MARRI LAXMAN REDDY
INSTITUTE OF TECHNOLOGY AND MANAGEMENT
(AN AUTONOMOUS INSTITUTION)
(Approved by AICTE, New Delhi & Affiliated to JNTUH, Hyderabad)
Accredited by NAAC with 'A' Grade & Recognized Under Section 2(f) & 12(B) of the UGC act,1956

Department of Computer Science and Engineering(AIML)

AUGMENTED REALITY AND VIRTUAL REALITY LABORATARY

III B.TECH -II SEMESTER (AIML)

R24 (MLRS) REGULATION

A.Y: 2026 - 2027

INDEX

10.	Explore VR environment using Oculus. Experience basic interactions and movement in virtual environments.	46
11.	Install Vuforia in Unity, generate license key, create and configure database. Database. Place 3D models on image targets.	50
12.	Interact with augmented objects in the real world. Build and test AR application on mobile device or simulator.	54
OPEN ENDED EXPERIMENTS		
1.		
2.		
3.		

Department of Computer Science & Engineering (AI&ML)

CERTIFICATE

This is to certify that this manual is a Bonafide record of practical work carried out **in AUGMENTED REALITY AND VIRTUAL REALITY LABORATARY** for the **B. Tech Computer Science Engineering (AI&ML) VI Semester** Programme during the academic year **2026–2027.**

This manual has been prepared by **Mrs. B Soundarya (Assistant Professor),** Department of Computer Science Engineering (AI&ML), with my own efforts and to the best of our knowledge.

Signature of Lab Faculty **Signature of HOD**

PREFACE

This Lab Manual entitled "Augmented Reality And Virtual Reality Laboratory" is intended for the use of III B. Tech II Semester Computer Science and Engineering (AIML) students of Marri Laxman Reddy Institute of Technology and Management, Dundigal, Hyderabad. The main objective of

The field of Augmented Reality (AR) and Virtual Reality (VR) has emerged as one of the most transformative technologies in modern computing, enabling immersive and interactive experiences across various domains such as gaming, education, healthcare, and industrial training. This laboratory course is designed to provide students with hands-on experience in developing real-time 2D/3D applications and immersive environments using industry-standard tools.

By,

Mrs. B Soundarya

ACKNOWLEDGEMENT

It was really a good experience, working at Augmented Reality And Virtual Reality Laboratory. First, I would like to express my sincere thanks to Dr. B Ravi Prasad, Head of the Department of Computer Science & Engineering (AI&ML), Marri Laxman Reddy Institute of technology & Management, for his concern towards me and gave me opportunity to prepare Augmented Reality And Virtual Reality laboratory manual.

I am deeply indebted and gratefully acknowledge the constant support and valuable patronage of Dr. B Ravi Prasad, Dean Academics, Marri Laxman Reddy Institute of technology & Management. I am unboundedly grateful to him for timely corrections and scholarly guidance.

I express my heartful thanks to Dr. P. Sridhar, Director, and Dr. R. Murali Prasad, Principal, Marri Laxman Reddy Institute of technology & Management, for giving me this wonderful opportunity for preparing the Augmented Reality And Virtual Reality laboratory manual.

At last, but not the least I would like to thank the entire Computer Science & Engineering Department (AI&ML) faculties those who had inspired and helped me to achieve my goal.

By,

Mrs. B Soundarya

Department of Computer Science
& Engineering (AI&ML)

GENERAL INSTRUCTIONS

1. Students are instructed to come to Augmented Reality And Virtual Reality Laboratory on time. Late comers are not entertained in the lab.
2. Students should be punctual to the lab. If not, the conducted experiments will not be repeated.
3. Students are expected to come prepared at home with the experiments which are going to be performed.
4. Students are instructed to display their identity cards before entering into the lab.
5. Students are instructed not to bring mobile phones to the lab.
6. Any damage/loss of system parts like keyboard, mouse during the lab session, it is student's responsibility and penalty or fine will be collected from the student.
7. Students should update the records and lab observation books session wise. Before leaving the lab the student should get his/her lab observation book signed by the faculty.
8. Students should submit the lab records by the next lab to the concerned faculty members in the staff room for their correction and return.
9. Students should not move around the lab during the lab session.
10. If any emergency arises, the student should take the permission from faculty member concerned in written format.
11. The faculty members may suspend any student from the lab session on disciplinary grounds.
12. Never copy the output from other students. Write down your own outputs.

Department of Computer Science & Engineering (AI&ML)

SAFETY MEASURES

To ensure the safe and efficient use of the Computer Science and Engineering (AI&ML) laboratory, all students must strictly adhere to the following safety guidelines:

1. General Conduct

- Maintain silence and discipline during lab sessions.
- Do not bring food, drinks, or chewing gum into the lab.
- Use lab resources responsibly and follow all instructions provided by the instructor or lab assistant.

2. Electrical Safety

- Do not touch electrical switches, sockets, or plugs with wet hands.
- Avoid overloading power sockets with unauthorized devices.
- Immediately report any loose connections, sparks, or unusual noises from equipment.

3. Computer and Equipment Handling

- Handle all computer systems, keyboards, mice, and peripherals with care.
- Do not attempt to open or tamper with any hardware components.
- Use only the assigned computer system; do not switch systems without permission.

4. Software and Data Safety

- Use only authorized software installed by the lab administrator.
- Do not attempt to install, uninstall, or modify any software without approval.
- Save your work frequently and ensure backups of important files.

5. Cybersecurity and Network Usage

- Keep your login credentials confidential.
- Do not attempt to access restricted websites or server
- Avoid activities such as hacking, gaming, or the use of pirated content.

6. Emergency Preparedness

- Be familiar with the location of emergency exits, fire extinguishers, and first aid kits.
- In the event of a fire, electrical hazard, or anyemergency, remain calm and inform the lab instructor immediately.
- Follow the evacuation procedure as instructed.

7. Post-Lab Procedures

- Log out of your session and shut down the system properly after use.
- Leave your workstation clean and organized.
- Return any borrowed materials or equipment to their proper place.

8. Hygiene and Cleanliness

- Wash or sanitize your hands before and after using shared devices.
- Do not write or place unnecessary items on the workstation.
- Report any spills or cleanliness issues to the lab staff.

Department of Computer Science & Engineering (AI&ML)

VISION & MISSION OF THE INSTITUTE

Vision of the Institute:

To be a globally recognized institution that fosters innovation, excellence, and leadership in education, research, and technology development, empowering students to create sustainable solutions for the advancement of society.

Mission of the Institute:

- To foster a transformative learning environment that empowers students to excel in engineering, innovation, and leadership.
- To produce skilled, ethical, and socially responsible engineers who contribute to sustainable technological advancements and address global challenges.
- To shape future leaders through cutting-edge research, industry collaboration, and community engagement.

Department of Computer Science & Engineering (AI&ML)

VISION & MISSION OF THE DEPARTMENT

Department Vision:

To nurture globally competent professionals in Artificial Intelligence and Machine Learning through excellence in education, research, and innovation, committed to developing sustainable and impactful solutions for the betterment of society.

Department Mission:

- To provide a transformative learning environment that equips students with in-depth knowledge and practical skills in Artificial Intelligence and Machine Learning, fostering innovation, leadership, and lifelong learning.
- To advance AI and ML through cutting-edge research, strong industry collaboration, and community engagement, preparing students to address real-world challenges on a global scale.
- To produce competent and ethical AI professionals who contribute to technological progress while addressing societal and environmental challenges with sustainable solutions.
- To foster a research-driven culture by partnering with industry and academia, encouraging entrepreneurship, and engaging in community-centered technology development.

Program Outcomes (POs)

PO:

PO1.Engineering Knowledge:

Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.

PO 2. Problem Analysis:

Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.

PO 3. Design /Development of Solutions:

Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.

PO 4. Conduct Investigations of Complex Problems:

Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.

PO 5. Modern Tool Usage:

Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.

PO 6. The Engineer and Society:

Apply reasoning informed by the contextual knowledge to assess societal, health,

safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.

PO 7. Environment and Sustainability:

Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

PO 8.Ethics:

Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

PO 9. Individual and Team Work:

Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

PO 10. Communication:

Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

PO 11. Project Management and Finance:

Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

PO 12. Life-long Learning:

Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

Program Educational Objectives (PEOs)

PEO:

Professional Competence:

Graduates will possess strong theoretical and practical knowledge in Artificial Intelligence and Machine Learning, enabling them to solve complex real-world problems, pursue higher education, or excel in professional careers.

Innovation and Research Orientation:

Graduates will engage in innovative practices, cutting-edge research, and contribute to the advancement of AI and ML technologies through collaboration with industry and academia.

Leadership and Lifelong Learning:

Graduates will exhibit leadership qualities, effective communication, and teamwork skills, and will continuously upgrade their knowledge to adapt to evolving technological landscapes.

Entrepreneurial and Community Engagement:

Graduates will leverage entrepreneurial skills and a sense of civic responsibility to create AI-driven solutions that benefit local and global communities.

PROGRAM SPECIFIC OUTCOMES (PSOs)

PSO:

PSO 1: Able to identify, analyze and solve the problems related to Artificial Intelligence and Machine Learning by applying the fundamental knowledge of Computer Science and Engineering.

PSO 2: Able to use innovative tools and techniques to build project models in the areas related to Deep Learning, Machine learning, Artificial Intelligence.

PSO3: Able to apply their Artificial Intelligence and Machine Learning knowledge in interdisciplinary domains to address societal, environmental, health, safety challenges while aligning Sustainable development goals.

MARRI LAXMAN REDDY
INSTITUTE OF TECHNOLOGY AND MANAGEMENT
(AN AUTONOMOUS INSTITUTION)
(Approved by AICTE, New Delhi & Affiliated to JNTUH, Hyderabad)
Accredited by NBA and NAAC with 'A' Grade & Recognized Under Section2(f) & 12(B)of the UGC act,1956

ACADEMIC CALENDER 2026 – 2027

MARRI LAXMAN REDDY
INSTITUTE OF TECHNOLOGY AND MANAGEMENT
(AN AUTONOMOUS INSTITUTION)
(Approved by AICTE, New Delhi & Affiliated to JNTUH, Hyderabad)
Accredited by NBA and NAAC with 'A' Grade & Recognized Under Section2(f) & 12(B)of the UGC act,1956

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING (AI&ML)

B.TECH VI SEMESTER

CSM TIME TABLE

ACADEMIC YEAR: 2026-2027

2466675: AUGMENTED REALITY AND VIRTUAL REALITY LABORATARY

L T P C
0 0 2 1

COURSE OUTCOMES: After Completion of the course, student should be able to

- Understand Unity environment and interface for creating 2D/3D applications.
- Develop interactive games with physics and animation using Unity.
- Implement UI elements and score systems in game environments.
- Explore AR/VR platforms and develop basic immersive applications.
- Design complete game applications using Unity.

LIST OF EXPERIMENTS

1. Install Unity and Visual Studio. Explore Unity 2D/3D templates. Overview of Unity Editor: Scene, Game, Hierarchy, Project, Inspector, Game Objects, Components.
2. Create simple 3D objects. Apply transformations. Add basic lighting, materials, and shaders. Understand how to use prefabs and assets.
3. Add Rigidbody, Colliders, Physics Materials. Apply forces and detect collisions using OnCollisionEnter. Demonstrate physics joints and triggers.
4. Set up project structure with folders (Materials, Prefabs, Scripts, Scenes). Create environment and moving player. Write movement scripts.
5. Move the camera, build play area boundaries. Create and collect pick-up objects (gems), update score, and display on screen using UI Text.
6. Add an enemy to follow the player using NavMesh or scripting. Create a health bar and display it. Show "Game Over" and "You Win" conditions.
7. Create a complete UI with canvas, buttons, health bar, score, and pop-ups. Script UI interactions using C#.
8. Build the Infinite Runner game into a Windows executable. Package

resources. Test for debugging and playability.

9. Define AR & VR with real-time examples. Introduction to tools like Oculus, Vuforia, Kudan, Wikitude, ARKit, and ARCore.

10 Explore VR environment using Oculus. Experience basic interactions and movement in virtual environments.

11 Install Vuforia in Unity, generate license key, create and configure database. database. Place 3D models on image targets.

12 Interact with augmented objects in the real world. Build and test AR application on mobile device or simulator.

TEXTBOOKS

1. Steven M. LaValle, Virtual Reality, Cambridge University Press, 2016.
2. William R. Sherman & Alan B. Craig, Understanding Virtual Reality, Morgan Kaufmann, 2002.
3. Alan B. Craig et al., Developing Virtual Reality Applications, Morgan Kaufmann, 2009.
4. Allan Fowler, AR Game Development, Apress, 2018. ISBN: 978-1484236178.
5. Schmalstieg & Hollerer, Augmented Reality: Principles & Practice, Pearson, 2016. ISBN-10: 933257849.

REFERENCE BOOKS

1. Gerard Jounghyun Kim, Designing Virtual Systems, 2005.
2. Doug A. Bowman et al., 3D User Interfaces: Theory and Practice, Addison Wesley, 2005.
3. Oliver Bimber & Ramesh Raskar, Spatial Augmented Reality, 2005.
4. Grigore C. Burdea & Philippe Coiffet, Virtual Reality Technology, Wiley, 2003.
5. Kharis O'Connell, Designing for Mixed Reality, O'Reilly Media, 2016.
6. Sanni Siltanen, Theory and Applications of Marker-Based AR, 2012. ISBN: 978-951-38-7449-0.

MARRI LAXMAN REDDY
INSTITUTE OF TECHNOLOGY AND MANAGEMENT
(AN AUTONOMOUS INSTITUTION)
(Approved by AICTE, New Delhi & Affiliated to JNTUH, Hyderabad)
Accredited by NBA and NAAC with 'A' Grade & Recognized Under Section2(f) & 12(B)of the UGC act,1956

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING (AI&ML)

Virtual lab details

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING (AI&ML)

Augmented Reality And Virtual Reality Laboratory

LAB PLANNER CSM- BATCH-1

S. No	Experiment	CO	Virtual Lab Availabilty	Date planned	Date Conducted
1	Install Unity and Visual Studio. Explore Unity 2D/3D templates. Overview of Unity Editor: Scene, Game, Hierarchy, Project, Inspector, Game Objects, Components.	CO1	NA		
2	Create simple 3D objects. Apply transformations. Add basic lighting, materials, and shaders. Understand how to use prefabs and assets.	CO1	NA		
3	Add Rigidbody, Colliders, Physics Materials. Apply forces and detect collisions using OnCollisionEnter. Demonstrate physics joints and triggers.	CO2	NA		
4	Set up project structure with folders (Materials, Prefabs, Scripts, Scenes). Create environment and moving player. Write movement scripts.	CO2	NA		
5	Move the camera, build play area boundaries. Create and collect pick-up objects (gems), update score, and display on screen using UI Text.	CO3	NA		
6	Add an enemy to follow the player using NavMesh or scripting. Create a health bar and display it. Show “Game Over” and “You Win” conditions.	CO3			
MID-I					

7	Create a complete UI with canvas, buttons, health bar, score, and pop-ups. Script UI interactions using C#.	CO3	NA		
8	Building Infinite Runner Game into Windows Executable, Packaging Resources, and Testing for Debugging & Playability	CO4	NA		
9	Define AR & VR with real-time examples. Introduction to tools like Oculus, Vuforia, Kudan, Wikitude, ARKit, and ARCore.	CO4	NA		
10	Exploring VR Environment Using Oculus and Experiencing Basic Interactions & Movement	CO5	NA		
11	Installation of Vuforia in Unity, License Key Generation, Database Creation, and Placing 3D Models on Image Targets	CO5	NA		
12	Interacting with Augmented Objects in Real World and Building & Testing AR Application on Mobile Device	CO5			
MID-II					

Augmented Reality And Virtual Reality Laboratory

LAB PLANNER CSM- BATCH-2

S.No	Experiment	CO	Virtual Lab Availabilty	Date planned	Date Conducted
1	Install Unity and Visual Studio. Explore Unity 2D/3D templates. Overview of Unity Editor: Scene, Game, Hierarchy, Project, Inspector, Game Objects, Components.	CO1	NA		
2	Create simple 3D objects. Apply transformations. Add basic lighting, materials, and shaders. Understand how to use prefabs and assets.	CO1	NA		
3	Add Rigidbody, Colliders, Physics Materials. Apply forces and detect collisions using OnCollisionEnter. Demonstrate physics joints and triggers.	CO2	NA		
4	Set up project structure with folders (Materials, Prefabs, Scripts, Scenes). Create environment and moving player. Write movement scripts.	CO2	NA		
5	Move the camera, build play area boundaries. Create and collect pick-up objects (gems), update score, and display on screen using UI Text.	CO3	NA		
6	Add an enemy to follow the player using NavMesh or scripting. Create a health bar and display it. Show “Game Over” and “You Win” conditions.	CO3			
MID-I					

7	Create a complete UI with canvas, buttons, health bar, score, and pop-ups. Script UI interactions using C#.	CO3	NA		
8	Building Infinite Runner Game into Windows Executable, Packaging Resources, and Testing for Debugging & Playability	CO4	NA		
9	Define AR & VR with real-time examples. Introduction to tools like Oculus, Vuforia, Kudan, Wikitude, ARKit, and ARCore.	CO4	NA		
10	Exploring VR Environment Using Oculus and Experiencing Basic Interactions & Movement	CO5	NA		
11	Installation of Vuforia in Unity, License Key Generation, Database Creation, and Placing 3D Models on Image Targets	CO5	NA		
12	Interacting with Augmented Objects in Real World and Building & Testing AR Application on Mobile Device	CO5			
MID-II					

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING (AI&ML)

Augmented Reality And Virtual Reality Laboratory

Date planed																																
Date conducted																																
Roll Number	Exp No	CO	VL*	Exp No	CO	VL*	Exp No	CO	VL*	Exp No	CO	VL*	Exp No	CO	VL*		Exp No	CO	VL*	Exp No	CO	VL*	Exp No	CO	VL*	Exp No	CO	VL*	Exp No	CO	VL*	

Note: VL *-Virtual Lab Availability

Augmented Reality And Virtual Reality Laboratory

RUBRICS USED TO ASSESS LEARNINGS IN LABORATORIES

1. RUBRICS FOR DAY TO DAY EVALUATION

Parameter	Max Marks	Level-1 (Very Poor)	Level-2 (Poor)	Level-3 (Average)	Level-4 (Good)	Level-5 (Excellent)
Observation Book	05	No observations or irrelevant data. (0–1)	Incomplete or incorrect data. (2)	Basic values with some errors. (3)	Mostly correct with good format. (4)	Fully correct, clear, and well-formatted. (5)
Record Writing	05	Not submitted. (0-1)	Submitted but mostly incomplete. (2)	Submitted with some missing/wrong parts. (3)	Submitted with minor issues. (4)	Fully complete, correct algorithm & flowchart. (5)
Result	05	No result or major errors. (0–1)	Result partially obtained. (2)	Acceptable result with limited error. (3)	Near-correct result and reasonable error. (4)	Accurate result. (5)
Viva-Voce	05	Did not answer any questions. (1)	Answered very few questions. (2)	Answered some questions with help. (3)	Answered most questions correctly. (4)	Answered all questions accurately. (5)

1. RUBRICS FOR INTERNAL EVALUATION

Criterion	Max Marks	Level-1 *(Very Poor)*	Level-2 *(Poor)*	Level-3 *(Average)*	Level-4 *(Good)*	Level-5 *(Excellent)*
Design/Tool/Apparat us Selection	2 Marks	Incorrect tool/design and no reasoning. **(0)**	Tool/desig n selection attempte d with unclear logic. **(0.5)**	Satisfacto r y selection with partial justificati o n. **(1)**	Correct selection and proper analysis with few errors. **(1.5)**	Smart selection with accurate, relevant analysis. **(2)**
Execution (Code/Debug/Run) /Analysis/Method Used	4 Marks	Did not attempt or complete ly failed to execute. **(0)**	Attempte d but unable to proceed or with major errors. **(1)**	Partial execution with some logic/syn ta x errors. **(2)**	Mostly correct executio n with minimal help. **(3)**	Fully correct and independentl y executed program. **(4)**
Results& Documentation	2 Marks	Incomplete or poorly presented. **(0)**	Basic structure but lacks clarity or formattin g. **(0.5)**	Complete but genericor with formattin g issues. **(1)**	Well-structure d and mostly clear. **(1.5)**	Well-organized, professional , and engaging documentat io n. **(2)**
Viva-Voce (Understandin g of Concepts)	2 Marks	No understandi n g; could not answer questions. **(0)**	Answere d a few with difficulty . **(0.5)**	Answere d half the question s with basic clarity. **(1)**	Good understandi n g with confident answers. **(1.5)**	Answered all questions with clarity and depth. **(2)**

1. RUBRICS FOR SEMESTER END EXAMINATIONS

Criterion	Max Marks	Level-1 (Very Poor (*0–2 marks)*	Level-2 (Poor)*(3–4 marks)*	Level-3 (Average)*(5–6 marks)*	Level-4 (Good)*(7–9 marks)*	Level-5 (Excellent)*(1 0–12 marks)*
Preparedness for the Experiment	12 marks	No clarity or objective or procedure. Unable to explain basics.	Limited idea of the objective/procedure. Needed prompting.	Has basic understanding; minor gaps in concept or preparation.	Well-prepared, with clear understandin gof steps and background.	Fully prepared with strong conceptual clarity and confident explanation.
Performance in the Laboratory	12 marks	Unable to perform experiment. Relied entirely on examiner's help.	Performed with multiple errors and constant support.	Performed with some errors; required occasional help.	Performed mostly independently with minimal support.	Performed independently, efficiently, and with precision.
Calculations & Graphs	12 marks	No or incorrect calculations. Graphs missing or irrelevant.	Multiple calculation errors. Graphs/plots inaccurate or poorly labeled.	Calculations partially correct. Graphs present but with some flaws.	Correct calculations and graphs with minor errors.	Accurate calculations and well-labeled graphs with proper interpretation.

Results & Error Analysis	12 marks	No result or invalid result. No error analysis attempted.	Incorrect result with vague or no error discussion.	Acceptable result. Error analysis attempted but limited.	Correct result with sound error discussion.	Accurate result with detailed and relevant error analysis.
Viva-Voce (Subject Knowledge)	12 marks	Unable to answer any questions. No conceptual understanding.	Answered few questions with poor logic.	Answered half of the questions with average understanding.	Answered most questions with clarity and confidence.	Answered all questions with depth, clarity,and reasoning.

Experiment - 1

Install Unity and Visual Studio. Explore Unity 2D/3D templates. Overview of Unity Editor: Scene, Game, Hierarchy, Project, Inspector, Game Objects, Components.

Aim

To install Unity and Visual Studio, explore Unity 2D/3D templates, and understand the Unity Editor interface including Scene, Game, Hierarchy, Project, Inspector, GameObjects, and Components.

Objectives

- Install Unity and Visual Studio
- Explore Unity 2D and 3D templates
- Understand Unity Editor interface
- Learn about GameObjects and Components

Software Requirements

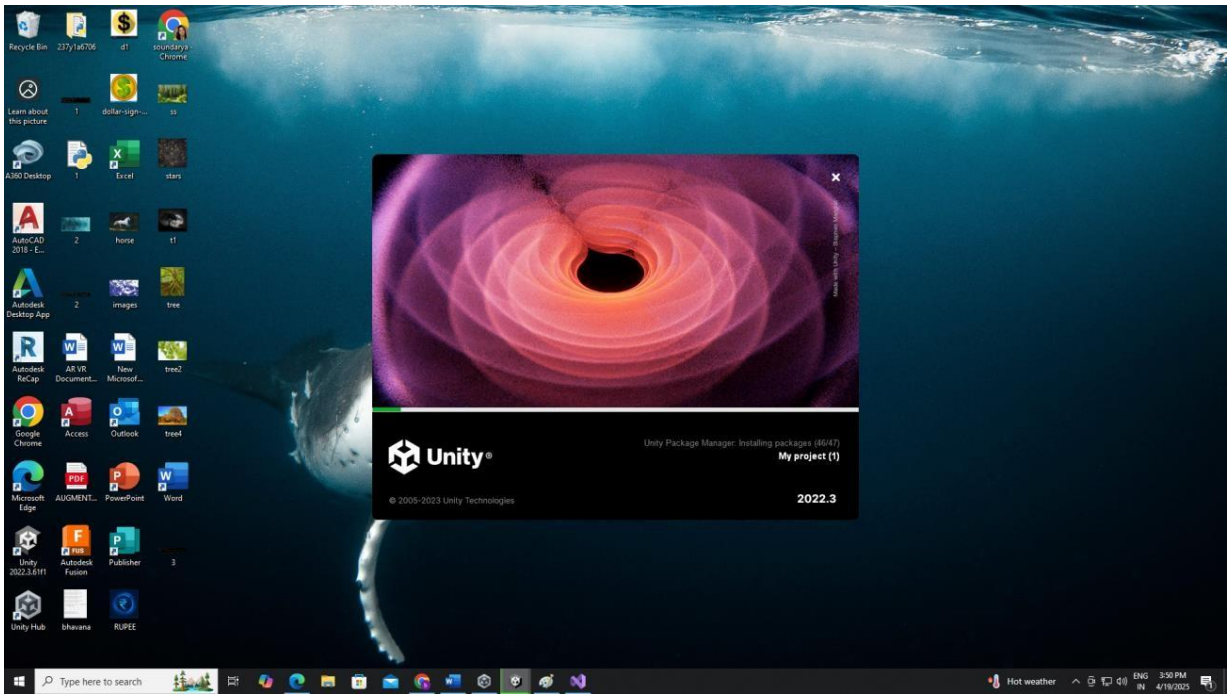
- Unity Hub
- Unity Editor
- Visual Studio (or Visual Studio Code)
- Internet connection

Theory

1. Unity

Unity is a cross-platform game engine used for:

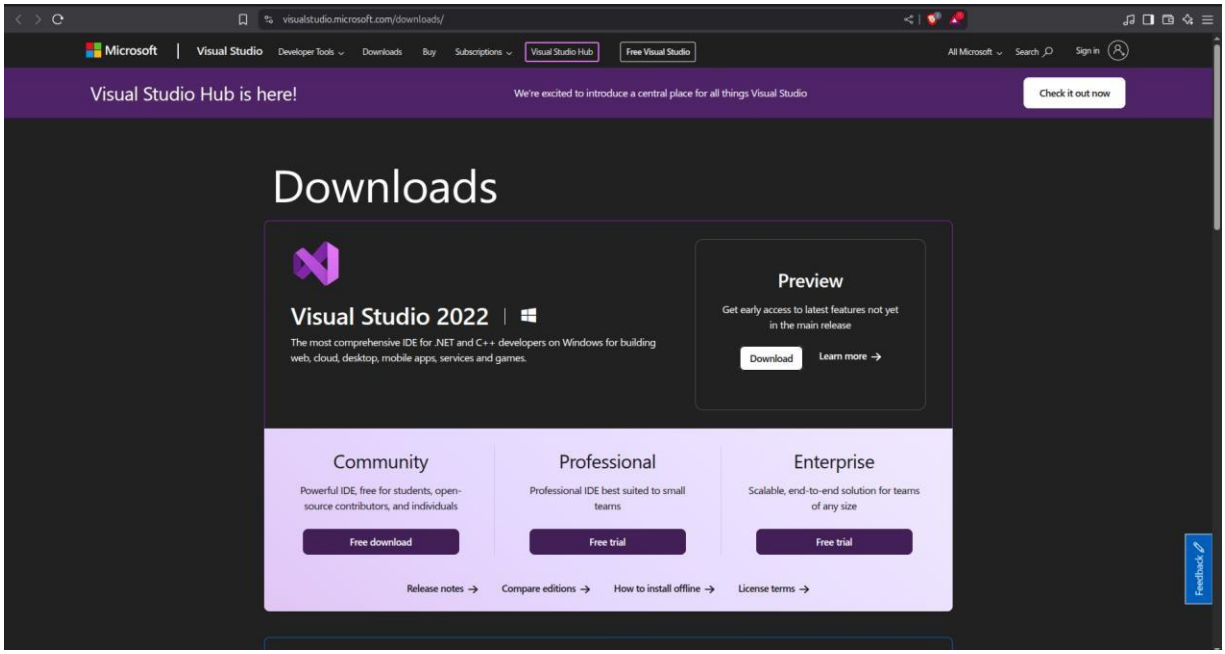
- Game development
- AR/VR applications
- Simulation systems



2. Visual Studio

Visual Studio is an IDE used for:

- Writing C# scripts
- Debugging Unity programs



3. Unity Templates

- **2D Template** → Used for 2D games
- **3D Template** → Used for 3D environments and VR

4. Unity Editor Components

1. Scene Window

- Used to design and arrange objects
- Shows development view

2. Game Window

- Displays final output
- Represents player view

3. Hierarchy Window

- Lists all GameObjects in the scene
- Shows parent-child relationships

4. Project Window

- Contains all assets (scripts, materials, models)

5. Inspector Window

- Displays properties of selected object
- Used to modify components

5. GameObjects

- Basic elements in Unity scenes
- Examples: Cube, Camera, Light

6. Components

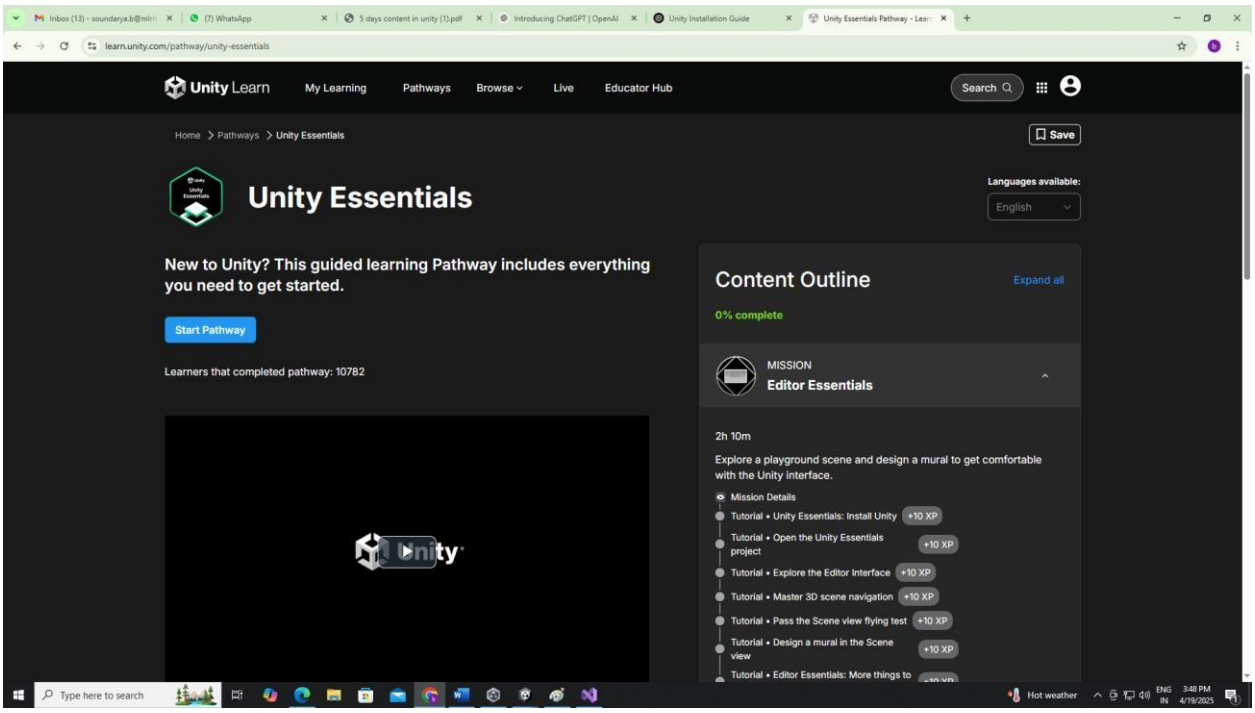
- Functional units attached to GameObjects
- Example:
  - Transform
  - Rigidbody
  - Collider

Procedure

Step 1: Install Unity Hub

1.Download Unity Hub from official website

2.Install and launch



Step 2: Install Unity Editor

1.Open Unity Hub

2.Go to **Installs → Add Version**

3.Select latest version

4.Include modules:

- Android Build Support
- Windows Build Support

Step 3: Install Visual Studio

1.Install Visual Studio

2.Select:

- .NET Desktop Development
- Game Development with Unity

Step 4: Create New Project

1.Open Unity Hub

2.Click **New Project**

3.Select:

- 2D Template
- 3D Template

4.Name project

Step 5: Explore Unity Editor

1.Open project

2.Observe:

Scene view, Game view, Hierarchy, Inspector, Project window

Step 6: Create GameObjects

1.Right-click Hierarchy

2.Add: Cube, Sphere

3.Observe properties

Step 7: Add Components

1.Select object

2.Click **Add Component**

3.Add Rigidbody or Collider

Output

- Unity and Visual Studio installed
- 2D/3D projects created
- Unity Editor interface explored
- GameObjects and components created

Result

Successfully installed Unity and Visual Studio and understood Unity Editor interface and basic concepts.

Applications

- Game development
- AR/VR applications

- Simulation and training systems

VIVA QUESTIONS

S.No	Question	CO	Bloom's Taxonomy
1	What is Unity?	CO1	Remember
2	What are the applications of Unity?	CO1	Understand
3	What is the role of Visual Studio in Unity?	CO1	Understand
4	What are Unity templates?	CO1	Remember
5	Difference between 2D and 3D templates in Unity.	CO1	Analyze
6	What is the Scene window?	CO1	Remember
7	What is the Game window?	CO1	Understand
8	Difference between Scene and Game view.	CO1	Analyze
9	What is the Hierarchy window?	CO1	Remember
10	What is the Project window?	CO1	Understand
11	What is the Inspector window?	CO1	Understand
12	What is a GameObject?	CO1	Remember
13	Give examples of GameObjects.	CO1	Understand
14	What are components in Unity?	CO1	Remember
15	What is the Transform component?	CO1	Understand
16	How do you add components to a GameObject?	CO1	Apply
17	What is the purpose of the toolbar in Unity?	CO1	Understand
18	How do you create a new project in Unity?	CO1	Apply
19	What are assets in Unity?	CO1	Remember
20	How can Unity Editor efficiency be improved?	CO1	Evaluate

Experiment - 2

Create simple 3D objects. Apply transformations. Add basic lighting, materials, and shaders. Understand how to use prefabs and assets.

Aim

To create basic 3D objects in Unity, apply transformations, add lighting and materials, understand shaders, and learn the use of prefabs and assets.

Objectives

- Create simple 3D objects in Unity.
- Apply transformations (Position, Rotation, Scale).
- Add lighting to the scene.
- Apply materials and understand shaders.
- Create and use prefabs.
- Import and manage assets.

Software Requirements

- Unity Hub
- Unity Editor (3D Core Template)
- Basic system with GPU support

Theory

1. 3D Objects

In Unity, 3D objects are called **GameObjects**. Basic shapes include:

- Cube
- Sphere
- Capsule
- Cylinder
- Plane

These are used to build scenes and environments.

2. Transformations

Every GameObject has a **Transform Component**:

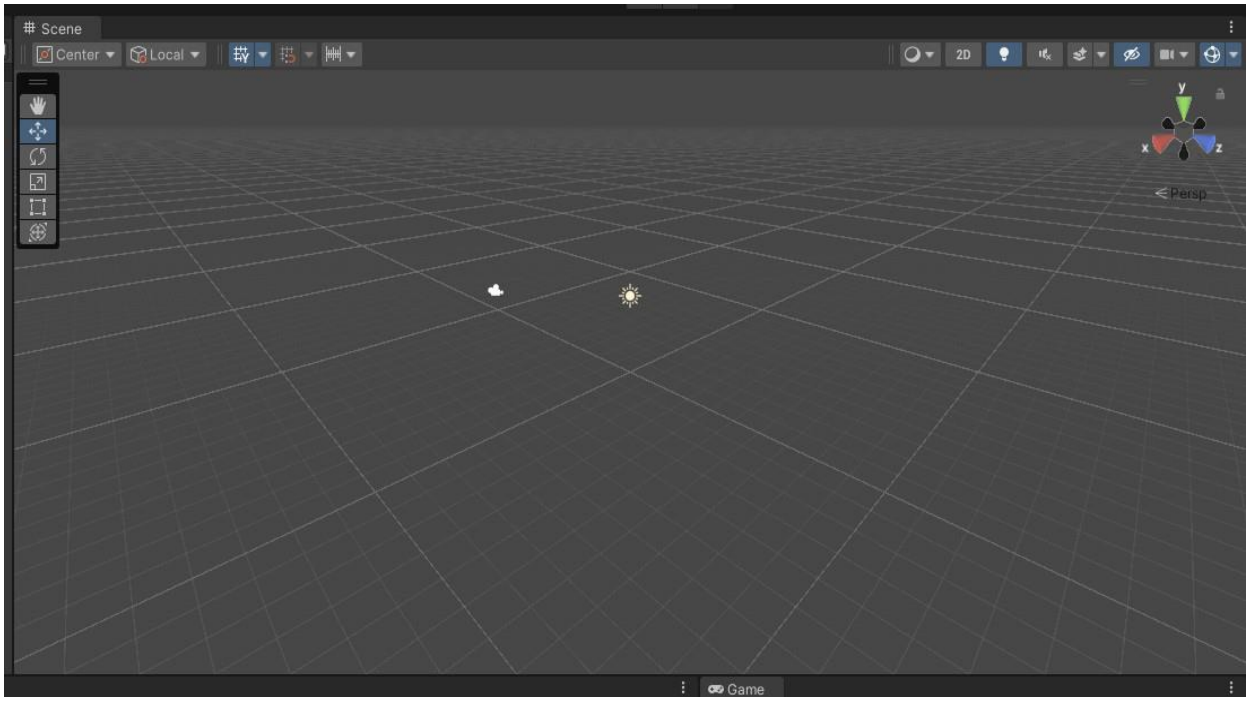
- **Position** → Moves object in X, Y, Z axes

- **Rotation** → Rotates object
- **Scale** → Changes size of object

3. Lighting

Lighting improves realism in a scene:

- Directional Light (Sunlight)
- Point Light (Bulb)
- Spot Light (Flashlight)



4. Materials & Shaders

- **Material** defines surface appearance.
- **Shader** controls how light interacts with the material.

Example:

- Metallic surface
- Matte surface

5. Prefabs

Prefabs are reusable GameObjects.

- Save time by reusing objects
- Maintain consistency

6. Assets

Assets include:

- Models
- Textures
- Scripts
- Materials

Imported via **Assets Folder**

Procedure

Step 1: Create New Project

1.Open Unity Hub

2.Click **New Project**

3.Select **3D Core Template**

4.Name: *Basic3DScene*

5.Click **Create**

Step 2: Create 3D Objects

1.Right-click in Hierarchy → **3D Object**

2.Add:

- Cube
- Sphere
- Cylinder

3.Rename objects appropriately

Step 3: Apply Transformations

1.Select object

2.In Inspector:

- Change Position (X, Y, Z)
- Rotate object
- Scale object

Example:

- Cube → Scale (2,2,2)
- Sphere → Position (0,1,0)

Step 4: Add Lighting

1.Go to **GameObject → Light**

2.Add:

- Directional Light

3.Adjust:

- Intensity
- Rotation

Step 5: Apply Materials

1.Right-click in Assets → Create → Material

2.Rename: *RedMaterial*

3.Change color to Red

4.Drag material onto Cube

Step 6: Understand Shaders

1.Select Material

2.In Inspector → Shader dropdown

3.Try:

- Standard
- Unlit/Color

4.Observe changes

Step 7: Create Prefab

1.Drag object from Hierarchy → Assets folder

2.Prefab is created

3.Reuse by dragging into scene

Step 8: Import Assets

1.Right-click Assets → Import New Asset

2.Add textures/models

3.Apply to objects

Output

- A 3D scene with basic objects
- Objects transformed (moved, rotated, scaled)
- Lighting applied
- Materials and shaders visible
- Prefabs created and reused

Result

Successfully created a 3D scene in Unity with transformations, lighting, materials, shaders, and prefabs.

Applications

- Game development
- AR/VR environments
- Simulation systems
- Architectural visualization

VIVA QUESTIONS

S.No	Question	CO	Bloom's Taxonomy
1	What is Unity?	CO1	Remember
2	What are the main features of Unity?	CO1	Understand
3	What is a 3D object in Unity?	CO1	Remember
4	Name some basic 3D objects available in Unity.	CO1	Remember
5	What is a GameObject?	CO1	Understand
6	What is the Hierarchy window in Unity?	CO1	Remember
7	What is the Scene view?	CO1	Understand
8	What is the Game view?	CO1	Understand
9	What is the Inspector window?	CO1	Remember
10	What is the purpose of the Project window?	CO1	Understand
11	What is meant by components in Unity?	CO1	Understand
12	What is the default component of every GameObject?	CO1	Remember
13	What is a prefab in Unity?	CO1	Understand
14	What are assets in Unity?	CO1	Remember
15	What is the difference between Scene and GameObject?	CO1	Analyze
16	What is the use of the toolbar in Unity?	CO1	Understand
17	What is meant by a Unity scene?	CO1	Remember
18	What is the role of the camera in Unity?	CO1	Understand
19	What is the purpose of creating 3D objects in Unity?	CO1	Understand
20	What is the difference between 2D and 3D objects in Unity?	CO1	Analyze

Experiment -3

Add Rigidbody, Colliders, Physics Materials. Apply forces and detect collisions using OnCollisionEnter. Demonstrate physics joints and triggers.

Aim

To understand and implement physics components such as Rigidbody, Colliders, Physics Materials, apply forces, detect collisions using scripts, and demonstrate joints and trigger interactions in Unity.

Objectives

- Understand Rigidbody and physics behavior
- Apply different types of Colliders
- Use Physics Materials for surface interaction
- Apply forces to objects
- Detect collisions using scripting
- Implement joints between objects
- Understand triggers and events

Software Requirements

- Unity Hub
- Unity Editor (3D Core Template)
- Basic C# scripting knowledge

Theory

1. Rigidbody

A Rigidbody component enables physics behavior:

- Gravity effect
- Mass, Drag, Velocity
- Allows object movement using physics

2. Colliders

Colliders define the physical shape for collision detection:

- Box Collider
- Sphere Collider
- Capsule Collider
- Mesh Collider

3. Physics Material

Controls surface properties:

- **Friction** (sliding effect)
- **Bounciness** (restitution)

4. Forces

Used to move objects:

- AddForce()
- Impulse force
- Continuous force

5. Collision Detection

Occurs when two colliders interact.

- Function used:
  OnCollisionEnter()

6. Joints

Used to connect objects:

- Fixed Joint
- Hinge Joint
- Spring Joint

7. Triggers

Special colliders that detect entry without physical collision:

- Enabled using **Is Trigger** option
- Function used:
  OnTriggerEnter()

Procedure

Step 1: Create New Project

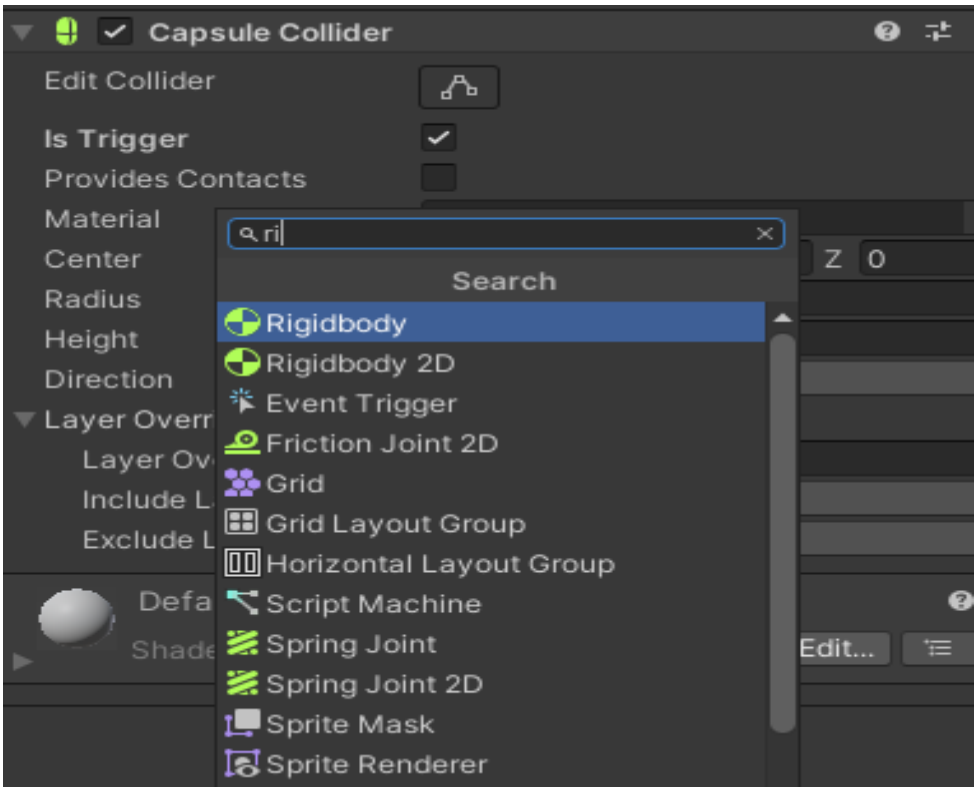
1.Open Unity Hub
2.Create a **3D Core Project**
3.Name: *PhysicsDemo*

Step 2: Create Objects

1.Add Plane (Ground)
2.Add Cube (Falling Object)
3.Add Sphere

Step 3: Add Rigidbody

1.Select Cube
2.Click **Add Component → Rigidbody**
3.Enable Gravity



Step 4: Add Colliders

1.Ensure objects have colliders:
- Cube → Box Collider
- Sphere → Sphere Collider
- Plane → Mesh Collider

Step 5: Apply Physics Material

1.Right-click Assets → Create → **Physics Material**
2.Set:
- Bounciness = 0.8

3.Apply to Sphere

Step 6: Apply Force via Script

Create C# Script: **ApplyForce.cs**

```
using UnityEngine;

  public class ApplyForce : MonoBehaviour
  {
     public float force = 500f;
```

```
    void Start()
    {
        GetComponent<Rigidbody>().AddForce(Vector3.up * force);
    }
}
```

Attach to Cube or Sphere.

Step 7: Detect Collision

Create Script: **CollisionDetect.cs**

```
using UnityEngine;

public class CollisionDetect : MonoBehaviour
{
    void OnCollisionEnter(Collision collision)
    {
        Debug.Log("Collision with: " + collision.gameObject.name);
    }
}
```

Step 8: Demonstrate Trigger

1.Add Cube → Set **Is Trigger = TRUE**
2.Create Script:

```
using UnityEngine;

public class TriggerDetect : MonoBehaviour
{
    void OnTriggerEnter(Collider other)
    {
        Debug.Log("Triggered by: " + other.name);
    }
}
```

Step 9: Add Joint

1.Select Sphere
2.Add Component → **Hinge Joint**
3.Connect to another object

Output

- Objects interact with gravity
- Collisions detected and printed
- Objects bounce based on Physics Material
- Trigger events detected

- Connected objects move using joints

Result

Successfully implemented physics components, collision detection, forces, joints, and triggers in Unity.

Applications

- Game physics (jumping, falling, bouncing)
- AR/VR simulations
- Robotics simulations
- Real-time interaction systems

VIVA QUESTIONS

S.No	Question	CO	Bloom's Taxonomy
1	What is Rigidbody in Unity?	CO2	Remember
2	What are the properties of a Rigidbody?	CO2	Understand
3	What is a Collider?	CO2	Remember
4	Name different types of Colliders.	CO2	Remember
5	What is the purpose of a Collider?	CO2	Understand
6	What is a Physics Material?	CO2	Remember
7	What are friction and bounciness?	CO2	Understand
8	How do Physics Materials affect object behavior?	CO2	Analyze
9	What is force in Unity physics?	CO2	Remember
10	What is AddForce() method?	CO2	Understand
11	Difference between Force and Impulse?	CO2	Analyze
12	What is collision detection?	CO2	Remember
13	What is OnCollisionEnter()?	CO2	Understand
14	When is OnCollisionEnter() triggered?	CO2	Understand
15	What is a trigger in Unity?	CO2	Remember
16	What is OnTriggerEnter()?	CO2	Understand
17	Difference between collision and trigger?	CO2	Analyze
18	What are joints in Unity?	CO2	Remember
19	Name different types of joints.	CO2	Remember
20	How do joints help in physics simulation?	CO2	Evaluate

Experiment -4

Set up project structure with folders (Materials, Prefabs, Scripts, Scenes). Create environment and moving player. Write movement scripts.

Aim

To organize a Unity project using proper folder structure, create a basic environment, and implement player movement using scripts.

Objectives

- Understand project organization in Unity
- Create and manage folders (Materials, Prefabs, Scripts, Scenes)
- Design a simple 3D environment
- Create a player object
- Implement movement using C# scripting

Software Requirements

- Unity Hub
- Unity Editor (3D Core Template)
- Basic knowledge of C#

Theory

1. Project Structure

Organizing assets improves project management and scalability.

Common folders:

- **Scenes** → Stores scene files
- **Scripts** → Stores C# scripts
- **Materials** → Stores materials
- **Prefabs** → Stores reusable objects

2. Environment Creation

Environment includes:

- Ground (Plane)
- Objects (Cube, Trees, etc.)
- Lighting and Camera

3. Player Object

A player is a GameObject that interacts with the environment:

- Usually uses **Capsule** or **Cube**
- Has components like Rigidbody and Collider

4. Movement Script

Movement is implemented using:

- Input controls
- Transform or Rigidbody movement

Procedure

Step 1: Create New Project

1.Open Unity Hub
2.Click **New Project → 3D Core Template**
3.Name: *PlayerMovementProject*

Step 2: Create Folder Structure

1.In Project Window → Right-click → Create Folder
2.Create folders:
- Scenes
- Scripts
- Materials
- Prefabs

Step 3: Create Scene

1.File → Save As → Save in **Scenes** folder
2.Name: *MainScene*

Step 4: Create Environment

1.Add **Plane** → Rename as Ground
2.Add **Cube** objects as obstacles
3.Adjust scale and position

Step 5: Create Player Object

1.Add **Capsule** → Rename as Player
2.Add Components:
- Rigidbody
- Capsule Collider

Step 6: Write Movement Script

Create C# Script: **PlayerMovement.cs**

```
using UnityEngine;

  public class PlayerMovement : MonoBehaviour
  {
```

```
    public float speed = 5f;

    void Update()
    {
        float moveX = Input.GetAxis("Horizontal");
        float moveZ = Input.GetAxis("Vertical");

        Vector3 movement = new Vector3(moveX, 0, moveZ);
        transform.Translate(movement * speed * Time.deltaTime);
    }
}
```

Step 7: Attach Script

1.Drag script to Player object
2.Adjust speed in Inspector

Step 8: Create Prefab

1.Drag Player into Prefabs folder
2.Reuse if needed

Step 9: Test Movement

1.Click Play
2.Use:
- **W/A/S/D** or Arrow Keys

3.Observe movement

Output

- Organized project structure
- Created environment with ground and objects
- Player object moves using keyboard input
- Prefab created successfully

Result

Successfully structured a Unity project, created an environment, and implemented player movement using scripting.

Applications

- Game development
- AR/VR simulations
- Training environments
- Interactive applications

VIVA QUESTION

S.No	Question	CO	Bloom's Taxonomy
1	What is a Unity project structure?	CO2	Remember
2	Why is folder organization important in Unity?	CO2	Understand
3	Name common folders used in Unity projects.	CO2	Remember
4	What is the purpose of the Scenes folder?	CO2	Understand
5	What is the role of Scripts folder?	CO2	Understand
6	What is an environment in Unity?	CO2	Remember
7	How do you create a ground in Unity?	CO2	Apply
8	What objects are used to design a simple environment?	CO2	Understand
9	What is a player object?	CO2	Remember
10	Which components are required for a player object?	CO2	Understand
11	Why is Rigidbody added to a player?	CO2	Analyze
12	What is a Prefab?	CO2	Remember
13	What is a movement script?	CO2	Remember
14	What is the use of Input.GetAxis()?	CO2	Understand
15	Difference between Horizontal and Vertical inputs?	CO2	Analyze
16	What is the Update() function?	CO2	Understand
17	How does Time.deltaTime affect movement?	CO2	Analyze
18	Difference between Update() and FixedUpdate()?	CO2	Analyze
19	How do you attach a script to a GameObject?	CO2	Apply
20	How can you improve player movement in Unity?	CO2	Evaluate

Experiment - 5

Move the camera, build play area boundaries. Create and collect pick-up objects (gems), update score, and display on screen using UI Text.

Aim

To implement camera movement, define play area boundaries, create collectible objects (gems), update score, and display it using UI Text in Unity.

Objectives

- Control camera movement and follow player
- Define play area boundaries
- Create collectible objects (gems)
- Detect collection using triggers
- Update score dynamically
- Display score using UI Text

Software Requirements

- Unity Hub
- Unity Editor (3D Core Template)
- Basic knowledge of C#

Theory

1. Camera Movement

The camera is used to view the scene:

- Can follow player
- Provides perspective of gameplay

2. Play Area Boundaries

Limits player movement within a defined area:

- Prevents player from leaving scene
- Implemented using code or colliders

3. Pick-Up Objects (Gems)

- Objects that player collects
- Use **Collider with Is Trigger**
- Destroyed or disabled after collection

4. Score System

- Score increases when player collects objects
- Stored in a variable
- Updated in real-time

5. UI Text

- Displays score on screen
- Created using Canvas → Text
- Updated using script

Procedure

Step 1: Create Project

1.Open Unity Hub
2.Create **3D Core Project**
3.Name: *PickupGame*

Step 2: Create Environment

1.Add **Plane** → Ground
2.Add **Walls (Cubes)** → Boundaries
3.Arrange to form play area

Step 3: Create Player

1.Add **Capsule → Player**
2.Add:
- Rigidbody
- Collider

Step 4: Camera Follow Script

Create Script: **CameraFollow.cs**

```
using UnityEngine;

  public class CameraFollow : MonoBehaviour
  {
     public Transform player;
     public Vector3 offset;

     void LateUpdate()
     {
        transform.position = player.position + offset;
     }
  }
```

Attach to Camera and assign Player.

Step 5: Player Movement Script (with Boundaries)

```
using UnityEngine;

  public class PlayerMovement : MonoBehaviour
  {
```

```
    public float speed = 5f;
    public float boundaryX = 10f;
    public float boundaryZ = 10f;

    void Update()
    {
        float x = Input.GetAxis("Horizontal");
        float z = Input.GetAxis("Vertical");

        Vector3 move = new Vector3(x, 0, z);
        transform.Translate(move * speed * Time.deltaTime);

        // Boundaries
        float clampedX = Mathf.Clamp(transform.position.x, -boundaryX, boundaryX);
        float clampedZ = Mathf.Clamp(transform.position.z, -boundaryZ, boundaryZ);

        transform.position = new Vector3(clampedX, transform.position.y, clampedZ);
    }
}
```

Step 6: Create Pick-Up Objects

1.Add **Sphere → Gem**
2.Add:
○ Collider → Enable **Is Trigger**
3.Duplicate multiple gems

Step 7: Pickup Script

```
using UnityEngine;

public class Pickup : MonoBehaviour
{
    void OnTriggerEnter(Collider other)
    {
        if(other.CompareTag("Player"))
        {
            gameObject.SetActive(false);
        }
    }
}
```

Step 8: Score Script

```
using UnityEngine;
 using UnityEngine.UI;

public class GameManager : MonoBehaviour
{
    public int score = 0;
    public Text scoreText;

    public void AddScore(int value)
```

```
    {
        score += value;
        scoreText.text = "Score: " + score;
    }
}
```

Modify Pickup Script:

```
void OnTriggerEnter(Collider other)
{
    if(other.CompareTag("Player"))
    {
        FindObjectOfType<GameManager>().AddScore(1);
        gameObject.SetActive(false);
    }
}
```

Step 9: Create UI Text

1.Right-click Hierarchy → UI → Text
2.Rename → ScoreText
3.Adjust position (Top Left)
4.Assign to GameManager

Step 10: Test

1.Click Play
2.Move player
3.Collect gems
4.Observe score update

Output

- Camera follows player
- Player restricted within boundaries
- Gems collected using trigger
- Score increases dynamically
- Score displayed on screen

Result

Successfully implemented camera movement, play area boundaries, collectible objects, score system, and UI display in Unity.

Applications

- Game development
- AR/VR interactive environments
- Educational simulations
- Real-time tracking systems

VIVA QUESTIONS

S.No	Question	CO	Bloom's Taxonomy
1	What is the role of a camera in Unity?	CO3	Remember
2	What is camera follow?	CO3	Understand
3	What is the purpose of LateUpdate() in camera movement?	CO3	Analyze
4	What is meant by offset in camera follow?	CO3	Understand
5	What are play area boundaries?	CO3	Remember
6	Why are boundaries important in a game?	CO3	Understand
7	How can boundaries be implemented in Unity?	CO3	Apply
8	What is Mathf.Clamp()?	CO3	Understand
9	What are pick-up objects?	CO3	Remember
10	How do pick-up objects work in Unity?	CO3	Understand
11	What is the use of Is Trigger option?	CO3	Understand
12	What happens when a player collects a pick-up object?	CO3	Analyze
13	What is a score system?	CO3	Remember
14	How is score updated in Unity?	CO3	Understand
15	What is the role of GameManager script?	CO3	Analyze
16	What is UI Text in Unity?	CO3	Remember
17	How do you display score on screen?	CO3	Apply
18	What is Canvas in Unity UI?	CO3	Understand
19	How do scripts communicate to update UI?	CO3	Analyze
20	How can you enhance the scoring system further?	CO3	Evaluate

Experiment -6

Add an enemy to follow the player using NavMesh or scripting. Create a health bar and display it. Show "Game Over" and "You Win" conditions.

Aim

To implement an enemy that follows the player using NavMesh or scripting, create a health bar UI, and display "Game Over" and "You Win" conditions.

Objectives

- Implement enemy movement toward the player
- Use NavMesh or scripting for AI behavior
- Create and manage player health system
- Display health using UI elements
- Implement Game Over and Win conditions

Software Requirements

- Unity Hub
- Unity Editor (3D Core Template)
- Basic knowledge of C#

Theory

1. Enemy AI Movement

Enemy behavior can be implemented using:

- **NavMesh Agent** (pathfinding)
- Manual scripting (Vector movement)

2. NavMesh

- Used for pathfinding
- Works on baked surfaces
- Automatically avoids obstacles

3. Health System

- Health is stored as a variable
- Decreases when player is attacked
- Triggers Game Over when zero

4. UI Health Bar

- Represented using Slider
- Updates dynamically based on health

5. Game Conditions

- **Game Over** → Player health = 0
- **You Win** → Enemy defeated or goal reached

Procedure

Step 1: Create Environment

1.Create Plane (Ground)
2.Add Player (Capsule)
3.Add Enemy (Cube or Capsule)

Step 2: Setup NavMesh

1.Select Ground
2.Mark as **Navigation Static**
3.Open Navigation Window
4.Click **Bake**

Step 3: Add Enemy AI

Add Component → **NavMesh Agent** to Enemy

Create Script: **EnemyFollow.cs**

```
using UnityEngine;
  using UnityEngine.AI;

  public class EnemyFollow : MonoBehaviour
  {
     public Transform player;
     private NavMeshAgent agent;

     void Start()
     {
        agent = GetComponent<NavMeshAgent>();
     }

     void Update()
     {
        agent.SetDestination(player.position);
     }
  }
```

Alternative (Without NavMesh)

```
using UnityEngine;

  public class EnemyFollow : MonoBehaviour
  {
     public Transform player;
     public float speed = 3f;

     void Update()
```

```
    {
        transform.position = Vector3.MoveTowards(
            transform.position,
            player.position,
            speed * Time.deltaTime
        );
    }
}
```

Step 4: Create Health System

```
using UnityEngine;
using UnityEngine.UI;

public class PlayerHealth : MonoBehaviour
{
    public int health = 100;
    public Slider healthBar;

    void Update()
    {
        healthBar.value = health;
    }

    public void TakeDamage(int damage)
    {
        health -= damage;
    }
}
```

Step 5: Damage on Collision

```
void OnCollisionEnter(Collision collision)
{
    if(collision.gameObject.CompareTag("Enemy"))
    {
        GetComponent<PlayerHealth>().TakeDamage(10);
    }
}
```

Step 6: Create UI Health Bar

1.Right-click → UI → Slider
2.Rename → HealthBar
3.Set Min = 0, Max = 100
4.Assign to script

Step 7: Game Over Script

```
using UnityEngine;
using UnityEngine.UI;

public class GameManager : MonoBehaviour
{
    public Text resultText;
    public GameObject player;
```

```
    void Update()
    {
        if(player.GetComponent<PlayerHealth>().health <= 0)
        {
            resultText.text = "Game Over";
            Time.timeScale = 0;
        }
    }
}
```

Step 8: Win Condition

Modify script:

```
public GameObject enemy;

  void Update()
  {
      if(enemy == null)
      {
          resultText.text = "You Win!";
          Time.timeScale = 0;
      }
  }
```

Step 9: Test

1.Click Play
2.Enemy follows player
3.Player loses health on collision
4.Health bar updates
5.Game Over / Win message displayed

Output

- Enemy follows player
- Health bar updates dynamically
- Player loses health on collision
- Game Over message appears when health is zero
- Win condition displayed when enemy is destroyed

Result

Successfully implemented enemy AI, health system, and game state management in Unity.

Applications

- Game AI systems
- AR/VR simulations
- Interactive training systems
- Real-time decision-making applications

VIVA QUESTIONS

S.No	Question	CO	Bloom's Taxonomy
1	What is enemy AI in Unity?	CO3	Remember
2	What is NavMesh?	CO3	Remember
3	What is the purpose of NavMesh Agent?	CO3	Understand
4	How does an enemy follow the player using NavMesh?	CO3	Analyze
5	What is an alternative method to implement enemy movement?	CO3	Understand
6	What is a health system in games?	CO3	Remember
7	How is health represented in Unity?	CO3	Understand
8	What happens when player health reaches zero?	CO3	Understand
9	What is a UI Slider?	CO3	Remember
10	How is a health bar created using UI Slider?	CO3	Apply
11	How do you update the health bar dynamically?	CO3	Analyze
12	What is collision-based damage?	CO3	Remember
13	How does OnCollisionEnter() work in damage detection?	CO3	Understand
14	How can damage be applied to the player?	CO3	Apply
15	What is a Game Over condition?	CO3	Remember
16	How do you implement Game Over in Unity?	CO3	Apply
17	What is a Win condition?	CO3	Remember
18	How can a Win condition be triggered?	CO3	Apply
19	How is UI text used to display game results?	CO3	Understand
20	How can you improve enemy AI behavior?	CO3	Evaluate

Experiment - 7

Create a complete UI with canvas, buttons, health bar, score, and pop-ups. Script UI interactions using C#.

Aim

To design a complete user interface using Canvas, buttons, health bar, score display, and pop-ups, and implement UI interactions using C# scripting in Unity.

Objectives

- Understand Unity UI system (Canvas, Panels, Buttons, Text)
- Create and design UI elements
- Implement button interactions
- Display health bar and score
- Create pop-up messages (Game Over / Win)
- Control UI using C# scripts

Software Requirements

- Unity Hub
- Unity Editor (3D Core Template)
- Basic knowledge of C#

Theory

1. Canvas

- Root object for all UI elements
- Automatically scales UI for different screens

2. UI Elements

- **Text** → Displays score or messages
- **Button** → User interaction
- **Slider** → Health bar
- **Panel** → Group UI elements

3. UI Interaction

- Controlled using scripts
- Buttons trigger functions using **OnClick()**

4. Pop-ups

- Display messages like:
  - Game Over
  - You Win
- Implemented using Panels

5. Event System

- Handles input events (clicks, touches)
- Automatically added with Canvas

Procedure

Step 1: Create Project

1.Open Unity Hub
2.Create **3D Core Project**
3.Name: *UIDemo*

Step 2: Create Canvas

1.Right-click Hierarchy → UI → Canvas
2.Unity automatically adds:
○ Canvas
○ Event System

Step 3: Add UI Elements

1. Score Text

- UI → Text
- Rename → ScoreText
- Position → Top Left
- Set text: *Score: 0*

2. Health Bar

- UI → Slider
- Rename → HealthBar
- Set Min = 0, Max = 100

3. Buttons

- UI → Button
- Rename → StartButton
- Change text → "Start"

Step 4: Create Pop-up Panel

1.UI → Panel
2.Rename → GameOverPanel
3.Add Text → "Game Over"
4.Disable panel initially

Step 5: Score Script

```
using UnityEngine;
  using UnityEngine.UI;

  public class ScoreManager : MonoBehaviour
```

```
{
    public int score = 0;
    public Text scoreText;

    void Start()
    {
        UpdateScore();
    }

    public void AddScore(int value)
    {
        score += value;
        UpdateScore();
    }

    void UpdateScore()
    {
        scoreText.text = "Score: " + score;
    }
}
```

Step 6: Health Bar Script

```
using UnityEngine;
  using UnityEngine.UI;

  public class HealthManager : MonoBehaviour
  {
    public int health = 100;
    public Slider healthBar;

    void Update()
    {
        healthBar.value = health;
    }

    public void TakeDamage(int damage)
    {
        health -= damage;
    }
  }
```

Step 7: Button Interaction Script

```
using UnityEngine;

  public class ButtonManager : MonoBehaviour
  {
    public GameObject gamePanel;

    public void StartGame()
    {
        gamePanel.SetActive(true);
        Time.timeScale = 1;
```

```
    }

    public void ExitGame()
    {
        Application.Quit();
    }
}
```

Attach script to a GameObject and assign button **OnClick()** events.

Step 8: Game Over Pop-up Script

```
using UnityEngine;

public class GameOverManager : MonoBehaviour
{
    public GameObject gameOverPanel;
    public HealthManager playerHealth;

    void Update()
    {
        if(playerHealth.health <= 0)
        {
            gameOverPanel.SetActive(true);
            Time.timeScale = 0;
        }
    }
}
```

Step 9: Connect UI Elements

- Assign:
  - ScoreText → ScoreManager
  - HealthBar → HealthManager
  - Button → ButtonManager
  - Panel → GameOverManager

Step 10: Test

1.Click Play
2.Click Start button
3.Observe score update
4.Reduce health → Game Over popup

Output

- UI Canvas created
- Buttons working with scripts
- Health bar updating
- Score displayed dynamically
- Pop-up shown on Game Over

Result

Successfully created a complete UI system in Unity with interactive elements and scripting.

Applications

- Game UI systems
- AR/VR interactive interfaces
- Dashboard design
- Mobile applications

VIVA QUESTIONS

S.No	Question	CO	Bloom's Taxonomy
1	What is Canvas in Unity UI?	CO3	Remember
2	What is the role of Event System?	CO3	Understand
3	What are different UI elements in Unity?	CO3	Remember
4	What is a Panel in UI?	CO3	Understand
5	What is a Button component?	CO3	Remember
6	How does a Button interact with scripts?	CO3	Understand
7	What is OnClick() event?	CO3	Apply
8	How do you assign functions to a button?	CO3	Apply
9	What is a health bar in UI?	CO3	Remember
10	How is a Slider used as a health bar?	CO3	Understand
11	How do you update health dynamically?	CO3	Analyze
12	What is a score system in UI?	CO3	Remember
13	How is score displayed using Text?	CO3	Understand
14	How do you update UI Text using scripts?	CO3	Apply
15	What are pop-ups in Unity UI?	CO3	Remember
16	How do you create a Game Over panel?	CO3	Apply
17	How do you show/hide UI elements using scripts?	CO3	Apply
18	What is Time.timeScale used for?	CO3	Understand
19	How do UI elements communicate with scripts?	CO3	Analyze
20	How can UI design be improved for better user experience?	CO3	Evaluate

Experiment - 8

Building Infinite Runner Game into Windows Executable, Packaging Resources, and Testing for Debugging & Playability

Aim

To build an Infinite Runner game into a Windows executable file, package all required resources, and test the application for debugging and playability.

Objectives

- Understand Unity build process
- Configure build settings for Windows
- Package game assets and resources
- Generate executable (.exe) file
- Test for errors and gameplay performance

Software Requirements

- Unity Hub
- Unity Editor (3D Project)
- Windows OS

Theory

1. Build Process in Unity

- Converts project into a playable application
- Generates executable files (.exe for Windows)
- Includes all required assets and dependencies

2. Build Settings

Important options:

- Platform selection (Windows, Android, etc.)
- Scenes inclusion
- Player settings

3. Packaging Resources

- Ensures all assets (scripts, textures, models, sounds) are included
- Managed automatically by Unity during build

4. Debugging

- Identifying and fixing errors
- Checking console logs
- Testing game performance

5. Playability Testing

- Ensures smooth gameplay
- Checks controls and UI
- Verifies game logic

Procedure

Step 1: Open Project

1.Open your Infinite Runner project in Unity

Step 2: Save All Scenes

1.File → Save Scene
2.Ensure main scene is saved properly

Step 3: Open Build Settings

1.Go to **File → Build Settings**
2.Select Platform → **PC, Mac & Linux Standalone**
3.Choose **Windows**

Step 4: Add Scenes

1.Click **Add Open Scenes**
2.Ensure main game scene is included

Step 5: Configure Player Settings

1.Click **Player Settings**
2.Set:
- Company Name
- Product Name
- Resolution settings
- Icon (optional)

Step 6: Build the Game

1.Click **Build**
2.Select destination folder
3.Unity generates:
- .exe file
- Data folder (resources)

Step 7: Run Executable

1.Open build folder
2.Double-click .exe file
3.Run the game

Step 8: Debugging

1.Check for:
- Errors in Console
- Missing assets
- Script issues

2.Fix and rebuild if needed

Step 9: Playability Testing

Check:

- Player movement
- Game speed
- Obstacle spawning
- Score system
- UI responsiveness

Step 10: Packaging

- Ensure all files (.exe + Data folder) are kept together
- Compress into ZIP if sharing

Output

- Successfully generated Windows executable (.exe)
- Game runs outside Unity Editor
- All assets packaged correctly
- Smooth gameplay verified

Result

Successfully built and deployed the Infinite Runner game as a Windows executable and tested for debugging and playability.

Applications

- Game deployment
- Software distribution
- Prototype testing
- Commercial game release

VIVA QUESTIONS

S.No	Question	CO	Bloom's Taxonomy
1	What is the build process in Unity?	CO4	Remember
2	What is meant by an executable (.exe) file?	CO4	Remember
3	What is Build Settings in Unity?	CO4	Understand
4	What platforms are supported by Unity for building applications?	CO4	Understand
5	Why is it important to add scenes in Build Settings?	CO4	Analyze
6	What is the role of Player Settings?	CO4	Understand
7	What details are configured in Player Settings?	CO4	Remember
8	What is packaging of resources?	CO4	Understand
9	How does Unity include assets during build?	CO4	Analyze
10	What files are generated after building a project?	CO4	Remember
11	What is debugging in game development?	CO4	Remember
12	How do you identify errors in Unity?	CO4	Understand
13	What is the role of Console in debugging?	CO4	Understand
14	What are common errors during build?	CO4	Analyze
15	What is playability testing?	CO4	Remember
16	Why is playability testing important?	CO4	Understand
17	What aspects are checked during gameplay testing?	CO4	Analyze
18	How do you test game performance?	CO4	Apply
19	How can you distribute a built game?	CO4	Apply
20	How can you improve game performance after testing?	CO4	Evaluate

Experiment - 9

Define AR & VR with real-time examples. Introduction to tools like Oculus, Vuforia, Kudan, Wikitude, ARKit, and ARCore.

Aim

To understand the concepts of Augmented Reality (AR) and Virtual Reality (VR), along with real-time examples and an introduction to popular AR/VR development tools.

Objectives

- Define AR and VR concepts
- Differentiate between AR and VR
- Understand real-time applications
- Learn about major AR/VR tools and platforms

Theory

1. Augmented Reality (AR)

Augmented Reality is a technology that overlays digital content (images, text, 3D models) onto the real-world environment in real time.

Key Features

- Combines real and virtual world
- Interactive in real time
- Uses camera and sensors

Real-Time Examples

- **Pokémon GO** – Displays virtual Pokémon in real surroundings
- Snapchat filters – Adds effects to faces
- Google Lens – Recognizes objects using camera

2. Virtual Reality (VR)

Virtual Reality is a technology that creates a completely immersive digital environment, replacing the real world.

Key Features

- Fully immersive experience
- Requires VR headset
- Simulates real or imaginary environments

Real-Time Examples

- **Beat Saber** – Interactive music-based VR game
- Flight simulators – Training pilots
- Virtual tours of museums

3. Difference Between AR and VR

Feature	AR	VR
Environment	Real + Virtual	Fully Virtual
Immersion	Partial	Full
Device	Mobile, Tablet	VR Headset
Example	Pokémon GO	Beat Saber

4. AR/VR Development Tools

1. Oculus

- VR platform and headset
- Used for immersive gaming and applications
- Example: Oculus Quest

2. Vuforia

- AR SDK for Unity
- Supports image recognition and tracking
- Widely used in mobile AR apps

3. Kudan

- High-performance AR engine
- Works offline (no internet required)
- Used in industrial applications

4. Wikitude

- Supports image, object, and location-based AR
- Used in education and marketing

5. ARKit

- Developed by Apple
- Used for iOS devices
- Supports motion tracking and face tracking

6. ARCore

- Developed by Google
- Used for Android devices
- Supports environmental understanding

Applications of AR & VR

- Gaming

- Education
- Healthcare
- Military training
- Architecture and design

Result

Successfully understood the concepts of AR and VR, their real-time applications, and various development tools used in AR/VR systems.

VIVA QUESTIONS

S.No	Question	CO	Bloom's Taxonomy
1	What is Augmented Reality (AR)?	CO4	Remember
2	What is Virtual Reality (VR)?	CO4	Remember
3	What are the key features of AR?	CO4	Understand
4	What are the key features of VR?	CO4	Understand
5	Give real-time examples of AR.	CO4	Apply
6	Give real-time examples of VR.	CO4	Apply
7	What is the difference between AR and VR?	CO4	Analyze
8	What devices are used for AR and VR?	CO4	Understand
9	What is Oculus?	CO4	Remember
10	What is Vuforia used for?	CO4	Understand
11	What is Kudan SDK?	CO4	Remember
12	What is Wikitude platform?	CO4	Understand
13	What is ARKit?	CO4	Remember
14	What is ARCore?	CO4	Remember
15	Difference between ARKit and ARCore.	CO4	Analyze
16	What is an AR SDK?	CO4	Understand
17	What are the applications of AR?	CO4	Understand
18	What are the applications of VR?	CO4	Understand
19	How are AR/VR tools used in development?	CO4	Analyze
20	How can AR and VR technologies be improved in future?	CO4	Evaluate

Experiment - 10

Exploring VR Environment Using Oculus and Experiencing Basic Interactions & Movement

Aim

To explore a virtual reality environment using Oculus and understand basic interactions and movement within a VR space.

Objectives

- Understand VR environment setup using Oculus
- Experience immersive virtual environments
- Perform basic interactions in VR
- Implement movement in VR space

Software & Hardware Requirements

- Unity Hub
- Unity Editor (3D Core Template)
- Oculus VR Headset (Quest/Quest 2/Quest 3)
- Oculus Link / Air Link
- VR-enabled PC

Theory

1. Virtual Reality (VR) Environment

A VR environment is a fully immersive digital space where users interact using:

- Head movements
- Hand controllers
- Sensors

2. Oculus VR System

Oculus provides:

- Head-mounted display (HMD)
- Motion tracking
- Hand controllers
- Immersive experience

3. VR Interaction

Users interact using:

- Controller buttons
- Hand gestures
- Gaze-based interaction

4. VR Movement

Types of movement:

- Teleportation
- Continuous movement
- Room-scale movement

Procedure

Step 1: Setup Oculus Device

1.Connect Oculus headset to PC
2.Enable **Developer Mode**
3.Connect via:
o Oculus Link (USB) OR
o Air Link (Wireless)

Step 2: Create Unity Project

1.Open Unity Hub
2.Create **3D Project**
3.Name: *VRExperience*

Step 3: Install XR Plugin

1.Go to **Edit → Project Settings → XR Plugin Management**
2.Install and enable:
o Oculus Plugin

Step 4: Add XR Rig

1.Install XR Interaction Toolkit (Package Manager)
2.Add:
o XR Origin (Camera Rig)
3.This includes:
o Camera
o Controllers

Step 5: Create VR Environment

1.Add Plane (Ground)
2.Add Cubes / Objects
3.Add Lighting

Step 6: Enable Interaction

1.Add:
o XR Controller
o XR Ray Interactor
2.Enable grabbing and pointing

Step 7: Implement Movement

1.Add:
- Continuous Move Provider OR
- Teleportation Area

2.Assign input controls

Step 8: Build & Run

1.Connect Oculus
2.Click Play
3.Wear headset

Step 9: Experience VR

- Look around using head movement
- Move using controllers
- Interact with objects

Output

- VR environment displayed in headset
- User can move in virtual space
- Objects can be interacted with
- Real-time immersive experience

Result

Successfully explored a VR environment using Oculus and experienced basic interactions and movement.

Applications

- VR gaming
- Training simulations
- Virtual tours
- Medical applications

VIVA QUESTIONS

S.No	Question	CO	Bloom's Taxonomy
1	What is a Virtual Reality (VR) environment?	CO5	Remember
2	What are the features of VR?	CO5	Understand
3	What is immersive experience in VR?	CO5	Understand
4	What is Oculus?	CO5	Remember
5	What components are included in an Oculus system?	CO5	Understand
6	What is XR in Unity?	CO5	Remember
7	What is XR Plugin Management?	CO5	Understand
8	What is XR Interaction Toolkit?	CO5	Understand
9	What is XR Origin (Camera Rig)?	CO5	Understand
10	What are VR controllers used for?	CO5	Remember
11	What is interaction in VR?	CO5	Understand
12	What is a Ray Interactor?	CO5	Understand
13	What is teleportation movement in VR?	CO5	Remember
14	What is continuous movement in VR?	CO5	Understand
15	Difference between teleportation and continuous movement?	CO5	Analyze
16	What is room-scale movement?	CO5	Understand
17	How does head tracking work in VR?	CO5	Understand
18	How do you test a VR application in Unity?	CO5	Apply
19	What are the challenges in VR interaction?	CO5	Analyze
20	How can VR experience be improved?	CO5	Evaluate

Experiment - 11

Installation of Vuforia in Unity, License Key Generation, Database Creation, and Placing 3D Models on Image Targets

Aim

To install Vuforia in Unity, generate a license key, create and configure a database, and place 3D models on image targets for AR applications.

Objectives

- Install and enable Vuforia in Unity
- Generate Vuforia license key
- Create and configure image target database
- Import database into Unity
- Place 3D objects on image targets

Software Requirements

- Unity Hub
- Unity Editor (3D Project)
- Vuforia Engine (via Unity)
- Internet connection

Theory

1. Vuforia Engine

Vuforia is an AR SDK that enables:

- Image recognition
- Object tracking
- Marker-based AR

2. License Key

- Required to activate Vuforia
- Generated from Vuforia Developer Portal
- Linked to project

3. Database

- Stores image targets
- Used for tracking objects
- Can be:
  - Device Database
  - Cloud Database

4. Image Target

- A real-world image used as a marker
- Triggers display of virtual objects

Procedure

Step 1: Create Unity Project

1.Open Unity Hub
2.Create **3D Project**
3.Name: *VuforiaARApp*

Step 2: Enable Vuforia

1.Go to **Edit → Project Settings → XR Plugin Management**
2.Enable **Vuforia Engine**

Step 3: Generate License Key

1.Go to Vuforia Developer Portal
2.Create account and login
3.Navigate to License Manager
4.Click **Get Development Key**
5.Copy license key

Step 4: Configure AR Camera

1.Delete default Camera
2.Add **AR Camera**
3.In Inspector → Paste License Key

Step 5: Create Database

1.Go to Vuforia Target Manager
2.Create **Device Database**
3.Add Image Target:
o Upload image
o Set width
4.Download database (Unity package)

Step 6: Import Database

1.Import downloaded package into Unity
2.Activate database:
o AR Camera → Vuforia Configuration
o Enable database

Step 7: Add Image Target

1.GameObject → Vuforia Engine → Image Target
2.Select database and image
3.Adjust size

Step 8: Add 3D Model

1.Add Cube / 3D model
2.Place as child of Image Target
3.Adjust position and scale

Step 9: Test Application

1.Click Play
2.Show image target via webcam/mobile
3.Observe 3D model appearing

Output

- Image target detected successfully
- 3D model appears on marker
- Real-time AR interaction achieved

Result

Successfully installed Vuforia, generated license key, created database, and placed 3D objects on image targets.

Applications

- AR-based education
- Product visualization
- Marketing and advertising
- Interactive manuals

VIVA QUESTIONS

S.No	Question	CO	Bloom's Taxonomy
1	What is Vuforia?	CO5	Remember
2	What is the purpose of Vuforia in Unity?	CO5	Understand
3	What is an AR application?	CO5	Remember
4	What is a license key in Vuforia?	CO5	Remember
5	Why is a license key required?	CO5	Understand
6	How do you generate a Vuforia license key?	CO5	Apply
7	What is a database in Vuforia?	CO5	Remember
8	What are types of databases in Vuforia?	CO5	Remember
9	What is an image target?	CO5	Understand
10	How do you create an image target database?	CO5	Apply
11	What is the role of AR Camera?	CO5	Understand
12	How do you configure AR Camera in Unity?	CO5	Apply
13	How do you import a Vuforia database into Unity?	CO5	Apply
14	How do you activate a database in Unity?	CO5	Apply
15	How do you place a 3D model on an image target?	CO5	Apply
16	What happens when an image target is detected?	CO5	Understand
17	What are the limitations of image targets?	CO5	Analyze
18	How can you improve image tracking performance?	CO5	Evaluate
19	What are the applications of Vuforia?	CO5	Understand
20	How does Vuforia enable real-time AR interaction?	CO5	Analyze

Experiment - 12

Interacting with Augmented Objects in Real World and Building & Testing AR Application on Mobile Device

Aim

To interact with augmented objects in the real world and build and test an AR application on a mobile device or simulator.

Objectives

- Understand interaction with AR objects
- Implement real-time AR interaction
- Build AR application for mobile devices
- Test application on mobile or simulator

Software Requirements

- Unity Hub
- Unity Editor (3D Project)
- Vuforia or ARCore / ARKit
- Android Studio / Xcode
- Mobile device (Android/iOS)

Theory

1. Augmented Reality Interaction

AR interaction allows users to:

- View virtual objects in real world
- Move, rotate, and scale objects
- Interact using touch or gestures

2. AR Application Workflow

1.Detect real-world surface or image
2.Place virtual object
3.Enable interaction
4.Render in real-time

3. Mobile AR Deployment

- Build APK (Android) or IPA (iOS)
- Install on mobile device
- Test performance and interaction

Procedure

Step 1: Create AR Project

1.Open Unity Hub
2.Create 3D Project → *ARInteractionApp*

Step 2: Setup AR Framework

1.Enable:
- **Vuforia** OR
- **ARCore** / **ARKit**

2.Add AR Camera

Step 3: Add AR Object

1.Add Image Target or Plane Detection
2.Place 3D object (Cube/Model)
3.Adjust position

Step 4: Add Interaction Script

```
using UnityEngine;

public class ARInteraction : MonoBehaviour
{
    void Update()
    {
        if(Input.touchCount > 0)
        {
            Touch touch = Input.GetTouch(0);

            if(touch.phase == TouchPhase.Moved)
            {
                transform.Rotate(0, -touch.deltaPosition.x, 0);
            }
        }
    }
}
```

Attach to AR object.

Step 5: Configure Build Settings

1.File → Build Settings
2.Select:
- Android or iOS

3.Switch Platform

Step 6: Player Settings

- Enable:
  - Camera permission
  - AR support
- Set package name

Step 7: Build Application

1.Click **Build**
2.Generate APK (Android) / IPA (iOS)

Step 8: Deploy to Mobile

1.Connect mobile via USB
2.Enable Developer Mode
3.Install application

Step 9: Test Application

- Open app
- Point camera at target
- Interact with object using touch

Output

- AR object appears in real environment
- Object responds to touch interaction
- Application runs successfully on mobile

Result

Successfully implemented interaction with augmented objects and deployed AR application on a mobile device.

Applications

- AR gaming
- Education apps
- Product visualization
- Industrial training

VIVA QUESTIONS

S.No	Question	CO	Bloom's Taxonomy
1	What is Augmented Reality (AR) interaction?	CO5	Remember
2	What are augmented objects?	CO5	Understand
3	How do users interact with AR objects?	CO5	Understand
4	What is the role of touch input in AR?	CO5	Understand
5	What is Vuforia?	CO5	Remember
6	What is ARCore?	CO5	Remember
7	What is ARKit?	CO5	Remember
8	Difference between ARCore and ARKit.	CO5	Analyze
9	What is plane detection in AR?	CO5	Remember
10	How is a 3D object placed in AR?	CO5	Apply
11	What is mobile AR application?	CO5	Understand
12	What is an APK file?	CO5	Remember
13	What is the purpose of Build Settings in Unity?	CO5	Understand
14	How do you deploy an AR app to a mobile device?	CO5	Apply
15	What permissions are required for AR apps?	CO5	Understand
16	What is testing in AR development?	CO5	Remember
17	What is debugging in AR applications?	CO5	Understand
18	What are common issues during AR testing?	CO5	Analyze
19	How do you improve AR interaction performance?	CO5	Evaluate
20	What are real-world applications of mobile AR?	CO5	Understand