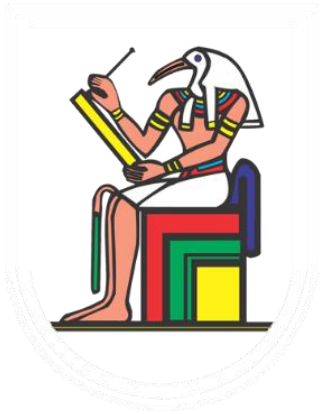
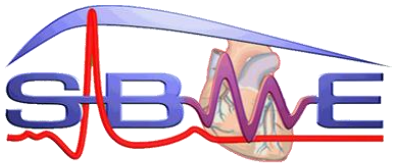




Systems & Biomedical Engineering Dept.

2020

Interactive Education using Augmented Reality powered Mobile Application

Ahmad Said, Ahmad Yasser, Hazem Muhammad,
Samar Abd-Elfattah, Hala Tarek

Under The Supervision of:

Dr. Sahar Fawzy

Dr.Ahmad El-bialy

Table of Contents

ACKNOWLEDGMENT

"I hear and I forget. I see and I remember. I do and I understand."

First, we want to thank ALLAH for everything; for supporting us in every moment, helping us and giving us the patience, strength and ability to manage our resources to launch.

The satisfaction that accompanies the successful completion of any task would be incomplete without the mention of people whose ceaseless cooperation made it possible, whose constant guidance and encouragement crown all efforts with success.

We are grateful to our project coordinator PhD. Sahar Fawzy and our project guide PhD. Ahmad El-Bialy for their guidance, inspiration and constructive suggestions that helped us in the preparation of this project.

We also thank our Head of SBME department PhD.Ayman El-Deeb for her knowledgeable insight and guidance.

Last but not the least; we thank our fellow students who have helped in successful completion of this project.

At last, we would like to thank Information Technology Industry Development Agency (ITIDA) for accepting our proposal and funding our project.

ABSTRACT

Students get bored of receiving the information as it is as they see it in the book without interacting with it or encountering it in their real life, so understanding the information and keeping it has become difficult for them as well.

Technology has rapidly conquered the world, as the developed world has dispensed with blackboards and chalk, and the world's attention has become tending to use technology in various areas of life such as education, health, travel, food, cleaning and other various fields.

Our app tries to make learning easier and more fun, and all this using augmented reality technology. Through the application, the student can interact with the curricula, hear information and see three-dimensional objects without relying on his imagination only. We achieved this through Markerless AR, object manipulation and animation and information panels and audio.

C H A P T E R 1
Introduction

Education

Certainly, the importance of education cannot be ignored by any country, and in today's world education has become of the utmost importance to achieve economic and social development for any country, and it is not clear to anyone in our time the importance of education and learning, especially for countries that consider ignorance, development, and identity a trinity of the problem with regard to employment, This form is an essential axis for their advancement

The education is conducted anywhere in the world on four axes or basic elements, namely:

Teacher, Student, School or university (place of learning), Curriculum teaching.

One of The problems of education in Egypt is the curricula, as it also suffers from the way it is presented and communicated to the student and made him imagine it through the use of images or sounds.

Curriculum problems

1- Reliance on memorization and memorization only, even in scientific subjects, statistics, mathematics and physics problems.
2- Fill out the curriculum without focusing on specific points.
3- By adopting the curriculum on the theoretical side only, not the practical side.
4- The insufficiency of the curricula for continuous scientific renewal, the old curricula, and the lack of qualitative updating of the books we study, even if there is some official modernization.
5- A clear gap between curriculum contents and labor market requirements.

These are the most important problems that education suffers from in Egypt, and it can be said that these problems are not for one reason, but for a large and overlapping group of reasons.

Among the proposed solutions that were used

1- Distance education (self-education) using educational materials and aids that are capable of individual education
2- Improving the level of teachers, providing it with modern educational means, and working to train and develop their educational capabilities

One of the most important reasons that made us think of education as a goal of our project is having tablets with students as these tablets have many benefits in making education fun, enjoyable, and easier for students.

From now on, it is okay if one of your children forgets to take his school bag to school. The tablet, whose size does not exceed the palm of the hand, sings a bag full of books, notebooks, and paper clutches, whatever the operating system, the tablet is a versatile tool and some educational applications can be added that will make it ready to start teaching your children through it.

Technology has rapidly conquered the world, as the developed world has dispensed with blackboards and chalk, and the world's attention has become tending to use technology in various areas of life such as education, health, travel, food, cleaning and other various fields.

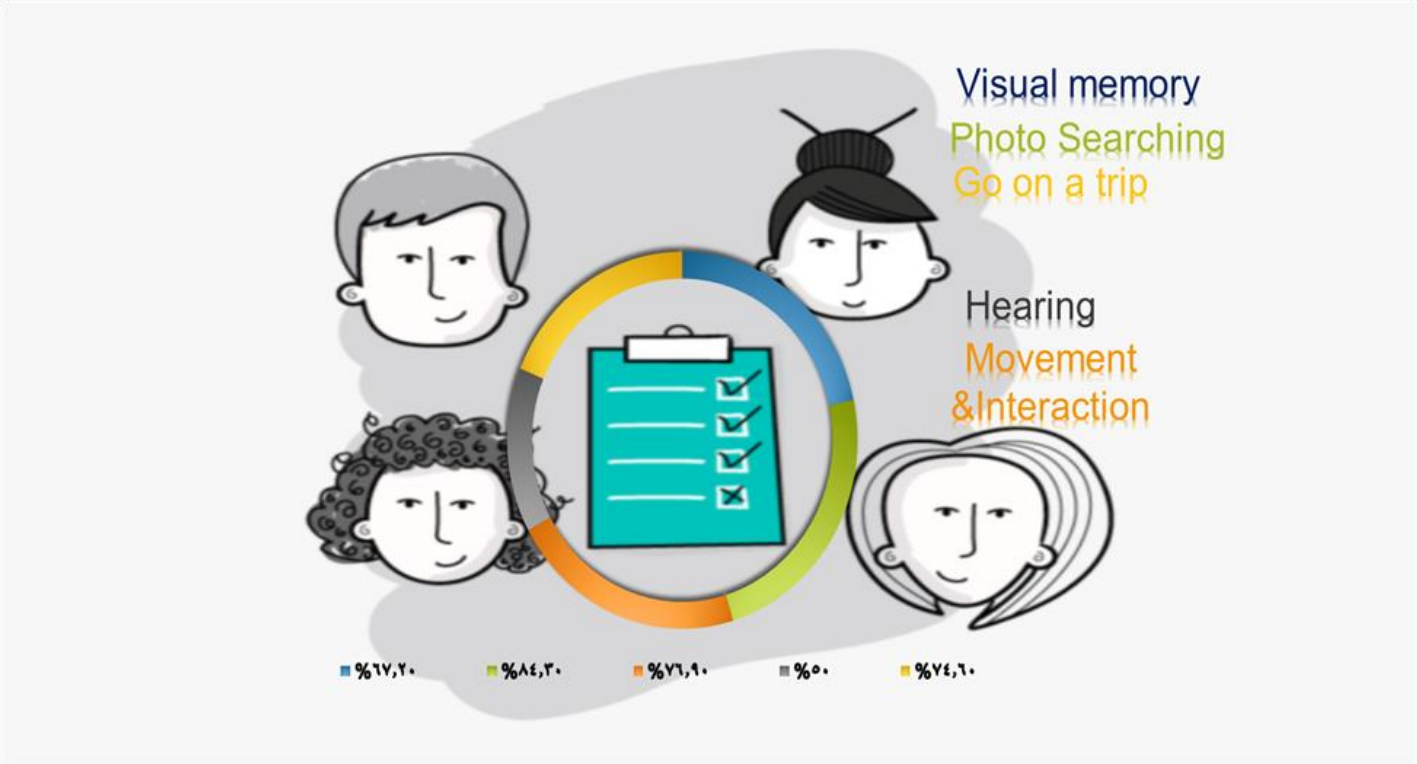
Most important benefits of using the tablets

1- Easy to use
2- more fun
3- less cost
4- faster delivery of information
5- better student skills development

Each student has a personal approach that differs from his peers, some students are slow learners and some are quick to learn and respond to the teacher, the teacher can use tablets to use tools that communicate information to all groups of students and help them in understanding and comprehension and presenting the educational material in different and enjoyable ways.

We have done a questionnaire to see whether the use of methods such as pictures and sounds makes it easier for the student to understand his curricula and their views were that this is actually better for them because it helps to understand information faster and better

And its results are as follows:



A 76.2 % from students have visual memory, a 84.3% always look for pictures to see information, a 50% that prefers to listen to information besides seeing it, and a 74.6%that wants to interact with the things they study.
This demonstrated that most students prefer to see and interact with things besides listening to them rather than relying on visualization only.

Incorporating the Five Senses to Stimulate Learning

Kids learn in a variety of different ways. Some may be auditory learners, while others may be visual learners. Each child is different so it is important to teach in a multitude of ways to engage each and every sense.

Our five senses work hard every minute of every day, but if we can control the senses and direct them towards certain activities it can be beneficial for our ability to learn and grow. With a multisensory environment, students will learn quicker, faster, and retain information longer, so this is what we are trying to achieve in our application.

Edger Dale cone

Edgar Dell classified teaching aids on the basis of experiences provided by the media in his book Audio-visual Teaching Methods in a cone shape called the Cone of Experience.

The purpose of Cone is to offer a range of experiences ranging from direct experience to symbolic communication. The cone was built on a chain that begins with the physical things and ends with the abstract things. Del Dale believed that abstract symbols and ideas could be understood by the learner and remembered more easily if they were based on tangible experiences. At the top of the cone he placed abstract experiences such as verbal and visual symbols, and at the base of the cone were tangible, sensory and realistic experiences. And he arranged the other educational aids in this cone according to the proximity or distance of the experiences that create them from abstraction or realism and not according to their difficulty, ease, importance, or any other criterion, on the basis that the farther from the base of the cone towards its summit, the fewer real direct experiences and the more aspects of abstraction.

This shows that the more interaction there is with the things that are taught, the easier it is to understand and remember them easily, and the more they rely on verbal and visual symbols only, they can be forgotten faster and also it is possible to form a false perception of the information.

C H A P T E R 2
Augmented reality

Introduction

This Chapter aims to give information about augmented reality and the achievement of the experience in an android mobile application through different available software development kits.

Definition

Augmented Reality offers a new way to interact with the physical world. It creates a modified version of the reality, enriched with virtual information, on the screen of desktop computer or mobile device. Merging and combining the virtual and the real can leverage a totally new range of user experience, going beyond what common apps are capable of.

There are different types of Augmented Reality applications

Marker Based Augmented Reality

It makes use of a camera and a virtual marker to activate some AR components just when the marker is identified by the camera.

Marker less Augmented Reality

Uses GPS, accelerometers and digital compasses embedded in a smartphone to offer AR content based on the user location. This can be used for example to make mapping systems more interactive.

Superimposition Based Augmented Reality

This type of AR uses object recognition to either partially or completely change the view of an object.

Projection Based Augmented Reality

Light is sent on a real-world object to create a user interface on which the user can interact with (the user interaction with the surface is detected by examining how the projected area is altered by this interaction).

- ✓ The first 3 types can be used in mobile AR applications.

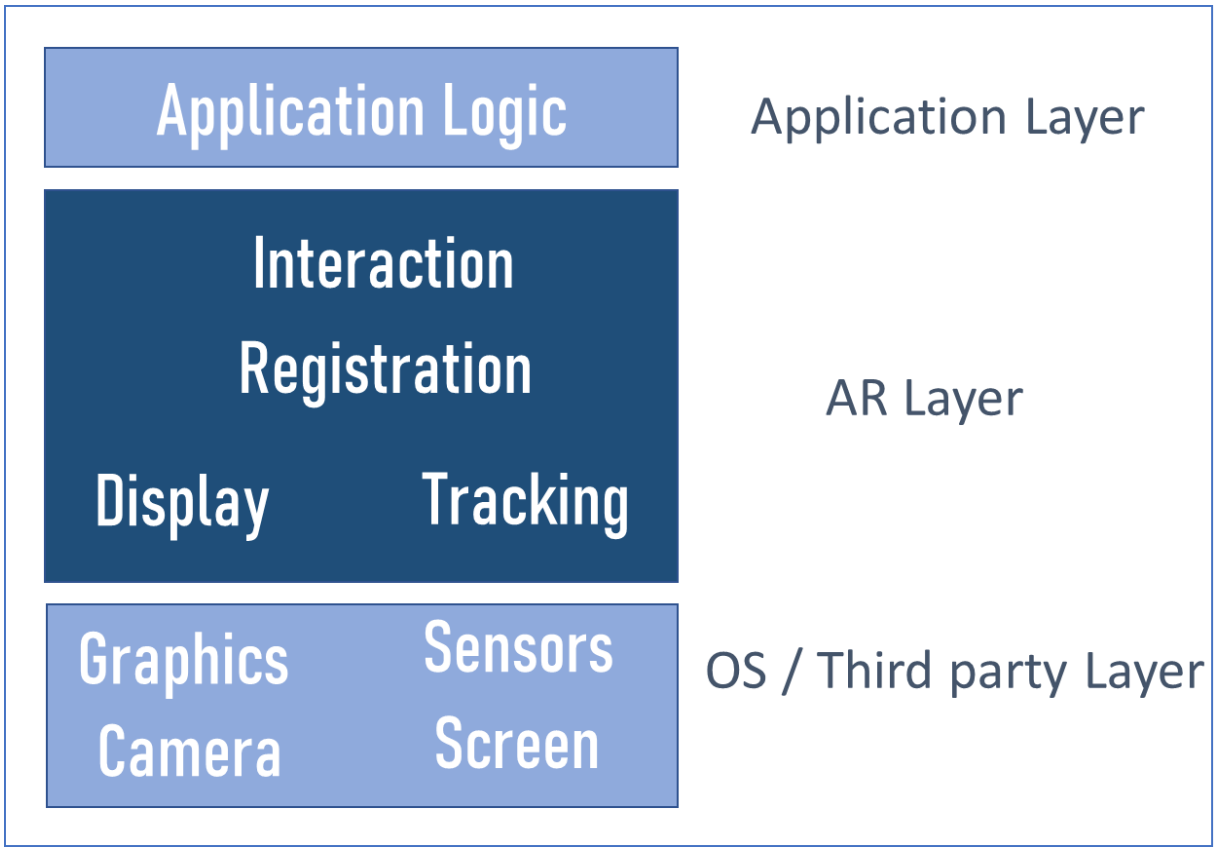
AR Software Components

An AR application can be structured in three layers

The application layer

The AR layer

The OS/Third Party layer.



The application layer corresponds to the domain logic of the application. If we want to develop an AR game, anything related to managing the game assets (characters, scenes, objects) or the game logic will be implemented in this specific layer.

The AR layer corresponds to the instantiation of each of the concepts of display, registration, tracking and interaction in terms of software, as a modular element, a component, or a service of the AR layer.

AR needs to know more about real and virtual content. It needs to know where things are in space (registration) and follow where they are moving (tracking).

Registration is basically the idea of aligning virtual and real content in the same space in real time.

Tracking is a major software component for an AR application, which provides spatial information to the registration service, it is a complex and computationally intensive process in any AR application.

Building a rich AR application needs interaction between environments; otherwise we end up with pretty, 3D graphics that can turn boring quite fast. AR interaction refers to selecting and manipulating digital and physical objects and navigating in the augmented scene. Rich AR applications allow to use objects which can be on a table, to move some virtual characters, hands usage to select some floating virtual objects, we have dedicated a whole chapter to give more information about interaction concepts

The OS/Third Party layer corresponds to existing tools and libraries which don't provide any AR functionalities, but will enable the AR layer. For example, the Display module for a mobile application will communicate with the OS layer to access the camera to create a view of the physical world. On Android, the Google Android API is part of this layer.

Mobile AR

Mobile AR sometimes refers to any transportable, wearable AR system that can be used indoors and outdoors. We will look at mobile AR with the most popular connotation used today—using handheld mobile devices, such as smartphones or tablets. With the current generation of smartphones, two major approaches to the AR system can be realized. These systems are characterized by their specific registration techniques and, also, their interaction range. They both enable a different range of applications. The systems, sensor-based AR and computer vision-based AR, are using the video see-through display, relying on the camera and screen of the mobile phone.

Sensor-based AR

The first type of system is called sensor-based AR and generally referred to as a GPS plus inertial AR (or, sometimes, outdoor AR system). Sensor-based AR uses the location sensor from a mobile as well as the orientation sensor. Combining both the location and orientation sensors delivers the global position of the user in the physical world.

The location sensor is mainly supported with a GNSS (Global Navigation Satellite System) receiver. One of the most popular GNSS receivers is the GPS (maintained by the USA), which is present on most smartphones.

There are several possible orientation sensors available on handheld devices, such as accelerometers, magnetometers, and gyroscopes. The measured position and orientation of handheld device provides tracking information, which is used for registering virtual objects on the physical scene. The position reported by the GPS module can be both inaccurate and updated slower than user movement.

The advantage of this technique is that the sensor-based ARs are working on a general scale around the world, in practically any physical outdoor position (such as if user are in the middle of the desert or in a city). One of the limitations of such systems is their inability to work inside (or work poorly) or in any occluded area (no line-of-sight with the sky, such as in forests or on streets with high buildings all around).

COMPUTER VISION-BASED AR

The other popular type of AR system is computer vision-based AR. The idea here is to leverage the power of the inbuilt camera for more than capturing and displaying the physical world (as done in sensor-based AR). This technology generally operates with image processing and computer vision algorithms that analyse the image to detect any object visible from the camera. This analysis can provide information about the position of different objects and, therefore, the user. The advantage is that things seem to be perfectly aligned. The current technology allows the recognition different types of planar pictorial content, such as a specifically designed marker (marker-based tracking) or more natural content (marker-less tracking). One of the disadvantages is that vision-based AR is heavy in processing and can drain the battery really rapidly. Recent generations of smartphones are more adapted to handle this type of problem, being that they are optimized for energy consumption.

SDKs

An augmented reality SDK (software development kit) is the core technological software engine that powers the development and creation of new AR apps and experiences. The role of the AR SDK is to perform the non-trivial task of fusing digital content and information with the real world. The capabilities of the SDK will ultimately underpin the features and functionality within the AR application, so it's essential to choose the correct platform based on the requirements of the project.

The AR SDK is responsible for many components of the applications, which are currently available, including content rendering, AR tracking, and scene recognition. Content rendering relates to the digital information and 3D objects that can be overlaid on top of the real world, tracking represents the 'eyes of the application,' and the scene recognition element acts as the central nervous system of the application. Each AR SDK will be equipped with its own unique properties that will enable AR developers to recognize, render, and track the application in the most optimal manner possible.

1. ARKit

In 2017, Apple released iOS 11, and the subsequent launch of ARKit witnessed arguably the most seismic event in the history of augmented reality technology. ARKit is a unique framework that enables brands and developers to design and create unparalleled experiences for compatible iPhone and iPad devices (compatible iPhone's and iPad's must be equipped with an A9 processor or above). The ARKit SDK functions in the same way as

most AR SDK's function, by enabling digital information and 3D objects to be blended with the real world but offers largely unparalleled accessibility in terms of the number of existing devices that it supports.

ARKit can be run on any device equipped with an Apple A9, A10, or A11 processor and utilizes VIO (Visual Inertial Odometry) to track the surrounding environment with seamless levels of accuracy. VIO enables the ARKit to combine Core Motion data with camera sensor data and provides the ability to develop applications that can detect horizontal planes (floors and tables) and vertical planes (walls). This enables the ARKit to accurately understand the dynamics and make-up of a particular scene and provides the ability to place 3D objects and overlay digital information in a contextually relevant way (for example, because ARKit understands the difference between a floor and a table, it knows to place a bottle of water on the table, rather than the floor).

It allows developers to create applications using ARKit and associated optimizations via third-party 3D engines such as Unity and Unreal Engine

ARKit provides the following functionalities:

- SLAM tracking (simultaneous localization and mapping) and sensor fusion
- Ambient lighting estimation
- Scale estimations
- Vertical and horizontal plane estimation with basic boundaries
- Stable and fast motion tracking

2. ARCore

ARCore is Google's proprietary augmented reality SDK. Similar to ARKit, it enables brands and developers to get AR apps up and running on compatible Google smartphones and tablets. One of the most notable features of ARCore is that it also supports iOS-enabled devices and gives developers access to users across both platforms.

ARCore possesses three significant features that enable developers to merge the real world with the virtual:

- Light estimation: Estimates real-world lighting conditions
- Environmental understanding: Detects the size and location of vertical, horizontal and angled surfaces
- Motion tracking: Understands the phone's position relative to its surroundings

The entire ARCore offering is heavily built around two key elements: real-time tracking and the calculation of the device's location, paired with the integration of virtual objects with the real-world environment. This enables development of rich and immersive mobile supported AR experiences, enabling 3D objects, text, and digital information directly into the surrounding real-world environment. ARCore is free to use for developers and supports a range of Android-enabled (and iOS enabled) smartphones and tablets including Samsung Galaxy and Google Pixel, plus many more.

3. UNITY'S AR Foundation

Unity's AR Foundation fundamentally provides a layer of abstraction to ARCore and ARKit. The framework supports devices running Android 7.0 or later and iOS 11.0 or later.

AR Foundation generally supports features provided in ARCore and ARKit. The framework provides additional support by allowing developers to access native ARCore and ARKit features directly via their respective Unity packages. AR Foundation is a great solution for incorporating AR features shared by ARCore and ARKit while also targeting features specific to the platform or framework.

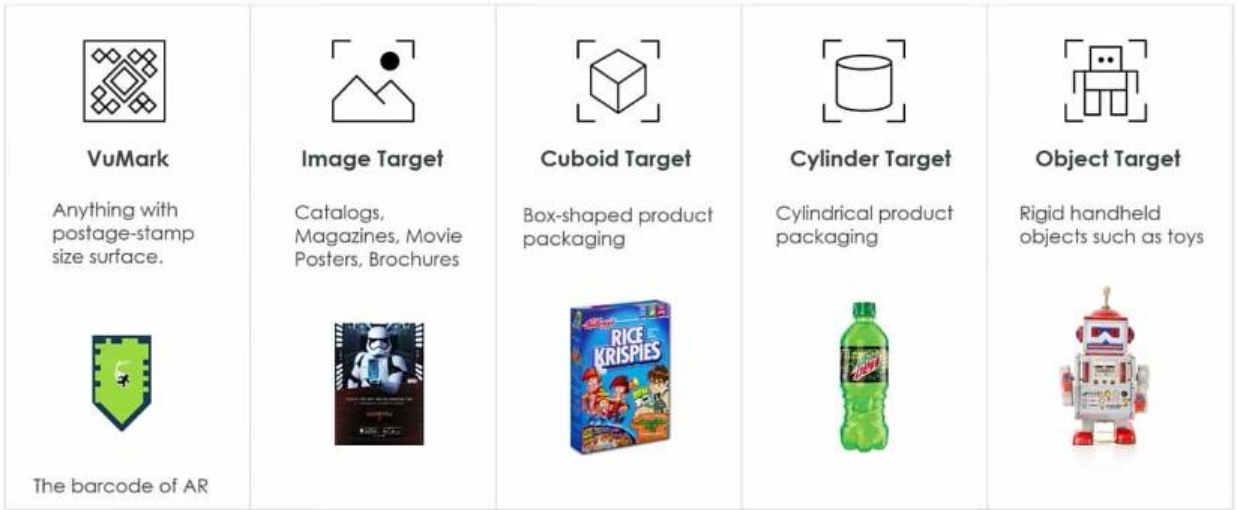
4. Vuforia

Vuforia is an augmented reality SDK that enables app developers to quickly spin-up high fidelity, mobile-centric, immersive AR experiences.

The Vuforia SDK leverages computer vision technology to identify and track image targets and 3D objects in real-time. This functionality allows orient and place virtual objects, including 3D models and other content, in relation to the real-world environment. 3D models and digital information can then be overlaid on top of the real-world scene and viewed in relation to the environment via an AR-enabled smartphone or tablet.

The Vuforia augmented reality SDK is capable of supporting a wide variety of 3D and 2D targets, including 3D multi-target configurations, markerless image targets, and fiducial markers referred to as a 'VuMark.' Some of the additional features in the Vuforia SDK include localized occlusion detection using virtual buttons, the capacity to develop and calibrate target sets at runtime, and runtime target image selection.

Vuforia provides API's (application programming interfaces) in Java, C++ and Objective C++, and .NET via an extension of the Unity game engine. With this in mind, Vuforia SDK is capable of supporting both native development for iOS and Android and the development of AR apps and prototypes in Unity that can easily be ported across both platforms.



5. Wikitude

Wikitude is an SDK specifically designed for developing mobile AR apps and prototypes. The company was founded back in 2008 in Salzburg, Austria. When the Wikitude SDK was initially launched, the platform was designed with a core objective: to enable AR developers to create location-centric augmented reality experiences through the Wikitude World Browser app. Fast forward to 2012 and Wikitude repositioned its core technology offering by launching the Wikitude SDK with geolocation features, tracking, and image recognition all baked directly into the core platform.

The Wikitude SDK allows to create immersive mobile AR experiences in the shortest possible time frame. The Wikitude SDK also includes functionality such as 3D model rendering, location-based AR, and video overlay. The company latterly rolled out SLAM technology (simultaneous localization and mapping), which facilitates seamless object tracking and recognition alongside markerless instantaneous tracking.

The Wikitude SDK works across multiple platforms and is currently available to support Windows OS, iOS, Android and a number of HUD's (heads up displays).

Wikitude claims to be the first ever SDK to focus entirely on a location-based approach to creating augmented reality applications for cross-platform mobile AR development and smart eyewear devices.

6. EasyAR

The EasyAR SDK is available across two-tiered pricing packages: EasyAR SDK Basic and EasyAR SDK Pro.

The basic package provides enhanced API's, workflow, and increased compatibility. The Pro package is brand new and is equipped with exclusive features that are not available in the basic package. The basic package is free to developers looking to develop AR applications and supports the Java API for Android, the Swift API for iOS, and also supports Windows OS. Some of the additional features supported by the basic package include video playback, transparent video playback, QR code scanning, and comprehensive Unity integration.

The EasyAR Pro package is available with all of the features associated with the free package on the basic version of the platform, plus support for SLAM, 3D object tracking, screen recording and simultaneous detection and tracking for multiple types of targets.

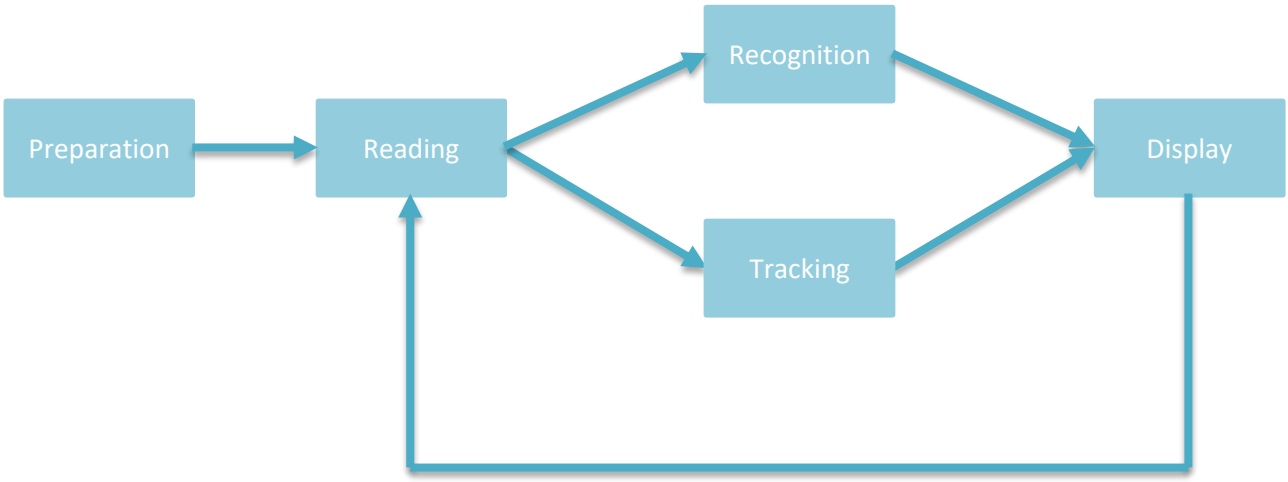
The core feature offering of the EasyAR Pro package focuses around the following: SLAM (including Monocular real-time 6 DOF camera pose tracking and full mobile compatibility), 3D-object tracking (equipped with the ability to recognise and track a common 3D object complete with textures in real-time), screen recording (provides a simple and efficient way of recording AR content), planar image tracking (ability to track and identify planar images in real-time), a concise API that integrates with all major mobile AR platforms and content, and interaction support in order to display the most compelling AR content with additional functionality.

AR Experience in our application

We used 2 SDKs: Vuforia SDK for marker based AR and AR foundation SDK for Marker-less AR.

Marker based AR

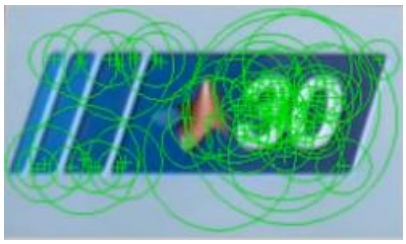
To achieve marker based AR experience in our application we needed to understand how the SDK works in steps.



Block Diagram Steps of Marker Based AR

1) Preparation

a) Target Image is loaded
b) Extract Features from target image
c) Initialize mobile device Camera



Target Image

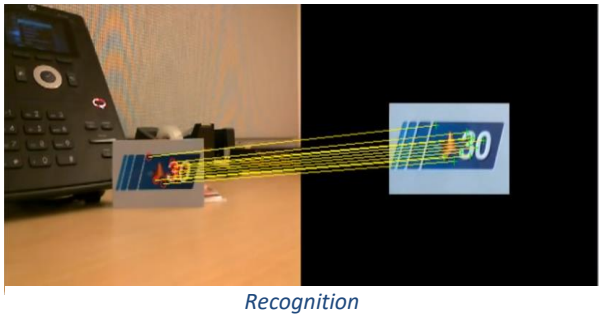
2) Reading

a) Capture mobile Camera Frame
b) Extract Frame Features

Reading

3) Recognition

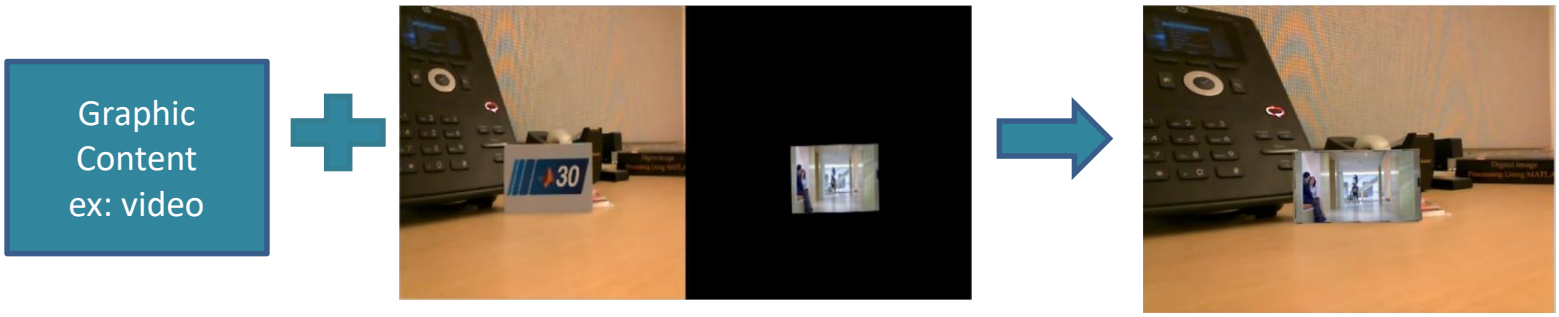
a) Match Features
b) Remove any Outliers
c) Find Transformation



Recognition

4) Display

a) Apply Transform on graphic contents
b) Combine it with camera frame



Display

5) Tracking

a) Once the AR Target is detected, its location is provided as an input to the Tracking Module.
b) Tracking module Finds Transformation between frames and updates it.

Tracking

- **These steps are achieved by applying the following Algorithms:**

1. Feature Detection , Description and Matching :
   SIFT or SURF or FAST or BRIEF or ORB or BRISK
2. Estimation of Geometric Transformation and removing any outliers : RANSAC
3. Tracking :
   Kanade–Lucas–Tomasi feature tracker (KLT)

- **AR Feature Detection and Description methods Comparison**

	SIFT	SURF	FAST	BRIEF	ORB	BRISK
Year	1999	2006	2006	2010	2011	2011
Feature Detector	Difference of Gaussian	Fast Hessian	Binary comparison	N/A	FAST	FAST or AGAST
Spectra	Local gradient magnitude	Integral box filter	N/A	Local binary	Local binary	Local binary
Orientation	Yes	Yes	N/A	No	Yes	Yes
Feature Shape	Square	HAAR rectangles	N/A	Square	Square	Square
Feature Pattern	Square	Dense	N/A	Random point-pair pixel compares	Trained point-pair pixel compares	Trained point-pair pixel compares
Distance Function	Euclidean	Euclidean	N/A	Hamming	Hamming	Hamming
Robustness	6 Brightness, rotation, contrast, affine transforms, scale, noise	4 Scale, rotation, illumination, noise	N/A	2 Brightness, contrast	3 Brightness, contrast, rotation, limited scale	4 Brightness, contrast, rotation, scale
Pros	Accurate	Accurate	Fast, real-time applicable	Fast, real-time applicable	Fast, real-time applicable	Fast, real-time applicable
Cons	Slow, compute-intensive, patented	Slow, patented	Large number of interest points	Scale and rotation invariant	Less scale invariant	Less scale invariant

Marker-less AR

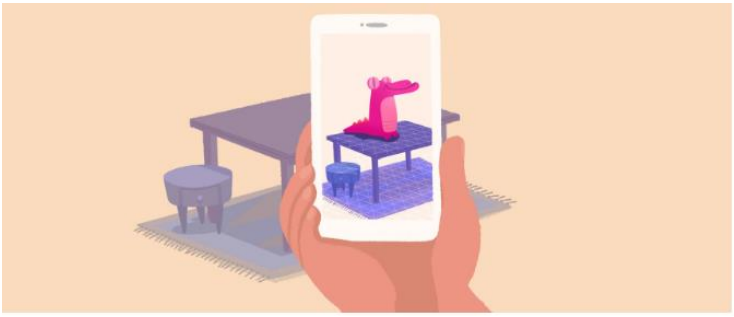
SLAM (Simultaneous Localization and Mapping) is a technology used to achieve marker-less AR it understands the physical world through feature points.

The device's sensors collect visual data from the physical world in terms of feature points. These points help the machine distinguish between floors, walls and any barriers.

Measurements are constantly taken as the device moves through the surroundings and SLAM takes care of the inaccuracies of the measurement method by factoring in 'noise'. Different sensors use different algorithms. SLAM largely makes use of mathematical and statistical algorithms. One of which is the Kalman filter. Kalman filter takes into account a series of measurements over time, instead of just a single one. It then predicts the position of unknown variables, in our case – unknown points on 3D objects in the device's point of view.

Inertial measurement unit (IMU) is the sensor used in mobile devices to estimate the pose (position and orientation) of the camera relative to the world over time.

Clusters of feature points that appear to lie on common horizontal or vertical surfaces, are likely to be tables or walls, thus surfaces and planes are detected, flat surfaces without texture, such as a white wall, may not be detected properly This makes it possible for AR applications to recognize 3D Objects & Scenes, as well as to Instantly Track the world, and to overlay digital interactive augmentation.



Marker Based AR in our application (Vuforia SDK)

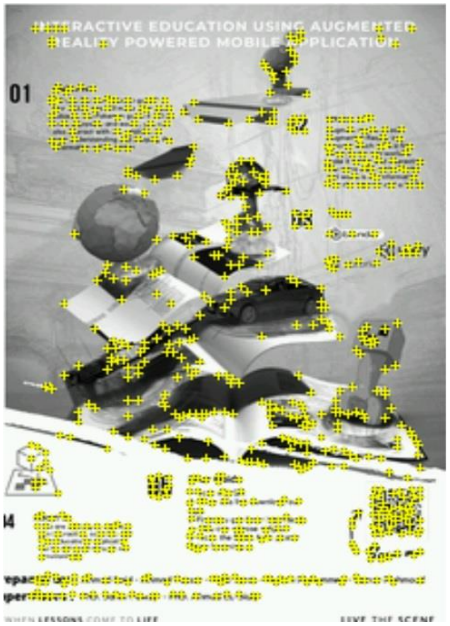
We used Vuforia SDK and Image targets as marker to achieve Marker based AR.

Image Targets represent images that Vuforia Engine can detect and track. The Engine detects and tracks the image by comparing extracted natural features from the camera image against a known target resource database. Once the Image Target is detected, Vuforia Engine will track the image and augment the graphic content.

We used our poster as Image Target:

The poster image is uploaded to the Vuforia Target Manager (which is a web-based tool that enables creating and managing target database)

The Target Manager processes the images to generate both data and visual representations of the target's features. It also reviews the target's expected detection and tracking performance rating. The Image Targets can thereafter be downloaded as a package suitable for integration with Unity.



Target ID: 4b25e533df4e4322a4bb197c5793bcd5

Augmentable: ★★★★★

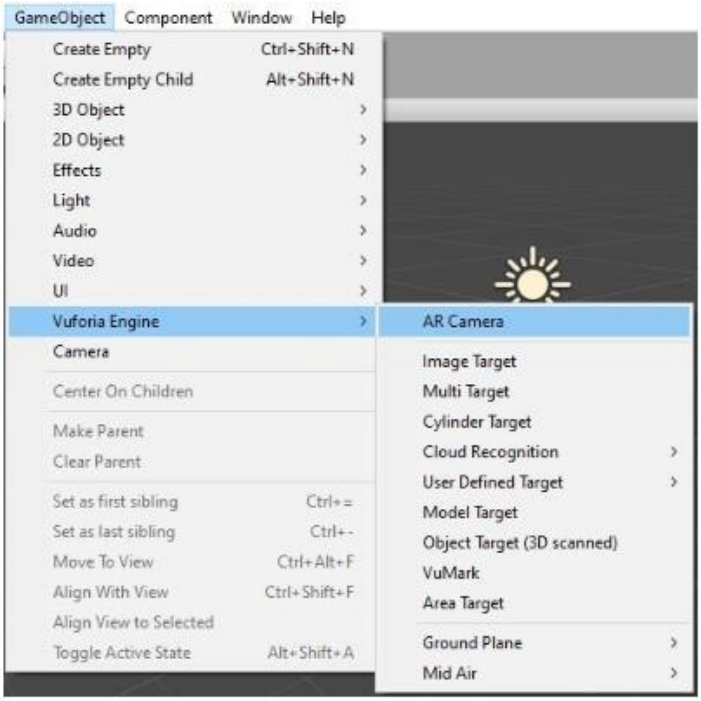
- After importing the SDK in Unity scene

1. Add an ARCamera

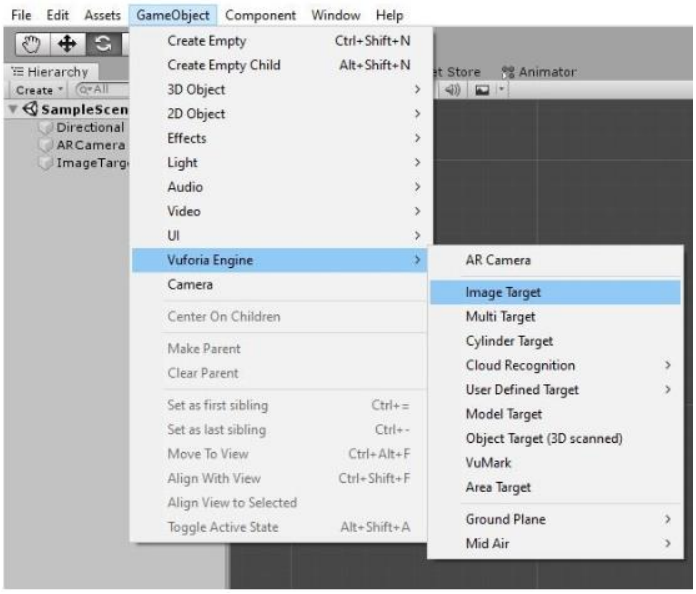
   Delete the default Main Camera after adding an ARCamera.

   The ARCamera contains its own scene Camera.

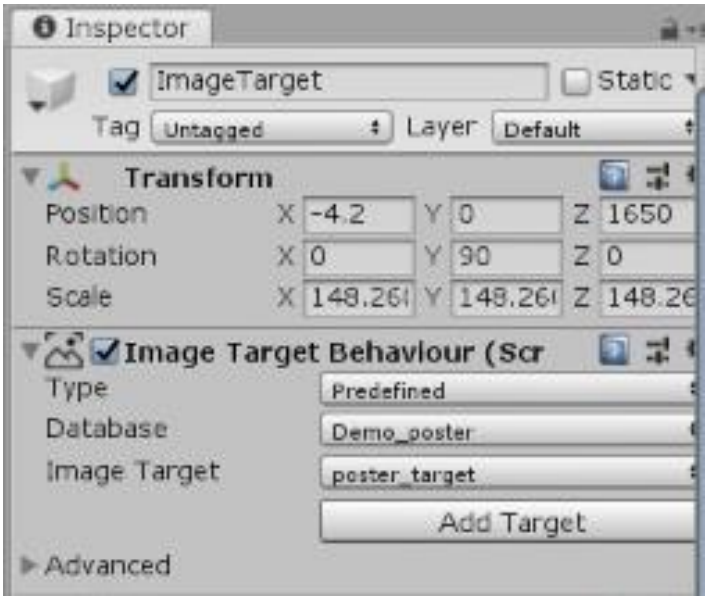
   There is no need for Main Camera unless we are using it to render a specific camera view.



2. Open the global Vuforia Configuration Inspector

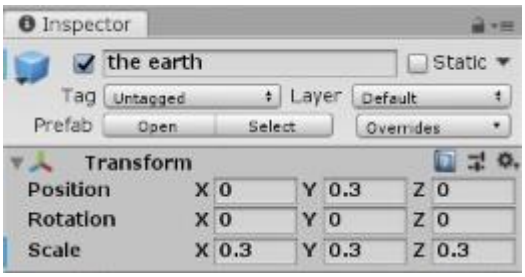


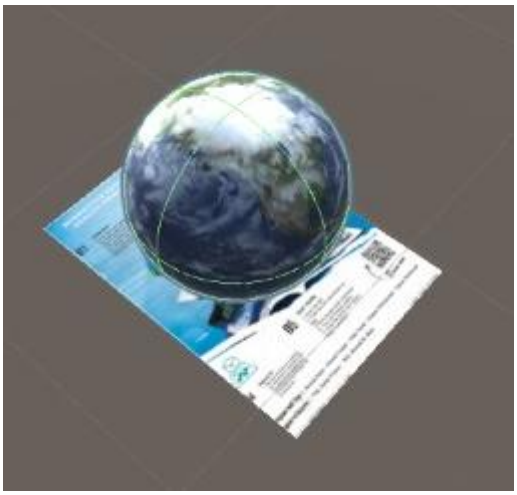
3. Activate the target database



4. Add target to the scene

5. Add digital content to augment the target by adding assets as children of the target in the scene hierarchy. Parenting content with a target object automatically sets up the necessary rendering and physics behaviors





Marker Less AR in our application (AR Foundation)

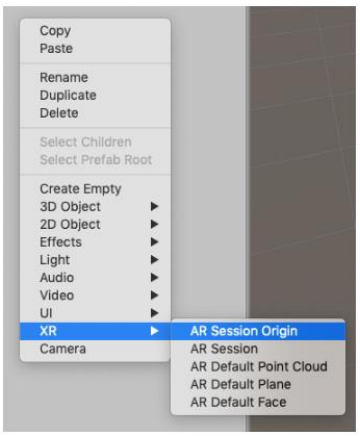
AR Foundation allows working with augmented reality platforms in a multi-platform way within Unity

AR Foundation is built on subsystems. A "subsystem" is a platform-agnostic interface for surfacing different types of information. The AR-related subsystems are defined in the AR Subsystems package and use the namespace “UnityEngine.XR.ARSubsystems”.

Each subsystem handles some specific functionality. For example, the plane detection interface is provided by the “XRPlaneSubsystem”.

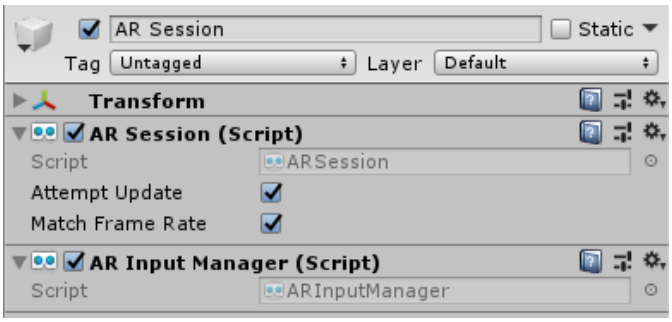
- After installing AR Subsystem and AR Foundation Packages, AR Core package is also imported, as the application will be launched to android

- To create AR: add AR Session and AR Session Origin



AR Session

The AR Session controls the lifecycle of an AR experience, enabling or disabling AR on the target platform. If the ARSession is disabled, the system no longer tracks features in its environment, but if it is enabled at a later time, the system will attempt to recover and maintain previously detected features

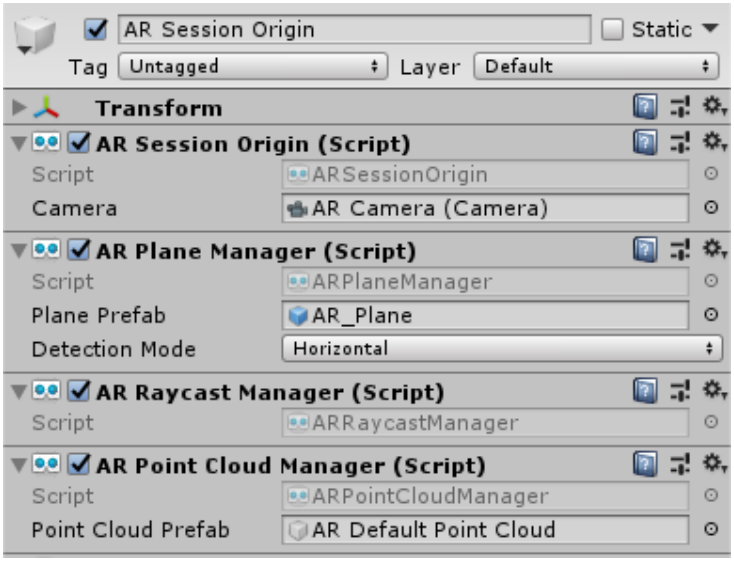


AR Input Manager

This component is necessary for enabling world tracking. Without it, the Tracked Pose Driver mentioned above will not be able to acquire a pose for the device.

AR Session Origin

The purpose of the ARSessionOrigin is to transform trackable features (such as planar surfaces and feature points) into their final position, orientation, and scale in the Unity scene. Because AR devices provide their data in "session space", an unscaled space relative to the beginning of the AR session, the ARSessionOrigin performs the appropriate transformation into Unity space.

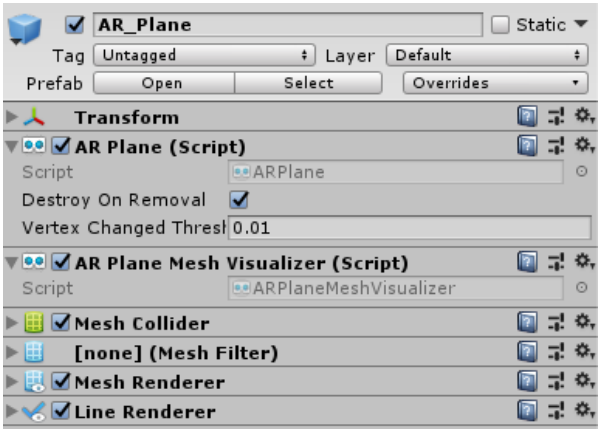


1. AR Plane Manger

The plane manager creates GameObjects for each detected plane in the environment. A plane is a flat surface represented by a pose, dimensions, and boundary points. The boundary

points are convex.

Examples of features in the environment that may be detected as planes are horizontal tables, floors, countertops, and vertical walls.



The AR Plane Manager Component allows user to specify the detection mode, which can be horizontal, vertical, or both. Some platforms require extra work to perform vertical plane detection, so if only horizontal planes are needed then disable vertical plane detection.

To visualize planes create a prefab or GameObject which includes a component that subscribes to ARPlane's boundaryChanged event.

ARFoundation provides an ARPlaneMeshVisualizer. This component will generate a Mesh from the boundary vertices and assign it to a MeshCollider, MeshFilter, and LineRenderer, if present.

2. AR Ray Cast Manger

Ray casting also known as hit testing, ray casting allows determination of where a ray (defined by an origin and direction) intersects with a trackable. The current ray cast interface only tests against planes and points in the point cloud. The ray casting interface is similar to the one in the Unity Physics module, but since AR trackables may not necessarily have a presence in the physics world, AR Foundation provides a separate interface.To perform a ray cast, add an AR Raycast Manager to the same GameObject as the AR Session Origin.

3. AR Point Cloud Manger

The point cloud manager creates point clouds, sets of feature points. A feature point is a specific point in the point cloud which the device uses to determine its location in the world. Feature points are typically notable features in the environment that the device can track between frames, such as a knot in a wooden table.

A point cloud is a set of feature points which can change from frame to frame. Some platforms only produce one point cloud, while others organize their feature points into different point clouds in different areas of space.

A point cloud is considered a trackable, while individual feature points are not. However, feature points can be uniquely identified between frames as they have unique identifiers.

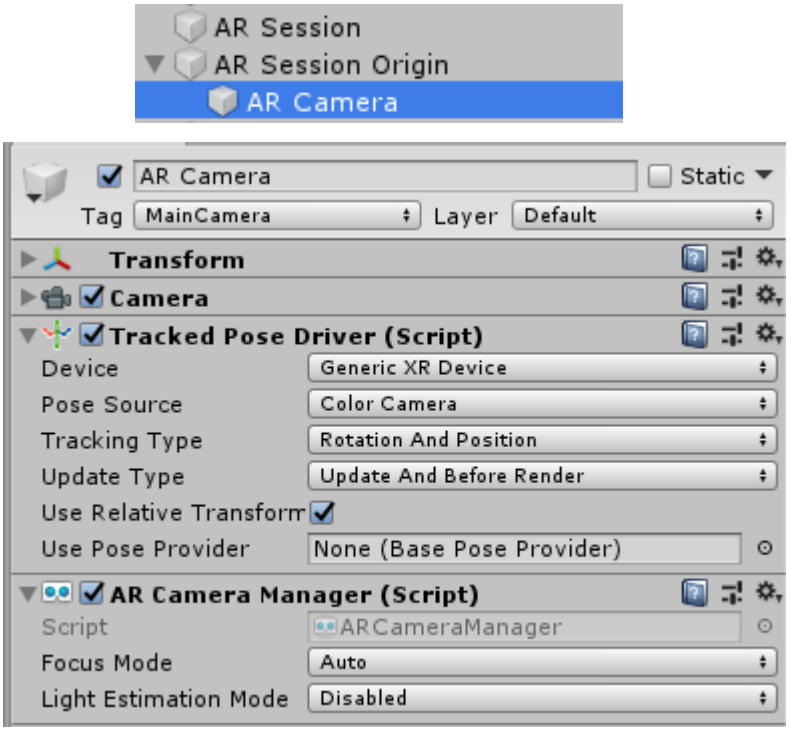
AR Camera

1. Tracked Pose Driver

Parented to the ARSessionOrigin's' GameObject should be (at least) one camera, which will be used to render any trackables to visualize. The camera should also have a TrackedPoseDriver component on it, which will drive the camera's local position and rotation according to the device's tracking information. This setup allows the camera's local space to match the AR "session space".

AR Camera Manager

The ARCameraManager enables camera features, such as textures representing the video feed and controls light estimation modes.



CHAPTER 3
Blender

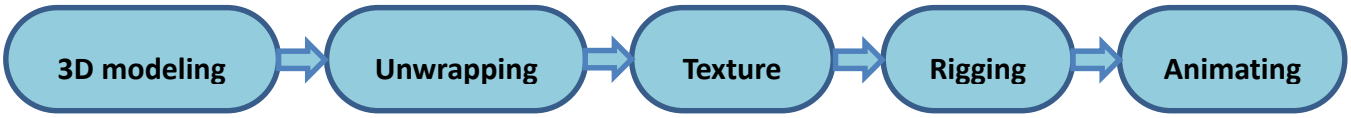
Introduction

Blender can be used to create 3D visualizations such as still images, 3D animations, VFX shots, and video editing.

Blender is well suited to individuals and small studios who benefit from its unified pipeline and responsive development process.

Blender is a cross-platform application, running on Linux, macOS, and Windows systems. Blender also has relatively small memory and drive requirements compared to other 3D creation suites. Its interface uses OpenGL to provide a consistent experience across all supported hardware and platforms.

Steps to Create 3D Model



3D modeling

3D modeling is a technique in computer graphics for producing a 3D digital representation of any object or surface.

These 3D objects can be generated automatically or created manually by deforming the mesh, or otherwise manipulating vertices.

The core of a model is the mesh which is best described as a collection of points in space.

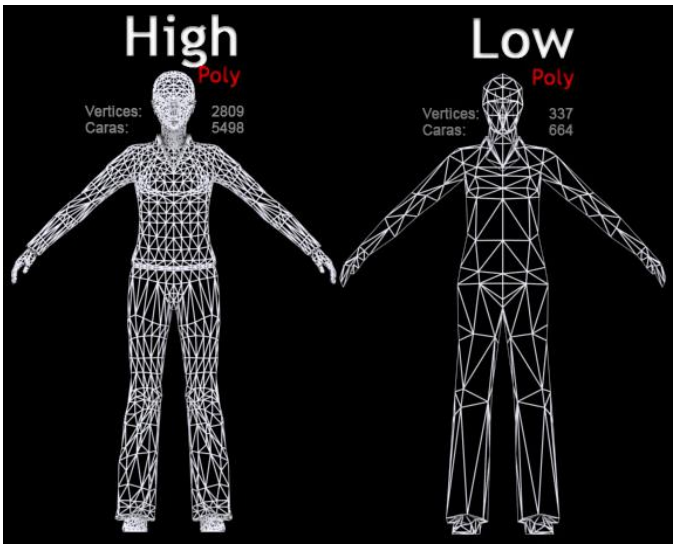
How Does 3D Modeling Work ?

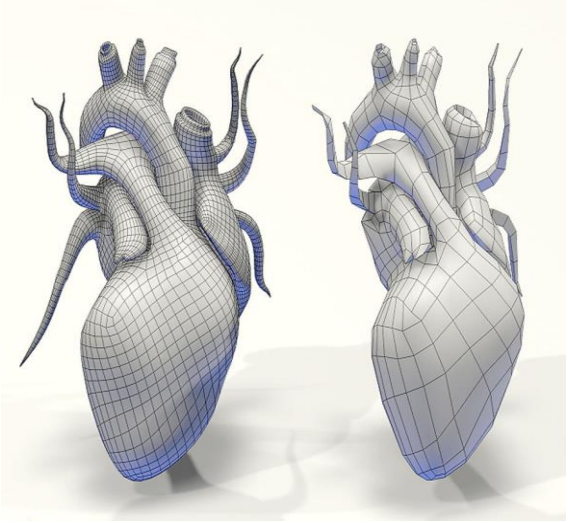
An artist usually begins by generating some type of primitive like a cube, sphere, or plane. The primitive is just a starting shape to begin modeling.

The exterior of the mesh is composed of polygons which can be subdivided into smaller shapes to create more detail. These subdivisions are especially necessary if the 3D model is to be animated.

There are two types of models:

- High-Polygons model
- Low-Polygons model





High-poly is important to achieve superb surface detail. The problem is how we control such enormous amount of vertices:

- with brushes (sculpting)
- through control low-poly cages (subdivision surfaces)
- through curve or nurbs surfaces
- generated by some simulation, 3D scanning, etc.

Low-poly is important for:

- game-engines
- subdivision modelling
- low-polygon proxy geometries of high-poly ones - for rigging and animating
- low-poly style and such

Problem with low-poly is topology, which has to follow some rules, so the shading looks right, the subdivision surfaces are right or the sculpts will be right. That's why we see so much about low-poly - with knowledge about low-poly topology the high-poly comes along.

- we convert low-poly to high-poly through sculpting, or catmull-clark subdivision
- we convert high-poly to low-poly through retopology, or decimation. The details from high-poly can be baked into texture maps .

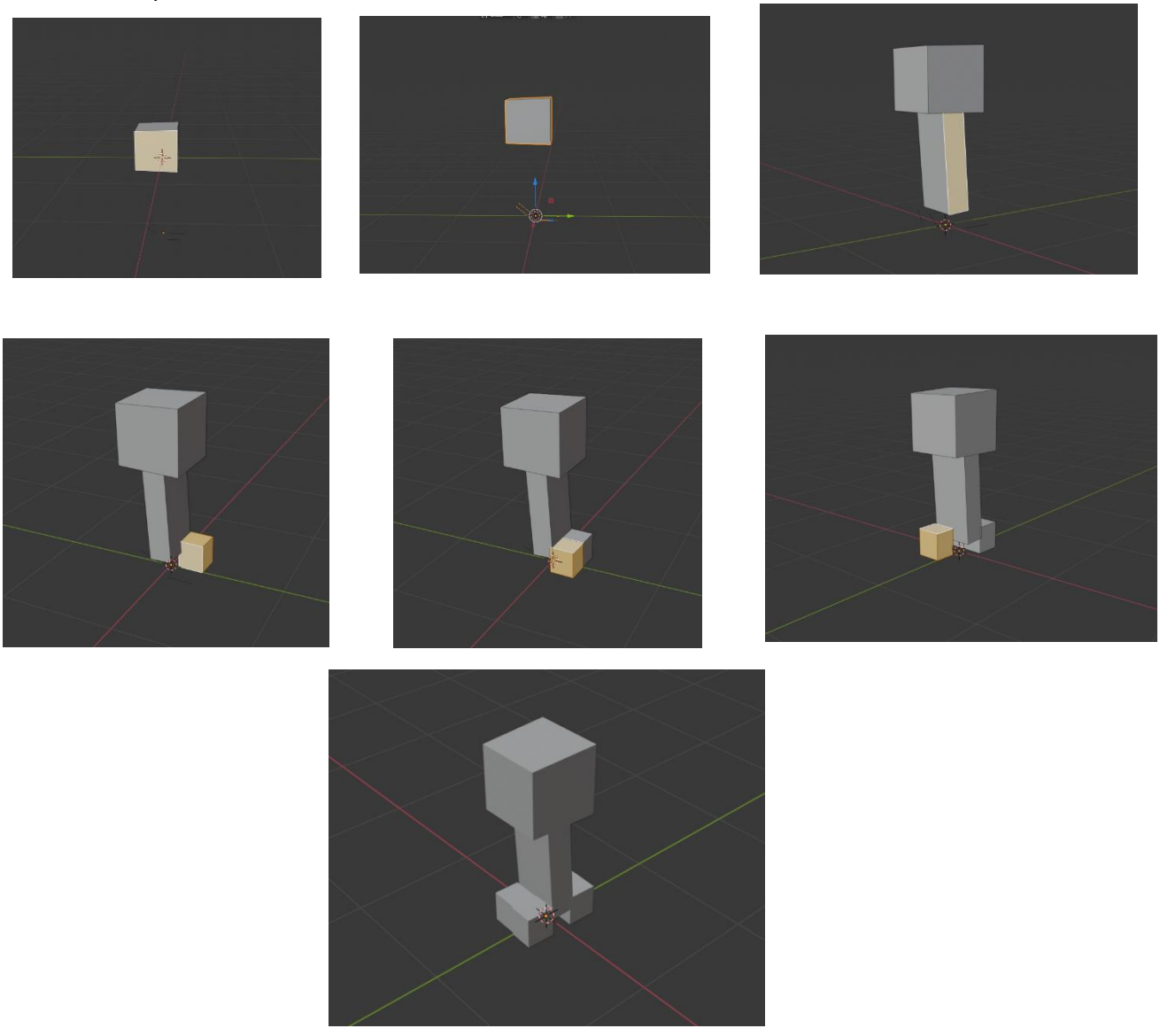
What is considered low-poly and high-poly changes with time and computer power, it's safe to say that low-poly is everything that can be displayed on target hardware in real-time. That means that for mobile phone this is different than for workstation.

Appropriate level of detail depends on how small the object will be on screen - basically how much pixels it will cover on screen:

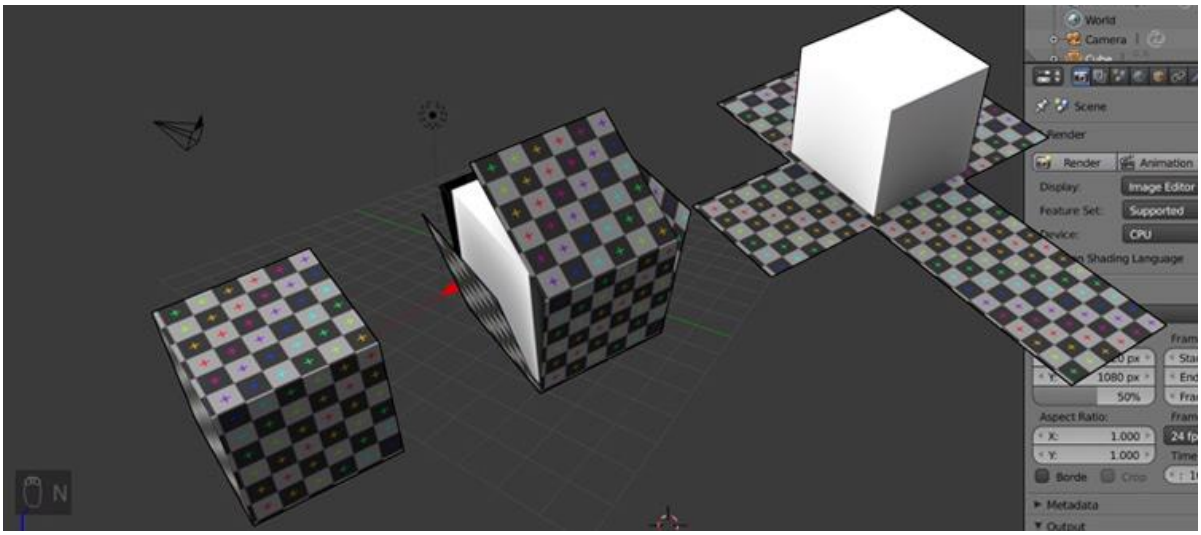
- for textures on lowpoly we want them 2x larger than a texture that has same size pixels on a model as a pixels on screen. This is because texture filtering. Or how the target hardware GPU memory that is we use.
- For game models this is usually from 1024x1024 to 4096x4096 or more for next-gen.
- For mobile-phones aroung 256x256 to 512x512.
- for low-poly polycounts:
    - around 10k for character or more for next-gen. Depends on the object size, hardware, if it is in background or not etc.
- for high-poly we want the polygons small enough that the surface detail will be well defined, this is usually smaller than screen pixel size or even smaller than that. Polycounts in millions.

Our 3D modelling (Creeper)

We made simple model from scratch



Unwrapping and Texture



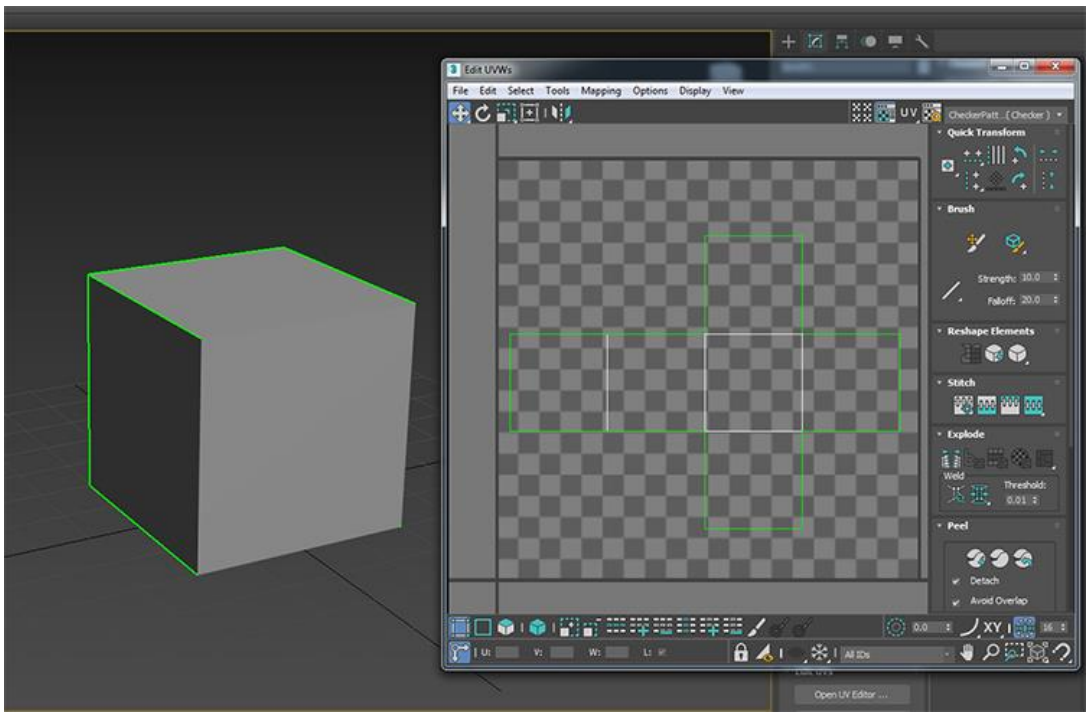
A UV map is the flat representation of the surface of a 3D model used to easily wrap textures. The process of creating a UV map is called UV unwrapping.

The U and V refer to the horizontal and vertical axes of the 2D space, as X, Y and Z are already being used in the 3D space.

Once the polygonal mesh has been created the next step is to “unwrap it” into a UV map. Now to give life to the mesh and make it look more realistic(or stylized) we want to add textures.

However there is no such thing as a 3D texture, as they’re always based on a 2D image.

This is where UV mapping comes in, as it is the process of translating our 3D mesh into 2D information so that a 2D texture can be wrapped around it.This may seem like a confusing idea at first, but it’s really very simple. If you’ve ever made a cube out of a cross of paper before, you have done the same process, only in reverse!



However, a cube is a basic example and as the meshes get more complex so do the UV maps. It can become quite a tedious process but it is essential to the 3D workflow.

Even if we don't intend to texture a model, many modern real time engines such as Unreal Engine 4 or Unity need our assets to be UV unwrapped to perform some of its light baking.

Exploring UV Unwrapping

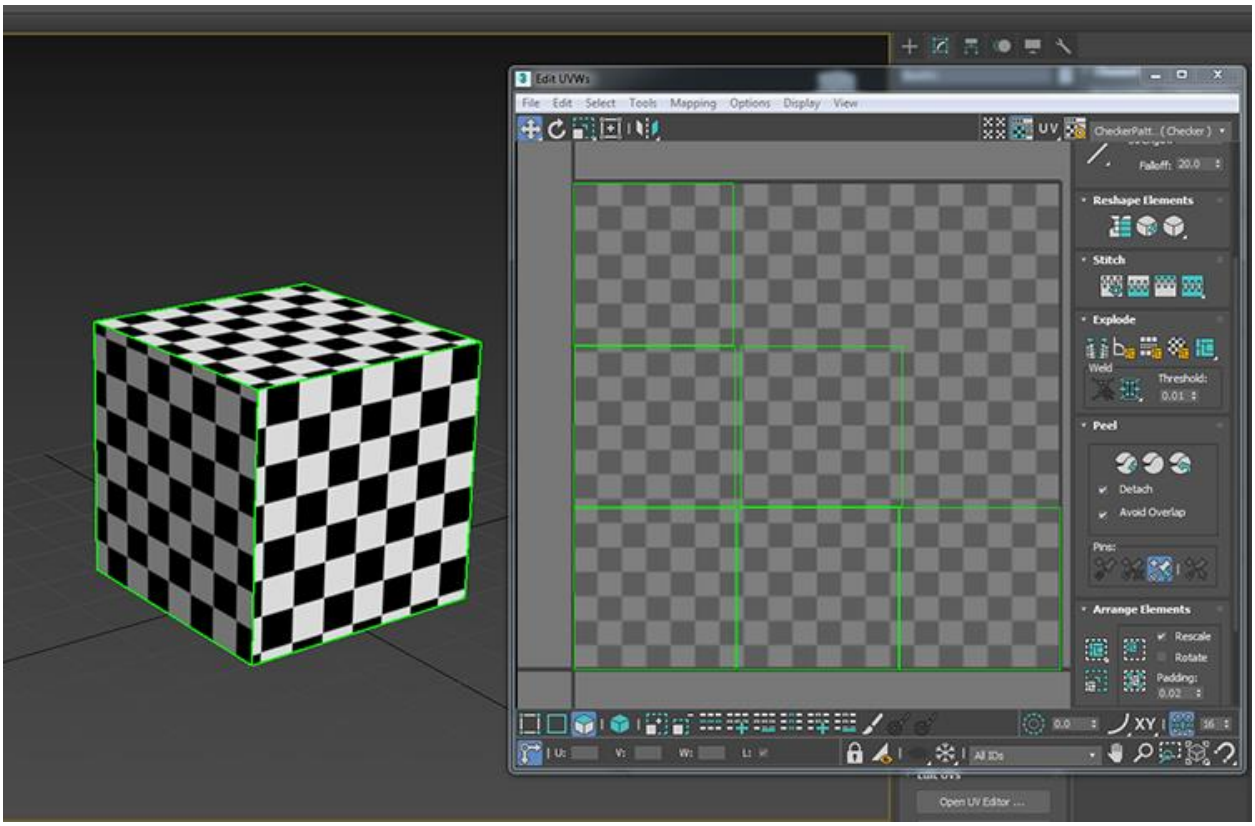
Now that the basic concept of UV maps has been outlined we can dive into the more intermediate parts to UV unwrapping, namely seams.

Seams are an unfortunate and unavoidable side effect of flattening any 3D geometry.

A seam is a part of the mesh that had to be split to be able to convert the 3D mesh into a 2D UV map.

UV unwrapping is always a compromise of causing as little distortion to the wireframe as possible, while also keeping seams to a minimum.

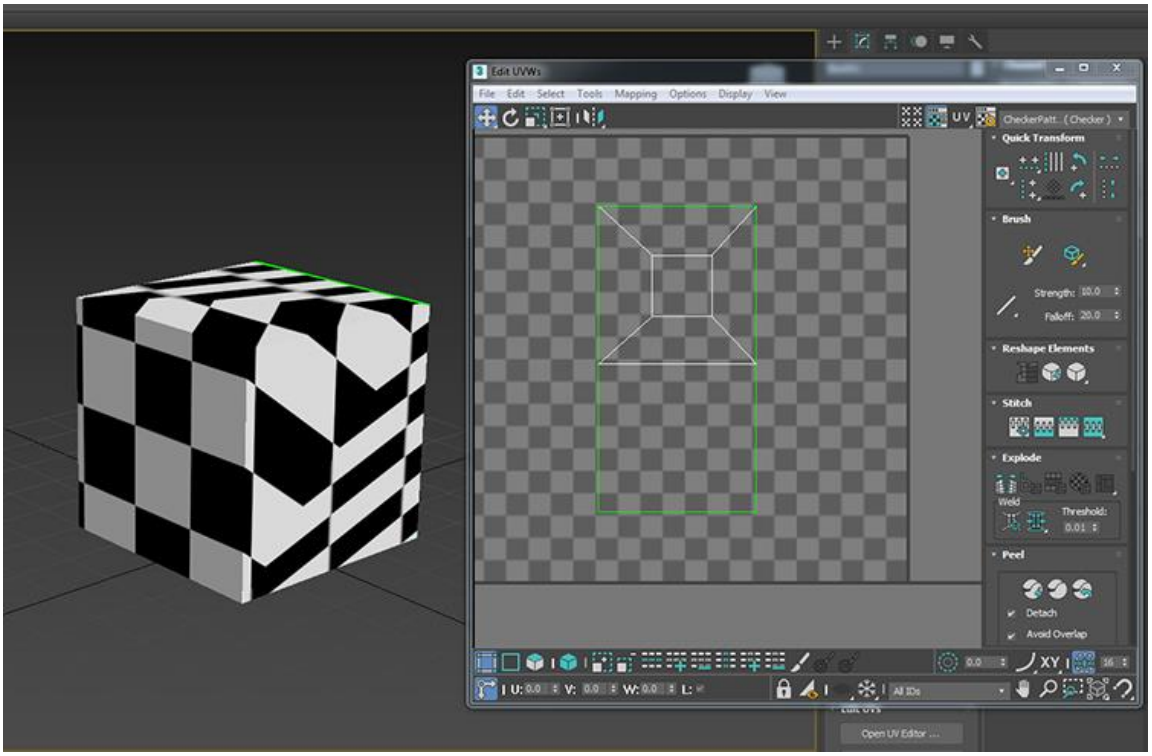
Distortion in terms of a UV map is how much the shape and size of the polygons have had to change to accommodate the flattening process. Too much distortion will affect the way details are displayed on the final model.



An unwrapped cube with a lot of seams

In this image the way the cube has been unwrapped has caused no distortion to the polygons.

This is easy to tell by applying a basic chequered texture. If the chequer pattern isn't stretched then we've avoided distortion in unwrapping.
However the downside to this method of just splitting all polygons apart is the number of seams it produces.
The cube on the left has the UV seams highlighted in green. we can see the pattern doesn't line up as it moves around the edges. This can become a problem in more complex meshes so you need to practice and get smart with seam placement.

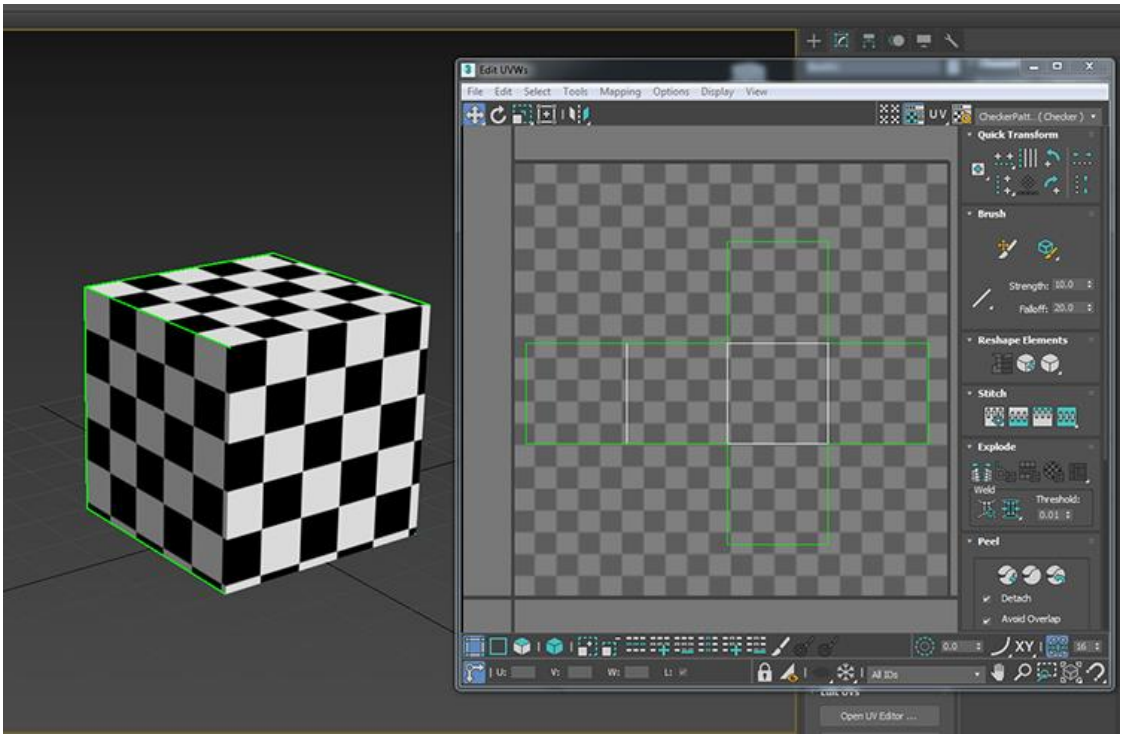


An unwrapped cube with almost no seams, but a lot of distortion

This is an example of what happens if there is heavily distort polygons in UV unwrap. The texture on this cube is no different than the previous example but as we can see it has been stretched and pulled out of shape.

However the lack of seams does mean that the pattern lines up and follows around the edges of the cube, but the payoff is not worth the sacrifice in this instance.

The obvious answer to this is to find a balance between the two.



A good example of how to unwrap a cube

In this image we can see that the chequer pattern is preserved and we have some nice continuous edges that flow around the front of the cube.

As we practice and do more unwrapping we learn where the best places to “hide” seams are to make them less noticea

Good rules for seams to make them less obvious are:

Make them follow hard edges where they are usually less noticeable

Hide them behind other parts of the model. For example if unwrapping a head place seams under where the hair will be.

Hide them underneath or behind the focal point of model, where people are less likely to see them.

Another thing to consider when unwrapping 3D mesh is overlapping UVs.

Overlapping UVs are when we have two or more of the polygons in UV map on top of one another. This means that these two parts of our model will display the same information of the texture as they both occupy the same UV space.

Overlapping UVs are usually something we want to avoid so we can keep our texture varied and not accidently end up with texture looking incorrect. That said there are times where we may intentionally use overlapping UVs. If a texture is quite basic then may have multiple parts of mesh over the same UV space to repeat the texture.

This technique can be very useful as it allows to keep texture sizes down which means if we are using a game engine it will generally run smoother.

This is even more important when developing for weaker machines like mobile phones.

One final thing to know about UV maps are UV channels.

UV channels allow the same object to have multiple UV maps. This is again very important for game engines.

As mentioned earlier in the article, game engines use UV maps to bake in lighting information.

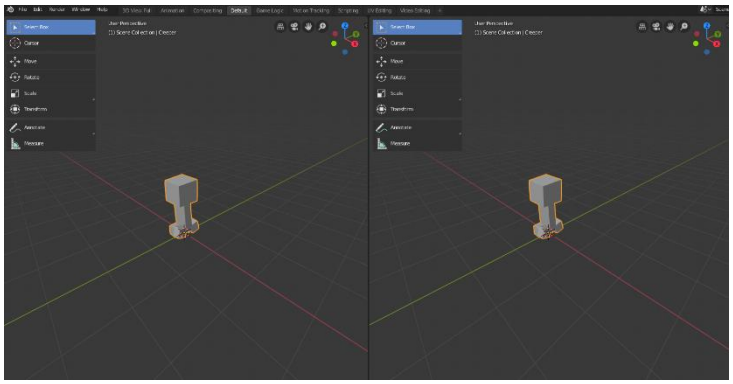
This means that there can be absolutely no overlapping UVs as shadow information will get put in wrong areas of the model and we will usually get some sort of error message.

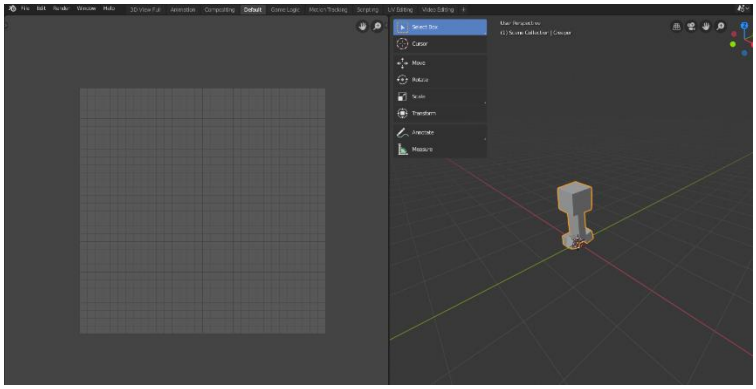
Now overlapping UVs are sometimes a good idea within game development as I mentioned.

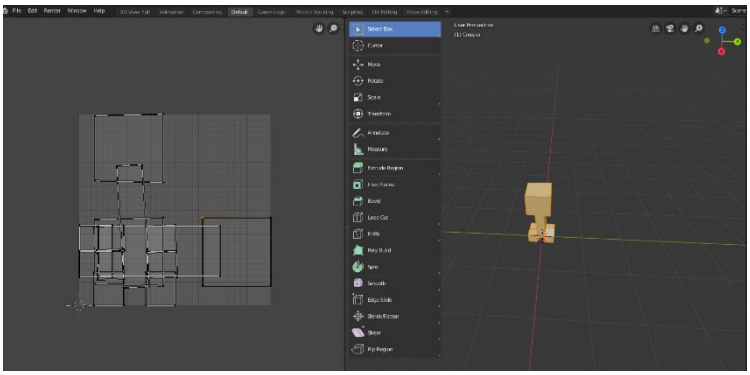
So the compromise is to have 2 UV channels. One with UV information for our textures and a second with UV information for lighting. It gets easier the more we do it but there is a lot of detail to the UV mapping process.

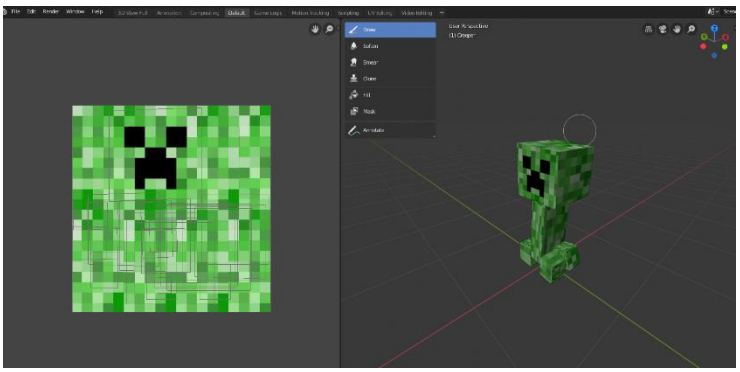
Hopefully we have a better understanding for UV mapping by now. It’s a simple enough process that can seem daunting at first, but quite easy to get the hang of with practice.

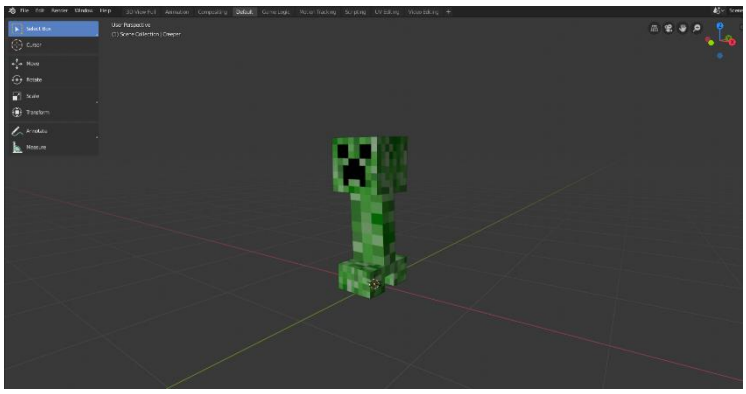
Unwrapping and Texture our Creeper

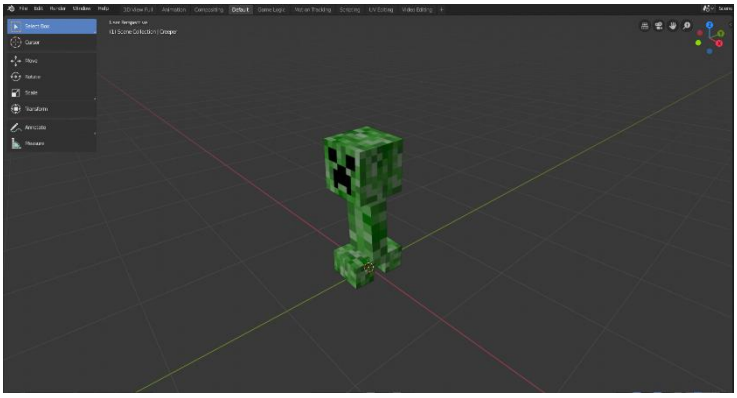












Rigging and Animating

Rigging is a technique used in skeletal animation for representing a 3D character model using a series of interconnected digital bones.

Specifically, rigging refers to the process of creating the bone structure of a 3D model. This bone structure is used to manipulate the 3D model like a puppet for animation.

Pretty much anything can be rigged. A space ship, a soldier, a galaxy, a door, it doesn't make any difference what the object is. Adding bones will allow any object to be animated freely.

Rigging is most common in animated characters for games and movies. This technique simplifies the animation process and improves production efficiency. Once rigged with skeletal bones, any 3D object can be controlled and distorted as needed.

In the entertainment industry rigging is a major step in the standard way of animating characters. Achieving smooth and complex animations is entirely dependent on the quality of the rigging phase in the animation pipeline.

How Does Rigging Work?

Rigging is one part of the larger animation process.

After a 3D model has been created, a series of bones is constructed representing the skeletal structure. For instance, in a character there may be a group of back bones, a spine, and head bones.

These bones can be transformed using digital animation software meaning their position, rotation, and scale can be changed.

By recording these aspects of the bones along a timeline (using a process called key framing) animations can be recorded.

A basic setup may take a few hours or less while a complex rig for a movie could take days.

The rigging process results in a hierarchal structure where each bone is in a parent/child relationship with the bones it connects to. This simplifies the animation process as a whole.

When an artist moves a shoulder bone, the forearm and hand bones will move too. The goal is to mimic real life as accurately as possible.

How the 3D model interacts with the bones is determined by a weight scale.

The weight controls how much influence a bone has over a section of the mesh. In this way, the sensitivity of the deformation can be fine-tuned for precise animation.

Weight painting is an integral part of the rigging process. The computer is often capable of automatically weight painting the model, but the result is sometimes lackluster.

To get great animations it's important to fine tune the weight of each bone.

Because some characters share the same skeletal structures, entire rigs can be copied and assigned to a new mesh. Animations can be copied in this way as well making it easy to produce models with similar designs.

Placing the bones is the easiest part of rigging a model. Once placed, many bones will need additional work to be animated properly.

Rigging a character usually requires the artist to add Inverse Kinematics to their bones. This will reverse the default Forward Kinematics properties of the bones.

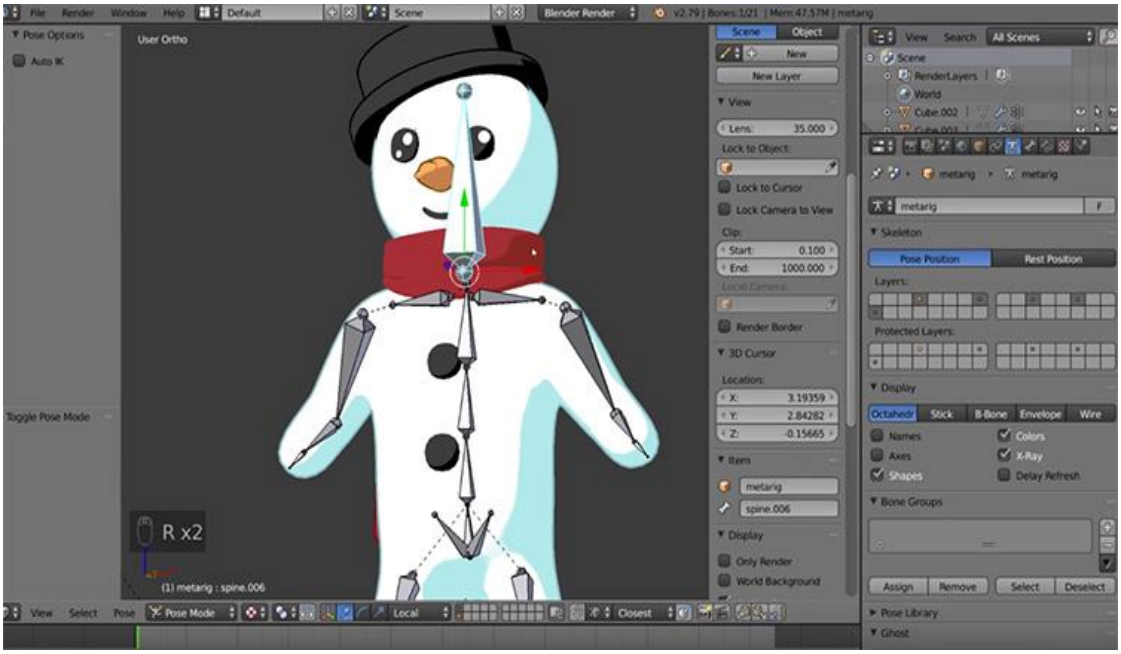
Inverse Kinematics is used mostly for arms and legs, or other extremities like a dragon's tail. A good IK setup will keep elbows and knees pointing in the right direction and allow an animator to achieve realistic motion more easily.

Constraints are another essential element to any decent rig. To create smooth animations, some bones need to have restrictions applied to their movement. For example a bone may be allowed to move in one direction only.

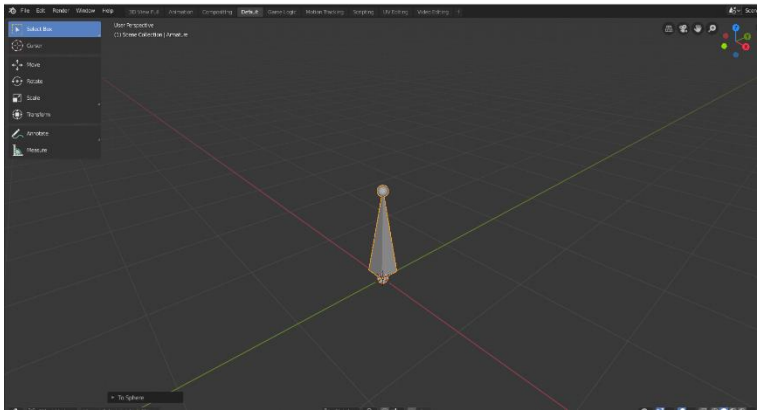
How is Rigging Used?

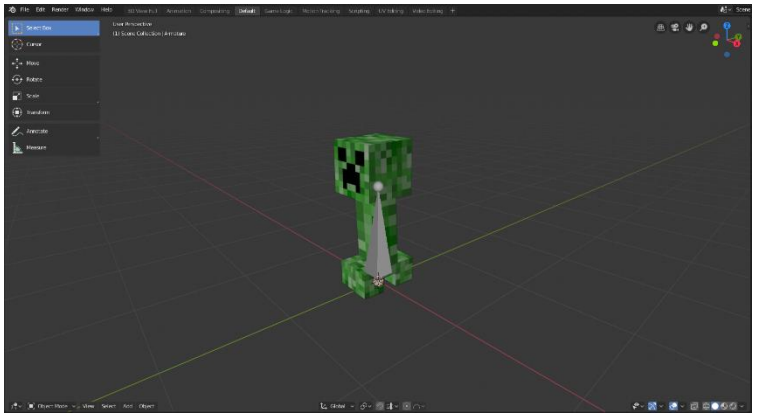
Rigging is a common technique for animating characters in video games, TV shows, and movies.

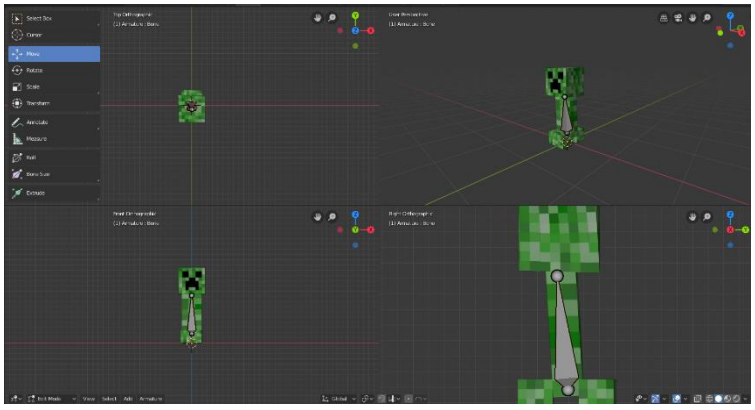
Mechanical objects such as springs and doors can also be animated using skeletal animation once a rig has been assigned to a 3D model.

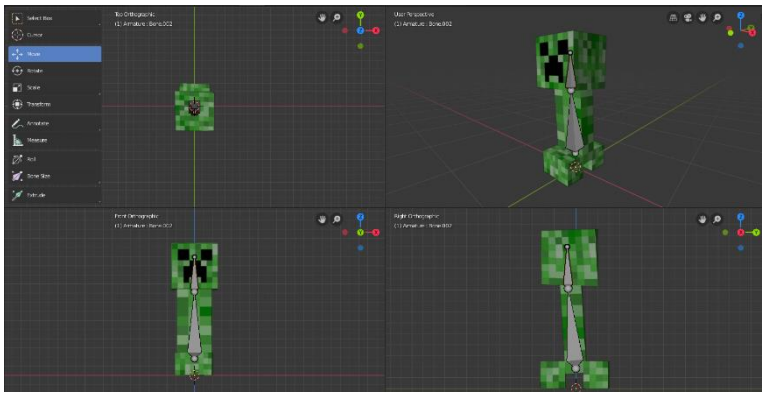


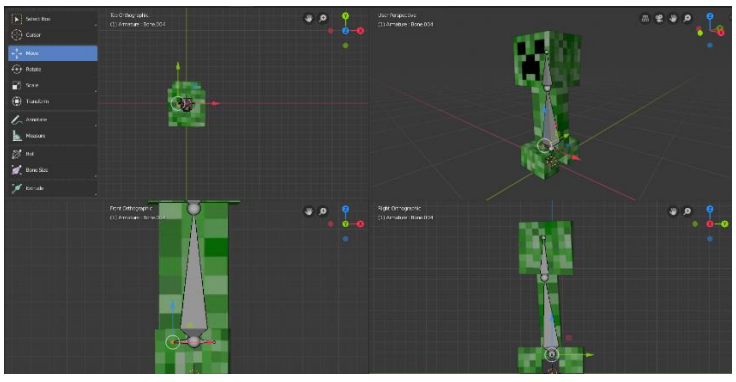
Rigging and Animating Creeper

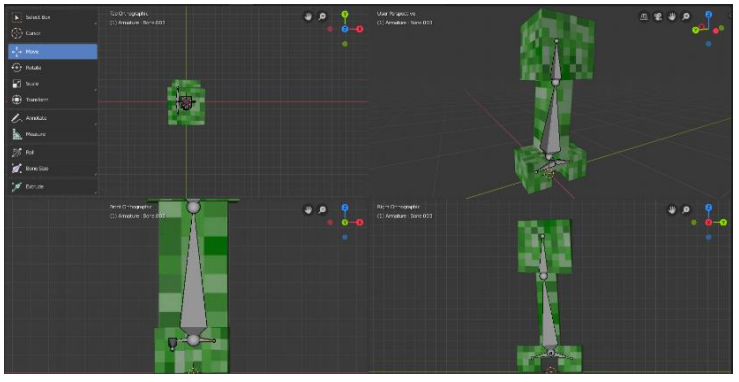


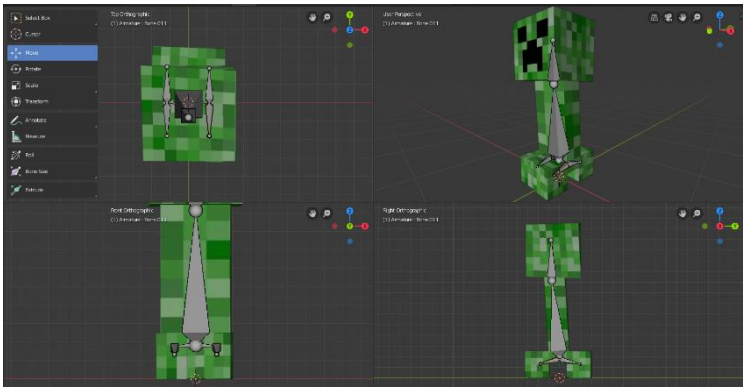












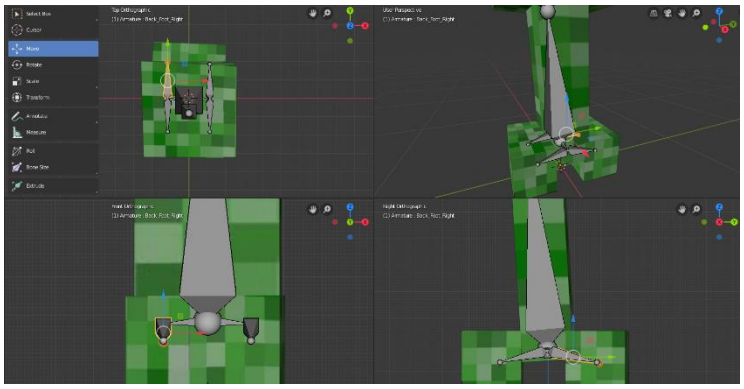


Image source

Simulations can be run on all skeletal structures too. The computer will apply physics to the bones in the rig and record the outcome over a series of frames. This is great for ragdoll physics and virtual stunts.

These virtual anatomy simulations are also useful outside of the arts and entertainment industry. For instance, the medical and education industries often need to create visuals for teaching or demonstrations.

Complex rigs are often necessary for quality facial animations too.

Creating a satisfactory facial rig requires a separate project from the rig of the character's body.

It's often best to use techniques outside of the traditional bone structure when rigging a face, such as morph targets or blend shapes.

The advantage of rigging is that it grants easy control over the deformation of the model, making it relatively simple to animate a character. The disadvantage of this technique is that it doesn't work well for animating surface details and it can take a very long time for complex projects.

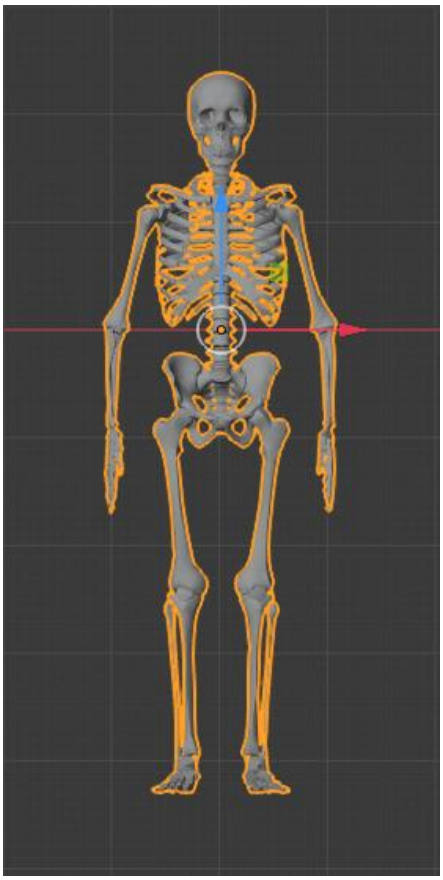
3D model modification

We need to use more advanced 3D models in our application and it's not easy to implement them, so we needed to get implemented models from stores but we faced some problems.

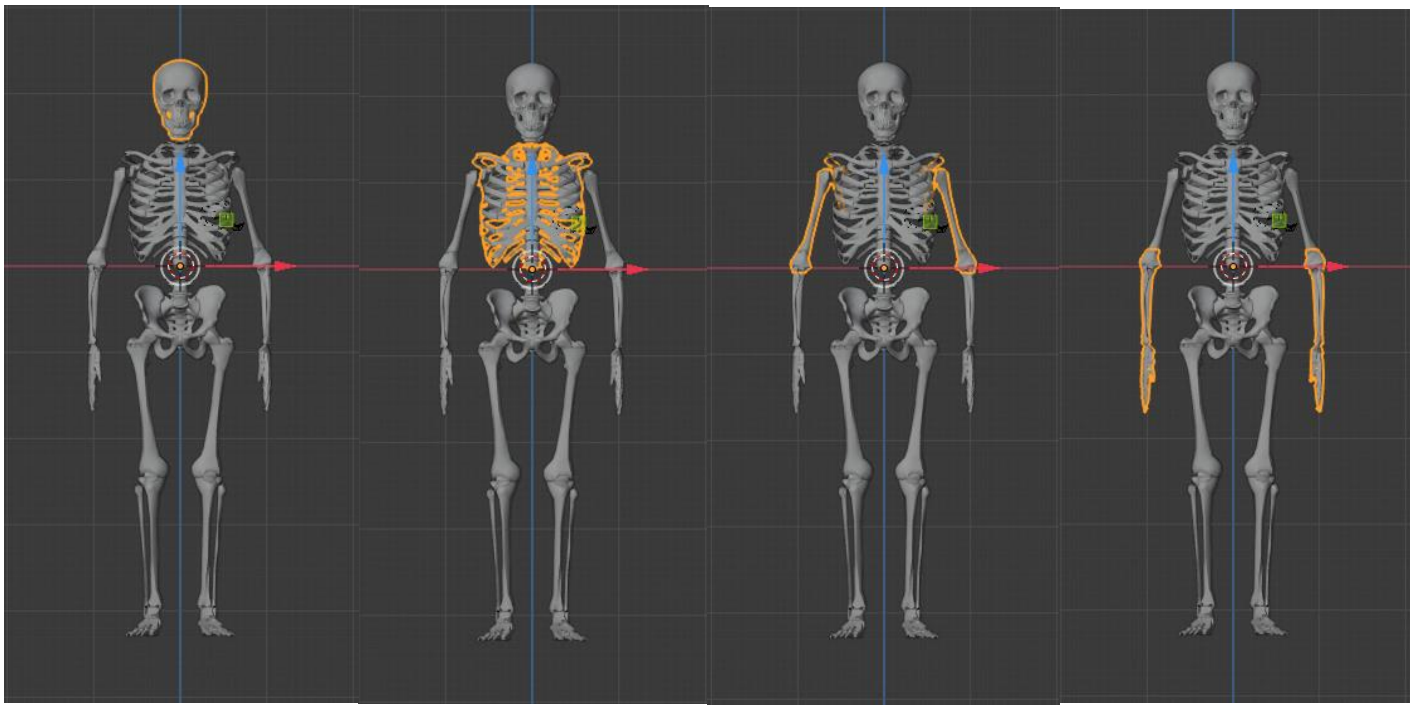
- We need separable 3D models and these models are so expensive, the price ranges (20$ - 100$) and free models usually not separable, so we decided to take these models and modify it using blender to be suitable with our needs and provides expensive cost.
- We also need animated 3D models which more expensive, we used free models and added animation using unity and blender.

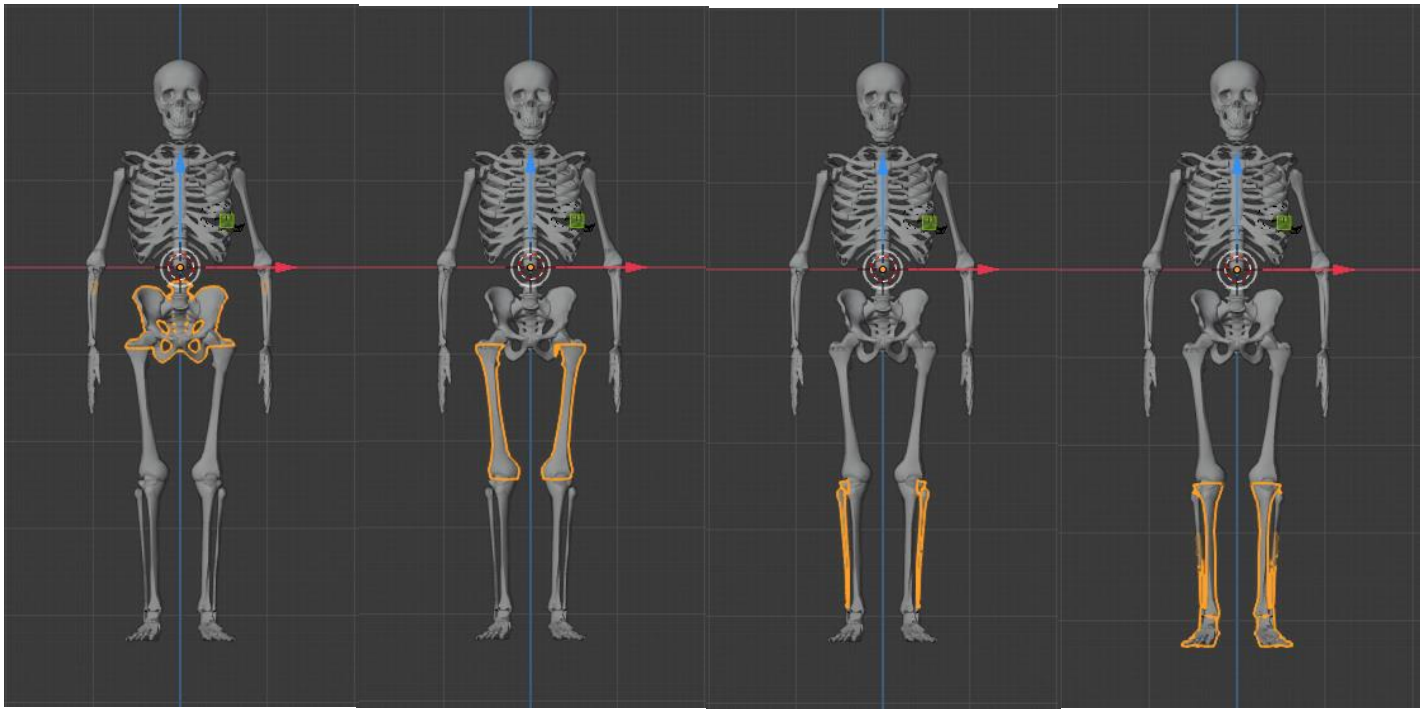
How did we modify models?

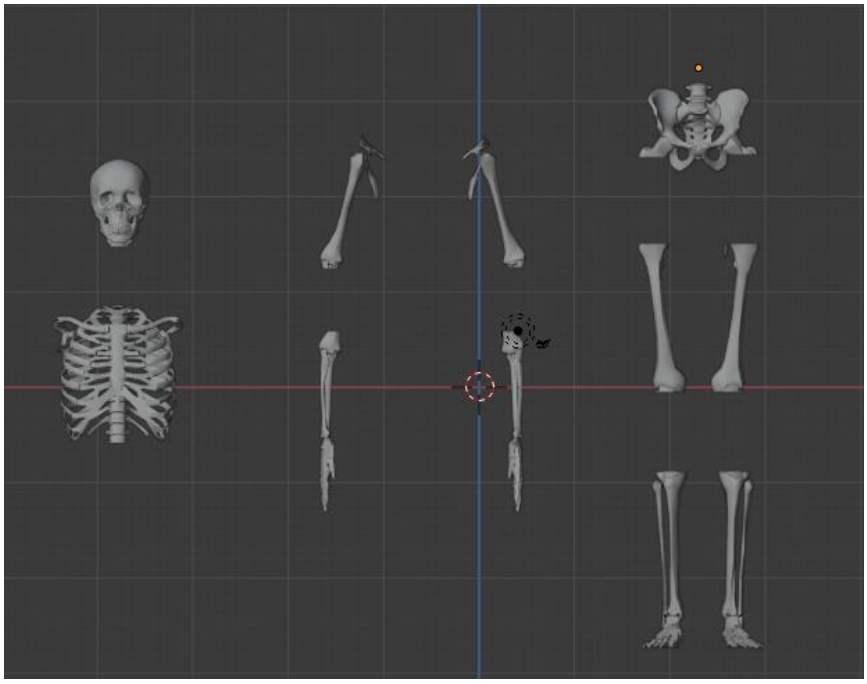
Skeleton free asset



Skeleton after modification







And we created animation for some models like heart and lungs using blender and imported it to unity.

CHAPTER 4
Unity Engine

INTRODUCTION

Unity is a cross-platform game engine developed by Unity Technologies, first announced and released in June 2005 at Apple Inc.'s Worldwide Developers Conference as a Mac OS X-exclusive game engine. As of 2018, the engine had been extended to support more than 25 platforms. The engine can be used to create three-dimensional, two-dimensional, virtual reality, and augmented reality games, as well as simulations and other experiences. The engine has been adopted by industries outside video gaming, such as film, automotive, architecture, engineering and construction.

We will talk about some tools that we used in Unity platform

User interfaces (UI)

Unity provides the following user interface (UI) toolkits for creating UI in either the Unity Editor or in an application:

- UIElements : User Interface Elements (UIElements)
- Unity UI (Package) : The Unity User Interface (Unity UI)
- IMGUI : Immediate Mode Graphical User Interface

But we use Unity UI in our application.

Unity UI (Package): Unity's in-game UI system used to create in-game user interfaces fast and intuitively. Using a couple included components such as a panel, and buttons so we can create a basic main menu for our application

Interaction Components in UI

This section covers components in the UI system that handles interaction, such as mouse or touch events and interaction using a keyboard or controller. The interaction components are not visible on their own, and must be combined with one or more visual components in order to work correctly.

Common Functionality Most of the interaction components have some things in common. They are selectable, which means they have shared built-in functionality for visualising transitions between states (normal, highlighted, pressed, disabled), and for navigation to other selectable using keyboard or controller. This shared functionality is described on the Selectable page.

The interaction components have at least one UnityEvent that is invoked when user interacts with the component in specific way. The UI system catches and logs any exceptions that propagate out of code attached to UnityEvent.

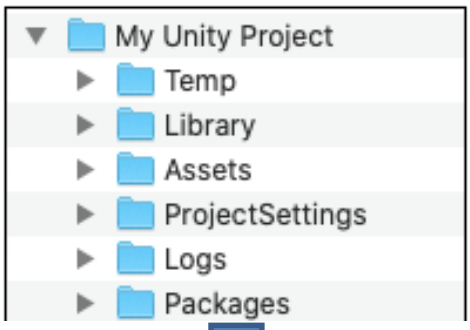
1- Button
2- Toggle
3- Toggle Group
4- Slider
5- Scrollbar
6- Dropdown
7- ScrollRect

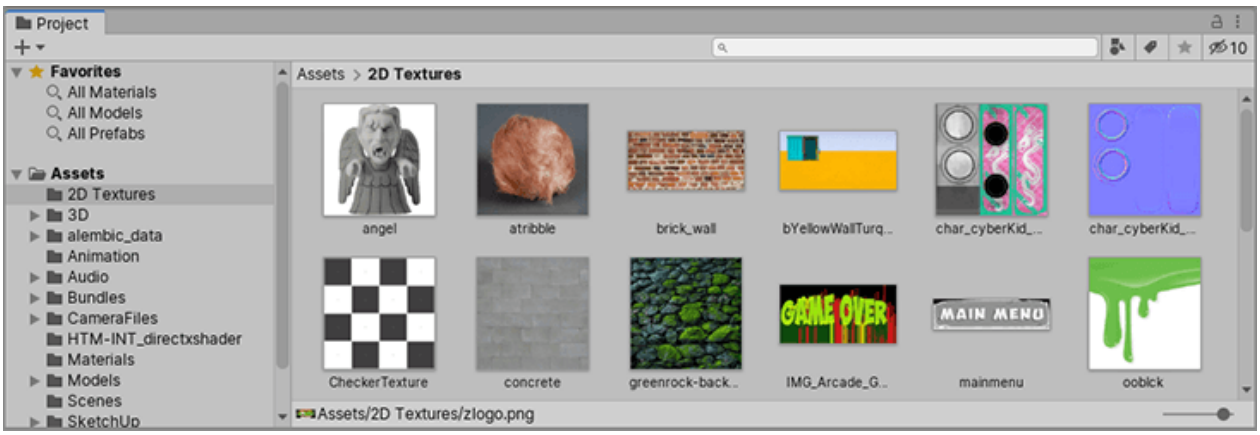
Importing

We can bring Assets created outside of Unity into our Unity Project by either exporting the file directly into the Assets folder under our Project, or copying it into that folder. For many common formats, we can save our source file directly into our Project's Assets folder and Unity can read it. Unity also detects when we save new changes to the file and re-imports files as necessary.

Common types of assets

1- Image files
2- FBX and Model files
3- Meshes and animations
4- Audio files





Creating Gameplay

Scenes

Scenes contain the environments and menus of our game. Think of each unique Scene file as a unique level. In each Scene, we place our environments, obstacles, and decorations, essentially designing and building our game in pieces

Game Objects

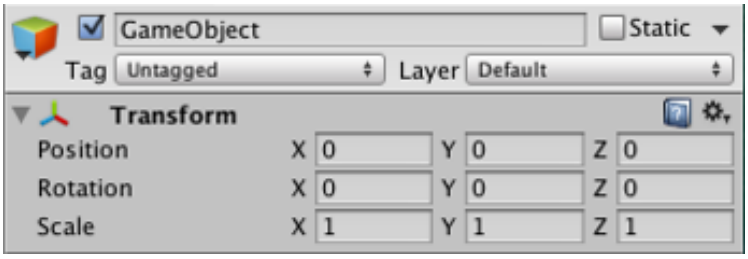
Is the most important concept in the Unity Editor.
Every object in our game is a GameObject, from characters and collectible items to lights, cameras component which creates an image of a particular viewpoint in our scene. The output is either drawn to the screen or captured as a texture, However, a GameObject can't do anything on its own; we need to give it properties before it can become a character, an environment, or a special effect.

A GameObject in the Unity Editor contains components. Components define the behaviour of that GameObject.

Common component:

Transform component: This component defines the GameObject's position, rotation, and scale in the game world and Scene view.

The Transform component also enables the concept of parenting, which allows us to make a GameObject a child of another GameObject and control its position via the parent's Transform component. This is a fundamental part of working with GameObjects in Unity



allows the user to control our application using a device, touch, or gestures.
We can program in-app elements, such as the graphic user interface (GUI) or a user avatar , to respond to user input in different ways.

Unity supports input from many types of input devices, including:

- Keyboards and mice
- Joysticks
- Controllers
- Touch screens
- Movement-sensing capabilities of mobile devices, such as accelerometers or gyroscopes

- VR and AR controllers

In our App we used Touch Screens and AR controllers

Platform development

Unity supports the following platforms:

1- Windows, macOS and Linux Standalone
2- tvOS
3- iOS
4- Lumin
5- Android
6- WebGL
7- PS4
8- Xbox One

We used Android platform for our App, because it is supported in students 'Tabs.

Physics

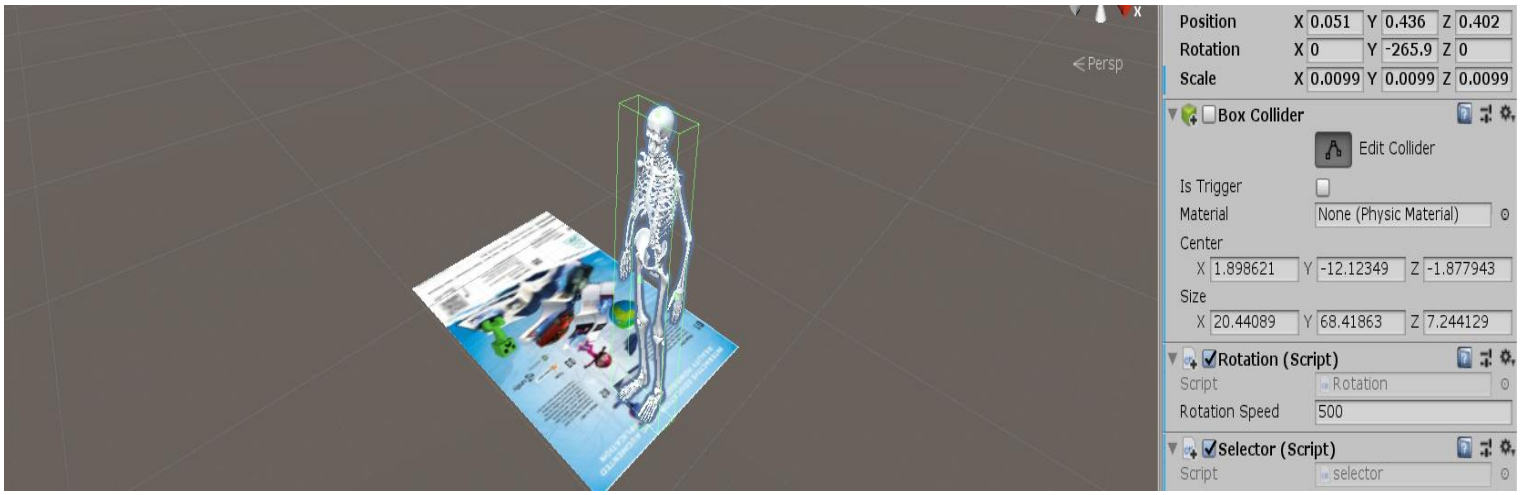
Unity helps you simulate physics in our Project to ensure that the objects correctly accelerate and respond to collisions, gravity, and various other forces.

Colliders

Collider components define the shape of a GameObject for the purposes of physical collisions. A collider, which is invisible, does not need to be the exact same shape as the GameObject's mesh. A rough approximation of the mesh is often more efficient and indistinguishable in gameplay.The simplest (and least processor-intensive) colliders are primitive collider types. In 3D, these are the Box Collider, Sphere Collider and Capsule Collider.

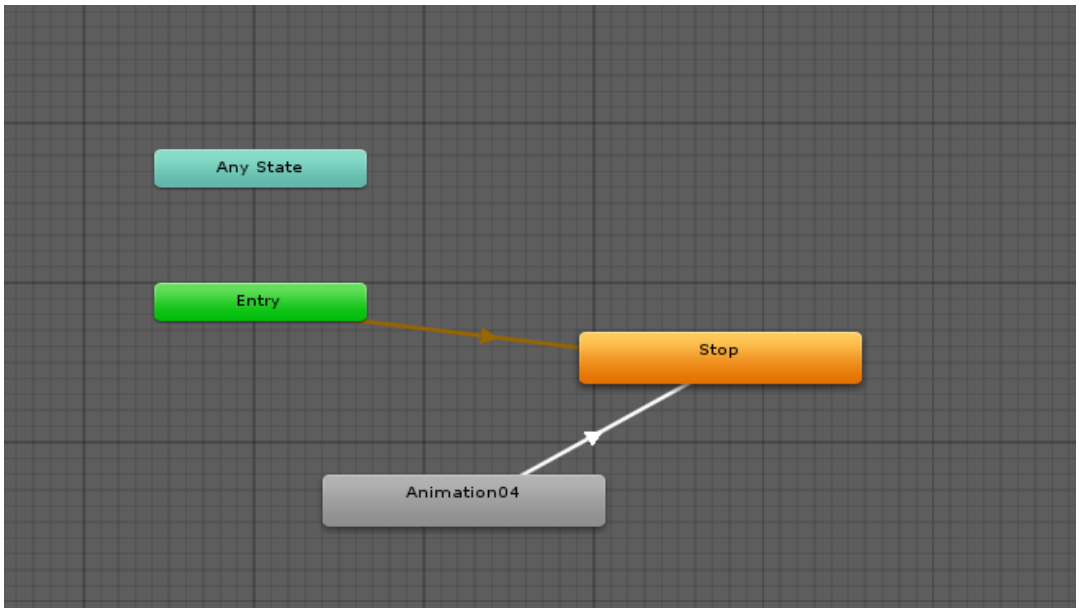
Collider interactions

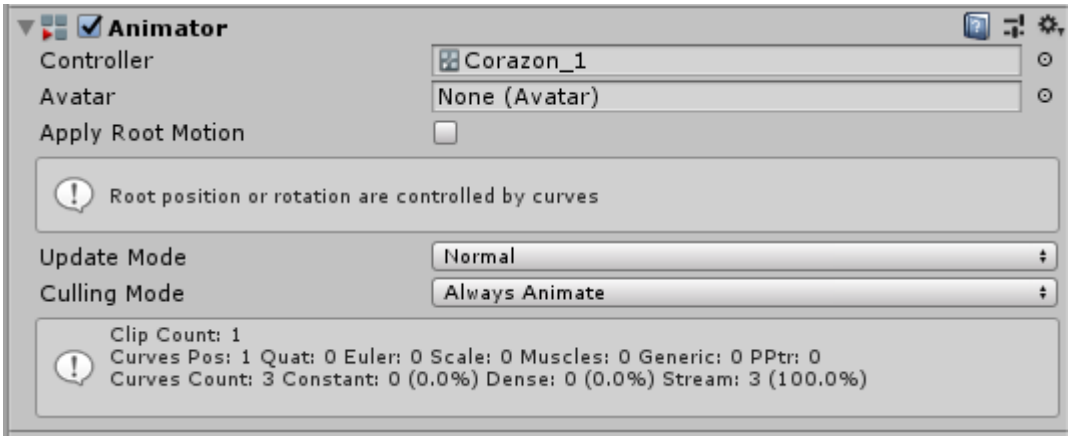
Colliders interact with each other differently depending on how their Rigidbody components are configured. The three important configurations are the Static Collider (ie, no Rigidbody is attached at all), the Rigidbody Collider and the Kinematic Rigidbody Collider.



Animator Controller

An Animator Controller allows you to arrange and maintain a set of Animation Clips and associated Animation Transitions for a character or object. In most cases it is normal to have multiple animations and switch between them when certain game conditions occur. For example, you could switch from a walk Animation Clip to a jump Animation Clip whenever the spacebar is pressed. However even if you only have a single Animation Clip you still need to place it into an Animator Controller to use it on a GameObject. The Animator Controller has references to the Animation clips used within it, and manages the various Animation Clips and the Transitions between them using a State Machine, which could be thought of as a flow-chart of Animation Clips and Transitions, or a simple program written in a visual programming language within Unity.





Unity automatically creates an Animator Controller when you begin animating a GameObject using the Animation Window, or when you attach an Animation Clip to a

GameObject. To manually create an Animator Controller, right click the Project window and click Create > Animator Controller.

Audio

Unity's Audio features include full 3D spatial sound, real-time mixing and mastering, hierarchies of mixers, snapshots, predefined effects and much more.

Audio Overview

A game would be incomplete without some kind of audio, be it background music or sound effects. Unity's audio system is flexible and powerful. It can import most standard audio file formats and has sophisticated features for playing sounds in 3D space, optionally with effects like echo and filtering applied. Unity can also record audio from any available microphone on a user's machine for use during gameplay or for storage and transmission.

Basic Theory

In real life, sounds are emitted by objects and heard by listeners. The way a sound is perceived depends on a number of factors. A listener can tell roughly which direction a sound is coming from and may also get some sense of its distance from its loudness and quality. A fast-moving sound source (like a falling bomb or a passing police car) will change in pitch as it moves as a result of the Doppler Effect. Also, the surroundings will affect the way sound is reflected, so a voice inside a cave will have an echo but the same voice in the open air will not.

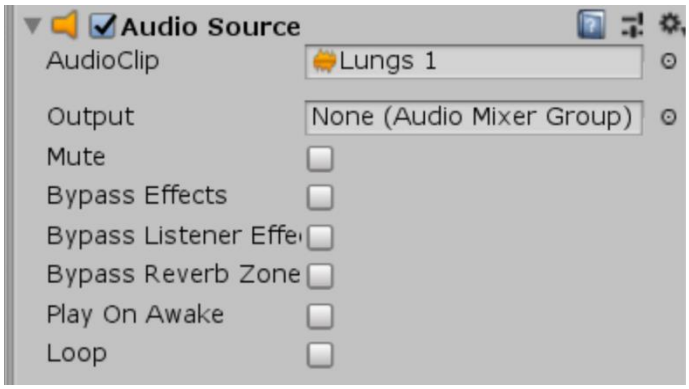
Working with Audio Assets

Unity can import audio files in AIFF, WAV, MP3 and Ogg formats in the same way as other assets, simply by dragging the files into the Project panel. Importing an audio file creates an Audio Clip which can then be dragged to an Audio Source or used from a script. The Audio Clip reference page has more details about the import options available for audio files. For music, Unity also supports tracker modules, which use short audio samples as "instruments" that are then arranged to play tunes. Tracker modules can be imported from .xm, .mod, .it, and .s3m files but are otherwise used in much the same way as ordinary audio clips.

Audio Source

The Audio Source plays back an Audio Clip in the scene.The clip can be played to an audio listener or through an audio mixer. The audio source can play any type of Audio Clip and can be configured to play these as 2D, 3D, or as a mixture (SpatialBlend). The audio can be spread out between speakers (stereo to 7.1) (Spread) and morphed between 3D and 2D (SpatialBlend). This can be controlled over distance with falloff curves. Also, if the listener is within one or multiple Reverb Zones, reverberation is applied to the source. Individual filters

can be applied to each audio source for an even richer audio experience. See Audio Effects for more details.



Scripting

Scripting is an essential ingredient in our applications we make in Unity. Most applications need scripts to respond to input from the user and to arrange for events in the gameplay to happen when they should.

Beyond that, scripts can be used to create graphical effects, control the physical behaviour of objects or even implement a custom AI system for characters in the game

Scripting Tools

This section covers the tools within the Unity Editor, and tools supplied with Unity that assist us in developing our scripts:

1- Console
2- Log Files
3- Unity Test Framework (formerly the Unity Test Runner)
4- C# compiler
5- IL2CPP
6- ScriptingToolsIDEs
7- Managed Code Debugging

We make C# script to control everything:

- Movement scripts: used to resize, drag, rotation objects

```
public class Rotation : MonoBehaviour
{
    public float RotationSpeed ;
    private void OnMouseDrag() {
        float x = Input.GetAxis("Mouse X") * RotationSpeed * Mathf.Deg2Rad;
        this.transform.Rotate(Vector3.up,-x);


        float y = Input.GetAxis("Mouse Y") * RotationSpeed * Mathf.Deg2Rad;
        this.transform.Rotate(new Vector3(0,0,1),-y);
    }
}
```

```
public class Drag : MonoBehaviour {

    Vector3 dist;
    float posX;
    float posY;

    void OnMouseDown(){
        dist = Camera.main.WorldToScreenPoint(transform.position);
        posX = Input.mousePosition.x - dist.x;
        posY = Input.mousePosition.y - dist.y;

    }

    void OnMouseDrag(){
        Vector3 curPos =
            new Vector3(Input.mousePosition.x - posX,
                Input.mousePosition.y - posY, dist.z);

        Vector3 worldPos = Camera.main.ScreenToWorldPoint(curPos);
        transform.position = worldPos;
    }
}
```

```
public class PinchtoResize : MonoBehaviour
{
    public float initialFingersDistance;
    public Vector3 initialScale;
    public static Transform ScaleTransform;


    void  Update (){
        int fingersOnScreen = 0;

        foreach(Touch touch in Input.touches) {
            fingersOnScreen++; //Count fingers (or rather touches) on screen as you iterate through all screen touches.

            //You need two fingers on screen to pinch.
            if(fingersOnScreen == 2){

                //First set the initial distance between fingers so you can compare.
                if(touch.phase == TouchPhase.Began){
                    initialFingersDistance = Vector2.Distance(Input.touches[0].position, Input.touches[1].position);
                    initialScale = ScaleTransform.localScale;
                }
                else{
                    float currentFingersDistance = Vector2.Distance(Input.touches[0].position, Input.touches[1].position);

                    float scaleFactor = currentFingersDistance / initialFingersDistance;

                    //transform.localScale = initialScale * scaleFactor;
                    ScaleTransform.localScale = initialScale * scaleFactor;
                }
            }
        }
    }
}
```

```
public class Resize : MonoBehaviour
{
    private void OnMouseDown() {
        PinchtoResize.ScaleTransform = this.transform;
    }
}
```

- To change scene

```
public class GameManger : MonoBehaviour
{
    public void ChangeScene(int sceneNumber)
    {
        SceneManager.LoadScene(sceneNumber);
    }

}
```

- To select which object we want to appear

```
public class selectorparts : MonoBehaviour
{
    public GameObject[] Objectstoshoworhide;
    private void OnMouseDown()
    {
        this.gameObject.SetActive(true);

        foreach (GameObject item in Objectstoshoworhide)
        {
            item.gameObject.SetActive(!item.activeSelf);
        }

    }
}
```

- Control the target that the camera should look at.

```
public class LookAtTarget : MonoBehaviour {

    static public GameObject target; // the target that the camera should look at

    void Start () {
        if (target == null)
        {
            target = this.gameObject;
            Debug.Log ("LookAtTarget target not specified. Defaulting to parent GameObject");
        }
    }

    // Update is called once per frame
    void Update () {
        if (target)
            transform.LookAt(target.transform);
    }
}
```

- Change the field of view on the perspective camera based on the distance from center of world, clamp it to a reasonable field of view.

```
public class ChangeLookAtTarget : MonoBehaviour {

    public GameObject target; // the target that the camera should look at

    void Start() {
        if (target == null)
        {
            target = this.gameObject;
            Debug.Log ("ChangeLookAtTarget target not specified. Defaulting to parent GameObject");
        }
    }

    // Called when MouseDown on this gameObject
    void OnMouseDown () {
        // change the target of the LookAtTarget script to be this gameobject.
        LookAtTarget.target = target;
        // change the field of view on the perspective camera based on the distance from center of world, clamp it to a reasonable field of view
        Camera.main.fieldOfView = Mathf.Clamp(60 * target.transform.localScale.x, 1, 100);
    }
}
```

- Dropdown allows selecting a preferred explanation language.

```
public class Dropdown : MonoBehaviour
{
    public List<GameObject> objectswithsources = new List<GameObject>();
    private List<AudioSource> _sources = new List<AudioSource>();
        void Start()
    {
        foreach(GameObject item in objectswithsources)
        {
            _sources.Add(item.gameObject.GetComponent<AudioSource>());
        }
    }
    public void HandleInputData (int val)
    {

        if (val == 0)
        {
            _sources[1].Stop();

            _sources[0].Play();
        }
        if (val == 1)
        {
            _sources[0].Stop();
            _sources[1].Play();
        }
    }

}
```

- Script to Spawn Object on the detected AR Plane in Marker-less AR

AR Ray Cast Manger use Ray casting to allow determination of where a ray (defined by an origin and direction) or a point (2D vector) intersects with a trackable(Plane is an example of a trckable)

So if touch input from a point on the screen is used as a ray start point

If the ray intersects with detected planes in the physical space the AR Plane Manger returns a list of hits.

The touch input from the screen can be passed to the Ray cast manger then getting the hit pose represents the location in the physical world corresponding to the touch position.

This can be used to spawn an object in the physical world based on user input touch on the screen.

```
public class SpawnObjectOnPlane : MonoBehaviour
{
    private ARRaycastManager _raycastManager;

    private Touch touch_position ;

    private GameObject spawnedObject;

    [SerializeField] private GameObject placableprefab;

    static List<ARRaycastHit> hits = new List<ARRaycastHit>();

    private void Awake()
    {
        _raycastManager = GetComponent<ARRaycastManager>();
    }

    void Update()
    {
        // Check User Touch

        if (Input.touchCount > 0 && Input.GetTouch(0).phase == TouchPhase.Began  )
        {
            touch_position = Input.GetTouch(0).position;
        }

        if (_raycastManager.Raycast(touch_position,hits,TrackableType.PlaneWithinPolygon))
        {
            var hit_pose = hits[0].pose;
            spawnedObject = Instantiate(placableprefab, hit_pose.position, hit_pose.rotation);
        }
    }
}
```

CHAPTER 5
Application

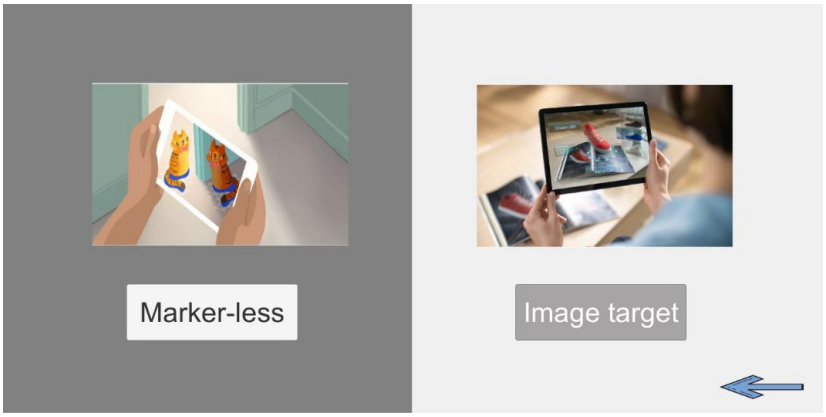
INTRODUCTION

After we showed our tools and explained the functions of them, now it's the time to show our application and its components.

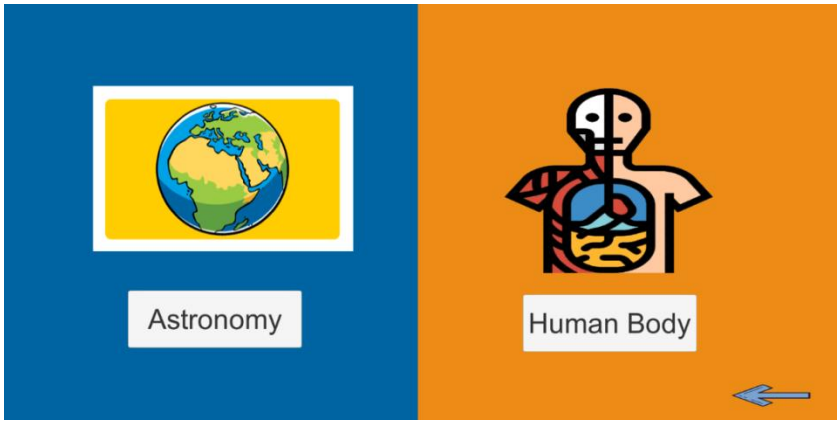
The application includes both of Marker Based AR and Marker-less AR

- In the first scene, the application asks you to start.

- Now we can choose which type of AR we want to interact with, let's begin with "Image target".

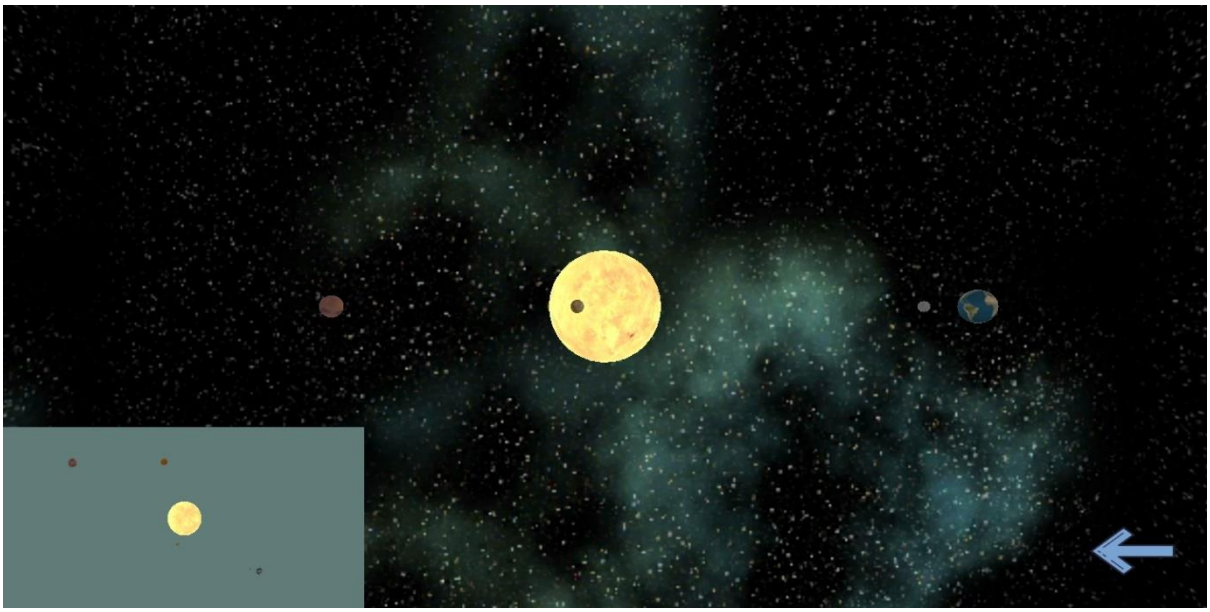


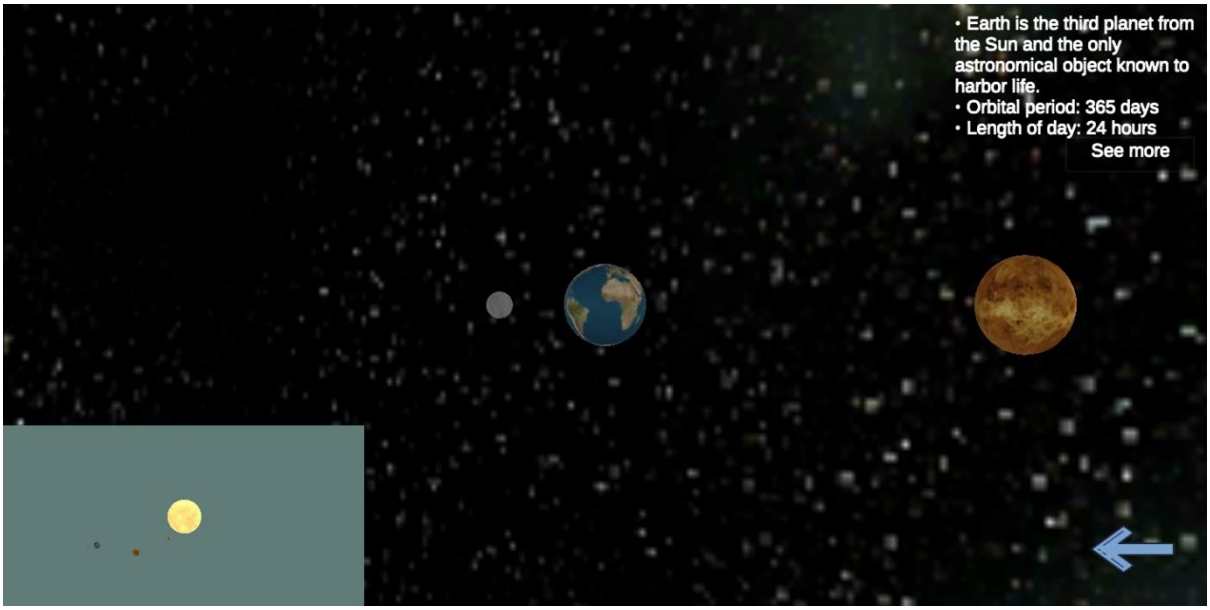
- Now we choose which subject to learn about.



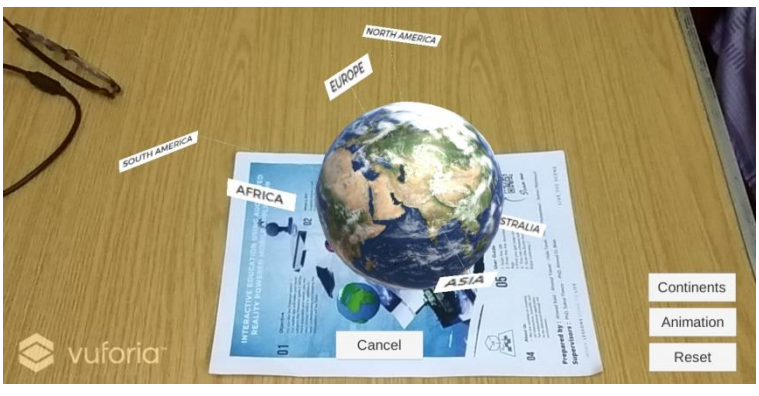
Marker-based AR with Image Target

- If we choose “Astronomy”, we can see the movement of the solar system, information of continents and we can see more about the earth.

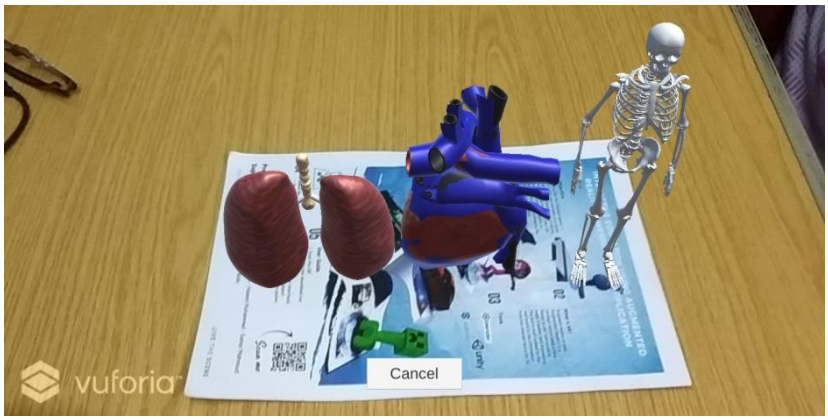




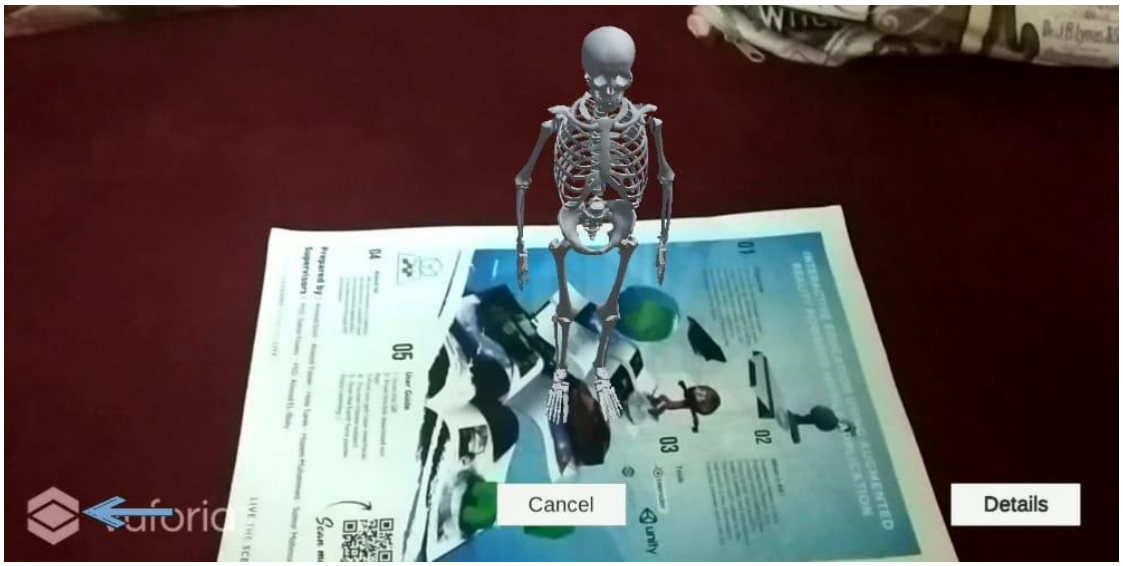
- This scene shows the earth in AR Camera with possibility to show and hide the continents, resize, rotate earth, and return to the solar system scene.

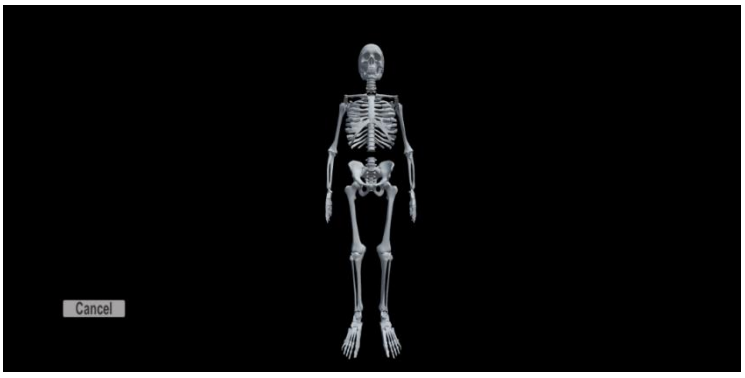


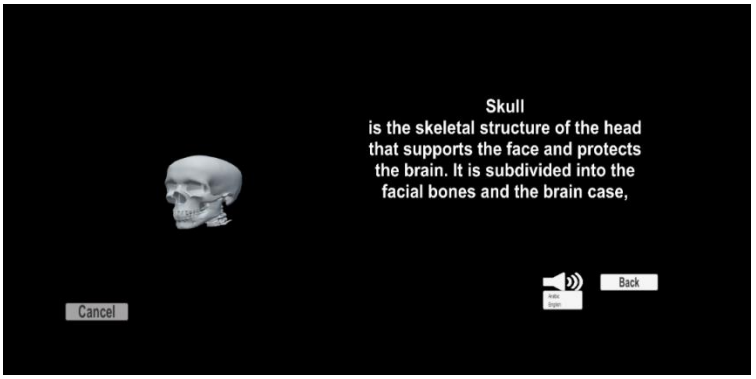
- The scene of “Human Body” shows the anatomy of heart, skeleton and lung.

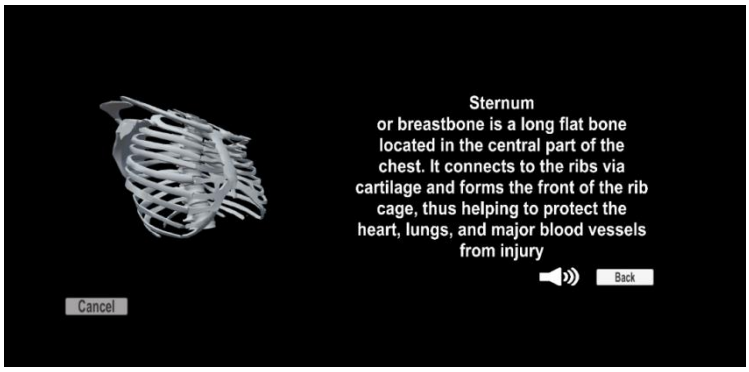


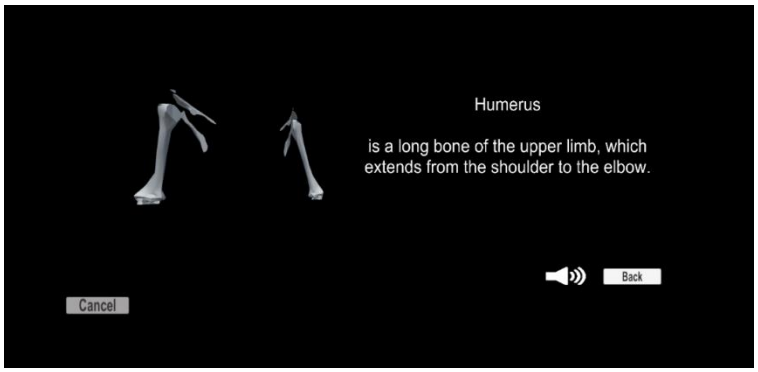
- When choosing the skeleton, the scene shows it in the middle with possibility to resize and rotate it, with a button to move to another scene with more details about skeleton parts with voice to explain this information.

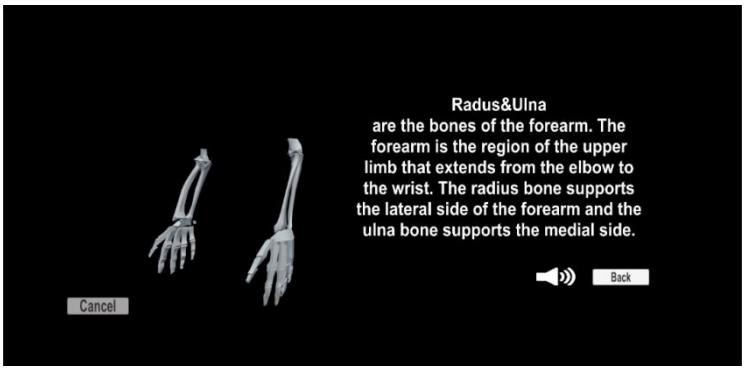


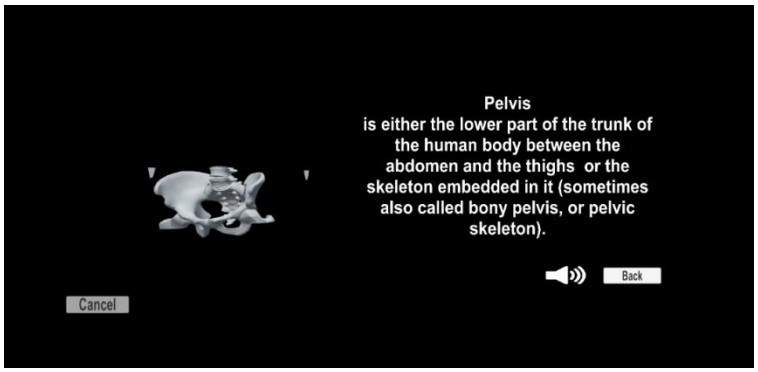


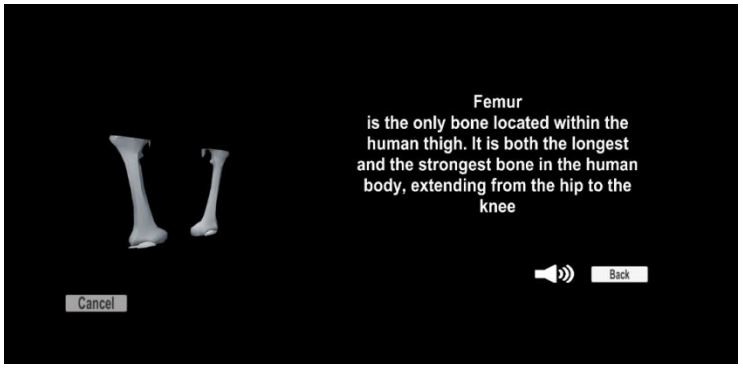


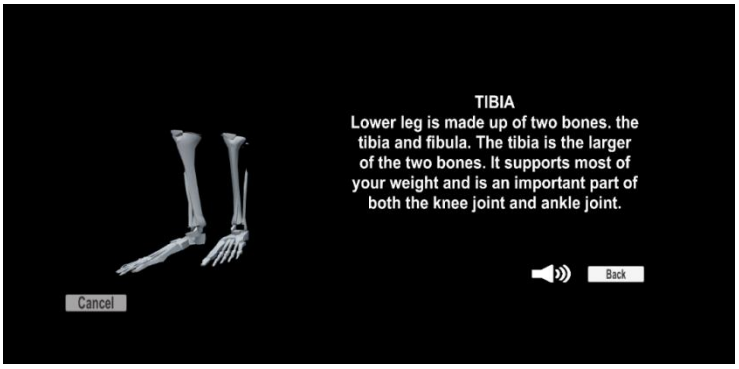




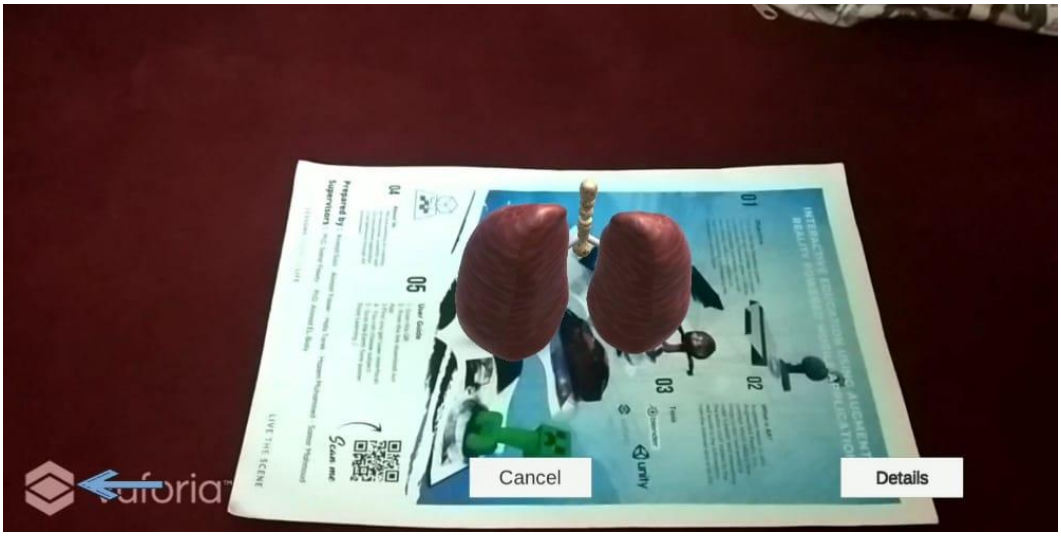


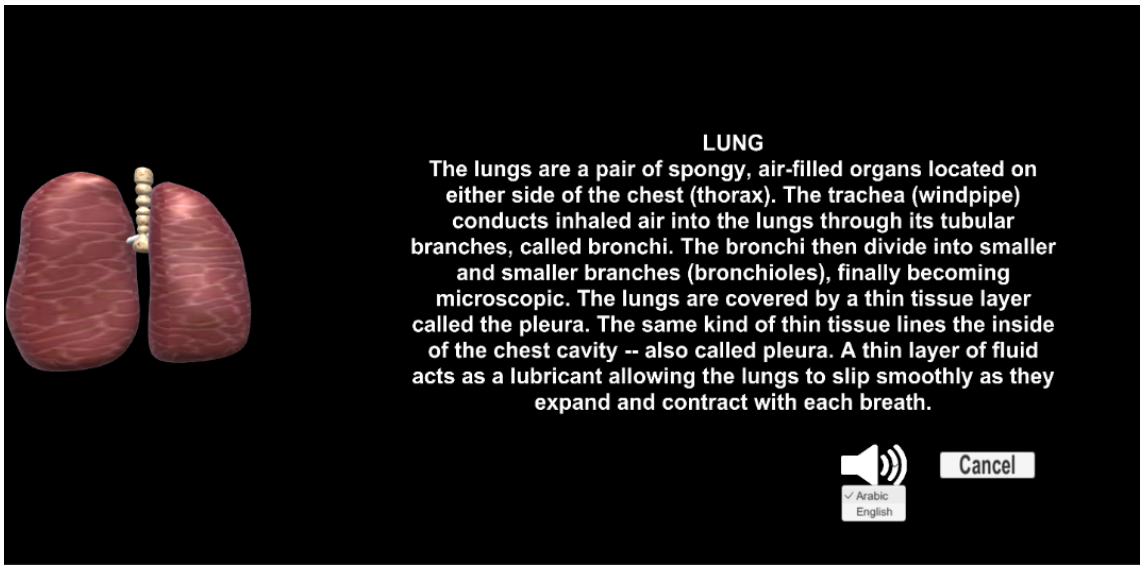




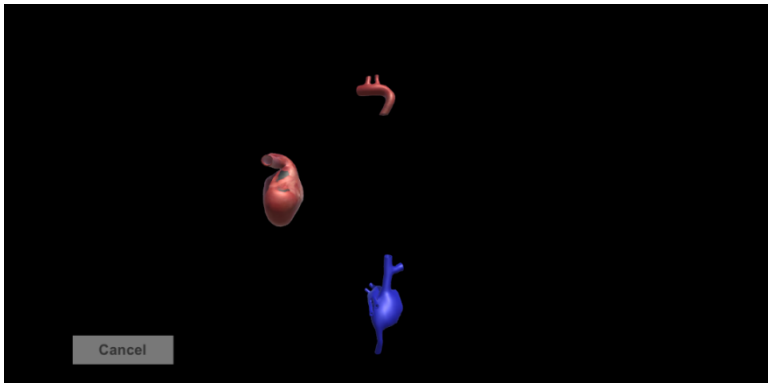


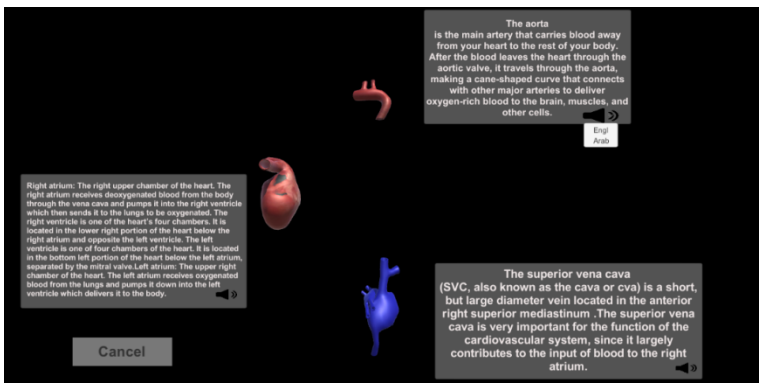
- When choosing lung, the scene shows it in the middle with the possibility to resize and rotate it, use a button to move to another scene with more details about the lung with voice to explain it.





- When choosing heart, the scene shows it in the middle with possibility to resize and rotate it, with a button to move to another scene with more details about the heart with voice to explain its parts.

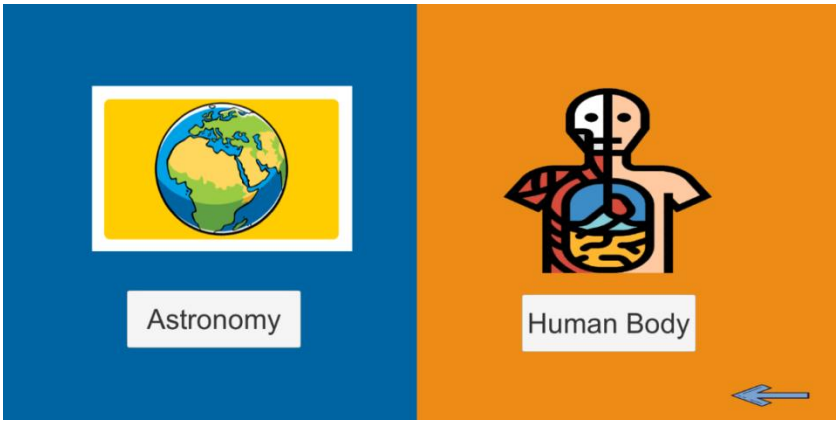




- Using cancel and back buttons we can return to the second scene , let's try "Marker-less AR"

Marker-less AR

- After choosing “Marker-less”, user can choose a subject to learn about

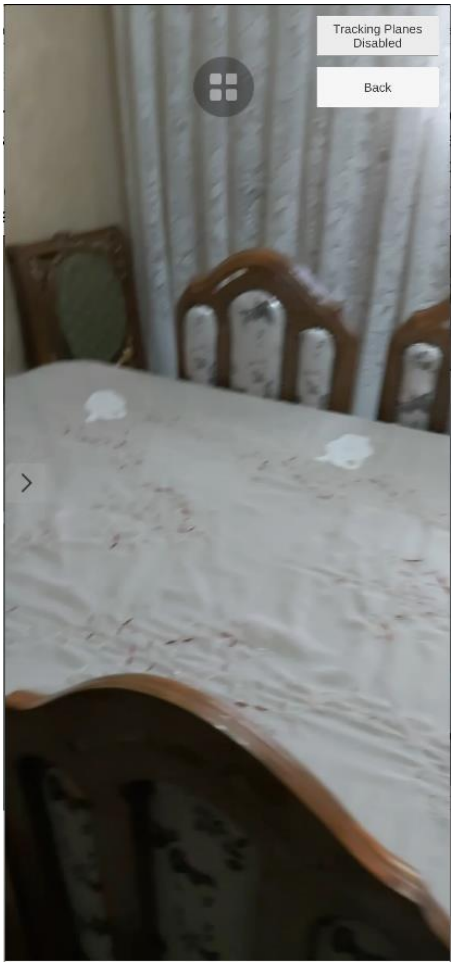


- When choosing “Human Body” an AR session begins , the user is instructed to Move the device and point the camera towards horizontal planes in the physical world around him so that the application can detect horizontal surfaces and planes
Once a plane is detected a virtual plane appears representing the detected plane.

- Tracking Planes button is a toggle button that allows the user to control the tracking process, When clicking the button for the first time the process is disabled and no new planes are being detected, Clicking the button again the process is enabled and detection is resumed and the user is instructed again to move the device and point the camera towards horizontal planes.

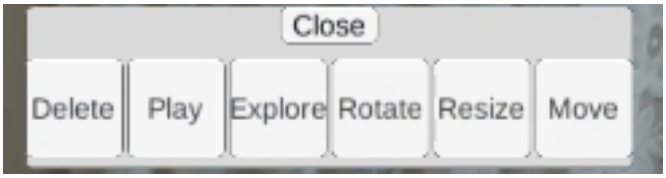
- A side panel on the left allows the user to choose 3d models related to human body

There are 4 models that the user can choose from including Human Skeleton, Heart, Lung and Eye by clicking the desired model button

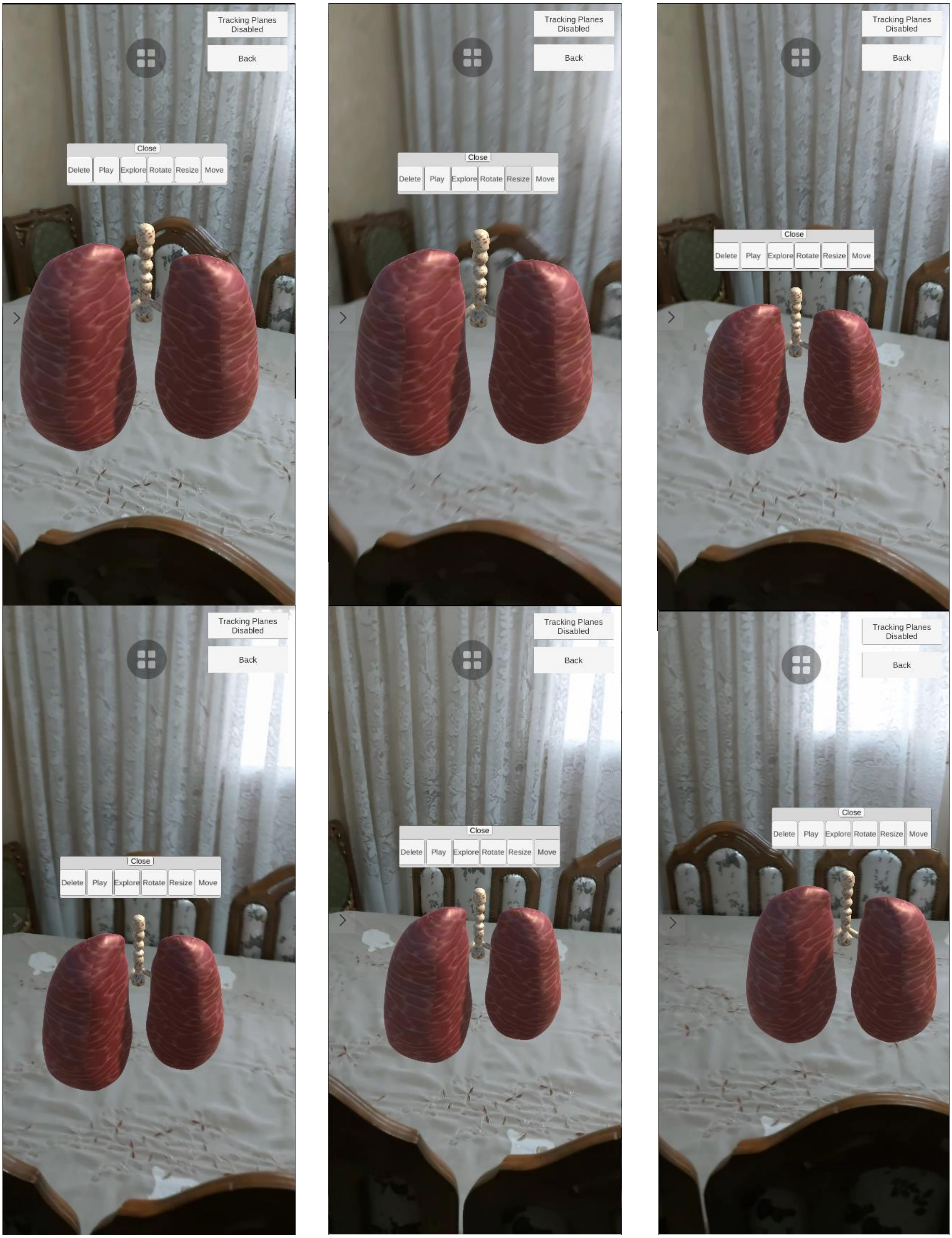


- After choosing a model by clicking on its button, then the user need to touch the screen so that the model can be instantiated and placed on a specific area within the detected plane corresponding to the user touch.

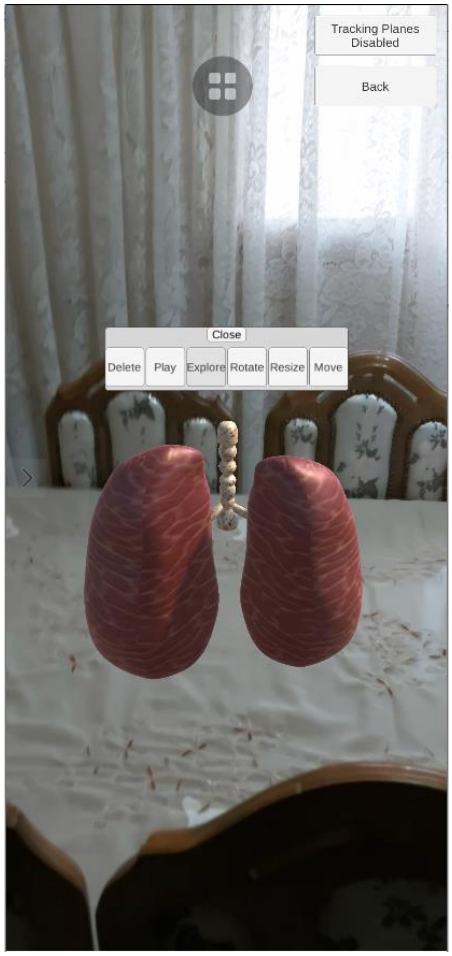
- Touching the placed model opens a popup menu containing buttons that allows The user to control and interact with the model.
  Buttons allow user to move, resize, rotate, explore the model and learn about it, See animation playing and deleting the model.

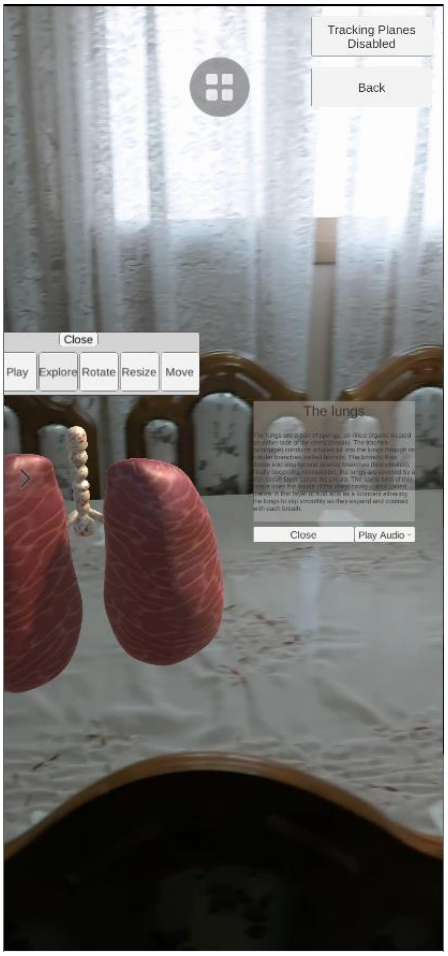


- To move the model the user simply needs to Click Move button then drag the model to the desired position within the plane or even in another detected plane then drop it.
- To resize the model the user simply needs to Click Resize button then pinch using 2 fingers, Moving the 2 fingers in a manner that increases the distance between them leads to scaling up the model similarly decreasing the distance scales down the model
- To rotate the model the user simply needs to Click Rotate button then swipe Right or left to rotate the model horizontally anticlockwise or clockwise.
  Up or down to rotate the model vertically anticlockwise or clockwise.

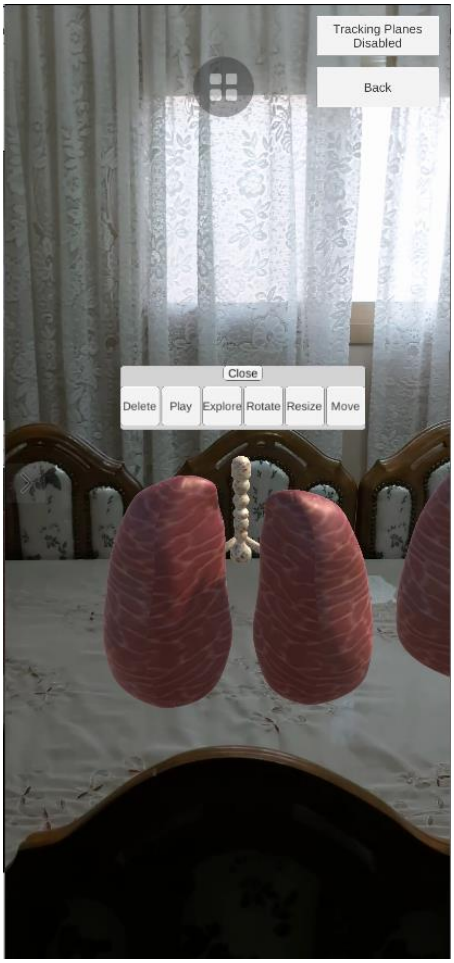

Tracking Planes Disabled
Back
Close
Delete
Play
Explore
Rotate
Resize
Move

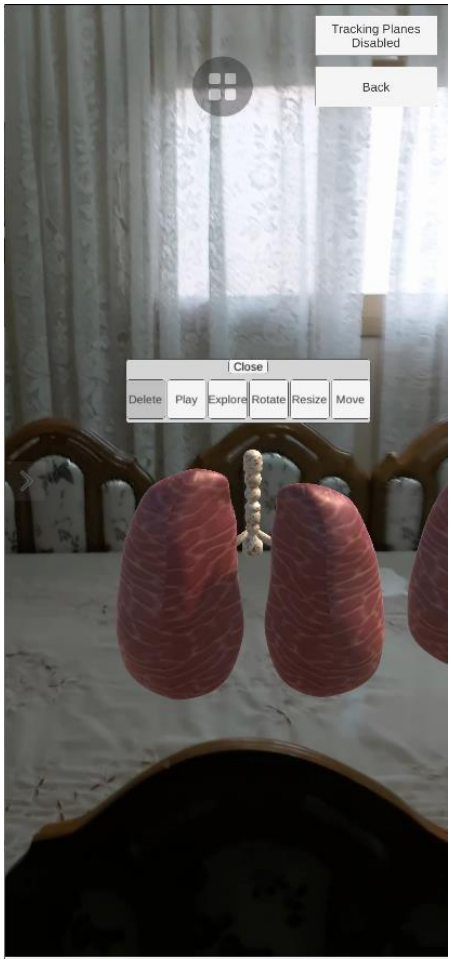
- On Pressing Explore button an information panel appears that provides the user with information about the model, the user can also hear the information in Arabic or English, by clicking and selecting language in Play Audio dropdown button.





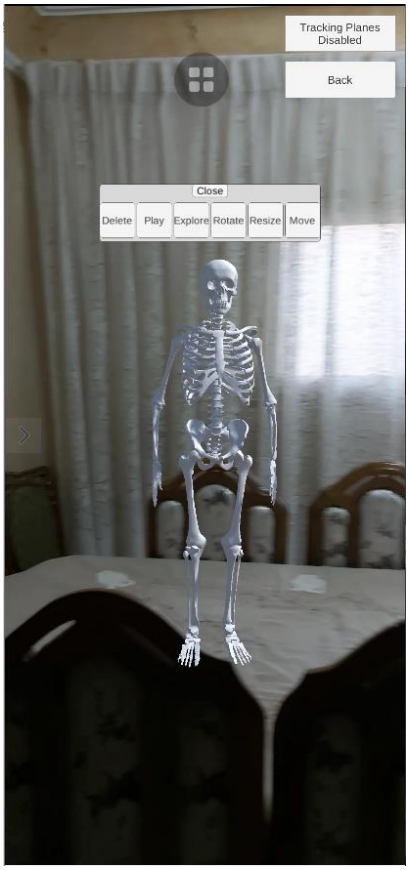
- Pressing Delete Button destroys the object of the model and removes it

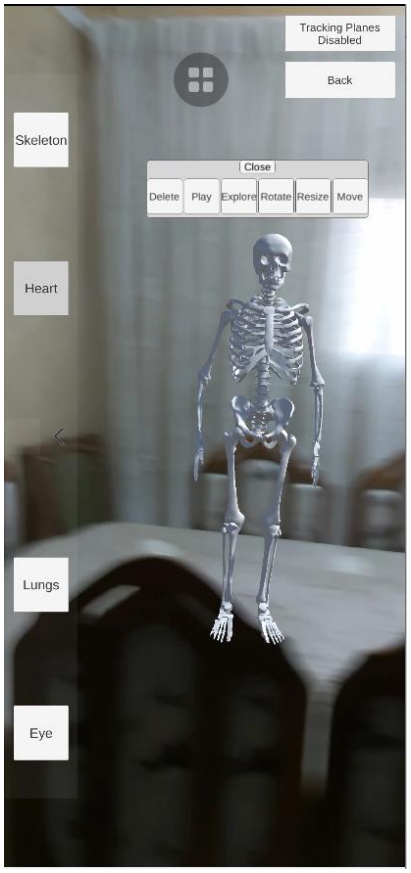


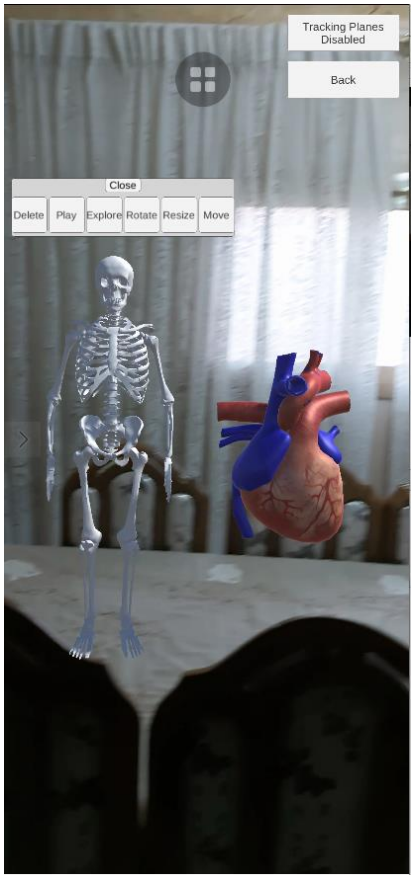


- The user can select another model and control it in the same way also more than one instance of the model can be placed on the plane as well as different models with many instances.

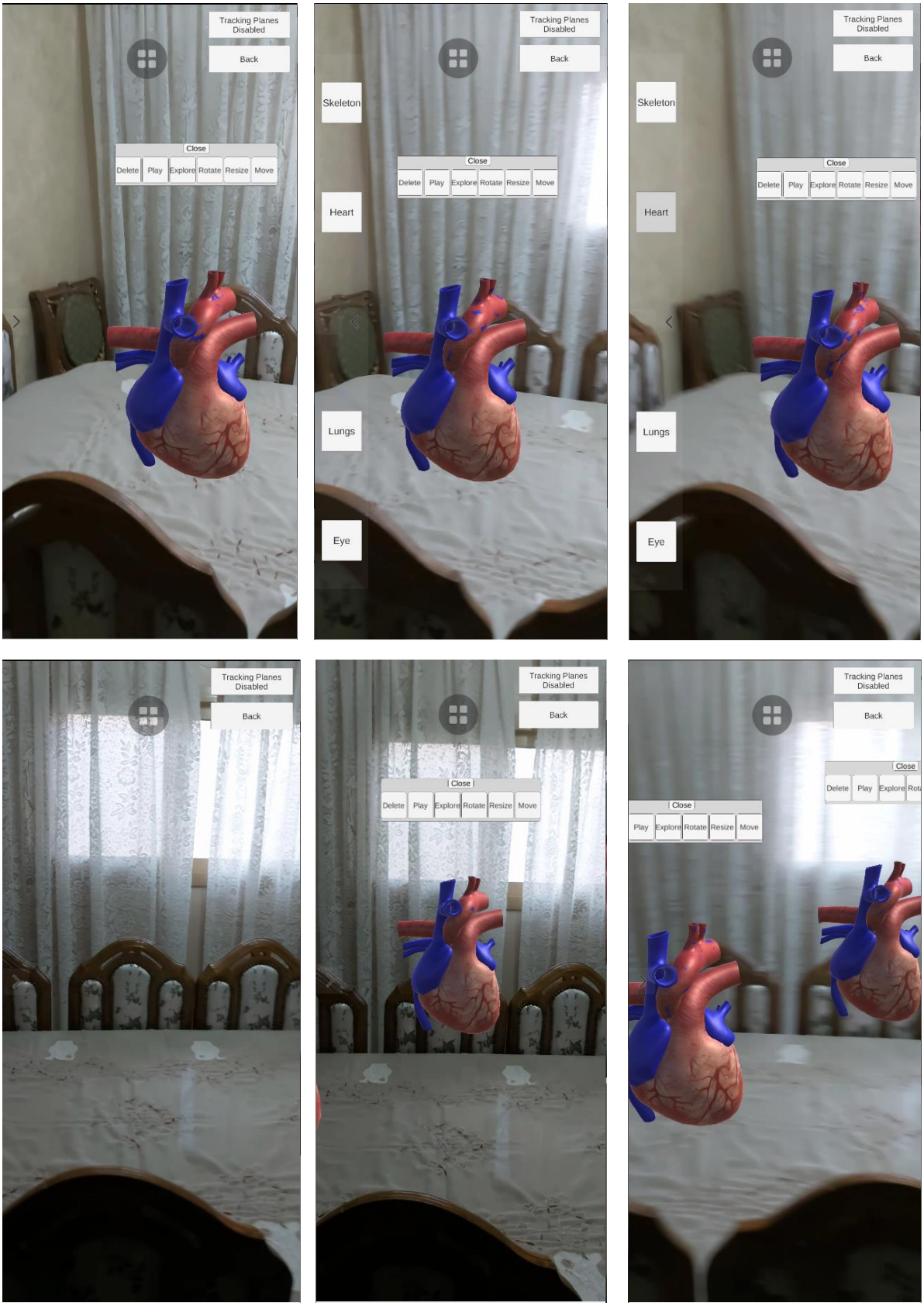
- 2 Different Models





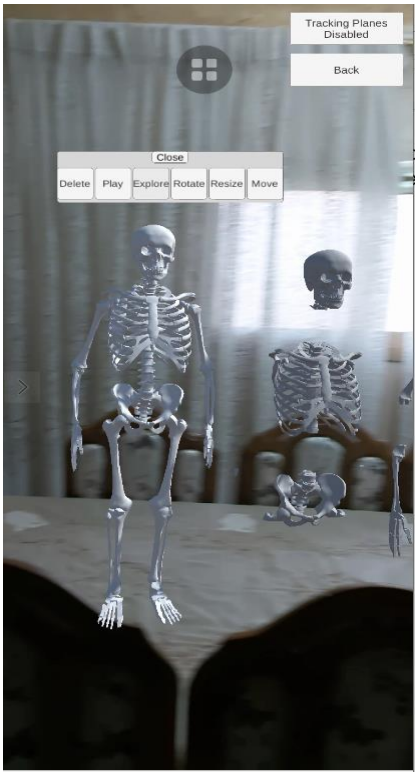


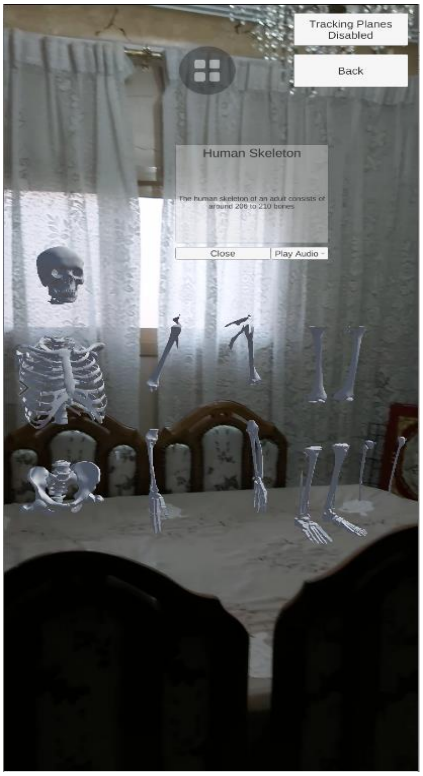
- Same model, 2 instances



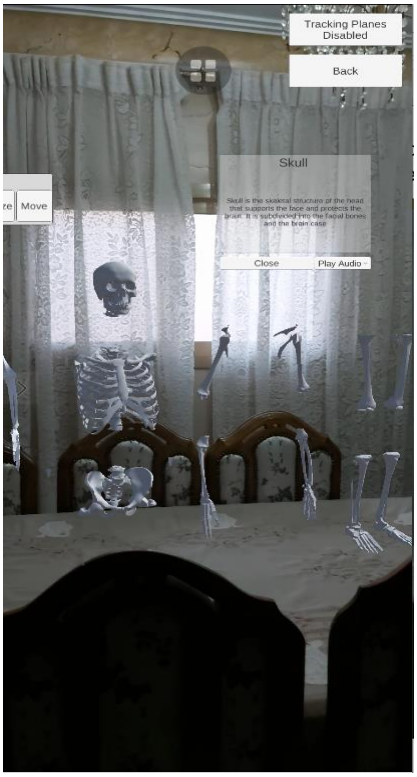
- When the user presses Play Button an animation starts playing, making the learning experience more entertaining.

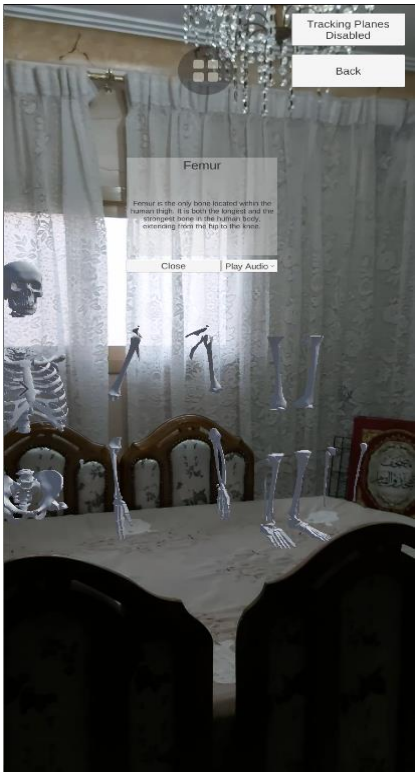
- The user can also explore model parts by simply pressing on the desired part after pressing explore button in the pop up menu that opens the information panel. Also listen to information in Arabic or english

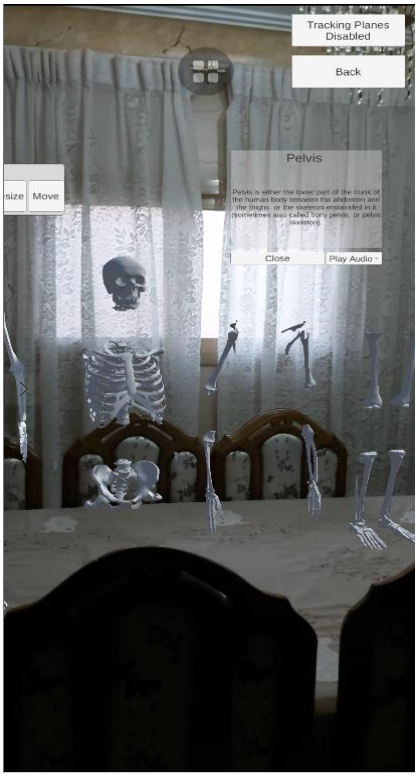




- Selection the skull Changes the information to information about skull, similarly for other parts

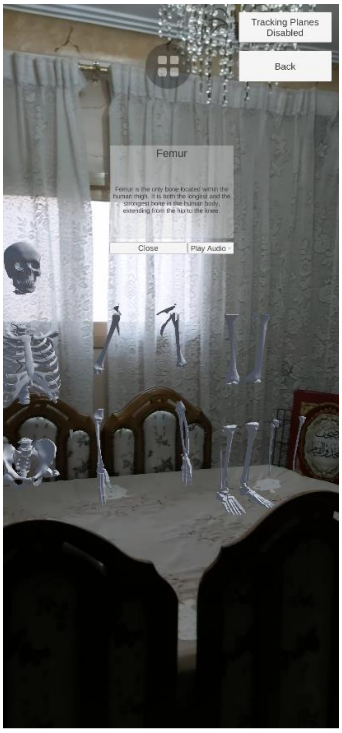


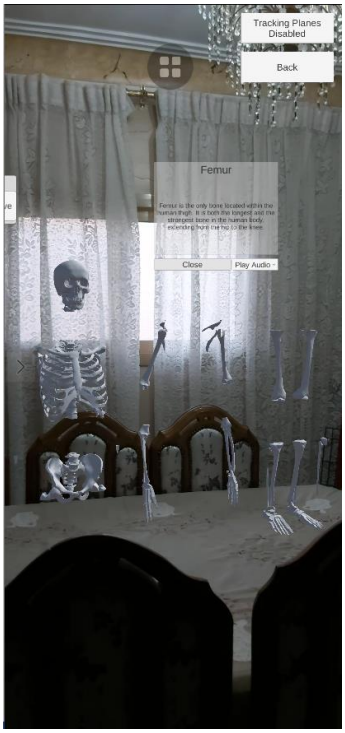


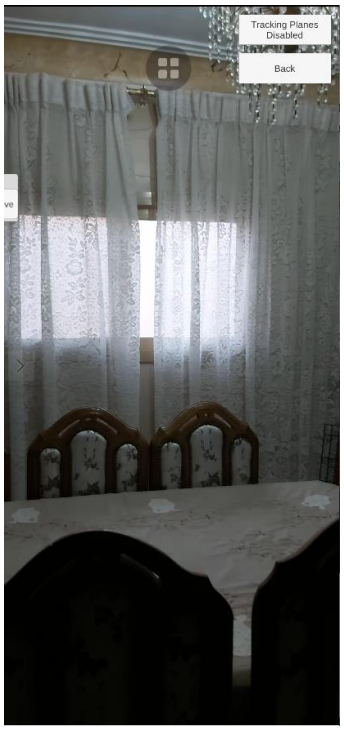


- The user can move close to the model to zoom in and around the model to see the details of the model.

- The user can click Close button to close the Information Panel and Back button to return to Subject selection menu.







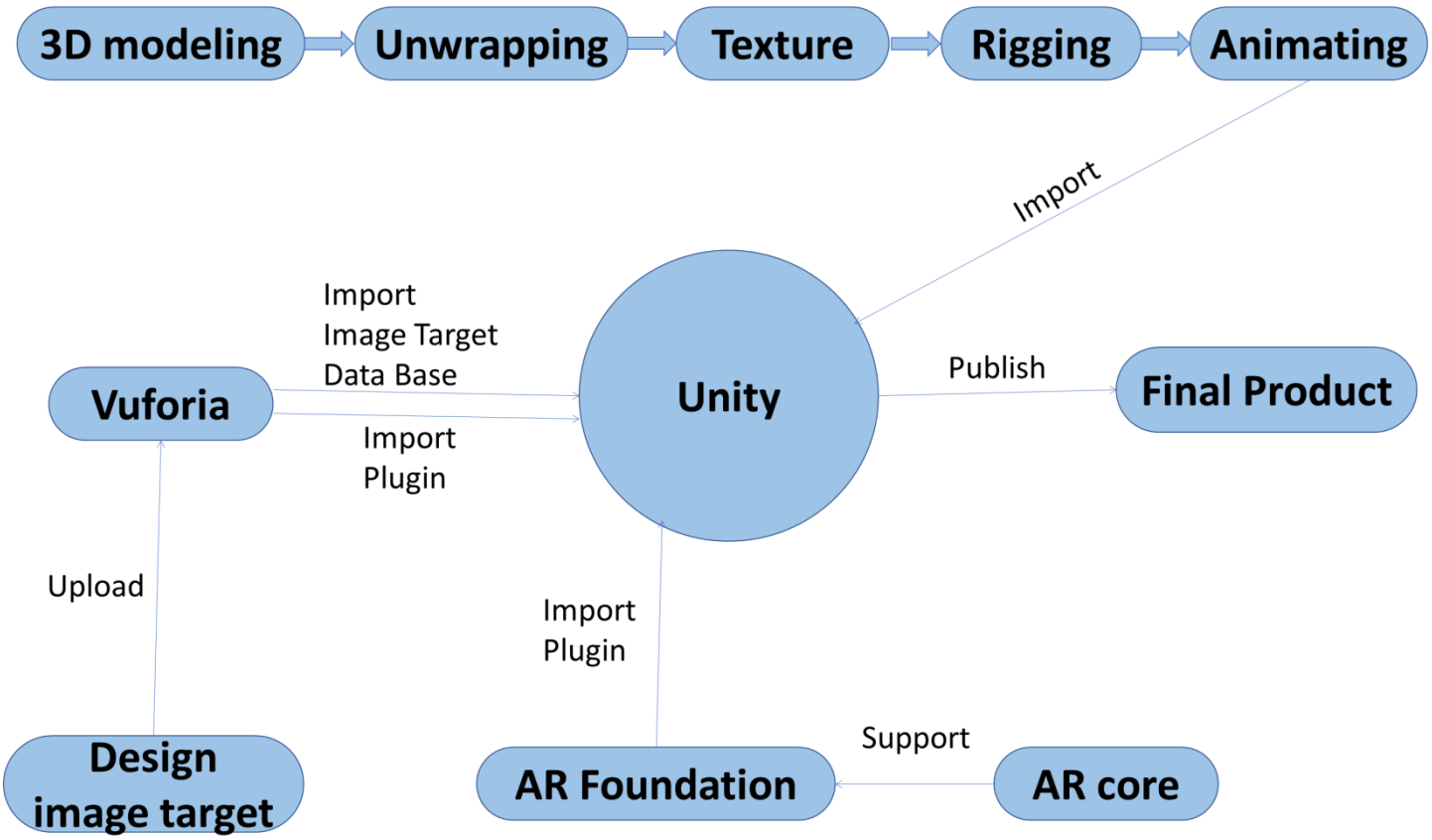
- After returning to the subject menu the user can select another language to learn about Choosing “Astronomy” opens an AR session that contains another models Including Solar System model and models of several Planets.

- User can select model to place from the left side panel including Solar System, Sun, Mercury, Venus, Earth & Moon and Mars.

- User Can control and manipulate placed model and animation of the model from the right side panel Including move, rotate, resize, explore, play animation, stop animation, reset and delete

CHAPTER 6
Conclusion

In recent years Augmented Reality (AR) has developed rapidly and is widely used in education teaching due to its strong sense of immersion and interaction. In this context, this paper analyzes the key technologies of AR and designs a complete display system of Earth, human skeleton and ability to add any subject with interactive functions. Specially, this system is packaged into an android app where The Egyptian Ministry of Education provided android devices for students. In our application we used Unity as main platform, Blender to create our graphic content (3D models) and Vuforia SDK based on marker then it was converted to Mark-less AR. We also modified scripts using visual studio and converted our work to mobile application with built in android studio. We created three models (1) Earth (2) Skeleton (3) Creeper and put all parts and explanations on models. And we could zoom in/out model to view more details about models and we could rotate and move models from its location, and users could listen to explanation which make learning easier. After all, we can add many fields of education with many virtual objects. Marker-less Augmented Reality's ability to overlay digital information on the real world is a boon to educators, who are using the tools to better illustrate complex concepts for students. And from its education becomes more enjoyable and beneficial.



REFRENCES

- C. L. a. B. Tang, "Research on The Application of AR Technology Based on Unity3D in," Conference Series, 2019.
- R. M. Yilmaz, "Augmented Reality Trends in Education between 2016 and 2017 Years," 9 Oct. 2017. [Online]. [Accessed 23 May 2018].
- MATLAB. documentation, 9 Apr. 2014. [Online]. Available: https://www.youtube.com/watch?v=JIh_rE1IcQc&feature=youtu.be.
- (Unity, Class Audio Source)https://docs.unity3d.com/Manual/class-AudioSource.html
- (Unity, UI Canvas) https://docs.unity3d.com/Packages/com.unity.ugui@1.0/manual/UICanvas.html
- (Unity, UI Canvas)https://docs.unity3d.com/ScriptReference/Canvas.html
- (Augmented Reality Trends in Education: A Systematic Review of Research and Application)
- Hedgehog Labeling: View Management Techniquesfor External Labels in 3D Space Markus by Tatzgern Denis, Kalkofen Raphael, Grasset Dieter, Schmalstieg Graz University of Technology
- Abas, H., and Zaman, H. B. (2011). “Visual learning through AR storybook for remedial student,” in LNCS, Vol. 7067, eds H. B. Zaman, P. Robinson, M. Petrou, P. Olivier, T. K. Shih, and S. Velastin (Berlin; Heidelberg: Springer), 157–167.
- Bülbül, M., Yigit, N., and Garip, B. (2016). Adapting smart phone applications about physics education to blind students. J. Phys. Conf. Series 707:012039. doi: 10.1088/1742-6596/707/1/012039
- Chen, C. H., Lee, I. J., and Lin, L. Y. (2015). AR-based self-facial modeling to promote the emotional expression and social skills of adolescents with autism spectrum disorders. Res. Dev. Disabil. 36C, 396–403. doi: 10.1016/j.ridd.2014.10.015
- Espinosa, C. (2015). AR and education: analysis of practical experiencies. Pixel-Bit. Rev. Med. Educ. 46, 187–203. doi: 10.1016/0140-1971(92)90048-A
- Hsu, Y.-C., Hung, J.-L., & Ching, Y.-H. (2013). Trends of educational technology research: more than a decade of international research in six SSCI-indexed refereed journals. Educational Technology Research and Development, 61(4), 685–705. doi:10.1007/s11423-013-9290-9
- J. Petty, "Concept Art Empire," [Online]. Available: https://conceptartempire.com/what-is3dmodeling/#:~:text=3D%20modeling%20is%20a%20technique,vertices%20that%20form%20an%20object..
- T. Denham, "Concept Art Empire," [Online]. Available: https://conceptartempire.com/uv-mapping%20%20unwrapping/#:~:text=A%20UV%20map%20is%20the,used%20in%20the%203D%20space..

- S. Mayekar, "Analytics Insight," 22 June 2018. [Online]. Available: https://www.analyticsinsight.net/future-ar-slam-technology slam/#:~:text=SLAM%20is%20short%20for%20Simultaneous,augmented%20reality%20(AR)%20science..
- ARCore. documentation. [Online]. Available: https://developers.google.com/ar/discover/concepts#environmental_understanding.
- (ARCoredocumentation)https://towardsdatascience.com/getting-started-with-augmented-and-virtual-reality-a51446661c3
- (SDKS) https://dzone.com/articles/12-best-augmented-reality-sdks
- (VuforiaImagetargetdocumentation) https://library.vuforia.com/articles/Training/getting-started-with-vuforia-in-unity.html https://library.vuforia.com/content/vuforia-library/en/articles/Training/Getting-Started-with-the-Vuforia-Target-Manager.html https://library.vuforia.com/features/images/image-targets.html
- (Marker-LessARdocumentation) https://www.analyticsinsight.net/future-ar-slam-technology-slam/#:~:text=SLAM%20is%20short%20for%20Simultaneous,augmented%20reality%20(AR)%20science.
- (Marker-LessARdocumentation)https://www.coursera.org/learn/ar-technologies-video-(Marker-LessARdocumentation)streaming/lecture/1NT5I/3-1-ar-feature-detection-description-method-comparison
- (Marker-LessARdocumentation)https://developers.google.com/ar/discover/concepts
- (Marker-LessARdocumentation)https://www.wikitude.com/wikitude-slam/#:~:text=SLAM%20(Simultaneous%20Localization%20and%20Mapping,to%20overlay%20digital%20interactive%20augmentations.
- (ARFoundationddocumentation)https://docs.unity3d.com/Packages/com.unity.xr.arfoundation@2.2/manual/index.html
- (ARFoundationddocumentation)https://docs.unity3d.com/Packages/com.unity.xr.arfoundation@2.2/manual/plane-manager.html