KESHAV MEMORIAL INSTITUTE OF TECHNOLOGY

PROJECT SCHOOL CERTIFICATE

PROJECT TITLE: VR/AR Medical Training Application

FACULTY INCHARGE: Dr. Devika Rubi

DURATION OF PROJECT: 31/03/2023 – 15/07/2023

NAME: M S Harshith Reddy

CLASS: II-CSM-A

ROLL-NO: 21BD1A6611

GitHub: https://github.com/harshithreddyms17?tab=repositories

Harshith
Signature of Student

Devika Rubi 18/7/23
Signature of Faculty

Signature of Examiner

INDEX

Introduction

Augmented Reality

Augmented reality (AR) is a technology that overlays computer-generated content onto the real world, enhancing the user's perception and interaction with their environment. AR integrates digital information, such as images, videos, and 3D models, into the user's real-time view of the physical world, usually through a device like a smartphone, tablet, or specialized AR glasses.

AR technology typically involves three main components:

- Display: AR experiences can be viewed through various devices, including smartphones, tablets, smart glasses, or headsets. These devices provide the visual overlay of digital content onto the real world.
- Tracking: To accurately position virtual objects in the real world, AR systems use tracking technologies like GPS, sensors (such as accelerometers, gyroscopes, and magnetometers), and computer vision techniques. These technologies enable the device to understand the user's location, orientation, and movements.
- Content: The digital content overlaid onto the real world in AR can include images, text, videos, 3D models, animations, and more. This content is often created through computer graphics and is rendered in real-time to align with the user's perspective and movements.

Applications of augmented reality are widespread and continue to expand. Some common examples include:

- Gaming
- Education and Training
- Retail and E-commerce
- Healthcare
- Architecture and Design
- Navigation and Wayfinding
- Industrial and Manufacturing

Augmented Reality in Medical Training

Augmented reality (AR) has shown significant potential in medical training, offering immersive and interactive experiences that can enhance learning and skill

development for healthcare professionals. Here are some ways AR is used in medical training:

- Anatomy Education: AR can provide medical students and trainees with interactive 3D models of anatomical structures. These virtual models can be overlaid onto physical objects, such as a human body or anatomical models, allowing students to visualize and explore complex anatomical structures in a more interactive and engaging manner.
- Surgical Training: AR can be utilized to simulate surgical procedures and provide virtual guidance during training. Surgeons can wear AR glasses or use AR-enabled devices to view real-time data, such as patient anatomy, vital signs, or instrument tracking, overlaid onto their field of view. This can help improve surgical precision, spatial awareness, and decision-making during procedures.
- Patient Simulations: AR can create realistic patient simulations for trainees to practice diagnostic and treatment procedures. Trainees can interact with virtual patients, examine symptoms, perform physical examinations, and practice making diagnoses and treatment decisions in a controlled and safe environment.
- Medical Imaging Interpretation: AR can enhance the interpretation of medical imaging, such as X-rays, CT scans, and MRIs. By overlaying relevant information, annotations, or 3D reconstructions onto the imaging studies, AR can assist trainees in understanding complex medical images and identifying abnormalities.

Virtual Reality

Virtual Reality (VR) is an immersive technology that creates a simulated, computer-generated environment that users can interact with and explore. Unlike augmented reality (AR), which overlays virtual elements onto the real world, VR completely replaces the user's sensory experience with a virtual environment.

VR typically involves the following components:

- Head-mounted display (HMD): Users wear a specialized VR headset that consists of a high-resolution display screen placed close to the eyes. The HMD blocks out the real world and presents a stereoscopic view of the virtual environment, providing a 360-degree field of view.
- Tracking and input devices: VR systems utilize tracking technologies, such as infrared sensors, cameras, or gyroscopes, to monitor the user's head movements and adjust the displayed image accordingly. Additionally, input devices like

handheld controllers or data gloves enable users to interact with objects in the virtual world.

- Computer graphics and rendering: To create realistic virtual environments, VR relies on powerful computer graphics and rendering techniques. These technologies generate and display three-dimensional objects, textures, lighting effects, and animations in real-time, ensuring a seamless and immersive experience.

VR offers various applications and benefits in different fields, including:

- Gaming and Entertainment
- Education and Training
- Architecture and Design
- Virtual Tourism and Exploration
- Therapy and Rehabilitation
- Collaboration and Communication

Virtual Reality in Medical Training

Virtual reality (VR) has significant applications in medical training, providing realistic and immersive simulations that allow healthcare professionals to practice various procedures, improve their skills, and enhance patient care. Here are some ways VR is used in medical training:

- Surgical Simulation: VR allows surgeons to practice complex surgical procedures in a virtual environment. By using haptic feedback devices and surgical instruments, trainees can interact with virtual patients, simulate surgical steps, and gain hands-on experience without the risks associated with real patients. VR surgical simulations can help improve surgical precision, dexterity, and decision-making.
- Procedural Training: VR is used to train healthcare professionals in a range of procedures, such as intubation, catheterization, central line insertion, and ultrasound-guided interventions. Trainees can repeatedly practice these procedures in a realistic virtual environment, receiving real-time feedback and guidance to refine their skills and build confidence.
- Anatomical Visualization: VR provides detailed and interactive 3D models of anatomical structures. Medical students and trainees can explore and manipulate virtual organs, bones, and systems, enhancing their understanding of complex anatomy and spatial relationships.
- Patient Interaction and Communication: VR can simulate realistic patient scenarios, allowing trainees to practice communication and interpersonal skills.

Trainees can interact with virtual patients, take medical histories, practice empathetic communication, and learn to handle challenging patient interactions.

- Medical Imaging Interpretation: VR is used to enhance the interpretation of medical imaging studies. By visualizing medical images, such as CT scans, MRI scans, or X-rays, in a 3D virtual environment, trainees can develop skills in identifying abnormalities, understanding spatial relationships, and improving diagnostic accuracy.

Project Overview

This project is a compilation of the tasks that were performed by me during the course of the project school. The objective is to be visualize medical procedures in Augmented Reality and Virtual Reality through Unreal Engine, and First-Person Perspective based tasks and Virtual Reality through Unity. By building/acquiring interactive 3D-models, animations, textures etc, we were able to identify and develop specific medical procedures, scenarios or skills we wanted to simulate in an augmented and virtual environment. This includes CT scan animation, CPR training, regular health check-up simulation, syringe animation etc.

System Configuration

Name - OMEN by HP Gaming Laptop 16-n0xxx

Processor - AMD Ryzen 7 6800H with Radeon Graphics 3.20 GHz

Installed RAM - 16 GB

System type - 64-bit operating system, x64-based processor

Implementation

Technology Stack

- **Unreal Engine 5.0.3** - Unreal Engine is a popular and powerful game development engine created by Epic Games. It provides developers with a comprehensive suite of tools and features to build high-quality games and interactive experiences for various platforms such as PC, consoles, mobile devices, and virtual reality.
- **Unity 2022.3.1f1** - Unity is a popular and versatile game development engine that allows developers to create games and interactive experiences for various

platforms. It offers a wide range of tools, features, and a user-friendly interface to streamline the game development process.

- **Blender 3.5** - Blender is a free and open-source 3D creation suite that offers a comprehensive set of tools for modelling, animation, rendering, compositing, and more. It is widely used by artists, animators, game developers, and visual effects professionals.
- **Android Studio 4.0** - Android Studio 4.0 is an Integrated Development Environment (IDE) for Android app development. It was released by Google in May 2020 and introduced several new features and improvements to enhance the development workflow.
- **Android Studio 4.2 Electric Eel** - Android Studio 4.2 Electric Eel was released by Google in May 2021. Android Studio is the official Integrated Development Environment (IDE) for Android app development.

Installation Guide

Unity

- Visit the official Unity website: Go to the Unity website at https://unity.com/ and navigate to the "Get Started" or "Download" section.
- Choose the appropriate version: Unity offers different versions, including the Personal (free) and Professional (paid) editions. Select the version that suits your needs and click on the download button.
- Select the operating system: Unity supports Windows and macOS. Choose the appropriate operating system you're using.
- Run the installer: Once the download is complete, run the Unity installer file.
- Customize the installation: The Unity installer provides customization options. You can choose the installation location, additional components, and modules you want to install. It's recommended to select the necessary components based on your project requirements.
- Accept the license agreement: Read the license agreement and accept it to proceed with the installation.
- Wait for the installation: The installer will start installing Unity and its components. This process may take some time depending on your system and the components you selected.
- Optional: Sign in or create a Unity account: Unity allows you to sign in with your existing Unity account or create a new one. This step is not mandatory for installation but can be useful for accessing additional Unity services and features.

- Complete the installation: Once the installation is finished, you'll receive a notification or see a completion message. At this point, Unity should be successfully installed on your computer.
- Launch Unity: After the installation, you can launch Unity from the Start menu (Windows) or the Applications folder (macOS). The Unity editor will open, and you'll be ready to start creating your projects.

Unreal

- Visit the official Unreal Engine website: Go to the Unreal Engine website at https://www.unrealengine.com/ and navigate to the "Get Unreal" or "Download" section.
- Create an Epic Games account: To download and install Unreal Engine, you'll need to create an account with Epic Games. Click on the "Sign In" or "Create Account" button and follow the instructions to set up your account.
- Download the Epic Games Launcher: Once you have an Epic Games account, download the Epic Games Launcher from the website. The launcher is a platform that allows you to manage and download various Epic Games products, including Unreal Engine.
- Install the Epic Games Launcher: Run the Epic Games Launcher installer file that you downloaded. Follow the on-screen instructions to install the launcher on your computer.
- Sign in to the Epic Games Launcher: Launch the Epic Games Launcher and sign in using your Epic Games account credentials.
- Install Unreal Engine: In the Epic Games Launcher, navigate to the "Library" section. Locate Unreal Engine in the list of available products and click on the "Install" button. You can choose the version of Unreal Engine you want to install, depending on the available options.
- Customize the installation: The Unreal Engine installation process allows you to customize the installation location and additional components you want to install. You can choose the desired components based on your requirements.
- Wait for the installation: The Epic Games Launcher will download and install Unreal Engine on your computer. The installation time may vary depending on your internet speed and the size of the selected components.
- Launch Unreal Engine: After the installation is complete, you can launch Unreal Engine from the Epic Games Launcher. Click on the "Launch" button next to the installed version of Unreal Engine.

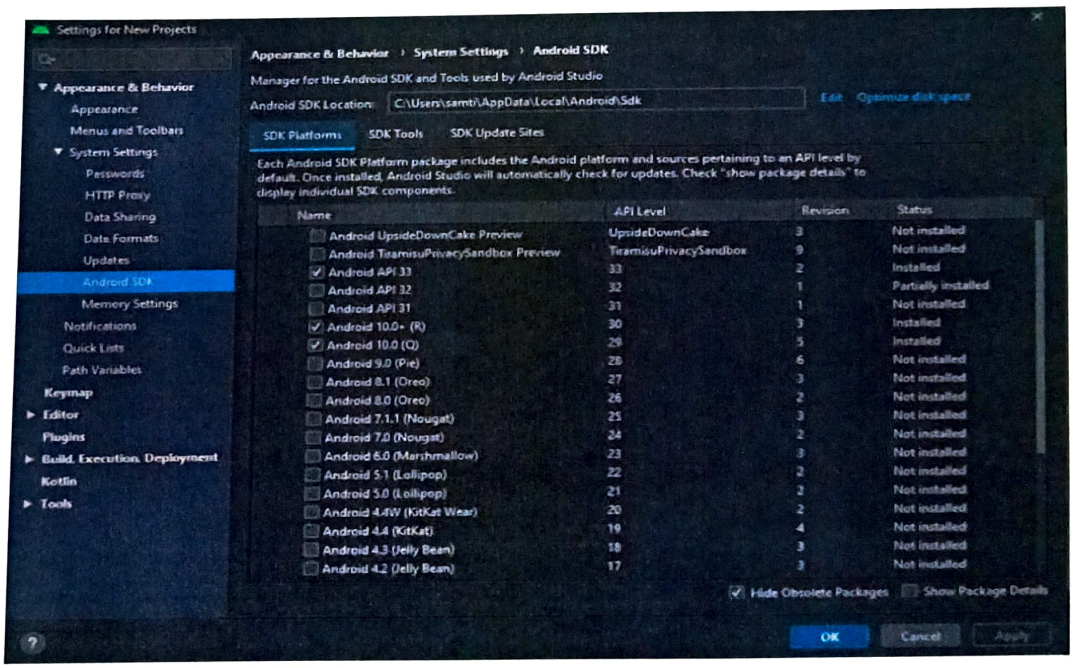
Android Setup for Augmented Reality in Unreal

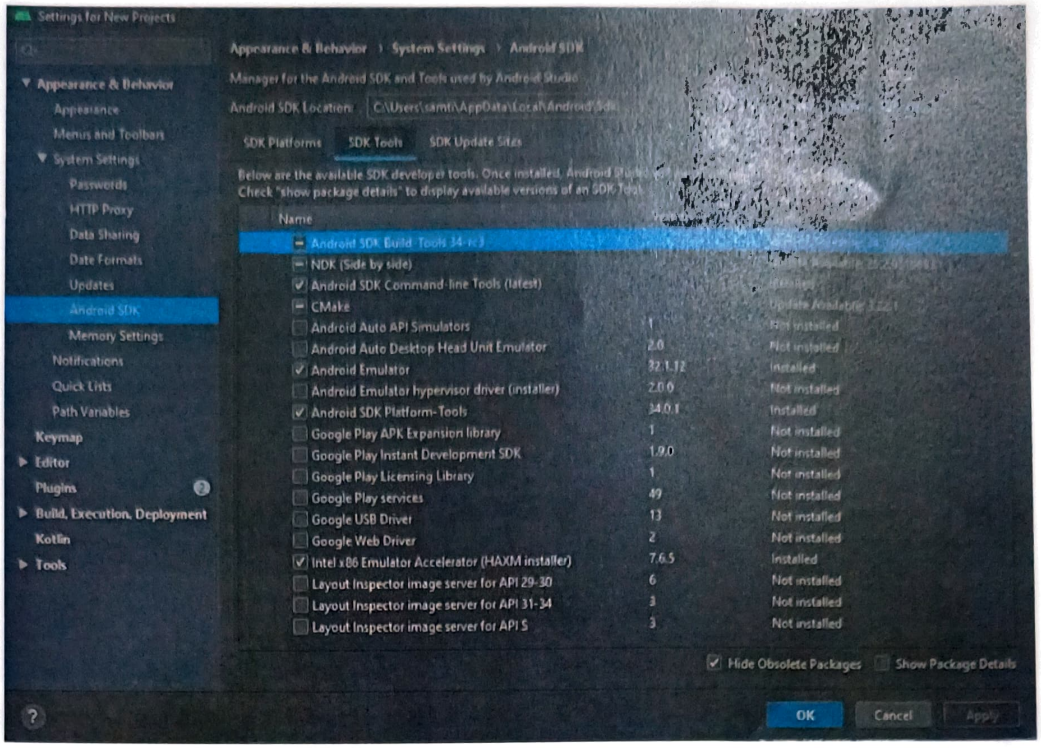
- **Install Android Studio:**
  Install the *Android Studio 4.0 May 28 2020 version (Windows IDE only 64-bit)*. Since we are using Windows, download the Windows 64-bit installer. This is found on the official *Android Studio website*.

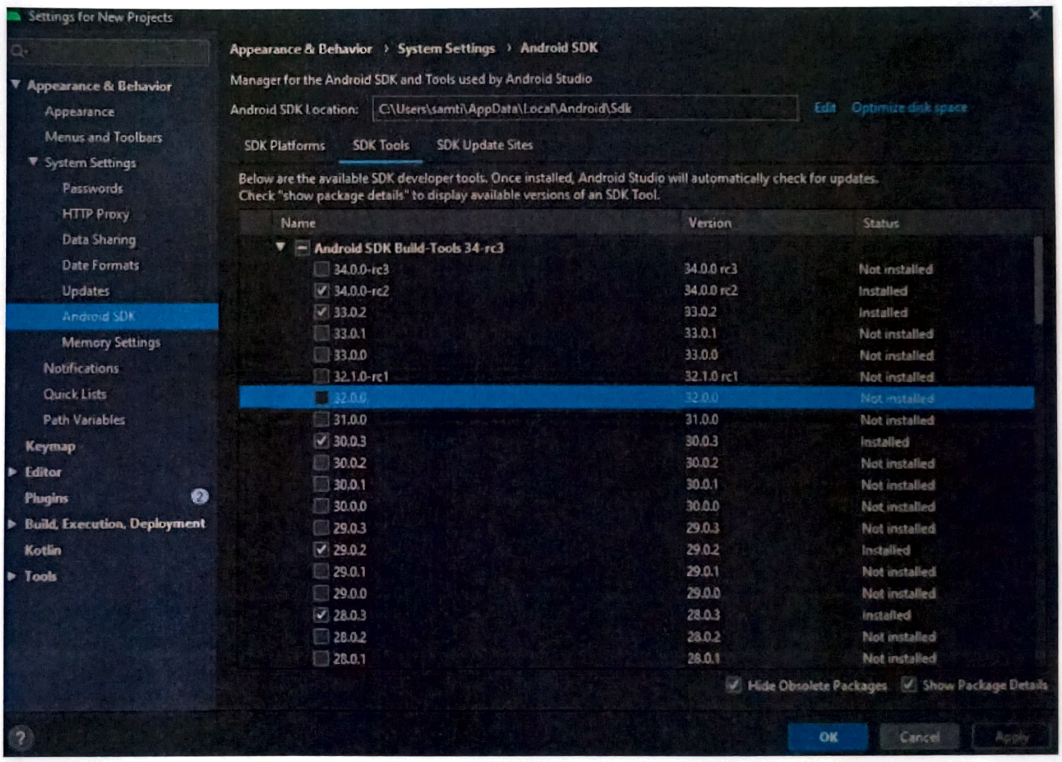
- **SDK and NDK Setup:**
  The Android SDK (Software Development Kit) is a set of development tools used to develop Android platform applications.
  The Native Development Kit (NDK) is a set of tools that allows you to use C and C++ code with Android and provides platform libraries to manage native activities and access physical device components, such as sensors and touch input.

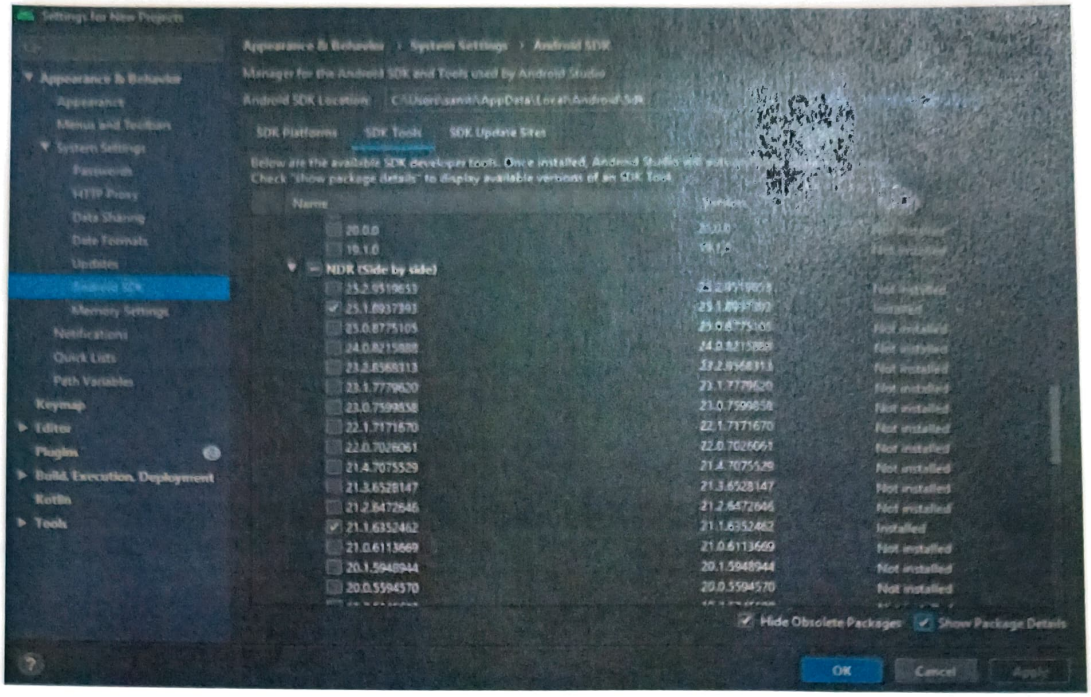
  Go to Android Studio -> Configure -> SDK Manager

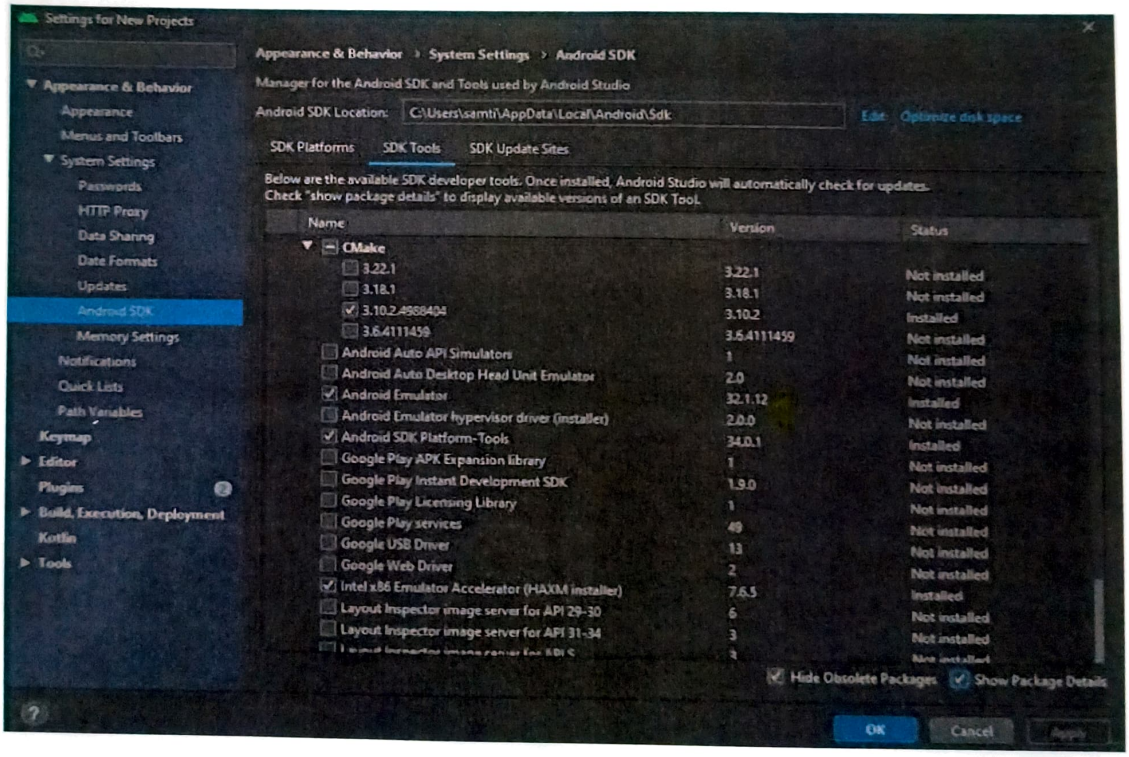
  ***Note*:** Make sure that when setting up Android Studio, you select the same options as we do or you may end up with an error.




Settings for New Projects
Appearance & Behavior
Appearance
Menus and Toolbars
System Settings
Passwords
HTTP Proxy
Data Sharing
Date Formats
Updates
Android SDK
Memory Settings
Notifications
Quick Lists
Path Variables
Keymap
Editor
Plugins
Build, Execution, Deployment
Kotlin
Tools
Appearance & Behavior › System Settings › Android SDK
Manager for the Android SDK and Tools used by Android Studio
Android SDK Location:
SDK Platforms
SDK Tools
SDK Update Sites
Below are the available SDK developer tools. Once installed, Android Studio will automatically check for updates.
Check "show package details" to display available versions of an SDK Tool.
Name
Android SDK Build-Tools 34-rc3
NDK (Side by side)
Android SDK Command-line Tools (latest)
CMake
Android Auto API Simulators
Android Auto Desktop Head Unit Emulator
Android Emulator
Android Emulator hypervisor driver (installer)
Android SDK Platform-Tools
Google Play APK Expansion library
Google Play Instant Development SDK
Google Play Licensing Library
Google Play services
Google USB Driver
Google Web Driver
Intel x86 Emulator Accelerator (HAXM installer)
Layout Inspector image server for API 29-30
Layout Inspector image server for API 31-34
Layout Inspector image server for API S
Hide Obsolete Packages
Show Package Details
OK
Cancel
Apply


Settings for New Projects
Appearance & Behavior
Appearance
Menus and Toolbars
System Settings
Passwords
HTTP Proxy
Data Sharing
Date Formats
Updates
Android SDK
Memory Settings
Notifications
Quick Lists
Path Variables
Keymap
Editor
Plugins
Build, Execution, Deployment
Kotlin
Tools
Appearance & Behavior › System Settings › Android SDK
Manager for the Android SDK and Tools used by Android Studio
Android SDK Location: C:\Users\samti\AppData\Local\Android\Sdk
Edit
Optimize disk space
SDK Platforms
SDK Tools
SDK Update Sites
Below are the available SDK developer tools. Once installed, Android Studio will automatically check for updates.
Check "show package details" to display available versions of an SDK Tool.
Name
Version
Status
Android SDK Build-Tools 34-rc3
34.0.0-rc3 34.0.0 rc3 Not installed
34.0.0-rc2 34.0.0 rc2 Installed
33.0.2 33.0.2 Installed
33.0.1 33.0.1 Not installed
33.0.0 33.0.0 Not installed
32.1.0-rc1 32.1.0 rc1 Not installed
32.0.0 32.0.0 Not installed
31.0.0 31.0.0 Not installed
30.0.3 30.0.3 Installed
30.0.2 30.0.2 Not installed
30.0.1 30.0.1 Not installed
30.0.0 30.0.0 Not installed
29.0.3 29.0.3 Not installed
29.0.2 29.0.2 Installed
29.0.1 29.0.1 Not installed
29.0.0 29.0.0 Not installed
28.0.3 28.0.3 Installed
28.0.2 28.0.2 Not installed
28.0.1 28.0.1 Not installed
Hide Obsolete Packages
Show Package Details
OK
Cancel
Apply





Once again, apply the necessary changes.

➢ Download JDK

The JDK is a development environment for building applications, applets, and components using the Java programming language. The JDK includes tools useful for developing and testing programs written in the Java programming language and running on the Java platform.

Before installing JDK, ensure that you have an *Oracle account*.

Later, download *JRE 20.0.1 Java Development Kit*. Alternatively, you can opt to download *JDK 8u77 Windows x64*.

After downloading the 64-bit installer, open it and take care of the formalities.

- **Visual Studio Download and Setup:**

The documentation for setting up Visual Studio for Unreal Engine is linked below:

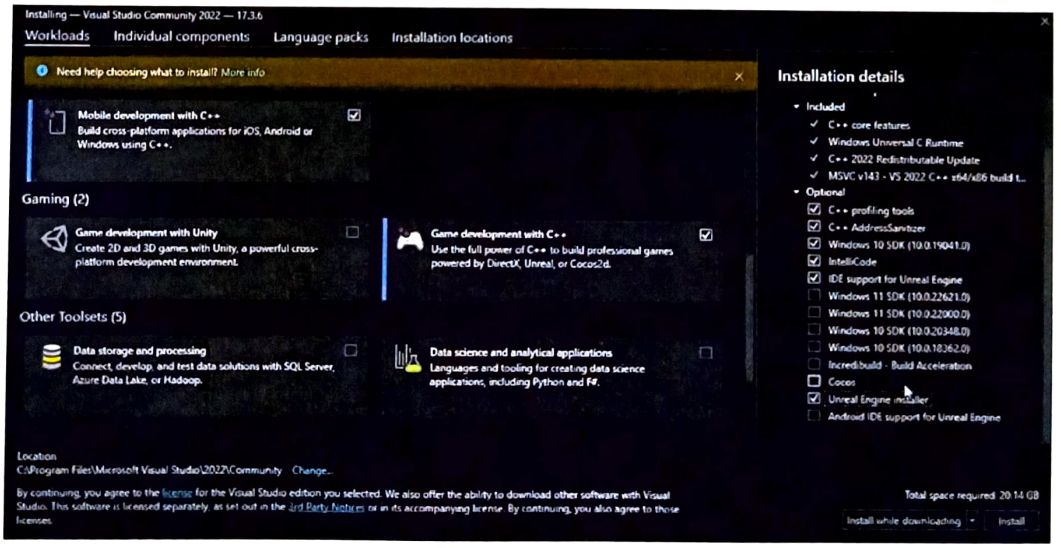
here

Visual Studio is an Integrated Development Environment (IDE) developed by Microsoft to develop GUI (Graphical User Interface), console, web applications, web apps, mobile apps, cloud, web services, etc.

Install the 2022 or preceding versions of Visual Studio. The link is given below:

here

***Note*:** Make sure that when setting up Visual Studio Code IDE, you select the same options as we do or you may end up with an error.



Click Install.

➢ Download Microsoft .NET Runtime:

To run console apps that use .NET 3.1 Core on your computer. Install the *.NET Core Runtime 64-bit installer*.

To run desktop apps that use .NET 3.1 Desktop Core on your computer. Install the *.NET Desktop Core Runtime x64 installer*.

Download the *.NET 5.0 Runtime - Windows x64 installer*. Once you install it, you don't have to do anything with this software. It runs in the background of your computer.

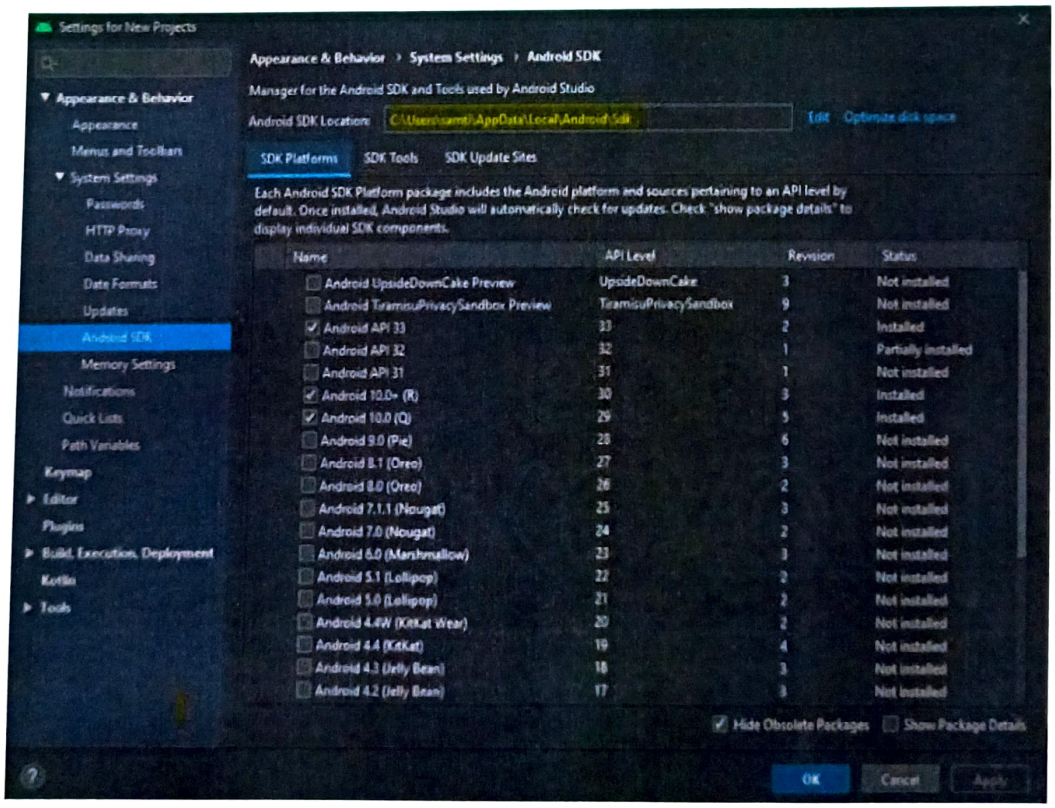
➢ Unreal Engine Setup:

Open a new AR project. Go to Project Settings -> Platforms -> Android SDK.

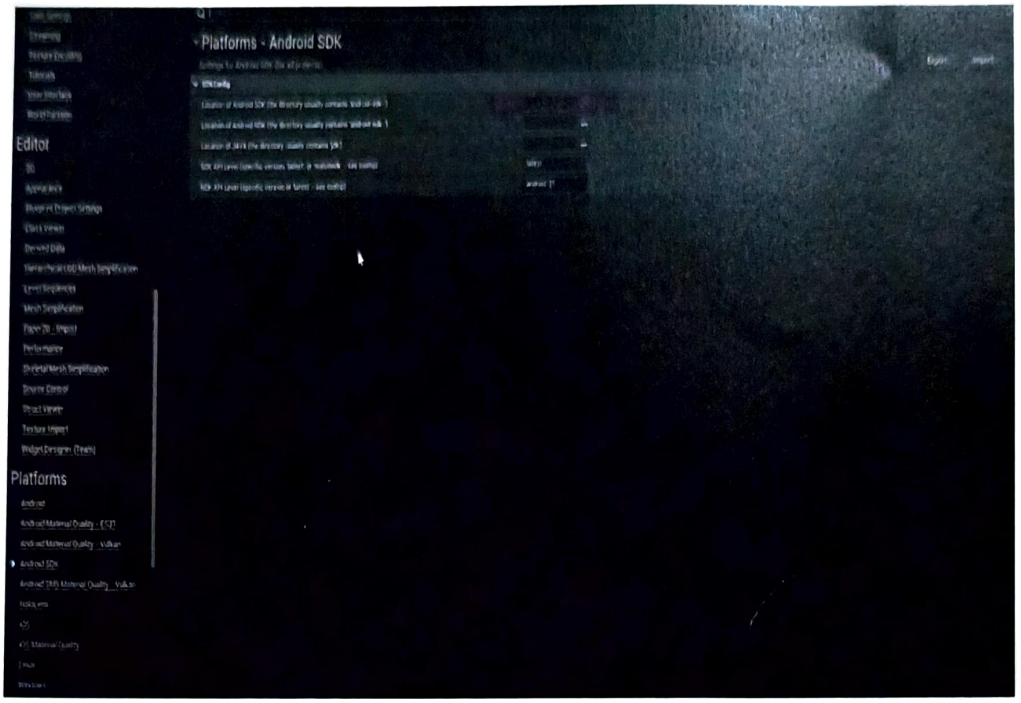
Follow the given steps:

Android Studio -> Configure ->SDK Manager.

Copy Android SDK Location.

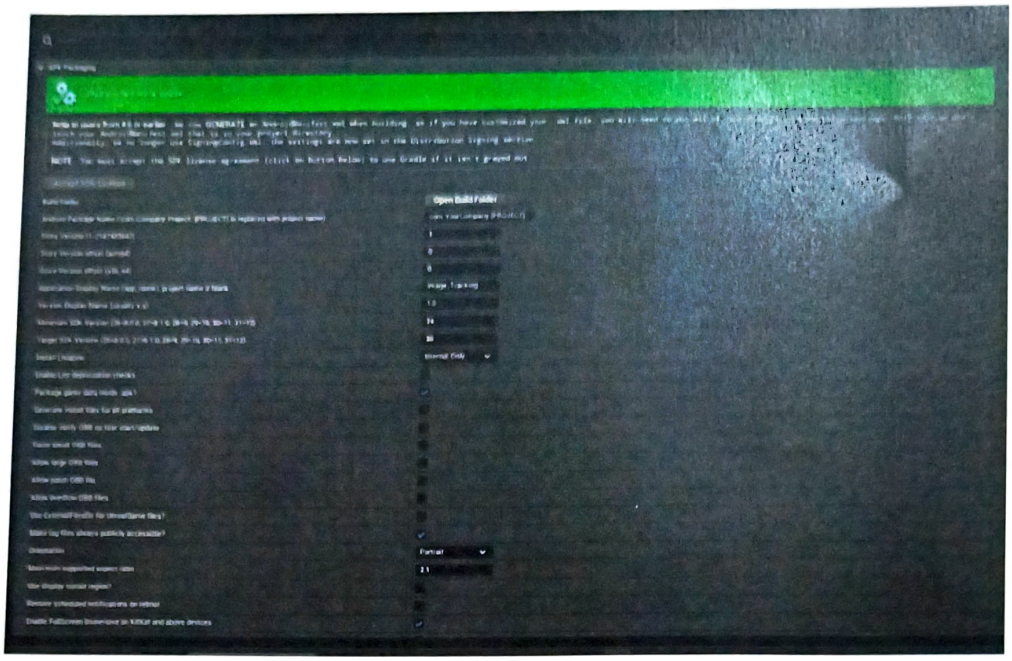


Paste the location here.

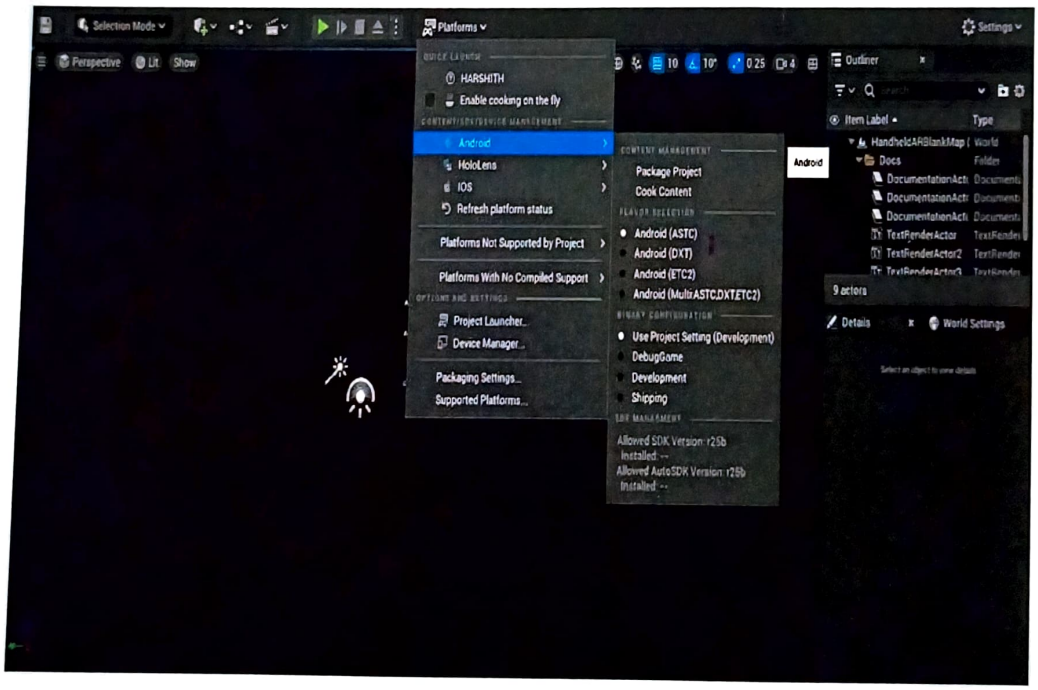


Follow the same process for Android NDK and JDK.

Now, go to Android under the Platforms section. Make the following changes:



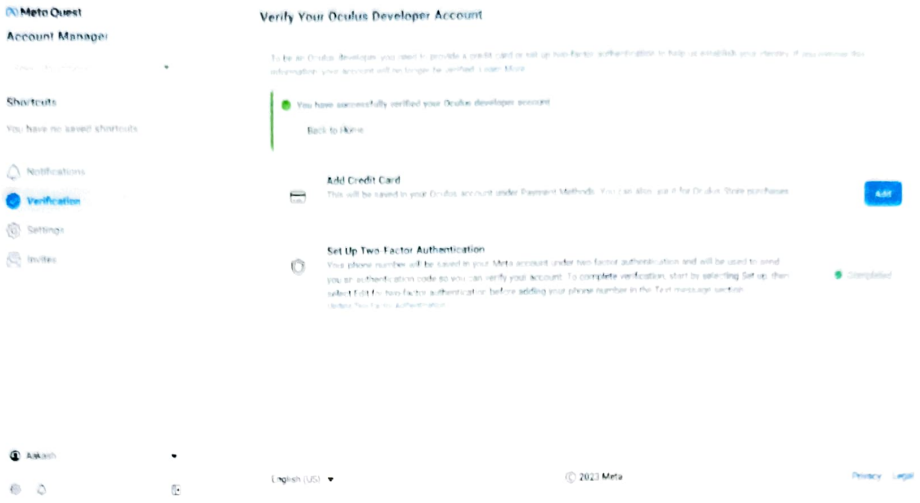
The final step is to package the project.

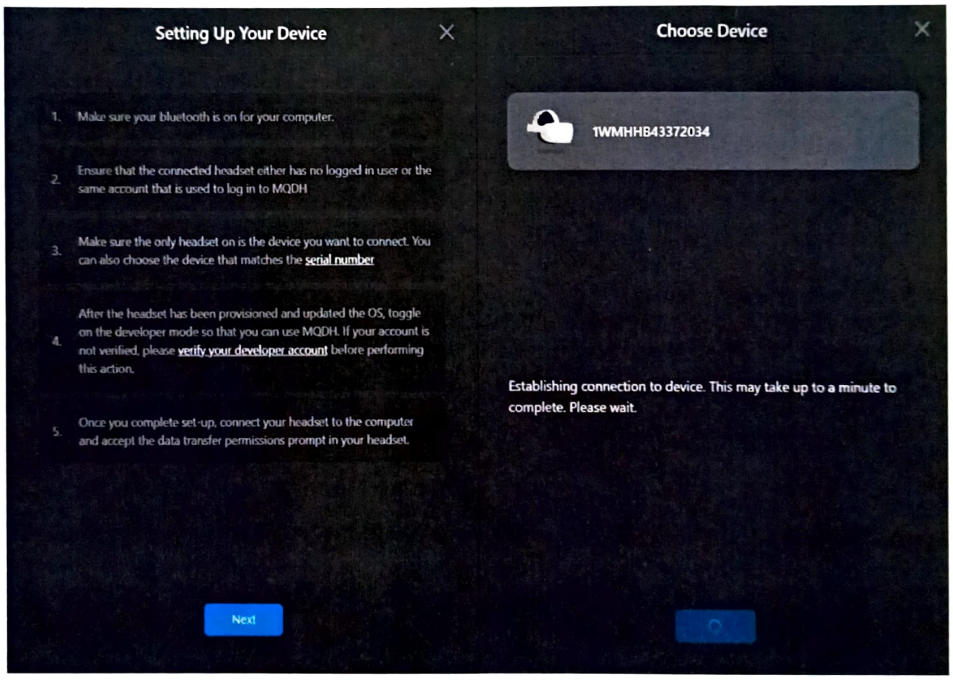


Copy the apk from the folder and transfer it to your Android device.

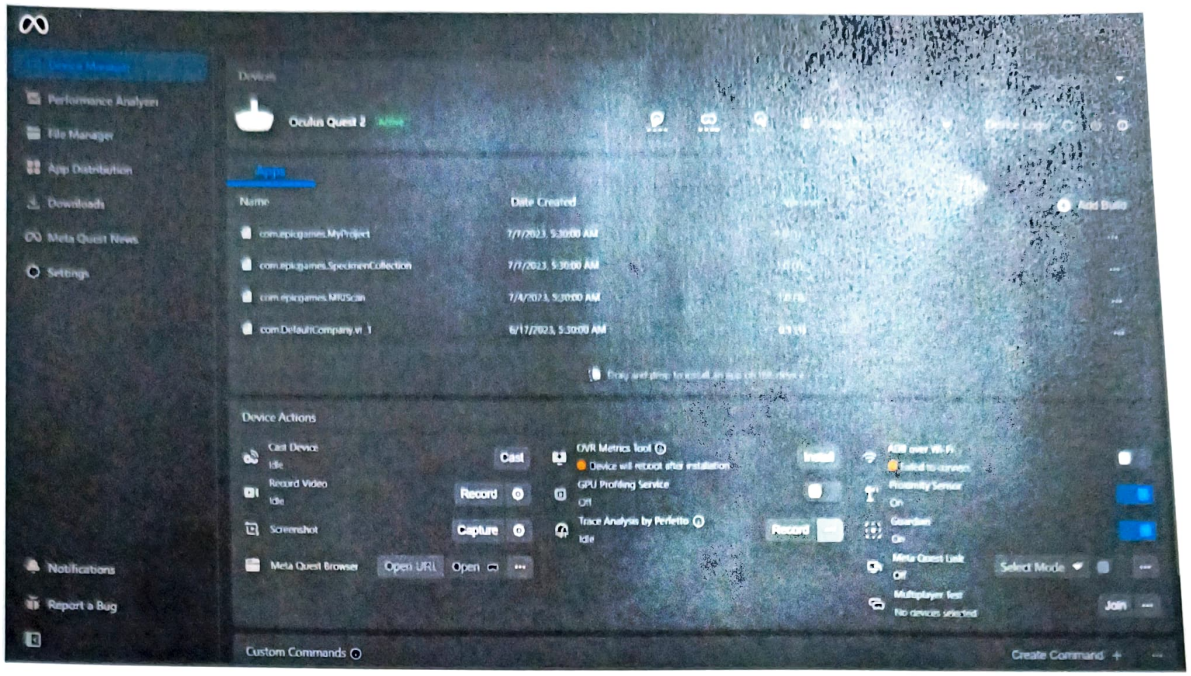
Setting up Oculus for VR project

Download the Meta Quest Developer Hub and create a meta-account. Then connect your oculus to your PC via USB cable. Make sure that your meta-account is a verified oculus development account by simply setting up two-factor Authentication.

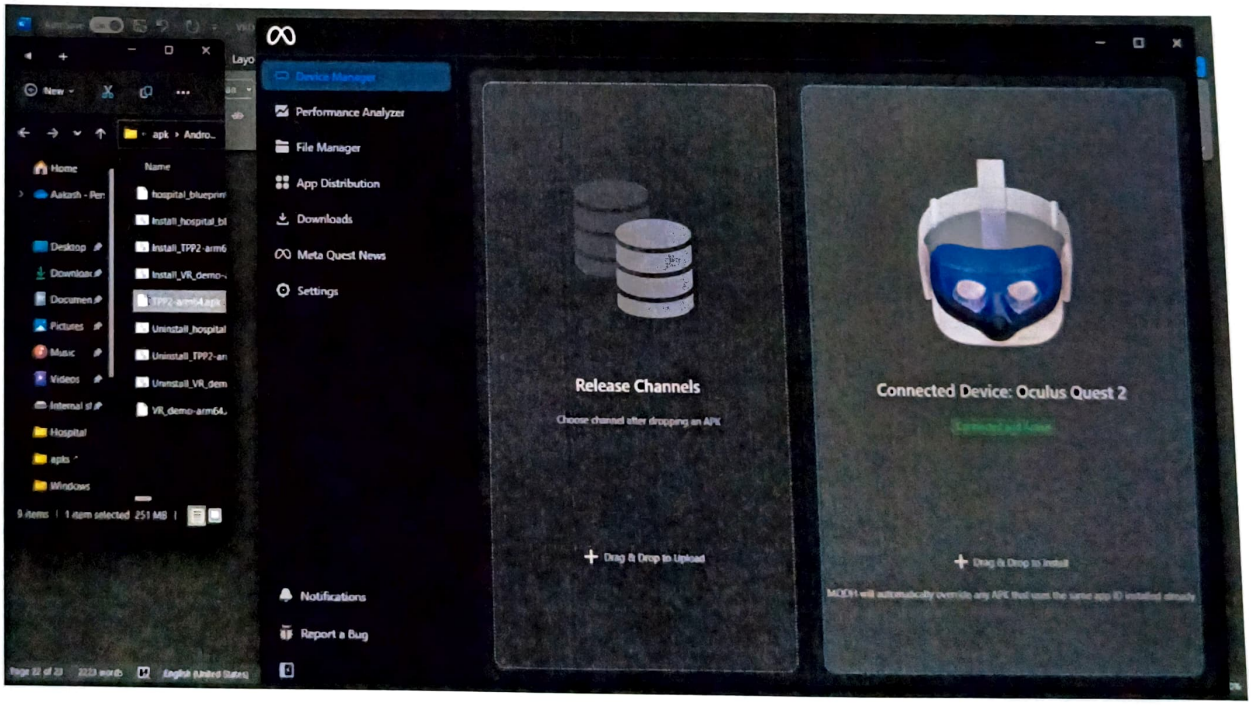




Choose your device and connect your Oculus to the Meta Quest Developer Hub.



Finally build your APK and simply drag the apk into the Meta Quest Developer Hub app. Once connected, users can immerse themselves in the VR experience through the Oculus device, interacting with the virtual environment as designed within the Unreal Engine project.



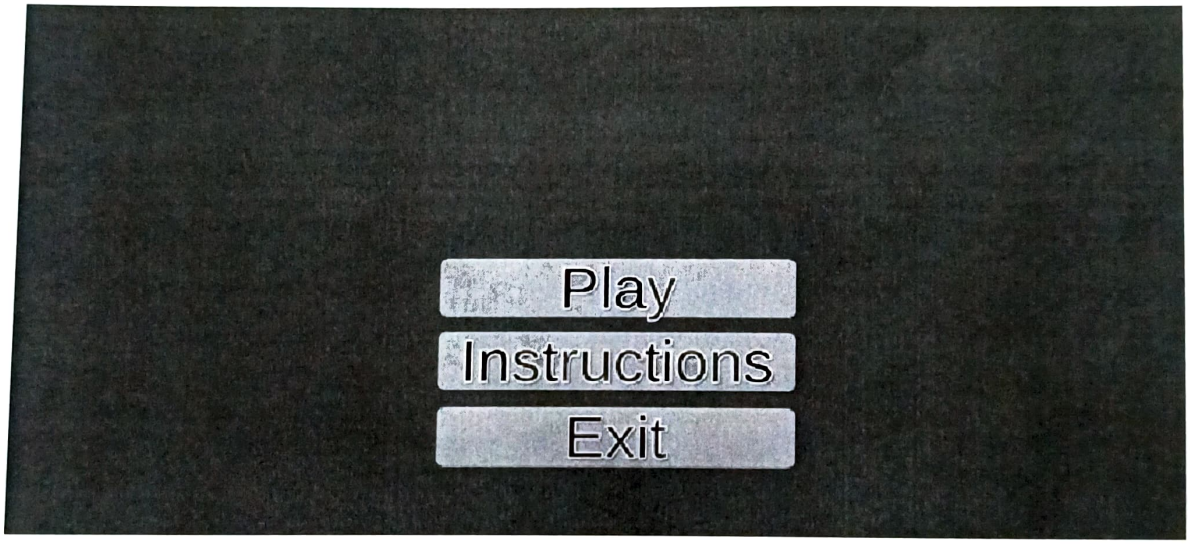
Tasks/Projects

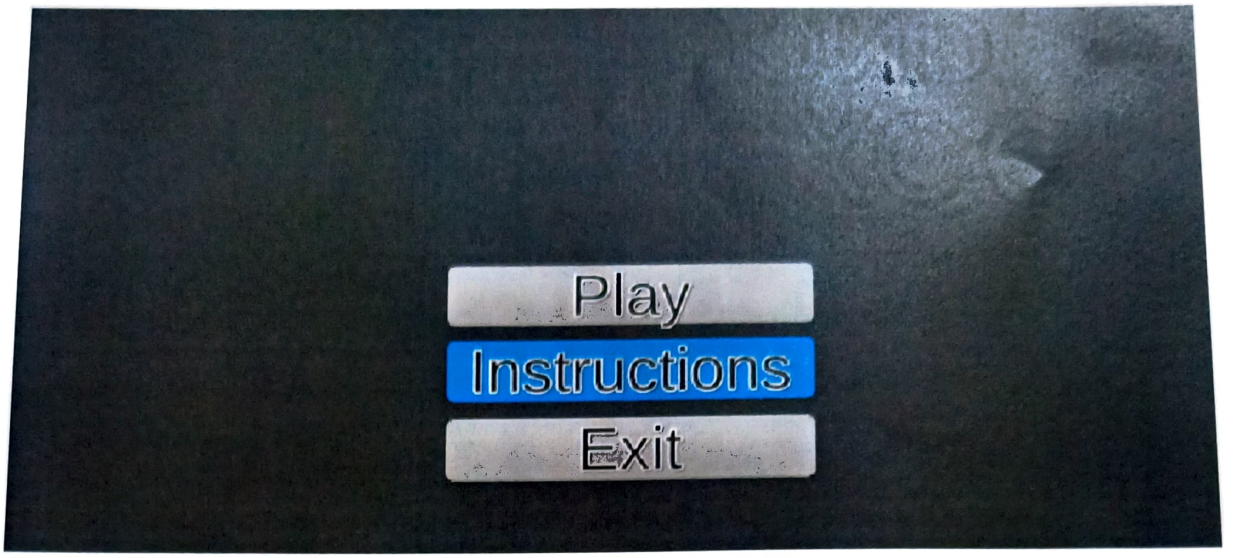
Unity

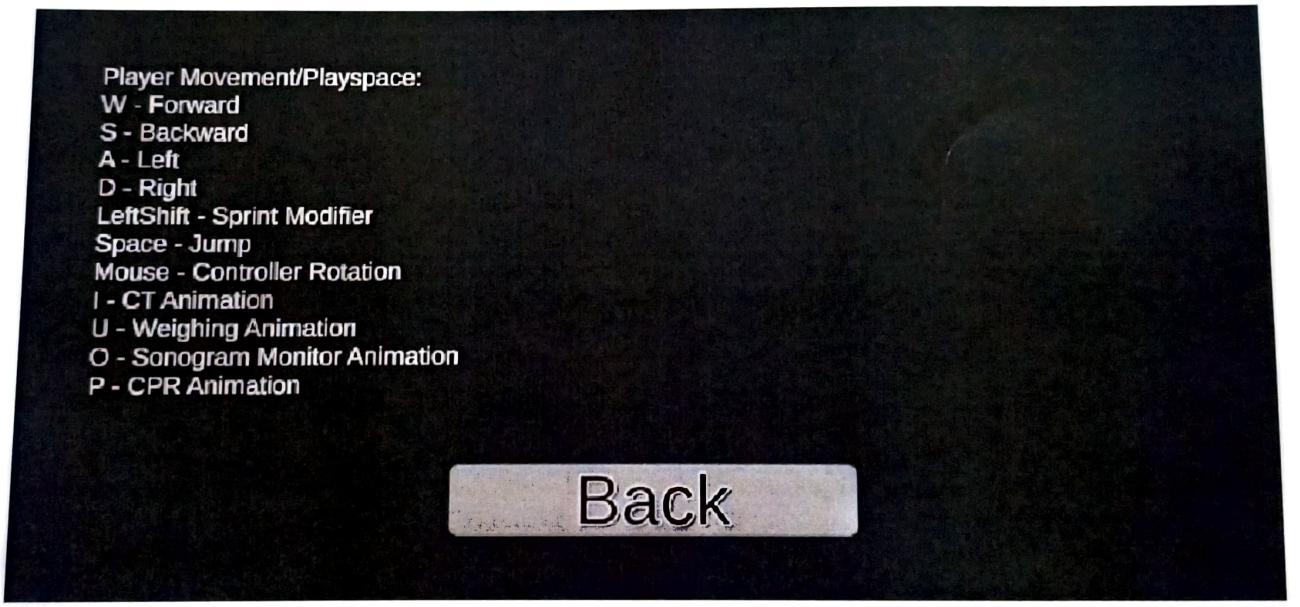
First-Person Perspective Based Model

This application was built using Unity, scripted using C#, with the idea of compiling all the medical procedures by placing them in a hospital environment under the same roof. This task features a reception, operation theatre, general ward, etc which were either built from the ground up in Blender or extracted from the internet. The rigged assets are imported from Mixamo, along with their animations; few of which are animated in Blender itself. The animations are triggered after pressing the key which is assigned to that animation using scripts.

The UI:



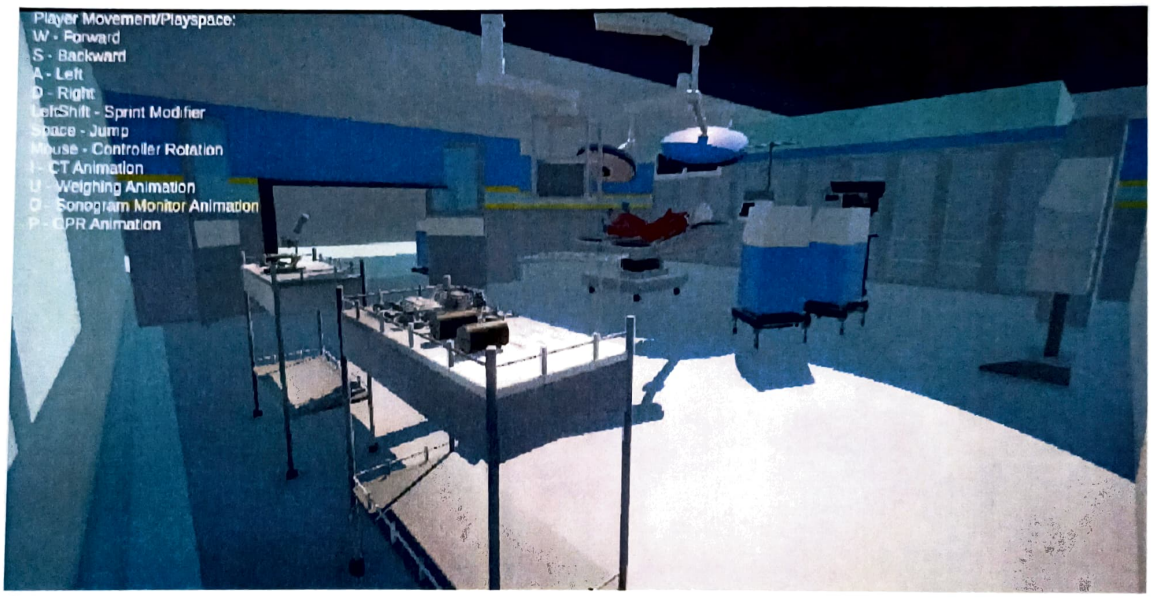

Play
Instructions
Exit

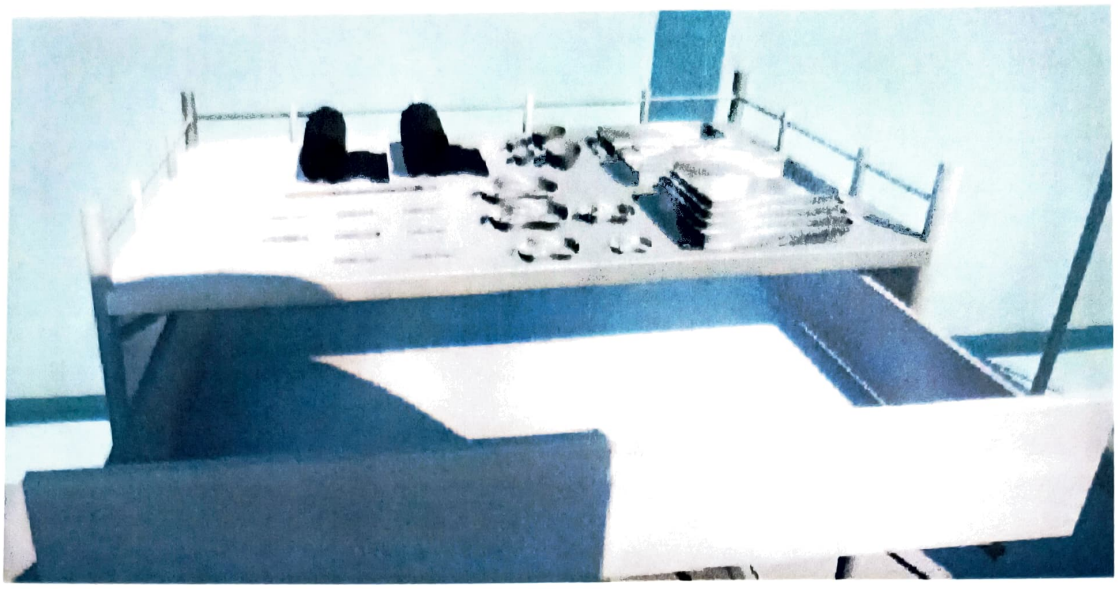

Player Movement/Playspace:
W - Forward
S - Backward
A - Left
D - Right
LeftShift - Sprint Modifier
Space - Jump
Mouse - Controller Rotation
I - CT Animation
U - Weighing Animation
O - Sonogram Monitor Animation
P - CPR Animation
Back

The exterior:

The reception:

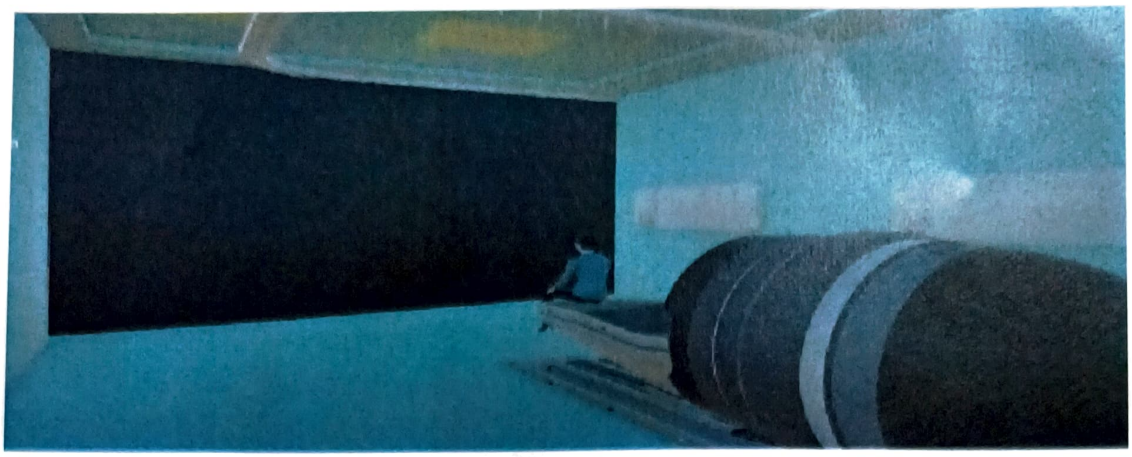
Operation Theatre:

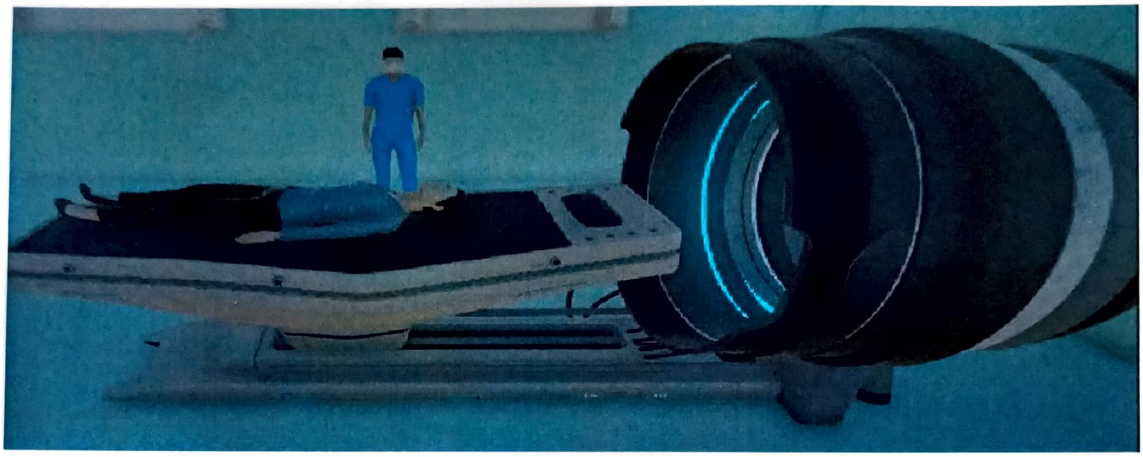


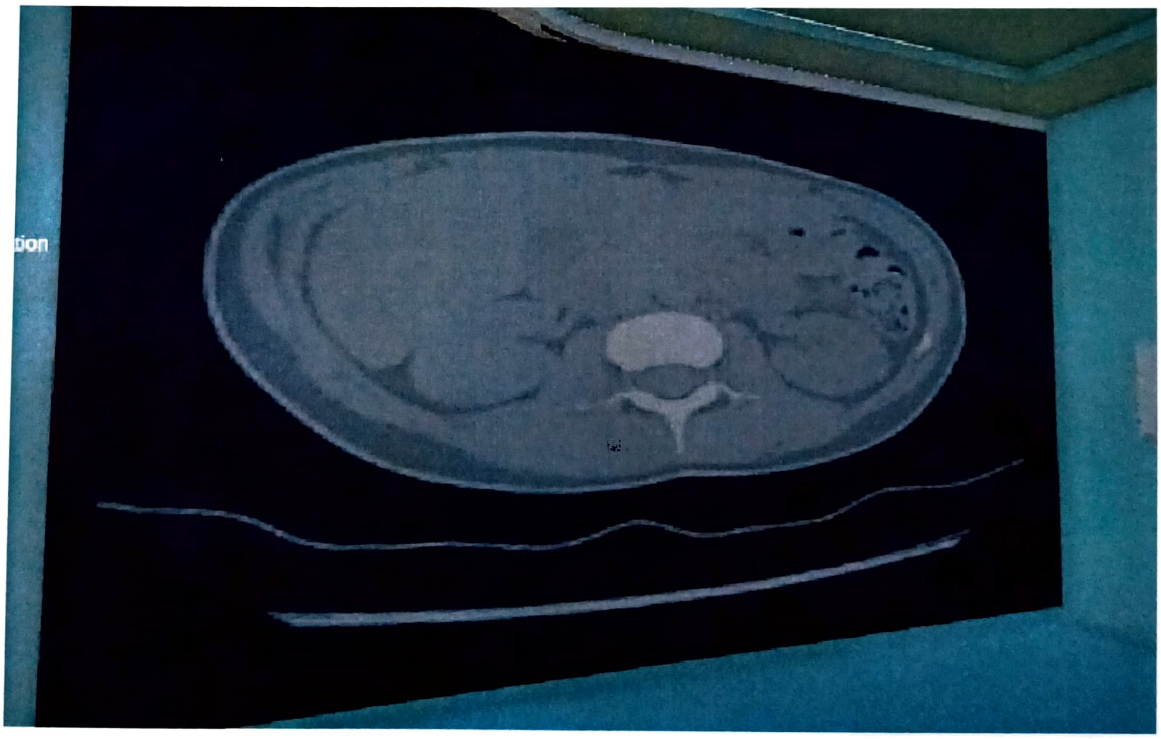


In the operation theatre, we have multiple interactive assets. Using the virtual mouse pointer, we can grab and hold the syringe. After moving the syringe close to the human body, the animation gets triggered. We have applied a box collider to the human body which upon interacting with the syringe triggers an animation, which make the plunger move.

CT scan:

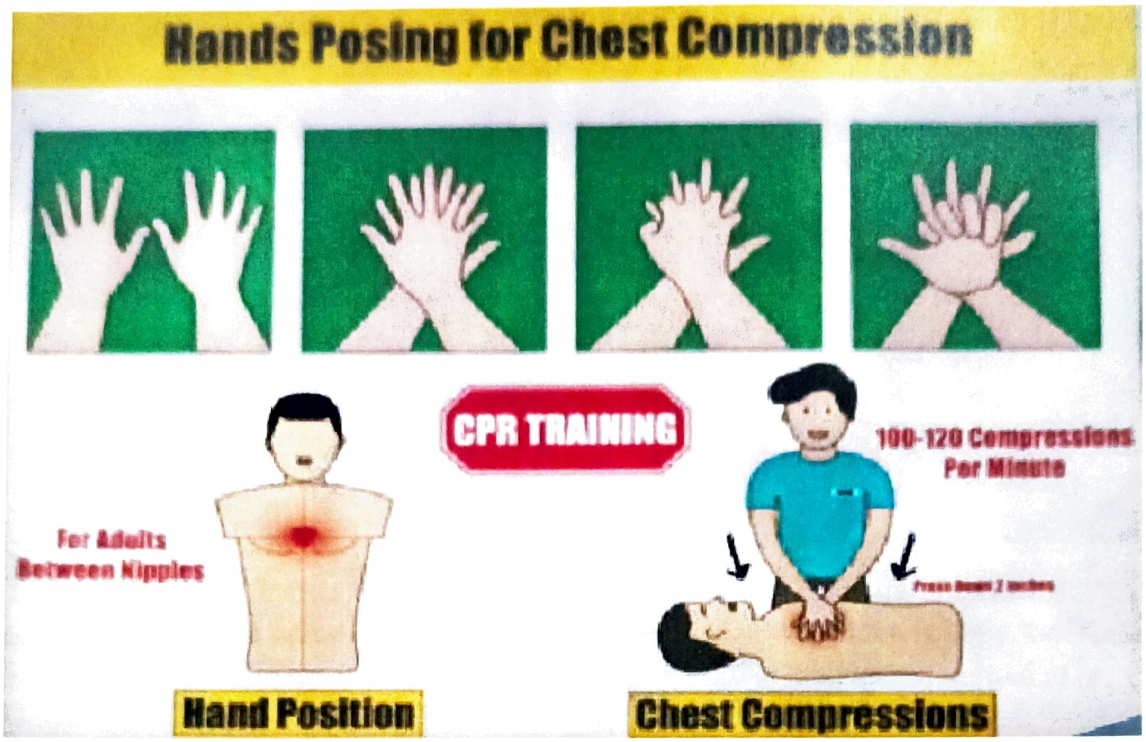


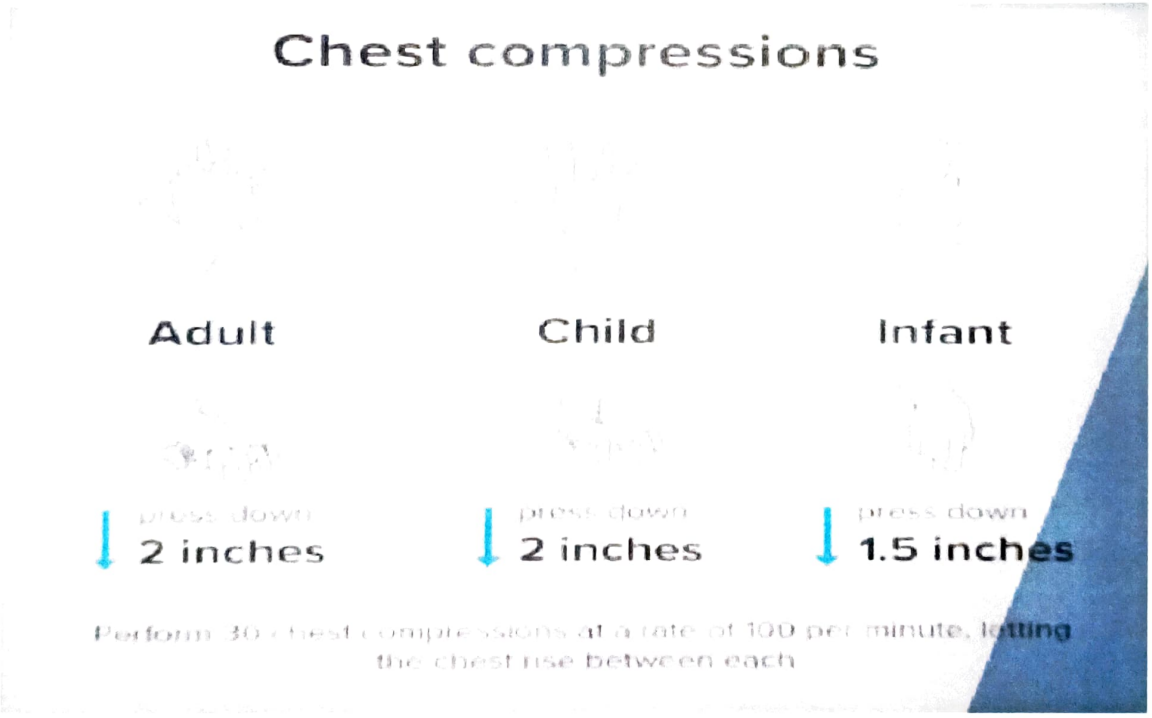




The CT scan model is extracted from the internet. The animated character is placed under the ct_bed gameobject as its subset. Upon pressing I, it triggers the CT Scan animation and a video is played on the screen.

CPR:



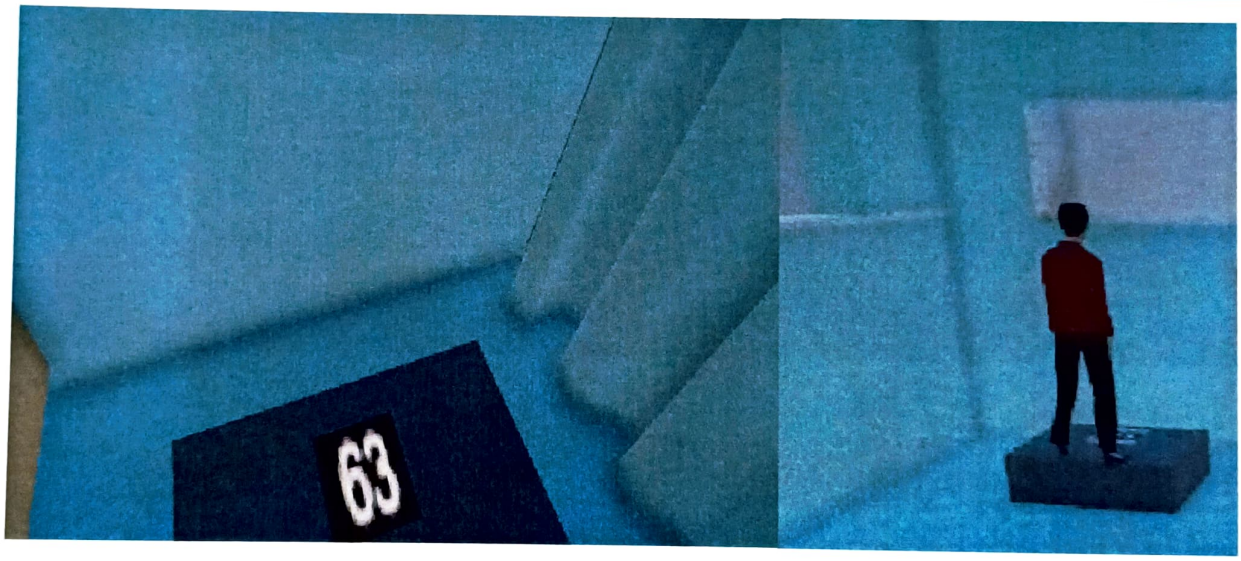


This room mainly focuses on providing the user a visual aid as to how a cpr is administered. Upon pressing P, the CPR animation is triggered which is imported from Mixamo. This animation is paired with a running animation, and later imported to Unity.

Sonogram:

The operation of the ultrasound animation is more or less the same as the real-world application of it. Additionally, upon pressing, a quad is used as a monitor to play the sonogram video which we imported from the internet.

Weighing machine animation:



There are two different types of weighing machine animations in this project – the first one is triggered by pressing U on the keyboard, whereas the other involves a more interactive approach. A box collider is set on the weighing machine, which gets

triggered when the player capsule enters the collider. This generates a number on the monitor.

Thermometer animation:

Thermometer animation is built in a hospital ward. This is an interactive task where the user is supposed to grab and hold a thermometer and move it close to the body to measure the temperature. A box collider is placed on the body. Upon entering the collider, a number is generated randomly depicting the temperature of the actor. If the number generated is greater than 99, the person is said to have fever.

Code Snippets:

First-Person Controller

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

[RequireComponent(typeof(CharacterController))]
public class FPSController : MonoBehaviour
{
    public Camera playerCamera;
    public float walkSpeed = 6f;
    public float runSpeed = 12f;
    public float jumpPower = 7f;
    public float gravity = 10f;


    public float lookSpeed = 2f;
    public float lookXLimit = 45f;


    Vector3 moveDirection = Vector3.zero;
    float rotationX = 0;

    public bool canMove = true;


    CharacterController characterController;
    void Start()
    {
        characterController = GetComponent<CharacterController>();
        Cursor.lockState = CursorLockMode.Locked;
```

```
        Cursor.visible = false;
    }

    void Update()
    {

        #region Handles Movment
        Vector3 forward = transform.TransformDirection(Vector3.forward);
        Vector3 right = transform.TransformDirection(Vector3.right);

        // Press Left Shift to run
        bool isRunning = Input.GetKey(KeyCode.LeftShift);
        float curSpeedX = canMove ? (isRunning ? runSpeed : walkSpeed) * Input.GetAxis("Vertical") : 0;
        float curSpeedY = canMove ? (isRunning ? runSpeed : walkSpeed) * Input.GetAxis("Horizontal") : 0;
        float movementDirectionY = moveDirection.y;
        moveDirection = (forward * curSpeedX) + (right * curSpeedY);

        #endregion

        #region Handles Jumping
        if (Input.GetButton("Jump") && canMove && characterController.isGrounded)
        {
            moveDirection.y = jumpPower;
        }
        else
        {
            moveDirection.y = movementDirectionY;
        }

        if (!characterController.isGrounded)
        {
            moveDirection.y -= gravity * Time.deltaTime;
        }

        #endregion

        #region Handles Rotation
        characterController.Move(moveDirection * Time.deltaTime);

        if (canMove)
        {
            rotationX += -Input.GetAxis("Mouse Y") * lookSpeed;
            rotationX = Mathf.Clamp(rotationX, -lookXLimit, lookXLimit);
            playerCamera.transform.localRotation = Quaternion.Euler(rotationX, 0, 0);
            transform.rotation *= Quaternion.Euler(0, Input.GetAxis("Mouse X") * lookSpeed, 0);
        }

        #endregion
    }
}
```

Interactive Weighing Machine

```
        if (!characterController.isGrounded)
        {
            moveDirection.y -= gravity * Time.deltaTime;
        }

        #endregion

        #region Handles Rotation
        characterController.Move(moveDirection * Time.deltaTime);

        if (canMove)
        {
            rotationX += -Input.GetAxis("Mouse Y") * lookSpeed;
            rotationX = Mathf.Clamp(rotationX, -lookXLimit, lookXLimit);
            playerCamera.transform.localRotation = Quaternion.Euler(rotationX, 0, 0);
            transform.rotation *= Quaternion.Euler(0, Input.GetAxis("Mouse X") * lookSpeed, 0);
        }

        #endregion
    }
}
```

Drag and Drop

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class DragnDrop : MonoBehaviour
{
    Vector3 mousePosition;

    private Vector3 GetMousePos()
    {
        return Camera.main.WorldToScreenPoint(transform.position);
    }
    private void OnMouseDown()
    {
        mousePosition= Input.mousePosition-GetMousePos();
    }
    private void OnMouseDrag()
    {
        transform.position=Camera.main.ScreenToWorldPoint(Input.mousePosition-mousePosition);
    }
}

```

Main Menu

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement;

public class MainMenu : MonoBehaviour
{
    public void playGame()
    {
        SceneManager.LoadScene(SceneManager.GetActiveScene().buildIndex+1);
    }
    public void GoToInstrumentsMenu()
    {
        SceneManager.LoadScene("SettingsMenu");
    }
    public void GoToMainMenu()
    {
        SceneManager.LoadScene("MainMenu");
    }
    public void quitGame()
    {
        Application.Quit();
    }
}

```

Unreal

Augmented Reality

CPR animation

This task features a consultation room which was extracted from sketchfab. The CPR animation used in this task was built in blender by combining two different animations – running and administering CPR which were extracted from mixamo.

The above image is used as the candidate image, which upon scanning triggers the AR Session. The following images are taken directly from the mobile application.

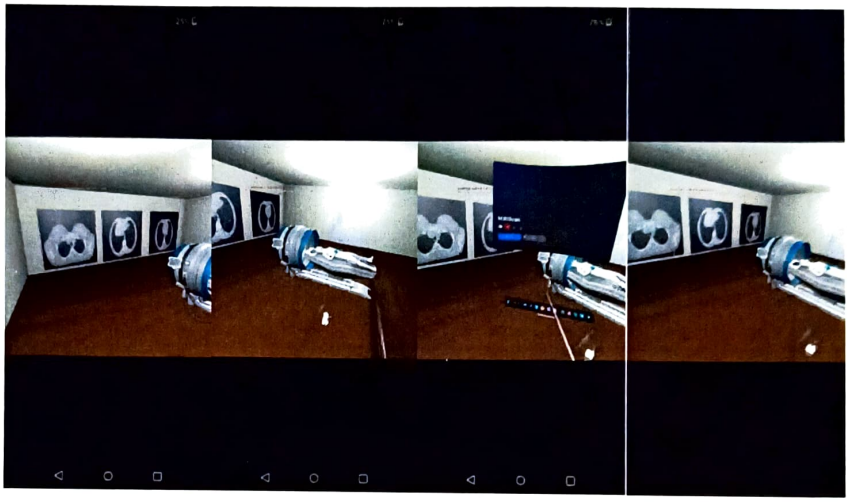
Instrument Selection

The objective of this task is to provide the user the opportunity to choose their instrument of choice between a syringe, mask and dropper. The 3d models were extracted from the internet but the textures and materials used were built in blender. The cover pages of three different books were used as triggers to spawn three different buttons. These pages form the candidate images of this AR session. Upon bringing the

phone camera over these books, the buttons along with their respective instruments are spawned. On button click, the instruments pop up.

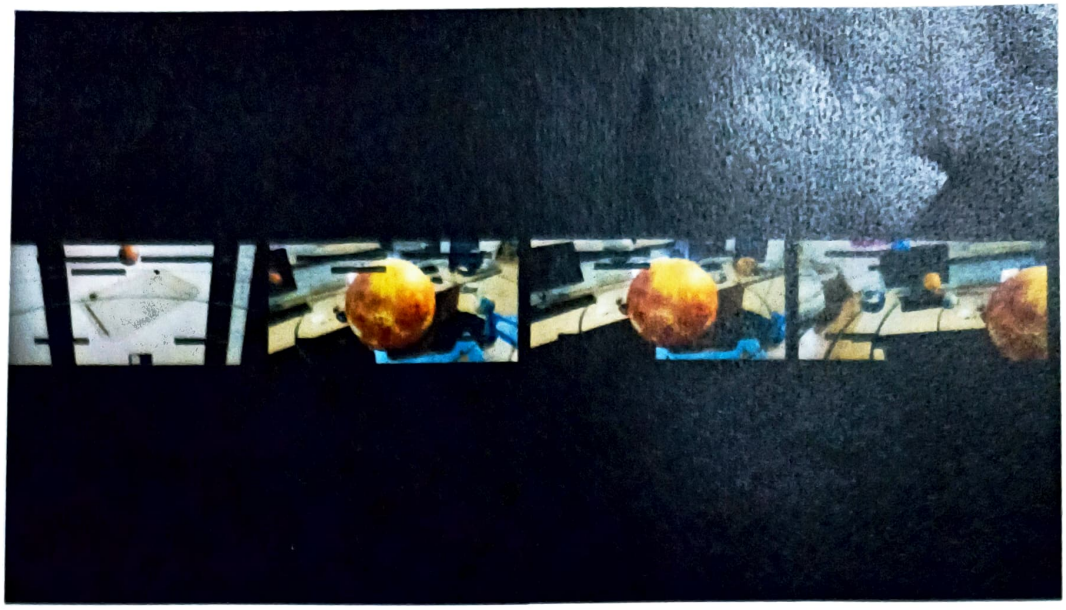
Virtual Reality

This task features a room in which a CT scan animation is placed in the centre of it. Initially a video was played in the background after the animation got triggered but due to an unknown reason, the animation was not getting baked after running the application in an oculus. So, we decided to replace the video with images of the CT scan. We ran this application, after packaging it, in an oculus. Few snippets of the recording are posted below.



Mini-Project

The idea behind this mini-project was to experiment with as many features as I can so that I familiarise myself with Unreal before working on other medical related projects. This is an AR project which is a replica of our solar system. The UI of this application displays the names of the planets revolving around the sun. The application displays the name of the planet on the UI upon a click input. After removing the input, the name disappears from the UI. The planetary revolution and rotation of each planet is different from the other. Asteroids and other comets, which are animated in blender are placed in between these planets to make sure that the model isn't too static in its nature and makes for a better viewing experience. The orbit along which the planets revolve was developed in Unreal Editor. The textures applied on the planets are imported from the NASA website. An image is set as a candidate image which basically triggers the AR Session upon scanning the it.

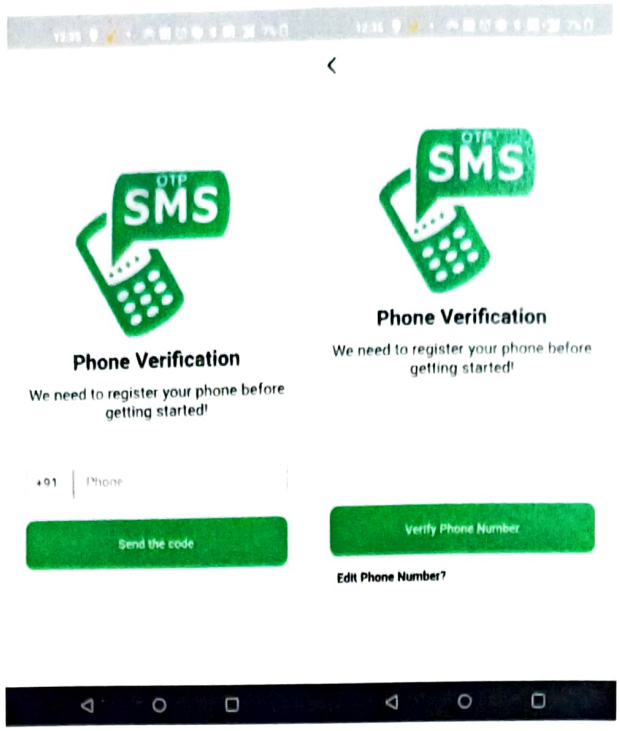


Flutter Application

This is a mobile application whose frontend is on Flutter and backend is on Firebase.

Phone Authentication:

It enables users to sign in to the mobile application using their email and password or phone number and SMS verification. Firebase Authentication manages user accounts, handles secure password storage, and simplifies the authentication process.



Home Page:

After the verification process, the app will direct us to the home page.

Summary

In conclusion, the AR/VR medical project holds tremendous potential in revolutionizing healthcare practices and patient experiences. Through the use of virtual reality technology, the project aims to address various challenges and enhance several aspects of medical care.

Firstly, the project offers immersive and realistic simulations that enable healthcare professionals to refine their skills and knowledge in a safe and controlled environment. AR/VR allows for realistic medical scenarios and procedures to be recreated, providing an invaluable training tool for doctors, nurses, and other medical personnel. This can result in improved clinical outcomes, reduced medical errors, and increased confidence among healthcare providers.

Moreover, AR/VR technology has the potential to transform patient experiences and outcomes. It can be utilized for pain management, anxiety reduction, and distraction during medical procedures. By creating immersive and engaging environments, AR/VR can alleviate patient stress, enhance their comfort, and contribute to better overall well-being. Additionally, AR/VR can facilitate remote consultations, allowing patients to receive expert medical advice and support from the comfort of their homes, thereby improving access to healthcare services.

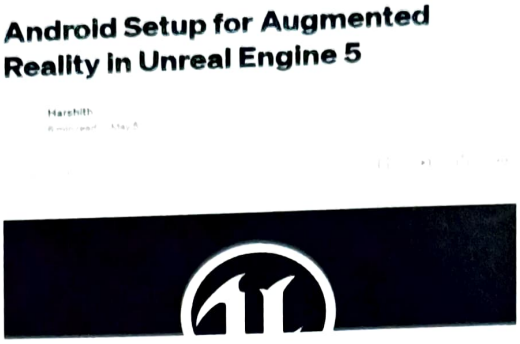
Furthermore, the AR/VR medical project has the potential to improve medical education by providing an interactive and engaging learning experience. Medical students can benefit from immersive anatomy lessons, surgical simulations, and collaborative problem-solving scenarios. This can enhance their understanding, retention, and application of medical knowledge.

While technology has become a key part of the classroom, there hasn't yet been widespread adoption of AR/VR in the classroom. Integrating AR/VR into the classroom can enhance lessons and enable students to learn more about different applications of media, the process of content creation and development, as well as presentation skills. The future of the classroom can change for the best, with the incorporation of AR/VR into the teaching methodology.

In conclusion, the AR/VR medical project represents a significant advancement in healthcare technology. By leveraging the power of virtual reality, it has the potential to transform medical training, patient care, research, and education. As the technology continues to evolve, it is crucial to ensure its widespread adoption and integration into healthcare systems to unlock its full potential and improve the overall quality of medical care.

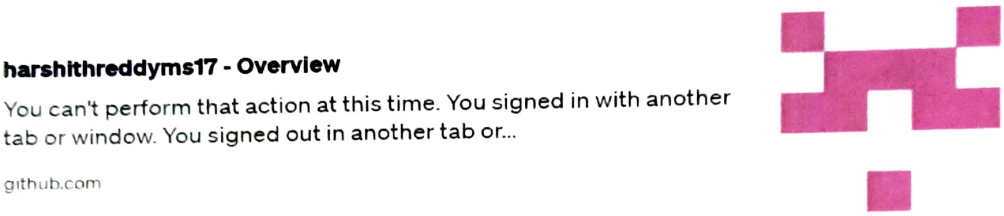
Contribution

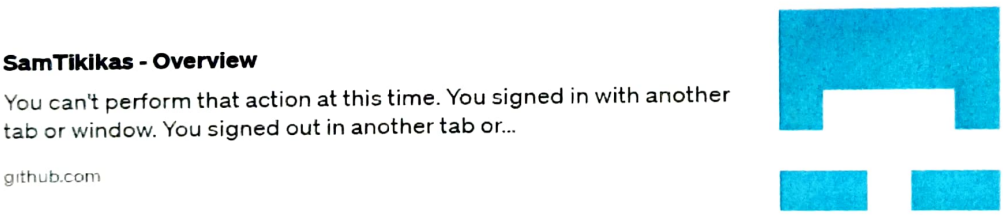
Article published on Medium: https://medium.com/@harshithreddyms17/android-setup-for-augmented-reality-in-unreal-engine-5-9b9f21ecae7e



M S Harshith Reddy and Sam Tikikas Gera

Our GitHub profiles:





The successful completion of this project is attributed to the guidance and supervision of Dr.Devikarubi Rajasekaran (Associate Professor)from Keshav Memorial Institute of Technology.

References

1. Adobe Systems Incorporated. (2021). Mixamo. Adobe Inc. https://www.mixamo.com/
2. Unity Technologies. (2021). AR Overview. Unity Documentation. https://docs.unity3d.com/Manual/AROverview.html
3. Epic Games, Inc. (2021). AR Overview. Unreal Engine Documentation. https://docs.unrealengine.com/4.26/en-US/SharingAndReleasing/XRDevelopment/AR/HandheldAR/AROverview/
4. EC_UnrealEngineTutorial. [YouTube channel]. (2021, September 15). Introduction to Unreal Engine. YouTube. https://www.youtube.com/watch?v=ABC123
5. Unity Technologies. (2021). VR Overview. Unity Documentation. Retrieved from https://docs.unity3d.com/Manual/VROverview.html
6. Flutter. (2021). Get Started: Install on Windows. Flutter Documentation. Retrieved from https://docs.flutter.dev/get-started/install/windows