fig. Q.3.1 Block diagram of NVIDIA's graphics processor (the components required for graphics processing are not shown)

- At a high level, the GPU architecture consists of several ***Streaming Multiprocessors*** **(SMs)**, which are connected to the GPU's DRAM.
- Each SM has a number of single-instruction multiple data (SIMD) units, also called **Stream Processors (SPs)**, and supports a multithreading execution mechanism. GPU architectures employ two important execution paradigms :
  - SIMD / SIMT
  - Multithreading.

SIMD / SIMT

- GPU processors supply high Floating Point (FP) execution bandwidth, which is the driving force for designing graphics applications. To make efficient use of the high number of FP units, GPU architectures employ a *SIMD* or *SIMT* execution paradigm.

Multithreading

- The other important execution paradigm that GPU architectures employ is ***hardware multithreading***.
- GPU processors use fast hardware-based context switching to tolerate long memory and operation latencies.

Q.4 Draw and explain the block diagram of i860.

[SPPU : May-14,15,16, Dec.-14,16,17, Marks 4, June-22, Marks 6]

Ans. : Fig. Q.4.1 illustrate the registers and data paths of the i860 microprocessor respectively.

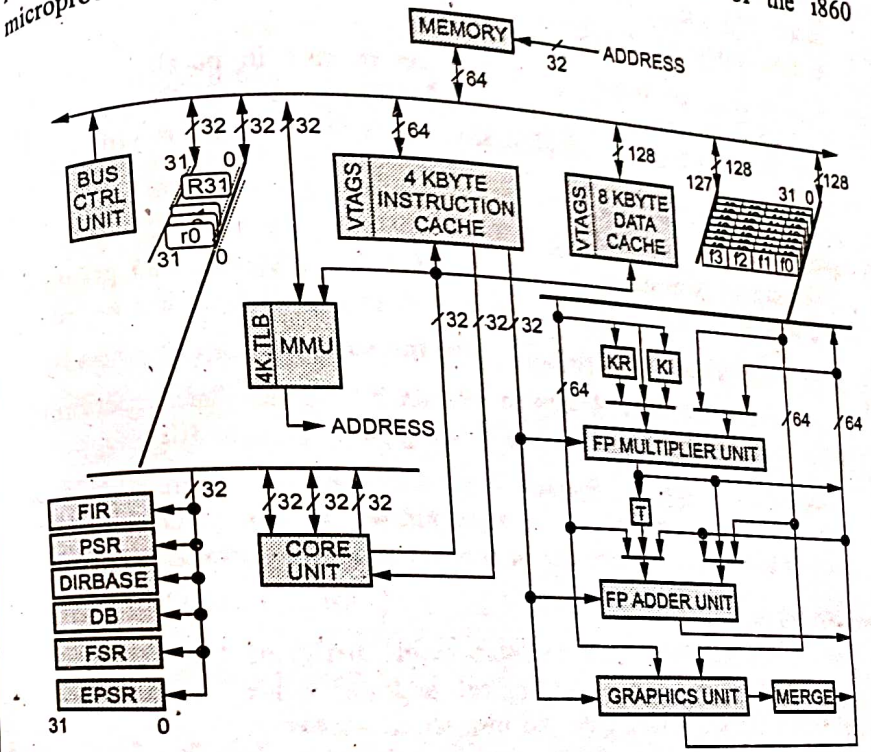


Fig. Q.4.1 i860 microprocessor architecture block diagram

Integer Core Unit

- The core unit is the administrative center of the processor. The core unit fetches both integer and floating-point instructions.
- It contains the integer register file and executes load, store, integer, bit and control-transfer operations. Its pipelined organization with extensive bypassing and scoreboarding maximizes performance.

Floating-Point Unit

- The floating-point unit contains the **floating-point register file**. This file can be accessed as 8 × 128-bit registers, 16 × 64-bit registers or 32 × 32-bit registers. Three additional registers **(KR KI, and T)** hold intermediate floating-point results.
- The floating-point unit contains both the floating-point adder and the floating-point multiplier. Both units support 64 and 32-bit floating-point values in IEEE Standard 754 format.
- Each of these units uses pipelining to deliver up to one result per clock. The adder and multiplier can operate in parallel, producing up to **two results per clock.**
- Furthermore, the floating-point unit can operate in parallel with the core unit, sustaining a rate of **two floating-point results per clock** rate by overlapping administrative functions with floating-point operations.

Graphics Unit

- The graphics unit has 64-bit integer logic that supports 3-D graphics drawing algorithms.
- This unit can **operate in parallel with the core unit.**
- It contains the special-purpose MERGE register and performs additions on integers stored in the floating-point register file.
- These special graphics features focus on applications that involve three-dimensional graphics with Gouraud or Phong color intensity shading and hidden surface elimination via the Z-buffer algorithm.

Memory Management Unit

- The on-chip MMU of i860 microprocessors performs the translation of addresses from the linear logical address space to the linear physical address for both data and instruction access.

Caches

- In addition to the page translation caches (TLBs), the i860 microprocessor contains separate on-chip caches for data and instructions.

14.2 : Advances in Gaming

Q.5 Write a note on advances in gaming technology.

Ans. : Currently, there are tremendous **advances in gaming** technologies to improve physical realism of environments and characters.

1. **Facial Recognition :** 3D scanning and facial recognition technology allows systems to actually create your likeness in the gaming world or to inventively transfer your own expressions to other digital creations. On top of that, the Intel® RealSense™ 3D camera could allow developers to create games that adapt to the emotions of the gamer by scanning 78 different points on a person's face.

2. **Voice Recognition :** Voice controlled gaming has been around for a while, but the potential of using the technology in gaming systems has finally caught up to reality-computers are now able to easily recognize voice commands from the user. Not only you can turn the console on and off using this technique, but you can also use voice commands to control gameplay, interact on social media, play selections from your media library, or search the web, all by simply talking to your gaming system.

3. **Gesture Control :** Gesture control allows users to connect with their gaming experience by using the natural movements of your body.

4. **Amazing Graphics :** Cutting edge advancements now allow gamers to experience games in fully rendered worlds with photo realistic textures. The ability to increase playability with higher image quality makes it seem like you're right inside the game.

5. **High-definition Displays :** Televisions with 4K capabilities (meaning it must support at least 4,000 pixels) or 4K laptops are now available to watch the games we play.

6. **Virtual Reality :** Though many virtual reality gaming consoles haven't been commercially released as of yet, those developing VR headset displays are poised to grant gamers a fully immersive gaming experience the likes of which nobody has seen before.

7. **Augmented Reality :** Augmented Reality (AR) is a live direct or indirect view of a physical, real-world environment whose elements are augmented (or supplemented) by computer-generated sensory input such as sound, video, graphics or GPS data.

8. Wearable Gaming : Whether it's smartwatches or glasses, wearable games make gaming portable without being too invasive.

9. Mobile Gaming : With the advent of smartphones, the gaming experience has been taken out of the arcade and the living room and put into the palm of your hand. As evidenced by the countless people on your morning train commute huddled over games on their devices, mobile technology has made the love of digital gaming spread beyond hardcore console-consumers and online gamers.

10. Cloud Gaming : Instead of creating video game systems that require more powerful hardware, developers are looking to lighten the load with the cloud. Games no longer need be limited by the amount of memory that discs or consoles have to offer. Using the cloud opens games up to massive server-size limits where images are streamed to your screen through the Internet.

11. On-demand Gaming : Gamers can already watch and share live-streams of games, but what about playing them ? Much like similar movie streaming services, the ability to stream video games is becoming more and more a reality, and it could lead game developers both big and small to compete for gaming glory.

END...