AR CONFIGURATIONS

Submitted in partial fulfillment of the requirements
for the award of the degree of
Bachelor of Computer Applications
To
Guru Gobind Singh Indraprastha University, Delhi

Guide:
Dr.Nidhi Verma
(Assistant Professor)

Submitted by:
Kuvam Bhanot
06113702021

Institute of Information Technology & Management,
New Delhi – 110058

Batch (2020-2023)

Certificate

I Kuvam Bhanot 06113702021 certify that the Major Project Report (BCA-308) entitled “AR CONFIGURATIONS” is done by me and it is an authentic work carried out by me at IITM. The matter embodied in this project work has not been submitted earlier for the award of any degree or diploma to the best of my knowledge and belief.

Signature of the Student
Date:

Certified that the Major Project Report (BCA-308) entitled “AR CONFIGURATIONS” done by the above student is completed under my guidance.

Signature of Guide
Date:
Name of the Guide:
Dr.Nidhi Verma
Designation:
Assistant Professor

Counter sign HOD

Counter sign Director

Acknowledgement

I really grateful and wish my profound my indebtedness to Dr.Nidhi Verma, Assistant Professor of Institute of Information Technology and Management, Janakpuri, New Delhi, India.

Her endless patience, scholarly guidance, continual encouragement, constant and energetic supervision, constructive criticism, valuable advice, reading many inferior drafts and correcting them at all stage have made it possible to complete this project.

Finally, I must acknowledge with due respect the constant support and patience of my parents.

Abstract

Augmented reality (AR) is the overlaying of digitally-created content such as computer-generated graphics, videos, texts and sounds are layered on top of a physical view. Augmented reality permits the users to have interaction with real and virtual elements or augmentation. Whereas Virtual reality (VR) is computer generated stereo visuals which replace the actual world surroundings of a consumer around them. Both Augmented and Virtual reality may be supplied to consumers through headsets like HTC Vive, Oculus Rift, and Microsoft's HoloLens or through the camera of a mobile phone. In both practical and experimental implementations, augmented reality and virtual reality can replace or lessen the consumer's belief of truth.

Augmented and Virtual reality has been adopted in various industries such as retail, healthcare, science educational and real estate. Different configurations provides different perspectives to the user and access to the recent development and areas in AR and XR.

This project provides the basic four types of AR configuration apps; which allows the user to see and react to the AR environment and digital content in the real world.It allows the readers to study the basic and getting the user to control and view the system and with perspective of the building relationship between digital and physical content and world.

List of Figures/Tables

Figures:

Synopsis

- Title:

AR Configurations

- Problems with the Existing System:
  - AR PLANE DETECTION

Unity Framework for AR provides a suitable environment and structure for application but, unity AR plane detection requires a color(surface) template for plane detection and can indulge in issues with the surface and location structure.

  - AR FACE FILTER INTERACTIVE

Unity Framework for face filters for interactive does not includes or detects the face of the user and can be skeptical for the user and the developer providing an face filter in AR but not the face detection.

- Description of the Proposed System:

  - AR PLANE DETECTION

In this project for the plane detection and working for the structure and environment compatibility for the user, better method is proposed known as “FEATURE POINTS” which forms a whole structure called “POINT CLOUDS” which detects the points/events in the user environment.

  - AR FACE FILTER INTERCATIVE

In this project for the face filter interactive, the face is detected and based on that the model/accessories will be placed and when the face will not be in sight/not detected by the user camera, the model will not be placed, ensuring that the user and the developer enjoys the AR with the model.

- Description and identification of the Functional Modules:

Modules:Windows PC, Unity, Unity AR SDK, Google AR Core, OpenJDK, Android SDK and NDK(Native Development Kit)

Target User Environment: Android OS, Camera(CMOS)

- Tools/Platforms:

Engine: UNITY BETA
SDK: OpenXR, Google AR Core, Apple ARKIT, Unity AR SDK
Template(CORE): AR CORE, VR CORE

Hardware:
MEMORY: 16 GB, 512 GB SSD
MICROPROCESSOR: AMD RYZEN 5 5600H
GPU: NVIDIA GTX 1650 TU117 Chip, AMD RYZEN VEGA
NETWORK: LAN(90 Mbps)
ARCHITECTURE: CPU - ZEN 2, GPU - Turing, RDNA

Software:
OS: Windows 11 - x86_64
BROWSER: Yandex

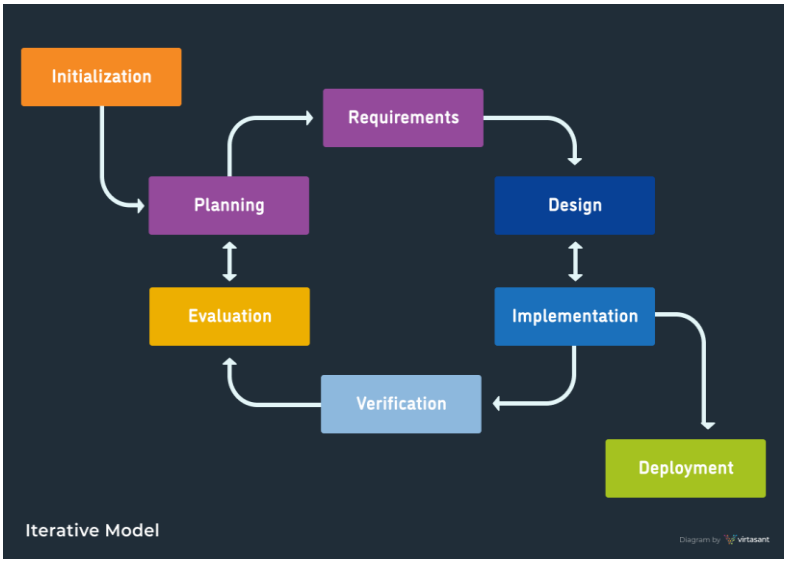
➢ Methodology:

Augmented reality (AR) works by combining real-world and computer-generated data, such as images, text, or 3D models, to create an enhanced or augmented view of the user's surroundings.

1. Capturing the real-world environment:
✓ AR systems use cameras or sensors to capture the user's view of the real-world environment.
✓ This can be achieved through cameras on smart phones, tablets, AR glasses, or other dedicated AR devices.
2. Analyzing the captured data:
✓ The captured data (images or video frames) is analyzed to identify specific features, objects, or surfaces.
✓ Computer vision techniques, such as object recognition, feature detection, and tracking, are employed to understand the real-world environment.
✓ This analysis provides information about the position, orientation, and scale of the identified objects or surfaces.
3. Rendering virtual content:
✓ Based on the analysis of the real-world environment, virtual content (3D models, text, images, animations, etc.) is rendered and overlaid on top of the real-world view.
✓ The virtual content is positioned, scaled, and oriented accurately to align with the real-world objects or surfaces.
4. Aligning virtual and real-world content:
✓ To create a seamless and realistic augmented experience, the virtual content needs to be aligned and registered with the real-world environment.
✓ This is typically achieved through techniques like marker tracking (using predefined visual markers) or marker less tracking (using natural features in the environment).
✓ Advanced techniques like simultaneous localization and mapping (SLAM) and depth sensing are also used for accurate alignment.
5. Displaying the augmented view:
✓ The augmented view, which combines the real-world environment and the virtual content, is displayed to the user through the AR device's screen or display.
✓ For AR glasses or head-mounted displays, the augmented view is projected directly into the user's field of view.
6. Interaction and feedback:
✓ AR systems often include interaction mechanisms, such as gesture recognition, voice commands, or input devices, to allow users to interact with the virtual content.
✓ User interactions and feedback can trigger updates to the virtual content or modify the augmented experience in real-time.

➢ SDLC Model:

ITERATIVE MODEL



➢ Justification for the Selection of Model:

- Flexibility
- Customer(User) Involvement
- Risk Management
- Complexity

➢ Future Scope:

This project provides a basic but interactive platform to explore and start development in AR, with the different AR configurations discussed below and allows the user to interact and model the usage of the AR environment for study or as a side project with further access to unity asset store and VR platform and XR while with or without the requirement of a VR Console and with the advantage of MR development and structure.

CONTENT

CHAPTER 1

INTRODUCTION

1.1 BACKGROUND/HISTORY

The first ever Augmented reality app was created in 1968 by the father of computer graphics “Ivan Sutherland” [1]. It turned into an AR head-installed display gadget. From then on AR was superior as wearables and digital shows. It could format virtual factors in the actual-international photo, as instance, show geological information approximately about a selected area. In the year 2008, Augmented reality opened a way of using augmented reality for industrial purpose. A Garman agency developed an augmented reality application for the BMW Mini. To enable it a customer had to point
a camera at a printed ad and the model of car auto came alive at the display. The customer could manipulate the car on the screen by controlling it and moving it around to
view it at different angles, simply by manipulating the printed image.
In the early 2010s Augmented reality introduced that might have interaction with actions in the real world in real time. For that reason, the virtual try on technology has all started by augmented reality’s instant face recognition revolutionized buying experience. Now you can try out anything form virtual making up your face to place a couch inside your home.

In 2016, augmented reality (AR) catapulted into the mainstream with the release of Pokemon Go [2], an AR game that grew to 45 million daily active users after just two months on the market. Since then AR has been adopted in various industries.
Let’s take a look at what augmented reality is, the benefits of augmented reality. Augmented reality [3] can be described in somewhere between physical world and virtual world where computer generated images, texts, sounds, animations, videos are layered on the top of the physical view. Users can interact with these elements via application on smartphones, tablets, augmented reality headsets like Microsoft HoloLens [4], and augmented reality glasses like Vuzix Blade [5].

1.1.1 Benefits of Augmented Reality:

One of the significant benefits of augmented reality is it can visualize information in 3D rather than 2D. As an Example, for an architecture planning, drawings and renderings are usually presented in a flat and 2D format, However, with Augmented reality those viewing plans can be shown as 3D format where walking around the model, look at every angle of the future building can build its own opinions about its design.
Another important benefit of augmented reality is it can not only make complex data easier to understand but also create an interactive experience that is intuitive, meaningful and also memorable to the users.
In addition, augmented reality can create personalized product experience, generating buzz, establishing repeat engagements of customers. Think about giving trials of

cloths without even putting them on or furnishing interior with furniture's without moving or buying them. Also, we can look into Pepsi bus shelter advertisement [6], for using augmented reality in real life.

1.2 MOTIVATION

The project which includes the different configurations through which the AR is deployed in the 3D environment and interacts with the user in real-world without interfering in the user actual world. AR provides the benefit of interaction and deployment of the 3D models in the existing and actual world and enhancing the relationship between "Digital" and "Physical" worlds.

The learning and exploring the AR or in the whole terms of XR, for the first time provided me with the knowledge and skills to understand how the apps and games which deploys the XR works and functions with the user and developer interaction including various SDK and build programs.

1.3 OBJECTIVES

Deployment of various AR configurations based App, discussed below with each app and its brief working and prerequisites necessary to understand purpose and inner working of the platform.

- ✧ AR MARKER(Object) Detection App

Marker-based AR makes it possible for your app to detect an image in the real world and trigger some action, such as displaying AR content, when the image is detected.

An **AR marker** is an image that your app will search for in the real world. When the app detects the marker, it will project a 3D model on top of the marker in AR. The marker works somewhat like a QR code, except that instead of taking you to a website, it can trigger an entire AR experience.

By the end of this learning project you will be able to do the following:

- ◆ By the end of this learning project you will be able to do the following:
- ◆ Implement image marker tracking in AR to spawn a 3D model.
- ◆ Implement screen space and world space UI in AR apps.
- ◆ Implement spatial audio in AR.
- ◆ Set up your project and select the 3D model that will appear when the marker is detected.
- ◆ Add a series of UI buttons to allow your user to rotate the object, display text, play audio, or to do some other action unique to your chosen model.

✧ AR PLANE Detection App

Create an AR app that makes use of plane detection technology. iOS and Android devices have the ability to detect flat surfaces — or planes — in your physical environment. This includes horizontal planes like the floor or a table, and vertical planes like the walls.

You will create an app where the user will be able to tap on a plane in order to place a portal onto it. This portal will be a window into a virtual 3D environment that the user can peer into and explore.

By the end of this learning project you will be able to do the following:

- Set up plane AR detection in an app
- Detect a user's touch
- Use AR ray casting to determine whether a user touched a tracked AR object
- Instantiate an object in the scene
- Test the frame rate of your app

✧ AR FACE FILTER(Base) App

Create your own custom face filter app that includes a wide variety of digital content. The face filter will include interesting textures, animated 3D models, dynamic particles, and post-processing effects.

By the end of this learning project you will be able to do the following:

- Set up facial tracking in your project.
- Use Unity's wide range of creative tools to overlay digital content on the user's face, including textures, animated 3D models, particles, and post-processing effects.

✧ AR FACE FILTER(Interactive) App

Create your own interactive app that allows a user to select, customize, and visualize different pairs of glasses on their face.

By the end of this learning project you will be able to do the following:

- By the end of this learning project you will be able to do the following:
- Add a user interface to an app.

- Use Visual Scripting to program an app's functionality.

1.4 ROLE OF AR IN COMPUTER GRAPHICS

Augmented Reality (AR) is quickly becoming one of the most transformative technologies in the graphic design industry. AR allows designers to take their designs to a whole new level by bringing them to life in real-time and offering a level of interactivity that was previously unimaginable. In this article, we'll explore the impact of AR in graphic design and how it's changing the landscape of visual communication.

The first thing that AR brings to the table is the ability to enhance the design process. Designers can now use AR to create and test designs in real-life scenarios, allowing them to see how their designs would look in the real world and make adjustments accordingly. This level of realism provides designers with a new level of insight and helps them create designs that are both functional and visually appealing.

AR also provides a new level of interactivity, allowing users to interact with designs in ways that were previously not possible. For example, AR can be used to create immersive experiences that allow users to explore designs in 3D and gain a deeper understanding of the designs. This level of interaction is crucial in fields such as product design and architectural design, where the ability to see a design in its intended environment can be invaluable.

Another advantage of using AR in graphic design is improved user experience. By allowing users to interact with designs in real-time, AR can provide a more engaging and immersive experience. This can lead to increased engagement and can help build a stronger connection between the user and the design. AR also offers new opportunities for collaboration, as designers can use AR to gather feedback from clients and stakeholders in real-time.

https://medium.com/@amirsly1372/the-impact-of-augmented-reality-in-graphic-design-the-future-of-visual-communication-f46f371c0f7d

Access to the Links and development areas in the GitHub:
https://github.com/KuvamInnovate/Test.git

CHAPTER 2

LITERATURE REVIEW

2.1 MODELS/ENGINES

AR involves the use of 3Dimensional models which are represented in the real world by the user interaction and further provides the operations and access to the custom model development.

What is a 3D model?

A 3D model is a digital representation of a physical object that can be inserted into the real world through AR technology.

There are several properties which are necessary while building or using a 3D model in AR project:

1. Low Poly-count
2. Correct Scaling
3. Optimized Textures
4. Efficient File Formats
5. 3D Tracking(Camera Focus)

AR or in general XR requires a engine to build apps and development and deployment from the developer to the user with necessary libraries and support for the effective and user-developer based interaction.

What is an Engine?

A game engine or design engine is a software framework designed for creating video games or interactive applications with real-time computer graphics. It provides a robust set of tools, libraries, and functionalities that developers can leverage to build games or other multimedia experiences more efficiently.

There are several core components in a engine:

- ✓ Rendering Engine: Responsible for rendering 2D or 3D graphics, handling shaders, textures, lighting, and visual effects.
- ✓ Physics Engine: Simulates physical properties such as collisions, rigid body dynamics, and realistic motion.
- ✓ Audio Engine: Handles sound effects, music, and audio playback.
- ✓ Input Handling: Manages user input from various devices like keyboards, gamepads, or motion sensors.
- ✓ Scripting System: Allows developers to write code or scripts to define game logic, rules, and behaviors.

- ✓ Scene Management: Manages the organization and loading of game levels, environments, and assets.
- ✓ Animation System: Facilitates the creation and playback of character animations and cut scenes.
- ✓ User Interface (UI) System: Provides tools for creating menus, heads-up displays (HUDs), and other on-screen elements.
- ✓ Asset Pipeline: Enables the importing, processing, and optimization of various game assets like models, textures, and audio files.
- ✓ Networking: Supports multiplayer functionality and communication between clients and servers.

Some popular game engines include Unity, Unreal Engine, CryEngine, Godot, and game-specific engines like Source (Valve), Frostbite (EA), and Infinity (Epic Games).

For this project Unity is used for the AR deployment using the Unity AR SDK and Google AR Core for the phone connection.

2.2 AUGMENTED REALITY(AR)

Augmented Reality (AR)

AR blends the digital world/content(images, models, videos, characters, etc) in the real world(environment) in real-time.It overlays or displays the digital content in the users environment using various sophisticated technologies and programs while utilizing the user system to overlay and providing minimal back-end showcase in the front-end.

What is Assisted Reality(aR)?How it differs from Augmented Reality(AR)?

Assisted Reality(aR) as the name suggests provides the advantage of operations and technologies to the user, aR utilizes some of the features of AR but usually consists of several other hardware components and generally does not overlays the content in the real-world but provides the advantage of using other components without interfering the whole system.

Assisted Reality(aR) can be termed as a subset of AR but only for few components, generally aR is not considered a part of XR; but because of relation between aR and AR for connecting real and digital world.

There are several “aR Glasses” which consists of SoC(System on chip) for the processing and DPU(Digital Processing Unit) for converting analog and digital signals.

Names of popular aR glasses:

- ➢ Xiaomi Mijia Smart Glasses
- ➢ Huawei Eyewear 2
- ➢ Oppo AIR Glasses
- ➢ Meta Rayban Smart Glasses

Xiaomi Mijia aR Glasses

Components in building AR(Augmented Reality) Hardware involves various systems to work together and ensure the deployment and accessibility of the AR app on the user system or hardware.AR Hardware involves various components such as:

1. Display: AR glasses feature a transparent or see-through display that overlays digital content onto the user's view of the real world. Common display technologies used include wave guides, projection-based systems, and optical combiners.
2. Cameras: Multiple cameras are used for various purposes, such as capturing the real-world environment, tracking the user's movements and position, and enabling features like gesture recognition and spatial mapping.
3. Sensors: AR glasses often include a variety of sensors, such as accelerometer, gyroscopes, magnetometers, and depth sensors, to track the user's head movements, orientation, and position accurately.
4. Processor and Graphics Processing Unit (GPU): A powerful processor and GPU are required to perform complex computations, render virtual content, and handle real-time tracking and mapping tasks.
5. Memory and Storage: AR glasses typically have on board memory (RAM) and storage (flash memory or solid-state drive) to store software, applications, and user data.
6. Connectivity: Wireless connectivity options like Wi-Fi, Bluetooth, and sometimes cellular connectivity (4G/5G) are included to enable data transfer, internet access, and communication with other devices.
7. Audio System: Speakers or audio output capabilities are often integrated into AR glasses to provide audio feedback, instructions, or environmental sounds to enhance the augmented experience.
8. Batteries: AR glasses require batteries or rechargeable power sources to operate, as they are typically self-contained and untethered devices.
9. User Input: AR glasses may include input methods like touch pads, voice recognition, gesture control, or eye-tracking systems to allow users to interact with the virtual content and user interface.

10. Cooling System: Due to the high computational demands and compact form factor, AR glasses may incorporate cooling solutions like heat sinks or fans to dissipate heat generated by the internal components.

Popular AR Glasses names:

- Microsoft HoloLens(Mixed Reality)
- Apple Vision Pro(AR Headset)
- XReal Air(AR Glasses)
- Huwaei Vision Glass
- TCL RayNeo
- Meta Quest 3(Mixed Reality)

XReal air 2 ultra

2.3 CONFIGURATION

✧ AR Marker Based App

What is AR Marker and how does AR Marker app works?

An AR marker is an image that your app will search for in the real world. When the app detects the marker, it will project a 3D model on top of the marker in AR. The marker works somewhat like a QR code, except that instead of taking you to a website, it can trigger an entire AR experience.

Marker-based AR makes it possible for your app to detect an image in the real world and trigger some action, such as displaying AR content, when the image is detected.

The whole process of creating the AR app is divided in various steps:

1. Create a new marker based AR App
2. Spawn an object on AR marker
3. Add UI with rotations

4. Add world space UI to the marker based App

✧ AR Plane Detection App

In the app, the user will be able to tap on a vertical plane (most likely a wall) or a horizontal plane (a table or the floor) in order to place a portal onto it. This portal will be a window into a virtual 3D environment that the user can peer into and explore.

What is a plane?Types of planes and working?

A plane refers to a flat surface or area in the real-world environment that is detected and tracked by the AR system. Planes are important in AR because they provide a reference point or anchor for placing and positioning virtual objects or content in the augmented scene.

✧ Horizontal Planes (Floor Planes): These are flat, horizontal surfaces detected by the AR system, such as floors, tables, or ground surfaces. Horizontal planes are commonly used for placing virtual objects, such as furniture, decorations, or interactive elements, in a way that respects the laws of gravity and appears grounded in the real-world environment.

✧ Vertical Planes (Wall Planes): These are flat, vertical surfaces detected by the AR system, such as walls, doors, or whiteboards. Vertical planes are often used for displaying virtual content like images, videos, or informational overlays, as well as for positioning virtual objects in a way that appears attached or anchored to the vertical surface.

The process of detecting and tracking planes in AR typically involves the following steps:

➢ Image Capture: The AR system uses the device's camera(s) to capture images or video frames of the real-world environment.
➢ Feature Detection: Computer vision algorithms analyze the captured images to identify and extract features, such as corners, edges, or distinct patterns, that can be used for plane detection.
➢ Plane Detection and Tracking: Using the extracted features, the AR system identifies and estimates the position, orientation, and dimensions of planar surfaces in the environment. This information is then used to track the planes as the user moves or changes the camera viewpoint.
➢ Surface Meshing: In some cases, the AR system may also generate a 3D mesh representation of the detected planes, which can be used for more accurate occlusion handling (virtual objects appearing behind real-world surfaces) and surface tracking.

The whole process of creating the AR app is divided in various steps:

1. Set up scene with plane detection
2. Detect the user’s touch
3. Detect when the user touches the plane
4. Instantiate the model on plane

5. Test the frame rate(FPS)

✧ AR Face Filter App

In this learning experience, you will learn to create your own AR face filter apps. The first app you'll make is a simple filter that applies materials and 3D models to the user's face.

The whole process of creating the AR app is divided in various steps:

1. Set up face-tracking project
2. Add a face texture
3. Add accessories to the filter
4. Add particle effects to the filter
5. Add post-processing in the filter

✧ AR Face Filter Interactive App

In the last configuration, you created a single face filter of your own design using various features, such as face textures, meshes, and animations. Although you added a variety of different features to your filter, it was still only one filter. Often, apps present their users with multiple filters to experiment with.

In this next configuration, you'll learn how to implement multiple filters that your users can select and customize through a user interface (UI). Along the way, you'll explore ways to create a user interface in Unity, deepen your knowledge of the AR face filters, and most importantly, build your first interactive features using visual scripting.

The whole process of creating the AR app is divided in various steps:

1. Build a scene for face filter.
2. Set up the UI
3. Turn on the glasses with button
4. Configure one color button
5. Save and load the glasses

2.4 EXTENDED REALITY(XR)

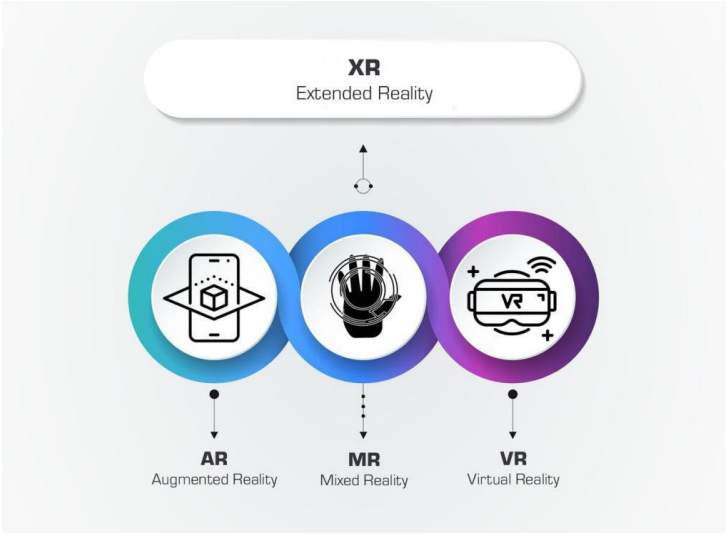
The main and the important of all is Extended Reality(XR) which comprises of all AR, MR, VR.

Extended Reality (XR) is an umbrella term that encompasses all real-and-virtual combined environments and human-machine interactions generated by computer technology. It includes Augmented Reality (AR), Virtual Reality (VR), and Mixed Reality (MR), covering the full spectrum from the completely real to the completely virtual. XR is used to describe immersive technologies that can merge the physical and digital worlds or create a fully
immersive experience.

Extended Reality(XR) and Human Computer Interaction(HCI):

XR technologies have the potential to create incredibly powerful and immersive experiences. But in order for these experiences to be truly effective, they need to be intuitive and easy for people to use. This is where HCI comes in. HCI principles can be used to design XR interfaces that are natural, engaging, and effective.

Brain-Computer Interfaces (BCIs) that could potentially be used to control XR experiences with thought alone. HCI research is essential for ensuring that BCIs are usable and accessible for a wide range of people.



CHAPTER 3

DESIGN AND IMPLEMENTATION

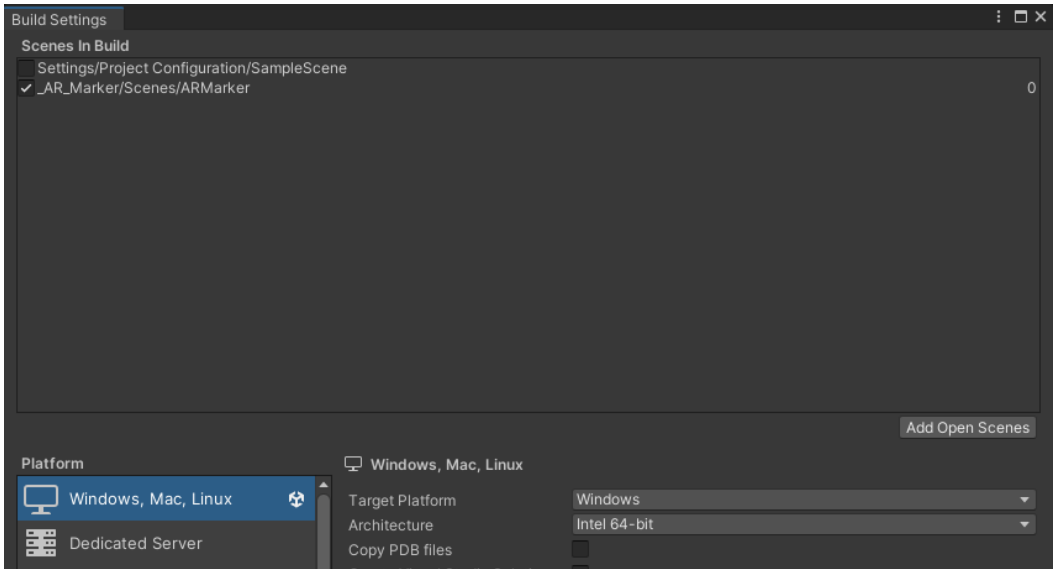
Prerequisites:

- Install Unity version 2021.3.1f1.
- Make sure your mobile device and computer are set up for AR development.
- Download and open a new Unity project configured for AR.
- Deploy a simple AR app to your mobile device that displays a spinning cube in your physical environment.

3.1 AR MARKER

3.1.1 Building the project:

✓ Go to File > New Scene, select the AR scene template from the list, and select Create.
✓ 2. Save the new scene as “ARMarker” in the Scenes folder.
✓ 3. Open the build settings (File > Build Settings).
✓ 4. Below the Scenes in Build section, select Add Open Scenes to add your new ARMarker scene to the Scenes in Build list.
✓ 5. Disable the SampleScene in that same list.

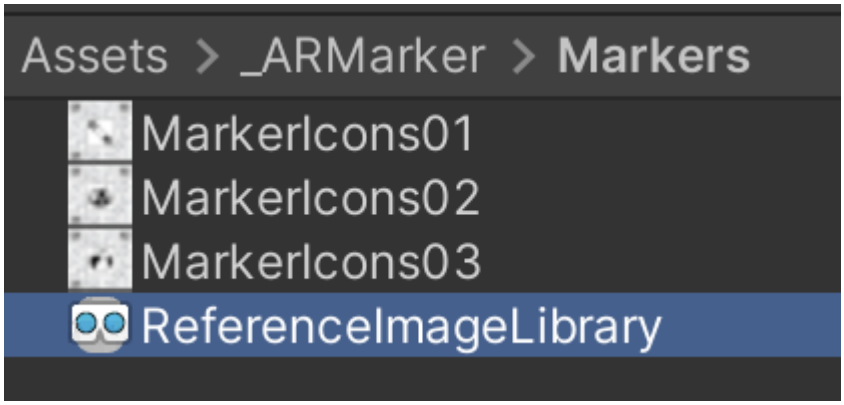


✓ Switch the build platform to iOS or Android, depending on your testing device.
✓ Build and test the app.

3.1.2 Spawn the object/model (XR Reference Image Library):

In the Project window, navigate to the Assets > _ARMarker > Markers folder. Choose which of the provided markers you would like to use for your application, or create and import your own.

2. In the Project window, in the Markers folder, right-click and select Create > XR > Reference Image Library.



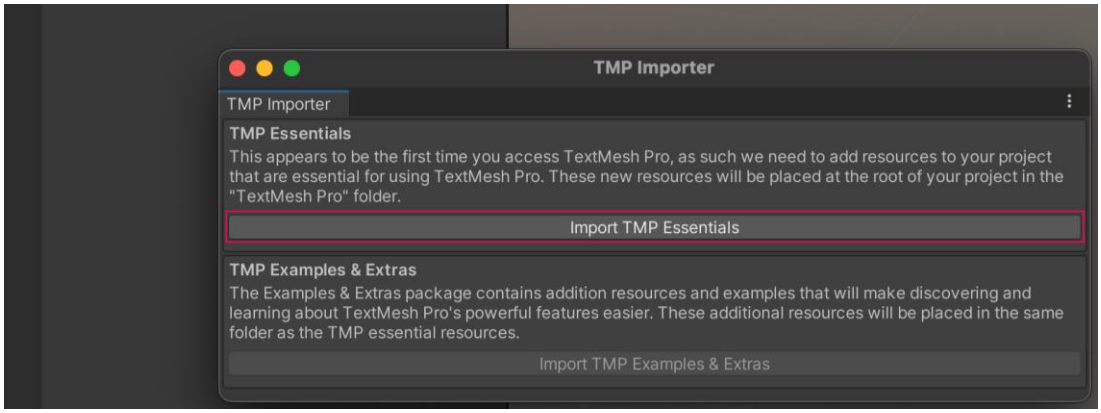
3. In the Markers folder, select the ReferenceImageLibrary asset you've created, and in the Inspector, select Add Image.

4. In the Inspector, use the Texture Selector in the Texture 2D slot, search for "Marker", and select the MarkerIcon image you want to use as the AR marker for your project.

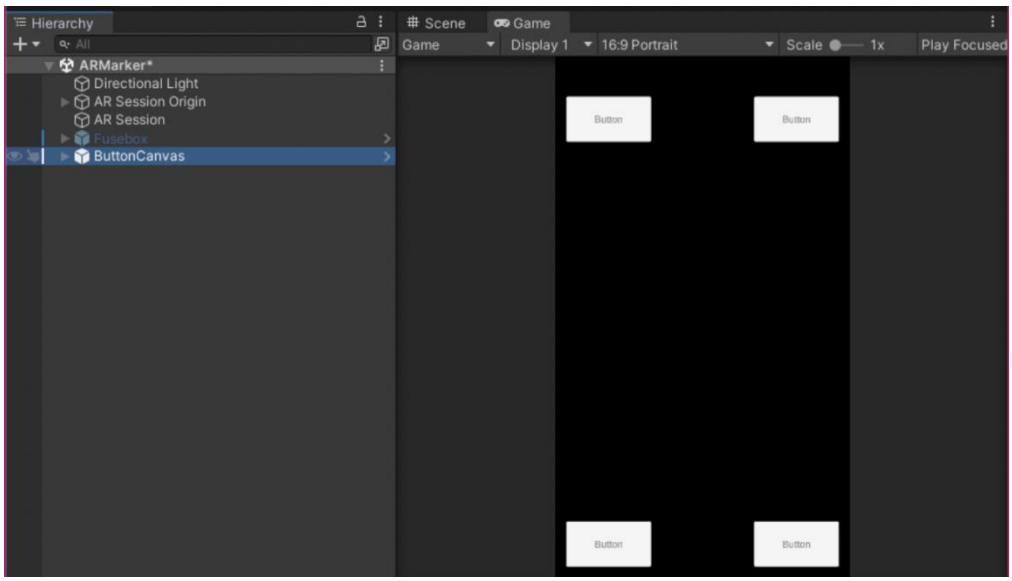
5. For iOS Only: You must specify the size (in meters) of the marker in the real world in order for your iOS project to build successfully. The marker may be recognized even if the size does not match what you specify, but the location of your anchor may be slightly off if there is a discrepancy in the size of the marker and the values you input.

3.1.3 Add UI with rotation buttons:

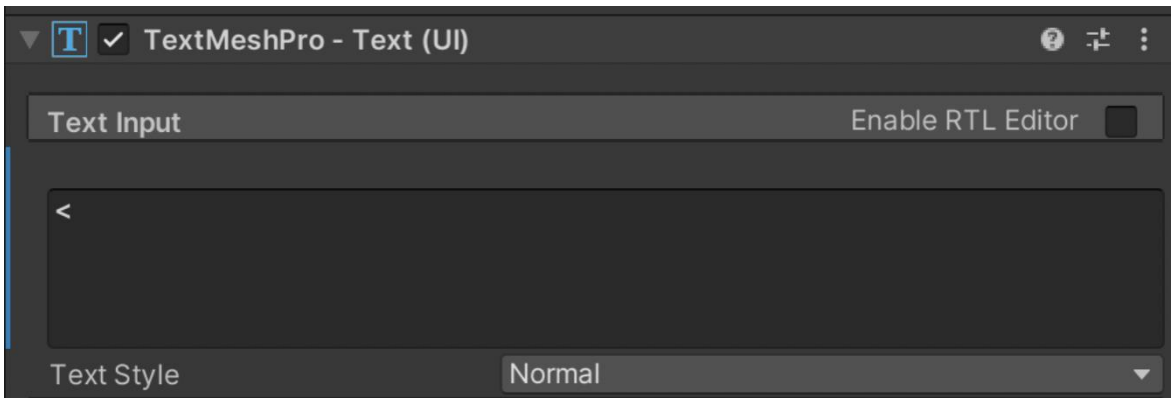
In the Project window, go to Assets > _ARMarker > Prefabs > UI, then click and drag Button_Canvas into the Hierarchy. In the Scene view, the canvas will be very large, but don't worry, it's just the way that Unity handles screen space overlays.



2. Go to the Game view to preview how the four default buttons will appear in your application.



3. In the Hierarchy, expand the ButtonCanvas GameObject and rename the Button1 and Button2 child GameObjects to “ButtonRotateLeft” and “ButtonRotateRight”, respectively.
4. In the Hierarchy, select the Text (TMP) child GameObject of the ButtonRotateLeft GameObject. In the Inspector, change TextMeshPro Text Input to “<”. Repeat for the ButtonRotate Right GameObject and change the Text Input to “>”.



5. Customize the text by selecting the Font Asset, Font Size, and Font Style properties you think look best.
Note: If you want to get additional fonts and styles, go to Window > TextMeshPro > Import TMP Examples and Extras.

6.If you want to adjust the size of the buttons, use the Rect tool.

3.1.4 Visual Scripting

1. In the Hierarchy, create a new empty GameObject named “VisualScripts”.

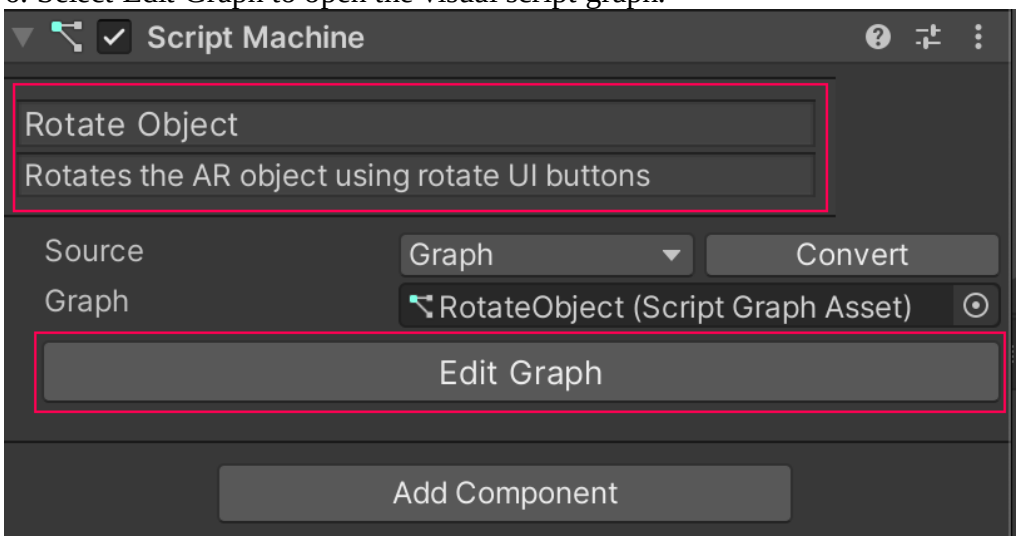
2. In the Inspector, add a Script Machine component to VisualScripts.

3. Select New to make a new graph.

4. When the File Explorer appears, open the _ARMarker folder, create a new “VisualScripts” folder there, and save the graph with the name “RotateObject”.

5. In the Inspector, enter the title “Rotate Object” and the summary “Rotates the AR
object using rotate UI buttons”.

6. Select Edit Graph to open the visual script graph.



Rotation:

In the Graph Editor, add an On Button Click node to the Graph Editor.

Drag the buttonRotateLeft variable onto the graph to the left of the On Button Click node, then connect them.

3. In the Graph Editor, add a Transform Rotate node (X Angle, Y Angle, Z Angle) and then assign it the following values: X Angle = 0, Y Angle = 15, Z Angle = 0.

4.Connect the On Button Click node to the flow input of the Transform Rotate node.

5.Drag the modelObject variable into the Graph Editor, then connect it to the transform input of the Transform Rotate node.

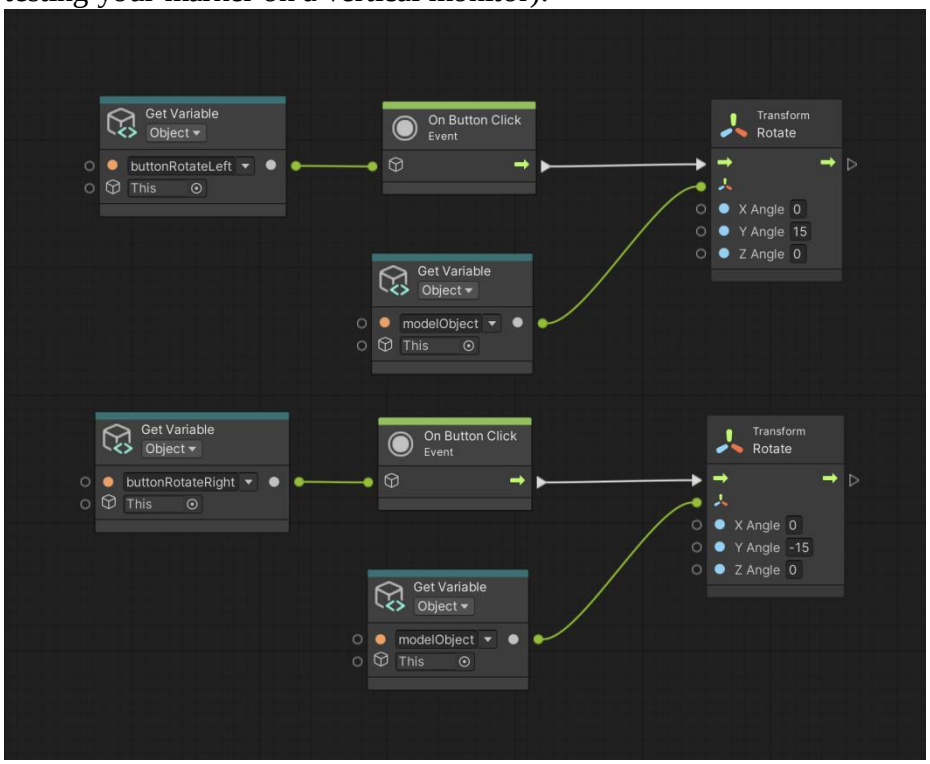
6.In the Hierarchy, select your in-Editor model. In the Inspector, enable it so it is visible in the Scene view.

7.Select Play in the Editor to test the script. You will be able to select the ButtonRotateLeft button in Scene view and see that your model rotates to the left. Then exit Play mode.

8.In the Script Graph window, click and drag across your existing graph to select all of the nodes.

9.Press Ctrl+D (macOS: Cmd+D) to duplicate the graph. Drag the new graph down so it's positioned directly below the old graph without overlapping.

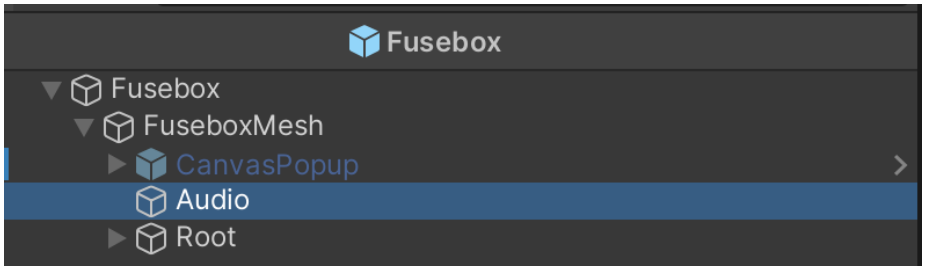
10. On the new graph, change the variable in the Get Variable node to buttonRotateRight, and change the Y Angle value to -15 (or X Angle if you are testing your marker on a vertical monitor).



3.1.5 Add Spatial Audio:

1. In the Project window, find your selected model's prefab and open it in Prefab Mode.

2. In the Hierarchy, create an empty GameObject as a child GameObject of the Mesh and name it "Audio". Audio should be next to CanvasPopup in the Hierarchy.

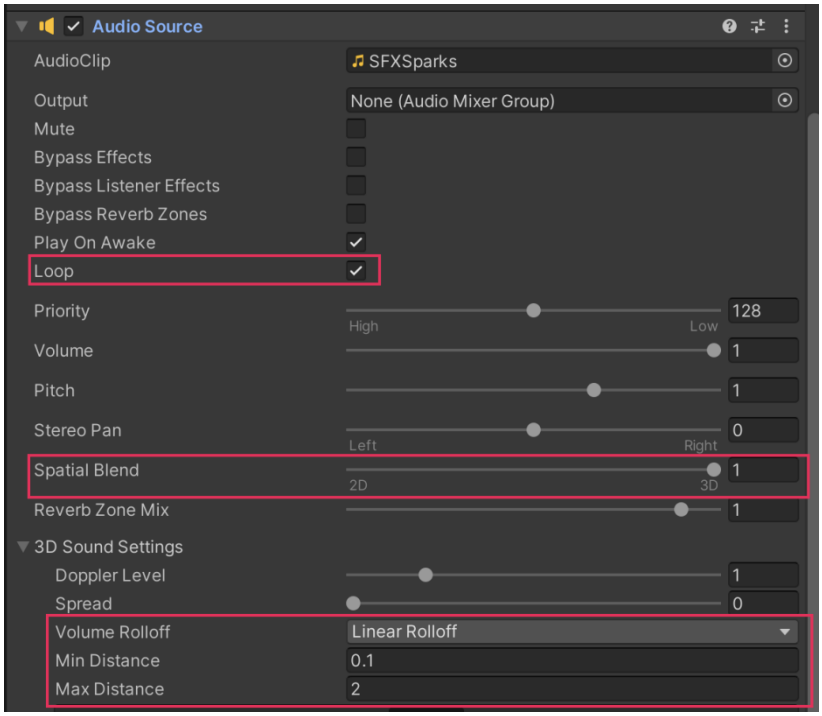


3. In the Inspector, add an Audio Source component to the Audio GameObject.

4. Find the AudioClip property of the Audio Source component, and use the picker (circle) to select the sound file you would like to play in your app. A few clips are included with the project files.

5. Enable Loop for testing purposes.
6. Set the Spatial Blend value to 1, which indicates spatial 3D audio. With a 3D spatial blend, the audio will get louder when the camera is closer to the audio source and quieter as the camera moves farther away.

7. Set the Volume Rolloff to Linear Rolloff. Set the Min Distance to 0.1 and the Max Distance value to 2.0. With these values, the volume will play at a maximum volume if you are 0.1 meters or closer to the model, but the volume will gradually decrease until you are two meters away, where you won't be able to hear it at all.



7.Build and test your app to see how your audio settings affect the experience. You can also test the audio in Unity by selecting Play, then gradually moving the model closer or farther away from the camera.

8. You will probably need to tweak the Min Distance and Max Distance properties of the Audio Source component as you get a feeling for the way spatial audio works.

You might need to build and test a few times to get the sounds to play exactly the way you want.

3.2 AR PLANE DETECTION

3.2.1 Building the project:

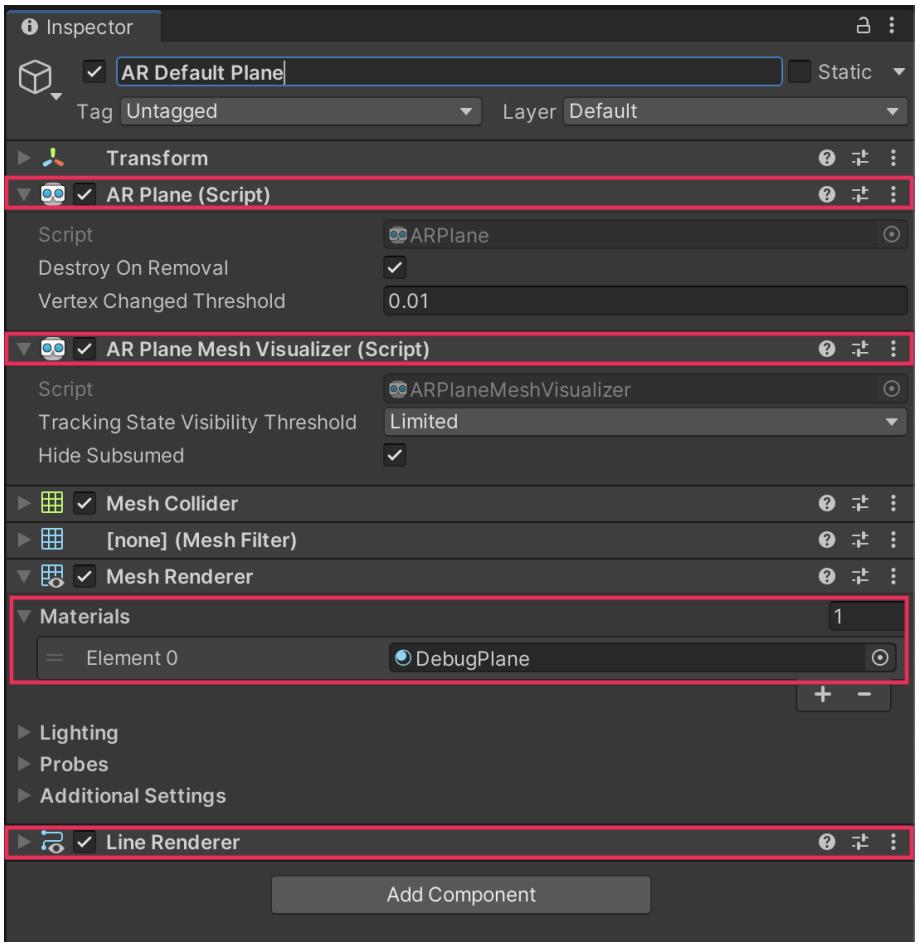
In the Hierarchy, right-click and select XR > AR Default Plane.
2. Select the new AR Default Plane GameObject and examine the following components in the Inspector:

The AR Plane (Script) component represents a plane detected by an AR device, which can be instantiated when a flat surface is detected.

The AR Plane Mesh Visualizer (Script) component handles the display of the material and outline of a detected AR plane.

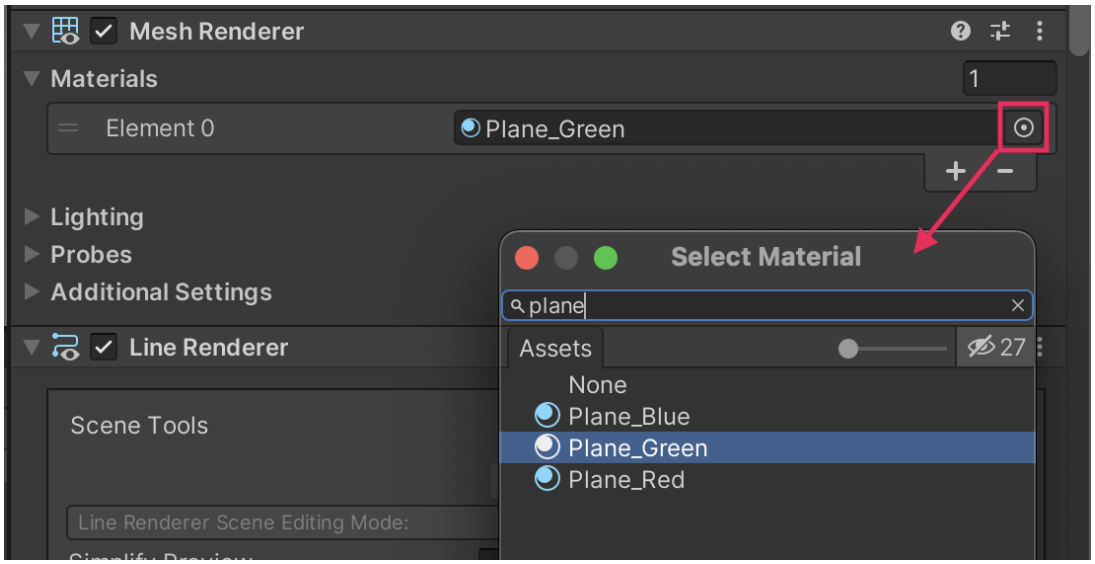
The Mesh Renderer component has a DebugPlane material assigned to it.

The Line Renderer component handles the appearance of the plane's border outline.



3. In the Mesh Renderer component, expand the Materials section.

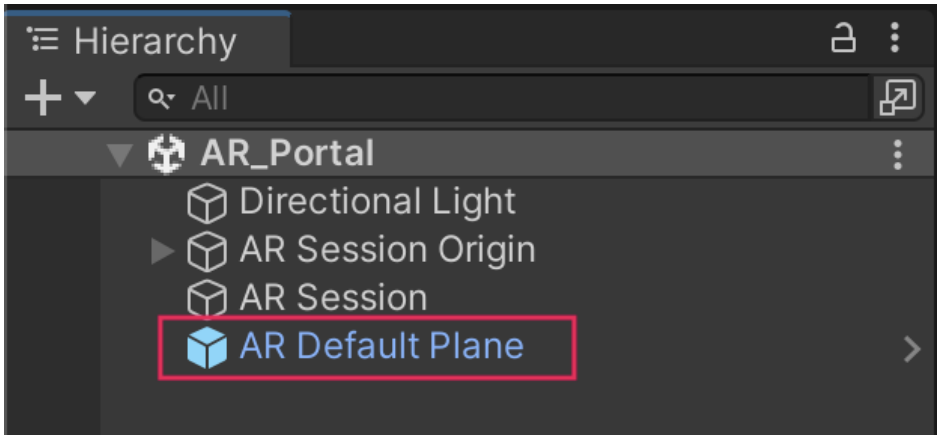
4. Select the Element 0 object picker (circle) and choose one of these three alternative materials: Plane_Green, Plane_Blue, or Plane_Red. The material that you select will be the color that detected planes will be highlighted.



Important: The default DebugPlane material does not work. If you skip the instruction above, your planes will not be displayed.

5. In the Project window, navigate to Assets > _ARPortal > Prefabs.

6. Click and drag the AR Default Plane GameObject from the Hierarchy into the Prefabs folder in the Project window. The GameObject icon and name should now be displayed in light blue in the Hierarchy.



7. Delete the AR Default Plane GameObject from the Hierarchy. AR Foundation will generate an AR Default Plane GameObject from the prefab that you have just created when a plane is detected.

3.2.2 Detect the user's touch:

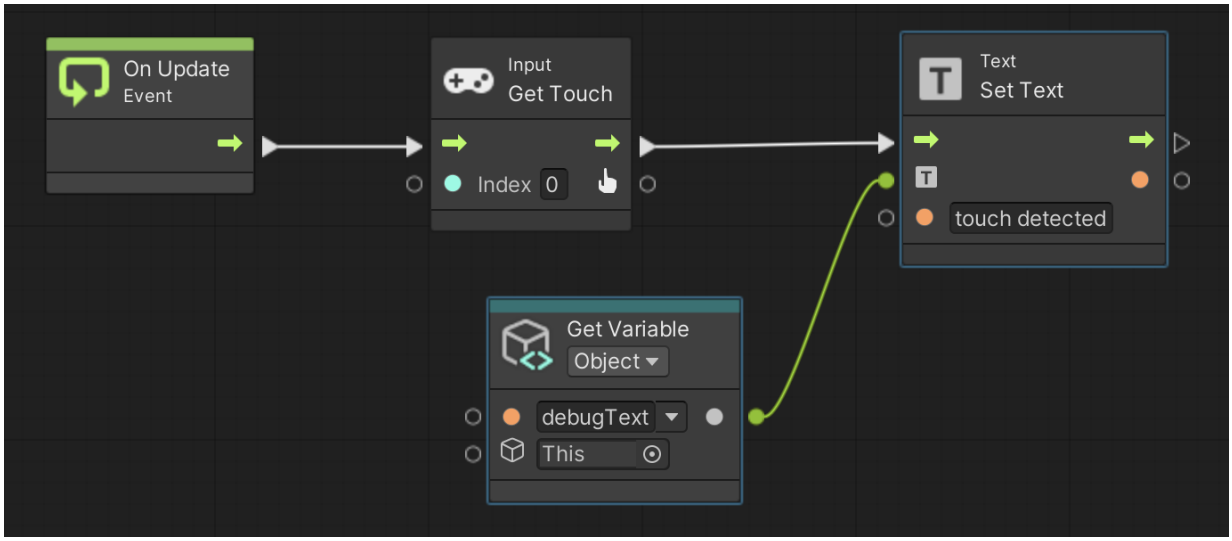
1. In the Graph Editor, use the fuzzy finder to add an Input Get Touch (Index) node. Connect the On Update event node flow output to your new node.

The Index represents the number of simultaneous screen touches. The Index value is set to 0 when no touches are detected. If one touch is detected, the Index value is set to

1. The second simultaneous touch increases the Index value to 2, and so on.

2. From the On Start event node sequence you created earlier, select and duplicate the Set Text and Get Variable nodes. Connect the Get Touch node flow output to the flow input of the duplicated Set Text node.

3. In the duplicated Set Text node, change the String value message to something like “touch detected”.



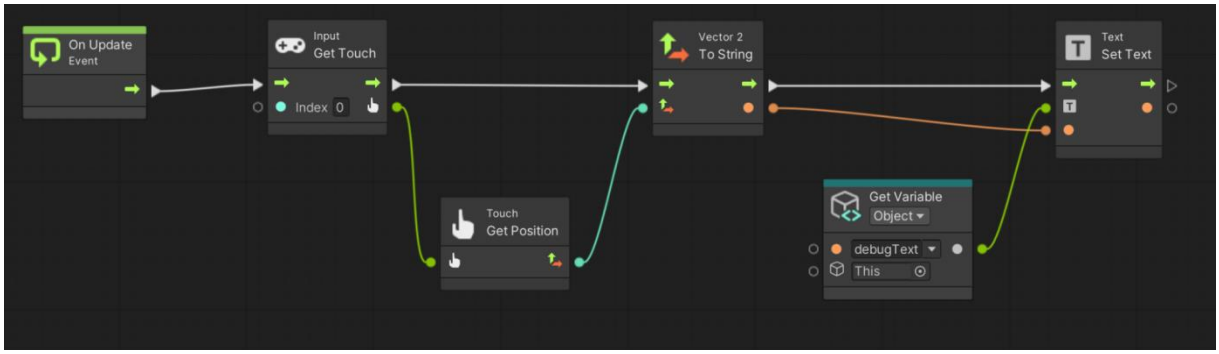
Build and run the app to preview the new functionality. When you touch the screen, the debug UI should display touch detected.

3.2.3 Display the touch position:

In the Graph Editor, add a Vector 2: To String node and insert it in the On Update sequence between the Get Touch and Set Text nodes.

2. Add a Touch: Get Position node, then connect it between the Get Touch and To String nodes.

3. Connect the String output from the To String node to the String input of the Set Text node.

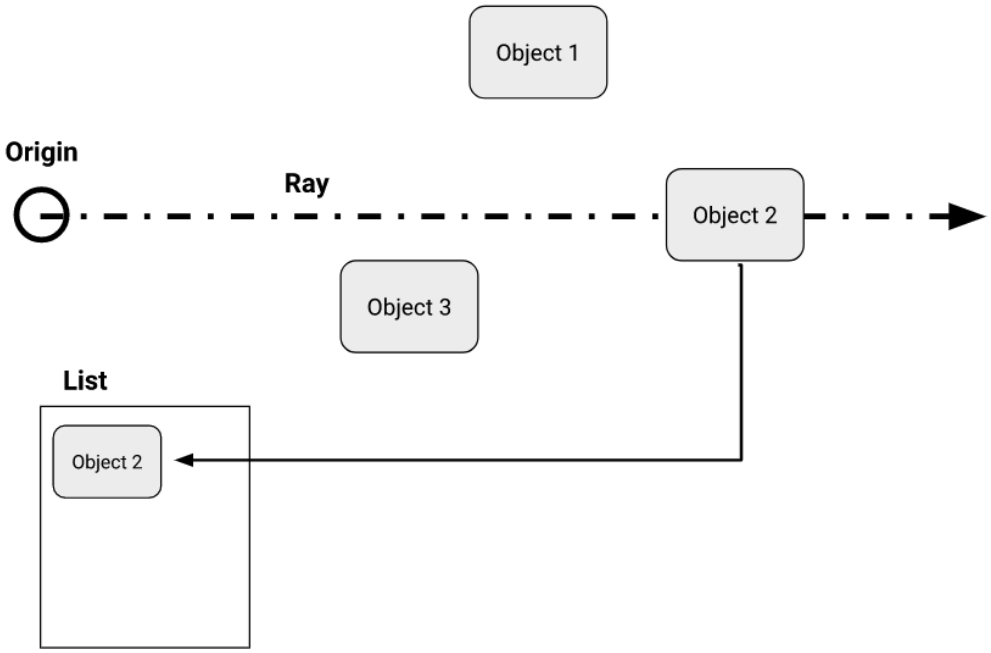


4. Build and run the app to preview the new functionality. As you touch the screen, the debug UI should display the touch position's X and Y coordinates.

3.2.4 Detect when user touches a plane(point):

Raycasting is the process of projecting an invisible straight line (a ray) from an origin point through a scene to determine what that line would hit. It's a little similar to fishing, where you cast a line to see what fish you can catch. With raycasting, you cast a ray to see what that ray would hit in the scene.

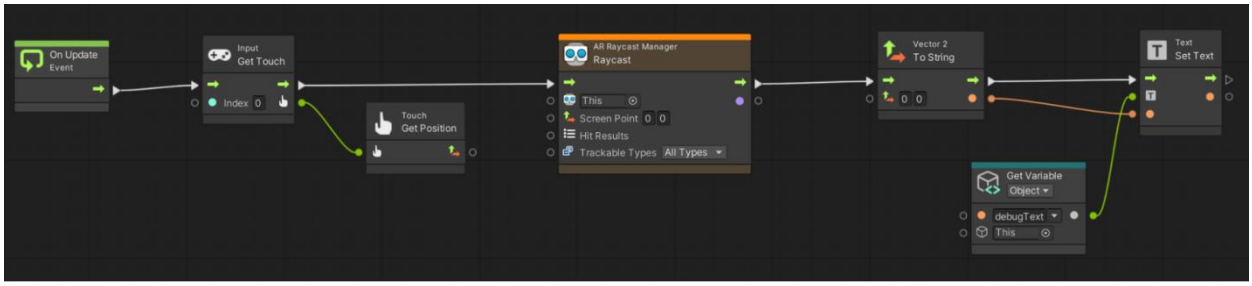
All of the objects hit by a raycast are usually stored in a List variable. You can then access the objects within that list and do whatever you want with them.



In the Graph Editor, make room for the new Raycast node after the Get Touch and Get Position nodes.

2. Add an AR Raycast Manager: Raycast (Screen Point, Hit Results, Trackable Types) node using the fuzzy finder. Important: Make sure to select the Raycast node with the specific inputs noted above – there are others with very similar inputs that don't provide the same functionality.

3. Disconnect and reconnect the nodes so that the raycast is performed after the Get Touch node, but before the To String node.

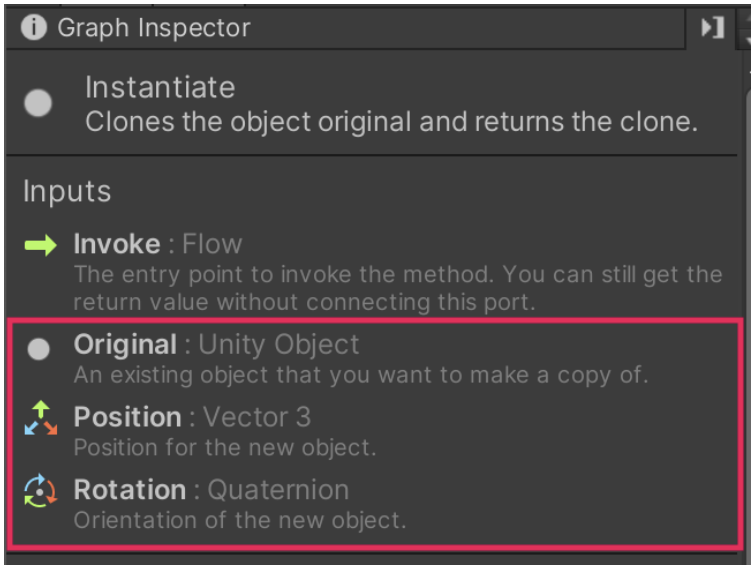


3.2.5 Instantiate the portal on point cloud:

When you spawn a new object (GameObject) in a scene it is often referred to as instantiating, as you are creating a new instance (version) of an object. In this case, you'll create a new instance of a portal prefab.

When you instantiate this object, you'll need to define the following inputs:

✓ The prefab to use as the Original
✓ The Position of the new instance
✓ The Rotation of the new instance



3.3 AR FACE FILTER(BASE)

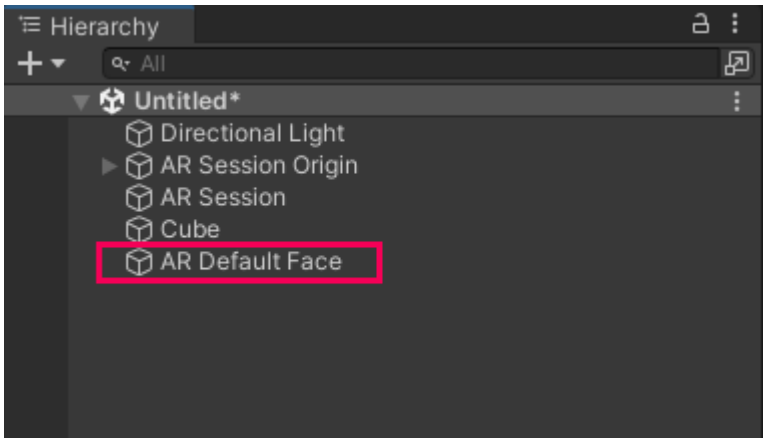
3.3.1 Building the project:

Augmented reality experiences that overlay digital content on your face require face tracking ability. Face tracking tells your mobile device where your face is when a camera sees it.
In order to see the AR Default Face when you test, you will create a prefab that will be activated when the camera sees your face. You will also need to apply a material to

that AR Default Face GameObject so that it shows up. A special material is provided as part of the project.

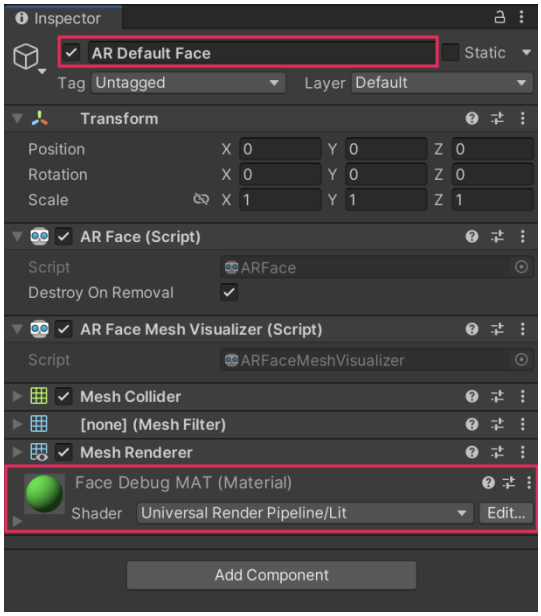
Follow these instructions to set up the AR Default Face prefab:

1. In the Hierarchy, right-click and select XR > AR Default Face to add a new AR Default Face GameObject to the scene. The AR Default Face is a GameObject you can customize to make your own creative face filters.



2. Next, you need to make sure that the AR Default Face will show up when a face is detected by your phone's camera. In the Project window, navigate to Assets > _BasicFaceFilter > Materials and find the Face Debug MAT material.

3. Drag the Face Debug MAT material from the Project window onto the AR Default Face GameObject in the Hierarchy to set the material. You can check if the material has been set by navigating to the Inspector window for the AR Default Face GameObject and scrolling down to the Material Inspector. Here, you should see the Face Debug MAT (Material).



4. In the Project window, navigate to Assets > _BasicFaceFilter > Prefabs.

5. Drag the AR Default Face GameObject from the Hierarchy into the Prefabs folder in the Project window. The GameObject icon and name should now be displayed in light blue in the Hierarchy.
You have just turned the AR Default Face GameObject into a prefab. Prefabs act as templates so that you can reuse GameObjects configured in a specific way.

3.3.2 Add a face texture:

One of the limitations of the AR Default Face prefab is that it only becomes visible when the application is running and a face is detected. This constraint makes debugging and design iteration a tedious task, since you have to build your project to see the results.

To solve this problem, a model is supplied by AR Foundation and included in the assets you imported earlier (Face_Reference_Mesh). A mesh is just another word for a 3D model. This Face Reference Mesh can be placed in the scene as a placeholder while you experiment with textures, accessories, and effects.

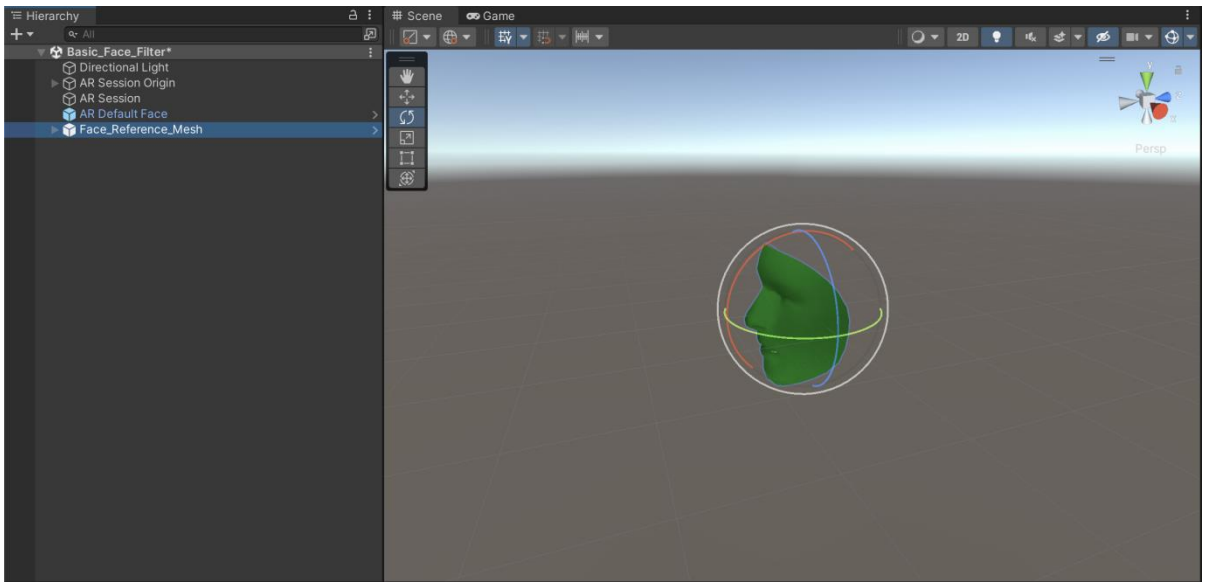
Follow these instructions to add the Face Reference Mesh to your scene:

1. Open your AR Template Unity project and the BasicFaceFilter scene, if you haven't already done so.

2. In the Project window, navigate to Assets > _BasicFaceFilter > Prefabs.

3. Drag the Face_Reference_Mesh prefab from the Project window into the Hierarchy or the Scene view.

4. To get a good view of the face mesh, double-click the Face_Reference_Mesh in the Hierarchy. This will frame the mesh in the Scene view.



3.3.3 Add accessories to the face filter:

Before you attach one of these 3D accessories to the face tracker, you need to add a visible face reference to your AR Default Face to help you place it properly.

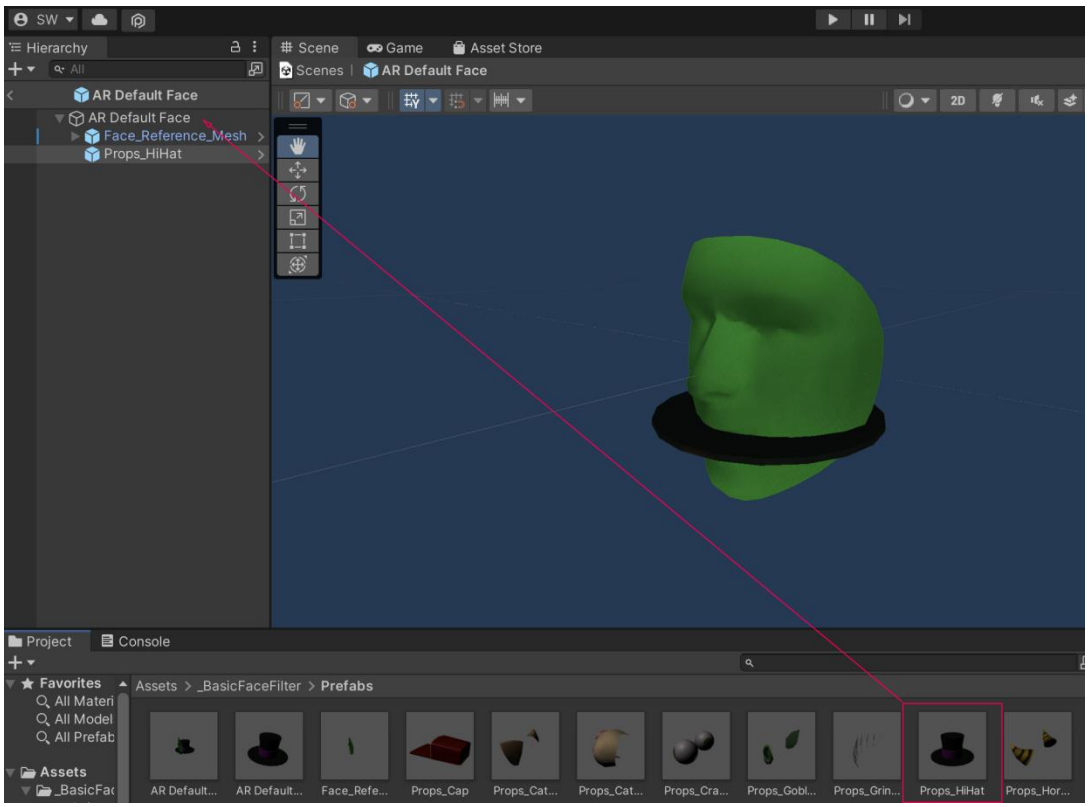
Follow the below instructions to add a face reference:

1. In the Project window, make sure that you are in the Prefabs folder.

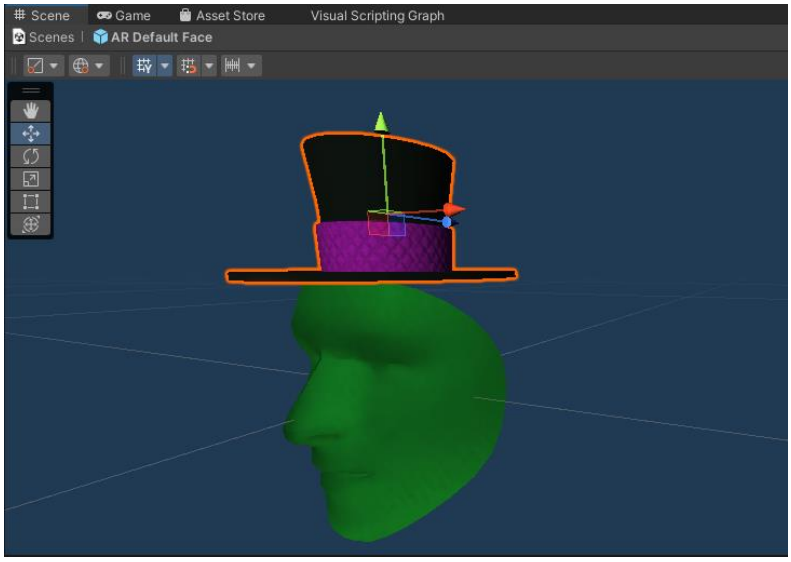
2. Double-click the AR Default Face prefab in the folder to open the prefab in the Scene view.

3. Drag the Face_Reference_Mesh prefab from the Project window into the Hierarchy to act as a reference. To zoom in on the Face_Reference_Mesh, double-click it in the Hierarchy.

4. Drag one of the provided accessories onto the AR Default Face GameObject in the Hierarchy. This will add the accessory to the prefab as a child of the GameObject.

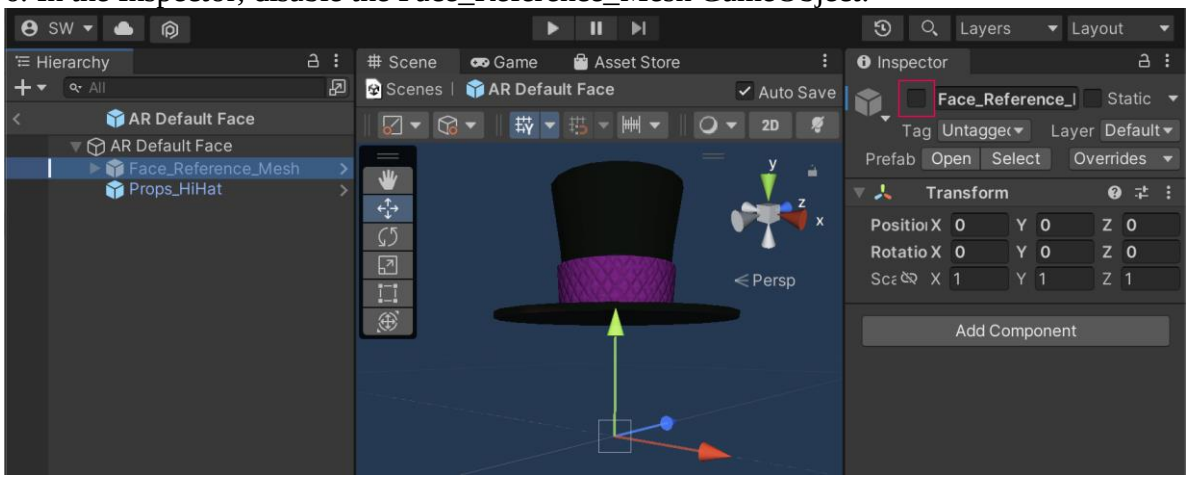


5. Use the Transform tools to place your chosen accessory in the correct position. You can also adjust the rotation and scale if necessary.

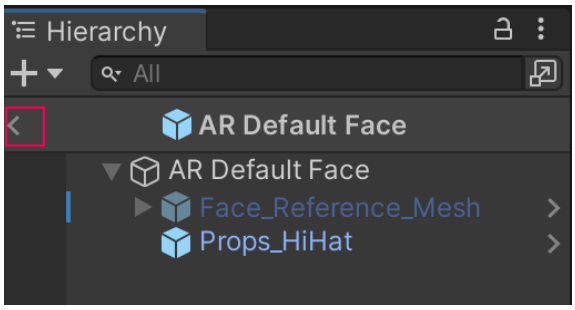


Note: If you need to refresh your memory, you can review the Transform tools and their shortcuts.

6. In the Inspector, disable the Face_Reference_Mesh GameObject.

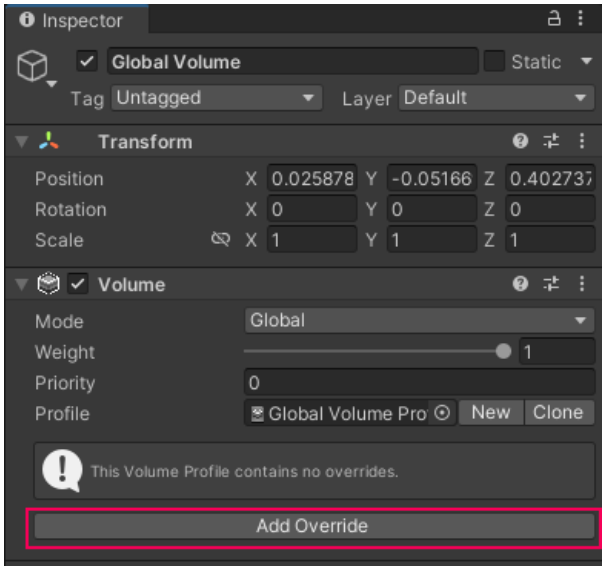


7. In the Hierarchy, select the back arrow in the top left of the window to exit Prefab editing mode.



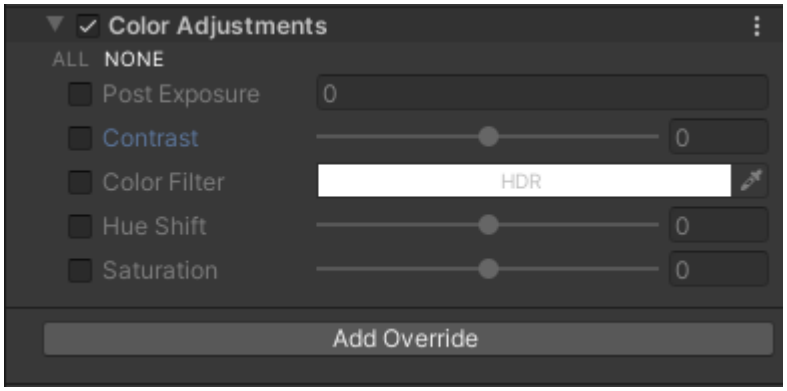
3.3.4. Add post-processing to the project:

1. In the Volume component, select Add Override.



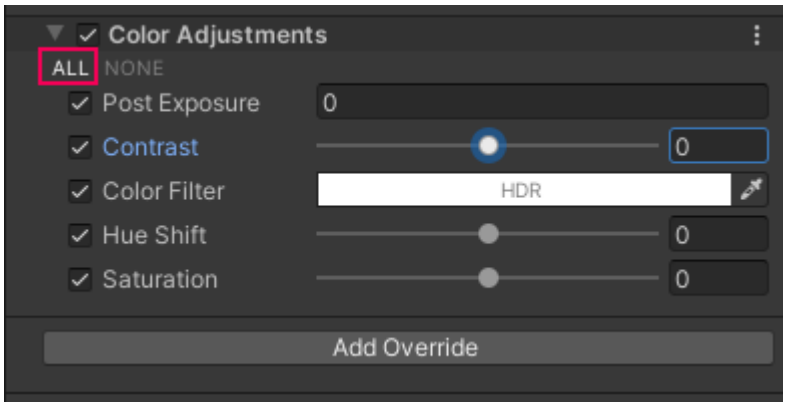
Each effect that you add to a volume is contained in a separate override that adjusts the volume's settings.

2. Search or navigate to select the Color Adjustments override.

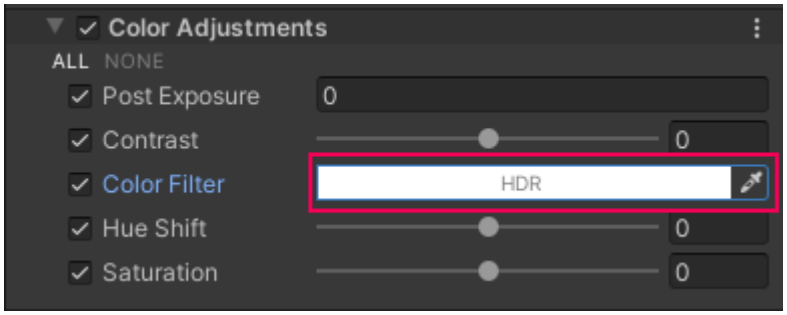


This effect will apply a color tint to everything that the camera sees. It's one of the most obvious post-processing effects, so it's a good one to test for your app.

3. Select All to enable every property in the Color Adjustments override.



4. Select the Color Filter swatch and choose tint color for the scene in the dialog window.



3.4 AR FACE FILTER(INTERACTIVE)

3.4.1 Building the project:

Follow these instructions to set up your new scene:

1. Select File > New Scene to create a new scene from the AR template.

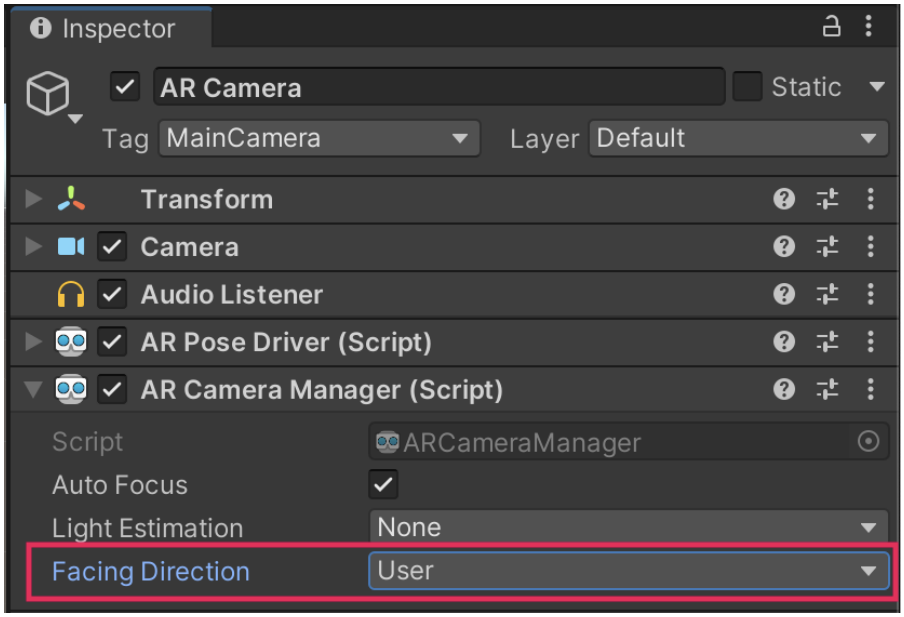
2. Select File > Save As, then save the scene as “InteractiveFilter” inside the Scenes folder.

3. In the Hierarchy, delete the Cube GameObject.

4. In the Hierarchy, select the AR Camera GameObject (it’s a child of AR Session Origin).

5. In the Inspector, find the AR Camera Manager component and set the Facing Direction to User.

6. Save the scene.
Your scene is now ready.



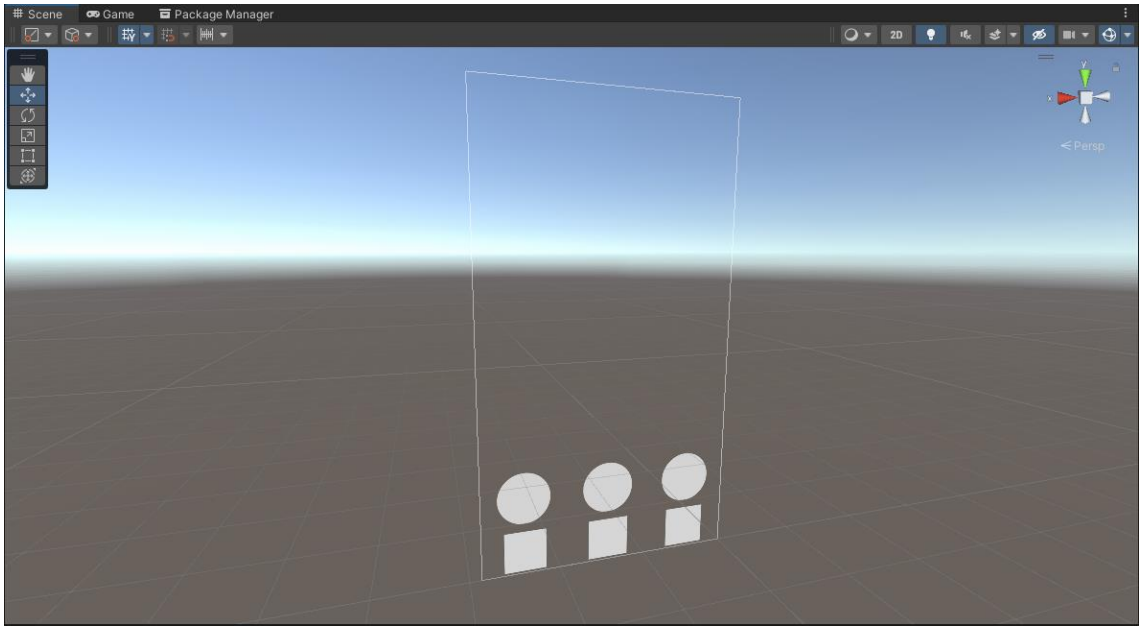
3.4.2 Setup the UI:

This project uses a UI system called Unity UI (or uGUI). This system organizes user interfaces using a Canvas component — you need to create every part of the user interface as the child of a GameObject with this Canvas component added to it. We've already done this for your app, but you'll need to configure it to suit your specific needs.

Follow these instructions to add the AR_Canvas prefab to your scene:

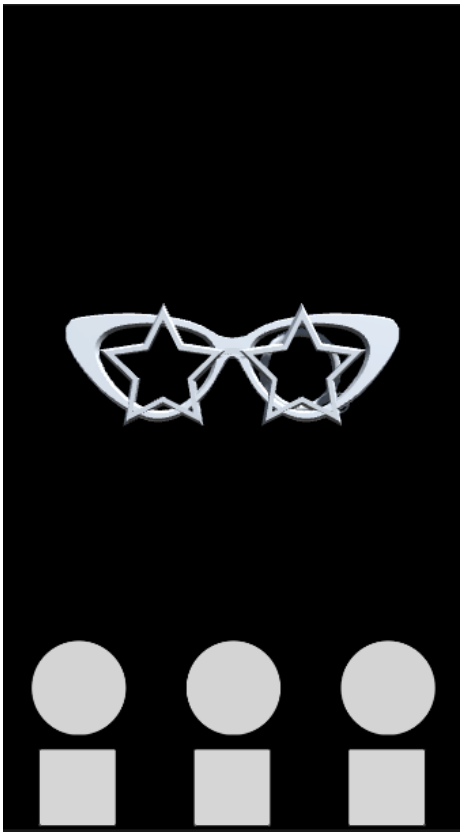
1. In the Project window, go to Assets > _InteractiveFaceFilter > Prefabs.

2. Drag the AR_Canvas prefab from the Project window into the Hierarchy.



3. To preview your changes without playing your app, open the Game view window.

4. In the Hierarchy, select the GlassesGroup GameObject and then enable the GameObject in the Inspector. The glasses models should appear in the Game view.



The AR_Canvas prefab is rendering (being displayed) as a Screen Space - Overlay UI. This means that the prefab will always render on top of the app camera and appear in precisely the position that it is displayed in the Game view.

5. In the Hierarchy, expand the AR_Canvas GameObject using the foldout icon to reveal its child GameObjects.

The AR_Canvas is a parent to three GameObjects: Glasses Buttons, Materials Buttons, and EventSystem. The glasses and material button groups are parent GameObjects of the individual buttons you see in the scene. The EventSystem is a GameObject that contains the components that the UI needs to detect user input such as tapping or clicking.

3.4.3 Visual Scripting to switch between glasses:

You now have an event that can be called whenever you need it to run. To actually call it, you're going to use an Event Trigger node:

In the Graph Editor, open the fuzzy finder and add a Trigger Custom Event node.

1. You can think of a Custom Event Trigger node like a link. When this node is called, it will initiate the sequence of nodes in the custom event it's associated with.

2. Disconnect the flow between the On Button Click node and the Find With Tag node in the first button graph by right-clicking on the flow output of the On Button Click node.

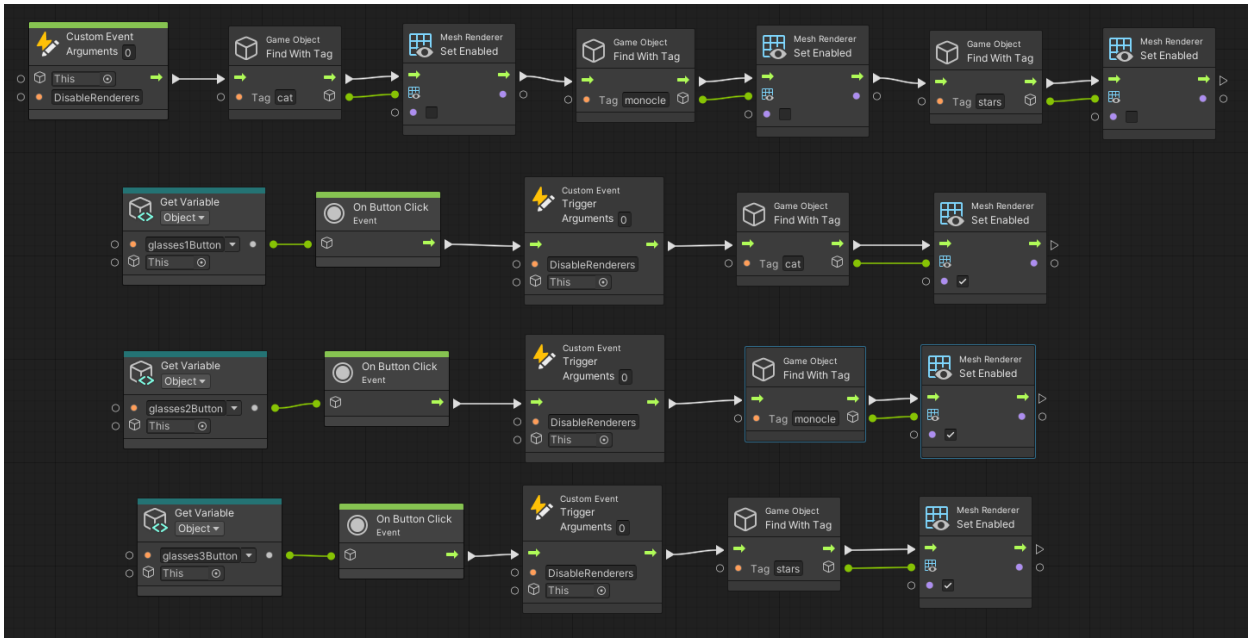
3. Move the Get Variable and On Button Click nodes to make room for the Trigger node.

4. Position the Trigger node in the space and then connect the flow output of the On Button Click node to the flow input of the Trigger node.

5. Connect the flow output of the Trigger node to the flow input of the Find With Tag node.

6. Duplicate the Trigger node twice more.

6. Repeat the previous steps for the remaining two button graphs.

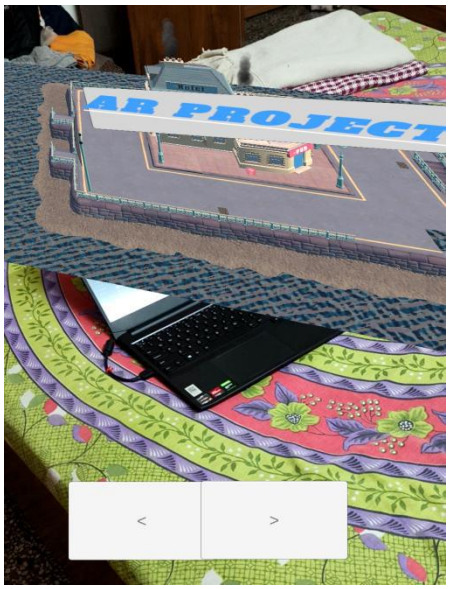


CHAPTER 4

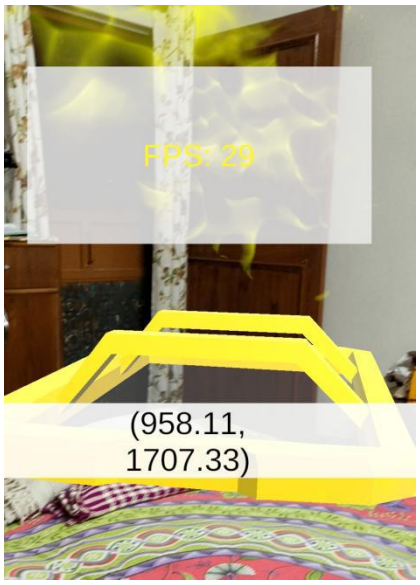
RESULTS AND DISCUSSION

4.1 RESULTS

AR MARKER



AR PLANE DETECTION



AR FACE FILTER(BASE)

AR FACE FILTER(INTERACTIVE)

4.2 WHY AND HOW?

Why AR/XR is used?

- ✧ Enhanced Visualization: AR/XR overlays digital information onto the real-world environment, providing an enhanced visualization experience. This is beneficial in fields such as architecture, design, engineering, and education, where users can visualize 3D models, annotations, and instructions seamlessly integrated with their physical surroundings.
- ✧ Training and Simulation: AR/XR offers immersive training experiences by simulating real-world scenarios in a safe and controlled environment. This is particularly useful in industries like healthcare, military, and manufacturing, where personnel can practice complex procedures or operations without actual risks.
- ✧ Remote Assistance: AR/XR enables remote collaboration and assistance by allowing experts to provide real-time guidance and instructions to on-site personnel. This is valuable in fields like maintenance, repair, and technical

support, where skilled professionals can virtually guide others through complex tasks.

How AR/XR is used?

- AR (Augmented Reality) and XR (Extended Reality) technologies can be used in various ways across different industries and applications. Here are some common ways in which AR/XR is used:
- Mobile Apps: Smartphones and tablets are widely used platforms for AR/XR experiences. Users can point their device's camera at a real-world object or environment, and the app overlays digital information, 3D models, or interactive elements on the screen.
- Head-Mounted Displays (HMDs): AR/XR headsets, such as Microsoft HoloLens, Magic Leap, and various VR headsets, provide a more immersive experience by projecting digital content directly into the user's field of view.
- Smart Glasses: AR smart glasses, like Google Glass or Vuzix Blade, allow users to view digital information overlaid on their real-world environment hands-free.

CHAPTER 5

CONCLUSION AND FUTURE

5.1 CLASSIFIACTION IN AR

In augmented reality (AR), classification plays a crucial role in identifying and understanding real-world objects and environments. Here are some common classification techniques used in AR:

- ✓ Object Detection and Recognition:
  - Image Classification: This technique involves training machine learning models to classify images into different categories, such as identifying objects, scenes, or activities.
  - Object Detection: In addition to classifying objects, object detection algorithms can locate and draw bounding boxes around objects of interest within an image or video frame.
  - Instance Segmentation: This advanced technique not only detects and classifies objects but also provides pixel-level segmentation, separating each instance of an object from the background and other objects.

- ✓ Pose Estimation:
  - 3D Pose Estimation: By analyzing images or video frames, pose estimation algorithms can determine the 3D orientation and position of objects relative to the camera or user's viewpoint. This information is crucial for accurately overlaying virtual content onto real-world objects.
  - Human Pose Estimation: Specialized algorithms can detect and track human body poses, enabling applications such as motion tracking, gesture recognition, and avatar animation.

- ✓ Scene Understanding:
  - Semantic Segmentation: This technique classifies and assigns a label to every pixel in an image or video frame, enabling a detailed understanding of the scene composition and the relationship between different objects and surfaces.
  - Depth Estimation: By estimating the depth or distance of objects and surfaces from the camera, AR systems can more accurately place and integrate virtual content within the real-world environment.

- ✓ Text Recognition:
  - Optical Character Recognition (OCR): OCR algorithms can detect and extract text from images, enabling AR applications to recognize and overlay digital information on top of printed text or signage.

- ✓ Machine Learning/Deep Learning Techniques:
  - Convolutional Neural Networks (CNNs): These deep learning models are widely used for image classification, object detection, and segmentation tasks in AR.

- Transfer Learning: Pre-trained models on large datasets can be fine-tuned for specific AR tasks, reducing the need for extensive training data and computational resources.
- Reinforcement Learning: In some cases, reinforcement learning techniques are used to train AR models to make decisions and take actions based on the recognized objects and scene understanding.

5.2 FURTHER READING

1. Books:

- "Augmented Reality: Principles and Practice" by Dieter Schmalstieg and Tobias Höllerer - A comprehensive textbook covering the fundamentals, algorithms, and applications of AR.
- "Practical Augmented Reality" by Steve Aukstakalnis - A practical guide to building AR applications and understanding the underlying technologies.

"The Fourth Transformation: How Augmented Reality & Artificial Intelligence Will Change Everything" by Robert Scoble and Shel Israel - Explores the potential impact of AR/XR on various industries and sectors.

2. Academic Publications and Journals:

- IEEE Transactions on Visualization and Computer Graphics (TVCG) - A leading journal that publishes research papers on AR, VR, and computer graphics.
- International Symposium on Mixed and Augmented Reality (ISMAR) - Proceedings of the premier conference on AR and mixed reality research.
- IEEE International Symposium on Mixed and Augmented Reality (ISMAR) - Another highly regarded conference for AR and XR research.

3. Online Resources and Tutorials:

- Augmented World Expo (AWE) - An annual conference and online resource dedicated to AR/XR technologies and applications.
- Unity Learn: Augmented Reality - Tutorials and resources for building AR applications using the Unity game engine.
- Google AR & VR - Google's resource center for developers, including documentation, tutorials, and samples for ARCore and other AR/VR platforms.
- Apple Developer: ARKit - Apple's developer resource for building AR applications using ARKit.

References

https://learn.unity.com/project/create-a-marker-based-ar-app?uv=2021.3&pathwayId=63e3a4c1edbc2a344bfe21d8&missionId=63f77ca2edbc2a007912d446

https://learn.unity.com/project/create-a-plane-detection-ar-app?uv=2021.3&pathwayId=63e3a4c1edbc2a344bfe21d8&missionId=63f77ca2edbc2a007912d446

https://learn.unity.com/project/create-a-basic-face-filter-tutorials?uv=2021.3&pathwayId=63e3a4c1edbc2a344bfe21d8&missionId=63e3a840edbc2a169e742f89

https://learn.unity.com/project/interactive-face-filter?uv=2021.3&pathwayId=63e3a4c1edbc2a344bfe21d8&missionId=63e3a840edbc2a169e742f89

"A Survey of Augmented Reality" by Ronald T. Azuma - One of the earliest and most cited papers that defines and introduces the concept of AR.

"A Review of Augmented Reality for Virtual, Remote and Augmented Virtuality Applications" by S. K. Ong and A. Y. C. Nee - A comprehensive review of AR applications and technologies.

"Simultaneous Localization and Mapping for Augmented Reality" by Georg Klein and David Murray - A seminal paper on SLAM (Simultaneous Localization and Mapping) techniques for AR.

"AR/VR Headsets: A Review of the Current State" by Vincenzo Ferraro and Massimo Verri - Reviews and compares various AR/VR headsets and their capabilities.