Technology & Interaction

Grau en Enginyeria en

Disseny Industrial

Prof. Narcís Parés.

e-mail: narcis.pares@upf.edu

T2 – Connecting MediaPipe Pose to Unity

Tracking solution: Media Pipe

MediaPipe

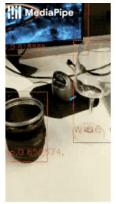
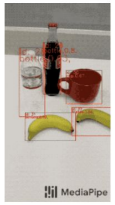
Live ML anywhere

MediaPipe offers cross-platform, customizable ML solutions for live and streaming media.

End-to-End acceleration: *Built-in fast ML inference and processing accelerated even on common hardware*	***Build once, deploy anywhere***: *Unified solution works across Android, iOS, desktop/cloud, web and IoT*
Ready-to-use solutions: *Cutting-edge ML solutions demonstrating full power of the framework*	***Free and open source***: *Framework and solutions both under Apache 2.0, fully extensible and customizable*

ML solutions in MediaPipe

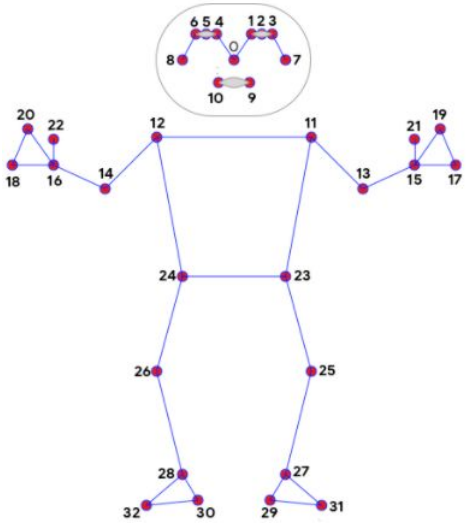
Face Detection	Face Mesh	Iris	Hands	Pose	Holistic
					

Hair Segmentation	Object Detection	Box Tracking	Instant Motion Tracking	Objectron	KNIFT
					

https://google.github.io/mediapipe/

Media Pipe Pose

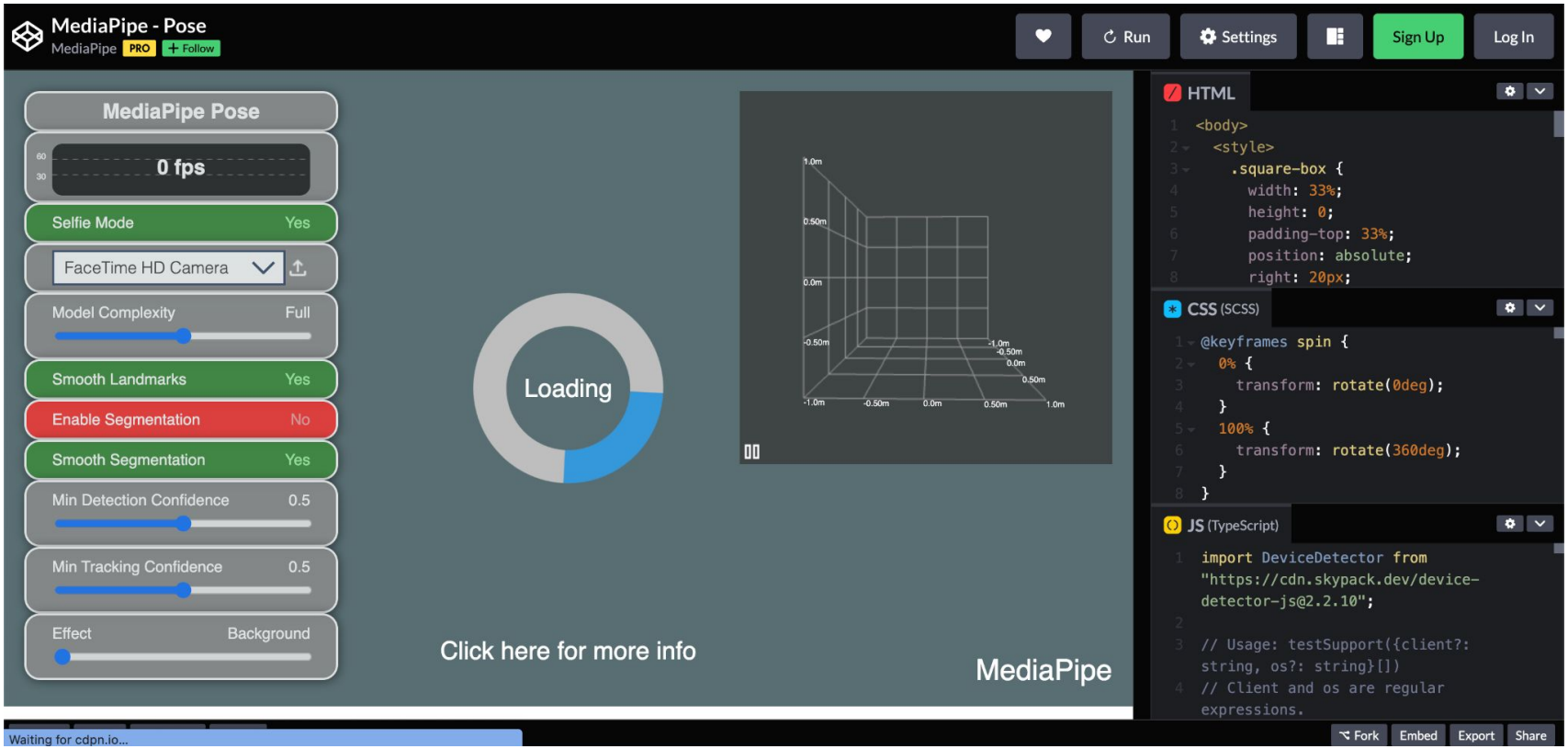
MediaPipe Pose is a ML solution for high-fidelity body pose tracking, inferring 33 3D landmarks and background segmentation mask on the whole body from RGB video frames



0. nose
1. left_eye_inner
2. left_eye
3. left_eye_outer
4. right_eye_inner
5. right_eye
6. right_eye_outer
7. left_ear
8. right_ear
9. mouth_left
10. mouth_right
11. left_shoulder
12. right_shoulder
13. left_elbow
14. right_elbow
15. left_wrist
16. right_wrist
17. left_pinky
18. right_pinky
19. left_index
20. right_index
21. left_thumb
22. right_thumb
23. left_hip
24. right_hip
25. left_knee
26. right_knee
27. left_ankle
28. right_ankle
29. left_heel
30. right_heel
31. left_foot_index
32. right_foot_index

https://codepen.io/mediapipe/pen/jOMbvxw

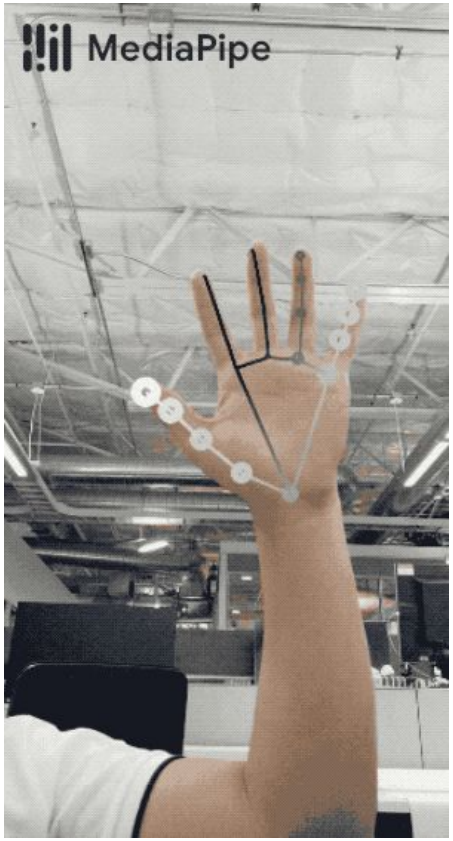
Media Pipe Pose: Live test

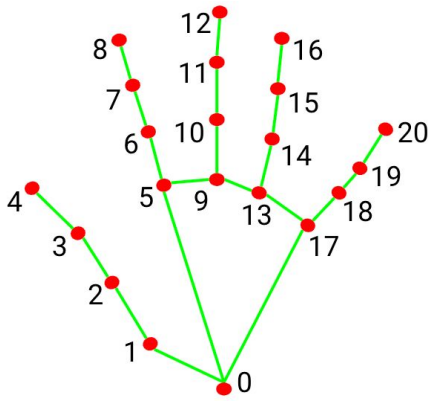


https://codepen.io/mediapipe/pen/jOMbvxw

Media Pipe Hands

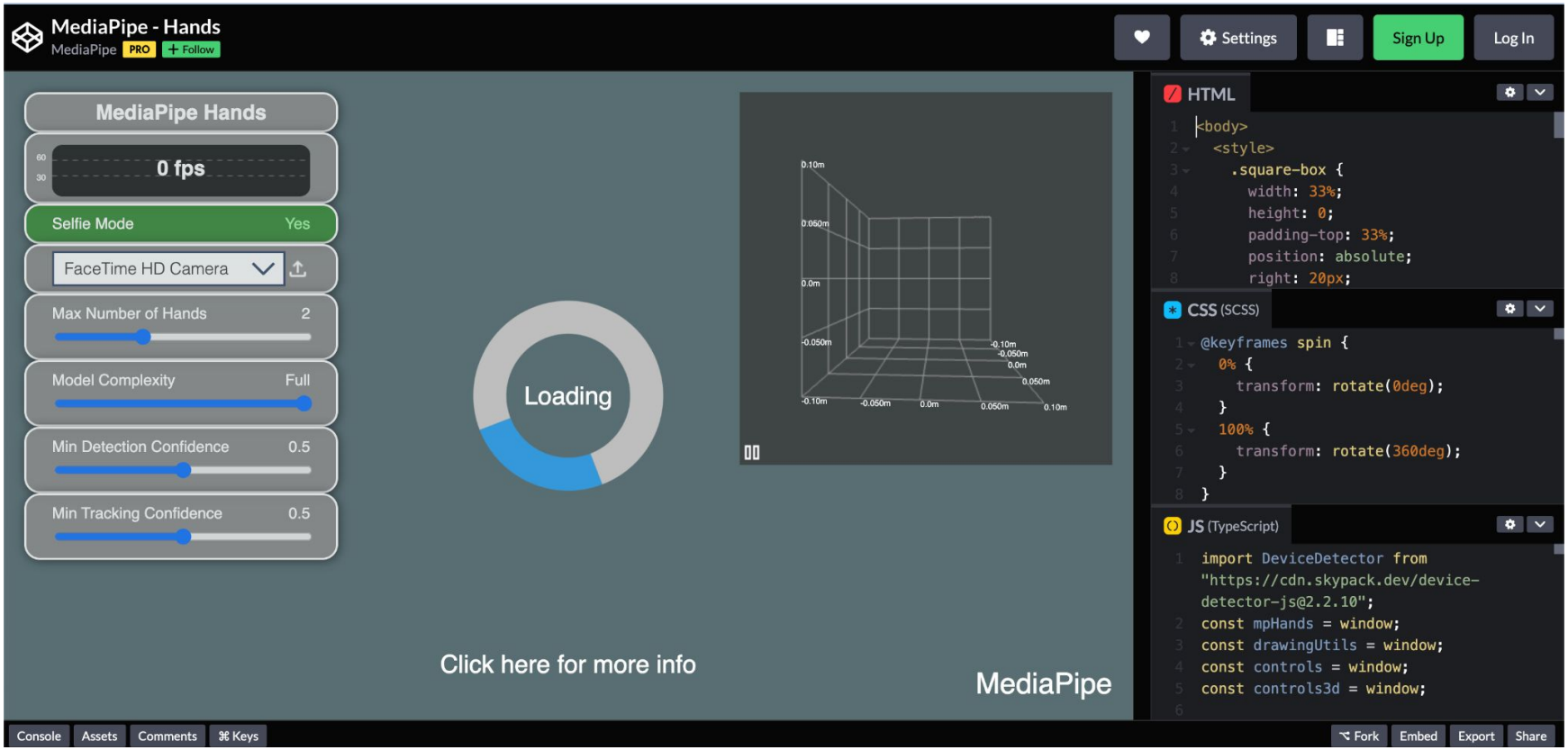
MediaPipe Hands is a high-fidelity hand and finger tracking solution. It employs machine learning (ML) to infer 21 3D landmarks of a hand from just a single frame.





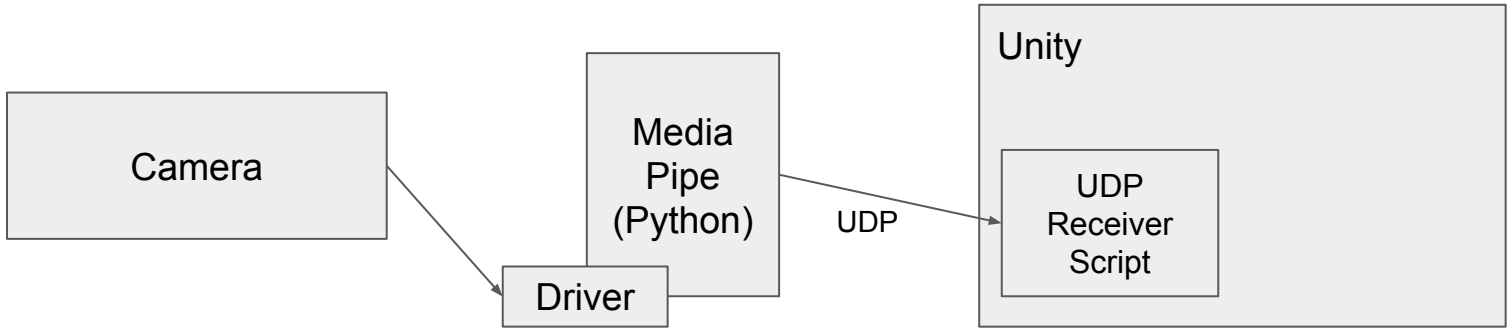
0. WRIST
1. THUMB_CMC
2. THUMB_MCP
3. THUMB_IP
4. THUMB_TIP
5. INDEX_FINGER_MCP
6. INDEX_FINGER_PIP
7. INDEX_FINGER_DIP
8. INDEX_FINGER_TIP
9. MIDDLE_FINGER_MCP
10. MIDDLE_FINGER_PIP
11. MIDDLE_FINGER_DIP
12. MIDDLE_FINGER_TIP
13. RING_FINGER_MCP
14. RING_FINGER_PIP
15. RING_FINGER_DIP
16. RING_FINGER_TIP
17. PINKY_MCP
18. PINKY_PIP
19. PINKY_DIP
20. PINKY_TIP

Media Pipe Hands: Live Test



https://codepen.io/mediapipe/pen/RwGWYJw

Connecting Media Pipe and Unity



- We need to run MediaPipe locally in our computers using our laptop camera.
- We will do it running a script in Python that is modified to send the tracking coordinates using the User Datagram Protocol (UDP) communication protocol.
- A script in Unity receives the coordinates through UDP.

Connecting Media Pipe and Unity

Prerequisite

- Check if we have python installed in our machine

Steps:

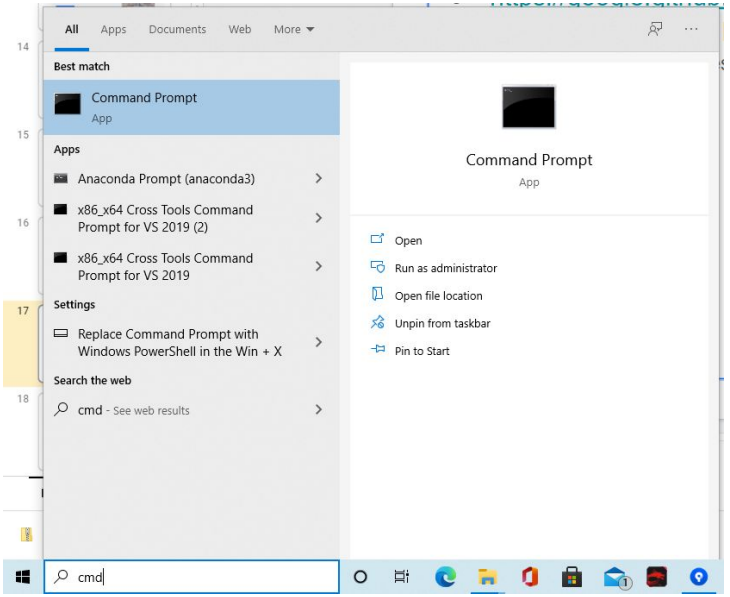
1. Get template project from Github

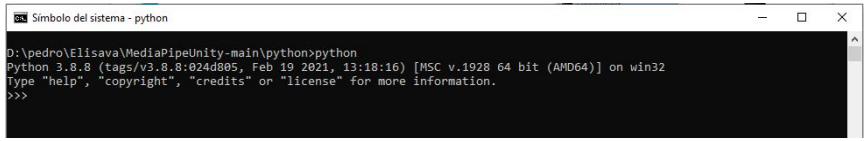
2. Install and run mediapipe in your computer (follow instructions according to your OS)
   a. Windows
   b. MacOs Intel
   c. MacOS M1

3. Open and run Unity project

Prerequisite: Python

- Check if you have python installed in your computer.
- To check if it is installed open a command line terminal (type cmd in search), once the terminal is open type “python”
- If it is installed, the version appears (we need version 3.6 or higher)
- to exit, write: quit()
- If it is not installed, download the installer from the python website



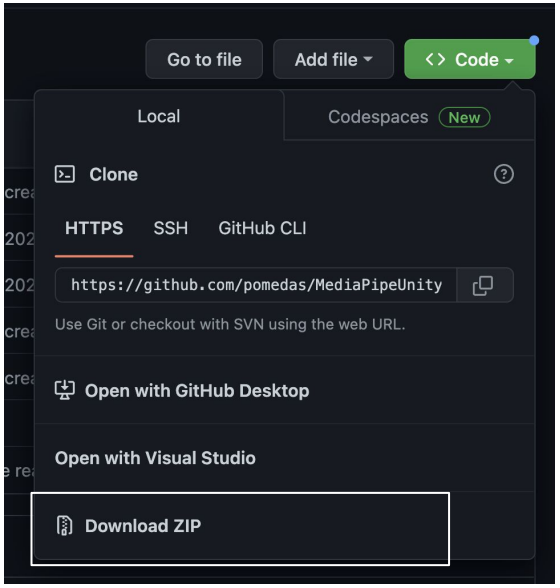


Python comes pre-installed on macOS

1. Get template project

A template project is prepared with modified python scripts that send the tracking coordinates using UDP protocol and a Unity project that receives the data

Get the template project from
https://github.com/pomedas/MediaPipeUnity
(click on Code -> Download Zip)



2a. Run mediapipe from script (Windows)

1. Open a command line
   (write cmd in the search box of windows)

2. Go to the python directory where you downloaded the template project.
   Type cd + route in the terminal and press enter:
   > *cd Downloads/MediaPipeUnity-main/python/*

3. Create and activate Python Virtual environment by typing this in the terminal:
   > *python -m venv mp_env*
   > *mp_env\Scripts\activate*

4. *(!) (This step only have to be done once, next time to run media pipe skip it)*
   Install Mediapipe dependencies using pip installer by typing:
   > *pip install mediapipe*

5. Execute MediaPipe Hands with the following commands.
   A new windows with the camera and the tracking points is displayed. Keep it open.

   > python hands.py

2b. Run mediapipe from script (MacOS Intel)

1. Open a terminal
   (Command + space and type terminal)

2. Go to the python directory where you downloaded the template project.
   Type cd + route in the terminal and press enter:
   > *cd Downloads/MediaPipeUnity-main/python/*

3. Create and activate Python Virtual environment by typing this in the terminal:
   > *python3 -m venv mp_env*
   > *source mp_env/bin/activate*

4. *(!) (This step only have to be done once, next time to run media pipe skip it)*
   Install Mediapipe dependencies using pip installer by typing:
   > *pip install mediapipe*

5. Execute MediaPipe Hands with the following commands.
   A new windows with the camera and the tracking points is displayed. Keep it open.

   > python3 hands.py

2c. Run mediapipe from script (MacOS M1)

1. Open a terminal
   (Command + space and type terminal)

2. Go to the python directory where you downloaded the template project.
   Type cd + route in the terminal and press enter:
   > *cd Downloads/MediaPipeUnity-main/python/*

3. Create and activate Python Virtual environment by typing this in the terminal:
   > *python3 -m venv mp_env*
   > *source mp_env/bin/activate*

4. *(!) (This step only has to be done once, next time to run media pipe skip it)*
   Make sure you have the latest version of pip and the latest version.
   > *python3 -m pip install --upgrade pip*

5. *(!) (This step only has to be done once, next time to run media pipe skip it)*
   Install requirements and the mediapipe package for M1 processor
   > pip install protobuf==3.20.1
   > pip install mediapipe-silicon

6. Execute MediaPipe Hands.
   A new windows with the camera and the tracking points is displayed. Keep it open.

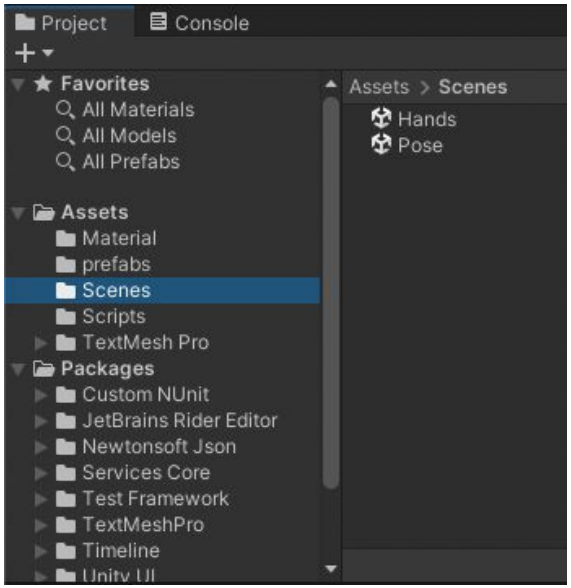
   > python3 hands.py

Explore the template project

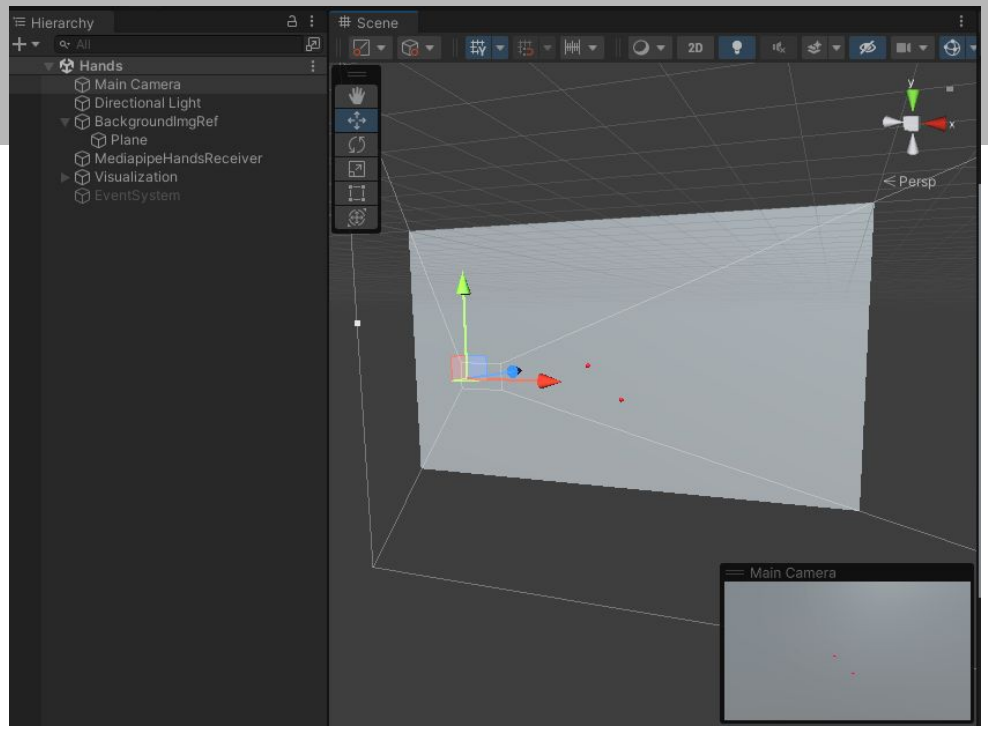
1. Open Unity Hub and click Open.
   Locate the folder of template project
   The project will open in the Unity Editor

2. Explore the project. It contains two scenes: Hands and Pose.
   Each one contains the basic functionality to use hands or full body movements.

3. Check Hands scene.
   On the Project Tab, go to Scenes and double-click on the scene Hands



Hands scene



Check in the hierarchy windows the objects

Main Camera: The main camera of the scene. Already positioned framing a plane that is used as reference to visualize the movements.

Directional Light: Basic light

BackgroundImgRef: Contains a plane that is used as reference of the limits of the movements (previously calibrated to a defined mapping). You can hide it or change the colour.

MediaPipeHandsReceiver: an empty object with the script MediaPipeHandsReceiver. In charge of receiving all the joint coordinates from mediapipe.

Visualization: Empty object that contains two empty objects with a script to visualize the hands movements in real time:

- **HandsJoints**: Visualize all the joints
- **HandCenter**: Visualize the center of the mass of all the hand joints

MediaPipe Hands Receiver

The GameObject **MediaPipeHandsReceiver** contains a script that is in charge of getting the coordinates from MediaPipe tracking (python code which must be running before).

<u>It doesn't need to be modified</u>. It receives the data from media pipe:

Collection of detected/tracked hands, where each hand is represented as a list of 21 hand landmarks and each landmark is composed of x, y and z.
x and y are normalized to [0.0, 1.0] by the image width and height respectively. z represents the landmark depth with the depth at the wrist being the origin, and the smaller the value the closer the landmark is to the camera. The magnitude of z uses roughly the same scale as x.

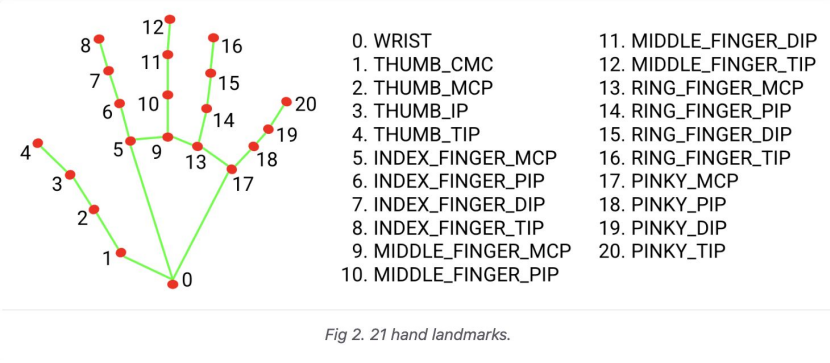
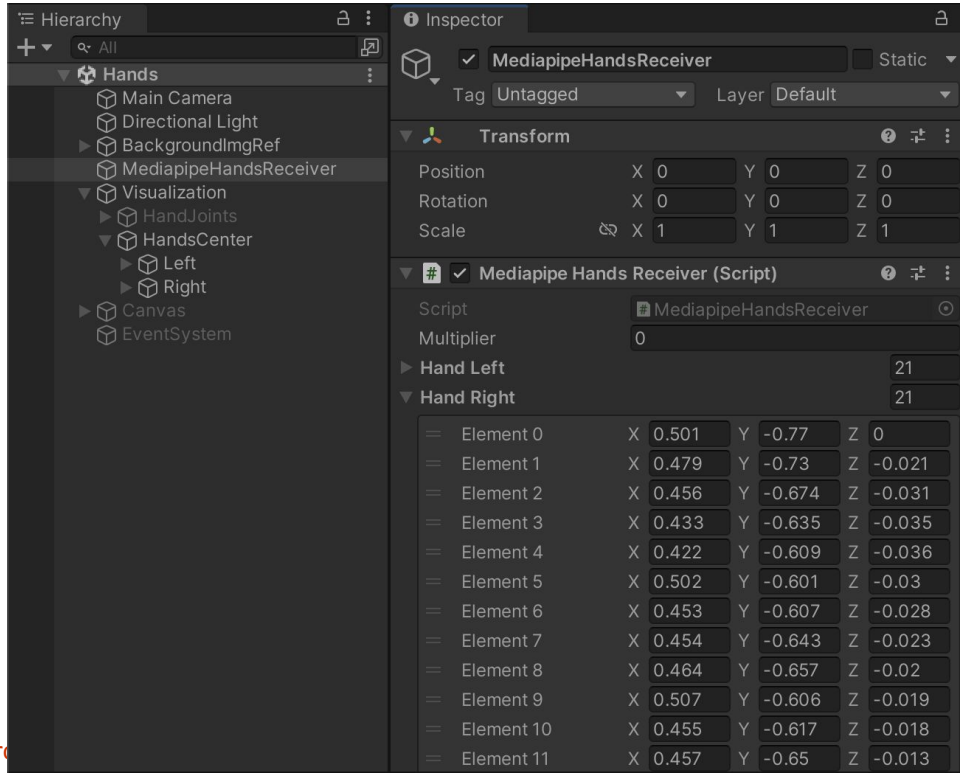


Fig 2. 21 hand landmarks.



https://google.github.io/mediapipe/solutions/hands.html

HandsCenter visualization

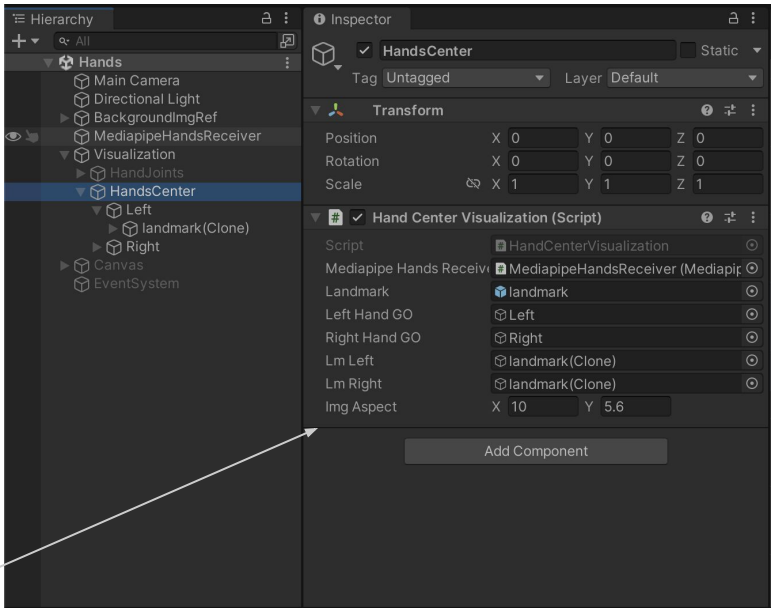
HandsCenter GameObject has a script that uses the information from **MediaPipeHandsReceiver**.

It creates two GameObjects in realtime (Landmarks, from a prefab) and update their positions every frame. The Landmarks will follow the left and right hands movements.

(!) Mapping: The MediaPipe coordinates are normalized (Between 0 and 1)
In this script we choose a mapping for the coordinates, using the values of the parameter ImgAspect .
This defines the amplitude of the movement.

Note that it maps x,y but **not** z. This maintains the movements in the plane.

The camera and Background Image Reference is set to this mapping, but this can be changed.




```
// Start is called before the first frame update
void Start()
{
    lmLeft = Instantiate(landmark, Vector3.zero, Quaternion.identity);
    lmLeft.transform.SetParent(leftHandGO.transform);

    lmRight = Instantiate(landmark, Vector3.zero, Quaternion.identity);
    lmRight.transform.SetParent(rightHandGO.transform);
}

// Update is called once per frame
void Update()
{
    Vector3 leftHand = Vector3.zero;
    Vector3 rightHand = Vector3.zero;

    for (int i = 0; i < 21; i++)
    {
        leftHand += new Vector3(mediapipeHandsReceiver.handLeft[i].x * imgAspect.x,
                                                                    mediapipeHandsReceiver.handLeft[i].y * imgAspect.y,
                                                                    mediapipeHandsReceiver.handLeft[i].z);


        rightHand += new Vector3(mediapipeHandsReceiver.handRight[i].x * imgAspect.x,
                                                                     mediapipeHandsReceiver.handRight[i].y * imgAspect.y,
                                                                     mediapipeHandsReceiver.handRight[i].z);

    }

    lmLeft.transform.position = leftHand / 21;
    lmRight.transform.position = rightHand / 21;
}
```

HandsJoints visualization

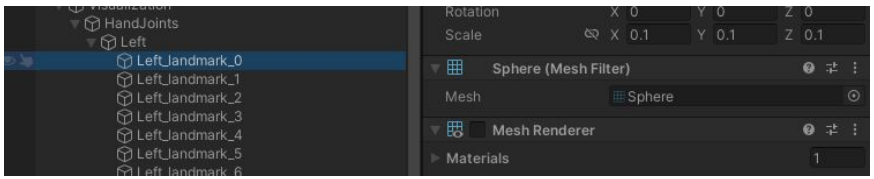
HandJoints GameObject has a script that uses the information from **MediaPipeHandsReceiver**. and shows all joints.

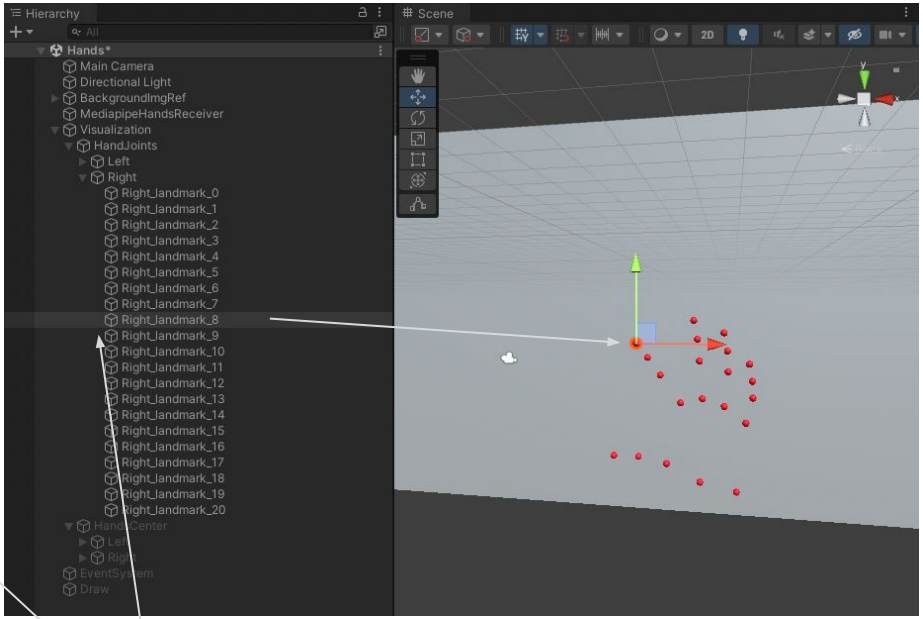
It has one GameObject per joint of each hand (Landmarks) and updates their positions every frame. The Landmarks will follow the left and right hand movements. The names correspond to the MediaPipe definition.

(!) Mapping: as in MediaHandsCenter, it defines a mapping.

The camera and Background Image Reference is set to this mapping, but this can be changed.

You can hide the red spheres that represent the landmarks by unchecking the Mesh Renderer property in the inspector





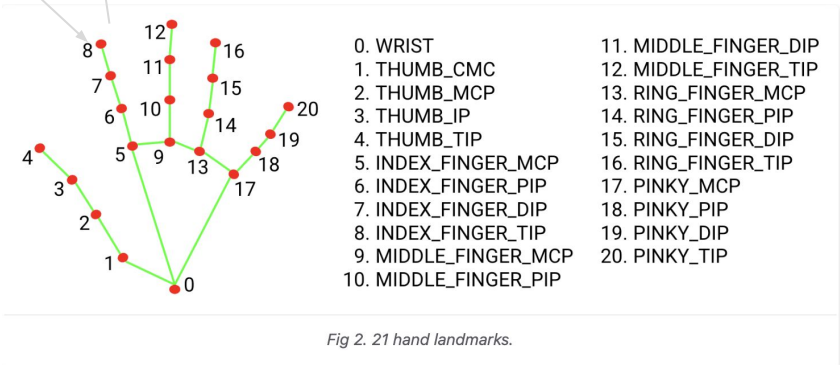


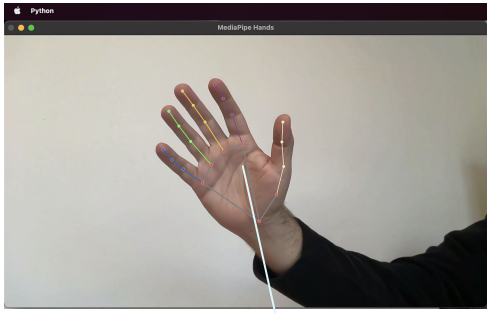
Fig 2. 21 hand landmarks.

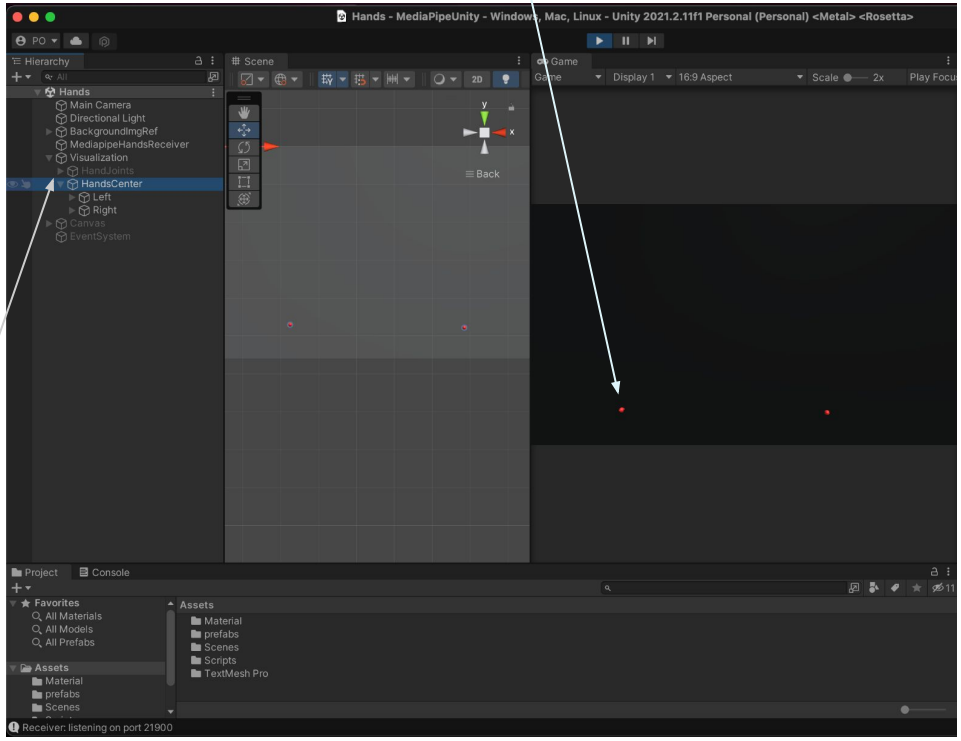
Running the example

1. (!) Run MediaPipe hands and leave it open.

2. Click on the Play button to run the project. You should see spheres on the Unity scene tracking your movement

3. You can see all hands joints.
   Go to the Hierarchy tab and deactivate HandsCenter and activate HandJoints





Drawing lines with your hand

Let's draw a line with the hand position

We will use the line renderer class from Unity
https://docs.unity3d.com/ScriptReference/LineRenderer.html

1. Create an empty object in the scene
   Right click on the hierarchy and click "Create Empty"
   Name it "MyDraw"

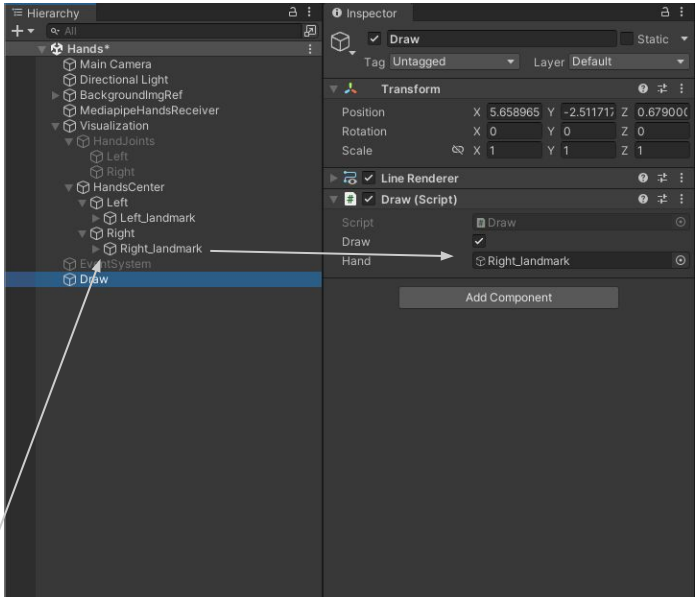
2. Select **MyDraw**, and in the Inspector click on Add Component
   Add a LineRenderer component (write on the top textbox "Line" and it will find it for you).

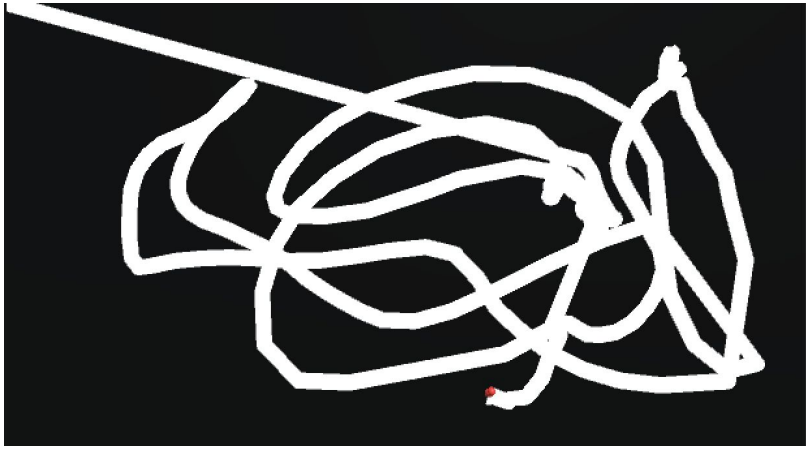
3. Click on Add Component again and type Draw, you will find a script with this name.
   Add it as a component.

4. You need to assign which landmark you want to use to draw. Drag the Left_landmark or Right_landmark object in the **HandsCenter** to the Hand property of the Draw component in the inspector of your MyDraw gameobject. You can also drag any of the landmarks in the **HandJoints** (for example landmark 8 to paint with the tip of the finger)

5. Make sure that the "Draw" checkbox in your **MyDraw** gameobject (just above the landmark) is activated!
   Then, click on Play; your hand will start to draw lines

6. Check how to modify properties of the LineRenderer component, such as colors and size





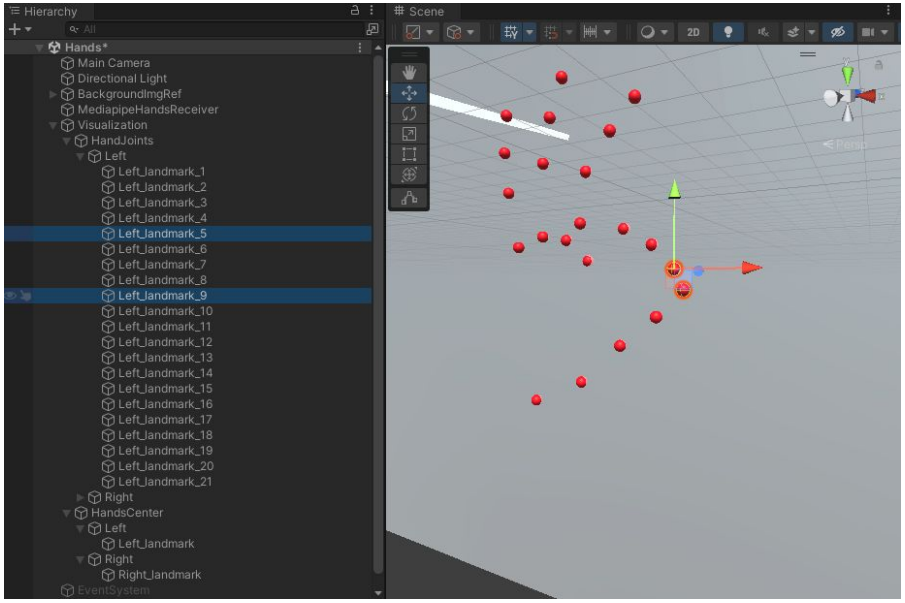
Next steps: Some ideas

1. Using the Draw example, you can modify or create a new script based on this one to start to add functionality
   a. Define when to start / stop to draw
   b. Modify properties of the line
   c. Reset the line drawing (clearing the points list)

2. Use the hand joints to detect gestures
   a. For example, a pinch, when the distance between the two joints of the tip for the fingers are in a close distance

3. Use gestures to change properties and options in the script Draw

4. Try the scene Pose and test how to draw with body landmarks. Check their precision



Homework for next class

- Finish this exercise.

- And add your own modifications: e.g. change the aspect of the lines.

- Also make an equivalent MyDraw gameobject in the other scene of the project, the “Pose” scene. Choose a part of the whole body to draw with and adapt the MyDraw gameobject so that it draws, for example, with the elbow, or the knee, or whichever other part you want.

- Deliver on the campus
  - Unity project in a zip file
  - Include a Build folder with the executable of the project

Resources

Mediapipe Pypi package
https://pypi.org/project/mediapipe/

MediaPipe examples in Python
https://google.github.io/mediapipe/getting_started/python

Line renderer in Unity
https://docs.unity3d.com/ScriptReference/LineRenderer.html