3D Euclidean Space AR App Documentation

Development Period: April 2019 – August 2019

Development Environment

1. Hardware

- Development Stations
  - Alienware Aurora R8
    CPU : Intel Core i7-9700K@3.6 Ghz
    RAM : 16GB
    OS : Windows 10, 1809 Build
    GPU : Nvidia RTX2080

  - iMac 27 inch Retina 5K (Late 2019)
    CPU : Intel Core i5@3 Ghz
    RAM : 32GB
    OS : macOS Mojave 10.14.4
    GPU : AMD Radeon Pro 570X

- Testing Devices
  - Nokia 8 (TA-1004)
    AP : Snapdragon 835
    RAM : 4GB
    Screen : 1440 X 2560 pixels (554 ppi)
    Camera : 13MP, f/2.0
    OS : Android 9 (Pie)

  - iPad Mini 5 (2019)
    AP : A12 Bionic
    RAM : 3GB
    Screen : 1536 X 2047 pixels (326 ppi)
    Camera : 8MP, f/2.4
    OS : iOS 12.3.1

2. Software

- Windows PC
  - Unity3D Game Engine 2019.1.6f Build – Setting up the scene and object interaction
    Imported Packages(Package manager) : AR Foundation 2.1.0, ARCore XR Plugin 2.1.0, TextMesh Pro
    Imported Packages from the web : Oculus Lipsync Unity SDK 1.39.0, UniVRM 0.53.0
  - VSCode 1.36.1 - Text editor for scripting
    Extensions : C# 1.21.0 by Microsoft
    Debugger for Unity 2.7.2 by Unity

Unity Code Snippets 1.3.0 by Kleber Silva

- Blender 2.80 - 3D modeling, converting model format and modifying animations
  Add-ons : VRM Importer for Blender 2.80 - import VRM model to Blender
  Cats Blender Plugin 0.14.0 - optimize the imported VRM model
- Vroid Studio 0.6.3 – 3D Character generation(VRM format)
- Gimp 2.10.8 - Creating and modifying 2D graphic textures and fonts
- Adobe Mixamo – Free web service providing animation assets
- Amazon Polly – Text-To-Speech web service
- Codebeautify.org - YAML to JSON converter web service

- iMac
  - Unity3D Game Engine 2019.1.6f Build
    Imported Packages(Package manager) : AR Foundation 2.1.0, ARKit XR Plugin 2.1.0, TextMesh Pro
    Imported Packages from the web : Same as Windows PC
  - VSCode 1.36.1 - Installed same extensions as Windows PC
  - Xcode 12 – Updated from the Apple App store

- Nokia 8
  - ARCore by Google from Google Play (Only available for the ARCore enabled devices)

- iPad Mini 5
  - TestFlight from Apple App store – Downloading test version of the app

Development Process

Graphic tools used in this project perform better on the Window PC. For that advantage, the entire project was built on the Windows PC first, and then transferred to iMac as a Unity package file. Files and data transfer between software and libraries is summarized and visualized in the following diagram.

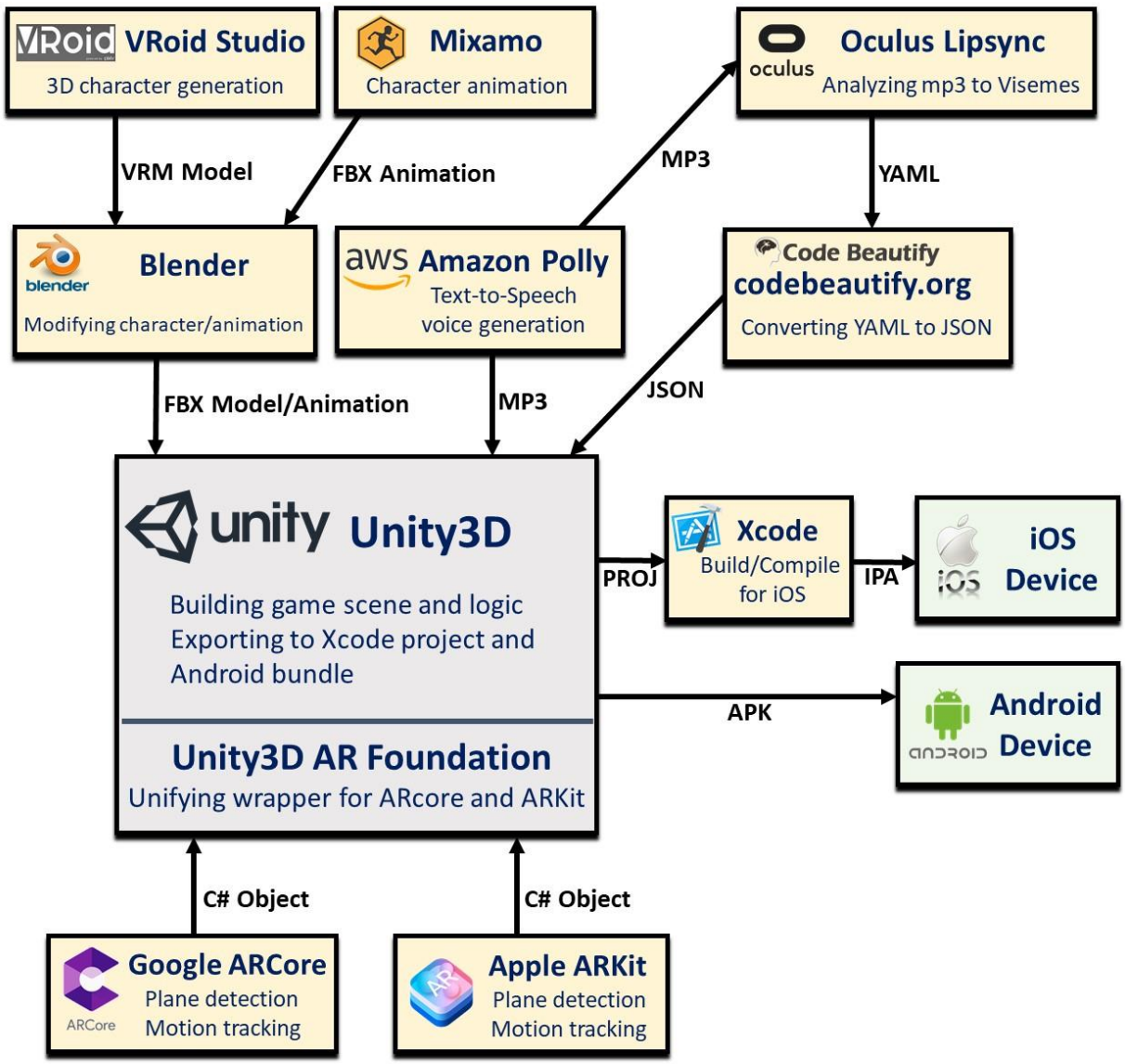

Figure 1. File/Data transfer between software and libraries

1. Asset: Generation, Importing and Set-up in the Unity Editor

I. Character Model Generation

- *Software*
  - VRoid Studio 0.7.2 - Download from the VRoid website. vroid.com/studio
  - Blender 2.80 - Download from the Blender website. www.blender.org
  - Gimp 2.10.12 - Download from the Gimp website. https://www.gimp.org
  - Cats Blender Plug-in 0.14.0 - https://github.com/michaeldegroot/cats-blender-plugin/releases
  - VRM Importer Plug-in for Blender 2.80 - https://github.com/iCyP/VRM_IMPORTER_for_Blender2_8/archive/master.zip

- *Process*
  a. Installed VRoid Studio loaded the preset "Sendagaya_Shibu" model.
  Under Face Editor, adjusted Outer Eye Slant(downward) to max.
  Under Hair Editor, removed the hairpin from the hair group(6th element).
  Under Body Editor, adjusted Head Width(downward) to 0.50., Breast Smallness(for Base) to 0.40, Waist Smallness to 0.50.
  Under Clothing Editor, change the outfit to Uniform(pants) and Uniform Vest(long sleeve) and adjust Cuff Tightness and Pants Crease to max, and Sleeve Crease to 0.
  Under Cloth Editor -> Texture tab, export Default image and import that texture(PNG) from Gimp. In Gimp, changed the hue of the vest into green and added COL logo on the vest with cartoon filter. Exported it as a new image from Gimp. Back in VRoid Studio, import this new image on Default image. Changed the Ribbon color into dark navy and pants color to dark grey using the same method. Saved and exported as "skinnysally".
  b. Installed Blender 2.80 and installed Cats Blender Plug-in and VRM Imported plug-in under Edit->Preference->Add-on. Cats and VRM Importer are found under Community and Testing tab, respectively.
  Import skinnysally.vrm to Blender.
  Opened the add-ons tap by clicking the tap expander on the right upper size of the main view port and expanded the CATS add-on located at the very bottom the expander.
  Clicked on the "Fix" button and "All" button in Translate(marked with red boxes in Figure 2).
  Made a new folder and exported the model into that folder as FBX in the export menu.

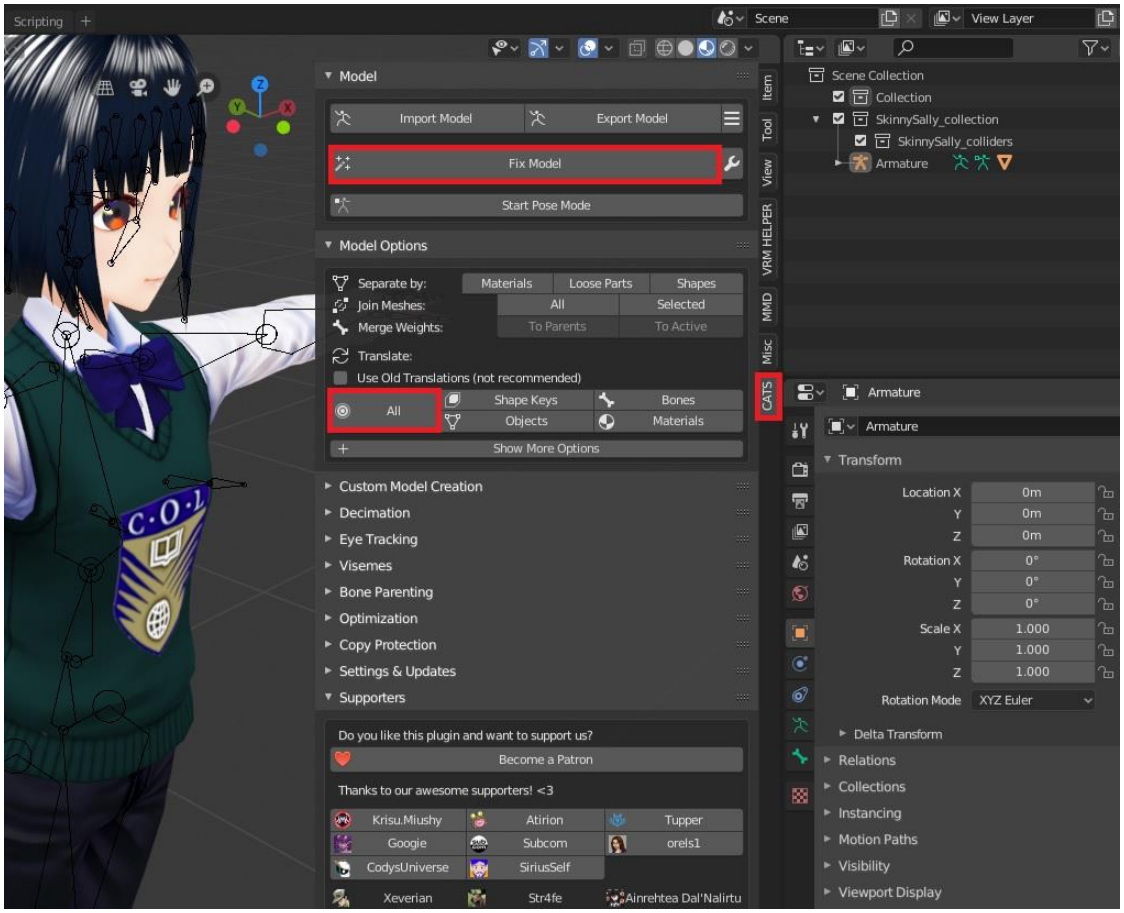

Figure 2. Blender CATS add-on expander screen

II. Voice and Lip Sync Asset Generation

- *Software*
  - Amazon Polly – Requires AWS account. https://aws.amazon.com/polly
  - Unity3D – Download from Unity website. https://unity.com
    From 2019, Installing Unity Hub is required to be able to install Unity3D. At Unity's webpage, downloaded and installed Unity Hub and Unity 2019.1.6f and Android build support with SDK/NDK and OpenJDK (iOS build support for iOS Version) was install through Unity Hub.
  - Oculus Lipsync Unity SDK 1.39.0 - Download from Oculus website.
    https://developer.oculus.com/downloads/package/oculus-lipsync-unity
  - Codebeautify.org - Free web service. https://codebeautify.org/yaml-to-json-xml-csv

- *Process*
  a. Voice files was generated through Amazon Polly with voice "Salli, Female" and save as mp3. The script is based on the open source material from OpenLearn, "1.6 Point, planes, lines and distance in three-dimensional Euclidean space" with an exception of the description for "right-hand system of axes".
  https://www.open.edu/openlearn/science-maths-technology/mathematics-statistics/vectors-and-conics/content-section-1.6

b. Started a new project in Unity. Made a new folder named "Resources" and copied all the mp3 files into that folder(the folder name has to be "Resources", otherwise, the voice files won't be loaded into memory through the C# script at runtime). While Unity is running, double-clicked on the Oculus Lipsync SDK unity project file in the File Explorer and import the whole packaged to Unity.

c. In the Unity Editor, selected all the mp3 files in the Resources folder. While files being selected, clicked on the Oculus->Lip Sync-> Generate Lip Sync Assets With Offline Model tab(Figure 3) to generate YAML files that store Visemes information for each voice file. The file names are "voicefilename_lipsync.asset".

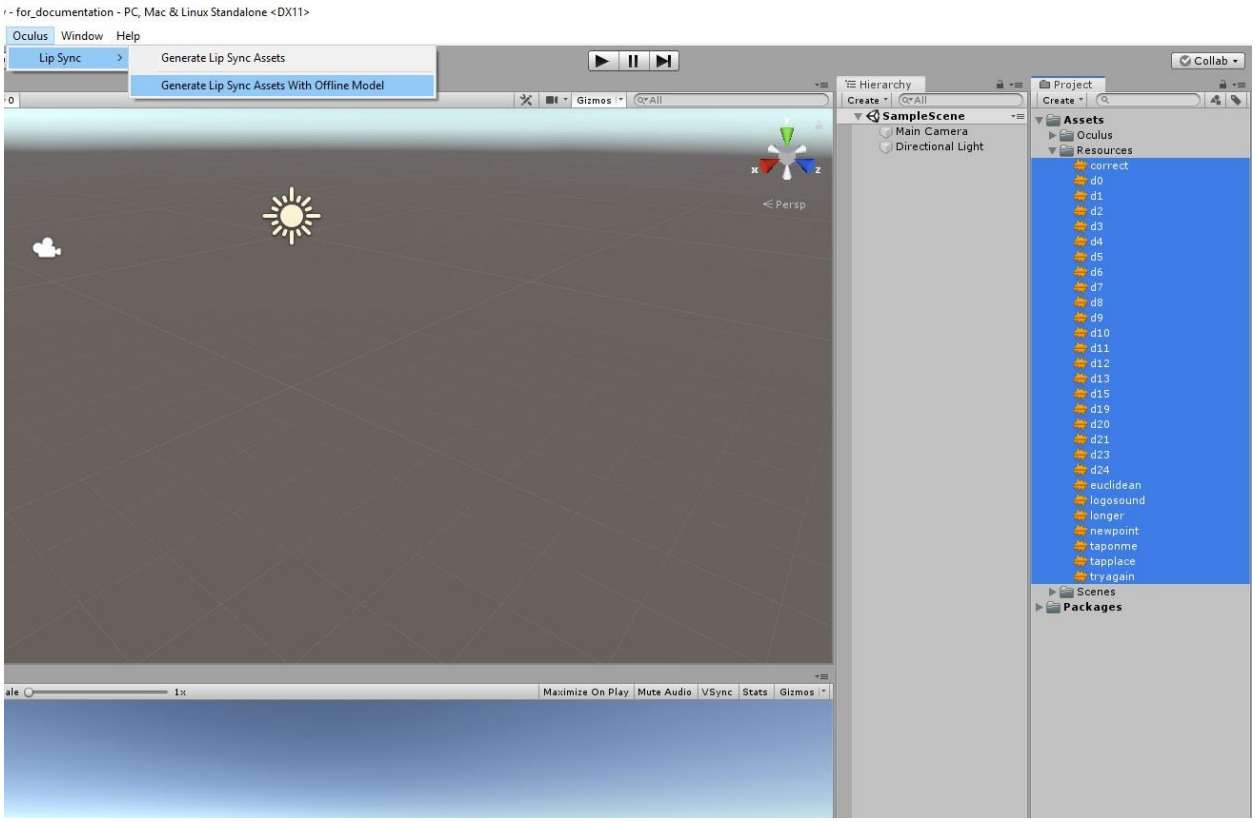

Figure 3. Pre-computing visemes from audio files and storing as YAML file(.asset) with Oculus Lipsync SDK

d. On codebeautify.org, import each YAML files one by one, converted them into JSON format and save them manually in VSCode. First, opened the YAML file(.asset) in VSCode and copied and pasted the contents into YAML Input on Code Beautify YAML to JSON conversion page. Clicked on the "YAML TO JSON" button(marked by the red box in Figure 3). Selected unnecessary part(from line 1 to the "m_EditorClassIdentifier" property, marked as selected in Figure 4) and deleted that part. Also deleted one curly bracket, "}" in the second last line to keep the proper JSON format leaving only two properties, "entries" and "length". Saved each file as "voicefilename.json". After converting all YAML to JSON files, deleted all YAML files and entire Oculus folder from the Unity's Asset folder.

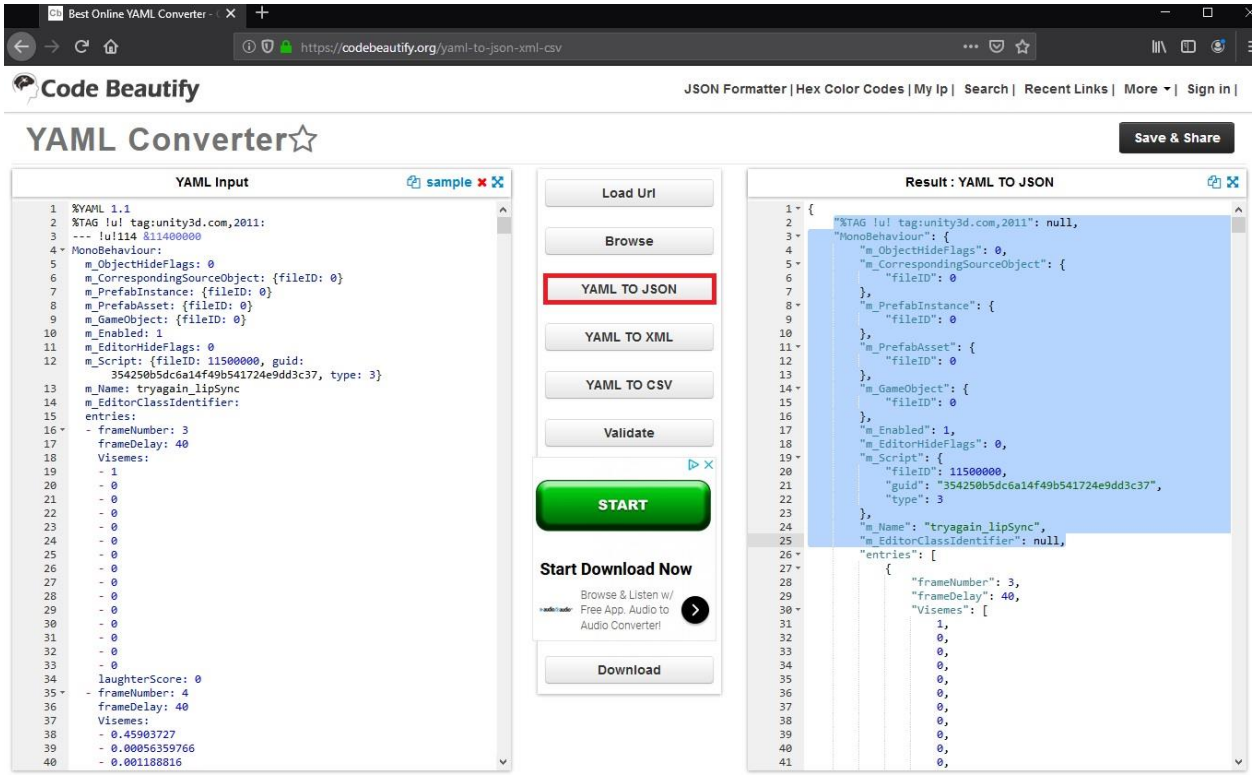

Figure 4. Code Beautify YAML to JSON converter page.

III. Importing Character Model and Texture Set-up

- *Software*
  - UniVRM 0.53.0 - Importing VRM model into Unity. https://github.com/vrm-c/UniVRM/releases/download/v0.53.0/UniVRM-0.53.0_6b07.unitypackage

- *Process*
  a. In Unity Editor, created a new folder named "skinnysally" in the Asset folder. Moved skinnysally.fbx file into that folder.
  b. Imported UniVRM 0.53.0 package into Unity by clicking Asset-> Import Package -> Custom Package tab.
  c. Imported skinnysally.vrm model into Unity by clicking the newly created menu, VRM -> UniVRM – 0.53.0 -> Import tab. The imported VRM model is saved as skinnysally.prefab (located under VRM folder, marked by yellow box in Figure 5) and 4 folders were generated under Asset folder named "skinnysally.Blendshapes", "skinnysally.Materials", "skinnysally.Meshes" and "skinnysally.Textures" after the import.
  d. Selected skinnysally.fbx model inside the "skinnysally" folder (marked by green box) by clicking it in the Project window, and clicked on the "Materials" tab of skinnysally Import Setting in the Inspector window. Expanded "skinnysally.Materials" folder by clicking on the triangle located on the left-hand side of the folder in the Project window. Under "Remapped Materials" in the Inspector window, matched all materials with the same name of materials found in the "skinnysally.Materials" folder by click/drag to the corresponding

slot(marked by blue box and arrow in Figure 5). Clicked on the "Apply" button at the bottom of the Inspector window.

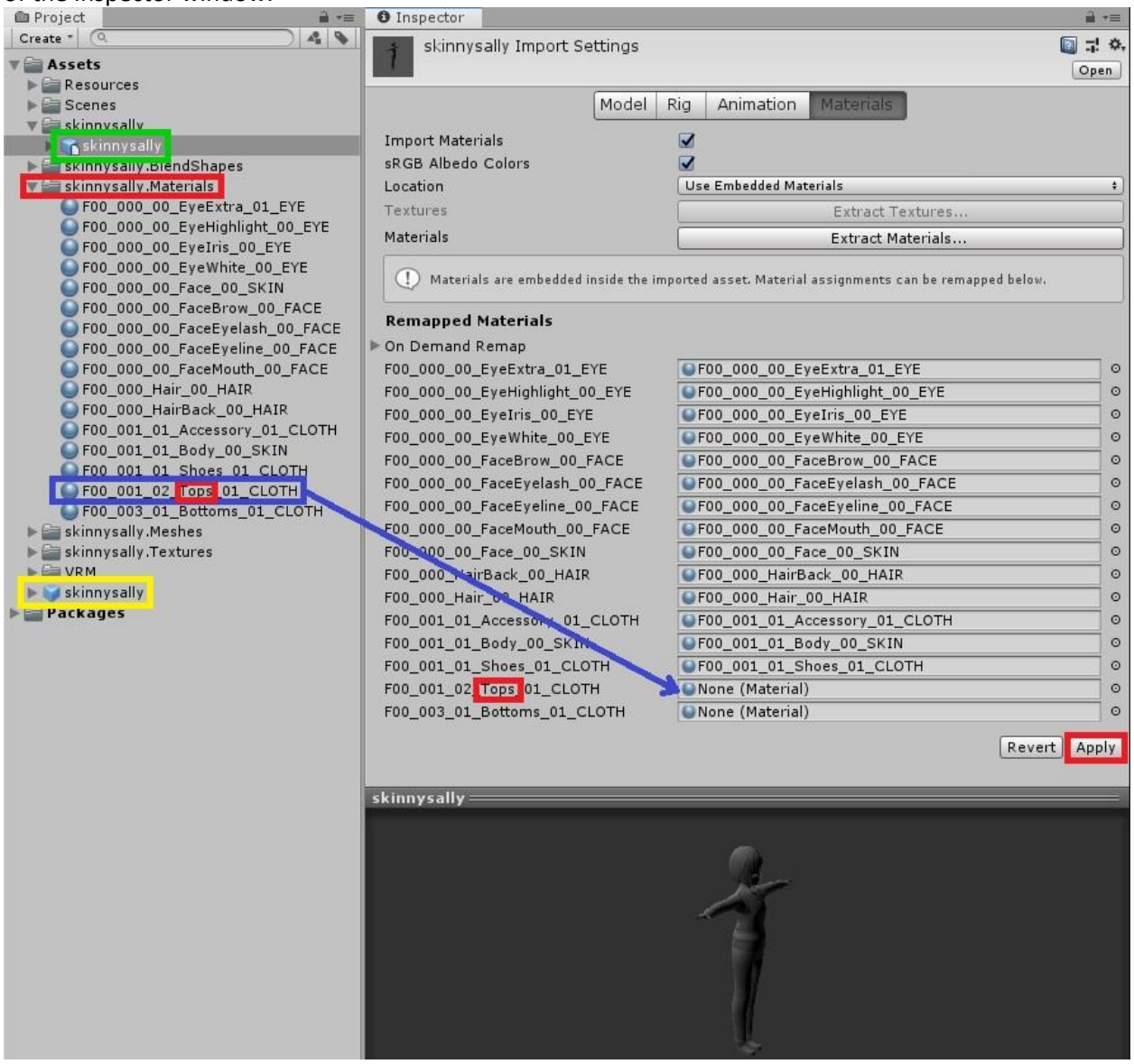

Figure 5. Importing FBX model into Unity and matching the materials with textures

e. Moved "skinnysally.Materials" and "skinnysally.Textures" folder into "skinnysally" folder and deleted "skinnysally.Blendshapes" and "skinnysally.Meshes" folder. Also deleted skinnysally.prefab (marked by yellow box in Figure 5). Finally, deleted all the folders and files under VRM folder except 3 folders under "VRM/MToon/MToon/" named "Editor", "Resources" and "Scripts" (marked by red box in Figure 6). After reorganizing the assets, the Project and Inspector windows appeared to be as shown in Figure 6.

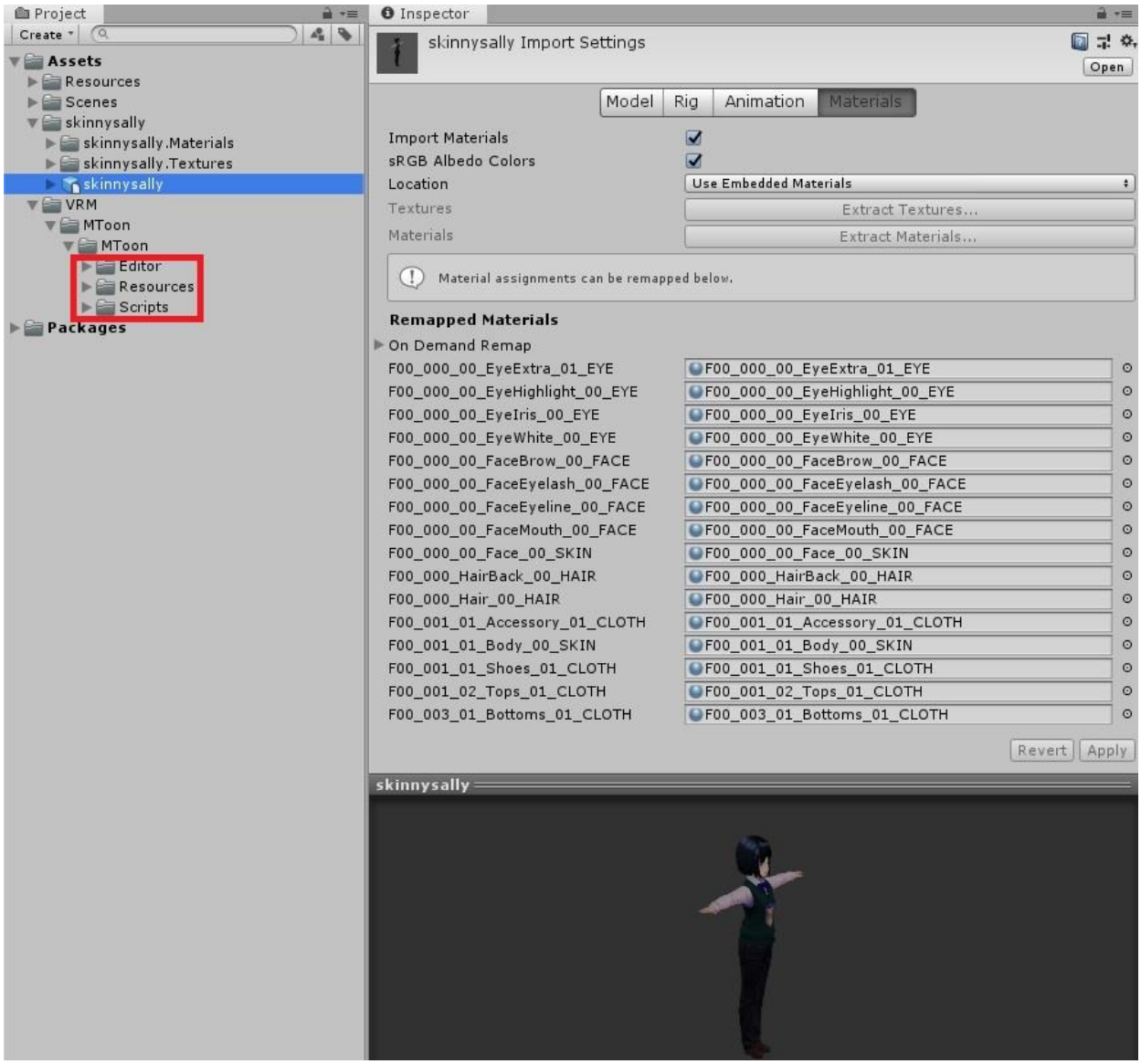

Figure 6. Project and Inspector Window of Unity Editor after asset reorganization

f. Expanded "skinnysally/skinnysally.Materials" folder and selected 6 materials, "Face", "Accessory", "Body", "Shoes", "Tops" and "Bottoms" (marked by selected in Figure 7). Under "Outline" in the Inspector window, switched the mode from "WorldCoordinates" To "None" to remove the outlines of selected materials.

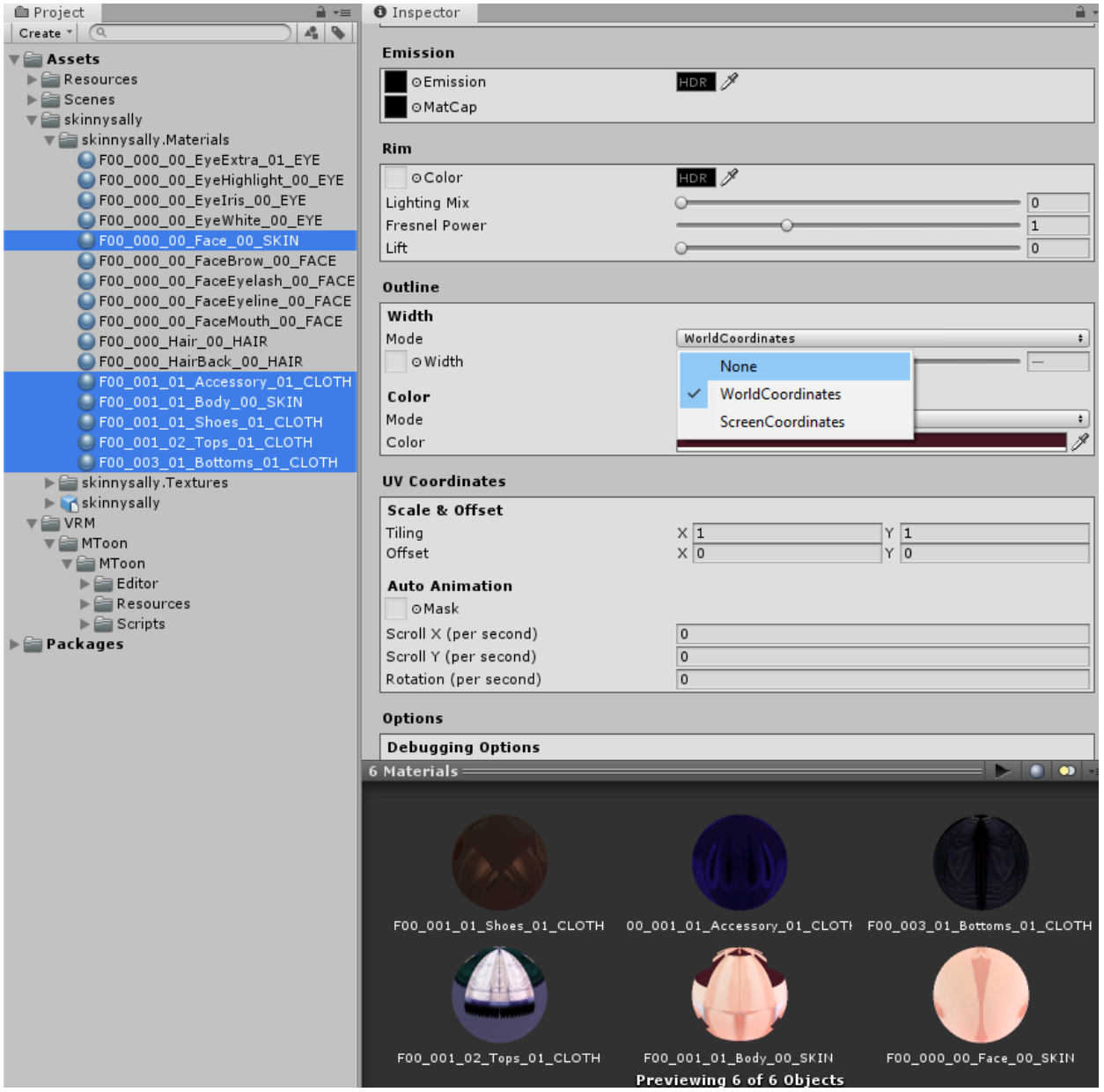

Figure 7. Removing the outlines of 6 materials

IV. Downloading and Importing Animations

- *Software*
  - Adobe Mixamo – Free web service. Requires Adobe ID. https://www.mixamo.com

- *Process*
  a. Created Adobe ID and logged in Mixamo. Found 9 hand-waving gestures appropriate for the character's dialogs. Uploaded skinnysally.fbx to Mixamo to see the preview of each animation. Adjusted arm span and the playback speed according to skinysally model and exported each animation as 60fps, FBX for Unity and without skin option.
  b. After realizing the angle of the arms or shoulder are quite awkward, the arm and shoulder position of each animation is adjusted in Blender and exported as FBX. In the Unity Editor, created a new folder named "Animations" in the Asset folder and moved the fbx animation files to that folder.
  c. Under Animations folder selected all the fbx animations just imported and clicked on the "Rig" tab under the Inspector window. Switched "Animation Type" to "Humanoid".
  d. The idle, walk and run animations were imported from "Unity Chan" model download from Unity Asset store. After downloading Unity Chan asset, imported 3 animations fbx files from "unitychan!/unitychat! Model/Art/Animations" folder. Each fbx files is named "unitychan_WAIT00", "unitychan_WALK00_F" and "unitychan_RUN00_F".

V. Instantiating the Character and Reconfiguring Bone Nodes

- *Process*
  a. Dragged skinnysally.fbx from the Project window into the Scene window to instantiate the character GameObject in the scene. Adjusted the Position of the "skinnysally" GameObject to (0, 0, 0) under Transform in the Inspector window.
  b. Selected skinnysally.fbx under "skinnysally" folder and switch its animations type to "Humanoid" under the "Rig" tab.
  c. Under "Avatar Definition", clicked on the "configure button".
  d. Dragged "Chest" node (GameObject) in the Hierarchy window into the slot named "Chest" under "Mapping" tab in the Inspector window(marked by blue box and arrow in Figure 8).
  e. Clicked on the "Head" tab under Mapping in the Inspector window, and clicked on the little circle on the right-hand side of "Jaw" slot and assigned it to "None". Clicked "Apply" button, and then "Done" button at the bottom.

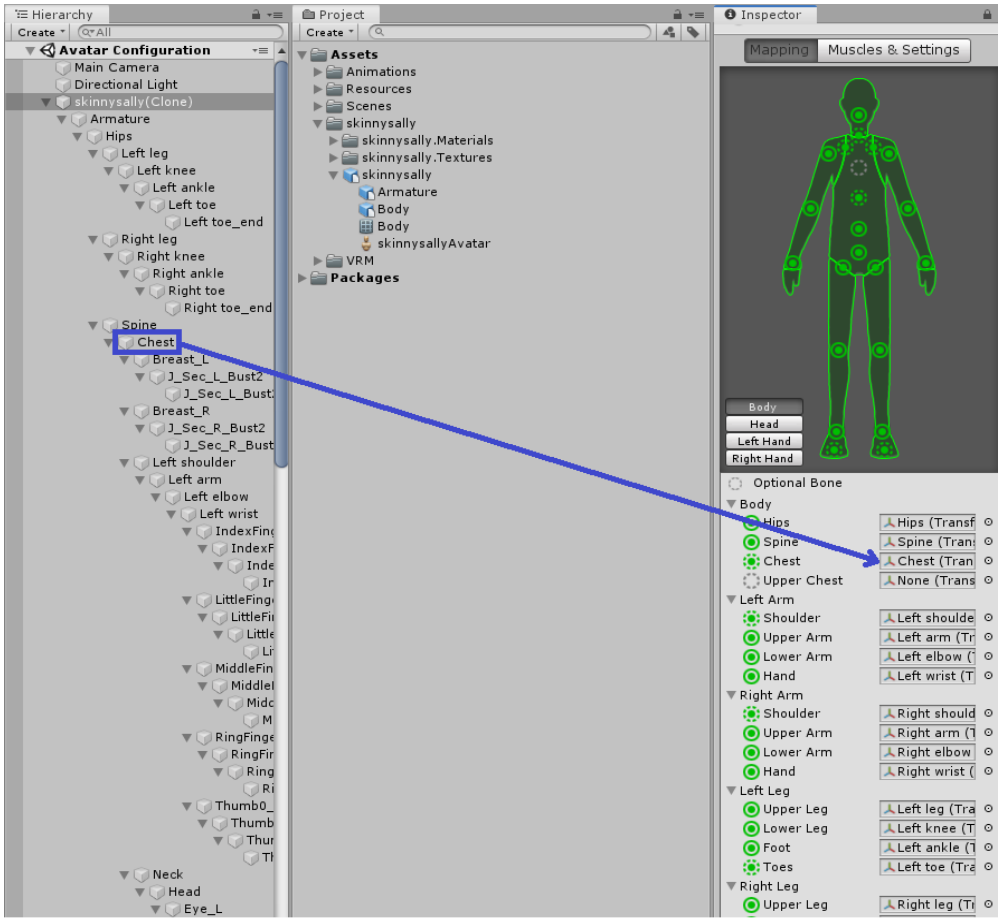

Figure 8. Reconfiguring Chest and Jaw node in skinnysally.fbx model

VI. Animator Controller Set-up

- *Process*
  a. Right-clicked on the "Animations" folder, created "Animation Controller" and named it "player", and switched to the "Animator" window by clicking "Animator" tab by the "Scene" tab in the left upper corner of window viewport in the Unity Editor.
  b. Dragged "unitychan_WAIT00" animation under "Asset/animation" folder to the "Base Layer" under animator window. Clicked on the "WAIT00" animation node and changed the name into "Idle" in the Inspector window.
  c. Right-Clicked on the empty space in the base layer and created a blend tree node and renamed it "IdleMoveBlend".
  d. Right-Clicked on the "Idle" node and made a transition targeting to "IdleMoveBlend", and also made a transition from "IdleMoveBlend" to "Idle".
  e. Clicked on the one of the transition arrow. Unchecked "Has exit time" and set "Transition Duration" to 0.1. Did the same thing for the other transition.
  f. Clicked on the "Parameters" tab in the animator window, and clicked on the "+" sign, added bool parameter and named it "isMoving". Renamed the float parameter "Blend" into "speedPercent".
  g. Clicked on the transition arrow from "Idle" to "IdleMoveBlend" and under the condition tab at the very bottom of the Inspector window, clicked on the "+" sign and set the condition

name to "isMoving" and value to "true" using the pull-down menu. Did the same thing for the other transition arrow except setting the condition value of "isMoving" to "false". After all these set-ups the Animator and Inspector windows appeared to be as in Figure 9.

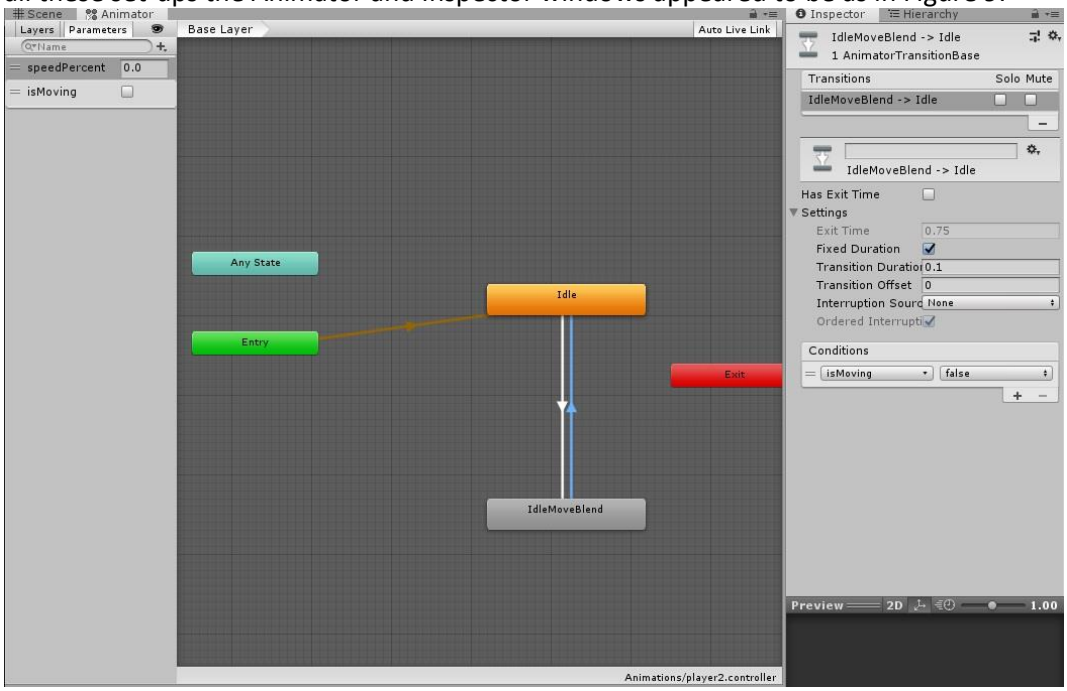

Figure 9. Base Layer of "Player" Animator Controller Set-up

h. Double-clicked on the "IdleMoveBlend" node. Right-clicked on the "Blend Tree" node and added 3 motions. Under "Motion" tab of the Inspector window, assigned 3 empty slots to "WAIT00", "WALK00_F" and "RUN00_F" from the top to bottom. Changed "Animation Speed" (by the Threshold" to 1.2 for "WALK00_F" and "RUN00_F". After this set-up, animator and Inspector windows appeared to be as in Figure 10.

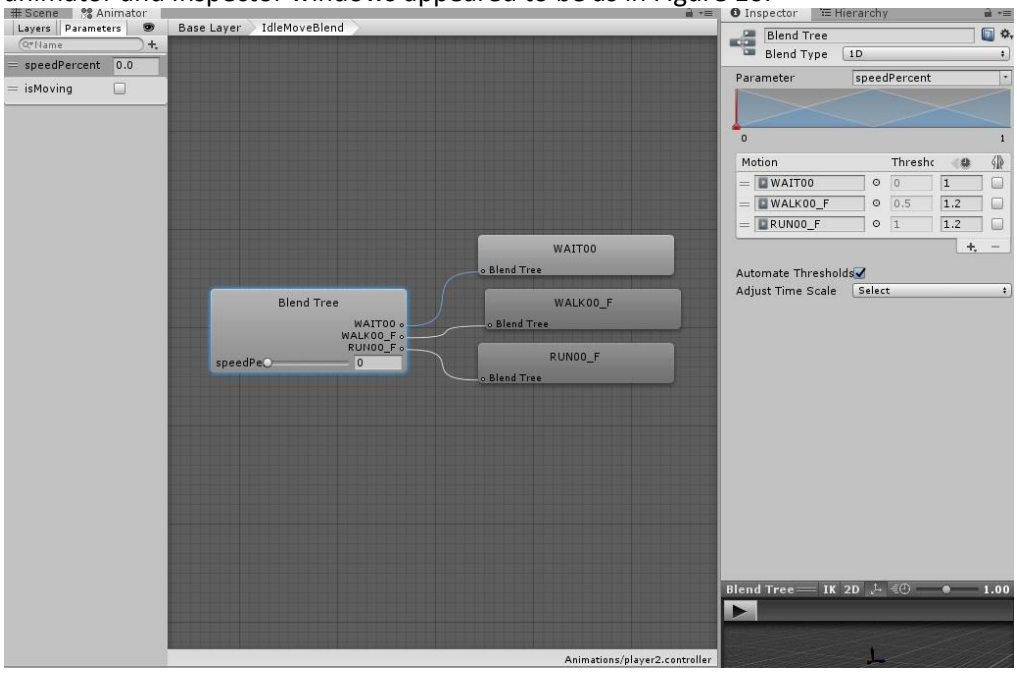

Figure 10. Blend Tree layer set-up.

VII. Handwave Animation Controller Layer Set-up

- *Process*
  a. Created an "Avatar Mask" under "animations" folder and renamed it "handwaveavatar".
  b. Selected "handwaveavatar" in the project window and expand the "Humanoid" tab in the Inspector window.
  c. Clicked on the both legs, root (the ellipse the humanoid is standing on) and "IK" on both sides of the legs to deactivated them in the mask (selected in red in Figure 11).

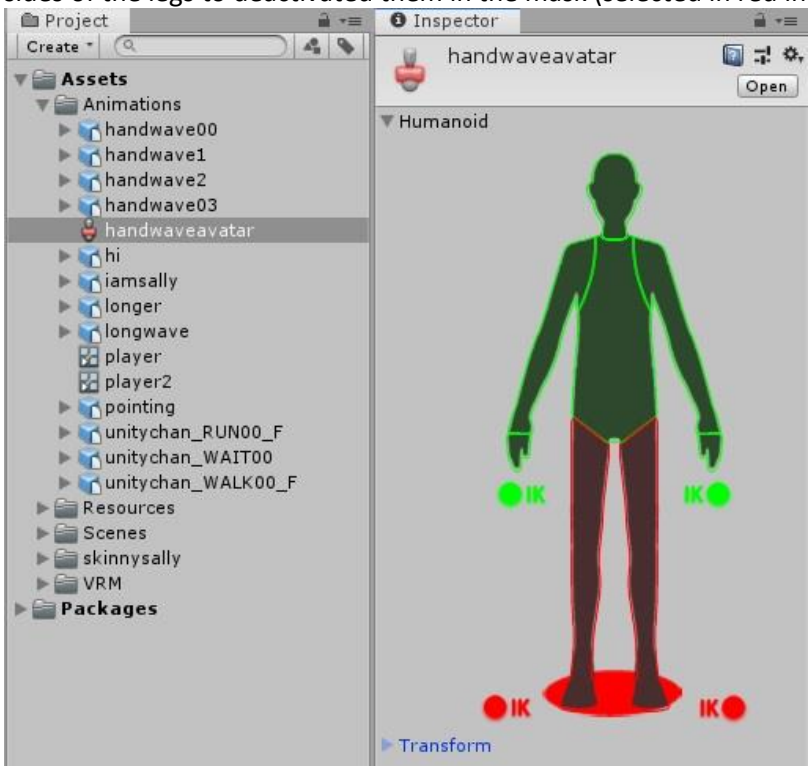

Figure 11. Avatar Mask Set-up

  d. Created a new layer under the Animator window by clicked in "+" sign under the layer tab and named it "handwave". Clicked on the little gear symbol by the name of the layer and assigned the "handwavemask" avatar mask to the "Mask" slot. Created a new state and leave it empty (named "New State" by default).
  e. Dragged all the animation fbx files from the "animations" folder into the handwave layer except "unitychan_WAIT00", "unitychan_WALK00_F" and "unitychan_RUN00_F". Created the mutual transitions between "New State" to all the other animation nodes except "hi" and "iamsally" animation nodes. Unchecked "Has exit time" for all the transition arrows from "New state" to all the animation nodes. For "hi" and "iamsally nodes, created a transition from "New State" to "hi, "hi to "iamsally" and "iamsally" to "New State". Unchecked "has exit time" from "New State" to "hi" node. Under parameter tab, add "Trigger" parameters each of which has a name of each animation node. After these set-ups, the Animator and Inspector windows appeared to be as in Figure 12.

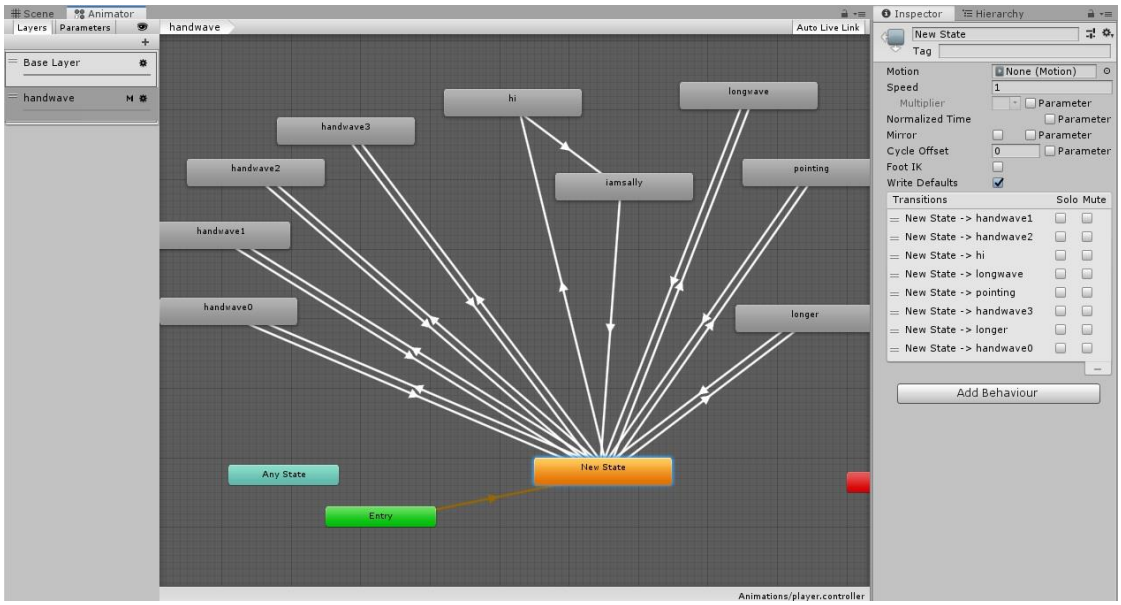

Figure 12. Handwave Layer Set-up

2. Customized Script: Implementation of the Scene Flow

The structure and interaction between game objects and scripts are summarized in Figure 13.

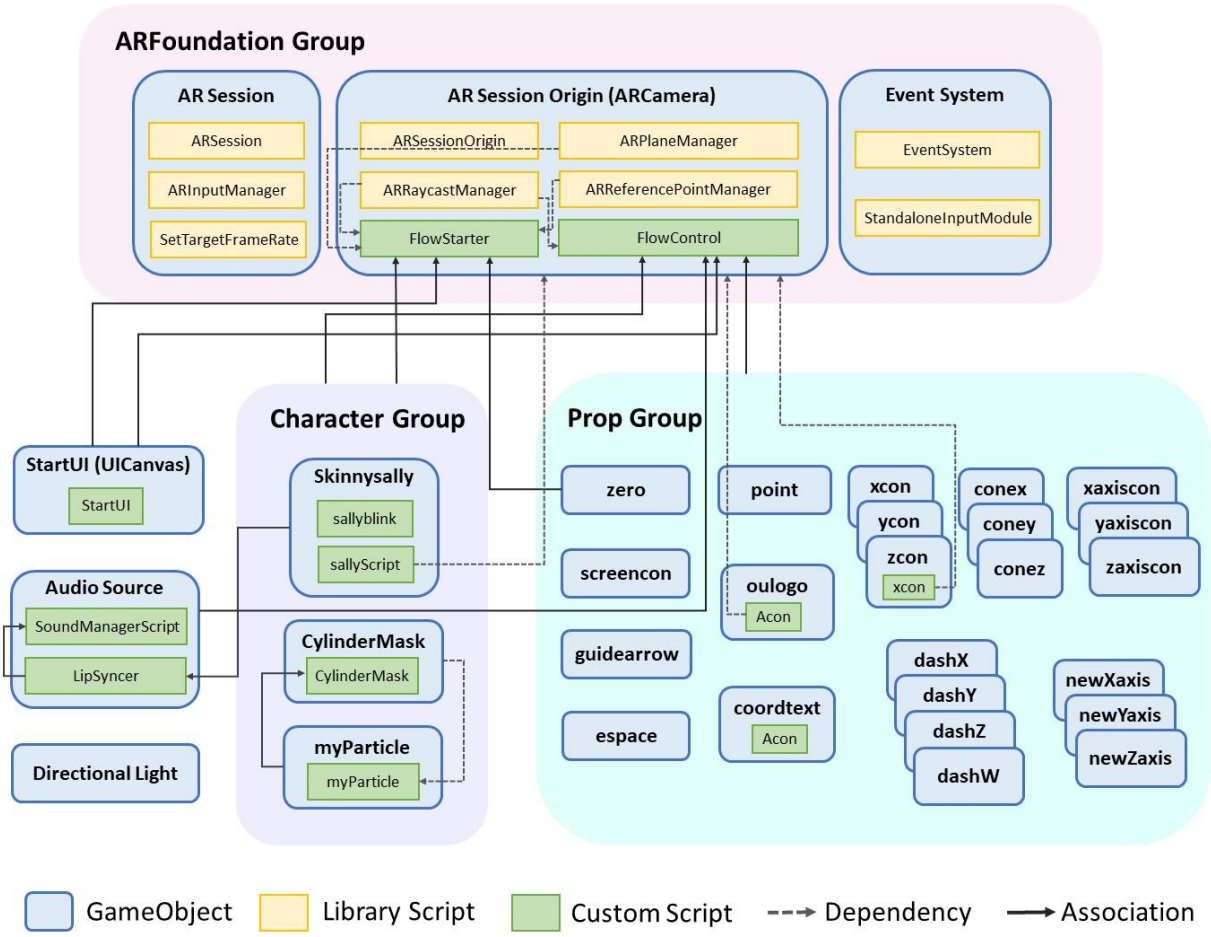

Figure 13. Object/Class Diagram of the Unity project

I. ARFoudation Group

- *FlowStarter*
  - Dependency – ARPlaneManager, ARRaycastManager, ARReferencePointManager
  - Association – Objects from the Character Group, StartUI, zero
  - Function and Execution
    i. The class receives plane data(plane extent, boundaries, normal vector) from ARFoundation library and updates it throughout the life time of the object which spans from the beginning to the end of the app.
    ii. Once one of the detected planes is greater than the minimum extent, it switches the text and image in the StartUI, prompting for a tap action to place the Virtual TA character on the plane. The minimum extent of the plane is set to 0.25(25cm) by default and can be set in the Unity Editor.

iii. At the starting of the app, skinnysally character and all the prop objects are load to the scene. However, the size of each objects are set to 0, so they are invisible. After the tap action, it sets the position and rotation of the skinnysally object so that it stands on the tap point with its up-vector aligned with the normal vector of the plane, and also resizes the skinnysally object to its default size 0.25. After that, it triggers the CylinderMask to be lifted up, so that the character get gradually visible and CylinderMask triggers particle effect once it reaches its highest elevation point.
iv. Creates a ARRefrenecePoint(equivalency of an anchor in ARCore), and set zero object and objects in Character Group as its children so that these object are under consistent tracking. Inclusion of the zero object under the ARReferencePoint is crucial because all the objects in the Prop Group will be set as children of the ARReferencePoint through the zero object instead of sending ARReferencePoint itself to FlowControl class.
v. Deactivates the visualization of all the planes through ARPlaneManager. When a new plane is detected during the life time of the app, receives a "planesChanged" event from ARPlaneManager and deactivates the newly detected plane.
vi. Deactivates StartUI upon the tap input.
vii. Sends the TrackableID and normal vector of the plane that was tapped on which is titled "mainPlane". The normal vector of mainPlane is utilized to reset the rotation of the skinnysally object as it changes its position in the scene.
    - In ARkit, this normal vector always seem to be aligned with Vector3.up, (0, 1, 0), hence "Straight-up" in the Unity's world coordinate. However, ARcore's horizontal planes can be slightly slanted, in case of which the normal vector of the plane is not straight-up. Therefore, sending the normal vector of the mainPlane to the FlowControl class is crucial for the Android devices.

- *FlowControl*
    - Dependency - ARPlaneManager
    - Association - Objects from the Character Group, Objects from the Prop Group, StartUI, AudioSource
    - Function and Execution
        i. The class is activated by FlowStarter class with a boolean value, "flowStarted", after the tap action and character placement/resizing takes place.
        ii. In each cycle, calls playScene function with current scene number until it reaches the total scene number
        iii. The total scene number is defined by the number of the cases in the switch statement of the playScene function. At the end of each scene, increments the current scene number so that it will execute the following scene which is the contents of the next case in the switch statement.
        iv. Depending on the scene, the end point of the scene is decided by either user input or termination of playing audio or animation files. During the scenes in which the dialog is played first and an user input is prompted, the transition between these two is recognized by the termination audio file. For that reason, "Audio Source" object is associated with this class to detect the end point of audio files even though methods in SoundManagerScript can be called statically.
        v. When an user input is required to proceed the scene execution, reactivated StartUI with a corresponding input action necessary for the scene explained in the UI's text.

vi. Once the origin is decided by user input, the position or zero is moved accordingly, and all the objects in the Prop Group are set to be the children of the parent of zero, which is the ARReferencePoint. This way, even when the tracking gets unstable and the position of ARReferencePoint is shifts, all the objects shown in the scene are moved by the same distance and direction retaining the relative positions to each other.

vii. In the last scene, tapping on the Open University logo will trigger playing final dialog and animation, including CylinderMask covering the character and the particle effect. 5 seconds after, the application opens the default browser of the device and opens the source material page. At this point, the app still runs at the background because Apple App Connect tends to disapprove any app that self-terminates during the execution of the app.

II. Character Group

- *SallyScript*
  - Dependency – ARCamera object in ARSessionOrigin
  - Association – None
  - Function and Execution
    i. The position of the skinnysally character is decided by the setDes function utilized by FlowStarter and Flowcontrol classes.
    ii. Idle animation is set as a default animation being constantly played. When setDes function is called IdleMoveBlend animation is played until the character reaches the point assigned by the first argument and rotate itself toward the second argument as it arrives to the destination.
    iii. During Idle animation, various hand-wave animations can be played by FlowControl class. The hand-wave animation are only applied to the nodes above the "Hip" bone, therefore, the legs and root node doesn't change from the Idle animation.
    iv. LookAtCamera function controls the rotation of neck bone according to the target object assigned by the argument, which is usually the ARCamera object throughout the app. The horizontal and vertical angle of rotation is calculated and applied separately when rotating the neck bone. The horizontal and vertical maximum rotation angle is set at 40 and 25 degrees, respectively, which can be set in the Unity Editor.

- *Sallyblink*
  - Dependency/Association - None
  - Function and Execution
    i. In a coroutine, a random float is generated and blinking routine is triggered when the value of this float is greater than the threhold set by "noblinkChance". Coroutine runs once every few seconds set by "blinkInterval". During the blink routine, coroutine's random float generation is devativated by a boolean value "isBlink".
    ii. The blink routine consists of one cycle of closing the eyelid, retaining the closure for the period set by "timeWithClosed" and opening the eyelid.
    iii. Skinnysally model has only one mesh named "Body" without having separate eyelid mesh or other facial mesh. However, the eye closure mesh formation is baked into the mesh as a blend key. Therefore, the percentage of the eye closure is calculated by the

time that passes from the blink trigger point over the time it takes to close the eye completely, set by "blinkTime", and eye closure blend key value is mapped by this percentage value.

iv. Once the time passed from the blink trigger point reaches "blinkTime" + "timeWithClosed", closing function gets deactivated and opening function gets activated by boolean value "isEyeOpen". When opening, the pecentage of eye closure is calculate by "(blinkTime – timePassed)" over "blinkTime".

v. Once a cycle of blink routine is accomplished, random float generation from coroutine is reactivated by "isBlink".

- *CylinderMask*
  - Dependency – None
  - Association – myParticle
  - Function and Execution
    i. CylinderMask object functions as a render mask for skinnysally character. The shader assigned to CylinderMask object has a Stencil reference number of 1 and this reference number is also added to the shader of skinnysally's material, Mtoon. When the pixels belonging to skinnysally are preceding in the ray toward the camera, Unity rendering engine skips the rendering of these pixels. Therefore, when skinnysally object is inside the CylinderMask object, skinnysally doesn't get rendered as if it as an "invisibility cloak".
    ii. FlowStarter and FlowContol control the visibility of skinnysally by changing the Y value of CylinderMask object's position. The direction of this changed is decided by the boolean arguments of activate function.
    iii. The distance of CylinderMask's movement is calculated by the distance between the pivot points of skinnysally and the CylinderMask. Skinnysally has the pivot point at the very bottom of the object while CylinderMask has it at the center. Therefore, this offset had to be compensated when calculating the distance of the travel.

- *MyParticle*
  - Dependency - CylinderMask
  - Association - None
  - Function and Execution
    i. The position of the particle is reset to the bottom of the CylinderMask once it is activated by FlowStarter or FlowControl
    ii. While CylinderMask is in move, the parameters of myParticle is maintained with its starting values, which forms a flat ring like shape by confining the particles' location and travel speed. Once CylinderMask reaches its destination, it triggers the changes in parameters values of myParticle making particles to spawn randomly in a hemi-sphere shaped zone and increases the travel speed of them. This creates the dispersion effect.

III. Prop Group

- *Acon and xcon*
  - Dependency – ARcamera object in ARSessionOrigin
  - Association – None
  - Function and Execution
    i. Acon and xcon essentially has the same function with which the object face toward the ARcamera, which is the user's device. This is implemented to provide the user with the better viewing angle toward the labels because by nature of AR, the user can move the device around the objects to have a better perspective.
    ii. The difference between Acon and xcon is the "sign" of the direction vector objects' forward vector is aligning with. While the objects with xcon look toward the ARCamera, Acon rotates the objects so that they look away from the device.
    iii. The labels (numbers) for the grid and x,y,z axes are generated with Gimp, and some label are rendered as an inverted image of what the letter is intended to be. This problem was found after all the materials and texture assets were generated, so rather that re-generating all of them, Acon.cs was written to replace xcon for those objects.

- *Other Prop Objects*
  - Dependency and Association - None
  - Function
    i. All other objects without any script attached serve as static props and their transforms(position, rotation, localsize) are controlled by FlowControl in the scenes that requires these prop.
    ii. They are loaded (instantiated and activated) in the scene when starting the app, however the size of them are set to (0, 0, 0), so they are rendered in the scene. FlowControl adjusts their transforms according to the contents of the scene and the user input.
    iii. Other static props, such as new axes and grid, grid labels are not loaded in the scene, but are set up as prefabs. These were instantiated by FlowControl in the scene in which the Euclidean space with the real numbers are set.
    iv. Espace is an empty object that serves as a container under which new points and labels can be nested as children. In that way, clearing up the Euclidean space (destroying the point and label objects) can be implemented conveniently.

IV. Audio Source

- *SoundManagerScript*
  - Dependency – None
  - Association – LipSyncer
  - Function and Execution
    i. This class functions in a singleton-like manner where it has a static method that plays audio assets in any other class without association. FlowStarter and FlowControl utilize it whenever the scene requires a sound effect or the character's narration.

ii. At the start of the app, it pre-load all the audio and lip sync assets to the memory. When an audio asset gets triggered to be played, it also sends the corresponding lips sync asset (JSON) to the LipSyncer class.

- *LipSyncer*
  - Dependency - None
  - Association - "Body" child object of the skinnysally
  - Function and Execution
    i. Receives JSON text objects from SoundManagerScript and serialized then into C# objects.
    ii. Has to serializable classes, Sequence and Frame. Sequence class has the Entries and Length property. Length stores the duration of the whole dialog being played in seconds, and Entries stores the list of Frames that include viseme information of every 0.05 seconds.
    iii. Frame has 4 properties generated by Oculus Lip Sync library, of which only one is utilized, Visemes. Visemes is array of 15 float values that signifies the blend key value of skinnysally mesh. According to these values at the correct time line, setVisemeToMorphTarget method change the each blendshape value in the "Body" child object of skinnysally.
    iv. Each index of Visemes signifies how close the 0.05 second-length of the sound wave in the audio asset is to actual pronunciation of 14 different vowels and consonants. Each blendshape in the skinnysally has to be manually assigned to these indices in the Unity Editor under the LipSyncer script component of skinnysally. The matching pronunciation of Visemes indices are silent, pp, ff, th, dd, kk, ch, ss, nn, rr, aa, E, ih, oh and ou.
    v. When no audio asset is being played, or when the value of the first Visemes index(silent) is 1, the skinnysally mesh is set to morphed into "Smiling" blend shape with the weight of 0.6 (60%), which can be reset in the Unity Editor.

V. StartUI

- *StartUI*
  - Dependency – None
  - Association – FlowStarter, FlowControl
  - Function and Execution
    i. This class includes methods that changes the raw image and text being displayed on the children object, UICanvas.
    ii. At the launching of the app, it starts to display the "scanning with the phone motion" animation with the corresponding message until the decent size of plane is detected. When it happens, FlowStarter switches the animation to "Tap Image" and with corresponding text. After the tap input, FlowStarter deactivates this object, so the whole UICanvas disappear.
    iii. When an user input is required in the scenes, FlowControl reactivates UICanvas with a new instruction text, but deactivates the raw image, so only the text message is displayed at the bottom of the canvas.

Deployment

The app is available for download in both Google Play and Apple App Store and searchable using the keyword, "euclidean ar" or "commonwealth of learning". Unity packages files and the source code can be found on the Github page of the project.

Google Play : https://play.google.com/store/apps/details?id=org.col.threeDEuclideanSpaceAR
Apple App Store : https://apps.apple.com/ca/app/3d-euclidean-space-ar/id1474420796
Github : https://github.com/COL-inno/3D-Eucliden-Space-AR

1. Google Play

I. Player Setting in Unity

- Set Company Name, Product Name and Version as "COL", "3DEuclideanSpaceAR", and correct version, respectively.
- Clicked default Icon and choose the icon for the App. The size has to be 1024X1024 for the Apple App store when building for iOS (All the smaller-size icons are automatically generated).
- Under Graphic APIs, removed "Vulkan" by selecting it and clicking "-" sign.
- Under Identification, set the Package Name as "org.col.threeDEuclideanSpaceAR". This is a unique identifier for each app, so even when uploading new versions to Google Play, this name has to be the same.
- Typed the correct version and Bundle Version Code. In each Google Play upload, either the Version or Bundle Version Code has to be different. Switched the minimum lever to "Android 8.0 'Oreo'".
- Under Configuration, switched "Scripting Backend" to "IL2CPP" and checked the "ARMv7" and "ARM64" boxes.
- Under Publishing Setting, clicked on the "Keystore Manager" and created a new keystore with a new key values which has an alias, "col_km_assistant". This keystore information is store in user.keystore file.

II. Build Setting

- Selected "Android" under platform and clicked "Switch platform" button.
- Clicked on the "Add Open Scenes" button under Scenes in Build.
- Checked the "Build App Bundle (Google Play)" box. This generates AAB package instead of APK, which is better for the optimization at Google Play. Click on the "Build" button.
    - Unity 2019.1.6f build has a bug where ARCamera doesn't track the objects in the app generated as AAB package while the APK package performs fine. According to Unity, it's a bug in organizing folders when compiling the ARFoundation library. Since this issue has been fixed in the newer versions, Unity 2019.2.0f build was used for compiling/building for this app.

III. Google Play Console

- On https://play.google.com/apps/publish/, signed in with the Android developer account.
- Clicked on the "Create Application" and typed the title "3D Euclidean Space AR".
- Filled up the Short and Full description of the app. The keyword "Commonwealth of Learning (COL)" was added in the Full description section for a few times because Google Play is known to generate app's metadata including the search keywords based on the repetitive words in the description.
- Uploaded the screenshots and icons. Resized the icon into 512X512. 32-bit PNG requirement doesn't have to be met as long as resolution is correct.
- Resized one of the screenshot into 1024X550 to meet the requirement and upload it for "Feature Graphic".
- Set the Application type and Category as "Game" and "Education", respectively. Filled in the website, email and privacy policy with those of the organization, Commonwealth of Learning. Saved the draft.
- Clicked on the "App Release" section of the dashboard menu on the left-hand side of the page, and clicked on the "Manage" link under "Production". Clicked on the "Create Release" button.
- Clicked on the "Continue" button under "App signing by Google Play" section, and upload the AAB package file. Google Play Console analyzes and generated optimized version of the package at this stage.
- Filled the "What's new in this release" with summarized description of the app. Also added the keyword "Commonwealth of Learning (COL)" in case it is utilized in generating app's metadata. Clicked on the "Save" button.
- Moved on to the "Content Rating" in the dashboard. Clicked "Continue", filled the email. Selected "REFERENCE, NEWS OR EDUCATIONAL" under category. Selected "No" for all the section in the questionnaire. Saved and clicked the "calculate rating" and "apply rating".
- Moved on the Pricing & distribution section. Set the price "Free" and checked "Available" circle under "Countries". This makes the app available in all the listed countries in Google Play. Select "No" for both "Primarily Child Directed" and "Contains ads". Checked the boxes for the mandatory consents, "Content guidelines" and "US export laws". Saved drafts.
- Moved back on to the "App releases" on the dashboard. Clicked on the "Edit release". Clicked on the "Review" at the very bottom of the page. If some of the requirements of the app is not met, the console will give you the error. Fixed all the issued according to the error message and clicked on the "Start Rollout" button.
- Under "Dashborad" section on the dashboard, it showed the app's status and "Pending Production". The app become available and searchable from Google Play after a day or two.

IV. Updating the Release

- When the change was made in the app, the app has to be re-built with either different version or different Bundle Version Code while keeping the same package name.
- After re-building AAB package in Unity, clicked the "App releases" section under "Release Management" of the dashboard on Google Play Console. Click "Manage" under Production and click "Create New Release".
- Uploaded new version of AAB package. The old version of package is automatically set to be discarded.
- Clicked on the "Review" and "Start Rollout".

- The new release is in effect after about a day or two.

2. Apple App Connect

I. Creating iPhone Developer Certificate on Mac

- Launch Keychain Access. Select "Request a Certificate from a Certificate Authority" under KeyChain Access -> Certificate Assistant.
- Fill the email and common name(organization name) and checked the circle for "Save to disk" and clicked on the continue. Selected an easily reachable directory for save the file in. The CSR file is created and saved.
- On https://developer.apple.com , clicked on "Account" menu on top and signed in with the Apple Developer Account of the organization. Clicked on the "Certificate, IDs & Profiles" on the dashboard.
- Clicked the "+" by "Certificates and selected "iOS App Development". Clicked "Continue".
- Uploaded the CSR file created in the Keychain Access and download the certificate file. Opened the downloaded certificate file and it was automatically registered in the Keychain Access.

II. Creating App Identifier

- In the "Certificates and selected "iOS App Development" of Apple Developer Account, clicked on the "Identifiers" on the dashboard menu.
- Clicked on the "+" sign by the Identifier. Filled the App ID as "3D Euclidean Space AR" and filled the Bundle ID with the same name used as Package name of the Unity's Player setting. (For iOS app, "com.col.3DEuclideanSpace" was used for this section by mistake). Clicked Continue.

III. Creating Provisioning Profile

- In the "Certificates and selected "iOS App Development" of Apple Developer Account, clicked on the "Profiles" on the dashboard menu.
- Clicked on the "+" sign by the Profiles. Checked the circles for "iOS App Development" and "App Store" and clicked the continue button. Selected the app id for "3DEuclideanSpace" and clicked on the continue. Checked the circles for the certificate for iOS distribution that had been created earlier. Typed the distribution name and clicked continue. Downloaded the profile file and double-clicked it to register. After this stage, there appeared two certificates in the Keychain Access, one for iOS development and the other for iOS distribution. These share the same "Organization Unit" number.

IV. Player Setting in Unity

- In Unity's player setting, set Company Name, Product Name and Version same as those from the Android platform if they are different. Upload the icon with the resolution of 1024X1024. This must be 1024X1024, otherwise Apple App Connect won't accept it.
- Filled the Bundle Identifier with the same string used for App ID, "com.col.3DEuclideanSpace". Filled the version and build number. When uploading new build for the same version, the build number has to be different.

- Typed "11.0" for "Target minimum iOS version", checked the box for "Requires ARKit support" and switched Architecture to "ARM64".

V. Build Setting

- Clicked on the "iOS" under platform in Build Setting. Clicked on the "Switch Platform" button. Click on the "Build
  button and selected the directory the inside the current project directory.

VI. Creating an App in App Store Connect

- Signed in with the Apple Developer Account of the organization on https://appstoreconnect.apple.com.
- Clicked on the "My Apps" and created a new app by clicking the "+" sign. Filled the name of the app and selected Bundle ID (same as App ID), and typed the SKU "COL0001" and continued.
- Filled the Name, "3D Euclidean Space AR" and Privacy Policy URL from the organization's webpage. Selected "Education", "Game", "Puzzle and "Board" under Category.
- Set the price as "CAD 0 (Free)" under "Pricing and Availability". Checked on the circles for "All territories selected" under "Availability".
- Moved on to the IOS APP section of dashboard. Uploaded screenshots of each device platform. The screenshots were generated from the screen captures from the app playthrough and resized in the correct resolution for each devices. Also, preview video was made and uploaded by capturing playthrough as a video and editing it.
- Filled the other information such as description and contact information under "General App Information" and "App Review Information", and saved.

VII. Uploading a build through Xcode

- Loaded the xcode project of the app by double-clicking on the xcode project file that was generated by Unity after build.
- Selected the "Unity-iPhone" with the icon under "TARGETS" menu. Clicked on the "General" tab, and fill all 4 boxes under Identity with the information that was set for the Player Setting in Unity.
- Checked the box by the Automatically manage signing and selected the organization's name in the "Team" section. iPhone Developer name automatically showed up under "Signing Certificate".
- Archived by clicking Product -> Archive. Upon successfully finishing archiving, the Organizer window popped up automatically.
- Clicked on the "Distribute App" button. Click "Next" until "Upload" button shows up and clicked on the "Upload" button. When the app file didn’t meet the requirements of Apple App Connect, the error message popped with the reason. Fixed the issued and uploaded again with the new Archive.

VIII. Selecting Build and Submit for Review

- Moved to "IOS APP" section in My Apps of App Store Connect. When the "processing" status of newly uploaded archive is lifted (can be confirmed under "Activity" menu), add the build by clicking the "+" sign by the Build. Once adding the build, the icon was automatically added.

- Clicked Submitted for the Review. Selected "No" for all the rating questionnaires. Selected "No" for the encryption. Selected "Yes" for using Advertising Identifier(IDFA) and checked the first two boxes, "Serve advertisements within the app" and "Attribute this app installation to a previously served advertisement". This is due to the inclusion of "UnityAds" package.
- Once the review period is over, the app is available in the Apple App Store in about 24 hours.

IX. Updating the Release

- Under "App Store" section of My Apps, clicked on the "VERSION OR PLATFORM" under "IOS APP" and selected iOS.
- Typed the number of new version that is at least greater than the previous version.
- In Unity, built the new version with the version number chosen from App Store Connect. Also match this version number and build number in the Xcode before Archiving.
- After uploading the archive in a same way as before, added the new build to the new version and clicked on the Submit for the Review.
- Submitted the same answer for the encryption and Advertising Identifier. Once the review period was over, Apple App Connect showed the new version.

Lesson Learnt

1. Understanding and Learning Game/Physics Engine(Unity)

- The structure of scene, objects and components and how they interact
- The process of importing and exporting assets and libraries
- Transferring projects between different versions of editor
- Optimization of the scene structure
- Debugging using both console and graphic gizmos
- Structure and semantics of C# programming language
- Building Animation layers and masks
- Problem solving through the documentation and forum
- Compiling and building targeting various platforms

2. Understanding 3D Graphics and Learning 3D Graphic Software

- Concepts of vertex and mesh and how they are represented as data
- Structure of material, texture and UV and how they are linked
- Camera perspective and ray
- Concept of shader
- Basic modelling skills in Blender
- Concepts of bone structure, weight, blend keys in the 3D model
- Generating and modifying 3D animation using both timeline and coding
- Importing various types of model format and converting them into FBX format

3. Learning Concepts Used in Augmented Reality and Handheld AR Libraries

- Mechanism through which the device interprets its accurate coordinate in the real world(SLAM)
- Concept of point clouds, plane generation as mesh and ray-casting
- ARCore library for Unity and Instant Preview for real-time debugging
- ARFoundation library and replacing ARCore library with it
- Understanding that a same method call in ARFoundation can receive different return values depending on the underlying libraries (ARKit vs ARCore) and how to adjust them

4. Deployment of the App for Two Mobile Platforms

- Acquiring cooperate developer accounts for both Google Play and Apple App Connect
- Generating developer and distribution certificates for the Apple platform
- Generating and modifying icon, screenshots, preview according to the requirements for various size of the target devices
- Filling up rating, privacy and advertisement questionnaire accordingly
- Writing the description, keyword and tags effectively for generating optimal metadata for the app search

- Meeting the app requirement by replacing dynamic libraries with custom scripts for Apple App Connect
- Optimizing the app size by removing unnecessary assets and code for Google Play Console
- Updating the release after finding and fixing the bug
- Understanding figures in App Analytics and monitoring it daily

Future Plan

1. Polishing the 3D Model and Animations

- Current model seems to have a little glitch in the mesh and weight displaying separation between the vest and the shirts around the point the arms meets the body when swinging the arms in a certain way. This problem is inherent to all the character models generated using VRoid Studio software and has to be fixed after-the-fact in a 3D modeling software such as Maya or Blender.

- Oculus Lip Sync mapped 15 Viseme values to blend keys of the model. Currently, the model only has 7 corresponding blendshapes for the Viseme, so the missing Visemes are assigned to blendshape displaying facial mesh set for the vowel "I" sound. Blendshape can be add or modified in 3D modelling software, therefore matching the unassigned blendshape for other 8 Visemes will increase accuracy of the lip syncing.

- Currently, the character doesn't seem to face the camera directly when the character is playing the handwave animation while it does during the idle animation. This might be the rotation of the root bone or spine bone for each animation is off axis to that of idle animation.

2. Refactoring FlowControl.cs

- The FlowControl Class has a very long method where it implement all the logic in the scenes and separate then by switch cases. For this reason, adding a new scene between existing could be a bit cumbersome because the number of all the switch cases that follow this inserted scenes has to be incremented. The better design might be refactoring each switch case into separate "scene method", having an array of this scene methods and invoking these functions by incrementing the index of that array. In that way, a new scene can be simply added by inserting the names of method in the array and write the scene method. This also has a benefit when the exactly same scene is repeated in certain part of the script.

- The FlowControl class works as the whole scene and flow manager associating with the all the prop objects in the scene, and controls their transform when necessary according to the script or the user input. The reason this approach was taken is many of these actions happen just once throughout the whole scene. However, this might not be the best approach when an user input affects multiple objects to respond in the same way. In this case an event base system(Observer/Observed) will be more effective where FlowControl takes an input and notify all the prop objects subscribed to that action. This approach is better aligned with the principle of Object Oriented Programing, as well.

3. Implementing the Following Units From the Source Material

- The scripts and contents of this app are based on a unit in the multi-variable calculus series from the open source educational material. Now that the backbone of the scene control system, character and asset generation work flow are established, implementing other units will take a lot less time than it took for building this app. If the app is well received among the users and the potentiality of it is recognized, it will be exciting to see the whole chapter of the series implemented in this new platform of learning.