Dwight Look College of
ENGINEERING
TEXAS A&M UNIVERSITY

ECEN 403 Final Presentation

Hand Gesture Recognition

Team Members: Samuel Oncken, Steven Claypool
Sponsors: Stavros Kalafatis, Pranav Dhulipala

Problem Overview

Problem Statement

- Collecting large amounts of data for Gesture Recognition NN is **time consuming**, **resource intensive**, and **limited**

Goals

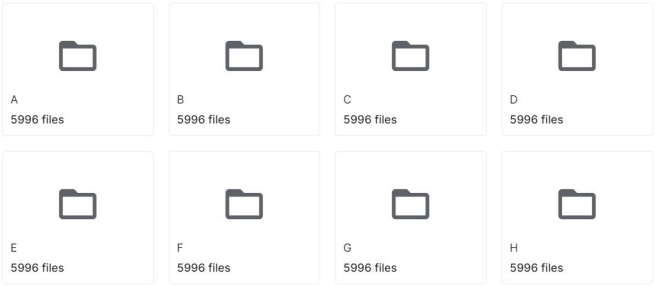
- Test viability of using **virtual data** to train a hand gesture recognition neural network
- Provide large amounts of **diverse data** (skin tones, hand sizes, accessories, etc.)
- Achieve **similar recognition accuracy** when tested against real, benchmark datasets

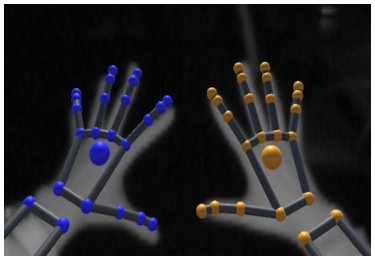
Why?

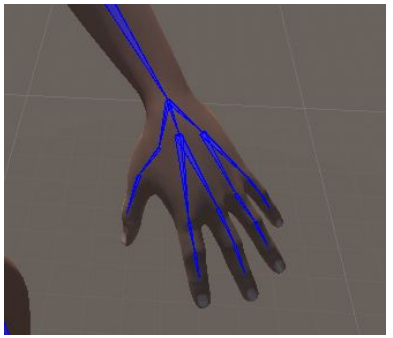
- Using a virtual environment, **scalability is easy** (# of images, # of participants, lighting conditions, background conditions, etc).

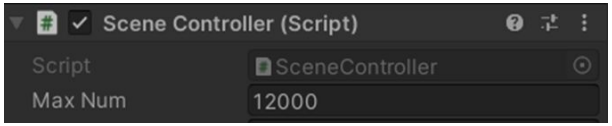
How?

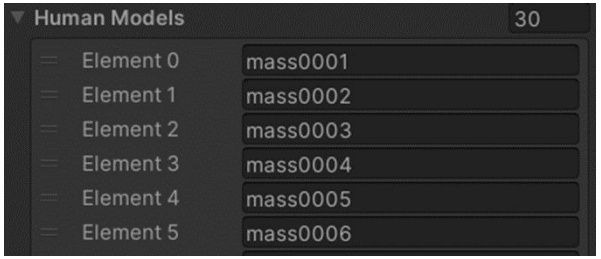
- **Replicate** multiple existing real datasets using the Leap Motion Controller (LMC) and Unity software
- **Test** hand gesture recognition neural network with purely real benchmark data, purely virtual data, and various compositions of each



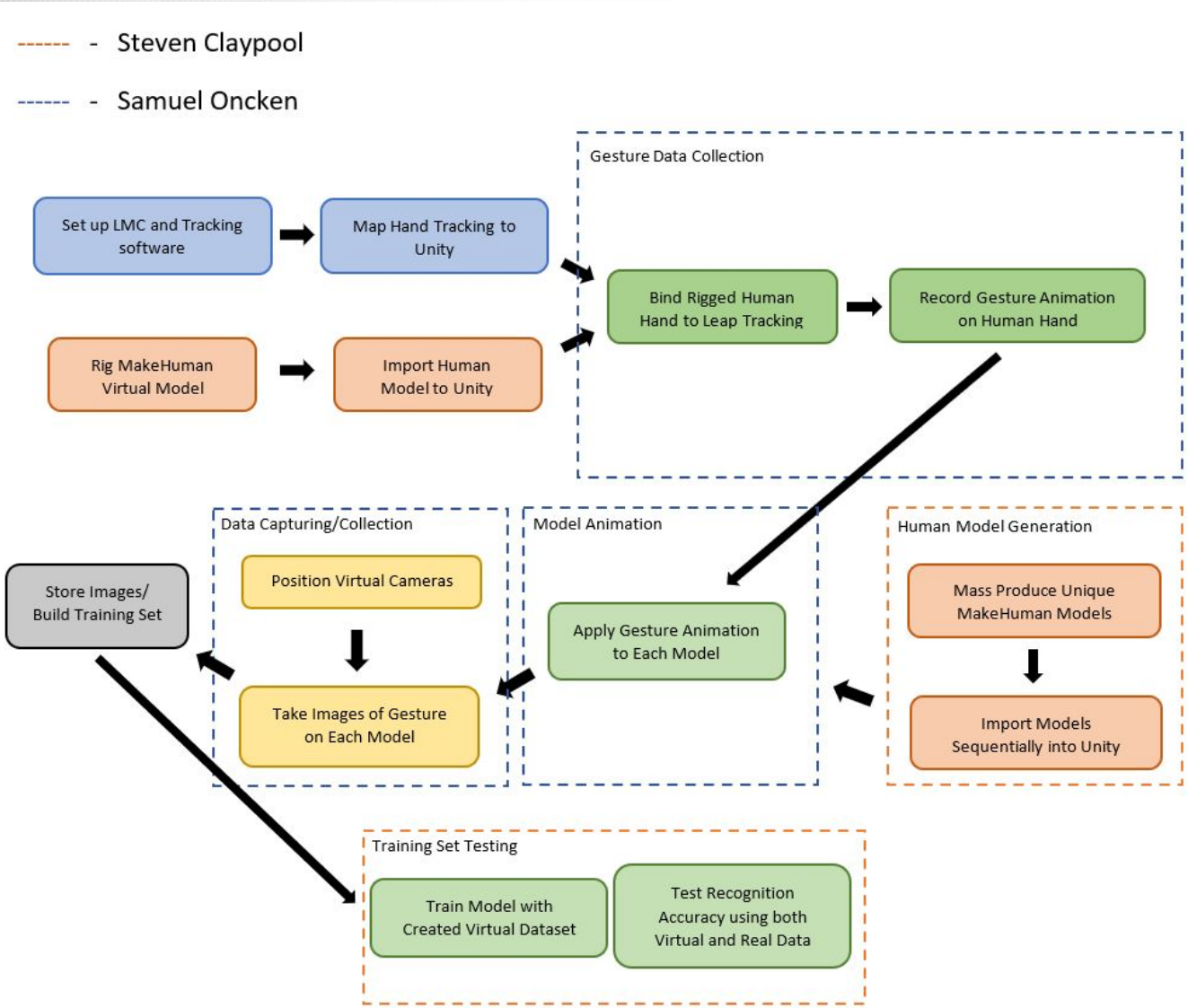








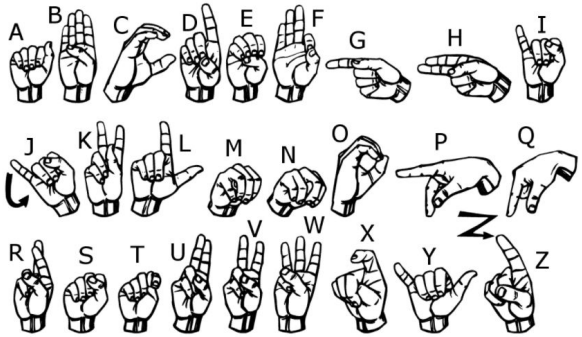
Subsystem Overview



Gesture Data Collection Subsystem - Complete

What has been accomplished:

- ASL alphabet replication
  - 28 animation clips
- ASL for digits replication
  - 10 animation clips
- HaGRID dataset replication
  - 18 animation clips

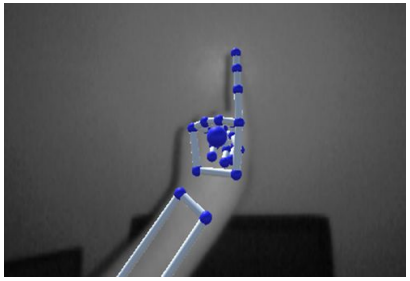


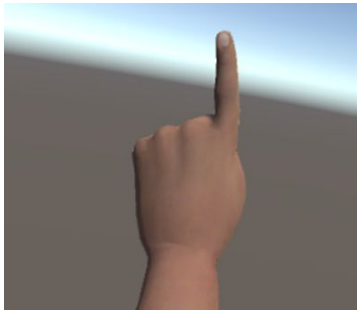
Adjustments since last presentation:

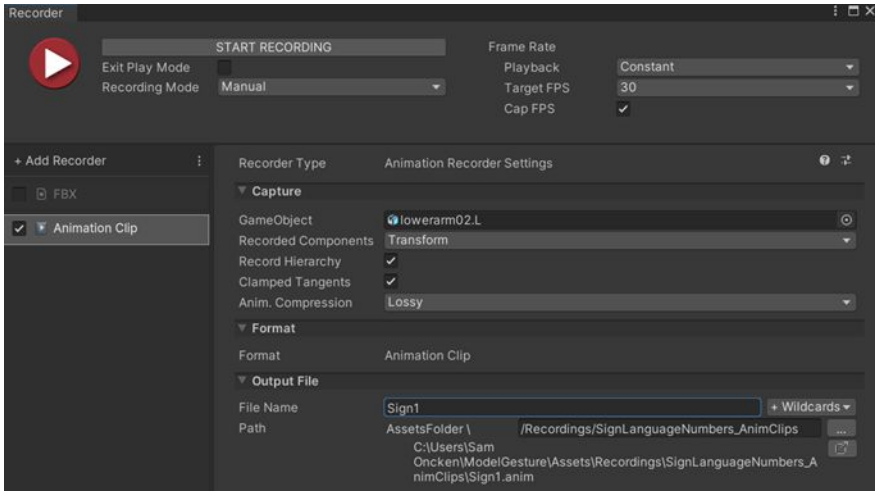
- Hand gesture is **recorded** and **exported** as an animation clip
  - Using Unity Recorder
  - Must set the lowerarm02 as root node of recording to record transformation data of child bones
- Recording solely with LMC as opposed to creating animation clips in Blender (for ease of use)

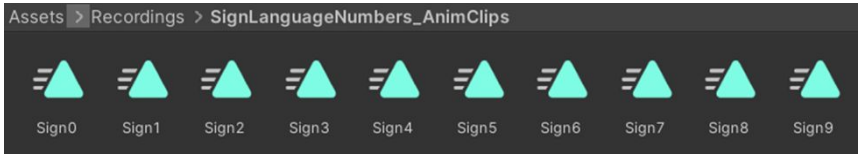
Gesture Data Collection Subsystem - Complete

- LMC is head mounted
- Animation clips are stored in dataset specific folder after recording









Human Model Generation Subsystem - Complete

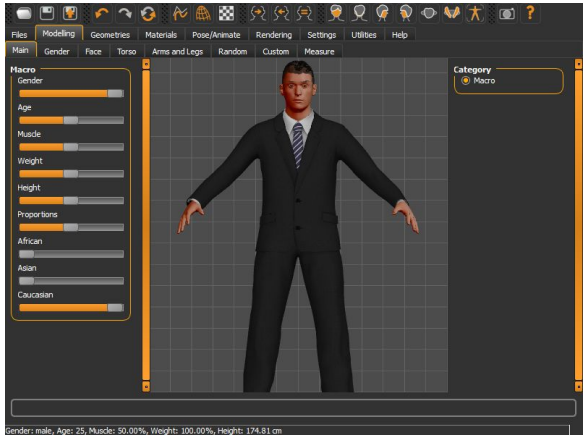
What has been accomplished:

- Models with a default rig can be exported from Makehuman
- Scripting to load the models into the scene and spawn a random model was created

```
//GetModels inputs all created MakeHuman models into an array for GenerateRandom to use
1 reference
void GetModels()
{
    DirectoryInfo dir = new DirectoryInfo("Assets/Resources/Models");   //selects directory to grab models from
    FileInfo[] files = dir.GetFiles("mass*.fbx");   //places all files with name starting with human and ending in .fbx
                                                    //from chosen directory into a files folder

    foreach (FileInfo file in files)
    {
        string name = file.Name.Split('.')[0];      //for each file, disconnect the .fbx from the name
        fileList.Add(name);                         //and add the model name into the list of file names
    }

    humanModels = fileList.ToArray();           //translates file of models into GameObject Array for use
}

//GenerateRandom is used to spawn a randomly selected human model into scene
1 reference
void GenerateRandom()
{
    if (spawned[0] != null)         //checks if there is already a spawned model in the scene
    {
        Destroy(spawned[0]);        //if there is, delete it before placing a new one
    }
    int HumanIndex = Random.Range(0, humanModels.Length);       //selects random index of human model array
    string filename = $"Models/{humanModels[HumanIndex]}";      //gets human model filename depending on randomly chosen index
    GameObject spawnedModel = Instantiate(Resources.Load<GameObject>(filename));    //places chosen model in scene
    spawned[0] = spawnedModel;      //indicates a new model is spawned
    PlayAnimation(spawnedModel);    //Calls PlayAnimation function
    //synth.OnSceneChange();
}
```

Changes from last presentation:

- Choosing export folder, adding handwear, and setting fixed height must be done manually
  - Deemed outside the scope of our project

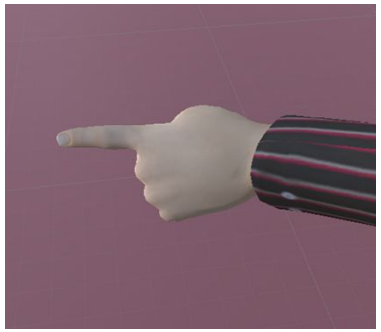


Model Animation Subsystem - Complete

What has been accomplished

- Placement of animation clips for each dataset into unique animation controllers

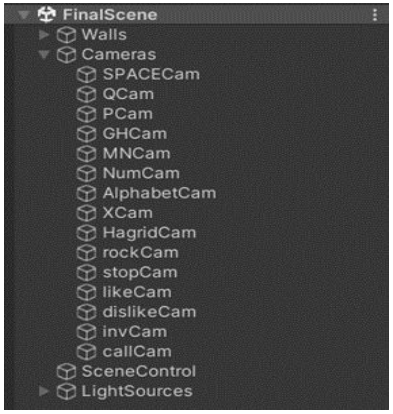
- Scripting accomplishments:
  - Placement of Animator component onto the lower arm of each spawned human model
  - Placement of correct animation controller onto Animator depending on dataset
  - Playing of random animation clip/gesture
  - Animation will no longer be selected to play after it has been the maximum user-selected number of times



```
//Indexes through the models bone hierarchy until the lowerarm02 bone is reached
GameObject rig = model.transform.Find("MakeHuman default skeleton").gameObject;
GameObject root = rig.transform.GetChild(0).gameObject;
GameObject spine5 = root.transform.GetChild(2).gameObject;
GameObject spine4 = spine5.transform.GetChild(0).gameObject;
GameObject spine3 = spine4.transform.GetChild(0).gameObject;
GameObject spine2 = spine3.transform.GetChild(0).gameObject;
GameObject spine1 = spine2.transform.GetChild(2).gameObject;
GameObject clavicle = spine1.transform.GetChild(0).gameObject;
GameObject shoulder = clavicle.transform.GetChild(0).gameObject;
GameObject uparm1 = shoulder.transform.GetChild(0).gameObject;
GameObject uparm2 = uparm1.transform.GetChild(0).gameObject;
GameObject lowarm1 = uparm2.transform.GetChild(0).gameObject;
GameObject lowarm2 = lowarm1.transform.GetChild(0).gameObject;

//Places animator component onto the lowerarm02 bone
Animator animatorArm = lowarm2.AddComponent(typeof(Animator)) as Animator;
//Gets animator component for manipulation
animatorArm = lowarm2.GetComponent<Animator>();
animatorArm.enabled = true;             //enables the animator and applies root motion
animatorArm.applyRootMotion = true;
```

```
//This block is for playing ASL for Numbers animations since ChooseDataset is marked as 0
if (ChooseDataset == 0)
{
    //Places animation controller fit with ASL number animations into arm animator component
    animatorArm.runtimeAnimatorController = (RuntimeAnimatorController)AssetDatabase.LoadAssetAtPath("Assets/Resources/Controllers/DigitsData.controller",

    //chooses random value 0-10 for random determination of animation to play
    int randNum = Random.Range(0, 11);
    if (randNum < 10)                                   //if the number is less than 10,
    {
        animatorArm.Play("Sign" + randNum.ToString());      //the index means exactly the animation to play
    }
    else
    {
        Destroy(spawned[0]);     //if 10 is chosen, we want to show "nothing" so we delete the model before taking image
    }
```

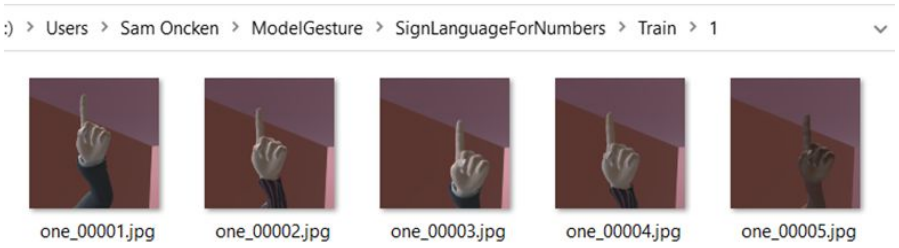
Data Capturing/Collection Subsystem - Complete



What has been accomplished

- Placement of virtual cameras in positions to accurately display gesture animation
- Scripting accomplishments
  - Depending on dataset and animation clip playing, correct camera is enabled
  - Slight randomization in camera location for diversity in data
  - Screenshot is captured after gesture animation is played and correct camera is enabled
  - Images are sorted by gesture in distinct dataset folders located within the Unity project
  - Images are named as they are in the real dataset we are replicating

```
IEnumerator TakeImage(float delayTime, char letter, int gestureNum, int dataset)
{
    int gesturesDone = 0;
    string folderPath;
    string filename = $"{gestureNum.ToString().PadLeft(5, '0')}.jpg";   //naming output file as dataset we are replicating has
    yield return new WaitForSeconds(delayTime);  //again delays so animation can play out before screen capture
    if (dataset == 1)
    {
        folderPath = Directory.GetCurrentDirectory() + $"/AmericanSignLanguage/Train";
        //these next if-else statements decide where to store the output images depending on the character being animated
        if (letter != 's' & letter != 'n')
        {
            folderPath = Directory.GetCurrentDirectory() + $"/AmericanSignLanguage/Train/{letter.ToString()}";
        }
        else if (letter == 's')
        {
            folderPath = Directory.GetCurrentDirectory() + $"/AmericanSignLanguage/Train/Space";
        }
        else
        {
            folderPath = Directory.GetCurrentDirectory() + $"/AmericanSignLanguage/Train/Nothing";
        }
```



Data Capturing/Collection Subsystem - Complete

- Scripting accomplishments cont.
  - System will stop running once the max number of images per gesture have been reached

```
//sifts through digitCounter to determine if all gestures are complete
foreach (int i in digitCounter)
{
    if (i >= maxNum)
    {
        gesturesDone += 1;
    }
}
//if all gestures are complete with maxNum pictures for each,
if (gesturesDone == 11)
{
    EditorApplication.isPlaying = false;        //stops Unity player
}
```

- As recommended by our project sponsor, we have also included a few extra customizations
  - Background image randomization
  - Lighting condition randomization

one_00002.jpg

one_00005.jpg

```
LightSource1.intensity = Random.Range(0f, .7f);
LightSource2.intensity = Random.Range(0f, .7f);
```

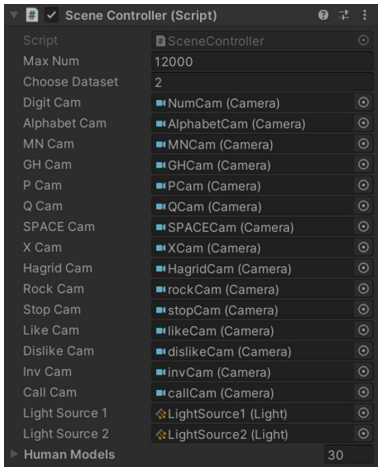
Entire Unity Environment Completion

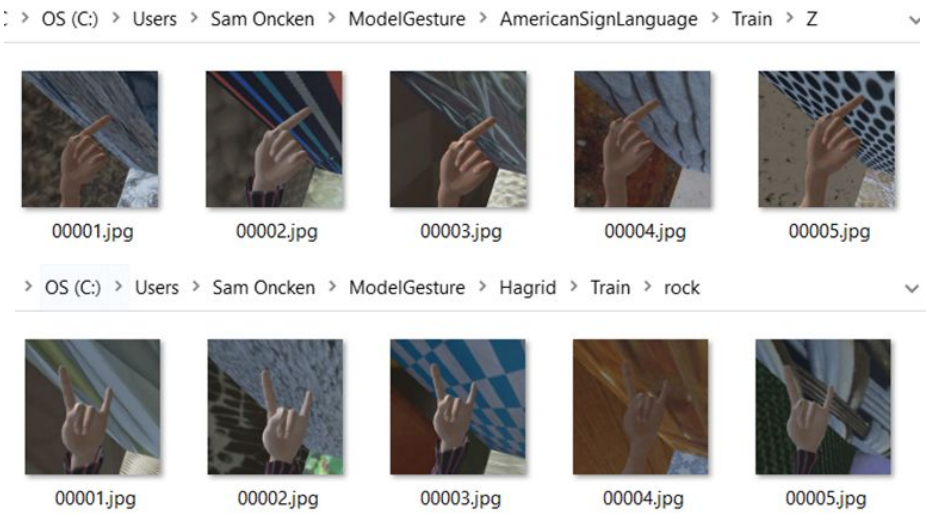
What has been accomplished

- All subsystems have been merged and are working together in a singular environment

Dataset Generation

- Created replicated dataset for Sign Language for Numbers - 12,000 images per gesture (132,000 images total)
- Created replicated dataset for American Sign Language - 12,000 images per gesture (336,000 images total)
- Will be running dataset replication of HaGRID this week with ~24,000 images per gesture and 120,000 “no gesture” images (553,000 images total)





Training Set Testing

What has been accomplished:

- Training, validation, and testing using synthetic data with >99% accuracy

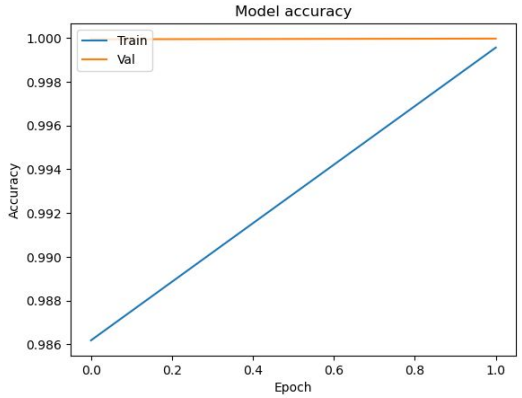
Changes from last presentation:

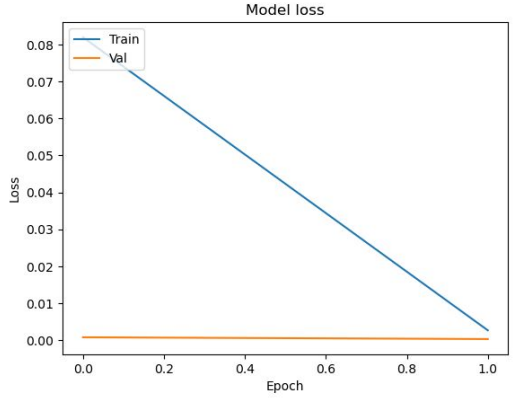
- Fine-tuning pretrained models (ResNet18)

Next steps:

- Work on synthetic data training and real data testing

```
Epoch 2/2
2062/2062 [==============================] - 531s 258ms/step - loss: 0.0026 - accuracy: 0.9996 - val_loss: 2.8501e-04 - val_acc
uracy: 1.0000
```






```
1031/1031 [==============================] - 264s 90ms/step - loss: 3.5477e-04 - accuracy: 1.0000
```

Execution Plan

	October 12th	October 26th	November 9th	November 23rd	November 30th	December/Winter Break
Gesture Data Collection	- Choose real datasets to replicate - Complete hand mapping configurations	- Complete gesture animation recording according to chosen datasets to replicate				
Human Model Generation	- Produce 6 models into Unity for animation testing	- Mass produce models into Unity environment sequentially				
Model Animation	- Apply recorded test gesture to a single model	- Randomly apply a recorded gesture to any model				
Data Capture/Collection	- Place at least 5 virtual cameras in Unity environment to face model hand from multiple angles	- Record and store images of gesture performed on any model				
Training Set Completion/ Testing		- Completion of training set creation system. - Testing process begins	- Preprocess each training set - Create new training sets ranging in composition of real and synthetic data	- Train each neural network with new training sets and record the metric used in the real dataset paper for proper comparison	- Evaluate results after comparison and prepare system and outcomes for final presentation.	- Continue training/testing for improved results when training on synthetic data and testing on real data - Run previous tests with other datasets (alphabet and HaGRID)

Validation Plan

Test Name	Success Criteria	Methodology	Status	Responsible Engineer(s)
Benchmark Dataset Training	Gesture recognition neural network can run on our home computer and train using the real dataset. Results quantified	Download the benchmark data set and the code for the CNN. Run the code and confirm similar accuracy to benchmark logs provided.	TESTED - Pass	Steven Claypool
Virtual Dataset Training	Gesture recognition neural network can train using our built dataset and provide accuracy results	Take a final virtual dataset modeled after a real benchmark dataset and use it to train the same CNN as the benchmark. Ensure similar accuracy results.	TESTED - Pass	Steven Claypool
Gesture Recognition Accuracy	Accuracy of gesture recognition is within 5% of benchmark accuracy using our virtual dataset	Train gesture recognition neural network using real and synthetic sets and compare accuracy	TESTED - Pass	Steven Claypool
Synthetic Data on Real Data Accuracy	Test real data with CNN trained on synthetic data and achieve accuracy similar to benchmark.	Train a CNN using different methods/compositions with synthetic and/or real data, and test using real data.	IN PROGRESS	Steven Claypool
Unity Hand Mapping	Real hand movement is mapped in Unity	Set up Unity, install Ultraleap plug-ins, map hand motion.	TESTED - Pass	Samuel Oncken
Import Rigged MakeHuman Model	A fully rigged MakeHuman model is imported into Unity	Import model into Unity and confirm appearance and functionality.	TESTED - Pass	Steven Claypool
Virtual Model Unity Hand Mapping	Map hand motion onto an imported MakeHuman model.	Use Hand Binder component/configure settings. Confirm natural motion.	TESTED - Pass	Samuel Oncken
Mounting Stability	Head mounted LMC remains in place during head motion	Mount LMC and plug the device into the computer. Rotate head in all directions and shake head left to right.	TESTED - Pass	Samuel Oncken
Apply Example Animation to Model	MakeHuman model is able to perform an imported full body gesture accurately.	Import an animation .fbx and apply the animation to the rigged human model. Confirm that motion is as expected.	TESTED - Pass	Samuel Oncken
Apply Recorded Gesture Animation to Model	Rigged MakeHuman model can perform a recorded gesture animation.	After recording an animation, apply it to an imported MakeHuman model using the Animator component.	TESTED - Pass	Samuel Oncken
Create and Import Rigged MakeHuman Models to Unity	Minimum 30 MakeHuman models can be generated and imported into Unity	Use MakeHuman “mass produce” function to generate unique character models, each fit with a “Default” rig, with 20% edge cases	TESTED - Pass	Steven Claypool
Data Capture Output and File Type	Virtual camera outputs image data as a .jpg files	Record images of gesture, validate that the data is stored, organized, and is of the desired file type.	TESTED - Pass	Samuel Oncken
Final System Validation	With the press of a button, a large, diverse virtual training set is produced	Run system and validate in output files that each gesture has at least 500 images of gesture performance on differing human models from numerous angles	TESTED - Pass	Samuel Oncken