ERI1

- Prof. Dr. Kalpana Shankhwar
- This course aims to introduce extended reality (XR/AR, VR, MR) and its applications in Industry 4.0. It covers various topics that are integral parts of XR technology, for example, haptic technology, 3D modeling, tracking, metaverse and digital twin. The main focus of this course is to enable students to develop their own Android/iOS augmented reality (AR) applications. In addition, the students will also get familiar with using Microsoft HoloLens2 AR headset.

Table of Contents

Lecture 1: Introduction

Instructor Information

- **Instructor:** Dr. Kalpana Shankhwar
- **Department:** Human Centered Design, IIIT Delhi
- **Email:** kalpana@iiitd.ac.in
- **Office:** A-403 (R&D Block)
- **Office Hours:** Monday 1:30–3:30 pm
- **Class Timings:** Monday and Thursday 11:00–12:30 pm
- **Google Classroom:** https://classroom.google.com/c/NzkxMzY2MDQyNDgw?cjc=egnlvkyb
- **Class Code:** egnlvkyb

About the Instructor

- PhD in Mechanical Engineering (National Taiwan University)
- M.Tech in Mechanical Engineering (IIT Guwahati)
- B.E. in Mechanical Engineering (SGSITS Indore)
- Industry experience as Engineer at MAN Trucks India Pvt. Ltd.
- Assistant Professor at KIIT University, Bhubaneswar and currently at IIIT Delhi since July 2023.

Lab Focus: Emerging Technology Integrated Design & Manufacturing (ETIDM) Lab

- Research on **Emerging Technologies** like:
  - Extended Reality (XR): Includes Augmented Reality (AR), Virtual Reality (VR), Mixed Reality (MR)
  - Artificial Intelligence (AI)
  - Digital Twin
  - Internet of Things (IoT)
  - Haptic Technology
- Smart systems for **Industry 4.0** and **Education 4.0** contexts.
- Interdisciplinary applications: education, healthcare, manufacturing, teleoperation, safety guidance.

Grading Policy

Evaluation Type	Contribution to Final Grade
Class Exercise (participation)	10%
Assignments (4 total)	30%
Final Project (individual or group)	20%
Midterm Exam	15%
End Term Exam	25%

Extended Reality (XR)

- XR is an umbrella term for technologies that blend the physical and virtual worlds.
- Key components include VR, AR, and MR.

Virtual Reality (VR)

- Completely artificial environment experienced via computer-generated sensory stimuli (sight, sound).
- Provides *immersive experience* enabling navigation and interaction in 3D spaces.
- Users' actions influence the environment dynamically.

Augmented Reality (AR)

- Enhances reality by overlaying digital information onto the physical world.
- Usually accessed via smartphone cameras or smart glasses.
- Virtual objects appear integrated into the real-world scene.

Mixed Reality (MR)

- Combines AR and VR by allowing interaction between physical and virtual objects in real time.
- Virtual and real objects coexist and can be manipulated interactively.

History of Virtual Reality

- **1838:** Charles Wheatstone demonstrated depth perception using stereoscopic photos viewed through a stereoscope, creating a 3D illusion by presenting slightly different images to each eye.
- **1929:** Edward Link created the first commercial flight simulator ("Link Trainer")—an electromechanical device mimicking turbulence and aircraft motion.
- **1950s:** Morton Heilig developed the Sensorama, an arcade-style multi-sensory theatre.
- **1960:** Heilig invented the Telesphere Mask, the first head-mounted display (HMD) with stereoscopic 3D and stereo sound but without motion tracking.
- **1961:** Philco engineers created the Headsight, the first motion-tracking HMD linked to remote camera viewing—used mainly for military applications.
- **1968:** Ivan Sutherland and Bob Sproull developed the Sword of Damocles, the first computer-connected VR HMD, which was bulky and suspended from the ceiling.
- **1987:** Jaron Lanier coined the term "virtual reality" and developed early VR devices including data gloves and HMDs.
- **1997:** VR was used for PTSD treatment for war veterans by Georgia Tech and Emory University.
- **2007 onward:** Mass adoption begins, with Google Street View (360-degree imagery), Oculus Rift Kickstarter in 2012, and later commercial VR products (HTC Vive, Oculus Quest, Microsoft HoloLens).

Industry 4.0 and XR in Industry

- **Industry 4.0:** The current industrial revolution characterized by digital integration of manufacturing processes with intelligent automation, IoT, AI, and smart machines for efficient and flexible production.
- In Industry 4.0, smart devices, sensors, and machines are interconnected for autonomous communication and minimal human intervention.
- XR technologies play a key role in:
  - Smart training and tutorials
  - Real-time visualization of manufacturing processes
  - Remote operation and teleoperation of machinery
  - Assembly and maintenance tasks

- Human-robot collaboration
- Health and safety training
- Architecture and design simulation

XR Applications in Industry

- Assembly and maintenance process assistance
- Teleoperation of robotic systems
- Welding and manual milling training simulators
- Visualization of finite element analysis data
- Medical and surgical training with haptics
- Skill evaluation and performance tracking
- Architecture and customized design visualization
- Safety and emergency response training
- Interactive gaming and educational tools

Important XR Devices and Tools Timeline (Recent)

Year	Device/Technology	Description
2007	Google Street View	Street-level immersive 360-degree imagery
2010	Oculus VR prototype	Development of modern VR headset prototype
2012	Oculus Rift Kickstarter	Crowdfunded launch of a consumer VR headset
2014	Google Cardboard, Samsung Gear VR	Affordable mobile VR platforms powered by smartphones
2016	HTC Vive	PC-tethered VR with room-scale tracking
2019	Oculus Quest	Standalone wireless VR headset
Present	Vision Pro, HoloLens 2, Meta Quest 3	Advanced AR/VR/MR devices blending digital and real worlds

---

Summary: Industrial Revolutions

Wave	Period	Core Technology	Industrial Impact
1st	1750–1850	Mechanization, water & steam power	Mass production replaces manual labor
2nd	1870–1914	Assembly line, electricity, oil & gas	Increased production volume & communication tech
3rd	1970–2010	Computing, automation, telecommunications	Digitization of factories, programmable automation

Wave	Period	Core Technology	Industrial Impact
4th	2011–present	Cyber-physical systems, IoT, AI, XR	Smart factories, automation with minimal human input

Additional Notes on XR Technologies

- XR not only enhances immersion but also offers **interactive experiences** that improve learning effectiveness and operational safety.
- Haptic devices integrated with XR provide **tactile feedback**, crucial for applications like surgical training or remote machinery operation.
- Gesture-based interfaces and real-time 3D reconstruction improve the naturalness of human-machine interaction.
- GenAI (Generative AI) visualization combined with XR creates highly customized and dynamic virtual environments for architecture and design.

Lecture 2: Introduction to Extended Reality Hardware and Software

---

Overview

Extended Reality (XR) encompasses technologies that merge the real and virtual worlds to create immersive experiences. XR includes **Virtual Reality (VR)**, **Augmented Reality (AR)**, and **Mixed Reality (MR)**. These technologies play a crucial role in Industry 4.0 by enhancing manufacturing, training, design, and maintenance processes.

Course & Instructor Details

- Instructor: Dr. Kalpana Shankhwar, Assistant Professor, IIIT Delhi
- Class Timings: Monday & Thursday 11:00-12:30 pm
- Office Hours: Monday 1:30-3:30 pm
- Contact: kalpana@iiitd.ac.in
- Grading Structure:
  - Class Exercises: 10%
  - Assignments: 30%
  - Final Project: 20%
  - Mid Term Exam: 15%
  - End Term Exam: 25%

Weekly Topics and Focus Areas

Week	Topic	Activities
1	Historical overview, trends, and future applications of XR technologies	Introduction; types of game engines

Week	Topic	Activities
2	3D creation, design and modeling theory for XR	Engineering drawing, computer graphics basics
3	Basics of 3D modeling software and interface	3D modeling of objects
4	Game engine tools for XR, importing 3D models	In-class exercise
5	User interface elements in game engines	Assignment/In-class exercise
6	Tracking for AR, types of tracking methods	Vuforia Engine introduction
7	Building Android/iOS AR apps	Assignment/In-class exercise
8	Particle systems and 3D modeling in them	
9	Haptic technology and visuo-haptic XR	Assignment/In-class exercise

---

Key Concepts in XR

Types of XR

- **Extended Reality (XR):** Umbrella term for VR, AR, and MR technologies.
- **Virtual Reality (VR):** Fully immersive digital environments.
- **Augmented Reality (AR):** Overlaying virtual elements on the real world.
- **Mixed Reality (MR):** Blending real and virtual worlds interactively.

Haptic Technology

Haptics add the *sense of touch* to XR to make experiences more immersive.

Types of Haptic Feedback

- **Kinesthetic:** Senses force and motion involving muscles and joints. Simulates size and density of objects through physical sensations.
- **Tactile:** Relates to skin sensations such as vibration, surface texture, temperature, and pressure.

Devices and Application

- Kinesthetic devices use motors or actuators to simulate forces.
- Tactile devices use vibrations or temperature changes on the skin.

Examples of haptic feedback implementation include VR controllers that provide force feedback or mobile phone vibrations.

XR Development Platforms

- **Unity Engine:** Most accessible, supports 2D and 3D apps using C#. Extensive online resources available.

- **Unreal Engine:** Known for powerful rendering, performs well with fewer computing resources, uses C++.
- **Vuforia Studio:** Platform focused on AR development.

XR Hardware

Device Types

- **Mobile devices, tablets, laptops:** Common platforms for lightweight XR applications.
- **Head Mounted Displays (HMDs):** Key XR hardware with varying specificity for AR or VR.

Examples

- AR HMDs: Microsoft HoloLens 2, Google Glass, Epson.
- VR HMDs: Oculus Quest, HTC Vive, Valve Index.

Advantages of AR devices

- Immersive experiences
- Connection with the real world
- Hands-free operations

Disadvantages of AR devices

- Small field of view
- User discomfort for long use
- Limited battery life
- Higher cost

Tracking in XR

Tracking is essential to synchronize physical and virtual environments by detecting position and orientation of objects in real-time.

Tracking Types

- **Head Tracking:** Tracks the position/orientation of the HMD.
- **Hand Tracking:** Tracks hand movements and gestures.
- **Eye Tracking:** Detects gaze direction for interaction and foveated rendering.
- **Full Body Tracking:** Tracks the entire body’s movements.
- **Object Tracking:** Detects and tracks other physical objects.

Degrees of Freedom (DoF) in Tracking

Degrees of Freedom define the types of movements a device can track.

3DoF (Three Degrees of Freedom)

- Tracks rotational movements only:
  - **Roll:** Tilting head side to side.
  - **Pitch:** Nodding head up/down.
  - **Yaw:** Turning head left/right.
- Does **not** track translational movement (walking, sidestepping).

6DoF (Six Degrees of Freedom)

- Tracks rotational and translational movements:
  - Rotation: Roll, Pitch, Yaw.
  - Translation: Moving forward/backward (surge), left/right (strafe), up/down (elevate).
- Enables full 3D movement and immersive interaction in virtual spaces.

Emerging Technologies Related to XR & Industry 4.0

- Robotics
- Artificial Intelligence (AI)
- Drones
- Internet of Things (IoT)
- Large Language Models (LLM)
- Generative AI
- Haptics
- Brain Computer Interface (BCI)
- Autonomous vehicles (Self-driving cars)

Lecture 3: Introduction to 3D Creation

Dr. Kalpana Shankhwar, IIIT Delhi

Introduction

- A picture is worth a thousand words.
- Use of drawing language to convey ideas.
- Engineering drawing is a “standardized” drawing language.
- Important for product development:
  - Product specification
  - Product drawing
- Engineering drawing is an effective communication language between engineers.
- Geometric theorems are essential in engineering drawing.

What is 3D modeling?

- Constructing drawings on a computer screen using specialized software and hardware is called Computer Aided Drafting (CAD) or 3D modeling.
- CAD drawings are clearer and more precise than manual drawings.

- It creates interactive designs representing real-life objects.
- Allows quick creation and analysis of physical properties of parts and iterative updating of models.
- Widely used for internal and client presentations.
- Can be time and cost-efficient.

Visual Representation

- 3D modeling is visually represented in 2D using 3D rendering or visualization techniques.

3D Modeling Methods

- **Polygonal Modeling**: Represents points in 3D connected by line segments forming polygon meshes.
- **Curve Modeling**:
  - Uses curves to generate surface geometry.
  - Curves can be parametric (based on geometric/functional relationships) or freeform.
  - Curves influenced by mathematical equations and designer's control points.
- **Digital Sculpting**:
  - Newer method.
  - User interacts with the model like sculpting virtual clay.
  - Enables pushing, pulling, pinching, or twisting of the digital clay.

Importance of Computer Graphics

- CAD systems rely on Interactive Computer Graphics (ICG).
- Converts user commands into graphical representations.
- Supports creating, modifying, storing, and exploring drawings.
- Also essential in Computer-Aided Manufacturing (CAM) where graphics data is converted to machining data for CNC machines.
- Computer graphics hardware and software are primary constituents.
- 3D modeling software is designed to be interactive, intuitive, and user-friendly.
- Software interface has two main parts:
  - Graphics window: visual feedback of object design.
  - Command window: for entering commands.

Advantages of CAD

- **Accuracy:** Higher accuracy than manual drawing.
- **Speed:** Faster drawing creation; features like copy, mirror, array, automatic hatching, text, and dimensioning.
- **Easy Editing:** Drawings can be edited easily; components can be reused.
- **Standard Libraries:** Standard parts like gears, valves included.
- **Scaling:** Objects can be scaled up or down automatically.
- **Better Visualization**
- **Freedom from manual instruments**
- **Space Effectiveness**

Applications of 3D/CAD Modeling

- **Viewing:** Easily create and visualize 3D components and generate multiple views including sectional.
- **Mechanism:** Model components to simulate real kinematic assembly.
- **Finite Element Method (FEM):** Numerical method for solving physical problems using differential equations.
- **Manufacturing:**
  - Computer-aided manufacturing via CNC machines,
  - Robotic applications,
  - Automated measuring and inspection.
- **3D Printing:** Physical object creation via layer-wise material addition controlled by computer.
- **Entertainment:** Character creation in films and TV using 3D sculpting.
- **Automation:** Programming CAD for automatic component generation with variable parameters.
- **Fashion:** Virtual clothes design and dynamic visualization.
- **Medicine:** Design prosthetics and parts for organ repair.

Dimensioning

- Shows exact size (length, width, height, diameter) on drawings.
- Uses numerical values with lines, symbols, and notes.
- Most common measurement unit: millimeters (mm).
- Symbols for special features like diameter (Ø), radius (R), square (SQ), cylinder (CYL), etc.
- Dimension elements: dimension line, extension lines, arrowheads, leader, notes.
- Uses standard rules:
  - Align system,
  - Do not mix dimensioning system styles,
  - Follow preferred dimensioning styles for clarity.

Scales in Drawing

- Objects drawn to scale: enlarged or reduced proportionally.
- Scale represented by Representative Fraction (RF) or Scale Factor.
- RF = (Length on drawing) / (Actual length)
- Example: A 1.5 m long steel bar shown as 15 cm line on drawing gives RF = 1/10.

Thank you!

Lecture 4: Structure of 3D Modeling

Dr. Kalpana Shankhwar, IIIT Delhi

Theory of Projection

- Projection in engineering drawing means creating an image or view of an object.
- Engineers use various projection techniques to construct views of objects.

Projection System

- Light rays from a bulb striking an object create a shadow (image) on a screen.
- The image is larger due to divergent light rays.
- The observer replaces the light source; lines of sight form the view.
- The screen is called the Plane of Projection (POP).
- Lines of sight are called projection lines or projectors.
- Three basic elements: object, observer, and POP.

Classification of Projection Methods

- **Multi-view projection**
- **Orthographic projection**
- **Axonometric projection**
- **Parallel projection**
- **Oblique projection**
- **Convergent projection**
- **Perspective projection**

Parallel Projection

- Projectors are parallel.
- **Orthographic projection:** Projectors are perpendicular to the POP.
  - **Multiview projection:** Two or more views on different POPs showing two dimensions per view.

Axonometric Projection

- One view showing all three dimensions.
- Object oriented so three perpendicular edges are inclined to POP.
- Types:
  - **Isometric:** All three edges equally inclined to POP.
  - **Dimetric:** Two edges equally inclined.
  - **Trimetric:** All three edges different inclinations.

Oblique Projection

- Projectors are parallel but inclined at 30°, 45°, or 60° to POP.
- One face is parallel to POP called principal face.
- Types:
  - **Cavalier Projection**
  - **Cabinet Projection**

Convergent Projection

- Projectors converge to a station point (observer's eye).

- Parallel edges appear to converge at vanishing points.
- Object appears smaller with distance.
- Most common is perspective projection.

Perspective Projection

- Linear perspective: Size relates proportionally to distance.
- One, two, or three vanishing points used for different views.
- Aerial perspective considers atmospheric effects: contrast and color fade with distance to create depth illusion.

Perspective Viewing Types

- **Three-point perspective:** Three vanishing points on three principal directions.
- **Two-point perspective:** Two vanishing points, one direction parallel to POP.
- **One-point perspective:** One vanishing point, two directions parallel to POP.

Principal Planes

- POP where views are projected.
- In multiview, different POPs needed for different views.
- Three mutually perpendicular planes called principal/reference planes (RP):
  - Horizontal Plane (HP): parallel to ground.
  - Vertical Plane (VP): perpendicular to ground and HP.
  - Profile Plane (PP): perpendicular to HP and VP.
- Reference planes are imaginary and transparent.

Structure of 3D Modeling

- 3D models consist of entities and features like datum, solids, surfaces, etc.
- Operations on these features are accessed through CAD software commands via menus and icons.

Unit Settings in CAD

- CAD has predefined unit sets; users must select units before modeling.
- Unit systems include:
  - SI (International System)
  - MKS (meter-kilogram-second)
  - IPS (inch-pound-second)
  - FPS (foot-pound-second)
  - CGS (centimeter-gram-second)
- SI base units: meter, kilogram, ampere, kelvin, mole, candela.

Sketch Entities, Objects, and Classification

- CAD models include sketches, solids, annotations, datums, parameters.
- Sketches generate solids or surface models.
- Annotations display dimensions and process info.
- Objects facilitate drawing generation, analysis, manufacturing planning.

Datum Entities

- Have no physical mass but assist in model creation and analysis.
- Include planes, axes, and points.
- Planes are required to create sketches on flat surfaces.

Plane

- Imaginary/real flat surface where a straight line lies.
- Used as sketch planes or reference for measurement.
- Default CAD planes: Front, Top, Right.

Axis

- Imaginary infinite straight lines used as sketch/reference geometry or rotational axes.

Point

- Created in sketches or 3D space.
- Used for creating geometry or aiding measurement.

Sketch Entities in CAD

- Basic geometry tools include:
  - Point
  - Line
  - Arc or Circle
  - Rectangle
  - Parallelogram
  - Polygon
  - Ellipse
  - Spline

3D Curves

- Curves whose points do not lie on a single plane.
- Created by equations, special CAD tools, or tracing mechanisms.
- Used for creating objects or defining paths.

Surfaces

- Imagined as the "skin" of solids.
- Used in shaping objects like vehicle bodies, airplane wings.
- Mathematical representation allowing multi-angle viewing.

Solids

- Created via primitives or by extruding sections along paths.
- Used to visualize shape, analyze models, and produce drawings.

Regional Operations (Boolean Operations)

- Used to manipulate regions to create complex shapes.
- Types:
  - Union (addition)
  - Difference (subtraction)
  - Intersection
- Volumes after operations follow certain volume formulas involving overlaps.

Wireframes

- Show basic structure with lines and transparency.
- Wireframes represent object boundaries with points and connections, not solid.

Common 3D Modeling Software

- AutoCAD
- Pro-E (Pro/Engineer)
- Catia
- SolidWorks
- Blender
- Maya
- SketchUp
- 3DS Max
- Zbrush
- ArchiCAD
- Fusion 360

File Types in CAD

- Contain 3D/2D models including geometry, manufacturing, and material data.
- **Native formats:** Software-specific files (*.CATPart for CATIA).
- **Neutral formats:** Interoperable formats like STL, FBX, 3DS, OBJ, STEP.
  - STL: Popular in 3D printing, rapid prototyping.

  - FBX: Used in film/video games; geometry plus texture and color.
  - 3DS: Used in engineering, architecture.
  - OBJ: Polygonal models for 3D printing.
  - STEP: Widely accepted cross-industry format for interoperability.

Low Poly and High Poly Models

- Polygon count affects smoothness, accuracy, and file size.
- **High Poly Models:**
  - Photorealistic and detailed.
  - Complex geometry with many polygons.
  - Suitable for close inspection but costly to render.
- **Low Poly Models:**
  - Faster processing and smaller files.
  - Suitable for real-time manipulation (games, VR/AR).
  - Lower cost but less visual detail.
  - Ideal for applications prioritizing speed and interactivity.

Lecture 5: Structure of 3D Modeling

- [ERI Lecture 5](ERI/ERI, Lecture 5.pptx_compressed.pdf)
- Study .pdf too

Introduction to 3D Modeling in Computer Graphics

Problem Statement: Real objects (e.g., a teapot) exhibit curved surfaces, hidden lines, complex shading, perspective distortions, textures, and varied materials (pottery, ceramic, copper, etc.). Achieving realism requires solving these challenges via research in computer graphics.

Physical vs. Synthetic Images

- **Physical Images:** Captured by cameras or digitizers; objects exist in the real world.
- **Synthetic Images:** Generated entirely in software by defining 2D/3D primitives (points, lines, polygons) and applying rasterization, shading, and lighting models.

Core Entities: Object and Viewer

- **Object:** Defined in 3D space with geometric primitives independent of any viewer.
- **Viewer:** Human, camera, or digitizer that "sees" the object and defines the image formation process.

Imaging Models

- **Ray Tracing:** Simulates light-ray interactions—reflection, refraction, absorption—with scene objects to produce highly realistic images.
- **Basic Ray–Object Interactions:**
  - Ray enters camera directly.
  - Ray reflects off a mirror.
  - Ray refracts through a transparent sphere.

- Ray misses all objects (“goes to infinity”).

Multiview Orthographic Projection

- **Definition:** Projectors are parallel and perpendicular to the projection plane (POP). Multiple views are drawn on planes at right angles.
- **Principal Planes:**
  - VP (Vertical Plane) → Front View
  - HP (Horizontal Plane) → Top View
  - PP (Profile Plane) → Side View
- **First-Angle vs. Third-Angle Projection:**
  - *First-Angle:* Object lies between observer and plane; FV above XY, TV below.
  - *Third-Angle:* Plane lies between observer and object; FV below XY, TV above.

Principal Orthographic Views

1. **Front View (FV):** Observer looks at the front face; drawn on VP.
2. **Top View (TV):** Observer looks from above; drawn on HP.
3. **Side Views (SV):**
   - Left-Hand Side View (LHSV) or Right-Hand Side View (RHSV) on PP.
4. **Bottom View (BV):** Observer looks from below.
5. **Rear View (RV):** Observer looks from the back.

Rules for Projection of Faces and Edges

1. Face ⊥ viewing direction → True shape and size visible.
2. Face || viewing direction → Edge (line view).
3. Face inclined → Foreshortened view (true shape not visible).
4. Edge ⊥ viewing direction → True length visible.
5. Edge || viewing direction → Point view.
6. Edge inclined → Foreshortened length between projected endpoints.

Hidden Features and Sectional Views

- **Hidden Features:** Internal/external geometry not visible in a given view; shown with dashed lines.
- **Limitations:** Excessive dashed lines clutter drawings.
- **Sectional Views:** Cut planes (vertical, horizontal, profile, auxiliary) slice the object to reveal internal features.
  - **Vertical Section:** Plane perpendicular to HP cutting front/back.
  - **Horizontal Section:** Plane parallel to HP cutting top/bottom.
  - **Profile Section:** Plane parallel to PP cutting side features.
  - **Auxiliary Section:** Any nonprincipal plane used to reveal inclined faces.
- **Hatching:** Sectioned surfaces are filled with 45° hatch lines to denote cut material.

Example Exercise & Solution

- [ERI Lecture 5](ERI/ERI, Lecture 5.pptx_compressed.pdf)
- Provided an exercise with a composite object having vertical, horizontal, and profile cut planes.
- Sectional front, top, and side views illustrate internal geometry, dimensions, and relationships.

Key Takeaway: Understanding projection methods, viewing rules, and sectional techniques is essential for accurately representing an object’s three-dimensional depth and internal features in engineering and design documentation.

Lecture 6: User Interface Elements in Unity

This resource provides a detailed overview of User Interface (UI) elements within the Unity development environment, specifically tailored for Extended Reality (XR) applications in Industry 4.0. It begins by explaining the Canvas as the fundamental container for UI elements and differentiates between three render modes for positioning these elements. The document then describes basic layout components like Rect Transform and anchors, crucial for responsive UI design. Finally, it outlines and demonstrates various visual and interaction components such as Panels, Text, Images, Buttons, Sliders, Toggles, Scrollbars, Dropdowns, and Input Fields, offering step-by-step class activities for practical application of each.

Content Overview

The lecture covers the following key areas:

- **Canvas**
- **Basic Layout**
- **Visual Components**
- **Interaction Components**
- **Demo (Class Activities)**

Introduction to Unity UI

The **Unity user interface** is the most commonly used toolkit for creating UIs in various applications and games. At the core of Unity's UI system is the **Canvas**.

Canvas

A **Canvas** is a fundamental **rectangular area** that acts as a container for all UI elements. When you create a UI element, such as an **Image**, via the menu path `GameObject > UI > Image`, a Canvas is automatically generated in your scene if one doesn't already exist.

Render Modes for Canvas

The Canvas has three primary render modes that dictate how UI elements are displayed relative to the game world:

1. **Screen Space – Overlay**
    - **Description:** This mode places UI elements directly **on top of the scene**, meaning they are always visible and rendered last, "overlaying" everything else in the game world.
    - **Behavior on Resize:** If the screen's size or resolution changes, the Canvas will **automatically adjust its size** to match these changes, ensuring UI elements maintain their relative positions and scale on the screen.

2. **Screen Space - Camera**

   - **Description:** Similar to "Screen Space - Overlay," but with a crucial difference: the Canvas is positioned at a **specified distance in front of a designated Camera**.
   - **Rendering Impact:** The UI elements are rendered by this specific camera, which means that the **camera's settings (e.g., field of view, culling mask)** directly influence the appearance of the UI.
   - **Depth Interaction:** A key feature here is that **other game objects in the scene can be rendered in front of the UI**. This allows for more dynamic interactions where 3D objects can occlude or interact with UI elements.

3. **World Space**

   - **Description:** In this mode, the Canvas behaves like **any other 3D object within your game scene**. It exists in the 3D world, rather than being a flat overlay on the screen.
   - **Manual Sizing:** The **size of the Canvas can be set manually** using its **Rect Transform** component.
   - **3D Placement:** UI elements within a World Space Canvas will render **in front of or behind other objects in the scene** based on their actual 3D placement.
   - **Use Case:** This mode is particularly useful for UIs that are intended to be an **integral part of the game world**, such as interactive displays on a computer screen within a game, health bars above characters, or signs in a virtual environment.

Basic Layout Components

Unity's UI system uses specialized components for layout and positioning.

Rect Transform

- **Core Component:** The **Rect Transform** is a specialized transform component used exclusively for **all UI elements**, replacing the standard `Transform` component.
- **Function:** It provides properties for controlling the position, size, and anchoring of UI elements within a Canvas, making it suitable for 2D layout.

Anchors

- **Layout Concept:** Anchors are a layout concept integrated into Rect Transforms.
- **Visual Representation:** They appear as **four small triangular handles** in the Scene View.
- **Parent-Child Relationship:** If a UI element's parent is also a Rect Transform, the **child Rect Transform can be anchored to its parent** in various ways.
- **Examples:** A child element can be anchored to the **center of its parent**, or to **one of its corners**, ensuring it maintains its relative position even if the parent's size changes. This is crucial for creating responsive UI layouts that adapt to different screen sizes and aspect ratios.

Visual Components

These components are primarily for displaying information or decorative elements without direct user interaction (though they can be part of interactive elements).

- **Panel**
- **Text**
- **Image**

Interaction Components

Interaction components are the building blocks for user input. They are designed to **handle various interaction events** such as mouse clicks, touch events, or input from keyboards or controllers.

- **Button**
  - **Function:** Buttons control the **response to a user's click** (or touch) and are typically used to **initiate or confirm an action**. For example, a "Start" button to begin a game or a "Confirm" button in a dialog box.
- **Slider**
  - **Function:** A slider allows the user to **vary a decimal number value** within a predefined **minimum and maximum range** by dragging its handle.
  - **Event:** It has an `OnValueChanged UnityEvent`, which allows you to define actions that happen when the slider's value changes. This is useful for things like volume control or adjusting an object's size.
- **Toggle**
  - **Function:** A toggle control acts like a **checkbox**, allowing the user to **switch an option on or off**.
  - **Event:** Similar to the Slider, it has an `OnValueChanged UnityEvent` to define actions when its state (on/off) changes.
- **Scrollbar**
  - **Function:** The scrollbar control enables users to **scroll through content (like an image or text) that is too large to fit completely** within its display area.
- **Dropdown**
  - **Function:** A dropdown presents a **list of options** from which the user can **choose a single one**. When clicked, it expands to show the list, and selecting an item updates the displayed choice.
- **Input Field**
  - **Function:** An input field allows the user to **enter text**, which can then be used to initiate or complete an action.
  - **Event:** It features a `UnityEvent` that defines what happens once the user has finished editing the text (e.g., pressing Enter or clicking away).

Class Activities (Demos) - Practical Application for Beginners

The lecture includes several class activities to demonstrate the practical application of these UI elements.

Class Activity 5: UI Element: Button (Scene Switching)

This example demonstrates how to use buttons to switch between different scenes in Unity.

Objective: Create multiple scenes and use buttons to navigate between them. **Scenes Involved:** Sample Scene, Scene 1, Scene 2, Scene 3. **Buttons Used:** Start, Next, End.

Steps for Beginners:

1. **Create a Panel:** Start by creating a `Panel` (a visual container) and set its color to **red with a transparent background**. Panels are good for organizing UI elements.
2. **Create a Button:** Make a `Button` as a child of this red panel.
   - Set its **Height to 50**.
   - Set its **scale to 2**.
3. **Name the Button:** Change the text displayed on the button to **"Start"** using the text component of the button.
4. **Create a Scene Manager Object:** Create an **empty GameObject** in your scene and name it **"SceneManager"**. This object will hold the script responsible for scene loading.
5. **Create a C# Script:** Create a new C# script and name it **"LoadScene"**. This script will contain the logic to load different scenes.
6. **Attach Script:** Drag and drop the "LoadScene" script onto the "SceneManager" GameObject in the Inspector.
7. **Add On Click Event to Button:** Select your "Start" button. In its Inspector, find the `Button` component and locate the **"On Click ()" event list**. Click the **"+" symbol** to add a new event.
8. **Link Scene Manager:** Drag the **"SceneManager" GameObject** from your Hierarchy into the empty object slot (Runtime Only) that appeared in the "On Click ()" event list.
9. **Select Scene Loading Function:** From the dropdown menu next to the "SceneManager" object, find the **"LoadScene" (the script you created)** and select the function within it that handles scene loading (e.g., `SceneLoader`, which you would implement in your script).
10. **Set Scene Index:** In the numerical input field that appears, **enter "1"**. This corresponds to the index of "Scene 1" in your build settings.
11. **Create "Scene 1":** Create a new Unity scene and save it as "Scene 1".
12. **Repeat for "Next" Button:** Duplicate the panel and button setup, but for "Scene 1," make the **panel color green**, name the button **"Next,"** and set its scene index to **"2"** (for "Scene 2").
13. **Repeat for "End" Button:** Similarly, create "Scene 2" with a **blue panel**, name the button **"End,"** and set its scene index to **"3"** (for "Scene 3").
14. **Create "Scene 3":** Create a final blank scene and name it "Scene 3".
15. **Add Scenes to Build Settings:** Go to `File > Build Settings`. Drag all your created scenes ("Sample Scene," "Scene 1," "Scene 2," "Scene 3") into the **"Scenes in Build" space**. This assigns them numerical indices (0, 1, 2, 3) that you used for the buttons.
16. **Test:** Hit the **Play button** and click your "Start," "Next," and "End" buttons to verify scene switching.

Class Activity 6: UI Element: Slider (Object Scaling)

This example shows how to use a slider to dynamically change the size of a 3D object.

Objective: Use a `Slider` to vary the size of a GameObject.

Steps for Beginners:

1. **Create a Scene:** Start by creating a new Unity Scene and name it **"SliderExample"**.
2. **Add a Panel:** Add a `Panel` to your scene. Set its color to **transparent black**. This acts as a background for your UI.
3. **Add a Slider:** Underneath the panel (as a child), add a `Slider`.
   - Set the slider's fill color to **yellow**.
   - Set the handle color to **green**.
4. **Import 3D Object:** Import a 3D model, such as **"HelloKitty"**, into your project's `Assets` section. Then, drag this "HelloKitty" GameObject onto your Unity scene. This is the object whose scale will be controlled.
5. **Create a Script:** Create a new C# script and name it **"ScaleObject"**. This script will contain the logic to adjust the object's scale.
6. **Attach Script:** Drag the "ScaleObject" script to the Inspector of the **"HelloKitty" GameObject**.
7. **Link Slider Event:** Select your Slider. In its Inspector, find the `Slider` component and locate the **"OnValueChanged (Single)" event**.
   - Drag the **"HelloKitty" GameObject** (the one with your "ScaleObject" script attached) to the empty object slot in the "OnValueChanged" event list.
   - From the dropdown menu, select **`ScaleObject.AdjustScale`** (you would have implemented this function in your "ScaleObject" script to take a float value and set the object's scale).
8. **Set Slider Range:** Set the **Min Value of the Slider to 5** and the **Max Value to 15**. This means the "HelloKitty" object's scale will vary between 5 and 15 units.
9. **Test:** Hit the **Play button** and drag the slider from left to right. Observe how the size of the "HelloKitty" GameObject increases as you drag.

Class Activity 7: UI Element: Toggle (Enable/Disable Object)

This example demonstrates how to use a toggle to enable or disable a GameObject.

Objective: Use a `Toggle` control to enable and disable a 3D object.

Steps for Beginners:

1. **Create a Toggle:** Create a `Toggle` UI element from the UI menu, making it a child object of a `Panel` (or directly under the Canvas).
2. **Configure Toggle Text:** To ensure good quality for the toggle's label text, adjust its `Text` settings:
   - Set **Width = 300**.
   - Set **Height = 150**.
   - Check the **"Best Fit" option**. These settings ensure the text is clear and readable.
3. **Create 3D Object:** Create a **"Cube"** from the 3D object menu (GameObject > 3D Object > Cube).
   - Attach a **material with a red color** to the Cube. This makes it easily visible.
4. **Create a Script:** Create a new C# script named **"ToggleExample"**. This script will contain the logic to enable/disable the cube.
5. **Attach Script:** Drag the "ToggleExample" script onto the **"Cube" GameObject** in the Hierarchy.
6. **Link Toggle Event:** Select your `Toggle` UI element. In its Inspector, find the `Toggle` component and locate the **"On Value Changed (Boolean)" event**.

   - Drag the **"Cube" GameObject** (the one with the "ToggleExample" script) to the empty object slot in the event list.
   - From the dropdown menu, select **`ToggleExample.Toggle_Changed`** (you would implement this function in your script to take a boolean value and set the cube's active state).
7. **Test:** Hit the **Play button**. Check (turn on) and uncheck (turn off) the toggle and observe how the "Cube" GameObject is enabled and disabled in the scene.

Class Activity 8: UI Element: Scrollbar (Text Scrolling)

This example illustrates how to use a scrollbar to scroll through a large block of text.

Objective: Apply a `Scrollbar` to scroll through text that exceeds the display area.

Steps for Beginners:

1. **Create and Resize Panel:** Create a `Panel` UI element and **resize it to cover the entire Game view**.
   - Add a **"Mask" component** to this panel from the `Add Component` menu. The Mask component will hide any child elements that extend beyond the panel's boundaries, creating the scrollable effect.
2. **Create TextContainer (Panel with Scroll Rect):** Create another `Panel` (or an empty GameObject with a Rect Transform) and name it **"TextContainer"**. This will hold the scrollable text and the scrollbar.
   - **Match the borders of "TextContainer" with the borders of the outer "Panel"**.
   - Add a **"Scroll Rect" component** to "TextContainer". This component manages the scrolling behavior.
3. **Create Scrollbar:** Create a `Scrollbar` UI element from the UI menu, making it a child of **"TextContainer"**.
4. **Create Text Element:** Create a `Text` UI element from the UI menu, also making it a child of **"TextContainer"**.
5. **Add Text Content:** Add any **random text with at least 12 lines** into the `Text` UI component.
   - Match its borders with the "Panel" (the `TextContainer` in this case).
   - **Drag the `Text` element downwards** so that a portion of it extends beyond the "TextContainer's" visible area. This ensures there's content to scroll through.
6. **Set Scrollbar Direction:** Select the `Scrollbar` and set its **direction to "Bottom to Top"**. This means scrolling up will move the text upwards, revealing content that was previously at the bottom.
7. **Set Canvas UI Scale Mode:** Select the main `Canvas` in your scene. In its Inspector, set the **UI Scale Mode to "Scale with Screen Size"**. This helps your UI adapt to different screen resolutions.
8. **Link Scroll Rect Content:** Select the **"TextContainer"** GameObject (the one with the "Scroll Rect" component).
   - Drag the **"Text" UI element** into the **"Content" section** of the "Scroll Rect" component. This tells the Scroll Rect what content it needs to scroll.
9. **Link Vertical Scrollbar:** Drag the **"Scrollbar" UI element** into the **"Vertical Scrollbar" section** of the "Scroll Rect" component. This connects the visual scrollbar to the scrolling behavior of the content.
10. **Test:** Hit the Play button and try dragging the scrollbar handle to scroll through the long text content.

Class Activity 9: UI Element: Dropdown (Option Selection)

This example demonstrates the application of the `Dropdown` UI element.

Objective: Use a `Dropdown` to allow users to select from a list of options and display the choice.

Steps for Beginners:

1. **Create Dropdown:** Create a **"Dropdown – TextMeshPro"** UI element from the UI menu. (TextMeshPro is an advanced text solution for Unity, often preferred for better text rendering).
2. **Resize Dropdown:** Resize the dropdown element to a **width of 350** and a **height of 70**.
3. **Customize Options:** In the Inspector of the `Dropdown` component, find the **"Options" section**.
   - Change the default text for option A, B, and C to **"Your favorite color," "Your favorite band," and "Your favorite city"**, respectively. You can add more options if needed.
4. **Write C# Script:** Write a new C# script (e.g., named "DropdownScript") that will handle the dropdown's value change and update a display text.
5. **Create Empty GameObject:** Create an **empty GameObject** in your scene and attach the "DropdownScript" to it. This GameObject will act as the controller for the dropdown.
6. **Create Display Text:** Create a **"Text – TextMeshPro"** UI element from the UI menu. Make it a child of the `GameObject` you created in the previous step. This text element will display the chosen option.
7. **Link Text to Script:** Select your "GameObject" (the one with the script). In its Inspector, you'll need a public variable in your "DropdownScript" (e.g., `public TextMeshProUGUI outputText;`). Drag the **"Text (TMP)" UI element** you just created to the corresponding **"output section"** of your script in the Inspector.
8. **Link Dropdown Event:** Select your `Dropdown` UI element. In its Inspector, find the `Dropdown` component and locate the **"On Value Changed (Int32)" option**.
   - Drag the **"GameObject" (your script controller)** to the empty object slot in the event list.
   - From the dropdown menu, select **`DropdownScript.HandleInputData`** (you would implement this function in your script to take the integer index of the selected option and update the display text).
9. **Test:** Hit the **Play button**. Select different options from the dropdown to verify that the corresponding text is displayed by your "Text (TMP)" element.

Class Activity 10: UI Element: InputField (Text Entry Display)

This example demonstrates the application of the `InputField` UI element.

Objective: Use an `InputField` to allow the user to enter text and then display that text on the Canvas.

Steps for Beginners:

1. **Create Input Field:** Create an `Input Field` UI element from the UI menu.
   - Change its **background color** for better visibility.
2. **Set Canvas Scale Mode:** Select your main `Canvas` in the scene. In its Inspector, set the **Canvas scalar settings to "Scale with screen size"**. This helps the Input Field and other UI elements adapt to different screen resolutions.
3. **Write a Script:** Write a C# script (e.g., named "InputFieldDisplay") that will handle reading text from the InputField and displaying it on another Text element.
4. **Attach Script to Main Camera:** Attach this "InputFieldDisplay" script to the **"Main Camera"** GameObject in your scene. (While it can be attached to any GameObject, the Main Camera is often a

convenient place for global UI scripts).

5. **Create Display Text:** Create a `Text` UI element under the Canvas and name it **"InputText"**. This will be where the entered text is displayed.
   - Set its **Width to 1000** and **Height to 150**.
   - Keep the **font size at 24**.
   - Check the **"Best Fit" option**. These settings provide a large, clear area for displaying text.
6. **Link Display Text to Script:** Select the "Main Camera" (where your script is attached). In its Inspector, you'll need a public variable in your script (e.g., `public TextMeshProUGUI newText;` or `public Text newText;` depending on your setup). Drag the **"InputText" UI element** to the corresponding **"NewText" option** of your script in the Inspector.
7. **Link Input Field Event:** Select your `Input Field` UI element. In its Inspector, find the `Input Field` component and locate the **"On End Edit (String)" event**. This event triggers when the user finishes editing the text (e.g., by pressing Enter or clicking outside the field).
   - Drag the **"Main Camera" GameObject** (the one with your script) to the empty object slot in the event list.
   - From the dropdown menu, select **`InputFieldDisplay.ReadStringInput`** (you would implement this function in your script to take the entered string and update the "InputText" element).
8. **Test:** Hit the **Play button**. Type any text into the `Input Field` and then press Enter (or click away). Observe that the entered text is now displayed on your "InputText" UI element on the Canvas.

Lecture 7: Augmented Reality with Vuforia: Tracking and Targets

- see slides

This academic presentation material, authored by Dr. Kalpana Shankhwar, outlines the principles and applications of Augmented Reality (AR) using the Vuforia Engine within the context of Extended Reality in Industry 4.0 (ERI). The core of the source explains Vuforia Engine as an Software Development Kit (SDK) for building AR applications, detailing various AR tracking methods, such as sensor-based and vision-based techniques. A significant portion of the text is dedicated to describing the different types of trackable targets supported by Vuforia, including Image Target, Multi Target, Object Target, Model Target, Cylinder Target, Ground Plane, and Area Target. Furthermore, the material provides practical, step-by-step instructions for installing and configuring Vuforia within the Unity game engine to create a basic AR application.

---

1. Introduction to Extended Reality and Tracking

Extended Reality in Industry 4.0 (ERI) covers AR applications, tools, and technologies.

1.1 AR Tracking Methods

Several methods are used for tracking in Augmented Reality:

- Sensor based tracking
- Vision based tracking
- Optical tracking
- Acoustic tracking

- Magnetic tracking
- Inertial tracking
- Marker based tracking
- Marker less tracking

1.2 Vuforia Engine

The **Vuforia Engine** is a software development toolkit (SDK) designed for building augmented reality (AR) applications.

It achieves AR by offering a variety of trackable targets. The engine recognizes these targets to bring virtual objects into the real world. This recognition enables the user to interact with the virtual objects in AR.

1.3 Supported Devices

Vuforia supports AR applications on the following devices:

- iPhone
- iPad
- Android
- Universal Windows Platform (UWP) devices
- Eyewear

---

2. Vuforia Targets and Tracking Methods

Vuforia offers various targets for tracking, including: Image target, Cylinder target, Multi target, Model target, Object target (3D scanned), VuMark, Area target, Ground plane, and Cloud recognition.

2.1 Image Target

An Image target represents images that the Vuforia engine can detect and track.

- **Detection Process:** The Vuforia engine detects and tracks the physical image captured by the camera. It does this by comparing the natural features of the camera image against a target resource database.
- **Augmentation:** Once the image target is detected, the engine tracks the physical image and augments virtual content over the real image.
- **Formats:** Image targets can be used directly from the image assets of the Vuforia engine package. Alternatively, they can be created with Vuforia Target Manager using JPG or PNG images in RGB or grayscale.

2.1.1 Attributes for an Ideal Image Target

An ideal image target should exhibit the following attributes:

- It should be **rich in detail**.
- It should have **good contrast**.
- It should contain **no repetitive patterns**.
- It should have an **organic shape with round and soft details** (Fig. 3e).

- It can be a **Non-Rectangular image target** (Fig. 3f).
- It should display **feature distribution** (Fig. 3g).

2.1.2 Physical Properties of Image Targets

The physical properties of the printed target are important for tracking:

- **Print Material:** Flexible pieces of paper should be avoided to keep the object in focus. A hard material, such as card stock, plastic, or paper fixed to a non-flexible surface, should be used to ensure the marker's flatness.
- **Size:** The printed image target should be at least **12 centimetres in width** with a reasonable height. The target's size varies depending on the distance from the camera. The size of the target should ideally be **1/10 of the distance** between the target and the camera.
- **Surface Finish:** Printed image targets should **not be glossy**. Glossy reflection, which can occur under certain light angles, may cover a large portion of the original texture of the printed image.

2.2 Multi Target

Multi targets are combinations of image targets arranged in a **geometrical arrangement**, such as boxes or cuboids.

- **Tracking Advantage:** This arrangement allows the object to be detected and tracked from all sides. Due to the three-dimensional arrangement, all faces of a multi target can be tracked simultaneously.

2.3 Object Tracking Methods

Object tracking allows the camera to detect physical objects based on their geometry and then overlay virtual content onto the real object. There are three types:

2.3.1 Object Target

- **Creation:** Object targets are created by scanning the physical object from all directions. During scanning, the physical object is placed on a printed image target.
- **Usage:** The generated object data file is uploaded onto the Vuforia Target Manager and is then used for tracking the real object.

2.3.2 Model Target

- **Requirements:** This method requires both a **CAD model** of an object and its physical counterpart.
- **Creation:** The CAD model is imported into the Model Target Generator to generate the model target.
- **Usage:** Vuforia uses the model target to detect the real object and augment virtual content onto it.

2.3.3 Model Target from 3D Scan

- **Purpose:** This is used when the CAD model of the object is not available.
- **Creation:** A mesh can be created from a 3D scan. Model targets are generated from the 3D scan of the real object using hardware-aided scanning, such as an **occipital structure sensor**.

2.4 Cylinder Targets

Cylinder targets enable the camera to detect and track image targets that are wrapped into a cylindrical or conical shape.

- **Suitable Objects:** Examples of suitable objects include soda cans, coffee cups, drum containers, and beverage bottles.
- **Tracking Scope:** Vuforia can track the sides, flat top, and bottom of the cylinder target.

2.5 Ground Plane

The Vuforia ground plane feature enables digital content to be placed on horizontal surfaces in the environment, such as floors and tables.

- **Capabilities:** It supports the detection and tracking of these surfaces. It also allows for placing virtual content in mid-air using an **Anchor point**.

2.6 Area Targets

Area target is an environment tracking feature that enables tracking and augmenting areas and spaces.

- **Creation:** A 3D scanning device is required to scan the environment. Suitable devices include the **Matterport Pro2 3D camera** or ARKit devices with **LiDAR sensors**.
- **Usage:** Once the area targets are created, virtual content can be augmented onto the stationary objects within the scanned environment.

---

3. Vuforia Installation Steps (Unity)

The following steps detail the process for installing and configuring Vuforia within Unity:

A. Unity and SDK Installation

1. Install Unity version 2020.3.1f1 (or later 19.2) or the latest version of Unity. Ensure you check the **Android build support** option.
2. Go to the webpage link: `https://library.vuforia.com/articles/Solution/vuforia-engine-package-ho sting-for-unity.html`.
3. Find the option under `Assets > Import Package > Custom Package`.
4. Navigate to the location where the Vuforia SDK has been downloaded.
5. Select the SDK package and hit "open".
6. Click on “Import”.
7. Select “Update”.

B. AR Camera and License Setup

8. After importing Vuforia to Unity, locate the Vuforia option in the “GameObject” menu.
9. Delete the “Main Camera” from the scene.
10. Add the **“AR Camera”** from the Vuforia menu.
11. Click **“Open Vuforia engine configuration”** under the “AR camera” gameobject.
12. Click the “Add license” button.
13. To obtain the license key, register on the Vuforia engine developer portal.

14. Log in to your Vuforia account and click “Get Development Key”.
15. Create the development key using any License name, uncheck the box for terms and conditions, and then click confirm.
16. Copy the License key and paste it into Unity.

C. Adding a Target and Virtual Object

17. Add “Image target” –> `Marker_Vuforia` to the `Assets > Resources` under the Project.
18. Add the Image Target from `GameObject > Vuforia Engine > Image Target`.
19. Choose `”Marker_Vuforia”` or any other marker image and drag it to the Image.
20. Create a gameobject, such as a “Cube,” and make it a **child** of the “Image Target”.
21. Change the Transform values of the “Cube” to adjust the marker borders.
22. Use the printed marker or download the marker image on a cellphone.

D. Testing and Configuration

23. Hit the play button and hold the marker in front of your device’s camera to track the virtual gameobject (Cube).
24. To disable the “Extending tracking,” check the “Tracked” option from the “Image Target”.
25. For **multiple object tracking**, enter the required number of Image targets in the **“Max Simultaneous Tracked Images”** section of “Vuforia Configuration”.