UNIVERSITY OF LEICESTER

Department of Informatics
University of Leicester
CO7201 Individual Project

Final Report

Interactive Augmented Reality Platform for Medical Education

Gopinath Varadarajan

gv58@student.le.ac.uk

Student ID: 239049127

Project Supervisor: Dr Zedong Zheng
Second Marker: Prof Shigang Yue

Word Count: 11884
Date: 06/09/2024

DECLARATION

All sentences or passages quoted in this report, or computer code of any form whatsoever used and/or submitted at any stages, which are taken from other people's work have been specifically acknowledged by clear citation of the source, specifying author, work, date and page(s). Any part of my own written work, or software coding, which is substantially based upon other people's work, is duly accompanied by clear citation of the source, specifying author, work, date and page(s). I understand that failure to do this amounts to plagiarism and will be considered grounds for failure in this module and the degree examination as a whole.

Name: Gopinath Varadarajan

Date: 06/09/2024

Abstract

In the rapidly evolving field of medical education, the ability to visualize and interact with complex anatomical structures is crucial for deep understanding and effective learning. My project, the "Interactive Augmented Reality Platform for Medical Education," harnesses the power of Augmented Reality (AR) to bring these complex concepts to life. By integrating advanced AR technologies, such as Vuforia and Unity, I have developed a mobile application that allows students to explore detailed 3D models of human anatomy in real-time, directly from their devices.

The application offers both marker-based and markerless AR experiences, enabling students to view and manipulate 3D models in various environments whether it's in a classroom, a lab, or even at home. This flexibility not only enhances engagement but also allows for a more immersive learning experience. The app is designed with the user in mind, focusing on intuitive controls and seamless performance across both iOS and Android platforms. This platform represents a significant step forward in medical education, offering a modern tool that complements traditional learning methods. It has the potential to revolutionize how students interact with educational content, making learning more engaging, accessible, and effective. Through this application, the aim is to bridge the gap between theoretical knowledge and practical application, ultimately contributing to the training of more skilled and confident medical professionals.

Table of Contents

1. Introduction

1.1 Aim

The primary aim of this project is to enhance medical education by integrating augmented reality (AR) into learning environments. Traditional teaching methods, such as static images and physical models, often limit students' ability to fully grasp the complexities of human anatomy and medical procedures. By introducing dynamic and interactive 3D simulations, this project seeks to revolutionize how medical concepts are taught, offering a more immersive and engaging learning experience that goes beyond conventional approaches.

Another key aim of the project is to improve students' understanding and interaction with complex anatomical structures and medical procedures. The platform leverages interactive 3D models and simulations to provide a deeper comprehension of these subjects, enabling students to visualize and manipulate anatomical structures in real-time. This hands-on approach not only aids in the retention of information but also fosters a more intuitive understanding of intricate medical concepts, which is crucial for effective learning and application in real-world scenarios.

Lastly, the project aims to increase the accessibility of medical education through the use of AR technology. By removing the need for physical presence in a lab, this technology allows students to explore medical concepts from virtually anywhere, at any time. The incorporation of audio and virtual video further enriches the learning experience, making it more engaging and accessible to a wider audience. This flexibility in learning is particularly beneficial for students who may face geographical, financial, or logistical barriers to accessing traditional educational resources.

1.2 Objectives

Developing Accurate 3D Models: A key objective is to create highly detailed and anatomically accurate 3D models for visualization in augmented reality. These models are central to the educational experience, requiring precision to reflect the complexities of human anatomy.

Creating an Interactive Interface: The project aims to implement a user-friendly interface that allows students to easily manipulate and explore 3D models. By enabling actions like zooming, rotating, and viewing structures from different angles, the interface helps students gain a deeper understanding of complex anatomical details. The design focuses on being intuitive, ensuring that all users can effectively engage with the app.

Ensure Device Compatibility: Another objective is to ensure the application is compatible across a wide range of mobile devices, including iOS and Android. This involves optimizing the app to perform well on various devices, from high-end to lower-spec, while maintaining a consistent user experience. Broad compatibility ensures that the platform is accessible to a diverse audience, making AR-based medical education widely available.

1.3 Challenges

- Accurate 3D Modeling: Developing precise 3D models that accurately represent human anatomy.
- Device Compatibility: Ensuring the application functions smoothly on various devices with different performance capabilities.
- User Interface: Creating an intuitive and user-friendly interface that enhances the learning experience without overwhelming the user.
- Learning Unity Software: Understanding the complete workings of Unity software, particularly in relation to augmented reality.
- Learning C# Code: Gaining knowledge of C#, especially how it is helpful in developing augmented reality apps.
- Understanding Vuforia Engine: Understanding the complete workings of the Vuforia developer engine to develop AR apps.

1.4 Risk

- Technical Challenges: Difficulty in creating highly detailed 3D models and ensuring smooth performance across all devices.
- Compatibility Issues: Ensuring the application is compatible with different operating systems and devices.
- User Acceptance: Ensuring the platform meets the educational needs of students and is easy to use.

2. Requirements

2.1 Essential

- 3D Model Visualization: Develop accurate and detailed 3D models of human organs that can be viewed in augmented reality. These models should be anatomically correct and provide a high level of detail to facilitate in-depth study.
- Interactive Manipulation: Enable users to manipulate the 3D models, including rotating, zooming, and exploring different layers. This functionality is crucial for understanding the relationships and complexities of human anatomy.
- Device Compatibility: Ensure that the platform is compatible with a range mobile devices, including smartphones and tablets running both iOS and Android operating systems. This broad compatibility is essential for accessibility and usability across different user groups.

2.2 Recommended

- Audio Integration: Incorporate audio content, such as narrated explanations or sound effects, to enhance the educational material and provide auditory support for the visual elements in the app.

- Video Integration: Include video content, such as demonstrations or supplementary tutorials, to further support the educational material and offer students a more comprehensive learning experience.

2.3 Optional

- User Feedback Integration: Allow users to provide feedback on the AR models and simulations, helping to refine and improve the educational content based on real user experiences.
- Interactive Quizzes: Add interactive quizzes or assessments within the app to test users' knowledge after exploring the 3D models. This feature would provide immediate feedback and reinforce learning outcomes.

3. Background Research

3.1 C# the programming language

C# (pronounced "C-sharp") is a robust, object-oriented programming language developed by Microsoft, designed to be both powerful and easy to use. It builds on the strengths of languages like C++ and Java, offering a more modern syntax and a rich set of features that make it particularly well-suited for developing complex applications across different platforms [4].

In the context of augmented reality (AR), C# stands out due to its strong support within Unity, a leading game engine used for AR development. Unity uses C# as its primary scripting language, enabling developers to create interactive 3D models and immersive AR experiences. For example, in AR applications, you might use C# to handle object interactions, like rotating a 3D anatomical model with a touch gesture [4].

```
1  void Update() {
2      if (Input.touchCount > 0) {
3          Touch touch = Input.GetTouch(0);
4          if (touch.phase == TouchPhase.Moved) {
5              transform.Rotate(0, -touch.deltaPosition.x * rotationSpeed, 0);
6          }
7      }
8  }
9
```

Code 3.1 C# Code Example 1

This snippet shows how C# can be used to detect touch input and rotate a 3D object accordingly. Such capabilities are crucial for AR applications, where intuitive and responsive interaction is key to enhancing the user experience. Overall, C#'s combination of performance, ease of use, and deep integration with Unity makes it an excellent choice for developing sophisticated AR applications.

3.2 Unity game engine

Unity is one of the most popular and versatile game engines used for developing a wide range of applications, including games, simulations, and augmented reality (AR) experiences. What makes Unity stand out is its user-friendly interface and powerful features that allow both beginners and experienced developers to create high-quality 2D and 3D content [7].

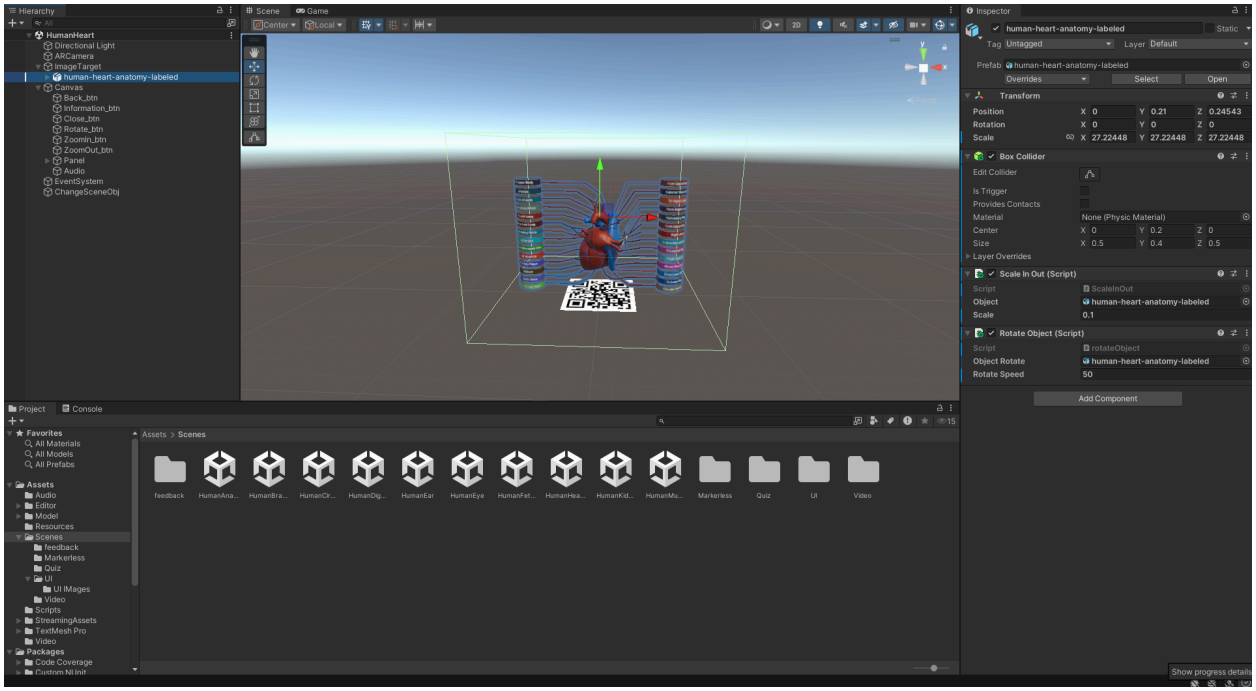


Figure 3.1 Screenshot of the Unity game engine in action

In the context of AR, Unity is particularly valuable because it provides an integrated development environment where you can build, test, and deploy AR applications. It supports a variety of platforms, including iOS and Android, so the apps you create can reach a broad audience. Unity's real strength lies in its vast library of assets and tools, which can help you quickly add complex features like physics, animations, and lighting to your AR projects. Unity also works seamlessly with AR-specific tools like Vuforia, making it easier to implement AR features such as object recognition and tracking [6].

3.3 What is Augmented Reality?

Augmented Reality (AR) is an advanced technology that superimposes digital information, including images, sounds, and interactive content, onto the physical world through devices such

as smartphones, tablets, or AR glasses. Unlike Virtual Reality (VR), which creates a completely immersive digital environment, AR enhances the real-world experience by adding digital elements that users can interact with in real-time. This technology enables a seamless blending of virtual objects with the real world, providing users with an enriched and interactive experience [1][8].

AR has gained significant attention across various industries, with education being one of the key areas where its impact is profoundly felt. In educational settings, AR transforms traditional learning methods by making the process more engaging and interactive. This interactive approach helps bridge the gap between theoretical knowledge and practical application, offering a more immersive learning experience that can significantly improve knowledge retention and understanding [1][8].

3.4 Augmented Reality Impact in Education

Augmented Reality (AR) is increasingly recognized for its transformative potential in education, offering immersive and interactive learning environments that bridge the gap between digital and physical worlds. By overlaying digital content onto real-world settings, AR enables students to engage with educational material in more dynamic and meaningful ways [9]. This technology has been shown to enhance student motivation, engagement, and overall learning outcomes.

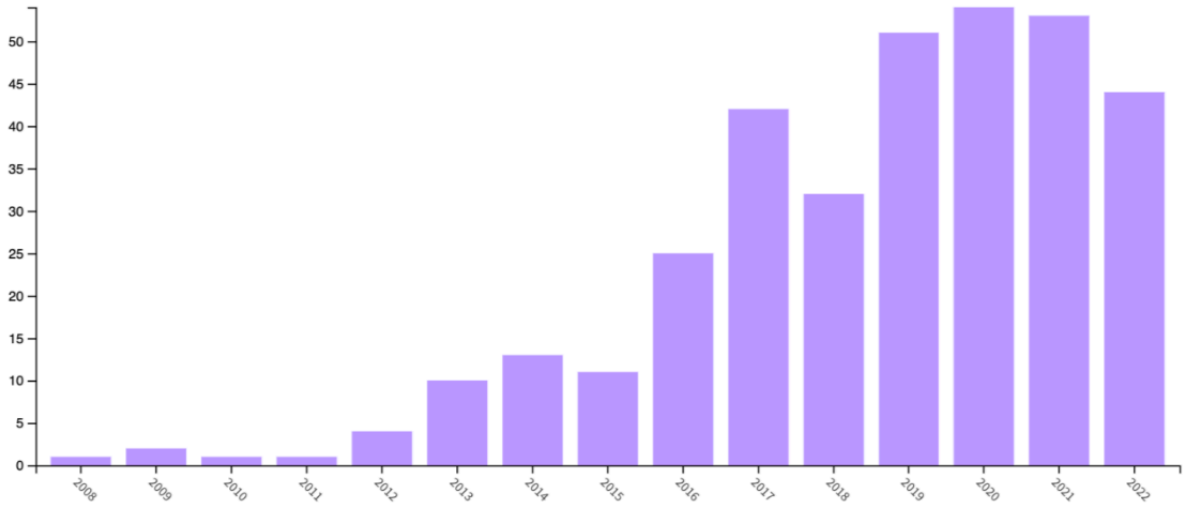


Figure 3.2 The distribution of the AR studies in education by year

A descriptive analysis of 344 AR studies in education was conducted using Excel, focusing on publication year, country, affiliations, journals, funding agencies, and citation trends. Initially, the AR studies were examined by their publication year as shown in Figure 3.2. According to Figure 3.2, no AR studies in educational contexts were published between 2000 and 2007, with the first appearing in 2008. A noticeable increase in AR studies occurred, with 51 studies published in 2019, 54 in 2020, 53 in 2021, and 44 in 2022 [2].

3.5 Trends and popularity of AR

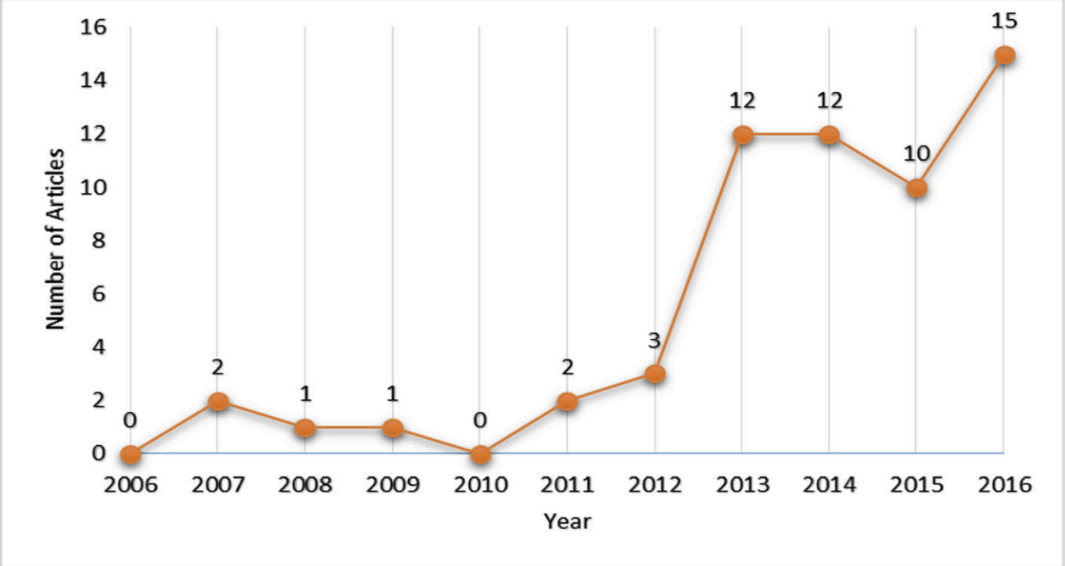


Figure 3.3 Changes in number of article published by years

An analysis was conducted to examine the yearly distribution of 58 articles featuring "augmented reality" in their titles, published in 8 SSCI-indexed journals (Computers & Education, Computers in Human Behavior, British Journal of Educational Technology, Interactive Learning Environments, Journal of Educational Technology & Society, Journal of Computer Assisted Learning, Journal of Science Education and Technology, Educational Technology Research and Development). The results are depicted in Figure 3.3. The graph indicates that no articles meeting these criteria were published in 2006 and 2010. Between 2006 and 2012, the number of articles remained relatively stable, with no significant increase. However, there was a sharp rise in 2013, with 12 articles published, and this upward trend continued into 2014 (12 articles) and 2015 (10 articles). The peak was observed in 2016, with 15 articles published [3]. This trend clearly demonstrates the growing dominance of AR in the educational field and suggests its significant impact in the near future.

3.6 Vuforia Engine

Vuforia Engine is a powerful tool that helps bring augmented reality (AR) to life on mobile devices. It's designed to recognize objects in the real world like images in a book or even surfaces in a room and then overlay digital content onto them, creating an interactive experience. For example, in a medical education app, Vuforia can make a flat image of a heart in a textbook come to life as a 3D model that students can explore from all angles [5].

One of the great things about Vuforia is its flexibility. It can work with specific markers (like QR codes) or recognize objects without any markers at all, just by understanding the shapes and surfaces around it. This makes it a versatile tool for creating educational experiences that are both engaging and easy to use [5].

Vuforia works really well with Unity, a popular game development platform. This means developers can create AR experiences with relative ease, especially if they're already familiar with Unity. Plus, Vuforia is reliable across different devices, ensuring that the AR experience is smooth and accurate, whether you're using an Android phone, an iPhone, or a tablet. This makes it an excellent choice for creating AR applications, especially in fields like education, where visual learning is key [10].

4. Technical Specification

The development of the Augmented Reality (AR) app for medical education involves several technical specifications that ensure the application is robust, efficient, and user-friendly across various platforms. I have created a technical specification table for building the app using all the modern technologies and tools.

Type	Name	Version	Summary
Programming Language	C#	9.0 or above	Used for scripting within Unity for AR functionalities and interactive features.
Framework	Unity	2021.3 or above	Game engine used for developing the AR application, providing cross-platform support.
AR SDK	Vuforia	10.12 or above	Augmented Reality SDK integrated with Unity for marker-based and markerless AR experiences.
Operating System	iOS	15.0 or above	Required for testing and running the app on iOS devices.
	Android	12.0 or above	Required for testing and running the app on Android devices
3D Modeling Software	Blender	3.6 or above	Used for creating detailed 3D anatomical models
File Format	FBX	2024 or above	Export format used for 3D models to ensure compatibility with Unity.
	OBJ	2024 or above	Alternative format used for simpler 3D models.
Audio/Video	OpenAI text to speech	Latest available	Used for converting the text to speech in narration format
User Interface (UI) Design	Unity UI Toolkit	2021.3 or above	Provides tools and components for designing and implementing the app's user interface.

Testing Tools	Xcode	14.0 or above	IDE used for testing and debugging iOS applications.
	Android Studio	2023.1 or above	IDE used for testing and debugging Android applications.

Table 4.1 Technical Specification

5. System Design

5.1 Architecture Diagram

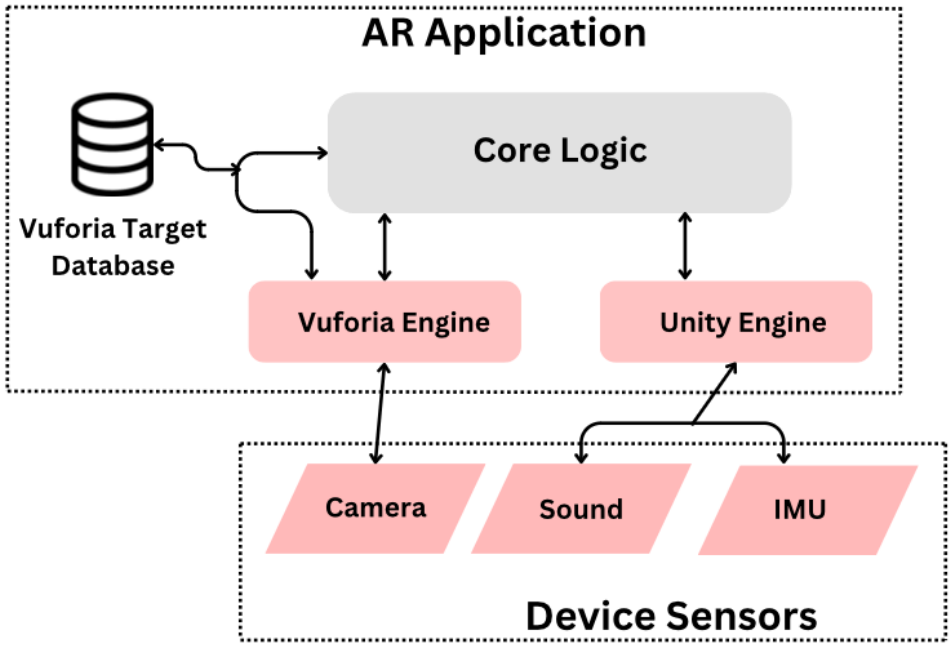


Figure 5.1 Architectural diagram of the AR Application

At the heart of the AR application is the Core Logic module. This component acts as the brain of the application, orchestrating the interaction between various subsystems, including the Vuforia Engine, Unity Engine, and the device sensors. It's responsible for all the logic, managing the flow of data, and ensuring that the AR experience is smooth and responsive.

The Vuforia Engine is a crucial part of the system, specifically designed to handle the AR functionalities such as image recognition and tracking. It communicates with the Vuforia Target Database, where reference images or objects are stored. When the application runs, Vuforia uses the device's camera to scan the environment. If it recognizes an object or image from its database, it triggers the corresponding AR content. This content is then fed into the Core Logic, which integrates it with the Unity Engine to produce the final AR display.

The Unity Engine handles the rendering and user interaction aspects of the AR experience. Unity takes the data processed by the Core Logic, such as the recognized objects or images from Vuforia, and uses it to display 3D models, animations, and other interactive elements. Unity's role is to bring the AR content to life on the screen, ensuring that it is responsive to user inputs and seamlessly integrated with the real-world environment captured by the device's camera.

At the base of the diagram, we see the Device Sensors, which include the camera, sound, and Inertial Measurement Unit (IMU). These sensors provide the raw data that the application needs to function. For instance, the camera captures the live feed for Vuforia to analyze, while the IMU provides information about the device's orientation and movement, which is crucial for maintaining a stable and accurate AR display.

Integration and Workflow: In this architecture, the device sensors feed real-time data into both the Vuforia and Unity engines. The Vuforia Engine processes visual data to recognize and track objects, while Unity uses this information to render and display the corresponding AR content. The Core Logic ties everything together, ensuring that the user’s interactions are accurately reflected in the AR experience and that the application runs efficiently across different devices.

5.2 Block Diagram of the App

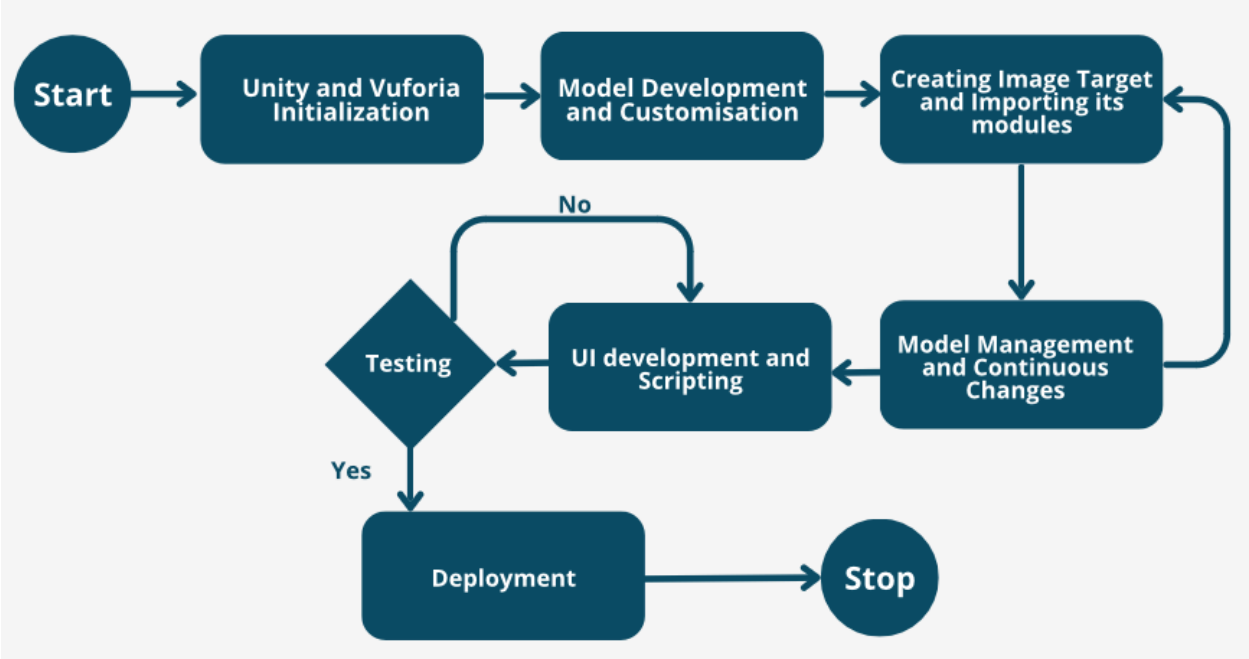


Figure 5.2 Block Diagram of the AR App

The block diagram outlines the development process for an AR application, beginning with the initialization of Unity and Vuforia, the foundational tools for creating AR experiences. Following this, the process moves into model development and customization, where 3D models are crafted and tailored to fit the application's needs. Image targets are then created and their

necessary modules imported, a crucial step for enabling the app to recognize different unique markers.

Once the models and targets are set, the process branches into UI development and scripting, where the user interface is designed and the application logic is programmed. Continuous testing is conducted at this stage, ensuring that any issues are identified and addressed early. If the testing results are unsatisfactory, the process loops back to UI development or model management for further refinement.

After successful testing, the app moves to the deployment phase, where it is prepared for release. The process concludes with the stop point, signaling the completion of the development cycle. This structured approach ensures a thorough and iterative development process, leading to a well-polished AR application.

5.3 Workflow Diagram

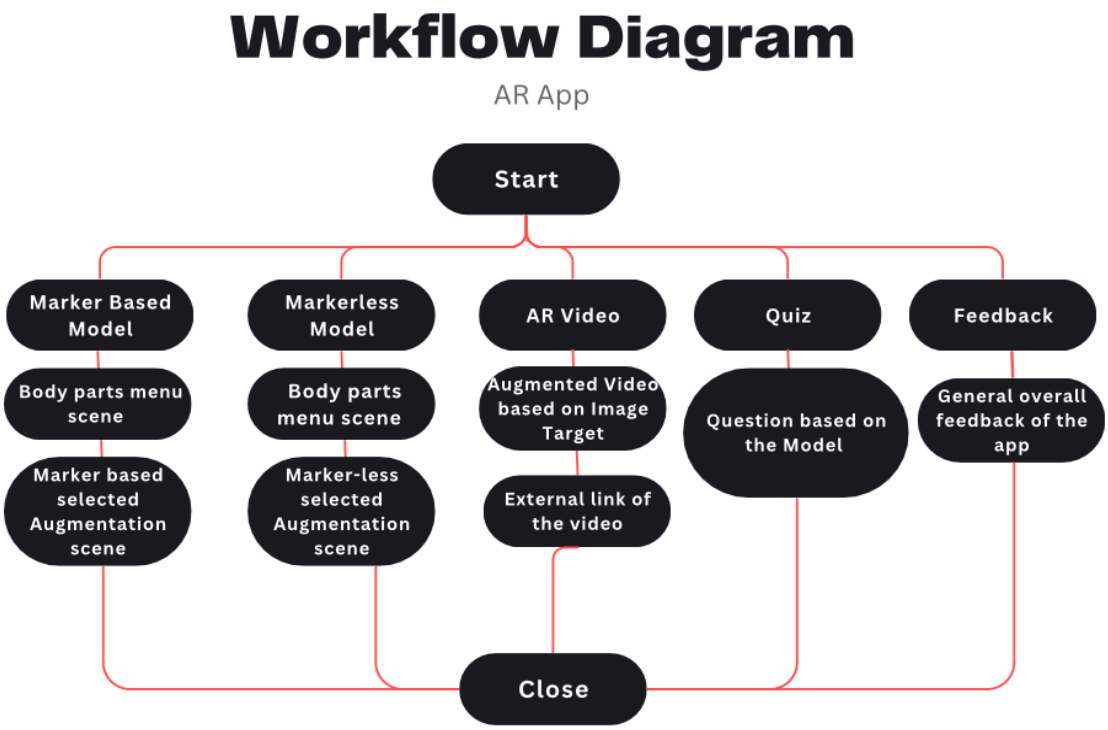


Figure 5.3 Workflow Diagram of the App

The workflow diagram for the AR application illustrates a clear and organized user journey, beginning with a start point and branching into five distinct paths. Users can choose between exploring marker-based or markerless AR models, each offering a body parts menu and corresponding augmentation scenes. Alternatively, they can view AR videos triggered by image targets or external links, providing an immersive multimedia experience. Additionally, the app includes a quiz feature to test users' understanding of the models, and a feedback section for

gathering user input. All paths lead to a common endpoint, ensuring a streamlined user experience.

5.4 Use Case Diagram

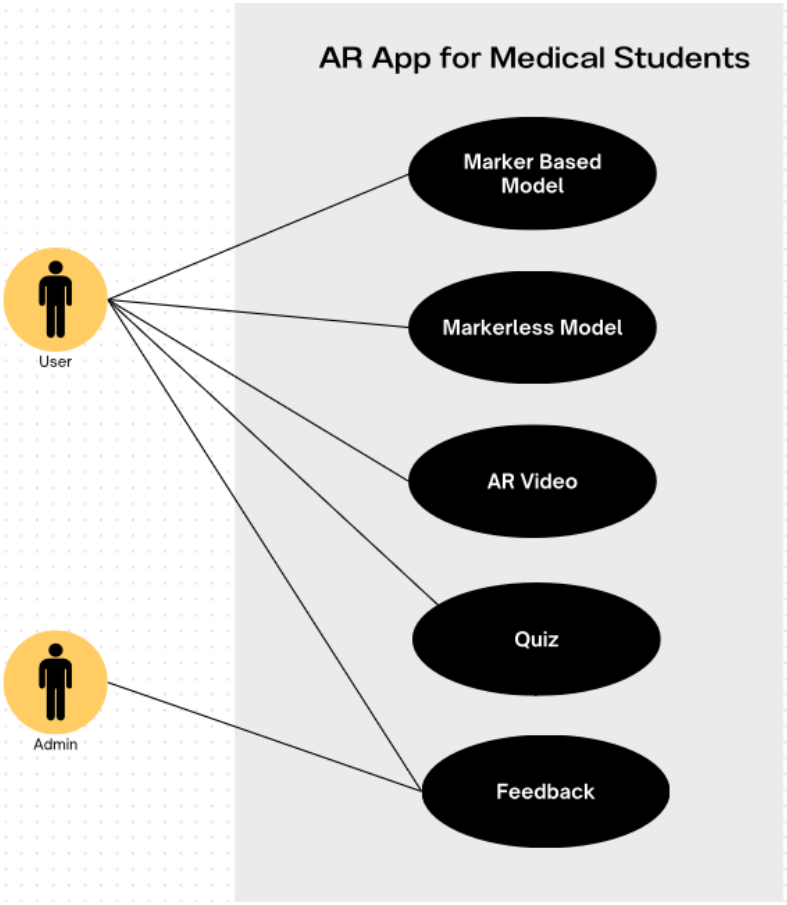


Figure 5.4 Use Case diagram

The use case diagram for the app illustrates the interactions between users (both students and administrators) and the various features of the application. The diagram highlights the primary functions of the app, including accessing marker-based models, markerless models, AR videos, quizzes, and feedback mechanisms.

- User Interactions: Medical students, represented as the "User," can interact with multiple features. They can explore anatomical models through marker-based or markerless AR technology, view augmented reality videos, participate in quizzes to test their knowledge, and provide feedback on their learning experience.
- Admin Interactions: The "Admin" role focuses on managing the feedback and overall application settings, ensuring that the app functions smoothly and meets the needs of the users.

This diagram provides a clear visual representation of how the app serves both educational and administrative purposes, highlighting the different ways users engage with the app to enhance their learning experience in a medical context.

6. Methodology

6.1 Project Planning and Management

The development of the AR application was planned and executed in distinct phases to ensure systematic progress and timely delivery. The project was managed using agile methodologies, where tasks were broken down into sprints. Each sprint focused on specific features or components, allowing for iterative development, testing, and refinement. This approach ensured that feedback could be integrated at various stages, leading to a more robust and user-friendly application.

6.2 Technology Stack

The development of the AR app involved a combination of tools and technologies:

- Unity Engine: Used for the overall development and rendering of the AR environment. Unity provided the flexibility and tools necessary for creating both marker-based and markerless AR experiences.
- Vuforia Engine: Integrated with Unity, Vuforia facilitated image recognition and tracking, essential for marker-based AR models.
- C# Programming Language: Utilized in Unity for scripting, implementing core logic, handling events, and managing user interactions.
- Canva/ Unity Asset Store: Utilized for creating and refining the visual assets, including markers and UI elements used in the app.
- Android Studio/Xcode: Depending on the target platform, Android Studio or Xcode was used for final deployment and testing of the AR app.

6.3 Development Phases

The development process was divided into several key phases:

6.3.1 Requirement Gathering and Analysis: The initial phase involved gathering requirements from potential users (medical students and instructors) to understand the specific needs and challenges in medical education. This was followed by an analysis phase to map these requirements to AR capabilities. While there are many challenges in the medical learning environment, I had to ensure that AR could effectively address these issues by gaining a deeper understanding of its capabilities.

6.3.2 Prototyping: Based on the gathered requirements, low-fidelity prototypes were created to visualize the user interface and interaction flow. These prototypes were reviewed and refined based on feedback before moving to the development phase.

6.3.3 AR Model Development: This phase involved working on the 3D models representing human anatomy and other educational content. Models were developed or sourced, optimized for AR performance, and integrated into the Unity project.

6.3.4 Unity and Vuforia Integration: The integration phase focused on setting up Unity with Vuforia, configuring image targets, and ensuring that marker-based AR experiences were functional. Scripting was done using C# to handle user interactions, model manipulation (like zooming and rotating), and transitions between different scenes.

6.3.5 UI/UX Design: A user-centered approach was taken in designing the UI, ensuring ease of navigation and accessibility. A consistent design approach was followed to make the app more visually pleasing for users. Additionally, the color combinations of the background and buttons were kept contrasting to enhance both visibility and accessibility, in line with design principles.

6.4 Deployment

After thorough testing, the application was deployed on both Android and iOS platforms. This phase involved configuring and fine-tuning the project settings to ensure that the app runs smoothly and consistently across all devices. For example, one critical setting ensures that the app always starts in landscape mode, maximizing the use of the mobile screen's full potential. Attention to these details was essential to ensure a smooth deployment process.

6.5 Feedback and Continuous Improvement

Feedback was continuously collected through the weekly meetings with professors, and direct interviews with my friends who were using the app. This data was analyzed to guide further updates and improvements to the application. For example, during a meeting with my second marker, I was advised to enhance the app's interactivity and incorporate more coding elements. In the second phase of my project, I successfully implemented all the feedback, including both the coding and interactivity enhancements. One such improvement was adding features like zooming in, zooming out, rotating the model, and including a quiz to test users' understanding of the concepts, thereby fulfilling the requirements for coding involvement and interactivity.

7. Implementation

7.1 Integration of Vuforia and Unity

Downloading and Installing Unity:

- Started by downloading and installing the Unity Hub from the official Unity website. Unity Hub allows you to manage different versions of Unity and my projects efficiently.
- Once Unity Hub is installed, I have chosen the version of Unity to use for my project. It is recommended to select a version that is compatible with the latest Vuforia Engine; the versions are clearly mentioned in the Technical Specification.

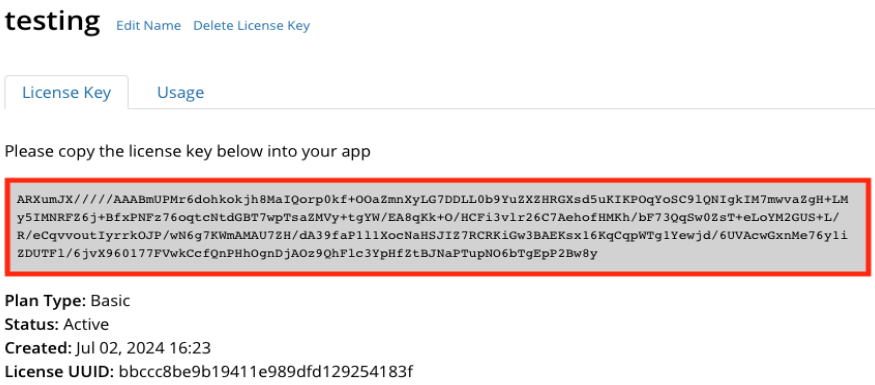


Figure 7.1 Vuforia Engine License Key

Creating a Vuforia License Key:

- Once logged in, navigate to the "Develop" section on the Vuforia website.
- Click on "Get Development Key" to create a new license key.
- Name your license key according to your project (e.g., "Medical AR App") and select the type of usage.
- After creation, copy the generated license key to your clipboard as shown in Figure 7.1 which is highlighted in the red box.

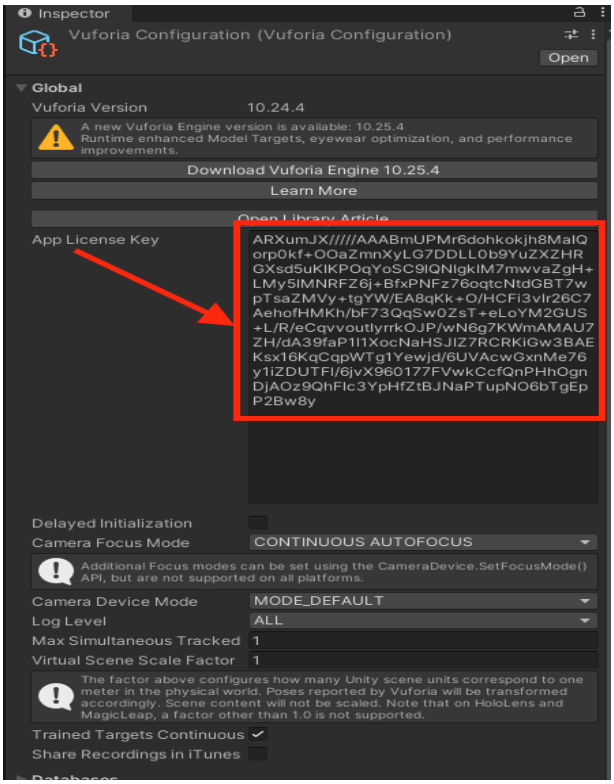


Figure 7.2 Unity Engine pasting license key in inspector panel of AR Camera

Configuring Vuforia in Unity:

- In Unity, select the "AR Camera" in the hierarchy or add a new AR Camera from the "GameObject" menu under "Vuforia Engine."
- With the AR Camera selected, go to the Inspector panel on the right.
- In the "Vuforia Configuration" section of the Inspector, you will see a field labeled "App License Key."
- Paste the license key you copied from the Vuforia Developer Portal into this field. The Screenshot of this process is shown in Figure 7.2 for better understanding.

7.2 Image Targets and Vuforia Database:

This section details the process of creating a Vuforia database, uploading image targets, understanding Vuforia's target rating system, and employing the "Show Features" tool to enhance performance.

Establishing the Vuforia Database: To integrate image targets into the augmented reality (AR) application, the initial step involved creating a dedicated database within the Vuforia Engine. This database serves as a repository for all the image targets that will be utilized in the AR experience. Vuforia leverages this database to accurately recognize and track the images throughout the AR interactions.

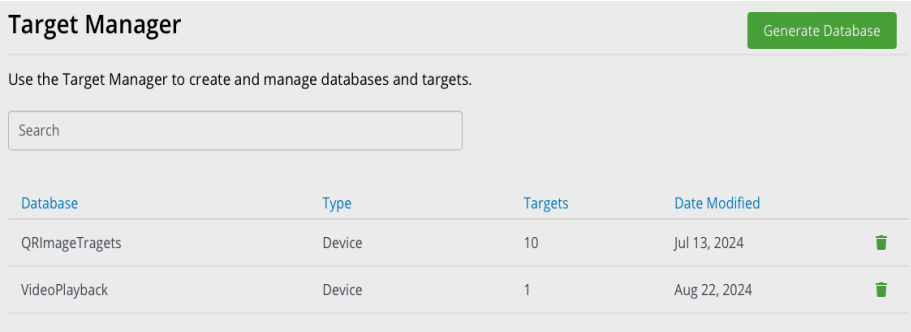


Figure 7.3 Vuforia Target Manager

Navigating to the Target Manager: After logging in, navigate to the "Develop" section and select the "Target Manager." This interface is used to manage databases and image targets within the Vuforia Engine.

Creating a New Database: To create a new database, select "Add Database" and provide an appropriate name for the database. For this project, "Device" was selected as the database type, as the image targets will be stored locally on the user's device.

Database Management: Once the database is created, it will appear in the Target Manager list. From this point, image targets can be added to the database, as illustrated in Figure 7.3.

Uploading Image Targets: With the database established, the next essential step was uploading image targets. These targets are critical, as they represent the key elements the AR application will recognize and track. The accurate placement of virtual objects within the real-world environment is contingent on the position and orientation of these images. The process for uploading image targets was approached as follows:

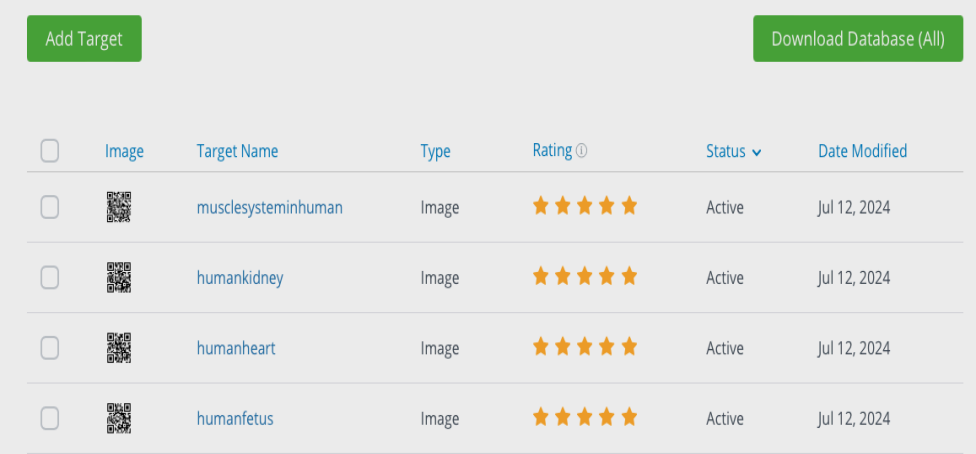


Figure 7.4 Vuforia Database

After creating the database in the Target Manager, the subsequent step was to add a new image target. I began by selecting the recently created database, then proceeded to click "Add Target," which prompted me to upload the image file intended for use as the AR marker.

Configuring the Image Target: Once the image was uploaded, it required configuration by specifying its width in scene units, typically measured in meters. This step was crucial, as it determined the size of the image target within the AR environment. After defining the appropriate width, the image was uploaded, and Vuforia processed it to generate the necessary tracking data.

Image Target Rating

Upon completing the upload, Vuforia automatically assigned a rating to the image target, ranging from 1 to 5 stars. This rating serves as an indicator of the image's suitability for AR tracking. A higher rating, closer to 5 stars, implies that the image possesses distinct and recognizable features, making it more effective for tracking in varied conditions.

For instance:

- 1 Star: Reflects poor tracking performance, indicating the image lacks distinct features, making it difficult for Vuforia to detect and track accurately.

- 5 Stars: Reflects optimal tracking performance, suggesting the image has a high concentration of unique features, allowing Vuforia to track it with ease.

To ensure reliable AR performance, I aimed for images with at least a 3-star rating. For images receiving lower ratings, I explored alternatives with enhanced contrast, sharper edges, and more varied textures to improve tracking capabilities.

Once the image targets were successfully configured and rated, I proceeded to download and import them into the Unity Engine. This allowed me to begin developing marker-based AR models within the Unity platform.

Show Features in Vuforia Image Target: After uploading an image target, Vuforia offers a valuable tool known as "Show Features." This tool enables users to visualize the specific features within the image that Vuforia utilizes for tracking. Understanding the functionality of this tool is essential for optimizing image targets, as it helps enhance AR performance by identifying the key elements that contribute to effective tracking.

Figure 7.5 Before and After using “Show Features” tool in Vuforia

After uploading an image target, I utilized Vuforia's "Show Features" tool to assess how effectively the image could be tracked. By selecting the "Show Features" option, I was able to visualize the specific points Vuforia identified within the image. These key points are critical for Vuforia's real-time recognition and tracking capabilities.

Interpreting the Highlighted Features: When reviewing the highlighted features, Vuforia displayed the identified key points as yellow dots. A higher concentration of these yellow dots generally indicated that the image contained more distinct features, enhancing its suitability as a tracking target.

Particular attention was given to areas where dense clusters of yellow dots appeared. These clusters represented robust features, which Vuforia could reliably recognize even under varying

lighting conditions or from different angles indicating that the image target would perform well in the AR environment.

Optimizing Image Targets: In instances where an image target exhibited sparse or minimal yellow dots, it was clear that tracking might be less reliable. To address this, I considered either editing the image to improve contrast or selecting an alternative image with more distinct features. The goal was to optimize the image by enhancing its key points to ensure better tracking performance during the AR experience.

7.3 Unity Scenes and UI

This section provides an in-depth exploration of Unity Scenes and the configuration of User Interface (UI) components within them. A clear understanding of these elements is essential for developing an interactive and visually engaging augmented reality (AR) application.

Unity Scene Overview: A Unity Scene serves as the primary space where all the components of a level or screen in an AR application are designed and organized. It functions as a virtual canvas where developers can position objects such as 3D models, lights, cameras, and UI elements. Each scene can represent different sections of the application, including the main menu, an AR model view, or interactive elements such as a quiz page.

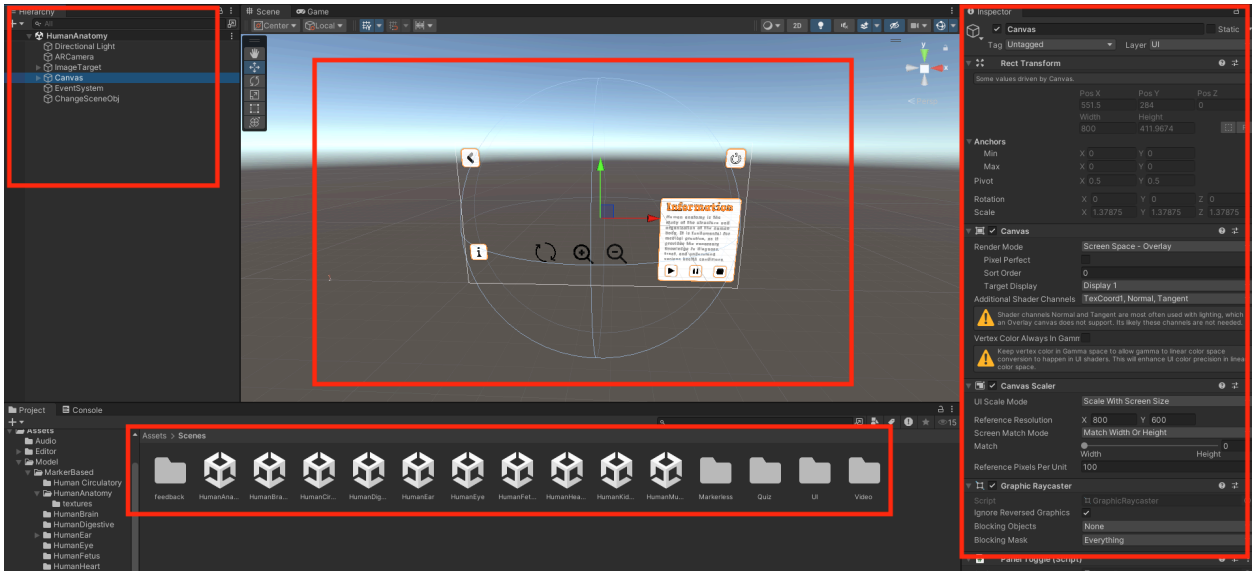


Figure 7.6 Unity Engine layout

In the figure above, the layout of the Unity interface is shown, with key panels highlighted for reference. The “Hierarchy Panel” is located in the top left corner, the “Main Scene Panel” is displayed in the center of the screen, the “Inspector Panel” is positioned on the right side, and the bottom section houses the **“**Project Folder”, where scenes, models, and other assets are stored.

- Hierarchy Panel (Top Left Corner): This panel lists all objects present within the current scene, allowing for efficient organization and management.
- Main Scene Panel (Center of the Screen): This panel serves as a visual workspace where users can view and manipulate the objects within the scene.
- Inspector Panel (Right Side of the Window): The Inspector Panel displays the properties and components of the selected object, providing the ability to modify its attributes and settings.
- Project Folder (Bottom Section): This section contains all the project's assets, including scenes, models, scripts, and resources, organized into folders for easy access and management.

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

public class ChangeScene : MonoBehaviour
{
    public void MainMenuScene()
    {
        SceneManager.LoadScene("MainMenu");
    }

    public void MarkerBasedMenuScene()
    {
        SceneManager.LoadScene("MarkerBasedMenu");
    }

    public void AnatomyScene()
    {
        SceneManager.LoadScene("HumanAnatomy");
    }

    public void BrainScene()
    {
        SceneManager.LoadScene("HumanBrain");
    }
```

Code 7.1 Scene Change code

In the development of the AR application, managing various functionalities across multiple scenes was crucial. For instance, created separate scenes to handle AR interactions, the main menu, and quiz or feedback features. To facilitate seamless navigation between these scenes, was done using Unity's Scene Manager. This enabled smooth transitions as users navigated through the app. The implementation involved writing custom C# scripts (as illustrated in Code 7.1), which were attached to GameObjects within each scene in Unity.

For example, utilization of a “changeScene” script to control scene transitions when a user clicked a button. In the Inspector window of each button, I configured the On Click() event field by dragging and dropping a GameObject, then selecting the relevant function from the “changeScene” script attached to that GameObject. This approach allowed me to trigger specific actions upon user interaction.

UI Components in Unity: The User Interface (UI) in Unity consists of various components used to create interactive elements. Below is an overview of the essential UI components employed during development:

- Canvas: The Canvas serves as the root component for all UI elements in Unity. It functions as the stage where UI components such as buttons, images, and text are displayed. Every UI element must be a child of the Canvas to ensure proper rendering across different device resolutions and aspect ratios.
- Panels: Panels were used to organize and group related UI elements. They facilitated efficient management of multiple elements as a single unit. For instance, grouping the buttons and labels in the main menu within a panel, making them easier to manipulate and arrange.
- Buttons: Buttons played a critical role in the application's interactivity. They allowed users to initiate actions, such as starting an AR experience, navigating to a new scene, or submitting quiz answers. customizing each button's appearance by adjusting images, colors, and text to align with the overall design of the application.
- Text: To display textual information within the UI, such as button labels or user instructions, employ Text components. Unity’s customization options enabled me to adjust the font, size, color, and alignment of the text to suit the app's visual style.
- Images: Images were used to present visual elements like icons, backgrounds, or decorative features within the UI. In many cases, these images served as button backgrounds or icons, contributing to the overall aesthetic and functionality of the application.

Developing interactive buttons and applying custom images was a key aspect of the UI design process. Below is a detailed outline of how this was accomplished:

Creating Buttons: To create a button in Unity, by accessing the Hierarchy window, right-click, and select UI > Button. This action added a default button to the Canvas. Although the button came with a standard appearance, by customizing it to align with the application's design by modifying the text, adjusting its size, and replacing the background image.

Applying Custom Images to Buttons: To use custom images for buttons, first convert these images into UI Sprites. The process began by importing the image into Unity by dragging it into the Assets folder. Once imported, I selected the image in the Assets folder to open the Inspector window, changed the Texture Type from "Default" to "Sprite (2D and UI)," and applied the changes.

After the image was converted to a sprite, I applied it as the button's background. To do so, select the button in the Hierarchy, locate the Image component in the Inspector, and drag the sprite from the Assets folder into the Source Image field.

Customizing Button Interactivity: To define the button's functionality, attach a script to it. In the Inspector window, the On Click() event field is allowed to assign a GameObject. Then select a specific function from the script attached to that GameObject, ensuring the desired action would be triggered when the button was clicked.

7.4 Importing and Executing AR model

This section outlines the steps involved in importing 3D models into Unity, applying materials to ensure accurate display of colors and textures, and preparing the models for augmented reality (AR) execution.

Importing 3D Models into Unity: Unity supports various 3D model formats, with FBX and OBJ being the most widely used. The process of importing a 3D model into the project is as follows:

Importing the Model: In Unity, navigate to the Assets folder where all project files are managed. In the Project Window, right-click and select “Import New Asset”. Using the file explorer, locate the appropriate FBX or OBJ file, select it, and click “Import”. Unity will automatically add the model to the project. Once imported, the 3D model will appear in the Assets folder, and it can be dragged and dropped into the Hierarchy or Scene Window for placement within the Unity scene (Figure 7.7).

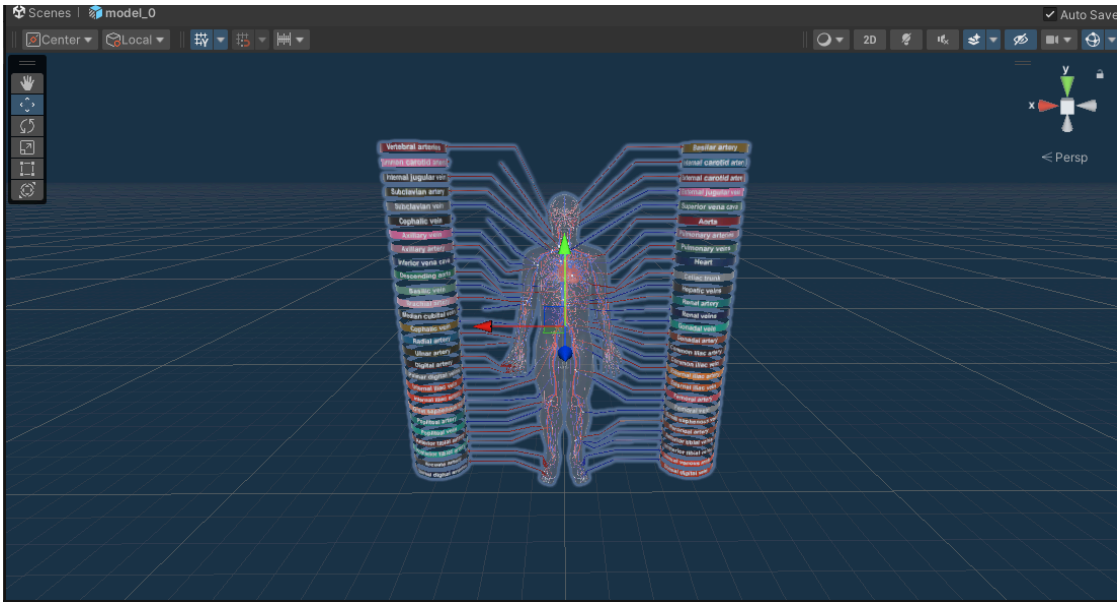


Figure 7.7 3d Model imported inside the Unity Scene Window

Applying Materials to the 3D Model: Once the 3D model was imported into Unity, the next step involved applying materials to ensure the correct display of colors, textures, and overall appearance. Materials are essential for achieving a realistic and visually appealing representation of the model.

1. Overview of Materials in Unity: In Unity, materials are assets that define the visual properties of a 3D model. They control elements such as color, texture, reflectivity, transparency, and other surface characteristics. Each material is associated with a shader, a small program responsible for calculating the model's surface appearance. For augmented reality (AR) models, the Standard Shader is commonly used due to its versatility and ability to support a wide range of visual effects.

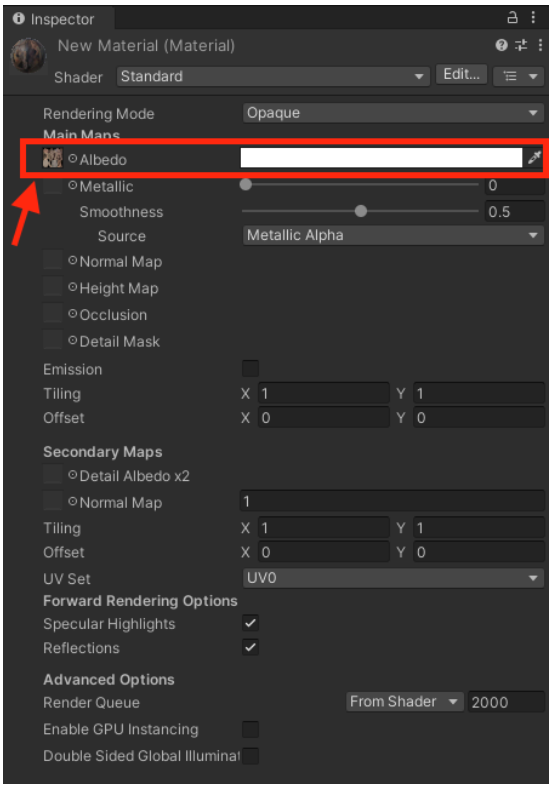


Figure 7.8 Unity Shader settings in Inspector Panel

2. Creating and Assigning Materials: If the 3D model was imported with textures, Unity may have automatically generated materials and applied them to the model. However, for greater control, materials were also created manually. This was accomplished by right-clicking in the Assets window and selecting Create > Material, naming each material according to the part of the model it would be applied to (e.g., "SkinMaterial" or "MetalMaterial"). In the Inspector window, various settings were adjusted, including the Albedo (color), Metallic, and Smoothness properties. A texture (image file) was assigned to the Albedo property by clicking the small circle next to it and selecting a texture from the Assets folder (Figure 7.8).

3. Applying Materials to the Model: To apply materials to specific parts of the model, the desired material was dragged from the Assets folder and dropped onto the model, either in the Scene or Hierarchy window. If the model comprised multiple parts (such as different meshes for distinct body sections), separate materials were applied to each part by selecting them individually and assigning the corresponding material.

4. Adjusting Material Properties: After applying the materials, further adjustments were made to refine the visual appearance. For instance, the Metallic and Smoothness sliders were modified to create surfaces that resembled metal or plastic. For models requiring transparency, the material's Rendering Mode was set to either Transparent or Cutout, and the Alpha value in the Albedo color picker was adjusted accordingly.

Finalizing the Model for AR: Once the materials were applied and adjusted, the 3D model was prepared for AR deployment. The model was carefully positioned and scaled within the scene, and AR-specific components, such as Vuforia Image Targets or AR Foundation components, were properly configured. The final step involved testing the model in an AR environment to ensure optimal performance across different devices and lighting conditions.

7.5 Maker Based Augmented Reality

This section outlines the process of creating a marker-based augmented reality (AR) model, beginning with the import of a Vuforia database into Unity, followed by placing the image target, and concluding with the positioning of a 3D model on top of the image target.

Step 1: Importing the Vuforia Database into Unity

After the image targets and database were created in the Vuforia Developer Portal, the next step was to export the database for use in Unity. Once the image targets were uploaded and processed in the portal, the Download Database button was selected, and the export format was set to "Unity Editor." The package containing the image targets and necessary metadata was then downloaded.

With the database package ready, it was imported into the relevant Unity project. This was done by navigating to Assets > Import Package > Custom Package. After locating the downloaded Vuforia database package, the package was selected, and Unity automatically imported the image targets and associated metadata into the project.

Step 2: Placing the Image Target in the Scene

Once the Vuforia database was integrated, the next task involved placing the image target in the Unity scene. This was achieved by selecting GameObject > Vuforia Engine > Image, which added a new Image Target GameObject to the scene, visible in the Hierarchy window. Selecting the Image Target GameObject allowed access to its properties in the Inspector window.

In the Inspector, the Image Target Behaviour component was used to configure the target. The Database field was set to the newly imported Vuforia database, which appeared in the

dropdown menu. The Image Target field was then configured to reference the specific image target to be recognized, corresponding to the one uploaded to the Vuforia Developer Portal.

It was essential to match the size of the Image Target in Unity with the actual physical dimensions of the printed image. The Width and Height parameters in the Inspector were adjusted accordingly, ensuring accurate scaling. For example, if the physical image measured 10 cm by 10 cm, the dimensions were set to 0.1 units in Unity, as Unity uses meters as the default unit of measurement.

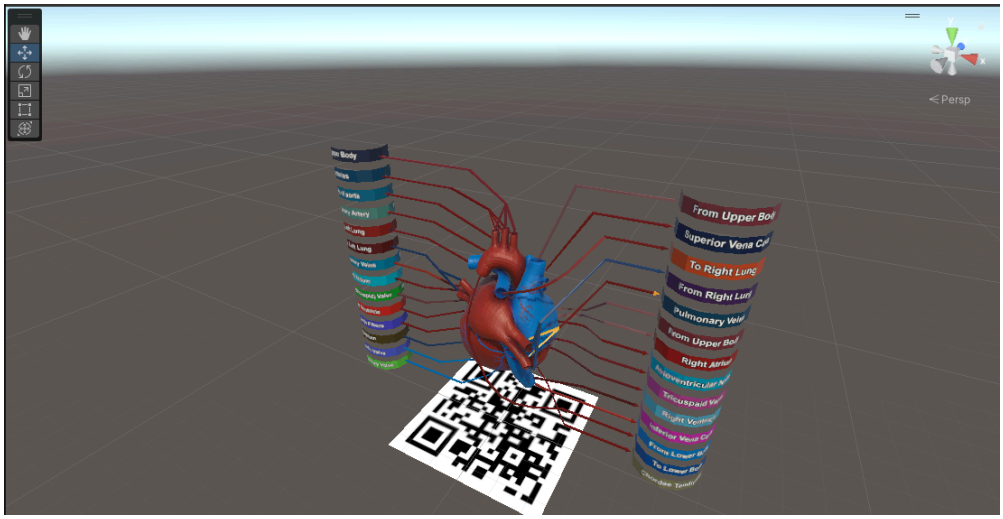


Figure 7.9 3D model placed on top of the Image Target

Step 3: Placing the 3D Model on the Image Target

With the image target in place, the next step involved adding the 3D model to the scene. The 3D model, in FBX or OBJ format, was imported into the Unity project by dragging the model file into the Assets window. Subsequently, the model was dragged from the Assets window into the Hierarchy window, making it a child of the Image Target GameObject. This configuration ensured that the 3D model would inherit the movement, rotation, and scaling properties of the Image Target.

The position, rotation, and scale of the 3D model were then adjusted relative to the Image Target. Typically, the model was positioned directly on top of the image target, with its base aligned to the plane of the image target. Unity's Move, Rotate, and Scale tools were used to fine-tune the placement of the model (see Figure 7.9).

To verify that the model was accurately positioned, the scene was tested within the Unity Editor. By placing a printed version of the image target in front of the device’s camera and running the scene, it was confirmed that the 3D model appeared correctly on top of the image target and maintained its position during target movement.

7.6 Markerless Augmented Reality

Markerless Augmented Reality (AR) provides a more flexible and immersive experience compared to marker-based AR by eliminating the need for predefined image targets. Instead, it utilizes the environment, such as horizontal or vertical surfaces, to place digital objects. Vuforia's Ground Plane feature plays a critical role in achieving this by enabling the placement of AR models on flat surfaces in the real world, such as floors, tables, or walls.

The Ground Plane feature in Vuforia facilitates markerless AR by detecting and tracking flat surfaces in the user's environment, accomplished through two main components: the Ground Plane Stage and the Plane Finder.

Understanding the Ground Plane Stage: The Ground Plane Stage in Unity acts as a placeholder for the location where the 3D model will appear in the real world. It represents the virtual anchor on the detected surface, allowing for precise placement of AR content. While the Ground Plane Stage is set up in Unity, it does not appear in the final application. Rather, it serves as a reference point, ensuring that the AR model is correctly positioned relative to the detected surface (see Figure 7.10).

Setting Up the Ground Plane Stage: To add a Ground Plane Stage in Unity, navigate to GameObject > Vuforia Engine > Ground Plane > Ground Plane Stage. Once added, it will appear in the Hierarchy window. The 3D model is typically parented to the Ground Plane Stage, ensuring that the model is placed accurately on the detected surface within the real-world environment.

Positioning Models: After the 3D model is parented to the Ground Plane Stage, adjustments to its position, rotation, and scale relative to the stage can be made (refer to Figure 7.10 for guidance). These adjustments determine how the model will appear when placed in the real world. For example, if the model represents a piece of furniture, it should be aligned to sit on the virtual floor (the plane detected by Vuforia) and should appear appropriately sized and oriented for optimal AR visualization.

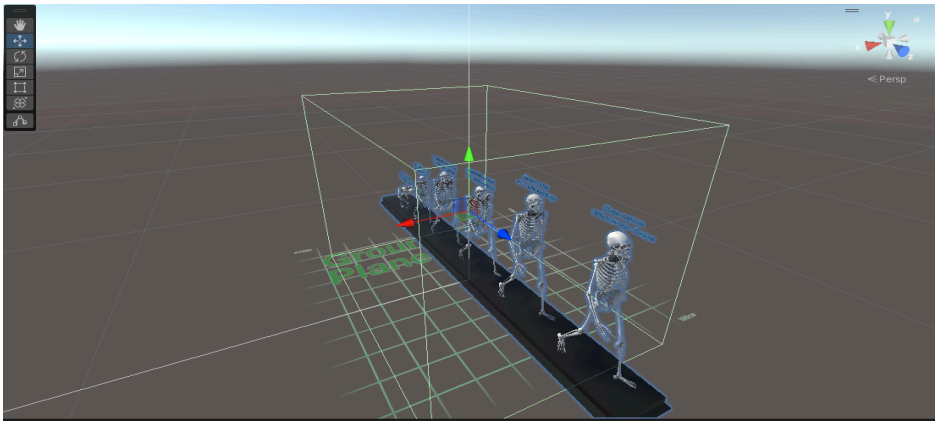


Figure 7.10 3D model placed on top of the Ground Plane

Plane Finder: The Plane Finder is integral to detecting flat surfaces in the environment, such as floors or tabletops, where AR models can be placed. By leveraging Vuforia's advanced algorithms, the Plane Finder identifies and tracks these planes, facilitating accurate placement of virtual objects. In Unity, the Plane Finder GameObject contains a Plane Finder Behaviour component, which handles surface detection and tracking.

Adding a Plane Finder: To incorporate a Plane Finder into the scene, navigate to GameObject > Vuforia Engine > Ground Plane > Plane Finder. This action adds a Plane Finder GameObject to the Unity scene. Typically, the Plane Finder includes a visual indicator, such as a reticle, that helps users understand where AR content will be placed as they move their device within the real world.

Configuring the Plane Finder: The Plane Finder's behavior can be customized in the Inspector window, including the types of surfaces it detects and how it responds when a suitable surface is found. This GameObject is responsible for triggering the placement of the AR model once the Plane Finder successfully identifies a flat surface.

Placing the Model: When the Plane Finder detects an appropriate surface, it sends a signal to the Ground Plane Stage, which then positions the AR model at the correct location. As users move their device, the AR model remains anchored to the detected surface, ensuring a stable and realistic AR experience.

7.7 Interactive features included in the App

In developing the AR application, several interactive features were designed to enhance the overall user experience. These features include the ability to rotate, zoom in, and zoom out on the 3D models presented in augmented reality. By giving users control over these interactions, the app becomes more engaging, allowing for a deeper and more informative exploration of the AR content.

Rotate Feature: The rotate feature enables users to manipulate the 3D model, allowing for a full view from different angles. This functionality is particularly valuable in educational applications, where viewing a model from all sides can help users gain a more comprehensive understanding of the object.

- A script, rotateObject.cs, was developed to manage the rotation functionality of the model.
- The rotation process is controlled by a boolean variable named rotatingStatus, which determines whether the model is currently in rotation.
- When the user initiates the rotation (e.g., through a button press), the RotateObject() method is triggered, toggling the value of rotatingStatus between true and false.
- If rotatingStatus is set to true, the Update() method continually rotates the model around the Y-axis, maintaining a smooth rotation by using a predefined rotation speed (rotateSpeed). The implementation of Time.deltaTime ensures that the rotation speed remains consistent across different frame rates, providing a seamless user experience.

```
public class rotateObject : MonoBehaviour
{
    public GameObject objectRotate;

    public float rotateSpeed = 50f;
    bool rotatingStatus = false;

    //rotate object function implemented using boolean values
    public void RotateObject()
    {

        if (rotatingStatus == false)
        {
            rotatingStatus = true;
        }
        else
        {
            rotatingStatus = false;
        }
    }

    void Update()
    {
        if (rotatingStatus == true)
        {
            //rotate object with speed of that respect to the deltatime
            objectRotate.transform.Rotate(Vector3.up, rotateSpeed * Time.deltaTime);
        }
    }
}
```

Code 7.2 program snippet of the rotateObject.cs script

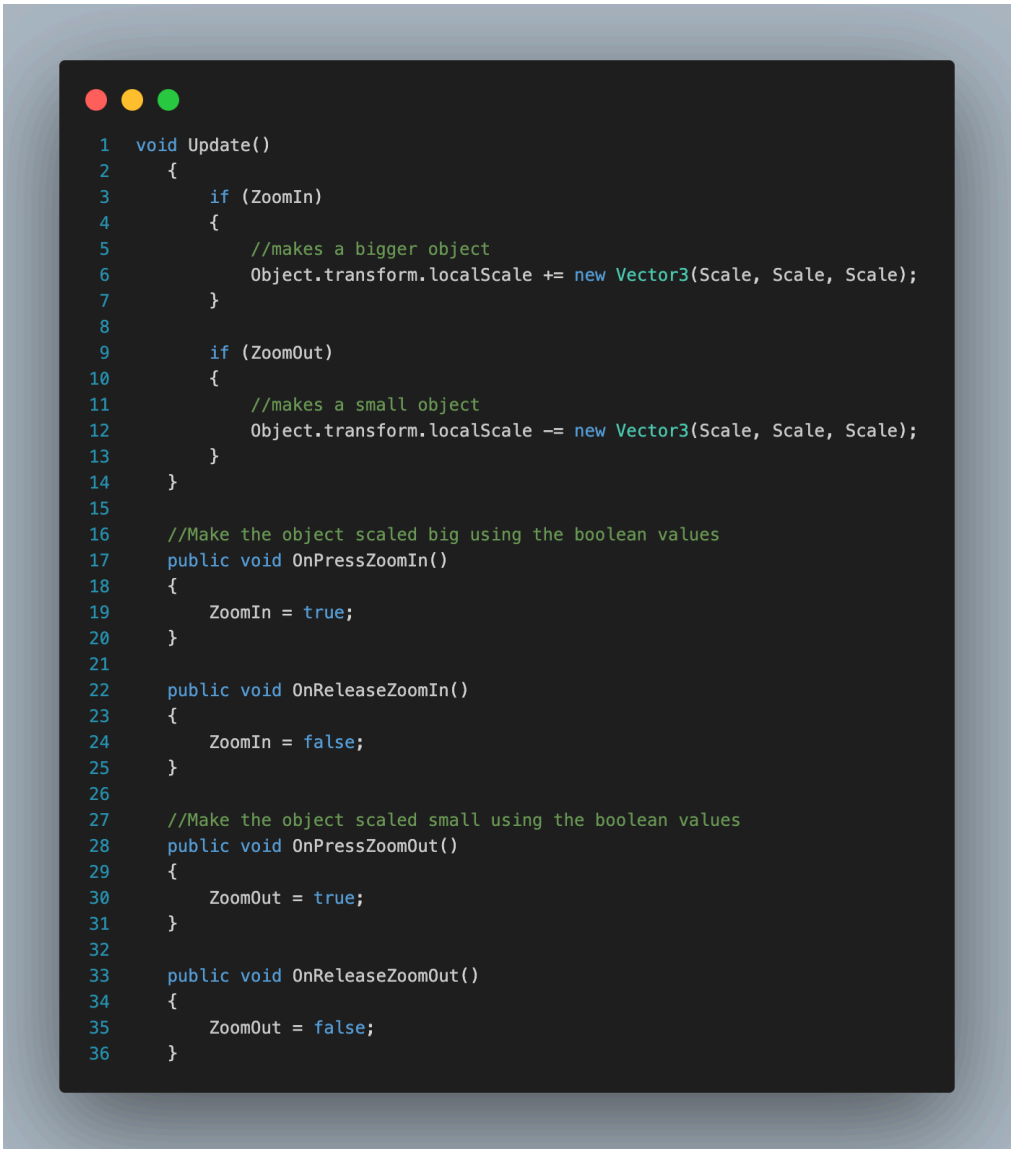
This snippet ensures that whenever rotatingStatus is true, the model rotates smoothly at a steady speed.

Zoom-In and Zoom-Out Feature: The zoom-in and zoom-out features provide users with the ability to adjust the size of the 3D model within the AR environment. These functions enhance user interactivity by allowing detailed inspection of specific areas or an overall view of the model.

Implementation Details:

- The zoom functionality is managed through a script named ScaleInOut.cs.
- Two boolean variables, ZoomIn and ZoomOut, are used to track whether the model is currently being scaled up or down.
- When users press the zoom-in or zoom-out buttons, corresponding methods (OnPressZoomIn and OnPressZoomOut) are invoked, setting the respective boolean variables to true.

- The Update() method continuously monitors these variables. If ZoomIn is set to true, the model's scale increases, causing it to enlarge. Conversely, if ZoomOut is set to true, the scale decreases, reducing the model's size.
- Scaling is achieved by adjusting the model's localScale attribute in small increments, ensuring smooth transitions. A predefined Scale value is used to control the rate at which the model enlarges or shrinks.



```
void Update()
    {
        if (ZoomIn)
        {
            //makes a bigger object
            Object.transform.localScale += new Vector3(Scale, Scale, Scale);
        }

        if (ZoomOut)
        {
            //makes a small object
            Object.transform.localScale -= new Vector3(Scale, Scale, Scale);
        }
    }

    //Make the object scaled big using the boolean values
    public void OnPressZoomIn()
    {
        ZoomIn = true;
    }

    public void OnReleaseZoomIn()
    {
        ZoomIn = false;
    }

    //Make the object scaled small using the boolean values
    public void OnPressZoomOut()
    {
        ZoomOut = true;
    }

    public void OnReleaseZoomOut()
    {
        ZoomOut = false;
    }
```

Code 7.3 program snippet of the ScaleInOut.cs script

This code ensures that the model scales uniformly along all axes, either increasing or decreasing in size based on the user's input.

Event Triggers and Box Collider: To facilitate user interaction with the 3D model, a Box Collider was added to detect user inputs, such as taps or clicks. This is essential for activating interactive features like rotation and zoom functionality.

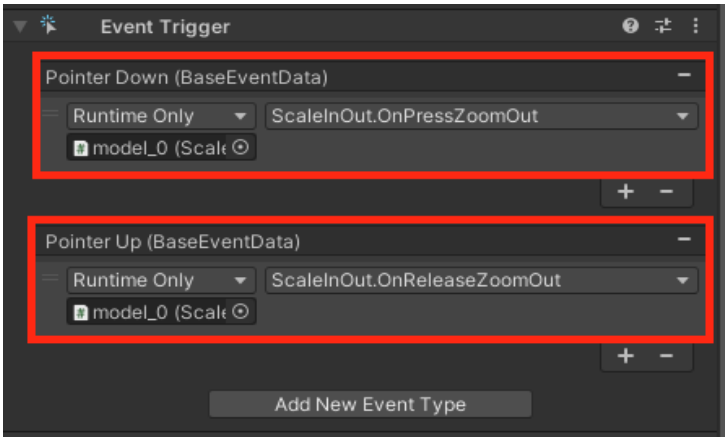


Figure 7.11 Event Trigger in Unity Engine

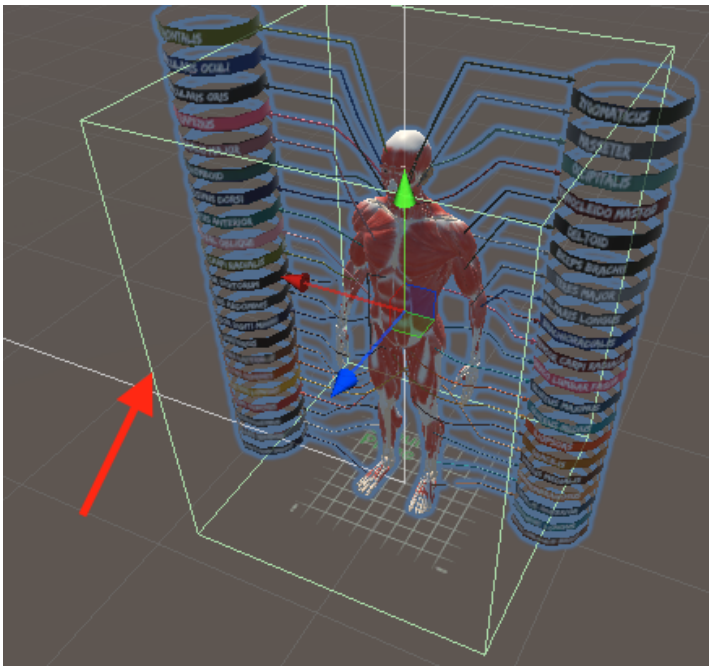


Figure 7.12 Box Collider been attached to the model

- The Box Collider is used to define the interaction zone around the model. It ensures the model can detect user inputs by creating a boundary around the object that responds to interactions.
- Event Triggers are employed to connect user inputs (such as pressing a button) to specific functions within the scripts controlling the model's behavior:

- The zoom-in button, for example, is linked to the OnPressZoomIn method using an event trigger. When the button is pressed, this method is activated, starting the zoom-in process.
- When the button is released, a separate event trigger calls the OnReleaseZoomIn method, stopping the zoom-in action.

These event-based interactions allow users to easily control the AR model's rotation and scaling with precision. The use of event triggers and colliders contributes to a more immersive and intuitive user experience. This level of interactivity is particularly useful in educational applications, such as medical training, where the ability to closely examine and manipulate complex 3D structures from multiple angles is critical to learning outcomes.

In addition to the interactive features like rotation and zoom, the AR application also incorporates a sound feature designed to enrich the educational experience by providing audio descriptions related to the 3D model displayed on the screen. This feature is particularly beneficial for users who prefer auditory learning or want to reinforce their understanding through a multi-sensory approach.

Audio Clips and Integration:

- Each 3D model in the application is associated with a specific audio clip that delivers detailed information or instructions about the content currently displayed.
- These audio files were imported into Unity as Audio Clips and then linked to the corresponding models within the AR environment. This ensures that each model has a relevant audio guide that enhances the user's understanding.

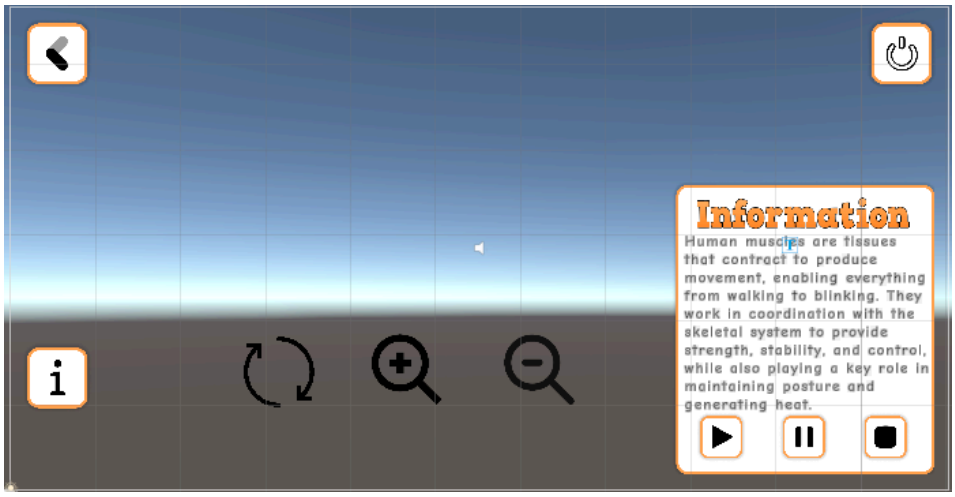


Figure 7.13 Image of Canva in Unity Scene Panel

User Controls:

Play Audio: Users can start the audio playback by pressing the "Play" button. When activated, the audio clip begins to play, offering a verbal explanation of the model currently visible on the

screen. This is managed by a script that accesses the Audio Source component attached to the model.

Pause Audio: The "Pause" button allows users to pause the audio at any point. This is particularly useful if the user needs more time to examine the model or if they want to temporarily stop the audio.

Stop Audio: The "Stop" button stops the audio playback completely. If the user presses "Play" again after stopping, the audio will restart from the beginning.

User Experience: The addition of sound to the AR experience significantly enhances engagement and informativeness. It caters to different learning styles, allowing users to absorb information both visually and audibly. This feature is especially beneficial in educational settings, where it can act as a guided tutorial, helping users grasp complex concepts more easily.

Overall Impact: The sound feature adds a valuable layer of interactivity and educational value to the AR application, making it a more versatile and effective tool for learning. It ensures that the application is not just visually compelling but also rich in content, appealing to a broader range of users and learning preferences.

7.8 Augmented Reality Video

In this section of the AR application, users can engage with augmented reality (AR) video content by simply scanning an image target with their device. This feature enables users to watch videos in a virtual space directly within the app, enhancing interactivity and immersion.

Image Target Scanning: The experience begins when the user scans a predefined image target using their device's camera. This image target acts as a trigger that prompts the AR application to load and display the associated video. The image target is pre-configured in the Vuforia database, ensuring that the app can recognize it quickly and accurately when scanned.

Virtual Video Playback: Once the image target is recognized, the video is overlaid onto the real-world environment as seen through the device’s screen. The video appears anchored to the image target, creating the illusion of a video playing within physical space. The video is rendered within the Unity scene using a Video Player component, which controls the playback functions.

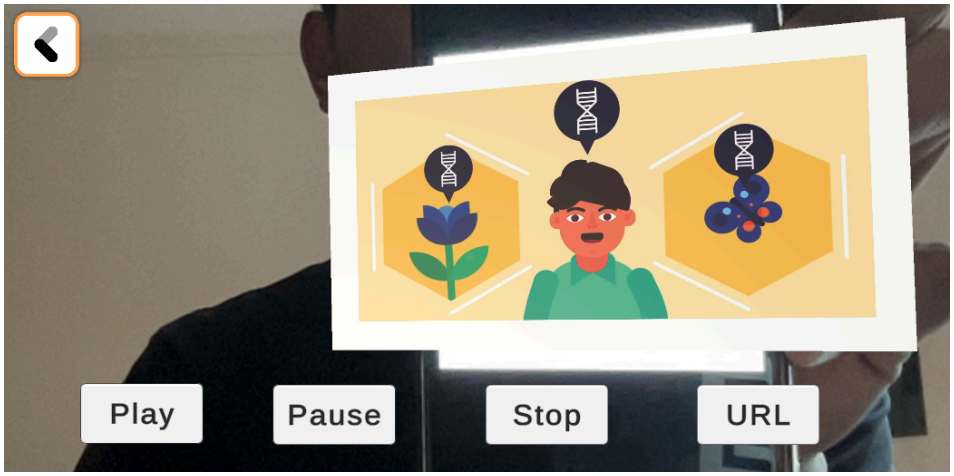


Figure 7.14 Image of Augmented Reality Video

User Controls: Users are given on-screen controls similar to those found in traditional video players, but these are embedded within the AR environment:

- Play: Pressing the "Play" button starts the video from the beginning or resumes it from where it was paused.
- Pause: The "Pause" button allows users to pause the video at any point, letting them take a closer look or pause the experience temporarily.
- Stop: The "Stop" button halts the video entirely, resetting it to the beginning. If the user presses "Play" again after stopping, the video restarts from the beginning.

URL Button: Besides the basic playback controls, the interface includes a URL button. When the user clicks this button, they are directed to the original source of the video, such as YouTube. I have taken this suggestion and done this feature after the final meeting with my professor. This feature is handy for users who want to access additional content or watch the full-length video.The code to handle this action is shown in the Code 7.4.

```
1
2  public class URL : MonoBehaviour
3  {
4      public void OpenURL()
5      {
6          Application.OpenURL("https://www.youtube.com/watch?v=6rv5Z8EBXwY&t=11s");
7          Debug.Log("This Link is wroking perfect");
8
9      }
10 }
11
```

Code 7.4 Code to handle the URL button

This AR video feature enhances the app by seamlessly integrating real-world elements with digital content. Users can play, pause, and stop the video within the AR environment, giving them control over their viewing experience and making it more interactive and personalized. The URL button further adds value by connecting users to external resources, offering them more in-depth information and extending their learning or entertainment experience.

The Augmented Reality Video feature in this app leverages AR technology to deliver multimedia content in an engaging and immersive way. It enriches the overall user experience by allowing users to interact with videos in a new, innovative context, blending digital media with the real world.

7.9 Detailed overview of the Quiz

The quiz feature in the AR application is designed to enhance the user's learning experience by testing their knowledge of the AR models they've interacted with. This feature provides an engaging way to review and reinforce the information presented in the app. The quiz is dynamic and interactive, offering immediate feedback and a final score summary to help users gauge their understanding.

Quiz Structure: The quiz is composed of multiple-choice questions, each related to specific AR models within the app. Users are required to select the correct answer from a set of options. For every correct answer, the user's score increases by one point. At the end of the quiz, the app calculates the user's performance as a percentage, based on the number of correct answers.

Core Components:

- Question Display: The questionText field displays the current question. The questions are sourced from a dataset (qtsData), ensuring they are relevant to the AR content.
- Reply Buttons: Users select their answers by clicking on one of the reply Buttons. These buttons are dynamically updated with the possible answers for each question.
- Score Tracking: The user's score is tracked in real-time and displayed on the screen through the “scoreTxt” (Variable name in the code) field.
- Feedback Display: After selecting an answer, the app provides immediate feedback. If the answer is correct, a "Correct" message is shown; if incorrect, a "Wrong" message appears. This immediate feedback helps users understand their performance on each question.

Code Walkthrough:

- Starting the Quiz: When the quiz begins, the score is reset to zero, and the first question is displayed. The SetQts() method is responsible for loading the question text and possible replies into the UI.

```
void SetQts(int questionIndex)
{
    questionText.text = qtsData.questions[questionIndex].questionText;

    foreach (Button r in replyButtons)
    {
        r.onClick.RemoveAllListeners();
    }

    for (int i = 0; i < replyButtons.Length; i++)
    {
        replyButtons[i].GetComponentInChildren<Text>().text = qtsData.questions[questionIndex].replies[i];
        int replyIndex = i;
        replyButtons[i].onClick.AddListener(() => {
            CheckRep(replyIndex);
        });
    }
}
```

Code 7.5 SetQts method code snippet to implement Quiz

- Answer Checking: The CheckRep() method evaluates whether the selected reply is correct. If the answer is correct, the score is increased, and positive feedback is provided. If the answer is incorrect, a "Wrong" message is shown. After displaying the feedback, the app automatically moves to the next question.

```
void CheckRep(int replyIndex)
{
    if (replyIndex == qtsData.questions[currentQuestion].correctReplyIndex)
    {
        score++;
        scoreTxt.text = score.ToString();

        Correct.SetActive(true);

        foreach (Button r in replyButtons)
        {
            r.interactable = false;
        }

        StartCoroutine(NextQuestion());
    }
    else
    {
        Wrong.SetActive(true);

        foreach (Button r in replyButtons)
        {
            r.interactable = false;
        }

        StartCoroutine(NextQuestion());
    }
}
```

Code 7.6 CheckRep method code snippet to evaluate the reply

- Progressing through the Quiz: The NextQuestion() method advances the quiz to the next question after a brief delay, allowing users to see their feedback before moving on. If there are no more questions, the quiz ends, and the final score is calculated and displayed.

```
IEnumerator NextQuestion()
{
    yield return new WaitForSeconds(2);

    currentQuestion++;
    if (currentQuestion < qtsData.questions.Length)
    {
        Reset();
    }
    else
    {
        GameFinish.SetActive(true);

        float scorePercent = (float)score / qtsData.questions.Length * 100;

        FinalScr.text = "You scored " + scorePercent.ToString("F0") + "%\n";

        if (scorePercent < 50)
        {
            FinalScr.text += "Game Finished";
        }
        else if (scorePercent < 60)
        {
            FinalScr.text += "Keep pushing!";
        }
        else if (scorePercent < 70)
        {
            FinalScr.text += "Good Job!";
        }
        else if (scorePercent < 80)
        {
            FinalScr.text += "Well Done!";
        }
        else
        {
            FinalScr.text += "You are Amazing!!";
        }
    }
}

```

Code 7.7 NextQuestion method code snippet

Final Scoring and Feedback:

- At the end of the quiz, the app calculates the user's score as a percentage of the total questions answered correctly. The “FinalScr” (Final Score) field displays this percentage along with a personalized message that encourages the user based on their performance.

- This feature not only adds an interactive component to the app but also serves as a valuable tool for reinforcing the educational content provided through the AR models.

The quiz feature is an essential part of the AR application, providing a fun and effective way for users to test and solidify their knowledge. Through real-time feedback and a final score summary, users can track their progress and identify areas for improvement, making the AR experience both educational and engaging.

7.10 User Feedback section

To gather valuable user insights, the AR application includes a feedback submission feature that allows users to provide their experiences and suggestions. This input is essential for evaluating the app's performance and implementing necessary improvements.

Implementation Details:

- Feedback Input: A TMP_InputField from the TextMeshPro package is used for the feedback text input. This field ensures smooth and responsive text handling, providing users with an intuitive experience when typing their feedback. Once the feedback is entered, users can submit it by pressing the designated "Submit" button.
- Sending Feedback: The feedback system is linked to a Google Form, which serves as the repository for all user submissions. The form's URL is embedded within the application, ensuring that feedback is automatically directed to the correct destination. This setup allows the admin to easily access and review the collected responses in one centralized location.

```
public class forms : MonoBehaviour
{
    // Reference to the TMP_InputField where the user will enter their feedback
    [SerializeField] TMP_InputField feedbackSent;

    // The URL for the Google Form that we're going to send data to
    string URL = "https://docs.google.com/forms/u/0/d/e/1FAIpQLSdalxtzvCobFsfb1clJABF-CSFHAKCjWKcBiM1rZJazO5jpNA/formResponse";

    // This method is called when the user wants to send their feedback
    public void Send()
    {
        StartCoroutine(Post(feedbackSent.text));
    }
```

Code 7.8 Feedback code snippet

Input Field Setup: Within the script, a TMP_InputField is referenced to capture user feedback. This input field is linked to a feedbackSentvariable in Unity's Inspector, ensuring that the user's feedback is correctly mapped to the UI element and prepared for submission.

Google Form URL: The URL of the Google Form where feedback is collected is hardcoded within the script. This URL directs the application to the appropriate form, ensuring that all user feedback is centralized in one location for the admin to review.

Triggering the Submission: When users click the "Send" button, the Send() method is executed. This method initiates a coroutine that handles the process of submitting the feedback to the Google Form.

Post Method for Submission: The Post() coroutine is responsible for sending the feedback data to the Google Form. It creates a WWWForm object to store the user's input and maps this feedback to the corresponding field in the form. The mapping is done using the unique entry ID, such as "entry.1678875753", ensuring that the feedback is accurately submitted to the designated field in the Google Form.

```
IEnumerator Post(string s)
    {
        WWWForm form = new WWWForm();
        // Add the feedback text to the form, mapped to the specific entry field in the Google Form
        form.AddField("entry.1678875753", s);

        UnityWebRequest www = UnityWebWebRequest.Post(URL, form);

        // Wait until the web request is finished before continuing
        yield return www.SendWebRequest();
    }
```

Code 7.9 Post method code snippet

The feedback submission process ensures that the user's input is successfully recorded by waiting for the web request to complete before allowing further actions. This is handled by the coroutine, which guarantees that the feedback is properly sent and acknowledged before the user resumes other interactions within the app.

User Experience: The feedback mechanism is designed to be straightforward and user-friendly. Users can easily type their feedback into the text field and submit it by pressing the "Send" button. The system handles the submission process in the background, ensuring that the feedback reaches the correct destination without requiring additional user actions.

Admin Review: On the admin side, feedback is automatically collected and stored within the linked Google Form. The admin can review all submissions in real-time, providing a continuous stream of user insights. This feedback enables the admin to identify areas for improvement and make informed decisions about updates and enhancements to the AR application.

Importance of the Feedback Loop: By integrating this feedback system, a direct communication channel between the users and the development team is established. This feedback loop is critical for the ongoing development of the AR app, helping to ensure that it meets user expectations and delivers a valuable, user-centric experience.

8. The Final Application

Below are Screenshots of each page mentioned in the Workflow Diagram (Figure 5.3) for better understanding of the App.

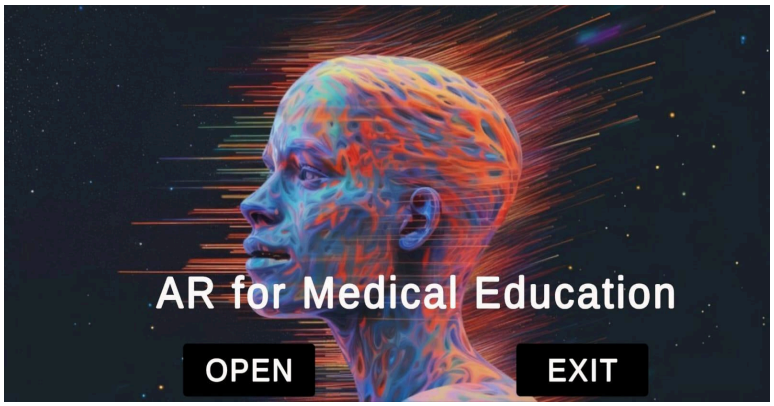


Figure 8.1 Entry page of the app

Figure 8.1 shows the entry page of the app that the user initially sees after opening the app. When the user clicks on the open button, they will be taken to the next menu page in the app.

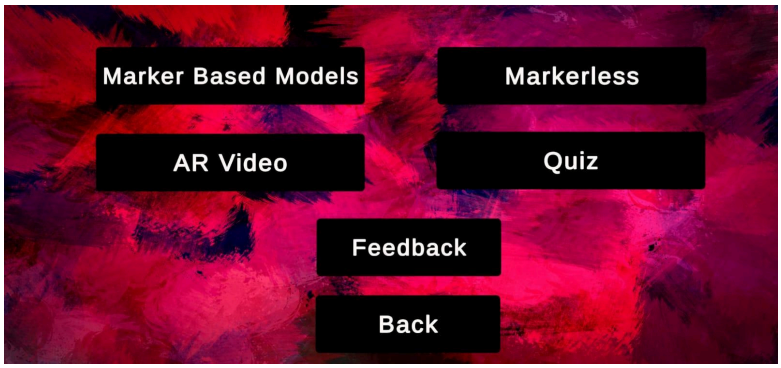


Figure 8.2 Main menu page of the app

As soon as the user clicks on the open page, they will be directed to the main menu page (Figure 8.2), where they have the options to choose from marker-based or markerless models, select an AR video, take a quiz, or provide feedback on the app based on their requirements.

Figure 8.3 Main menu page for marker based models

After clicking on the marker-based model, the user will be presented with options to view the augmented version of that particular model using the image target.

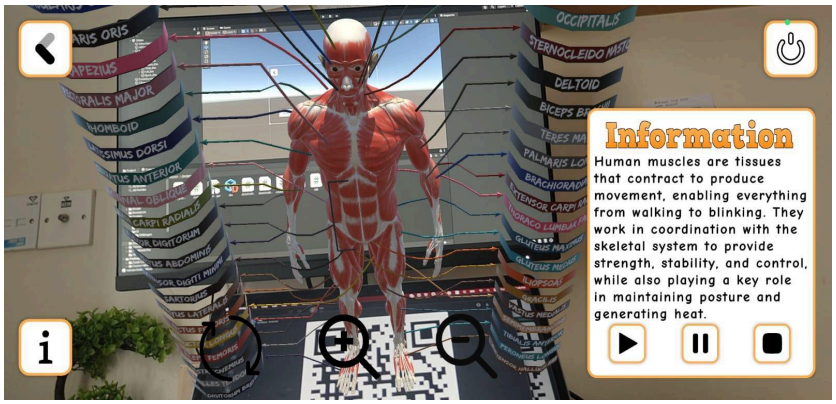


Figure 8.4 Augmented model of human muscle system

When the user selects the muscle system option from the choices given in Figure 8.3, the output will be as shown in Figure 8.4. Here, the user can zoom in, zoom out, and rotate the model for better visibility and understanding. Additionally, an audio guide provides a detailed explanation of the model.

Figure 8.5 Main menu page for markerless models

After clicking on the markerless model on the main menu page (Figure 8.5), the user will be presented with options to view the augmented version of the selected model using planes instead of an image target.

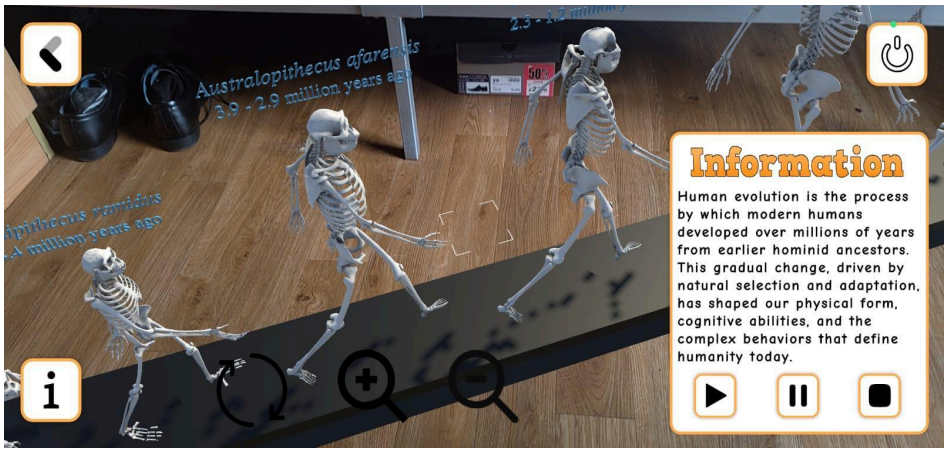


Figure 8.6 Augmented model of Evolution of Human

When the user selects the Evolution of human option from the choices given in Figure 8.5, the output will be as shown in Figure 8.6. Here, the user can zoom in, zoom out, and rotate the model for better visibility and understanding without the requirement of image target. Additionally, an audio guide provides a detailed explanation of the model.

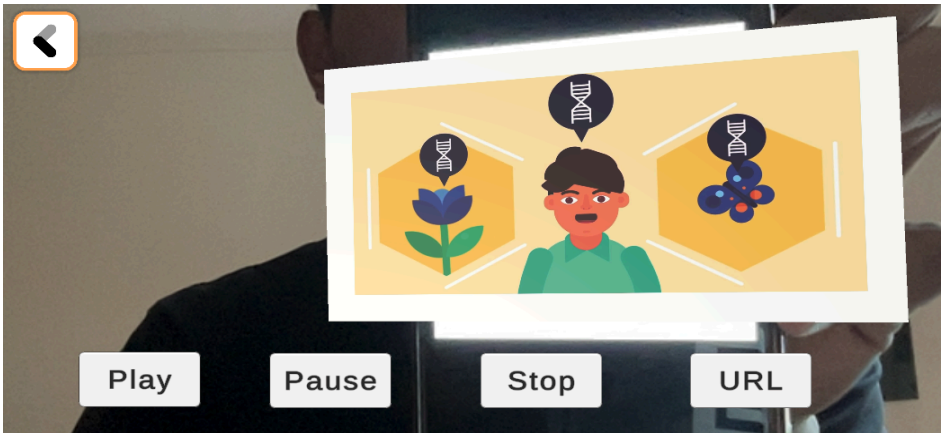


Figure 8.7 AR Video

After clicking on the AR video option as shown in Figure 8.2, the output will be as shown in Figure 8.7 after scanning the image target. You can play, pause, and stop the video playing virtually, and when you click on the URL button, it directs you to the original external source on YouTube.

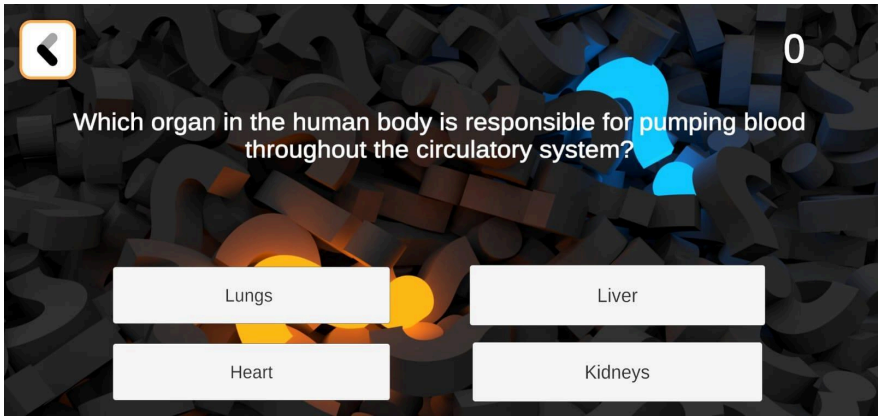


Figure 8.8 Quiz Template

When the user clicks on the quiz option from the main menu, they will be directed to the screen shown in Figure 8.8. The user must select the correct answer from four options, and for every correct answer, the score increases in the top right corner of the screen.

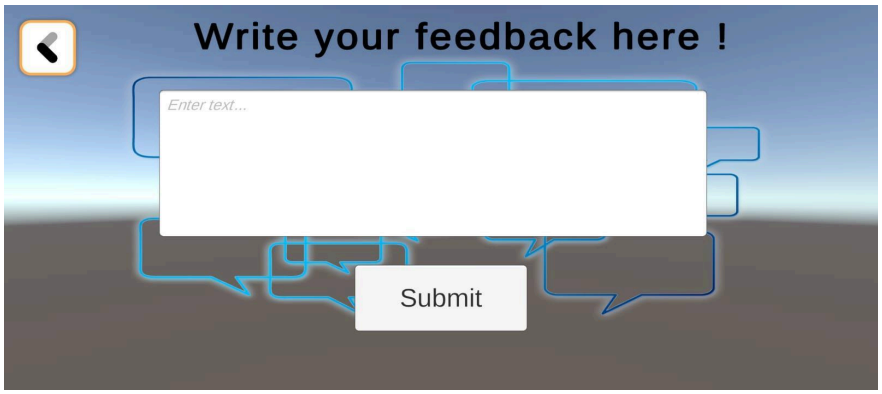


Figure 8.9 Feedback Section

Whenever the user wants to give feedback about the app, they can choose the feedback option from the main menu. After that, they will be directed to the feedback section, as shown in Figure 8.9. After entering feedback in the text field and clicking the submit button, the feedback will be sent and updated in the Google Form.

9. Testing

Testing was a pivotal step in the development of the AR application to ensure functionality, usability, and compatibility across diverse devices and environments. The testing phase focused on verifying the integrity of all features, user interaction, and AR-specific components.

9.1 Functional Testing

Purpose: The primary objective of functional testing was to confirm that all features of the AR application performed as expected. Key functionalities tested included image recognition, interactive features such as zooming, rotating 3D models, and user inputs.

Process: Friends and colleagues were invited to interact with the app, testing each feature extensively. For example, when rotating or zooming a 3D model, the app's response was expected to be smooth and accurate. Any discrepancies or issues encountered were addressed by adjusting the code accordingly, ensuring that all functionalities operated seamlessly.

9.2 Usability Testing

Purpose: Usability testing was conducted to assess the overall user experience, ensuring that the interface was intuitive and easy to navigate, particularly for the target audience medical students and instructors. The goal was to ensure the app provided a positive user experience that met the needs of its users.

Process: A group of friends participated in the usability tests, offering feedback on the layout, navigation, and ease of use. Their insights were invaluable in refining the design to enhance accessibility and intuitiveness. This testing helped identify areas where improvements could be made to align the user interface with the expectations of the intended audience.

9.3 Compatibility Testing

Purpose: Compatibility testing was essential to ensure that the AR application performed consistently across various devices, screen sizes, and operating systems, avoiding any inconsistencies in user experience.

Process: The app was tested on multiple smartphones and tablets running different versions of Android and iOS. Focus areas included screen resolution, aspect ratios, and hardware variations, ensuring that the app functioned correctly and looked consistent across all devices. Any compatibility issues identified during this phase were resolved to ensure a uniform experience across different platforms.

9.4 AR-Specific Testing

Purpose: Since the application heavily relies on AR technology, AR-specific testing was crucial to verify that marker-based and markerless tracking worked effectively in various environments and conditions.

Process: The Vuforia engine's image recognition capabilities were tested under different lighting conditions, varying image quality, and a range of physical settings. These tests ensured that both marker-based and markerless AR tracking remained reliable and accurate in real-world

scenarios. Ground plane detection was also tested to verify that digital objects were anchored correctly on flat surfaces, providing users with a stable AR experience.

10. Conclusion and Future Scope

The development of this Augmented Reality (AR) application represents a significant step forward in enhancing medical education through innovative technology. The primary objective was to create an interactive learning tool that effectively bridges the gap between theoretical knowledge and practical understanding. Through the integration of Vuforia and Unity, the project successfully developed both marker-based and markerless AR experiences that are not only functional but also engaging.

The inclusion of interactive features, such as object rotation, zooming, and quizzes, has contributed to a more immersive and dynamic learning experience. The deployment across both Android and iOS platforms ensures broad accessibility, while the user feedback mechanism facilitates ongoing refinement and improvement.

Overall, the project has achieved its intended goals, laying a robust foundation for future enhancements. The application is set to become a valuable resource in medical education, with the potential to significantly impact how students and professionals interact with complex medical concepts.

Future Scope

Looking ahead, there are several promising avenues for expanding and enhancing the capabilities of this AR application:

1. Enhanced Interactivity:
    - Future iterations could incorporate more advanced interaction techniques, such as gesture-based controls and voice commands, further improving the user experience and accessibility.
2. Expanded Content Library:
    - There is potential to expand the content library to include a broader range of medical models and educational topics, thereby increasing the application's utility across various medical disciplines.
3. Multi-User Collaboration:
    - Introducing multi-user features would enable collaborative interactions, allowing multiple users to engage with the same AR model simultaneously, fostering a more interactive and cooperative learning environment.
4. Localization and Multilingual Support:
    - Adding multilingual support and localized content would make the application more accessible to a global audience, thereby extending its reach and impact.

5. Augmented Reality in Surgical Training:
    - The application could be extended to include AR modules for surgical training, allowing students to practice procedures in a simulated environment, which would be a significant addition to its educational capabilities.

In conclusion, while the project has met its current objectives, it also opens up numerous opportunities for future development. The groundwork has been laid for this AR application to evolve into an even more powerful educational tool, with the potential to make a lasting impact in the field of medical education.

11. References

[1] Craig, A. B. (2013). Understanding augmented reality: Concepts and applications.

[2] Fayda-Kinik, F. S. (2023). Augmented Reality in Education: An Overview of Research Trends. *Online Submission*.

[3] Altinpulluk, H. (2019). Determining the trends of using augmented reality in education between 2006-2016. *Education and Information Technologies*, *24*(2), 1089-1114.

[4] Hejlsberg, A., Torgersen, M., Wiltamuth, S., & Golde, P. (2008). *The C# programming language*. Pearson Education.

[5] Xiao, C., & Lifeng, Z. (2014, June). Implementation of mobile augmented reality based on Vuforia and Rawajali. In *2014 IEEE 5th International Conference on Software Engineering and Service Science* (pp. 912-915). IEEE.

[6] Kim, S. L., Suk, H. J., Kang, J. H., Jung, J. M., Laine, T. H., & Westlin, J. (2014, March). Using Unity 3D to facilitate mobile augmented reality game development. In *2014 IEEE World Forum on Internet of Things (WF-IoT)* (pp. 21-26). IEEE.

[7] Haas, J. K. (2014). A history of the unity game engine.

[8] Arusoaie, A., Cristei, A. I., Chircu, C., Livadariu, M. A., Manea, V., & Iftene, A. (2010, September). Augmented reality. In *2010 12th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing* (pp. 502-509). IEEE.

[9] Radu, I. (2012, November). Why should my students use AR? A comparative review of the educational impacts of augmented-reality. In *2012 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)* (pp. 313-314). IEEE.

[10] Sarosa, M., Chalim, A., Suhari, S., Sari, Z., & Hakim, H. B. (2019, November). Developing augmented reality based application for character education using unity with Vuforia SDK. In *Journal of Physics: Conference Series* (Vol. 1375, No. 1, p. 012035). IOP publishing.

[11] Kalmkar, S., Mujawar, A., & Liyakat, D. K. K. S. (2022). 3D E-Commers using AR. *International journal of information Technology and computer engineering (IJITC)*, *2*(6), 18-27.

[12] Manning, J., & Buttfield-Addison, P. (2017). *Mobile Game Development with Unity: Build Once, Deploy Anywhere*. " O'Reilly Media, Inc.".

[13] Hameed, Q. A., Hussein, H. A., Ahmed, M. A., & Omar, M. B. (2022). Development of Augmented Reality-based object recognition mobile application with Vuforia. *Journal of Algebraic Statistics*, *13*(2), 2039-2046.

[14] Nilsson, E. G. (2009). Design patterns for user interface for mobile applications. *Advances in engineering software*, *40*(12), 1318-1328.

[15] https://www.canva.com/

[16] https://sketchfab.com/feed