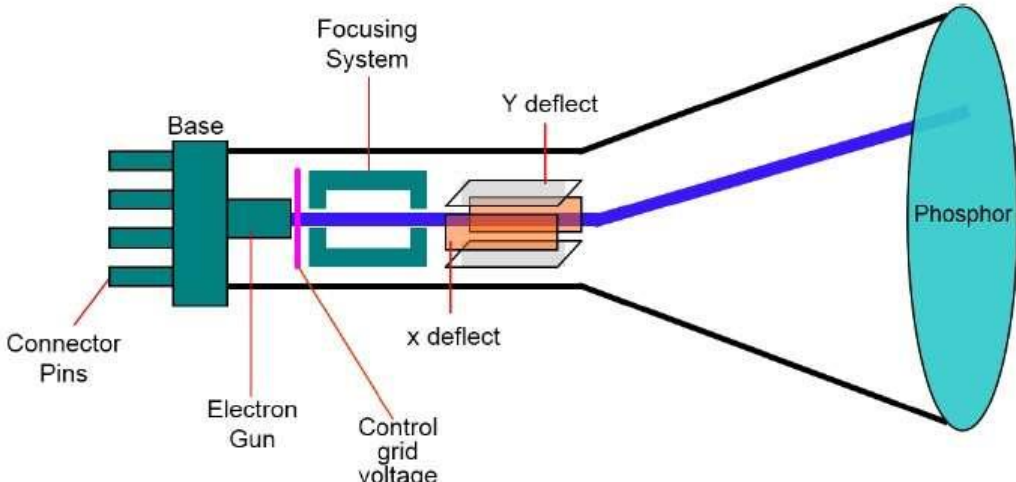


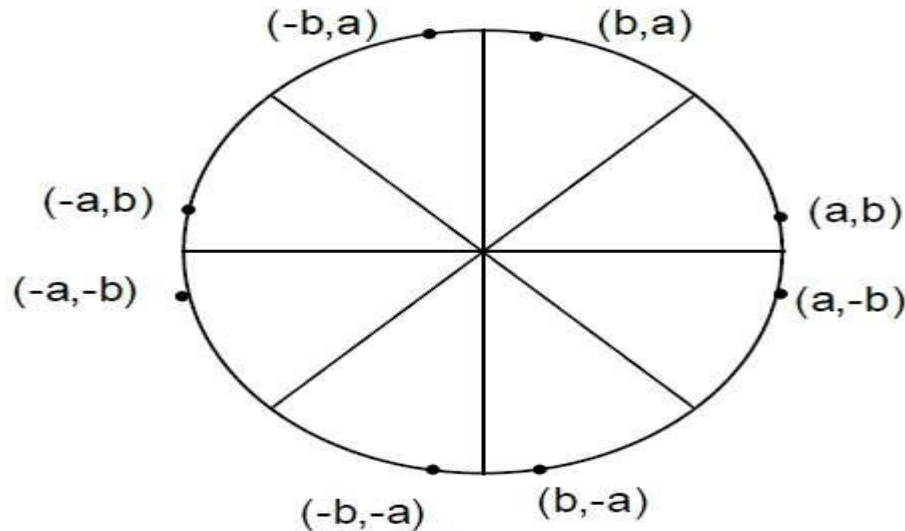
- N. B.: (1) **All** questions are **compulsory**.
 (2) Makes **suitable assumptions** wherever necessary and **state the assumptions** made.
 (3) Answers to the **same question** must be **written together**.
 (4) Numbers to the **right** indicate **marks**.
 (5) Draw **neat labeled diagrams** wherever **necessary**.
 (6) Use of **Non-programmable** calculators is **allowed**.

1 Attempt any three of the following:	1 5
<p>a What is Computer Graphics? How image is to be display on Video Display Device?</p> <p>Computer graphics is an art of drawing pictures on computer screens with the help of programming. It involves computations, creation, and manipulation of data. In other words, we can say that computer graphics is a rendering tool for the generation and manipulation of images.</p> <p>Video Display Device Cathode Ray Tube</p> <p>The primary output device in a graphical system is the video monitor. The main element of a video monitor is the Cathode Ray Tube (CRT), shown in the following illustration.</p> <p>The operation of CRT is very simple –</p> <ul style="list-style-type: none"> • The electron gun emits a beam of electrons (cathode rays). • The electron beam passes through focusing and deflection systems that direct it towards specified positions on the phosphor-coated screen. • When the beam hits the screen, the phosphor emits a small spot of light at each position contacted by the electron beam. • It redraws the picture by directing the electron beam back over the same screen points quickly. 	

There are two ways (Random scan and Raster scan) by which we can display an object on the screen.

b Explain the method of circle drawing using Midpoint Circle Algorithm.

The equation of circle is $X^2 + Y^2 = r^2$, where r is radius.



Midpoint Circle Algorithm

As in the raster line algorithm, we sample at unit intervals and determine the closest pixel position to the specified circle path at each step. For a given radius r and screen center position (x_c, y_c) , we can first set up our algorithm to calculate pixel positions around a circle path centered at the coordinate origin $(0, 0)$. Then each calculated position (x, y) is moved to its proper screen position by adding x_c to x and y_c to y . Along the circle section from $x = 0$ to $x = y$ in the first quadrant, the slope of the curve varies from 0 to -1.0 . Therefore, we can take unit steps in the positive x direction over this octant and use a decision parameter to determine which of the two possible pixel positions in any column is vertically closer to the circle path. Positions in the other seven octants are then obtained by symmetry.

To apply the midpoint method, we define a circle function as

$$f_{\text{circ}}(x, y) = x^2 + y^2 - r^2 \quad (29)$$

Any point (x, y) on the boundary of the circle with radius r satisfies the equation $f_{\text{circ}}(x, y) = 0$. If the point is in the interior of the circle, the circle function is negative; and if the point is outside the circle, the circle function is positive. To summarize, the relative position of any point (x, y) can be determined by checking the sign of the circle function as follows:

$$f_{\text{circ}}(x, y) \begin{cases} < 0, & \text{if } (x, y) \text{ is inside the circle boundary} \\ = 0, & \text{if } (x, y) \text{ is on the circle boundary} \\ > 0, & \text{if } (x, y) \text{ is outside the circle boundary} \end{cases} \quad (30)$$

The tests in 30 are performed for the midpositions between pixels near the circle path at each sampling step. Thus, the circle function is the decision parameter in the midpoint algorithm, and we can set up incremental calculations for this function as we did in the line algorithm.

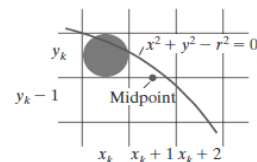


FIGURE 14

Figure 14 shows the midpoint between the two candidate pixels at sampling position $x_k + 1$. Assuming that we have just plotted the pixel at (x_k, y_k) , we next need to determine whether the pixel at position $(x_k + 1, y_k)$ or the one at position $(x_k + 1, y_k - 1)$ is closer to the circle. Our decision parameter is the circle function

$$p_k = f_{\text{circ}}\left(x_k + 1, y_k - \frac{1}{2}\right) = (x_k + 1)^2 + \left(y_k - \frac{1}{2}\right)^2 - r^2 \quad (31)$$

If $p_k < 0$, this midpoint is inside the circle and the pixel on scan line y_k is closer to the circle boundary. Otherwise, the midpoint is outside or on the circle boundary, and we select the pixel on scan line $y_k - 1$.

Successive decision parameters are obtained using incremental calculations. We obtain a recursive expression for the next decision parameter by evaluating the circle function at sampling position $x_{k+1} + 1 = x_k + 2$:

$$p_{k+1} = f_{\text{circ}}\left(x_{k+1} + 1, y_{k+1} - \frac{1}{2}\right) = [(x_k + 1) + 1]^2 + \left(y_{k+1} - \frac{1}{2}\right)^2 - r^2$$

or

$$p_{k+1} = p_k + 2(x_k + 1) + (y_{k+1}^2 - y_k^2) - (y_{k+1} - y_k) + 1 \quad (32)$$

where y_{k+1} is either y_k or $y_k - 1$, depending on the sign of p_k .

FIGURE 14 Midpoint between candidate pixels at sampling position $x_k + 1$ along a circular path.

Increments for obtaining p_{k+1} are either $2x_{k+1} + 1$ (if p_k is negative) or $2x_{k+1} + 1 - 2y_{k+1}$. Evaluation of the terms $2x_{k+1}$ and $2y_{k+1}$ can also be done incrementally as

$$2x_{k+1} = 2x_k + 2 \\ 2y_{k+1} = 2y_k - 2$$

At the start position $(0, r)$, these two terms have the values 0 and $2r$, respectively. Each successive value for the $2x_{k+1}$ term is obtained by adding 2 to the previous value, and each successive value for the $2y_{k+1}$ term is obtained by subtracting 2 from the previous value.

The initial decision parameter is obtained by evaluating the circle function at the start position $(x_0, y_0) = (0, r)$:

$$p_0 = f_{\text{circ}}\left(1, r - \frac{1}{2}\right) = 1 + \left(r - \frac{1}{2}\right)^2 - r^2$$

or

$$p_0 = \frac{5}{4} - r \quad (33)$$

If the radius r is specified as an integer, we can simply round p_0 to

$$p_0 = 1 - r \quad (\text{for } r \text{ an integer})$$

because all increments are integers.

As in Bresenham's line algorithm, the midpoint method calculates pixel positions along the circumference of a circle using integer additions and subtractions, assuming that the circle parameters are specified in integer screen coordinates.

c Distinguish between Active and Passive graphics device.

Parameter	Active Devices	Passive devices
Control	It is dynamic in nature. Control is provided to user to manipulate the graphics.	It is Static in nature. Control is not provided to user to manipulate the graphics. It Works on already written instructions
Communication	It provides two way communications, between user and computer.	It provides a one way communication, only through computer.
Interaction	High bandwidth user interaction with hardware devices.	No interaction between user and hardware devices.
Application	Modern application	Older Application
Motion and Updation	Facility available which supports 2-D and 3-D	No facility available which supports 2-D and 3-D

d Explain the problems of aliasing with example.

Points to be covered.

- 1.Jagged Profiles
- 2.Improperly rendering
- 3.Disintegrating textures.

e Explain different types of Video Formats.

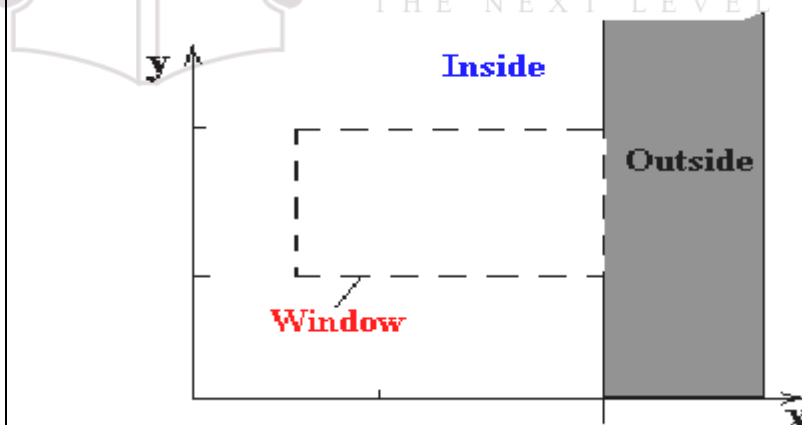
List of formats.
AVI (Audio Video Interleave) ...
FLV (Flash Video Format) ...
WMV (Windows Media Video) ...
MOV (Apple QuickTime Movie) ...
MP4 (Moving Pictures Expert Group 4)

- f Explain the Acceptance and Rejection test using bit codes in Cohen-Sutherland line clipping algorithm.

The Cohen-Sutherland line clipping algorithm quickly detects and dispenses with two common and trivial cases. To clip a line, we need to consider only its endpoints. If both endpoints of a line lie inside the window, the entire line lies inside the window. It is **trivially accepted** and needs no clipping. On the other hand, if both endpoints of a line lie entirely to one side of the window, the line must lie entirely outside of the window. It is **trivially rejected** and needs to be neither clipped nor displayed.

Inside-Outside Window Codes

To determine whether endpoints are inside or outside a window, the algorithm sets up a **half-space code** for each endpoint. Each edge of the window defines an infinite line that divides the whole space into two half-spaces, the **inside half-space** and the **outside half-space**, as shown below.



As you proceed around the window, extending each edge and defining an inside half-space and an outside half-space, nine regions are created - the eight "outside" regions and the one "inside" region. Each of the nine regions associated with the window is assigned a 4-bit code to identify the region. Each bit in the code is set to either a **1**(true) or a **0**(false). If the region is to the **left** of the window, the **first** bit of the code is set to 1. If the region is to the **top** of the window, the **second** bit of the code is set to 1. If to the **right**, the **third** bit is set, and if to the **bottom**, the **fourth** bit is set. The 4 bits in the code then identify each of the nine regions as shown below.

1001	0001	0101
1000	0000	0100
	Window	
1010	0010	0110

For any endpoint (x, y) of a line, the code can be determined that identifies which region the endpoint lies. The code's bits are set according to the following conditions:

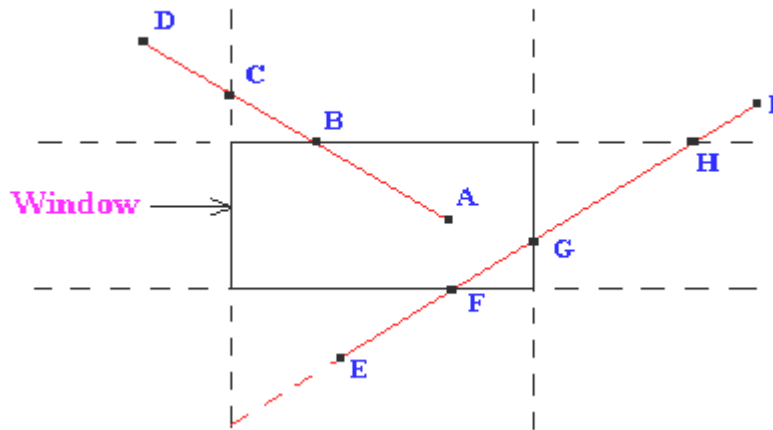
- First bit set **1** : Point lies to **left** of window $x < x_{min}$
- Second bit set **1** : Point lies to **right** of window $x > x_{max}$
- Third bit set **1** : Point lies below(**bottom**) window $y < y_{min}$
- fourth bit set **1** : Point lies above(**top**) window $y > y_{max}$

The sequence for reading the codes' bits is **LRBT** (Left, Right, Bottom, Top).

Once the codes for each endpoint of a line are determined, the logical **AND** operation of the codes determines if the line is completely outside of the window. If the logical AND of the endpoint codes is **not zero**, the line can be trivially rejected. For example, if an endpoint had a code of 1001 while the other endpoint had a code of 1010, the logical AND would be 1000 which indicates the line segment lies outside of the window. On the other hand, if the endpoints had codes of 1001 and 0110, the logical AND would be 0000, and the line could not be trivially rejected.

The logical **OR** of the endpoint codes determines if the line is completely inside the window. If the logical OR is **zero**, the line can be trivially accepted. For example, if the endpoint codes are 0000 and 0000, the logical OR is 0000 - the line can be trivially accepted. If the endpoint codes are 0000 and 0110, the logical OR is 0110 and the line can not be trivially accepted.

Before Clipping



1. Consider the line segment **AD**.

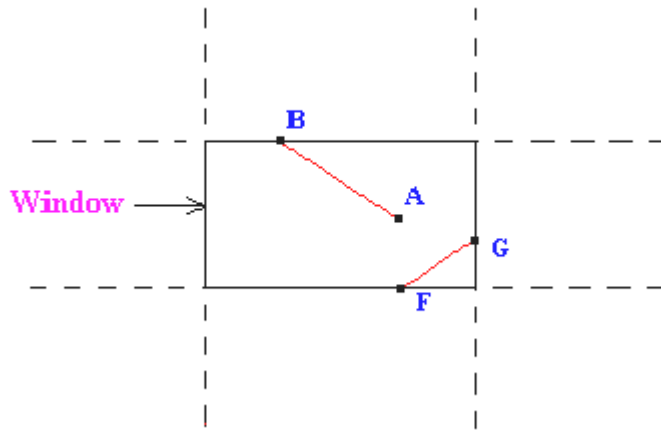
Point **A** has an outcode of **0000** and point **D** has an outcode of **1001**. The logical AND of these outcodes is zero; therefore, the line cannot be trivially rejected. Also, the logical OR of the outcodes is not zero; therefore, the line cannot be trivially accepted. The algorithm then chooses **D** as the outside point (its outcode contains 1's). By our testing order, we first use the top edge to clip **AD** at **B**. The algorithm then recomputes **B**'s outcode as **0000**. With the next iteration of the algorithm, **AB** is tested and is trivially accepted and displayed.

2. Consider the line segment **EI**

Point **E** has an outcode of **0100**, while point **I**'s outcode is **1010**. The results of the trivial tests show that the line can neither be trivially rejected or accepted. Point **E** is determined to be an outside point, so the algorithm clips the line against the bottom edge of the window. Now line **EI** has been clipped to be line **FI**. Line **FI** is tested and cannot be trivially accepted or rejected. Point **F** has an outcode of **0000**, so the algorithm chooses point **I** as an outside point since its outcode is **1010**. The line **FI** is clipped against the window's top edge, yielding a new line **FH**. Line **FH** cannot be trivially accepted or rejected. Since **H**'s outcode is **0010**, the next iteration of the algorithm clips against the window's right edge, yielding line **FG**. The next iteration of the algorithm tests **FG**, and it is trivially accepted and display.

After Clipping

After clipping the segments **AD** and **EI**, the result is that only the line segment **AB** and **FG** can be seen in the window.



2 Attempt any three of the following:

1
5

a Perform mapping from Window to Viewport coordinate transformation.

The viewing transformation is the combination of normalization transformation workstation transformations as shown in the Fig. 5.4. It is given as

$$V = N \cdot W$$

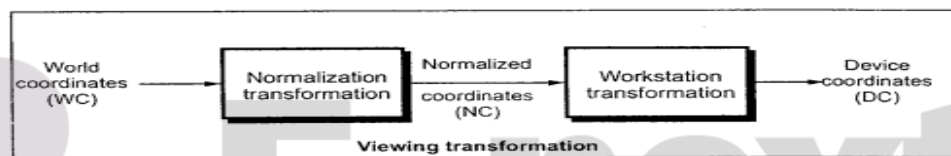


Fig. 5.4 Two dimensional viewing transformation

We know that world coordinate system (WCS) is infinite in extent and the display area is finite. Therefore, to perform a viewing transformation we select a finite world coordinate area for display called a **window**. An area on a device to which a window is mapped is called a **viewport**. The window defines what is to be viewed; the viewport defines where it is to be displayed, as shown in the Fig. 5.5.

The window defined in world coordinates is first transformed into the normalized device coordinates. The normalized window is then transformed into the viewport coordinate. This window to viewport coordinate transformation is known as workstation transformation. It is achieved by performing following steps :

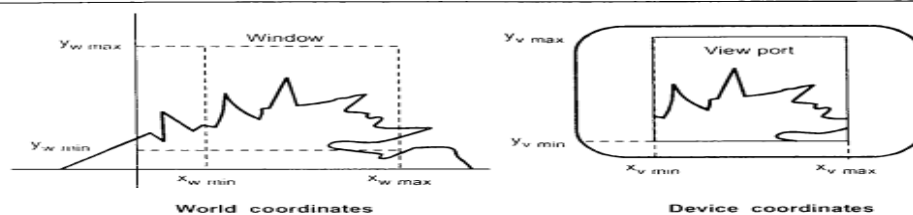


Fig. 5.5 Window and viewport

1. The object together with its window is translated until the lower left corner of the window is at the origin.
 2. Object and window are scaled until the window has the dimensions of the viewport.
 3. Translate the viewport to its correct position on the screen.
- This is illustrated in Fig.5.6.

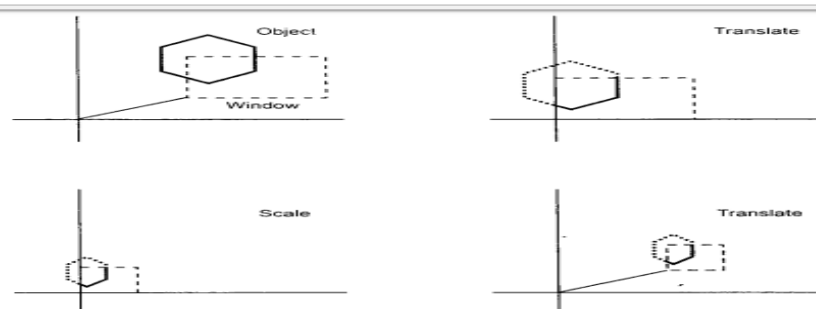


Fig. 5.6 Steps in workstation transformation

Therefore, the workstation transformation is given as

$$W = T \cdot S \cdot T^{-1}$$

The transformation matrices for individual transformation are as given below :

$$T = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -X_{w \min} & -Y_{w \min} & 1 \end{bmatrix}$$

$$S = \begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad \text{where} \quad S_x = \frac{x_{v \max} - x_{v \min}}{x_{w \max} - x_{w \min}}$$

$$S_y = \frac{y_{v \max} - y_{v \min}}{y_{w \max} - y_{w \min}}$$

$$T^{-1} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ x_{v \min} & y_{v \min} & 1 \end{bmatrix}$$

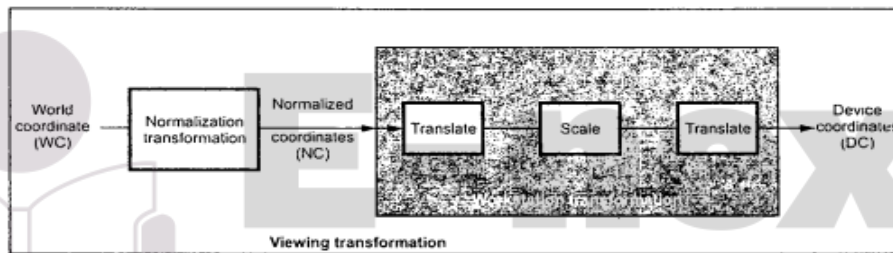
The overall transformation matrix for W is given as

$$W = T \cdot S \cdot T^{-1}$$

$$= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -X_{w \min} & -Y_{w \min} & 1 \end{bmatrix} \begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ x_{v \min} & y_{v \min} & 1 \end{bmatrix}$$

$$= \begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ x_{v \min} - x_{w \min} \cdot S_x & y_{v \min} - y_{w \min} \cdot S_y & 1 \end{bmatrix}$$

The Fig. 5.7 shows the complete viewing transformation.



b) Using homogeneous coordinate transformation matrix, apply following sequence of transformation to a unit square centered at origin.

- i) Translation by factor(1,1)
- ii) Rotation by angle $\theta=90^\circ$

$$T_T = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 1 & 1 \end{pmatrix}$$

$$T_R = \begin{pmatrix} \cos 90 & \sin 90 & 0 \\ -\sin 90 & \cos 90 & 0 \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

$$T = T_T \cdot T_R = \begin{pmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ -1 & 1 & 1 \end{pmatrix}$$

Unit square ABCD=

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

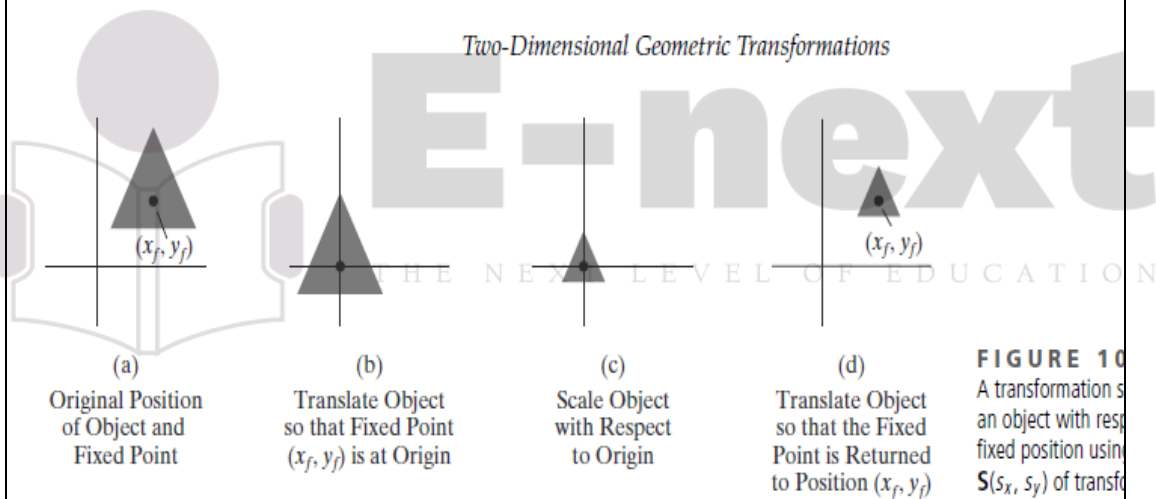
Calculate A',B',C',D' Multiply T with Unit square

A'=(-1/2,1/2) ,B'=(-3/2,3/2) ,C'=(-3/2,3/2), D'=(-3/2,1/2)

c Obtain the general combined matrix for scaling about an Fixed Point P(xf, yf).

scaling with respect to a selected fixed position (xf, yf), when we have a function that can scale relative to the coordinate origin only. This sequence is

1. Translate the object so that the fixed point coincides with the coordinate origin.



2. Scale the object with respect to the coordinate origin.

3. Use the inverse of the translation in step (1) to return the object to its original position.

Concatenating the matrices for these three operations produces the required scaling matrix:

$$\begin{bmatrix} 1 & 0 & x_f \\ 0 & 1 & y_f \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & -x_f \\ 0 & 1 & -y_f \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} s_x & 0 & x_f(1 - s_x) \\ 0 & s_y & y_f(1 - s_y) \\ 0 & 0 & 1 \end{bmatrix} \quad (37)$$

or

$$\mathbf{T}(x_f, y_f) \cdot \mathbf{S}(s_x, s_y) \cdot \mathbf{T}(-x_f, -y_f) = \mathbf{S}(x_f, y_f, s_x, s_y)$$

d Write a note on Affine and perspective geometry.

- Affine geometry is a geometry which is considered as the study of parallel lines.
- In affine geometry lines are generally parallel whereas in perspective geometry lines are not parallel.
- Affine transformations can be a combination of linear transformation i.e. it can be achieved by combining rotation, translation, scaling, shearing together.
- Affine means "connected with" or "connected"
- In affine transformation matrix the value of $h = 1$ implies the last column of 4×4 matrix is $[0 \ 0 \ 0 \ 1]$ here the transformed homogeneous coordinate is unity.
- Affine transformation is a function between affine spaces, that preserves points, straight lines, and planes. Set of parallel lines remains same after an affine transformation but it does not preserve angles between the lines or distance between the points.
- It forms a subset of bilinear transformations. Since product of two affine transformations is also affine.

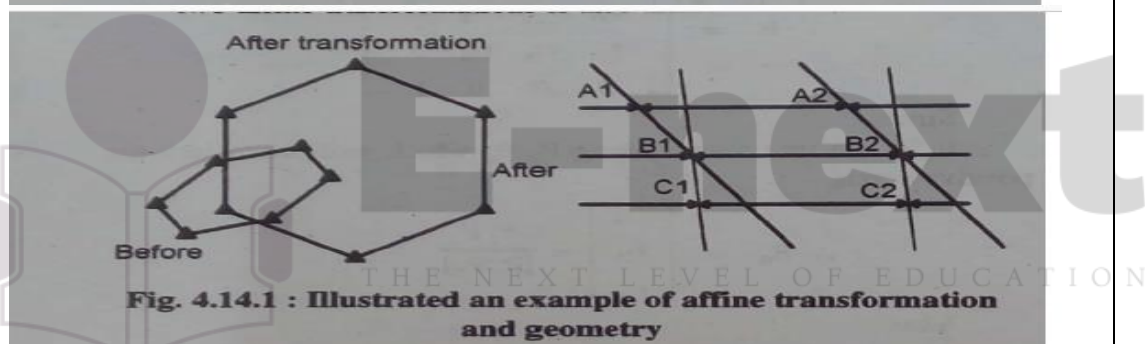
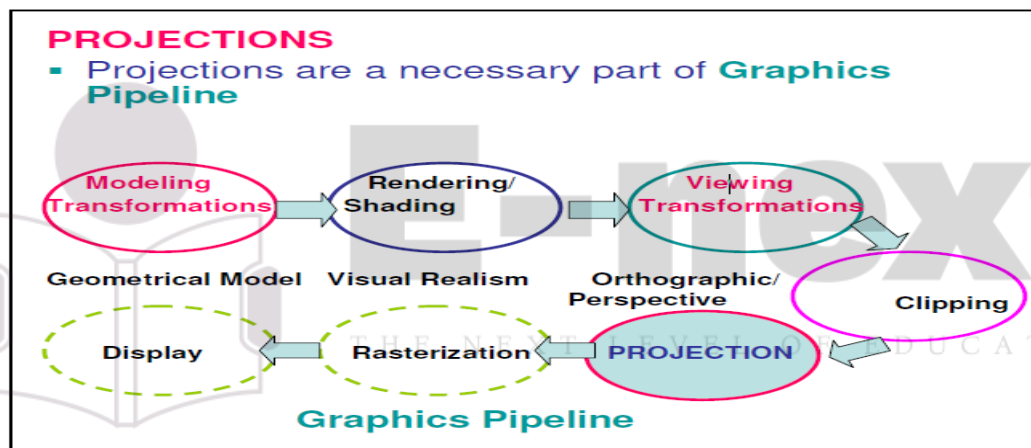


Fig. 4.14.1 : Illustrated an example of affine transformation and geometry

- Affine geometry is identical to Euclidean geometry and is generally used for graphical communication. Although perspective views or perspective geometry is used by artists and architects to generate more realistic pictures.
- Affine and perspective transformations, both are three dimensional. They can have transformations from one three dimensional space to another three dimensional space.
- The transformations which transforms a 3D object into a 2D object by projecting it onto a plane is called as projections.
- Perspective projection is defined by a center of projection and a view plane onto which a 2D image of the 3D object is projected.

- If the center of projection is located at a finite point in 3D space then it is called as perspective projection.
- We have another type of perspective projection in which the center of projection has been moved to infinity which is called as parallel projection.
- Here, rather than a center of projection, the direction of the parallel projectors defines the projection together with the view plane.

- e Explain Projection with the help of Orthographic projection.
- Viewing 3D objects on a 2D display requires a mapping from 3D to 2D
- A projection is formed by the intersection of certain lines (*projectors*) with the view plane
 - Projectors are lines from the *center of projection* through each point in the object.
- _ This is known as **plane geometric projection**



Orthographic (or orthogonal) projections:

- **Front, side and rear** orthographic projection of an object are called **elevations** and the **top** orthographic projection is called **plan view**.
- all have projection plane perpendicular to a principle axes.
- Here length and angles are accurately depicted and measured from the drawing, so engineering and architectural drawings commonly employ this.
- However, As only one face of an object is shown, it can be hard to create a mental image of the object, even when several views are available.

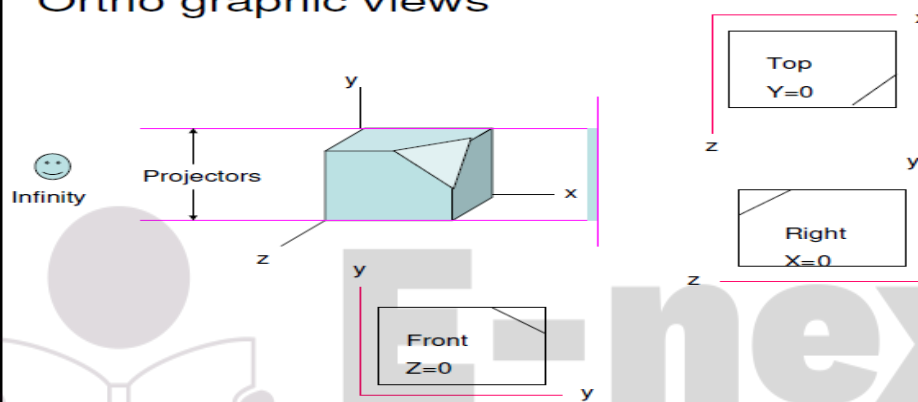
➤ Orthographic projection matrices

$$[T_z] = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}; [T_y] = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}; [T_x] = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

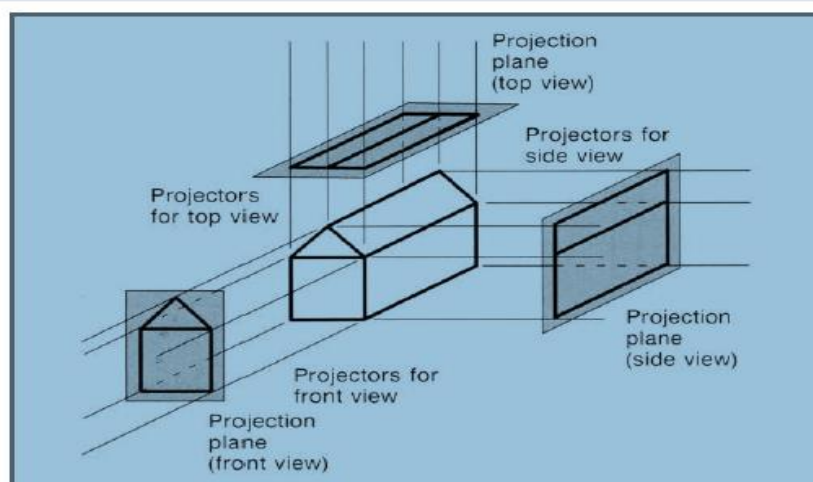
Orthographic Views

View	C.O.Projection	Proj. Plane
Front	On +ve z axis	Z=0 (xy)
Right Side	On +ve x axis	X=0 (yz)
Top	On +ve y axis	Y=0 (xz)
Rear	On -ve z axis	Z=0 (xy)
Left Side	On -ve x axis	X=0 (yz)
Bottom	On -ve y axis	Y=0 (xz)

Ortho graphic views

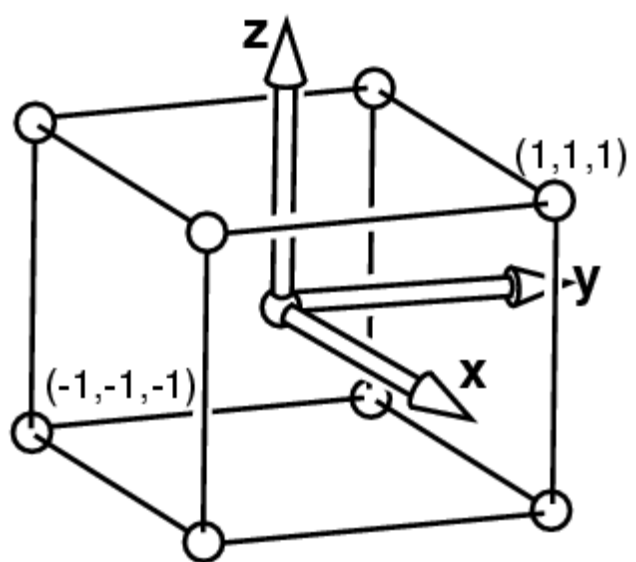


Orthogonal projections:



f. Shear a unit cube situated at origin with a shear transformation matrix:

$$T_{\text{shear}} = \begin{bmatrix} 1 & 1.5 & 3 & 0 \\ 0.8 & 0 & 1 & 0 \\ 0.5 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

<p>Soution</p> $V = \begin{pmatrix} A \\ B \\ C \\ D \\ E \\ F \\ G \\ H \end{pmatrix} = \begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}$ <p>To find output: Multiply $V' = V \cdot T_{\text{shear}}$</p>	
<p>3 Attempt <u>any three</u> of the following:</p>	<p>1 5</p>
<p>a What is Viewing? Explain Canonical View Volume.</p> <p>The primary use of clipping in computer graphics is to remove objects, lines, or line segments that are outside the viewing pane. The viewing transformation is insensitive to the position of points relative to the viewing volume – especially those points behind the viewer – and it is necessary to remove these points before generating the view.</p> <p>View Volume</p> <ul style="list-style-type: none"> Given the specification of the <i>view window</i>, we can set up a <i>View Volume</i>. • Only objects inside the view volume might appear in the display, the rest are clipped. <p>Canonical View Volumes</p> <ul style="list-style-type: none"> □ Canonical View Space: A cube, with the origin at the center, the viewer looking down $-z$, x to the right, and y up □ Canonical View Volume is the cube: $[-1,1] \times [-1,1] \times [-1,1]$ □ Variants (later) with viewer looking down $+z$ and z from 0-1 □ Only things that end up inside the canonical volume can appear in the window 	
<p>b Explain camera model and viewing pyramid with diagram.</p>	

To understand the process of image generation from a 2D or 3D model i.e. a camera we first need to understand that model. The following section describes the camera model and its viewing pyramid.

☞ The camera model

We specify our initial camera model by identifying the following parameters.

1. A scene, consisting of polygonal elements each represented by their vertices,
2. A point that represents the camera position – $C = (x_c, y_c, z_c)$,

3. A point that represents the “center-of-attention” of the camera (i.e. where the camera is looking) – $A = (x_a, y_a, z_a)$,
4. A field-of-view angle, α , representing the angle subtended at the apex of the viewing pyramid.
5. The specification of “near” and “far” bounding planes. These planes are considered perpendicular to the direction-of-view vector at a distance of n and f from the camera, respectively.

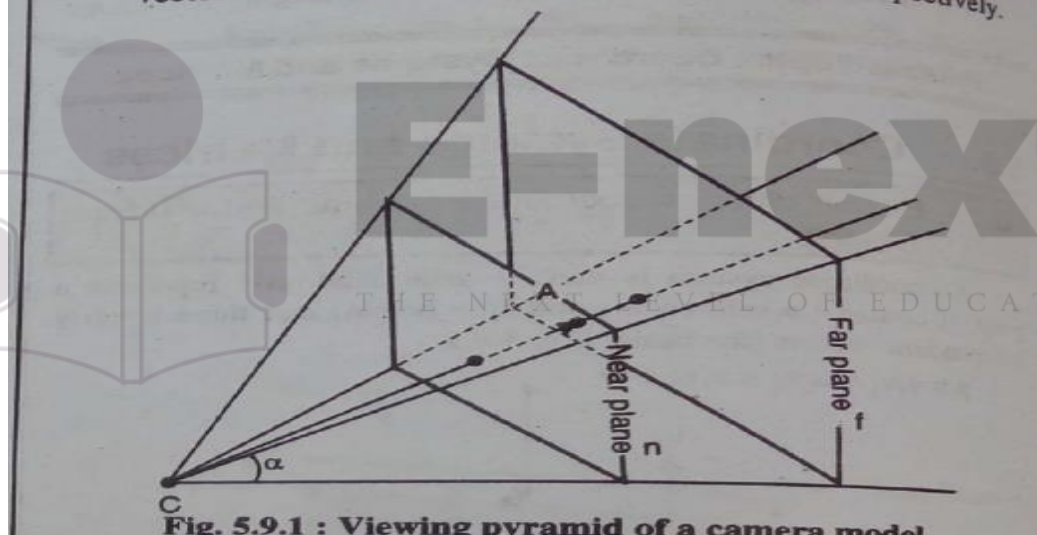


Fig. 5.9.1 : Viewing pyramid of a camera model

The specification of C , A and α forms a viewing volume in the shape of a pyramid with the camera position C at the apex of the pyramid and the vector $A - C$ forming the axis of the pyramid. This pyramid is commonly referred to as the viewing pyramid. The specification of the near and far planes forms a truncated viewing pyramid which gives the region of space which contains the primary portion of the scene to be viewed. The viewing transform, transforms this truncated pyramid onto the image space volume – $-1 \leq x, y, z \leq 1$.

c Explain different properties of BRDF.

Points to be covered:

- Domain
- Range
- Reciprocity

- Energy Conservation

d Write a note on Photometry.

For every spectral radiometric quantity there is a related *photometric quantity* that measures how much of that quantity is “useful” to a human observer. Given a spectral radiometric quantity $f_r(\lambda)$, the related photometric quantity f_p is

$$f_p = 683 \frac{\text{lm}}{\text{W}} \int_{\lambda=380 \text{ nm}}^{800 \text{ nm}} \bar{y}(\lambda) f_r(\lambda) d\lambda,$$

where \bar{y} is the *luminous efficiency function* of the human visual system. This function is zero outside the limits of integration above, so the limits could be 0 and ∞ and f_p would not change. The luminous efficiency function will be

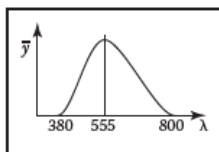


Figure 18.8. The luminous efficiency function versus wavelength (nm).

The luminous efficiency function is not equally sensitive to all wavelengths (Figure 18.8). For wavelengths below 380 nm (the *ultraviolet range*), the light is not visible to humans and thus has a \bar{y} value of zero. From 380 nm it gradually increases until $\lambda = 555$ nm where it peaks. This is a pure green light. Then, it gradually decreases until it reaches the boundary of the infrared region at 800 nm.

The photometric quantity that is most commonly used in graphics is *luminance*, the photometric analog of radiance:

$$Y = 683 \frac{\text{lm}}{\text{W}} \int_{\lambda=380 \text{ nm}}^{800 \text{ nm}} \bar{y}(\lambda) L(\lambda) d\lambda.$$

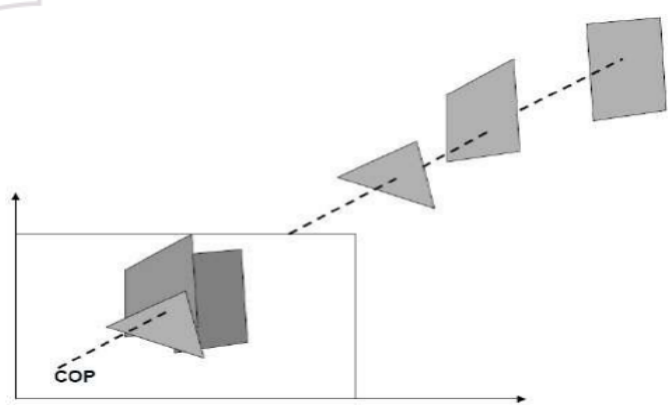
The symbol Y for luminance comes from colorimetry. Most other fields use the symbol L ; we will not follow that convention because it is too confusing to use L for both luminance and spectral radiance. Luminance gives one a general idea of how “bright” something is independent of the adaptation of the viewer. Note that the black paper under noonday sun is subjectively darker than the lower luminance white paper under moonlight; reading too much into luminance is dangerous, but it is a very useful quantity for getting a quantitative feel for relative perceivable light output. The unit *lm* stands for *lumens*. Note that most light bulbs are rated in terms of the power they consume in watts, and the useful light they produce in lumens. More efficient bulbs produce more of their light where \bar{y} is large and thus produce more lumens per watt. A “perfect” light would convert all power into 555 nm light and would produce 683 lumens per watt. The units of luminance are thus $(\text{lm}/\text{W})(\text{W}/(\text{m}^2\text{sr})) = \text{lm}/(\text{m}^2\text{sr})$. The quantity one lumen per steradian is defined to be one *candela* (*cd*), so luminance is usually described in units cd/m^2 .

e Explain Grassmann’s laws.

Given that humans have three different cone types, the experimental laws of color matching can be summed up as the *trichromatic generalization* (Wyszecki & Stiles, 2000), which states that any color stimulus can be matched completely with an additive mixture of three appropriately modulated color sources. This feature of color is often used in practice, for instance by televisions and monitors which reproduce many different colors by adding a mixture of red, green, and blue light for each pixel. It is also the reason that renderers can be built using only three values to describe each color.

The trichromatic generalization allows us to make color matches between any given stimulus and an additive mixture of three other color stimuli. Hermann Grassmann was the first to describe the algebraic rules to which color matching adheres. They are known as **Grassmann’s laws** of additive color matching (Grassmann, 1853) and are the following:

- **Symmetry law.** If color stimulus A matches color stimulus B , then B matches A .
- **Transitive law.** If A matches B and B matches C , then A matches C .
- **Proportionality law.** If A matches B , then αA matches αB , where α is a positive scale factor.

	<ul style="list-style-type: none"> • Additivity law. If A matches B, C matches D, and $A + C$ matches $B + D$, then it follows that $A + D$ matches $B + C$. <p>The additivity law forms the basis for color matching and colorimetry as a whole.</p>	
f	<p>Explain color with the help of colorimetry.</p> <p>Colorimetry is the science of color measurement and description. Since color is ultimately a human response, color measurement should begin with human observation. The photodetectors in the human retina consist of rods and cones. The rods are highly sensitive and come into play in low-light conditions. Under normal lighting conditions, the cones are operational, mediating human vision. There are three cone types and together they are primarily responsible for color vision.</p> <p>Although it may be possible to directly record the electrical output of cones while some visual stimulus is being presented, such a procedure would be invasive, while at the same time ignoring the sometimes substantial differences between observers. Moreover, much of the measurement of color was developed well before such direct recording techniques were available.</p>	
4	<p>Attempt <u>any three</u> of the following:</p>	<p>1 5</p>
a	<p>Explain Z-buffer algorithm with advantages and disadvantages.</p> <div> <p>Depth Buffer Z - Buffer Method</p> <p>This method is developed by Cutmull. It is an image-space approach. The basic idea is to test the Z-depth of each surface to determine the closest <i>visible</i> surface.</p> <p>In this method each surface is processed separately one pixel position at a time across the surface. The depth values for a pixel are compared and the closest <i>smallestz</i> surface determines the color to be displayed in the frame buffer.</p> <p>It is applied very efficiently on surfaces of polygon. Surfaces can be processed in any order. To override the closer polygons from the far ones, two buffers named frame buffer and depth buffer, are used.</p> <p>Depth buffer is used to store depth values for x, y position, as surfaces are processed $0 \leq \text{depth} \leq 1$.</p> <p>The frame buffer is used to store the intensity value of color value at each position x, y.</p> <p>The z-coordinates are usually normalized to the range $[0, 1]$. The 0 value for z-coordinate indicates back clipping plane and 1 value for z-coordinates indicates front clipping plane.</p>  </div>	

Algorithm

Step-1 – Set the buffer values –

Depthbuffer $x, y = 0$

Framebuffer $x, y = \text{background color}$

Step-2 – Process each polygon *Oneatatime*

For each projected x, y pixel position of a polygon, calculate depth z .

If $Z > \text{depthbuffer } x, y$

Compute surface color,

set depthbuffer $x, y = z$,

framebuffer $x, y = \text{surfacecolor } x, y$

Advantages

- It is easy to implement.
- It reduces the speed problem if implemented in hardware.
- It processes one object at a time.

Disadvantages

- It requires large memory.
- It is time consuming process

b What are the basic tests in Warnock's algorithm? Explain.

Area sub-division method

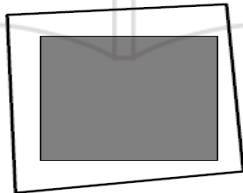
- Works in image-space
- Area Coherence is exploited
- Divide-and-conquer strategy.

Examine and divide if necessary

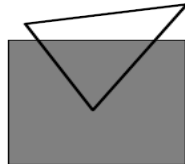
WARNOCK's Algorithm

Possible relationships of the area of interest (rectangular) and the projection of the polygon:

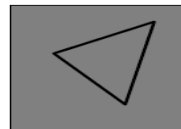
1. Surrounding polygons are completely inside the area of interest.
2. Intersecting polygons intersect the area.
3. Contained polygons are completely inside the area
4. Disjoint polygons area completely outside the area.



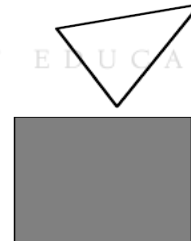
Surrounding



Intersecting



Contained



Disjoint

Treat interior part of the intersection and contained as equivalent. Disjoint is irrelevant. The decisions about division of an area is based on:

1. All polygons are disjoint from the area. Display background color.
2. Only one Intersecting part or interior (contained): Fill the area with background color and then scan-convert the polygon.
3. Single surrounding polygon, and no intersecting or contained polygon. Use color of surrounding polygon to shade the area.
4. More than one polygon intersect, contained in and surrounding the area. But the surrounding polygon is in front of all other polygons. Then fill the area with the color of the surrounding polygon.

Step 4 is implemented by comparing the Z-coordinates of the planes of all polygons (involved) at the four corners of the area

If all four tests fail, divide the rectangular area into four equal parts. Recursively apply this logic, till you reach the lowest minimum area or maximum resolution – pixel size. Use nearest surface in that case to paint/shade.

c Explain Parametric representation of Ellipse with example.

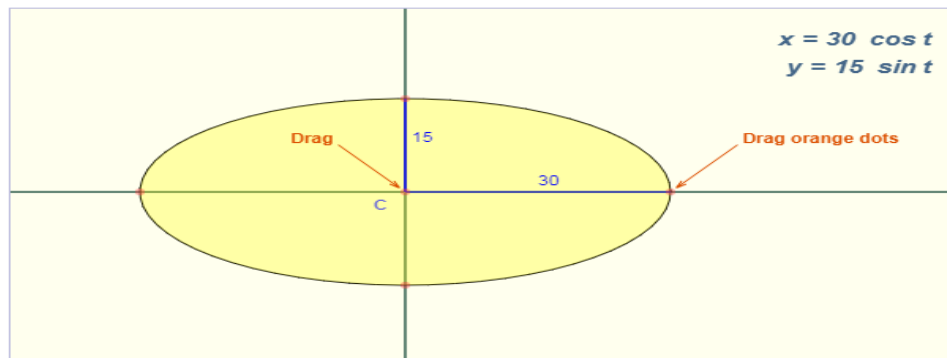
Parametric Equation of an Ellipse

An ellipse can be defined as the *locus* of all points that satisfy the equations

$$\begin{aligned}x &= a \cos t \\y &= b \sin t\end{aligned}$$

where:

x, y are the coordinates of any point on the ellipse,
 a, b are the radius on the x and y axes respectively, (* See radii notes below)
 t is the parameter, which ranges from 0 to 2π radians.



This equation is very similar to the one used to define a circle, and much of the discussion is omitted here to avoid duplication. See [Parametric equation of a circle](#) as an introduction to this topic.

The only difference between the circle and the ellipse is that in a circle there is one radius, but an ellipse has two:

- One radius is measured along the x -axis and is usually called a .
- The other is measured along the y -axis and is usually called b .



For a circle both these radii have the same value.

Ellipses centered at the origin

If the ellipse is centered on the origin (0,0) the equations are

$$\begin{aligned}x &= a \cos t \\y &= b \sin t\end{aligned}$$

where

a is the radius along the x -axis (* See radii notes below)
 b is the radius along the y -axis

Ellipses not centered at the origin

Just as with the [circle equations](#), we add offsets to the x and y terms to translate (or "move") the ellipse to the correct location. So the full form of the equations are

$$\begin{aligned}x &= h+a \cos t \\y &= k+b \sin t\end{aligned}$$

where, as before

a is the radius along the x -axis (* See radii note below)

b is the radius along the y -axis

(h,k) are the x and y coordinates of the ellipse's center.

In the applet above, drag the orange dot at the center to move the ellipse, and note how the equations change to match. Also, adjust the ellipse so that a and b are the same length, and convince yourself that in this case, these are the same equations as for a circle.

d Write a note on B-Spline Curves.

This spline category is the most widely used, and B-spline functions are commonly available in CAD systems and many graphics-programming packages. Like Bézier splines, B-splines are generated by approximating a set of control points. But B-splines have two advantages over Bézier splines: (1) the degree of a B-spline polynomial can be set independently of the number of control points (with certain limitations), and (2) B-splines allow local control over the shape of a spline. The tradeoff is that B-splines are more complex than Bézier splines.

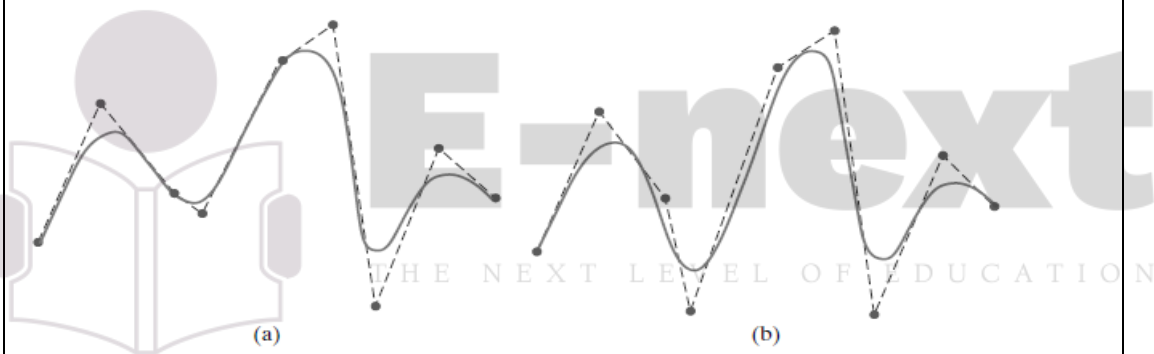
B-Spline Curve Equations

We can write a general expression for the calculation of coordinate positions along a B-spline curve using a blending-function formulation as

$$P(u) = \sum_{k=0}^n p_k B_{k,d}(u), \quad u_{\min} \leq u \leq u_{\max}, \quad 2 \leq d \leq n+1 \quad (37)$$

where p_k is an input set of $n+1$ control points. There are several differences between this B-spline formulation and the expression for a Bézier spline curve. The range of parameter u now depends on how we choose the other B-spline parameters. And the B-spline blending functions $B_{k,d}$ are polynomials of degree $d-1$, where d is the degree parameter. (Sometimes parameter d is alluded to as the "order" of the polynomial, but this can be misleading because the term order is also often used to mean simply the degree of the polynomial.) The degree parameter d can be assigned any integer value in the range from 2 up to the number of control points ($n+1$). Actually, we could also set the value of the degree parameter at 1, but then our "curve" is just a point plot of the control points. Local control for B-splines is achieved by defining the blending functions over subintervals of the total range of u .

Spline Representations



e Compare all visible surface detection methods.

Comparison of VSD (HSR) techniques

Algorithms/ Methods	Memory	Speed	Issues in Implementation	Remarks
Z-Buffer	Two arrays	Depth complexity	Scan conversion, Hardware	Commonly used
Painter's	One array	Apriori sorting helps speed-up	Scan conversion	Splitting and sorting the major bottleneck
Ray casting	Object database	$O(\#pixels,$ $\#surfaces \text{ or } objects)$	Spatial data structures help speedup	Excellent for CSG, shadows, transparency
Scanline, Area sub- division			Hard	Cannot be generalized for non- polygonal models

- f) Construct Bezier curve of order 3, with 4 polygon vertices A(1,1) B(2,3) C(4,3) D(6,4) for values of $u, 0 \leq u \leq 1$ where $p(u)$ is a point on curve with values for $u = (0, 1/4, 1/2, 3/4, 1)$.

Solution

The equation for curve is

$$P(u) = (1-u)^3 P_1 + 3u(1-u)^2 P_2 + 3u^2(1-u) P_3 + u^3 P_4$$

where $P(u)$ is the point on the curve P_1, P_2, P_3, P_4 .

Let us take $u = 0, \frac{1}{4}, \frac{1}{2}, \frac{3}{4}$

$\therefore P(0) = P_1 = (1, 1)$

$\therefore P\left(\frac{1}{4}\right) = \left(1 - \frac{1}{4}\right)^3 P_1 + 3\left(\frac{1}{4}\right)\left(1 - \frac{1}{4}\right)^2 P_2 + 3\left(\frac{1}{4}\right)^2\left(1 - \frac{1}{4}\right) P_3 + \left(\frac{1}{4}\right)^3 P_4$

$$= \frac{27}{64}(1, 1) + \frac{27}{64}(2, 3) + \frac{9}{64}(4, 3) + \frac{1}{64}(6, 4)$$

$$= \left[\frac{27}{64} \times 1 + \frac{27}{64} \times 2 + \frac{9}{64} \times 4 + \frac{1}{64} \times 6, \frac{27}{64} \times 1 + \frac{27}{64} \times 3 + \frac{9}{64} \times 3 + \frac{1}{64} \times 4 \right]$$

$$= \left[\frac{123}{64}, \frac{139}{64} \right]$$

$$= (1.9218, 2.1718)$$

$\therefore P\left(\frac{1}{2}\right) = \left(1 - \frac{1}{2}\right)^3 P_1 + 3\left(\frac{1}{2}\right)\left(1 - \frac{1}{2}\right)^2 P_2 + 3\left(\frac{1}{2}\right)^2\left(1 - \frac{1}{2}\right) P_3 + \left(\frac{1}{2}\right)^3 P_4$

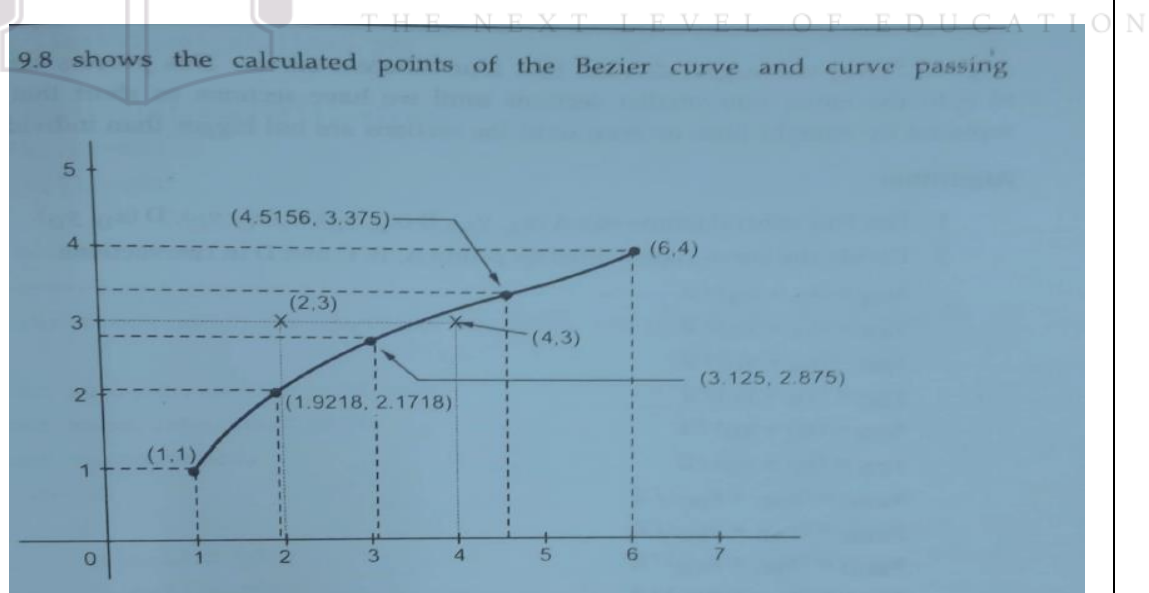
$$= \frac{1}{8}(1, 1) + \frac{3}{8}(2, 3) + \frac{3}{8}(4, 3) + \frac{1}{8}(6, 4)$$

$$= \left[\frac{1}{8} \times 1 + \frac{3}{8} \times 2 + \frac{3}{8} \times 4 + \frac{1}{8} \times 6, \frac{1}{8} \times 1 + \frac{3}{8} \times 3 + \frac{3}{8} \times 3 + \frac{1}{8} \times 4 \right]$$

$$= \left[\frac{25}{8}, \frac{23}{8} \right]$$

$$= (3.125, 2.875)$$

$\therefore P\left(\frac{3}{4}\right) = \left(1 - \frac{3}{4}\right)^3 P_1 + 3\left(\frac{3}{4}\right)\left(1 - \frac{3}{4}\right)^2 P_2 + 3\left(\frac{3}{4}\right)^2\left(1 - \frac{3}{4}\right) P_3 + \left(\frac{3}{4}\right)^3 P_4$



5 Attempt any three of the following:

1
5

- a) What is an Image? Explain different file formats of an image.
An image is a visual representation of something. In information technology, the term has several usages: An image is a picture that has been created or copied and stored in electronic form. An image can be described in terms of vector graphics or raster graphics. An image stored in raster form is sometimes called

a bitmap. An image map is a file containing information that associates different locations on a specified image with hypertext links.

Formats:

- JPEG
- GIF
- PNG
- BMP
- TIFF ,etc...

b Explain Animation with Character Animation.

computer animation generally refers to any time sequence of visual changes in a picture. In addition to changing object positions using translations or rotations, a computer-generated animation could display time variations in object size, color, transparency, or surface texture. Advertising animations often transition one object shape into another: for example, transforming a can of motor oil into an automobile engine. We can also generate computer animations by varying camera parameters, such as position, orientation, or focal length, and variations in lighting effects or other parameters and procedures associated with illumination and rendering can be used to produce computer animations.

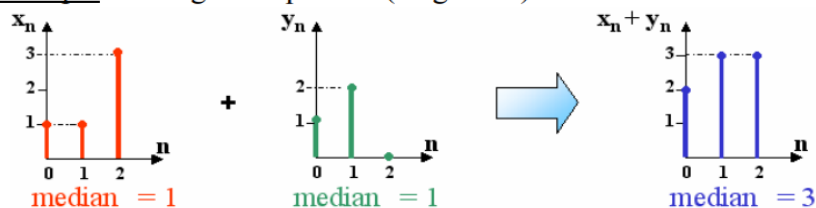
Character animation is a specialized area of the animation process, which involves bringing animated **characters** to life. The role of a Character Animator is analogous to that of a film or stage actor, and character animators are often said to be "actors with a pencil" (or a mouse). Character animators breathe life in their characters, creating the illusion of thought, emotion and personality. Character animation is often distinguished from creature animation, which involves bringing photo-realistic animals and creatures to life.

c Explain the concept of median filtering with example.

➤ **Median filtering is a non-linear operation:**

Generally: $\text{median} \{ x_m + y_m \} \neq \text{median} \{ x_m \} + \text{median} \{ y_m \}$

▪ example with signal sequences (length = 3):



➤ **Odd window sizes are commonly used in median filtering:**

3×3 ; 5×5 ; 7×7

▪ example: 5-pixel cross-shaped window



➤ **For even-sized windows (2 K values): the filter sorts the values then takes the average of the two middle values :**

$$\text{Output value} = \frac{(\text{K}^{\text{th}} \text{ classified value} + (\text{K}+1)^{\text{th}} \text{ classified value})}{2}$$

- 3×3 original image:

91	55	90
77	68	95
115	151	210

- Median filtering using the full 3×3 neighborhood:

↳ we sort the original image values on the full 3×3 window:
55 , 68 , 77 , 90 , **91** , 95 , 115 , 151 , 210

median value = 91

	55	
77	68	95
	151	

- Median filtering using 5-pixel cross-shaped size:

↳ We sort the original image values on the 3×3 cross window:
55 , 68 , **77** , 95 , 151

median value = 77

- d Distinguish Key frame animation with Procedural animation.

Key frame animation

- The original way to animate, and still the most common form for feature animation
 - Process has shifted to computers, but basic approach is the same
- Underlying technique is *interpolation*
 - The in-between frames are interpolated from the keyframes
 - Originally done by armies of underpaid animators
 - Now done with computers
 - Which of the techniques that we have learned about is used extensively for keyframe animation?

Procedural Animation

- Animation is generated by writing a program that spits out the position/shape/whatever of the scene over time
- Generally:
 - Program some rules for how the system will behave
 - Choose some initial conditions for the world
 - Run the program, maybe with user input to guide what happens
- Advantage: Once you have the program, you can get lots of motion
- Disadvantage: The animation is generally hard to control, which makes it hard to tell a story with purely procedural means

- e Explain different types of deformation.

- Bone Deformer(skeleton deformation)
- Curve Deformer(Skinning deformation)

- f Explain JPEG Compression process in detail.

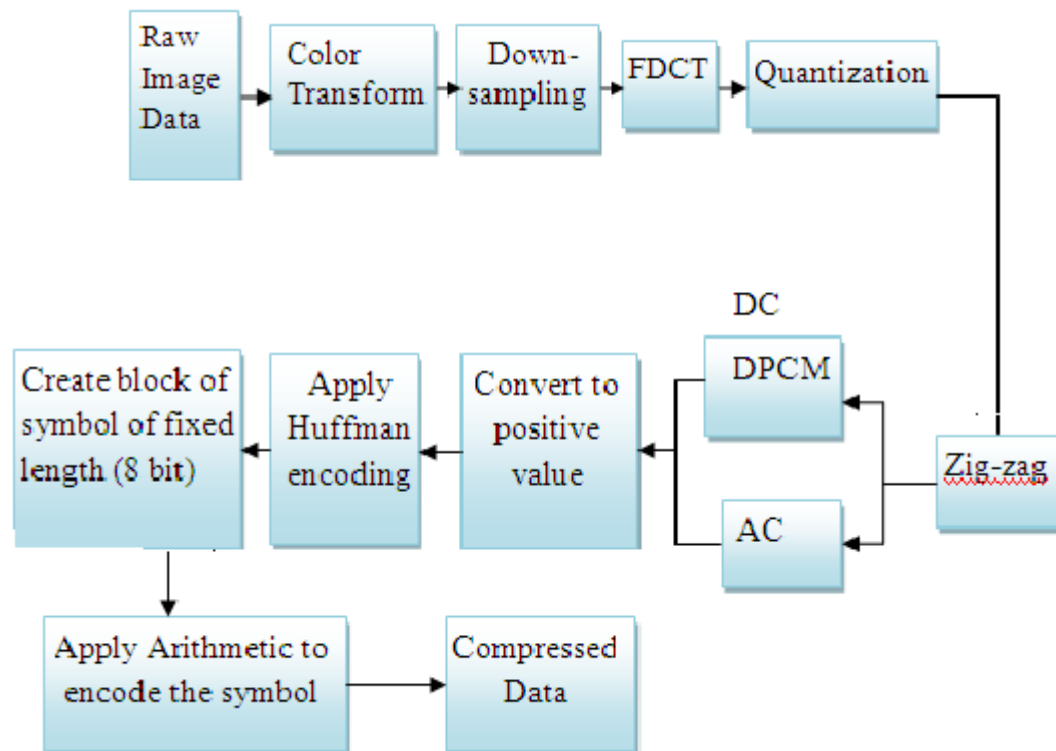


Image compression

Image compression is the method of data compression on digital images.

The main objective in the image compression is:

- Store data in an efficient form
- Transmit data in an efficient form

Image compression can be lossy or lossless.

JPEG compression

JPEG stands for Joint photographic experts group. It is the first international standard in image compression. It is widely used today. It could be lossy as well as lossless. But the technique we are going to discuss here today is lossy compression technique.

How jpeg compression works

First step is to divide an image into blocks with each having dimensions of 8 x8.

Let's for the record, say that this 8x8 image contains the following values.

52	55	61	66	70	61	64	73
63	59	55	90	109	85	69	72
62	59	68	113	144	104	66	73
63	58	71	122	154	106	70	69
67	61	68	104	126	88	68	70
79	65	60	70	77	68	58	75
85	71	64	50	55	61	65	82

80	71	64	57	50	61	60	60
87	79	69	63	65	76	78	94

The range of the pixels intensities now are from 0 to 255. We will change the range from -128 to 127.

Subtracting 128 from each pixel value yields pixel value from -128 to 127. After subtracting 128 from each of the pixel value, we got the following results.

$$\begin{bmatrix} -76 & -73 & -67 & -62 & -58 & -67 & -64 & -55 \\ -65 & -69 & -73 & -38 & -19 & -43 & -59 & -56 \\ -66 & -69 & -60 & -15 & 16 & -24 & -62 & -55 \\ -65 & -70 & -57 & -6 & 26 & -22 & -58 & -59 \\ -61 & -67 & -60 & -24 & -2 & -40 & -60 & -58 \\ -49 & -63 & -68 & -58 & -51 & -60 & -70 & -53 \\ -43 & -57 & -64 & -69 & -73 & -67 & -63 & -45 \\ -41 & -49 & -59 & -60 & -63 & -52 & -50 & -34 \end{bmatrix}$$

Now we will compute using this formula.

$$G_{u,v} = \alpha(u)\alpha(v) \sum_{x=0}^7 \sum_{y=0}^7 g_{x,y} \cos \left[\frac{\pi}{8} \left(x + \frac{1}{2} \right) u \right] \cos \left[\frac{\pi}{8} \left(y + \frac{1}{2} \right) v \right]$$

$$\alpha_p(n) = \begin{cases} \sqrt{\frac{1}{5}}, & \text{if } n = 0 \\ \sqrt{\frac{2}{5}}, & \text{otherwise} \end{cases}$$

The result comes from this is stored in let's say $A_{j,k}$ matrix.

There is a standard matrix that is used for computing JPEG compression, which is given by a matrix called as Luminance matrix.

This matrix is given below

$$Q_{j,k} = \begin{bmatrix} 16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\ 12 & 12 & 14 & 19 & 26 & 58 & 60 & 55 \\ 14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\ 24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \end{bmatrix}$$

$$\begin{bmatrix} 49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\ 72 & 92 & 95 & 98 & 112 & 100 & 103 & 99 \end{bmatrix}$$

Applying the following formula

$$B_{j,k} = \text{round} \left(\frac{A_{j,k}}{Q_{j,k}} \right)$$

We got this result after applying.

$$B_{j,k} = \begin{bmatrix} -26 & -3 & -6 & 2 & 2 & -1 & 0 & 0 \\ 0 & -2 & -4 & 1 & 1 & 0 & 0 & 0 \\ -3 & 1 & 5 & -1 & -1 & 0 & 0 & 0 \\ -4 & 1 & 2 & -1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Now we will perform the real trick which is done in JPEG compression which is ZIG-ZAG movement. The zig zag sequence for the above matrix is shown below. You have to perform zig zag until you find all zeroes ahead. Hence our image is now compressed.

$$B_{j,k} = \begin{bmatrix} -26 & -3 & -6 & 2 & 2 & -1 & 0 & 0 \\ 0 & -2 & -4 & 1 & 1 & 0 & 0 & 0 \\ -3 & 1 & 5 & -1 & -1 & 0 & 0 & 0 \\ -4 & 1 & 2 & -1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$



E-next

THE NEXT LEVEL OF EDUCATION