PRACTICAL - 8

AIM: Perform the Cohen-Sutherland Line Clipping Algorithm.

Description:

In this practical, students will explore and implement the Cohen-Sutherland Line Clipping Algorithm. This algorithm is used for efficiently clipping lines against a rectangular window in computer graphics. Understanding line clipping is essential for displaying only the relevant portions of a line within a specified window.

Objective:

- Understand the need for line clipping in computer graphics.
- Learn the principles of the Cohen-Sutherland Line Clipping Algorithm.
- Implement the algorithm to clip lines against a rectangular window.
- Visualize the results and observe the impact of the clipping process.

Cohen-Sutherland Line Clipping Algorithm:

Algorithm Steps:

1. Define the rectangular clipping window.
2. Classify each endpoint of the line (P1 and P2) into one of the nine regions based on the window.
3. If both endpoints are in the same region (accept), or both endpoints are outside the window (reject), clip the line.
4. If one endpoint is inside the window and the other is outside, calculate the intersection of the line with the window boundary.
5. Replace the outside endpoint with the intersection point.
6. Repeat steps 2-5 until the line is either accepted or rejected.

Procedure:

1. Define a rectangular clipping window (e.g., (xmin, ymin), (xmax, ymax)).
2. Choose a line segment with endpoints (x1, y1) and (x2, y2).
3. Apply the Cohen-Sutherland Line Clipping Algorithm to clip the line against the window.
4. Visualize and compare the original line with the clipped line.
5. Experiment with different lines and window configurations to observe the algorithm's behavior.

Coding:

```
#include<stdio.h>
#include<conio.h>
#include<graphics.h>
#include<math.h>
#define Round(val) ((int)(val+.5))
int maxx, maxy, miny, minx;
void main()
{
    int gd = DETECT, gm;
    void clipping (int xa, int ya, int xb, int yb);
    int xa, xb, ya, yb;
    printf("Enter the window coordination");
    scanf("%d%d%d%d", &minx, &maxy, &maxx, &miny);
    printf("Enter the two end Points for the line");
    scanf("%d%d%d%d", &xa, &ya, &xb, &yb);
    initgraph(&gd, &gm, "");
    rectangle(minx, miny, maxx, maxy);
    line(xa, ya, xb, yb);
    getch();
    closegraph();
}
```

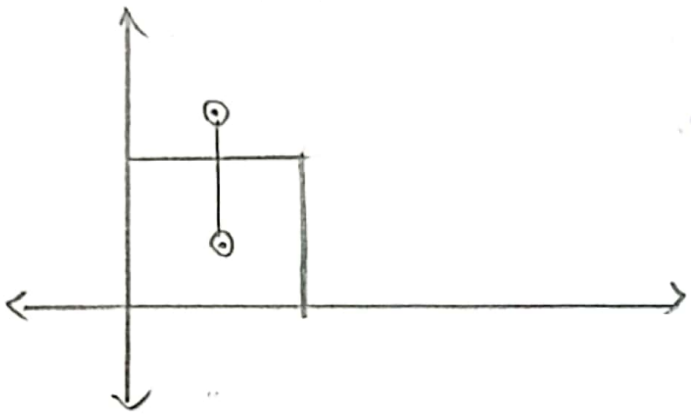
```
void clipping(int xa, int ya, int xb, int yb)
{
    int Dx = xb - xa, Dy = yb - ya, steps, k;
    int visible1 = 0, visible2 = 0;
    float xin, yin, x = xa, y = ya;
    if (abs(Dx) > abs(Dy))
        steps = abs(Dx);
    else
        steps = abs(Dy);
    xin = Dx / (float) steps;
    yin = Dy / (float) steps;
    putpixel (Round(x), Round(y), 2);
} else
    visible2 = 1;
} for (k = 0; k < steps; k++)
{
    x += xin;
    y += yin;
    if ((y > miny && y < maxx)
    {
        visible1 = 1;
        putpixel (Round(x), Round(y), 2);
    } else
        visible2 = 1;
    if (visible == 0)
        outtextxy (20, 200, "completely visible");
    if (visible == 1 && visible2 == 1)
        outtextxy (20, 20, "partialy visible");
    if (visible == 1 && visible2 == 0)
        outtextxy (20, 20, "completely visible");
}
```

SILVER OAK UNIVERSITY
EDUCATION TO INNOVATION

OUTPUT

The area of code of 1st point is 0000

The area code of 2nd point is 1000



Post Practical Questions:

1. **Clipping Window:** Explain the concept of a clipping window and its role in the Cohen-Sutherland Line Clipping Algorithm. Discuss how the algorithm determines whether a line is inside, outside, or partially inside the clipping window.

Ans: A clipping window, in computer graphics, defines the region of a screen where drawing or rendering is allowed. The cohen-sutherland clipping Algorithm is a method used to determine whether a line segment lies completely within, completely outside or partially within the clipping window.

2. Discuss the classification of lines in Cohen-Sutherland as entirely inside, entirely outside, or partially inside the clipping window. Illustrate each scenario with examples.

Ans: → In the cohen-sutherland algorithm lines are classified as either entirely inside, entirely outside, or partially inside the clipping window based on their endpoint codes. By classifying lines in this manner, the cohen-sutherland algorithm efficiently determines which portions of line need to be clipped.

3. **Optimization Techniques:** Explore optimization techniques for the Cohen-Sutherland Line Clipping Algorithm. Discuss strategies to improve its efficiency, especially in scenarios involving numerous lines.

Ans: → Here use the optimization technique for the cohen-sutherland line clipping algorithm

- bounding box optimization
- line segments Reordering
- cohen-sutherland reduction
- parallelization
- special data structures
- Hardware Acceleration

Conclusion:

In this practical, we learn about the cohen-sutherland line clipping Algorithm.

Marks out of 10	10
Signature with Date of Completion	Dey

SILVER OAK UNIVERSITY
EDUCATION TO INNOVATION

PRACTICAL - 9

AIM: Perform the Liang-Barsky Line Clipping Algorithm.

Description:

In this practical, students will explore and implement the Liang-Barsky Line Clipping Algorithm. This algorithm is widely used for clipping lines against a rectangular window in computer graphics. Understanding line clipping is crucial for displaying only the relevant portions of a line within a specified window.

Objective:

- Understand the principles of the Liang-Barsky Line Clipping Algorithm.
- Learn the steps involved in clipping lines against a rectangular window.
- Implement the algorithm to efficiently clip lines.
- Visualize the results and observe the impact of the clipping process.

Liang-Barsky Line Clipping Algorithm:

Algorithm Steps:

1. Define the rectangular clipping window.
2. Calculate the parameters p1, p2, p3, and p4 for the line segment using the window boundaries.
3. Check if the line is completely outside the window (reject) based on the calculated parameters.
4. If the line is not rejected, calculate the values of t1 and t2 using the window boundaries.
5. If t2 < t1, reject the line since it is entirely outside the window.
6. Clip the line using the calculated values of t1 and t2.
7. Update the line coordinates based on the clipping.

Procedure:

1. Define a rectangular clipping window (e.g., (xmin, ymin), (xmax, ymax)).
2. Choose a line segment with endpoints (x1, y1) and (x2, y2).
3. Apply the Liang-Barsky Line Clipping Algorithm to clip the line against the window.
4. Visualize and compare the original line with the clipped line.
5. Experiment with different lines and window configurations to observe the algorithm's behavior.

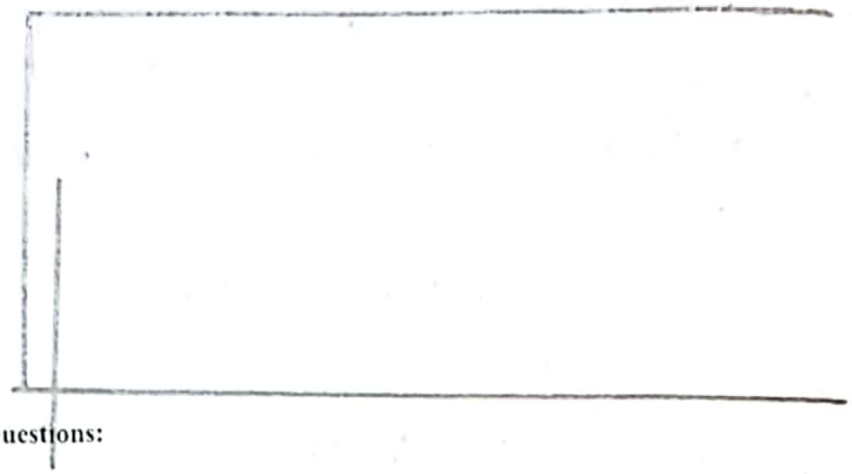
Coding:

```
#include<stdio.h>
#include<conio.h>
#include<graphics.h>
#include<math.h>
void main()
{
    int i, gm, gd;
    int x1, x2, y1, y2, xmin, xmax, ymin, xx1, xx2, yy1, yy2;
    float t1, t2, p[4], q[4], temp;
```

```
detectgraph(dgd, dgm)
initgraph(dgd, dgm, "C:\\TC\\BGI");
xmin = 150;
xmax = 450;
ymin = 100;
ymax = 400;
rectangle(xmin, ymin, xmax, ymax);
printf("Enter the starting point:");
scanf("%d%d", &x1, &y1);
printf("Enter the ending point:");
scanf("%d%d", &x2, &y2);
printf("Line after clipping:");
line(x1, y1, x2, y2);
getch();
clrscr();
printf("Line after clipping:")
rectangle(xmin, ymin, xmax, ymax);
p[0] = -(x2 - x1);
p[1] = (x2 - x1);
p[2] = -(y2 - y1);
p[3] = (y2 - y1);
q[0] = (x1 - xmin);
q[1] = (xmax - x1);
q[2] = (y1 - ymin);
q[3] = (ymax - y1);
for(i = 0; i < 4; i++)
{
    if(p[i] == 0)
    {
        printf("In line is parallel
                to one of the
                clipping boundary");
        if(q[i] >= 0)
        {
            if(y1 < ymin)
            {
                y1 = ymin;
            }
            if(y2 > ymax)
            {
                y2 = ymax;
            }
            line(x1, y1, x2, y2);
        }
        if(i >= 1)
        {
            if(x1 < xmin)
            {
                x1 = xmin;
            }
            if(x2 > xmax)
            {
                x2 = xmax;
            }
            line(x1, y1, x2, y2);
        }
    }
    getch();
    return();
}
}
```

OUTPUT

enter the starting point : 100
100
Enter the ending point : 200
line after : clipping



Post Practical Questions:

1. **Parameterization of Lines:** Explain the concept of parameterization of lines in the Liang-Barsky Line Clipping Algorithm. Discuss how this parameterization simplifies the clipping process.

Ans: In the Liang-Barsky line clipping Algorithm, parameterization of line involve representing a line segment in terms of parameter rather than explicitly defining its endpoints. improves efficiency compared to other method.

2. **Clipping against Multiple Edges:** Discuss the approach taken by Liang-Barsky for clipping against multiple edges simultaneously. Provide examples to illustrate the algorithm's behavior in different scenarios.

Ans: Here, an overview of approach –

- Initialization
- intersection test
- Iteration
- final clipping

→ If the line segment lies entirely inside the clipping window, the algorithm will not need to perform any clipping

3. **Comparative Analysis:** Compare the Liang-Barsky Line Clipping Algorithm with other line clipping algorithms, such as Cohen-Sutherland. Discuss advantages, disadvantages, and scenarios where one algorithm might be preferred over the other.

Ans:

→ Parameterization- uses parameterization of lines, making it computationally efficient.

→ Simplicity- the algorithm is relatively simple to implement.

→ Handles multiple edges- can clip against multiple edges simultaneously.

→ Suitable for applications where multiple edges need to be clipped simultaneously.

Conclusion:

In this practical, we learn about the Liang-Barsky Line clipping Algorithm.

Marks out of 10	10
Signature with Date of Completion	BJ

SILVER OAK UNIVERSITY
EDUCATION TO INNOVATION

PRACTICAL – 10

AIM: Perform the simple Animation of Moving car and bouncing ball.

Description:

In this practical, students will create a simple animation using basic graphics concepts. The animation involves a moving car and a bouncing ball, providing hands-on experience in animating objects in a 2D space.

Objective:

- Understand the fundamentals of animation in computer graphics.
- Learn to implement basic motion and rendering techniques.
- Create a simple animation with a moving car and a bouncing ball.
- Gain practical insights into translating and animating graphical objects.

Animation Algorithm:

Algorithm Steps:

Moving Car:

Initialize the car's position and direction.
Set up a rendering loop.
In each frame, update the car's position based on its direction.
Render the car at its updated position.
Repeat steps 3-4 for a specified duration or until a certain condition is met.

Bouncing Ball:

Initialize the ball's position, velocity, and bouncing parameters.
Set up a rendering loop.
In each frame, update the ball's position based on its velocity.
Check for collisions with the window boundaries.
- **If a collision is detected, reverse the velocity to simulate bouncing.**

Render the ball at its updated position.
Repeat steps 3-5 for a specified duration or until a certain condition is met.

Procedure:

Set up a graphical environment (e.g., using a programming language or graphics library).
Initialize the car and ball with their respective parameters.

SILVER OAK UNIVERSITY
EDUCATION TO INNOVATION

Implement the aaimation algorithm for the moving car and bouncing ball.
Adjust parameters such as speed, direction, and bouncing behavior to observe different effects.
Run the animation and visualize the movement of the car and the bouncing ball.
Experiment with modifying the algorithm to achieve different animation behaviors.

Coding:

```
# include<stdio.h>
# include<stdlib.h>
# include<conio.h>
# include<graphics.h>
# include<math.h>

void main()
{
    int gd = DETECT, gm;
    int i, x, y, flag = 0;
    initgraph(&gd, &gm, "C:\\TC\\BGI");
    x = getmaxx()/2;
    y = 30;
    while(!kbhit())
    {
        if(y >= getmaxy()-30 || y <= 30)
            flag = !flag;
        setcolor(RED);
        setfillstyle(SOLID_FILL, RED);
        circle(x, y, 30);
        floodfill(x, y, RED);
```

```
        delay(2+);
        cleardevice();
        if (flag)
        {
            y=y+2;
        }
        else
        {
            y=y-2;
        }
    }
    getch();
    closegraph();
}
```

OUTPUT

SILVER OAK UNIVERSITY
EDUCATION TO INNOVATION

Post Practical Questions:

1. Discuss the animation techniques used in your program for creating the motion of a car and bouncing ball. Explain how the frame-by-frame rendering and timing are managed.

Ans: → frame - by frame Animation, tweening, timing and frame rate management, interpolation, physics based Animation.

→ frame - by frame rendering and timing involves careful coordination of keyframes.

2. **Realistic Movement:** How would you enhance the realism of the car's movement? Discuss additional features or considerations that could be added to improve the natural appearance of motion in the animation.

Ans: → to enhance the realism of a car's movement in animation, could be implemented
- Realistic physics simulation
- Dynamic lighting and shading
- Particle effects
- Sound effects
- camera movement.

3. **User Interaction:** Implement user interaction in your animation, allowing users to control the speed or direction of the moving car and bouncing ball. Discuss the significance of user interaction in enhancing the overall animation experience.

Ans: Overall, user interaction plays a crucial role in enhancing the overall animation experience by increasing engagement, enabling

customization, promoting learning and exploration, fostering interactivity and providing feedback and responsiveness.

Conclusion:

In this practical, we learn about Animation algorithm for moving car and Bouncing ball.

Marks out of 10	10
Signature with Date of Completion	

SILVER OAK UNIVERSITY
EDUCATION TO INNOVATION

PRACTICAL-11

AIM: To Perform the following Animation methods.

a. Morphing Animation

b. Rendered Animation

Description:

In this practical, students will explore two different animation methods - Morphing Animation and Rendered Animation. Morphing involves smoothly transforming one image into another, while Rendered Animation focuses on creating dynamic scenes with rendered graphics. These methods provide insights into different aspects of computer graphics and animation.

Objective:

- Understand the concepts of Morphing Animation and Rendered Animation.
- Implement algorithms for creating morphing transitions between images.
- Explore rendering techniques to generate dynamic animations.

a. Morphing Animation:

Algorithm Steps:

1. Select two images (source and target) for morphing.
2. Define corresponding feature points in both images.
3. Generate a series of intermediate frames between the source and target images.
4. Interpolate the feature points to create smooth transitions.
5. Blend the pixel values of corresponding points in the source and target images for each intermediate frame.
6. Render the frames sequentially to create a morphing animation.

Procedure:

1. Choose two images with distinctive features for morphing.
2. Identify corresponding feature points in both images.
3. Implement the morphing algorithm to generate intermediate frames.
4. Visualize the morphing animation and observe the gradual transformation.
5. Experiment with different images and feature points to explore variations in morphing.

b. Rendered Animation:

Algorithm Steps:

1. Define a 3D scene with objects, lights, and camera parameters.
2. Set up a rendering loop to generate frames for animation.
3. In each frame, update the position, rotation, or properties of objects to create motion.
4. Apply shading and rendering techniques to simulate realistic lighting and materials.
5. Render each frame and combine them to generate a rendered animation.

Procedure:

1. Define a simple 3D scene with objects like spheres, cubes, or custom models.
2. Set up a rendering environment using a graphics library or programming language.
3. Implement the rendering loop with dynamic changes to the scene.
4. Apply shading and rendering techniques for realistic appearance.
5. Run the animation and observe the dynamic rendering of the scene.
6. Experiment with different scene configurations and rendering parameters.

Coding:

OUTPUT

Post Practical Questions:

a. Morphing Animation

1. **Morphing Concept:** Explain the concept of morphing animation and how it differs from traditional frame-based animations. Discuss the key steps involved in achieving a smooth morphing effect.

Ans: ⇒ Morphing Animation is a technique used to smoothly transform one image or shape into another over a sequence of frames.
→ Here are diffey from traditional frame based application
- key shapes or images, interpolation, vector bases

2. **Applications:** Identify and discuss potential applications for morphing animations in real-world scenarios. Consider fields such as entertainment, scientific visualization, or educational purposes.

Ans: → entertainment, scientific visualization, educational purposes, user interfaces, Art and design

Overall, Morphing animation offer a versatile and powerful technique for creating dynamic visual effects and conveying complex transformations in various real-world scenarios

b. Rendered Animation

1. Discuss rendering techniques used in your program for creating realistic animations. Explore concepts such as shading, lighting, and texture mapping and their impact on the final rendered animation.

Ans: In the context of creating realistic animations, several rendering techniques are employed to enhance the visual quality of the output. These techniques include shading, lighting and texture mapping.

2. **Realism vs. Performance:** Analyze the trade-off between realism and performance in rendered animations. Discuss scenarios where sacrificing realism for better performance is acceptable and situations where high realism is crucial.

Ans: → Acceptable Sacrifice of Realism for Performance –
- Real-time Applications
- Background elements
- Low-poly Art styles

→ Critical importance of High realism
- film and cinematic.
- Architectural visualization
- Product visualization.

Conclusion:

In this Practical, we learn about the Morphing Animation and Rendered Animation.

Marks out of 10	
Signature with Date of Completion	

SILVER OAK UNIVERSITY
EDUCATION TO INNOVATION

PRACTICAL-12

AIM: To Perform the following Animation methods.

a. Character Animation

b. Facial Animation

Description:

In this practical, students will explore two distinct animation methods - Character Animation and Facial Animation. Character Animation involves bringing characters to life through movement, while Facial Animation focuses on animating facial expressions and features. These methods provide valuable insights into the complexities of animating human-like figures.

Objective:

- Understand the principles of Character Animation and Facial Animation.
- Implement algorithms for animating characters and facial features.
- Explore techniques for expressing emotions through animation.

a. Character Animation:

Algorithm Steps:

1. Design or choose a character model with articulated joints (skeleton).
2. Define a set of keyframes to specify the character's poses at different time intervals.
3. Implement an interpolation algorithm (e.g., linear, cubic) to generate smooth transitions between keyframes.
4. Apply inverse kinematics (IK) or forward kinematics (FK) to control the movement of joints.
5. Incorporate secondary motion (e.g., hair, clothing) to enhance realism.
6. Render the frames sequentially to create a character animation.

Procedure:

1. Choose or design a character model with joints.
2. Define keyframes representing different poses for the character.
3. Implement the character animation algorithm, including interpolation and kinematics.
4. Observe the character's movement through the generated animation.
5. Experiment with adjusting keyframes and parameters to modify the animation.

b. Facial Animation:

Algorithm Steps:

1. Define a facial rig with key controls for various facial features (eyebrows, eyes, mouth, etc.).
2. Create a set of facial expressions and emotions as keyframes.
3. Implement a facial animation algorithm to smoothly transition between expressions.

4. Use blend shapes or bone-driven deformations to control facial movement.
5. Fine-tune parameters to convey different emotions through facial animation.
6. Render the frames sequentially to create a facial animation.

Procedure:

1. Design a facial rig with controls for essential features.
2. Define keyframes representing different facial expressions and emotions.
3. Implement the facial animation algorithm, incorporating blend shapes or bone-driven deformations.
4. Observe the facial expressions and emotions conveyed in the generated animation.
5. Experiment with modifying keyframes and controls to explore variations in facial animation.

Coding:

SILVER OAK UNIVERSITY
EDUCATION TO INNOVATION

OUTPUT

Post Practical Questions:

a. Character Animation

1. Explain the concepts of rigging and skeletons in character animation. Discuss how they contribute to creating flexible and realistic character movements.

Ans: Rigging and skeleton play a crucial role in character animation by providing animators with the tools to create flexible and realistic movements. They allow for precise control over the character's pose and expression, enabling animators to convey emotion and personality effectively.

2. Discuss the keyframe animation technique in character animation. Explain how keyframes are used to define significant poses and movements in the animation sequence.

Ans: → keyframe animation technique gives animators precise control over character movements, allowing them to create expressive performances with nuanced gestures and actions.

By strategically placing keyframes and adjusting the animation curves, animators can bring characters to life the believable and dynamic motion

3. Consider a scenario where two animated characters interact with each other. Discuss the challenges and techniques involved in coordinating the movements of multiple characters in an animation.

Ans: Coordinating the movements of multiple characters in an animation requires careful planning, attention to detail, and collaboration among animators. By addressing the challenges and applying effective techniques and engagement of the audience.

b. Facial Animation

1. Explain the process of facial rigging in character animation. Discuss the role of facial bones and controls in creating expressive facial animations.

Ans: → facial rigging in character animation involves creating a digital framework that controls the movement and deformation of a character's face. this process typically includes the following steps: - Bone placement, weight painting, control setup, Blendshape / shape keys, expression libraries

2. Discuss the challenges and techniques associated with lip syncing in facial animation. Explore how accurate synchronization of facial movements with audio enhances the overall animation quality.

Ans: → lip syncing in facial animation involves matching the movements of a character's lip to pre-recorded dialogue or audio. this process presents several challenges and requires careful attention to detail to achieve accurate synchronization: Phoneme Recognition, timing and pacing, expression variation

3. How do you convey different emotions through facial expressions in animation? Discuss the role of facial muscles, controls, and animation principles in portraying various emotions realistically.

Ans: → conveying emotions through facial expressions is a fundamental aspect of character animation that requires a deep understanding of human anatomy, psychology and animation principles Here's how different elements contribute to portraying various emotions realistically - facial muscles, controls / Rigging Animation principles

SILVER OAK UNIVERSITY
EDUCATION TO INNOVATION

Conclusion:

In this practical, we learn about the character Animation and facial Animation.

Marks out of 10	
Signature with Date of Completion	

traditional 2D drawings or even 3D models on a computer screen.

Time and Cost Savings:

- Iterative design changes were made efficiently in VR, reducing the need for costly physical models or time-consuming revisions during the construction phase.

Challenges:

Learning Curve:

- Initially, there was a learning curve for both the design team and the client in using VR technology. However, training sessions mitigated this challenge.

Hardware Requirements:

- Access to VR headsets and powerful computers was necessary, but advancements in technology are making these tools more accessible.

Conclusion:

The integration of Virtual Reality in the architectural design process proved transformative in this case study. The immersive nature of VR facilitated better communication, improved decision-making, and ultimately led to a house design that precisely matched the client's vision. As VR technology continues to advance, its adoption in the field of architecture is likely to become more widespread, offering new possibilities for innovation and efficiency in the design and construction industry.

Post Practical Questions:

1. Explain the essential components and technologies involved in creating a virtual reality (VR) experience for architectural visualization. Discuss the role of VR headsets, controllers, and immersive environments.

Ans: In architectural visualization, VR offers a powerful tool for presenting designs in an immersive and interactive manner. The essential components and technologies involved in creating a VR experience for architectural visualization include: VR Headsets, controllers, immersive Environments, Real time Rendering

2. How can users interact with the virtual architecture in your case study? Discuss the user interface, navigation methods, and any interactive elements incorporated to enhance the VR experience.

Ans: In our case study of architectural visualization in VR, users can interact with the virtual architecture through intuitive controls and immersive navigation methods. The following elements are incorporated to enhance the VR experience - User- Interface (UI

Navigation Method, interactive elements, collaborative features and Accessibility options.

3. Evaluate the benefits and challenges of using virtual reality for architectural visualization. Discuss how VR enhances the understanding of architectural designs and potential drawbacks or limitations.

Ans: → Benefits of using virtual Reality for Architectural Visualization

- immersive experience - Realistic Representation
- interactive exploration - early design evaluation
- Remote collaboration

→ challenges and limitations of virtual Reality for Architectural visualization

- cost and Accessibility.
- Technical limitations
- simulation sickness
- learning curve
- Subjective Interpretation

Marks out of 10	10
Signature with Date of Completion	Dg