Parshvanath Charitable Trust's

A. P. SHAH INSTITUTE OF TECHNOLOGY

(Approved by AICTE New Delhi & Govt. of Maharashtra, Affiliated to University of Mumbai)
(Religious Jain Minority)

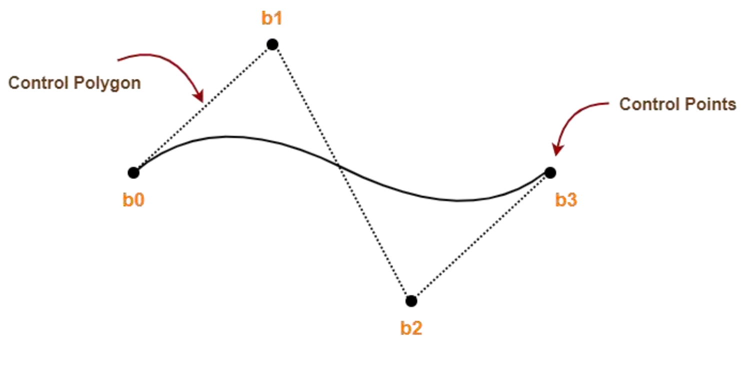
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

DATA SCIENCE

UNIT TEST-II SOLUTION

Class: SE **Semester: III** **Subject: Computer Graphics**

Max marks: 40

Q.N	Questions	MARKS
Q. 1.	**Attempt any two.**	
i)	Explain what is meant by Bezier curve? State the various properties of Bezier curve Ans: Bezier Curve may be defined as- • Bezier Curve is parametric curve defined by a set of control points. • Two points are ends of the curve. • Other points determine the shape of the curve. The concept of bezier curves was given by Pierre Bezier. **Bezier Curve Example-** The following curve is an example of a bezier curve-  Bezier Curve Example	[5]

Here,

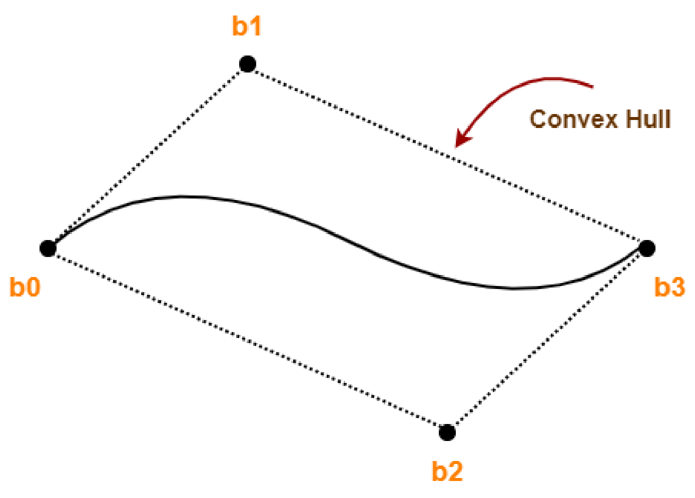
- This bezier curve is defined by a set of control points b_0, b_1, b_2 and b_3.
- Points b_0 and b_3 are ends of the curve.
- Points b_1 and b_2 determine the shape of the curve.

Bezier Curve Properties-

Few important properties of a bezier curve are-

Property-01:

Bezier curve is always contained within a polygon called as convex hull of its control points.



Bezier Curve With Convex Hull

Property-02:

- Bezier curve generally follows the shape of its defining polygon.
- The first and last points of the curve are coincident with the first and last points of the defining polygon.

Property-03:

The degree of the polynomial defining the curve segment is one less than the total number of control points.

Degree = Number of Control Points – 1

Property-04:

The order of the polynomial defining the curve segment is equal to the total number of control points.

Order = Number of Control Points

Property-05:

- Bezier curve exhibits the variation diminishing property.
- It means the curve do not oscillate about any straight line more often than the defining polygon.

Bezier Curve Equation-

A bezier curve is parametrically represented by-

$$P(t) = \sum_{i=0}^{n} B_i J_{n,i}(t)$$

Bezier Curve Equation

Here,

- t is any parameter where 0 <= t <= 1
- P(t) = Any point lying on the bezier curve
- B_i = i^{th} control point of the bezier curve
- n = degree of the curve
- $J_{n,i}(t)$ = Blending function = $C(n,i)t^i(1-t)^{n-i}$ where C(n,i) = n! / i!(n-i)!

ii) Write a short note on Fractals. Also explain how the fractal dimension is calculated [5]

A French/American mathematician Dr Benoit Mandelbrot discovered Fractals. The word fractal was derived from a Latin word fractus which means broken. Fractals are very complex pictures generated by a computer from a single formula. They are created using iterations. This means one formula is repeated with slightly different values over and over again, taking into account the results from the previous iteration.

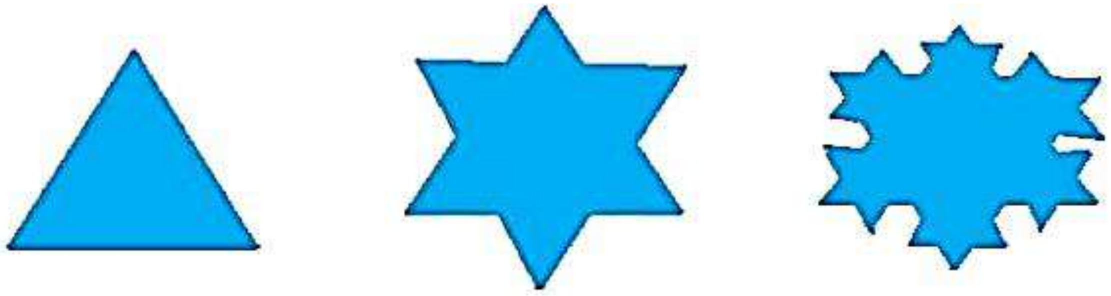
Fractals are used in many areas such as −

- Astronomy − For analyzing galaxies, rings of Saturn, etc.
- Biology/Chemistry − For depicting bacteria cultures, Chemical reactions, human anatomy, molecules, plants,
- Others − For depicting clouds, coastline and borderlines, data compression, diffusion, economy, fractal art, fractal music, landscapes, special effect, etc.

Generation of Fractals

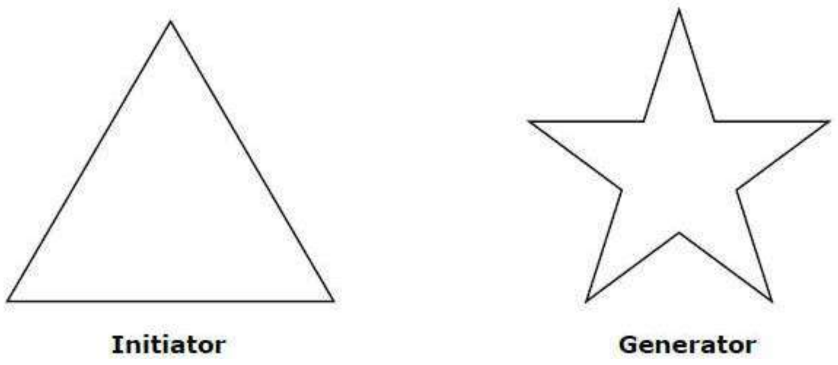
Fractals can be generated by repeating the same shape over and over again as shown in the following figure. In figure (a) shows an equilateral triangle. In figure (b), we can see that the triangle is repeated to create a star-like shape. In figure (c), we can see that the star shape in figure (b) is repeated again and again to create a new shape.

We can do unlimited number of iteration to create a desired shape. In programming terms, recursion is used to create such shapes.



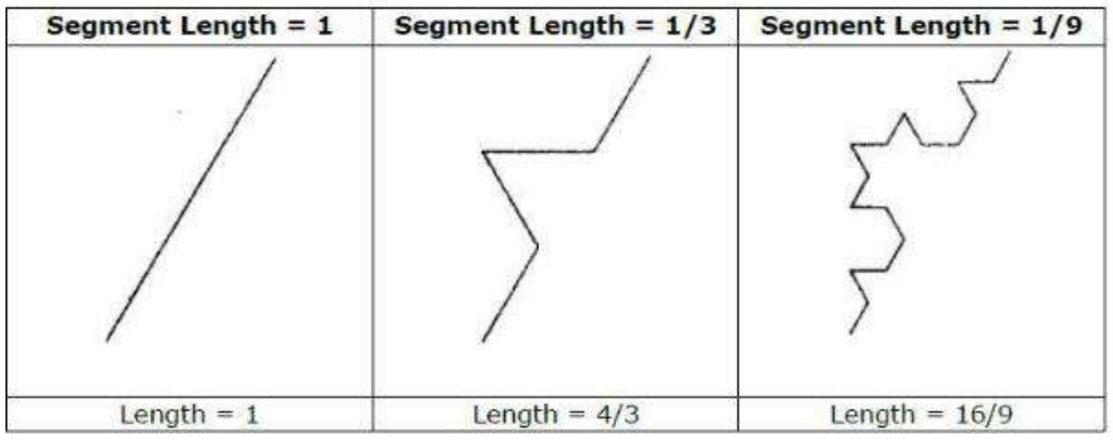
Geometric Fractals

Geometric fractals deal with shapes found in nature that have non-integer or fractal dimensions. To geometrically construct a deterministic (nonrandom) self-similar fractal, we start with a given geometric shape, called the initiator. Subparts of the initiator are then replaced with a pattern, called the generator.



As an example, if we use the initiator and generator shown in the above figure, we can construct good pattern by repeating it. Each straight-line segment in the initiator is replaced with four equal-length line segments at each step. The scaling factor is 1/3, so the fractal dimension is $D = \ln 4/\ln 3 \approx 1.2619$.

Also, the length of each line segment in the initiator increases by a factor of 4/3 at each step, so that the length of the fractal curve tends to infinity as more detail is added to the curve as shown in the following figure −



iii) What is meant by Parallel and Perspective Projection? Derive matrix for perspective projection [5]

Ans:

1. Projection operations convert the viewing-coordinate description (3D) to coordinate positions on the projection plane (2D).
2. There are 2 basic projection methods:

Parallel Projection:

i. In parallel projection, Z coordinate is discarded and parallel lines from each vertex on the object are extended until they intersect the view plane.

ii. The point of intersection is the projection of the vertex.

iii. We connect the projected vertices by line segments which correspond to connections on the original object.
iv. A parallel projection preserves relative proportions of objects.

v. Accurate views of the various sides of an object are obtained with a parallel projection. But not a realistic representation.

vi. Parallel projection is shown below in figure 30.

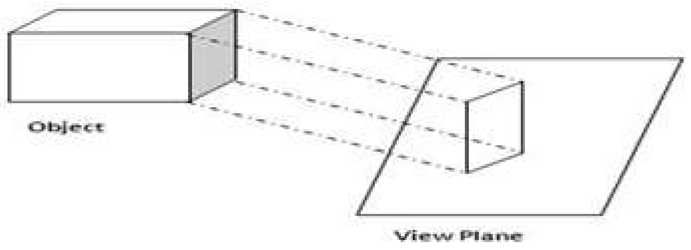


Figure 30

Perspective Projection:

i. In perspective projection, the lines of projection are not parallel.

ii. Perspective Projection transforms object positions to the view plane while converging to a center point of projection.

iii. In this all the projections are converge at a single point called the “center of projection” or “projection reference point”.

iv. Perspective projection produces realistic views but does not preserve relative proportions.

v. Projections of distant objects are smaller than the projections of objects of the same size that are closer to the projection plane.

vi. Perspective projection is shown below in figure 31

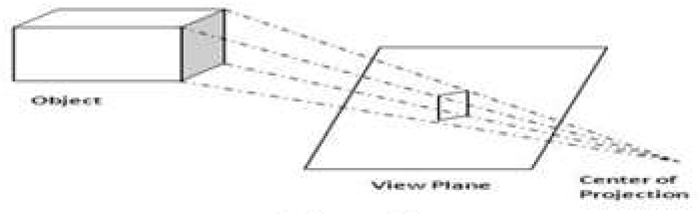


Figure 31

Matrix for perspective projection:

Let us consider the center of projection is at $P_c(X_c,Y_c,Z_c)$Pc(Xc,Yc,Zc) and the point on object is $P_1(X_1,Y_1,Z_1)$P1(X1,Y1,Z1), then the parametric equation for the line containing these points can be given as

$$X_2=X_c+(X_1-X_c)U \text{X2=Xc+(X1−Xc)U}$$

$$Y_2=Y_c+(Y_1-Y_c)U \text{Y2=Yc+(Y1−Yc)U}$$

$$Z_2=Z_c+(Z_1-Z_c)U \text{Z2=Zc+(Z1−Zc)U}$$

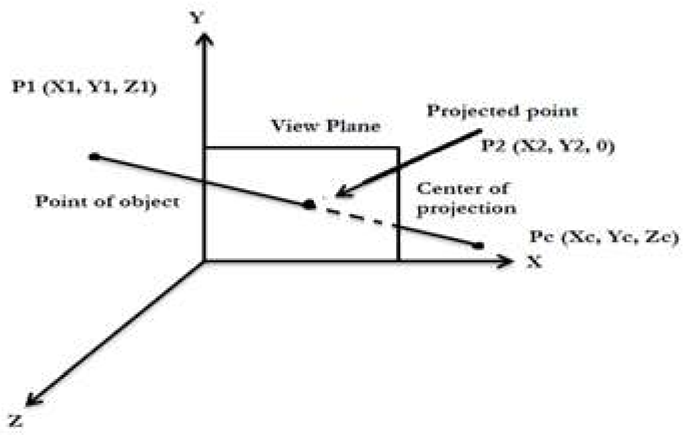
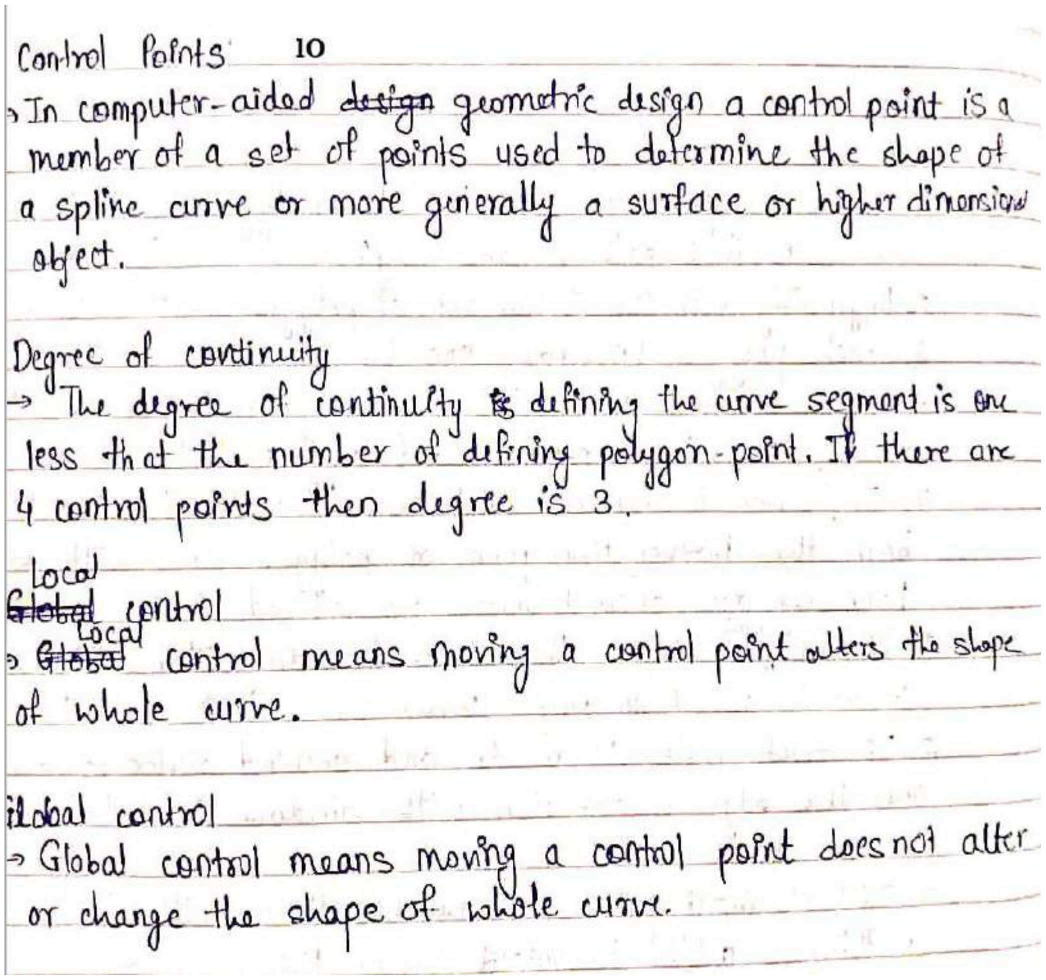


Figure 32

For projected point Z2 is 0, therefore the third equation can be written as

$0=Z_c+(Z_1-Z_c)U$0=Zc+(Z1−Zc)U
$U=-Z_c/Z_1-Z_c$U=−Zc/Z1−Zc
Substituting the value of U in first two equations we get,

$X2=(X_c–Z_c)*(X_1-X_c)/(Z_1-Z_c)$X2=(Xc–Zc)∗(X1−Xc)/(Z1−Zc)
$=X_CZ_1–X_CZ_C–X_1Z_C+X_CZ_C/Z_1-Z_C$=XCZ1–XCZC–X1ZC+XCZC/Z1−ZC
$=X_CZ_1–X_1Z_C/Z_1-Z_C$=XCZ1–X1ZC/Z1−ZC
$Y2=(Y_c–Z_c)*(Y_1-Y_c)/(Z_1-Z_c)$Y2=(Yc–Zc)∗(Y1−Yc)/(Z1−Zc)
$=Y_CZ_1–Y_CZ_C–Y_1Z_C+Y_CZ_C/Z_1-Z_C$=YCZ1–YCZC–Y1ZC+YCZC/Z1−ZC
$=Y_CZ_1–Y_1Z_C/Z_1-Z_C$=YCZ1–Y1ZC/Z1−ZC
The above equations can be represented in the homogeneous matrix form as given below:

	$[X_2\ Y_2\ Z_2\ 1] = [X_1\ Y_1\ Z_1\ 1]\begin{bmatrix} -Zc & 0 & 0 & 0 \\ 0 & -Zc & 0 & 0 \\ Xc & Yc & 0 & 1 \\ 0 & 0 & 0 & -Zc \end{bmatrix}\begin{bmatrix} -Zc & 0 & 0 & 0 \\ 0 & -Zc & 0 & 0 \\ Xc & Yc & 0 & 1 \\ 0 & 0 & 0 & -Zc \end{bmatrix}$ Here, we have taken the center of projection as $P_c(X_c,Y_c,Z_c)$Pc(Xc,Yc,Zc). If we take the center of projection on the negative Z – axis such that X = 0 Y = 0 $Z=-Z_c$	
iv)	What do understand by Control points, Degree of continuity, Local and Global control w.r.t Curve Generation?  Control Points 10 → In computer-aided ~~design~~ geometric design a control point is a member of a set of points used to determine the shape of a spline curve or more generally a surface or higher dimension object. Degree of continuity → The degree of continuity ~~is~~ defining the curve segment is one less that the number of defining polygon-point. If there are 4 control points then degree is 3. Local ~~Global~~ control → ~~Global~~ Local control means moving a control point alters the shape of whole curve. Global control → Global control means moving a control point does not alter or change the shape of whole curve.	[5]
Q. 2.	**Attempt any two**	
i)	Explain Liang Barsky line clipping algorithm, what are its benefits over Cohen Sutherland algorithm? Clip the line with Co-ordinated (5,10) and (35,30) against the window (xmin,ymin)=(10,10) and (xmax,ymax)=(20,20)	[10]

)

Que: Explain Liang Bansky line clipping algorithm, what are its benefits over cohen Sutherland algorithm? clip the line with co-ordinated (5, 10) and (35,30) againt the window $(x_{min}, y_{min}) = (10,10)$ and $(x_{max}, y_{max}) = (20,20)$

Ans:- Algorithm :-

Step 1:- Get the endpoints of line as (x_1, y_1) and (x_2, y_2)

Step 2:- Calculate $\Delta x, \Delta y, p_k, q_k$

Step 3:- Assign $t1 = 0$, $t2 = 1$

(a) If $p_k = 0$; line is parallel

If ~~p~~ $q_k < 0$; line lies outside window (Regict)

(b) If $p_k < 0$; find $t1$

$t1 = max(0, q_k / p_k)$

else $p_k > 0$; find $t2$

$t2 = min(1, q_k / p_k)$

(c) If $t1 > t2$; line is completely outside (rejected)

else find new value of x & y from the formula

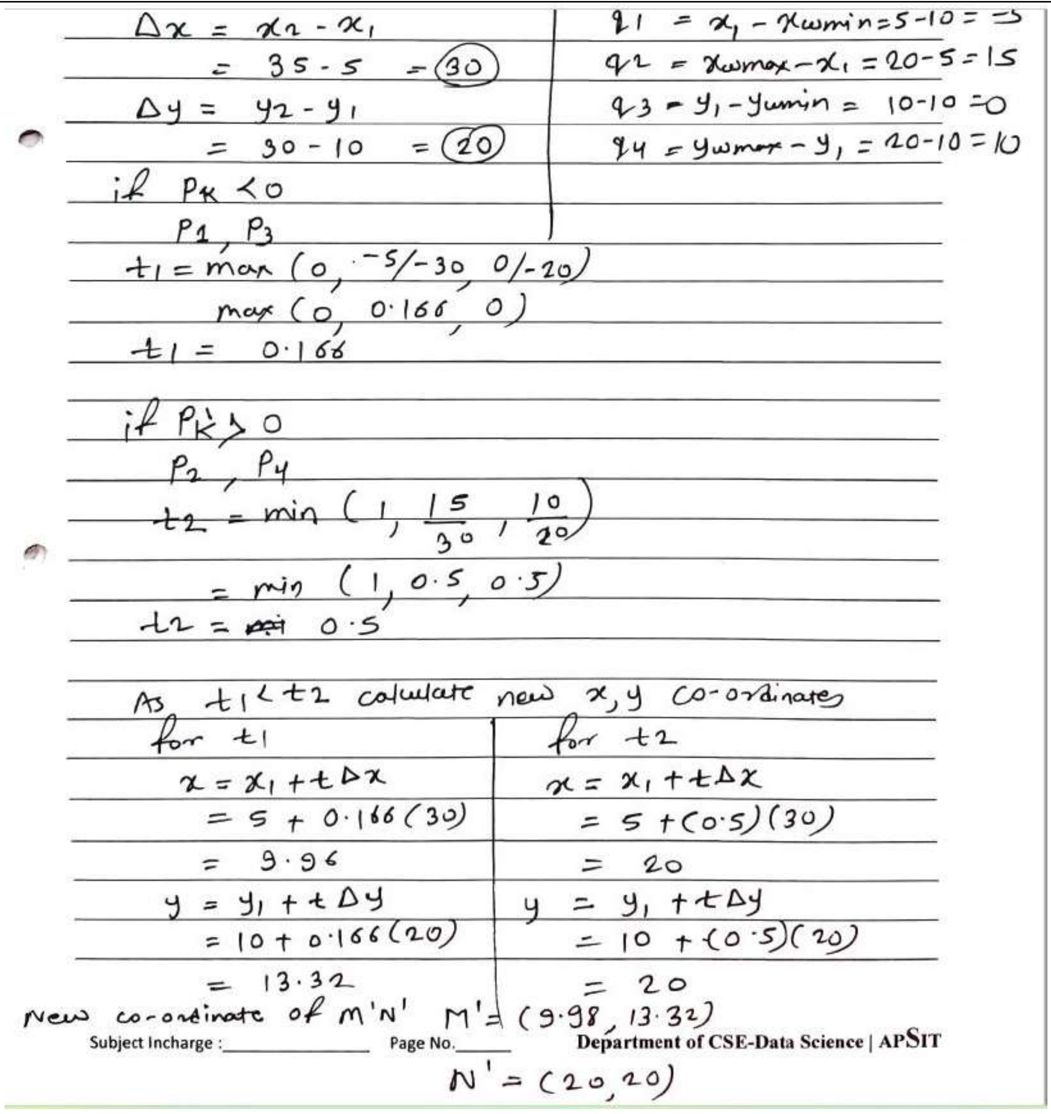
$$x = x_1 + t\Delta x$$
$$y = y_1 + t\Delta y$$

$P_1 = -\Delta x$	$q1 = x_1 - x_{wmin}$
$P_2 = \Delta x$	$q2 = x_{wmax} - x_1$
$P_3 = -\Delta y$	$q3 = y_1 - y_{wmin}$
$P_4 = \Delta y$	$q4 = y_{wmax} - y_1$

SR. No	Cohen Sutherland Algo	Liang Barsky Algo
1.	It is less efficient	It is more efficient
2.	In this Algo., each intersection requires both multiplication & a division	2. In this algo, each update of parameters requires only one division
3.	It follows the encoding approach	3) It follows the parametric approach
4.	It repeatedly calculates intersection along a line path even though the line may be completely outsid the clip window	4) In this window intersection are calculated only once when final values have been computed
5.	It can be used only on a rectangular clip window	5. It can be used for 1-D, 2-D, 3-D line clipping and Sometimes 4-D line clipping to

Numerical Solution :-

Given :-	
$x_{wmin} = 10$	$P_1 = -\Delta x = -30$
$x_{wmax} = 20$	$P_2 = \Delta x = 30$
$y_{wmin} = 10$	$P_3 = -\Delta y = -20$
$y_{wmax} = 20$	$P_4 = \Delta y = 20$
$m(5,10) \Rightarrow x_1 = 5 \quad y_1 = 10$	
$N(35,30) \Rightarrow x_2 = 35 \quad y_2 = 30$	

	 $\Delta x = x_2 - x_1$ $= 35 - 5 = \boxed{30}$ $\Delta y = y_2 - y_1$ $= 30 - 10 = \boxed{20}$ if $p_k < 0$ p_1, p_3 $q_1 = x_1 - x_{wmin} = 5 - 10 = -5$ $q_2 = x_{wmax} - x_1 = 20 - 5 = 15$ $q_3 = y_1 - y_{wmin} = 10 - 10 = 0$ $q_4 = y_{wmax} - y_1 = 20 - 10 = 10$ $t_1 = max(0, -5/-30, 0/-20)$ $max(0, 0.166, 0)$ $t_1 = 0.166$ if $p_k > 0$ p_2, p_4 $t_2 = min(1, \frac{15}{30}, \frac{10}{20})$ $= min(1, 0.5, 0.5)$ $t_2 = 0.5$ As $t_1 < t_2$ calulate new x, y co-ordinates for t_1 $x = x_1 + t\Delta x$ $= 5 + 0.166(30)$ $= 9.96$ $y = y_1 + t\Delta y$ $= 10 + 0.166(20)$ $= 13.32$ for t_2 $x = x_1 + t\Delta x$ $= 5 + (0.5)(30)$ $= 20$ $y = y_1 + t\Delta y$ $= 10 + (0.5)(20)$ $= 20$ New co-ordinate of M'N' $M' = (9.98, 13.32)$ Subject Incharge :_______ Page No._____ Department of CSE-Data Science | APSIT $N' = (20, 20)$	
ii)	Explain Sutherland Hodgman polygon clipping algorithm with suitable example and comment on its shortcoming.	[10]

Sutherland - Hodgeman algo :

- Polygons can be clipped against each edge of window one at a time.
- Edge intersections if any are easy to find since the x or y coordinates are already known.
- Vertices which are kept after clipping against one window edge are saved for clipping against the remaining edges.
- There are different cases to find the vertices of clipped polygon :

Case 1 :

Both the vertices are inside : Only the second vertex is added to output list.

Case 2 :

First vertex is outside while second one is inside :

Both the point of intersection of edge with clip boundary and second vertex are added to output list.

Case 3 :

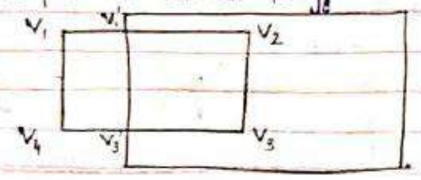
First vertex is inside while second one is outside :

Only the point of intersection of edge with the clip boundary is added to output list.

ase 4 :

Both vertices are outside : No vertices are added to output list.

Example for Convex Polygon



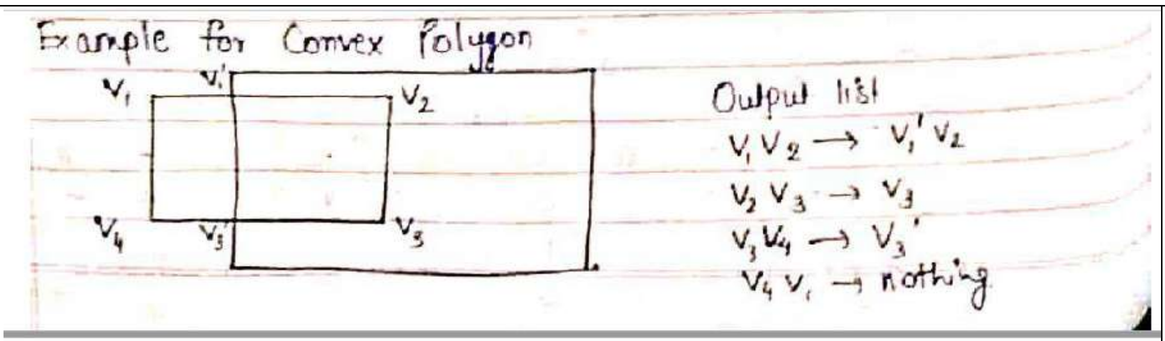
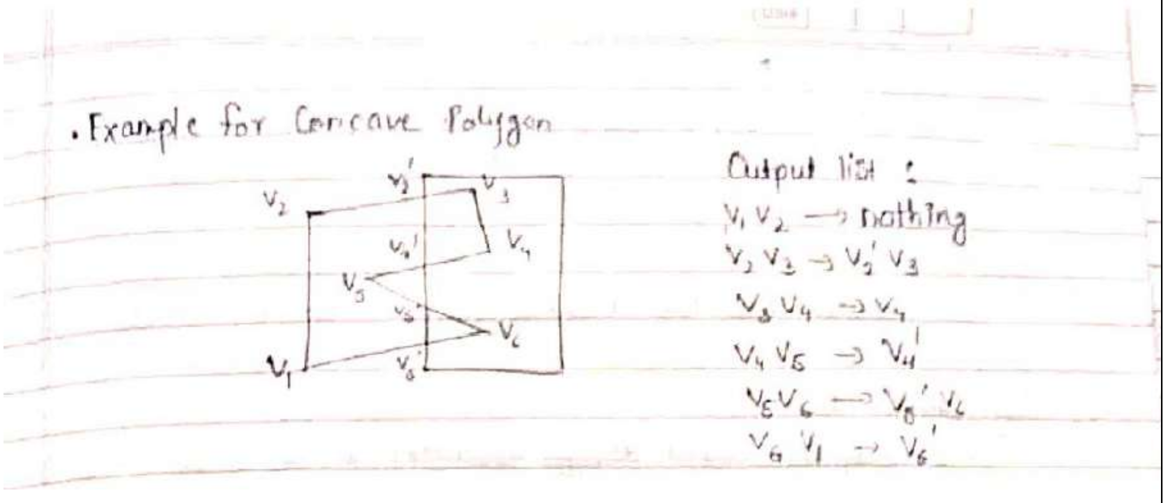
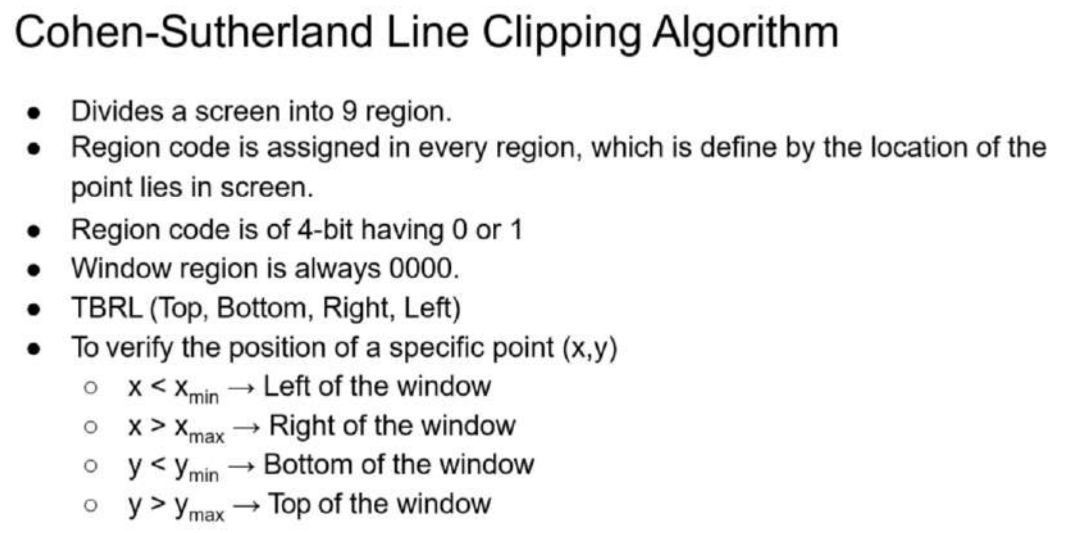
Output list

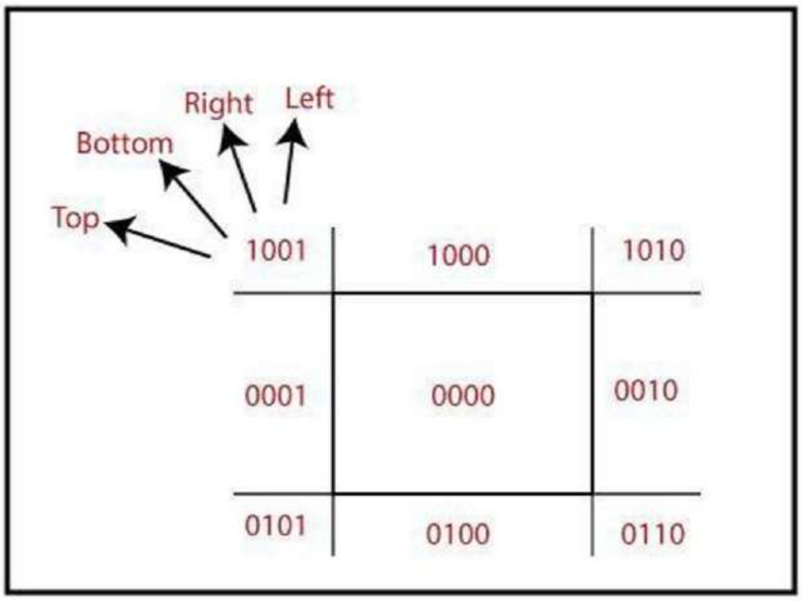
$V_1 V_2 \rightarrow V_1' V_2$

$V_2 V_3 \rightarrow V_3$

$V_3 V_4 \rightarrow V_3'$

$V_4 V_1 \rightarrow$ nothing

	 Example for Convex Polygon Output list $V_1 V_2 \rightarrow V_1' V_2$ $V_2 V_3 \rightarrow V_3$ $V_3 V_4 \rightarrow V_3'$ $V_4 V_1 \rightarrow$ nothing  V_2' V_3 V_2 V_4' V_4 V_5 V_5' V_6 V_1 V_6' • Example for Concave Polygon Output list : $V_1 V_2 \rightarrow$ nothing $V_2 V_3 \rightarrow V_2' V_3$ $V_3 V_4 \rightarrow V_4$ $V_4 V_5 \rightarrow V_4'$ $V_5 V_6 \rightarrow V_5' V_6$ $V_6 V_1 \rightarrow V_6'$	
iii)	Explain Cohen Sutherland line clipping algorithm. Apply the algorithm for ABCD be the rectangle window A(20,20) B(90,20) C(90,70) & D(20,70). Find region codes for the endpoint to clip the P1P2 with P1(10,30) P2(80,90) and P3P4 with P3(10,10) P4(70,60).  Cohen-Sutherland Line Clipping Algorithm • Divides a screen into 9 region. • Region code is assigned in every region, which is define by the location of the point lies in screen. • Region code is of 4-bit having 0 or 1 • Window region is always 0000. • TBRL (Top, Bottom, Right, Left) • To verify the position of a specific point (x,y) ○ $x < x_{min} \rightarrow$ Left of the window ○ $x > x_{max} \rightarrow$ Right of the window ○ $y < y_{min} \rightarrow$ Bottom of the window ○ $y > y_{max} \rightarrow$ Top of the window	[10]



Numerical

1) Let ABCD be the rectangle window A(20,20) B(90,20) C(90,70) & D(20,70). Find region codes for the endpoint to clip the P_1P_2 with $P_1(10,30)$ $P_2(80,90)$ and P_3P_4 with $P_3(10,10)$ $P_4(70,60)$.

Sol^n:

Line P_1P_2

Finding region code of P_1P_2

$P_1 = 0001$

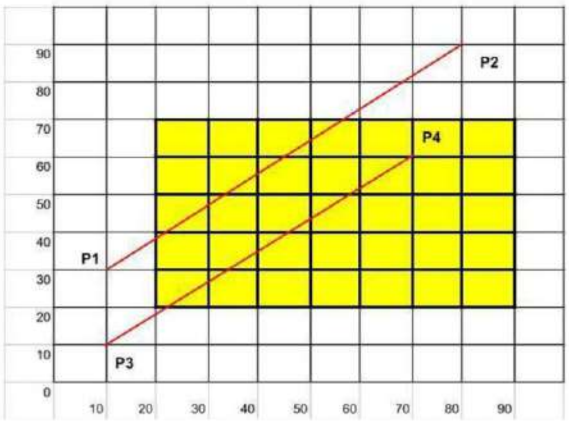
$P_2 = 1000$

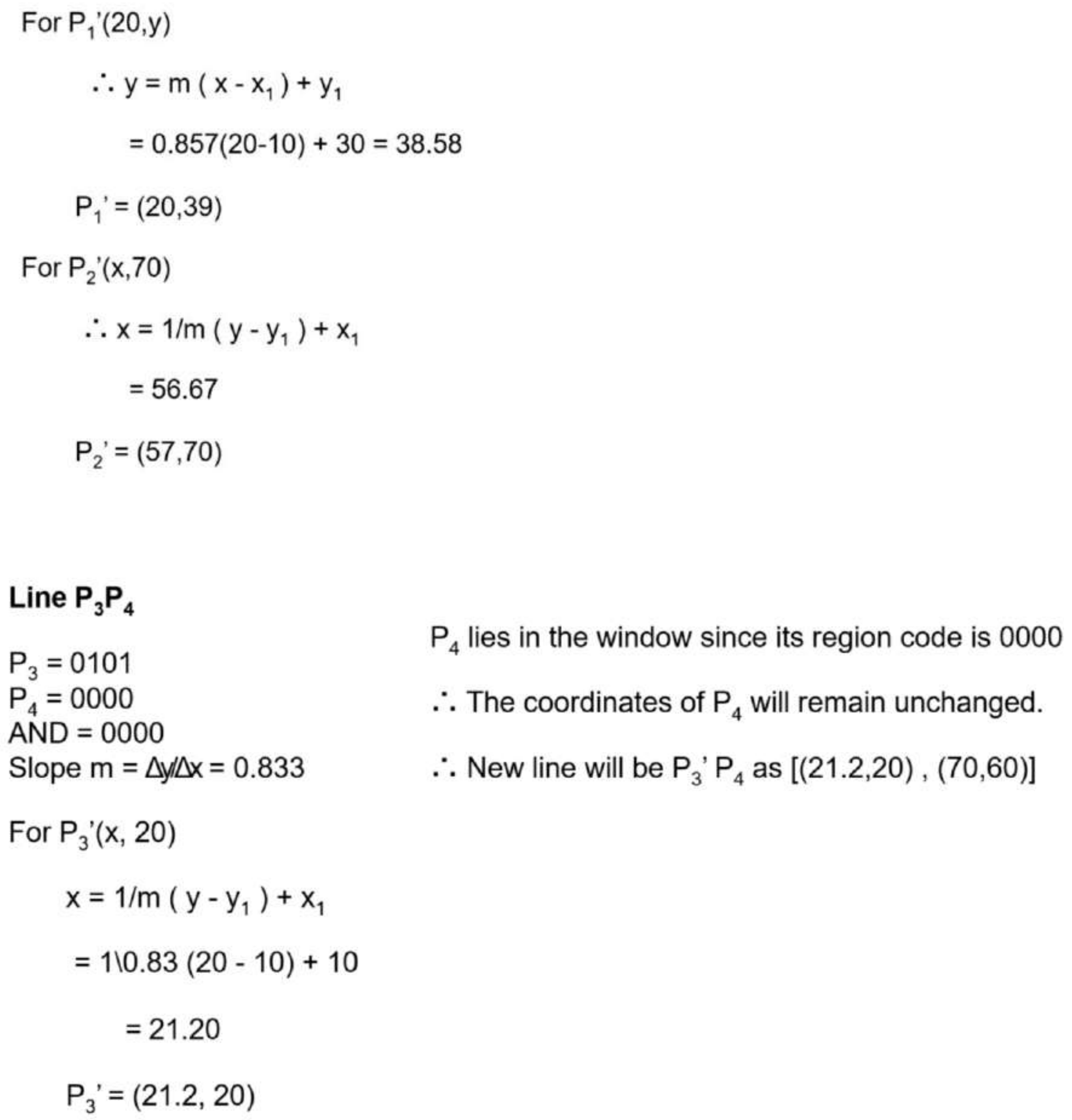
AND = 0000

∴ that means line is partially visible

So it needs clipping

Slope $m = \Delta y/\Delta x = 6/7 = 0.857$



	 For $P_1'(20,y)$ $\therefore y = m(x - x_1) + y_1$ $= 0.857(20-10) + 30 = 38.58$ $P_1' = (20,39)$ For $P_2'(x,70)$ $\therefore x = 1/m(y - y_1) + x_1$ $= 56.67$ $P_2' = (57,70)$ **Line P_3P_4** $P_3 = 0101$ $P_4 = 0000$ AND = 0000 Slope $m = \Delta y/\Delta x = 0.833$ For $P_3'(x, 20)$ $x = 1/m(y - y_1) + x_1$ $= 1\backslash 0.83(20 - 10) + 10$ $= 21.20$ $P_3' = (21.2, 20)$ P_4 lies in the window since its region code is 0000 $\therefore$ The coordinates of P_4 will remain unchanged. $\therefore$ New line will be $P_3' P_4$ as [(21.2,20) , (70,60)]	
Q. 3.	**Attempt any one.**	
i)	Explain the 12 Animation principles Ans: 1. Timing and Spacing Timing and Spacing in animation is what gives objects and characters the illusion of moving within the laws of physics. Timing refers to the number of frames between two poses, or the speed of action. For example, if a ball travels from screen left to screen right in 24 frames, that	[10]

would be timing. It takes 24 frames or 1 second (if you’re working within the film rate of 24 rates per second) for the ball to reach the other side of the screen. Timing can also establish mood, emotion, and personality.

Spacing refers to how those individual frames are placed. For instance, in the same example, the spacing would be how the ball is positioned in the other 23 frames. If the spacing is close together, the ball moves slower. If the spacing is further apart, the ball moves faster.

2. Squash and Stretch

Squash and stretch is what gives flexibility to objects. The easiest way to understand how squash and stretch work is to look at a bouncing ball. As the ball starts to fall and picks up speed, the ball will stretch out just before impact.

As the ball impacts the ground, it squashes before stretching again as it takes off. Please note, the volume of an object doesn’t change. In the case of the ball, when it is squashed or stretched, the width and depth need to correspond accordingly.

There’s a lot of examples of “squash and stretch” happening in real life that you may not notice. For instance, squashing and stretching occurs in the face when someone speaks because the face is very flexible. In animation, this can be exaggerated. Squash and stretch can be implemented in many different areas of animation to add comical effect or more appeal, like for the eyes during a blink or when someone gets surprised or scared.

3. Anticipation

Anticipation is used in animation to set the audience up for an action that is about to happen, and is required to sell believable movements.

An easy way to think about this is before a baseball player pitches the ball, they first need to move their entire body and arm backward to gain enough energy to throw the ball forward. So, if an animated person needs to move forward, they first must move back. Or, if a character is reaching for a glass on a table, they must first move their hand back. This not only gets up their momentum, but it lets the audience know this person is about to move.

Other cases where anticipation is used include when a character looks off screen when someone is arriving, or when a character’s attention is focused on something they are about to do.

4. Ease In and Ease Out

As any object moves or comes to a stop, there needs to be a time for acceleration and deceleration. Without ease in and ease out (or slow in and slow out), movements become very unnatural and robotic.

As a car moves away from a stop, it doesn't just reach full speed in an instant. It must first gain speed. As it comes to a stop, it doesn't go from sixty to zero in the blink of an eye. Instead, it slows down until it reaches a complete stop.

The same must be accomplished in animation and the easiest way to accomplish ease in and ease out is to utilize the principle of spacing. As a character stands up from a sitting position, the spacing of each pose will be closer together at the start so that they can ease into the movement. As they stand up, they will ease out of the movement by spacing the poses further apart at the end of the action. Without this acceleration and deceleration of actions, everything would be very abrupt and jerky.

5. Follow Through and Overlapping Action

Follow through is the idea that separate parts of the body will continue moving after the character has come to a stop. As a character comes to a stop from a walk, the arms may continue forward before settling in a down position. This could also be the case with articles of clothing.

Overlapping action (also called "drag" or "lead and follow") is very similar in that it means different parts of the body will move at different times. An example of overlapping action is when a character raises their arm up to wave: The shoulder will move first, then the arm, and then the elbow, before the hand lags behind a few frames. You can also see this when a blade of grass waves in the wind. The base moves first and then the rest of the grass follows behind at different rates, giving it that waving motion.

Additionally, characters who are remaining still need to display some sort of movement (blinking eyes, breathing, etc.) to prevent the animation from becoming "dead." This is called "moving hold."

6. Arcs

Everything in real life typically moves in some type of arcing motion. Since it's unnatural for people to move in straight lines, you should adhere to this principle of animation to ensure you get smooth, realistic movements. The quicker something moves, the flatter the arc and the broader the turn. The only time something would move in a perfectly straight line is a robot.

If a character is turning his head, he will dip his head down during the turn to create an arcing motion. You also want to ensure that more subtle things move in arcs. For example, when a character walks, even the tips of their toes should move in a rounded, arcing motion.

7. Exaggeration

Exaggeration is used to push movements further, adding more appeal to an action, and should always be implemented to some degree.

Exaggeration can be used to create extremely cartoony movements including physical alterations or supernatural elements. Or, exaggeration can be incorporated with a little more restraint for more realistic actions. But, even then you can still use exaggeration to make a more readable or fun movement while still staying true to reality.

So, if a character is preparing to jump off a diving board, you can push them down just a little bit further before they leap off. Alternatively, you can use exaggeration in the timing to enhance different movements or help sell the weight of a character or object.

8. Solid Drawing

In 2D animation, solid drawing is about creating an accurate drawing in terms of volume and weight, balance, shadow, and the anatomy in a pose. With 3D animation, animators need to think about how to pose out your 3D character rig to ensure there is correct balance and weight, as well as a clear silhouette.

Avoid “twinning,” which is creating a mirrored pose across to the other side (both arms on hips or both hands in pockets) because this creates a rather boring and unappealing pose.

9. Appeal

This principle can really come down to adding more appeal (charisma) in many different areas of your animation, such as in posing. The most obvious example, however, is appeal in the character design because you want to have a character that the audience can connect with or relate to, whereas a complicated or confusing character design can lack appeal.

You can find areas on the character to push and exaggerate in order to create a more unique design that will stick out in your audience's memory. One example is to simply exaggerate the jawline or push the youthfulness in the eyes. Either of these can help create more appeal.

Keep in mind that appeal is also required for villains.

10. Straight Ahead Action and Pose to Pose

Straight ahead action is a very spontaneous and linear approach to animating and is animated from start to finish, frame by frame. With this, you'll create each pose of the animation one after the other. So, if your character is landing on the ground after jumping in the air, you would create the poses where he is standing, then the poses where he is beginning to kneel down, and then completely crouched. In other words, you're really working through the animation as you're going to make quick action fluid and dynamic.

With pose to pose, the animation is much more methodical, with just the most important poses required to properly tell the story. You would animate the character landing on the ground after jumping in the air by using fewer poses (standing and crouched). This allows for more simple work and ensures the proportions and timing are correct before you add more intervals later, and is great for slow, dramatic, or emotional scenes.

Often, these two approaches are used in combination to great effect.

11. Secondary Action

Secondary action refers to the actions that support or emphasize the main action to breathe more life into the animation and create a more convincing performance. It's important to remember that the secondary action should typically be something subtle that doesn't detract from the main action happening (perhaps even thought of as a subconscious action). For this reason, dramatic movements take priority over things like facial expressions.

Let's say a character is talking to another character in a waiting room. The two of them talking would be the main action, but if one of them begins tapping their foot nervously, that would be the secondary action. Other examples would be a character whistling, leaning on a wall, or crossing their arms while a primary action is taking place.

12. Staging

Staging is how you go about setting up your scene, from the placement of the characters, to the background and foreground elements, the character’s mood, and how the camera angle is set up. Staging is used to make the purpose of the animation unmistakably clear to the viewer. You want to keep the focus on what you want to communicate to the audience (and avoid unnecessary detail) so they don’t become confused.

ii) Explain Z Buffer algorithm for hidden surface removal. [10]

Ans:

Z-Buffer Algorithm

It is also called a **Depth Buffer Algorithm**. Depth buffer algorithm is simplest image space algorithm. For each pixel on the display screen, we keep a record of the depth of an object within the pixel that lies closest to the observer. In addition to depth, we also record the intensity that should be displayed to show the object. Depth buffer is an extension of the frame buffer. Depth buffer algorithm requires 2 arrays, intensity and depth each of which is indexed by pixel coordinates (x, y).

Algorithm

For all pixels on the screen, set depth [x, y] to 1.0 and intensity [x, y] to a background value.

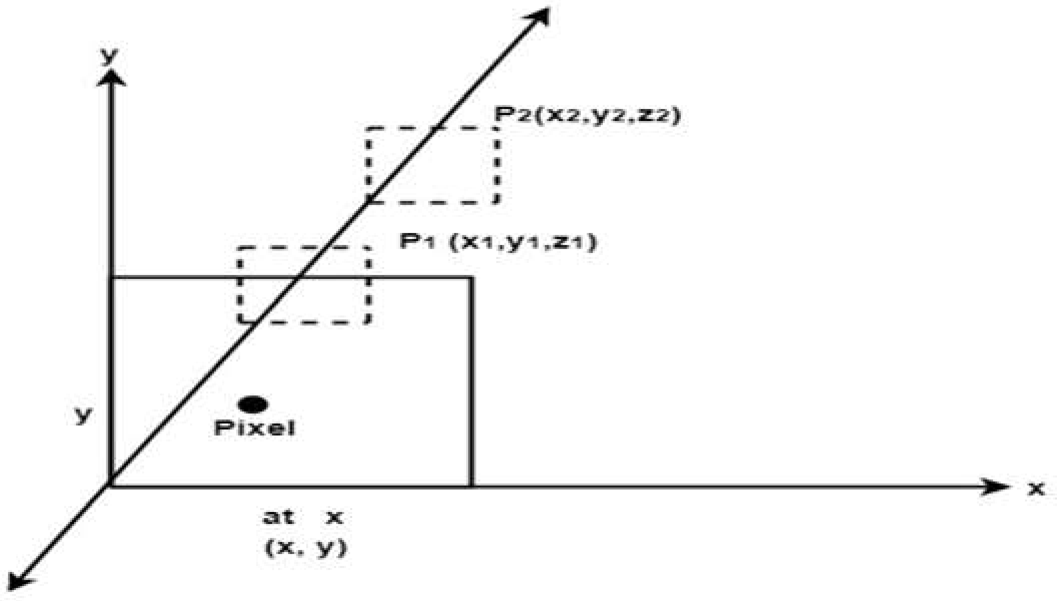
For each polygon in the scene, find all pixels (x, y) that lie within the boundaries of a polygon when projected onto the screen. For each of these pixels:

(a) Calculate the depth z of the polygon at (x, y)

(b) If z < depth [x, y], this polygon is closer to the observer than others already recorded for this pixel. In this case, set depth [x, y] to z and intensity [x, y] to a value corresponding to polygon's shading. If instead z > depth [x, y], the polygon already recorded at (x, y) lies closer to the observer than does this new polygon, and no action is taken.

3. After all, polygons have been processed; the intensity array will contain the solution.

4. The depth buffer algorithm illustrates several features common to all hidden surface algorithms.



5. First, it requires a representation of all opaque surface in scene polygon in this case.

6. These polygons may be faces of polyhedral recorded in the model of scene or may simply represent thin opaque 'sheets' in the scene.

7. The IInd important feature of the algorithm is its use of a screen coordinate system. Before step 1, all polygons in the scene are transformed into a screen coordinate system using matrix multiplication.

Limitations of Depth Buffer

1. The depth buffer Algorithm is not always practical because of the enormous size of depth and intensity arrays.
2. Generating an image with a raster of 500 x 500 pixels requires 2, 50,000 storage locations for each array.
3. Even though the frame buffer may provide memory for intensity array, the depth array remains large.
4. To reduce the amount of storage required, the image can be divided into many smaller images, and the depth buffer algorithm is applied to each in turn.
5. For example, the original 500 x 500 faster can be divided into 100 rasters each 50 x 50 pixels.
6. Processing each small raster requires array of only 2500 elements, but execution time grows because each polygon is processed many times.
7. Subdivision of the screen does not always increase execution time instead it can help reduce the work required to generate the image. This reduction arises because of coherence between small regions of the screen.