## BLENDED INTENSIVE PROGRAMME
## APPLIED ARTIFICIAL INTELLIGENCE

# AI FOR VIRTUAL REALITY

DR MARCIN BERNAŚ

**UNIVERSITY OF BIELSKO-BIALA, POLAND**

**FACULTY OF MECHANICAL ENGINEERING AND COMPUTER SCIENCE**

Co-funded by
the European Union

FAAI

Welcome Point

University of Bielsko-Biala

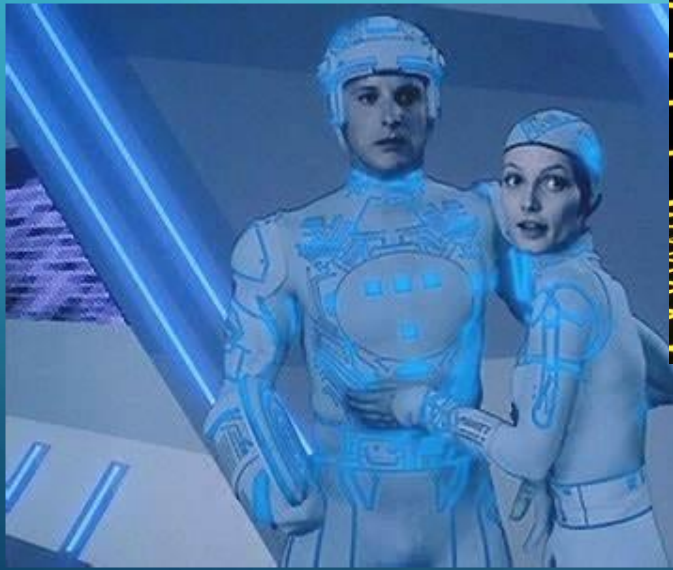
2022-1-PL01-KA220-HED-000088359

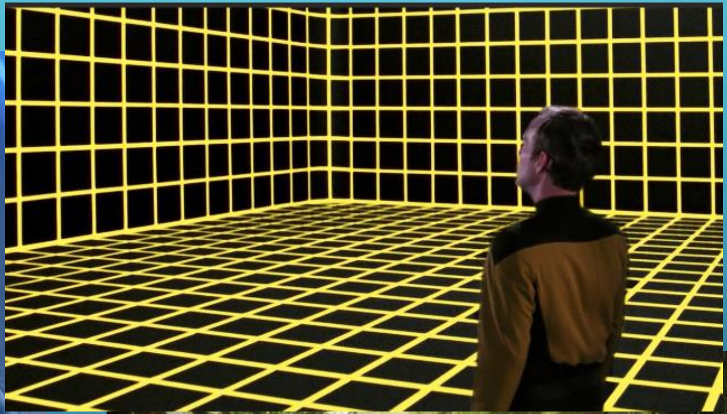
# AGENDA

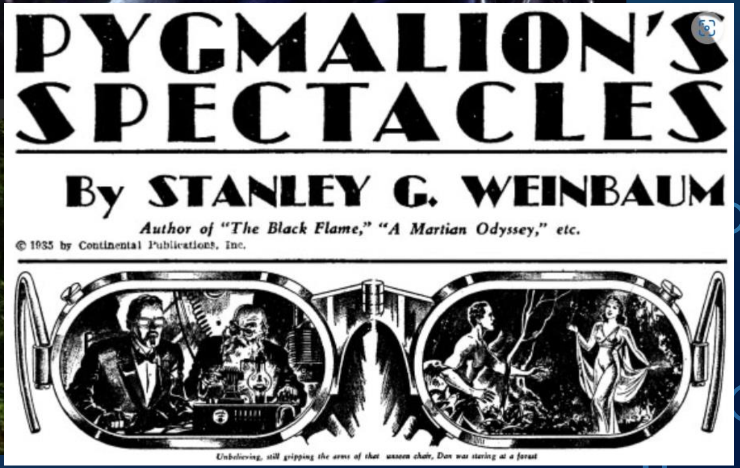
- VR history
- What AI models are used in VR?
- VR and AI in laboratory
- Practical implementation (examples)
  - How we can use models?
  - Building models and using them

# WHAT IS VIRTUAL REALITY?

- Virtual reality (VR) is a simulated experience that employs 3D near-eye displays and pose tracking to give the user an **immersive** feel of a virtual world.







PYGMALION'S SPECTACLES

By STANLEY G. WEINBAUM

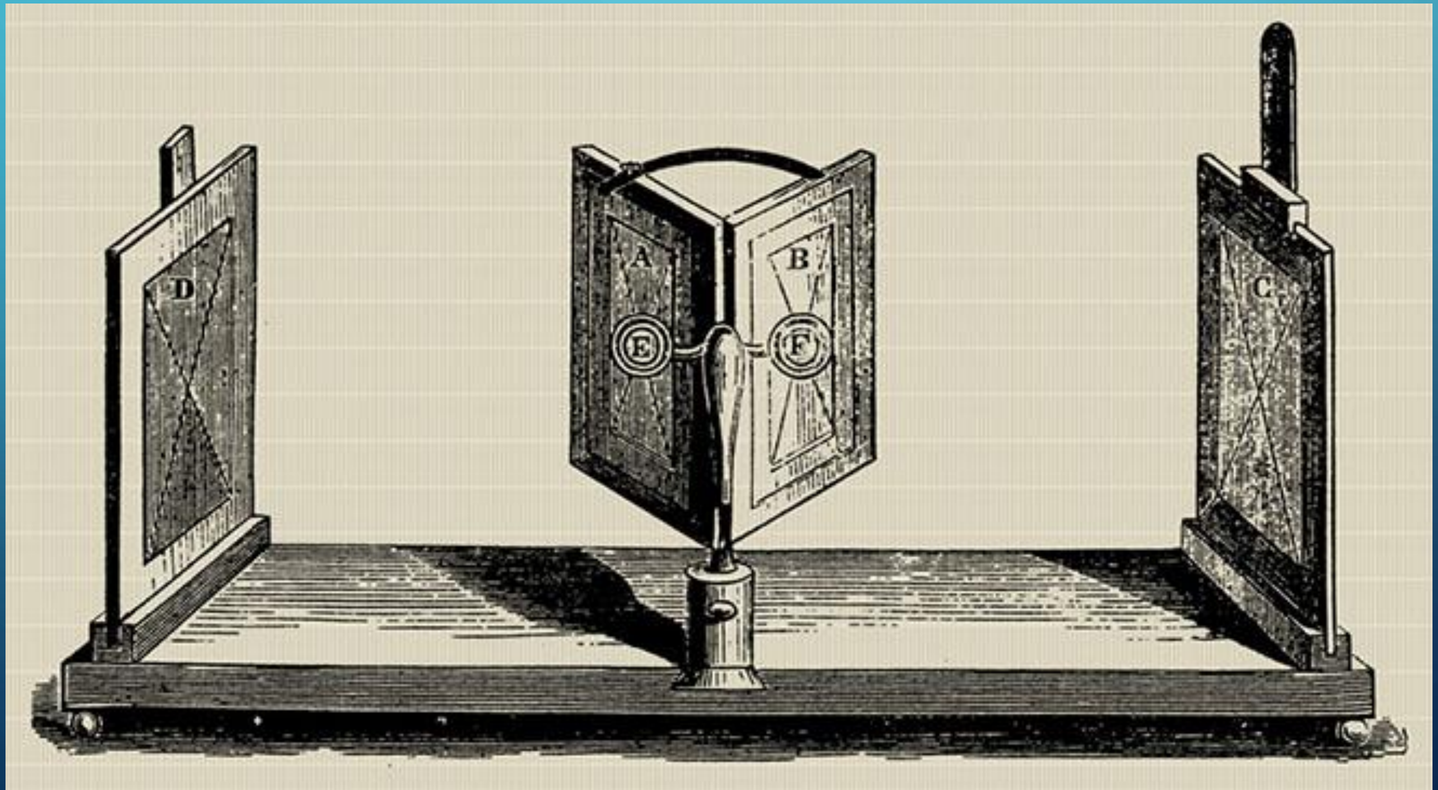
*Author of "The Black Flame," "A Martian Odyssey," etc.*

© 1935 by Continental Publications, Inc.

*Unbelieving, still gripping the arms of that unseen chair, Dan was staring at a forest*

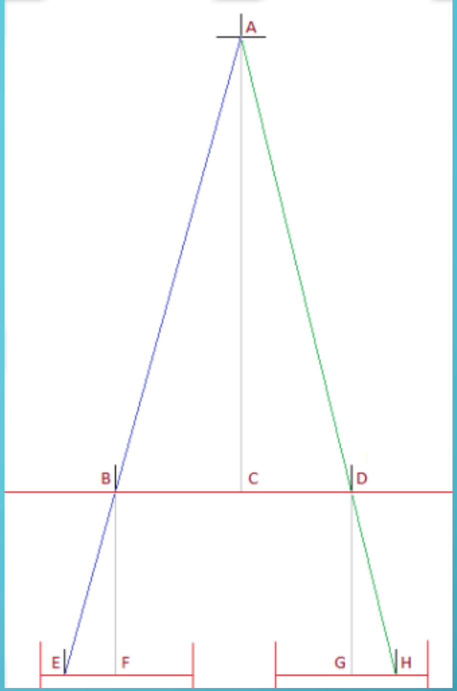
# VR HISTORY

- 1838 - Sir Charles Wheatstone first describe stereopsis
- 1840 - Royal Medal of the Royal Society - binocular vision, building stereoscope.



# STEREO-VISION PRINCIPLES

- The image is recorded by eye from different angle.
- The closer object the difference is bigger
- Brain analyse difference and estimate distance (depth map)



[2D to stereoscopic 3D with AI: Depth map from a single image (owl3d.com)]

1958

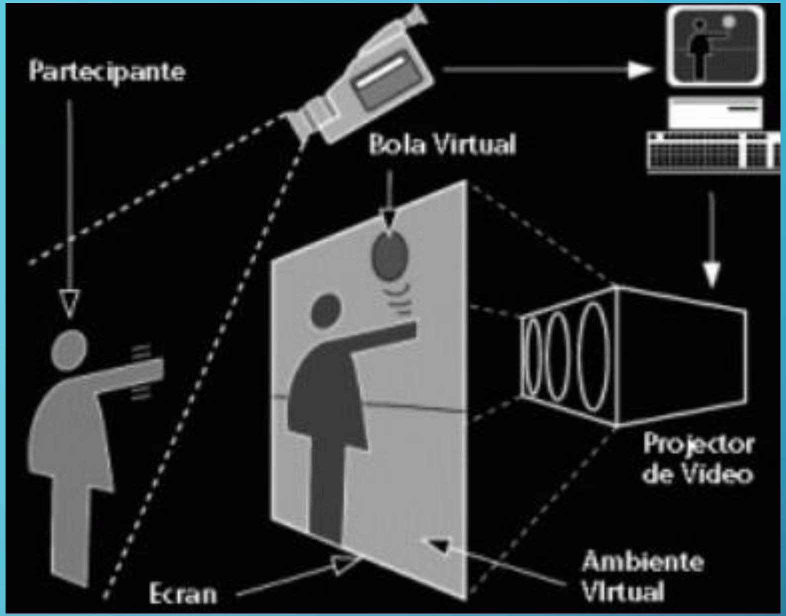
- Cinematographer Morton Heilig created Sensorama - first VR machine (patented in 1962).
- full color 3D video, audio, vibrations, smell and atmospheric effects, such as wind. (no interaction).

1968

- Sutherland created the first virtual reality head-mounted display (HMD).
- This head-mount connected to a computer
- quite primitive as it could only show simple virtual wire-frame shapes.

## 1975

- Krueger’s VIDEOPLACE - first interactive VR platform,  used computer graphics, projectors, video cameras, video displays and position-sensing technology.
- VIDEOPLACE  - dark rooms with large video screens to surround the user in “VR”.



## 1979

- McDonnell-Douglas Corporation integrated VR into its HMD, the VITAL helmet, for military use.
- A head followed the pilot’s eye movements to match computer-generated images.

1985

- Jaron Lanier and Thomas Zimmerman founded VPL Research, Inc.
- company sell VR goggles and gloves.
- developed DataGlove, EyePhone HMD and the Audio Sphere.

1989

- Scott Foster founded Crystal River Engineering Inc.
- develop the audio element of the Virtual Environment Workstation Project (VIEW).
- VR training simulator for astronauts.
- real-time binaural 3D audio processing was developed.

1991

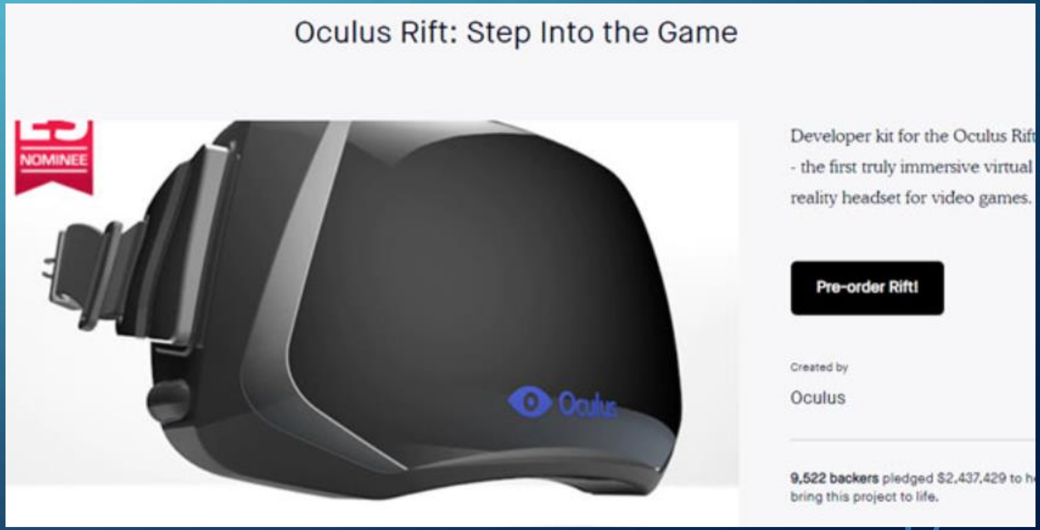
- Antonio Medina, a NASA scientist, designed a VR system to drive the Mars robot rovers from Earth.
- The Virtuality Group launched Virtuality. VR arcade could play in a 3D gaming world - first mass-produced VR entertainment system.
- VR headsets and real-time immersive stereoscopic 3D images. Create multiplayer Pac-Man VR versions (and others).

2012

- Luckey launched a Kickstarter campaign for the Oculus Rift which raised $2.4 million.

2014

- Facebook bought the Oculus VR company for $2 billion. VR gained momentum rapidly after this.
- Sony announced that they were working on Project Morpheus, a VR headset for the PlayStation 4 (PS4).



Oculus Rift: Step Into the Game

Developer kit for the Oculus Rif - the first truly immersive virtual reality headset for video games.

Created by
Oculus

**9,522 backers** pledged $2,437,429 to h bring this project to life.

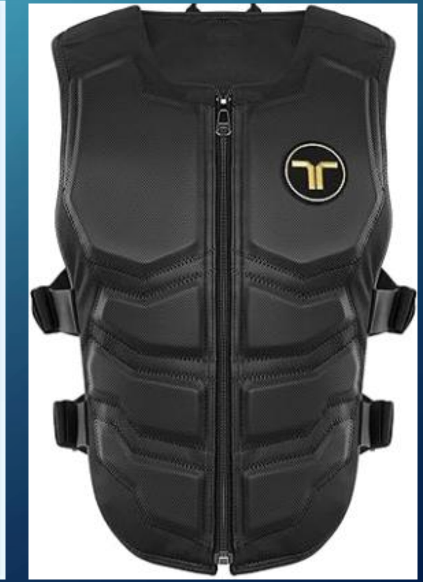
## 2016

- Most of the headsets had dynamic binaural audio.
- Haptic interfaces were underdeveloped. The handsets were typically button-operated. Especially phone based.
- HTC released its HTC VIVE SteamVR headset. This was the first commercial release of a headset with sensor-based tracking which allowed users to move freely in a space.

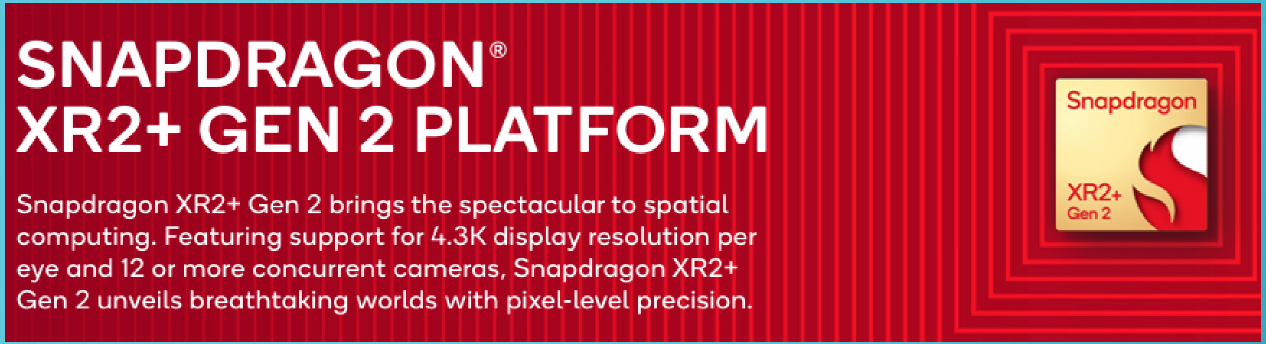
## 2019

- Forbes describes this as The Year Virtual Reality Gets Real.
- Oculus Quest, Facebook’s standalone headset, created a lot of interest and momentum, selling out in many locations and generating $5 million worth of content sales.

- 2023 - Apple Vision Pro - mixed-reality headset at Worldwide Developers Conference (WWDC). The headset will sell for $3,499.
- Meta announced the Meta Quest 3. A mixed-reality headset featuring significant advancements in display quality, processing power, and tracking capabilities. 2160 x 2160 per eye, the Snapdragon XR2+ processor, and inside-out tracking with improved accuracy and field of view. The headset is available for $600.
- Haptic suits – in development phase



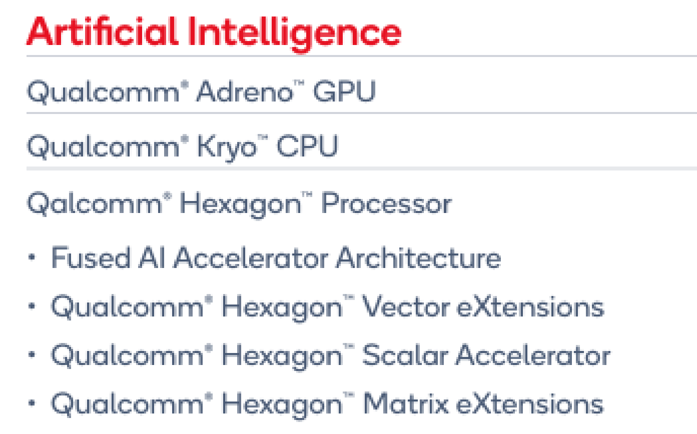
# WHAT IS INSIDE?



## SNAPDRAGON® XR2+ GEN 2 PLATFORM

Snapdragon XR2+ Gen 2 brings the spectacular to spatial computing. Featuring support for 4.3K display resolution per eye and 12 or more concurrent cameras, Snapdragon XR2+ Gen 2 unveils breathtaking worlds with pixel-level precision.

Snapdragon
XR2+
Gen 2



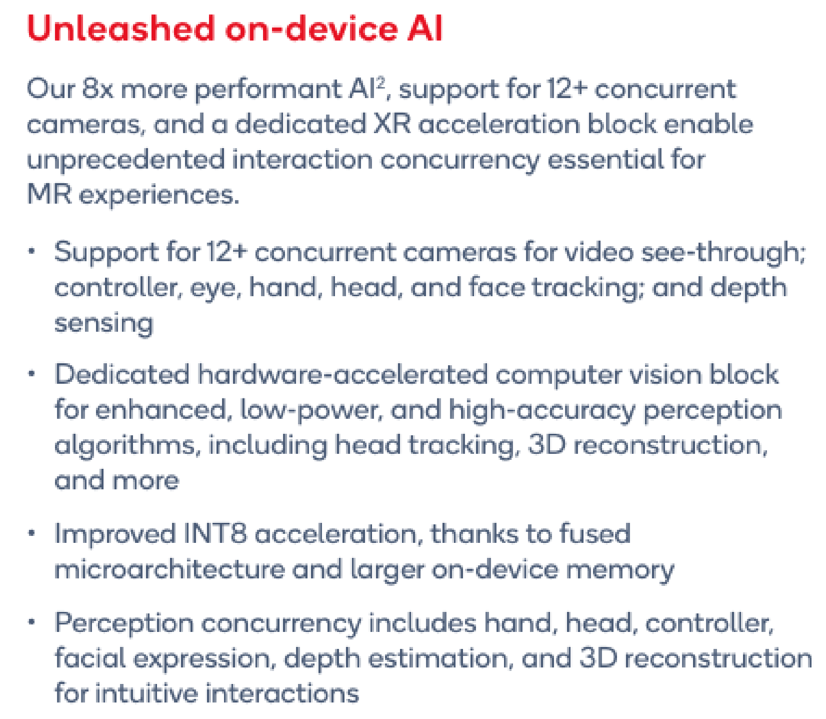
### Artificial Intelligence

Qualcomm® Adreno™ GPU

Qualcomm® Kryo™ CPU

Qalcomm® Hexagon™ Processor

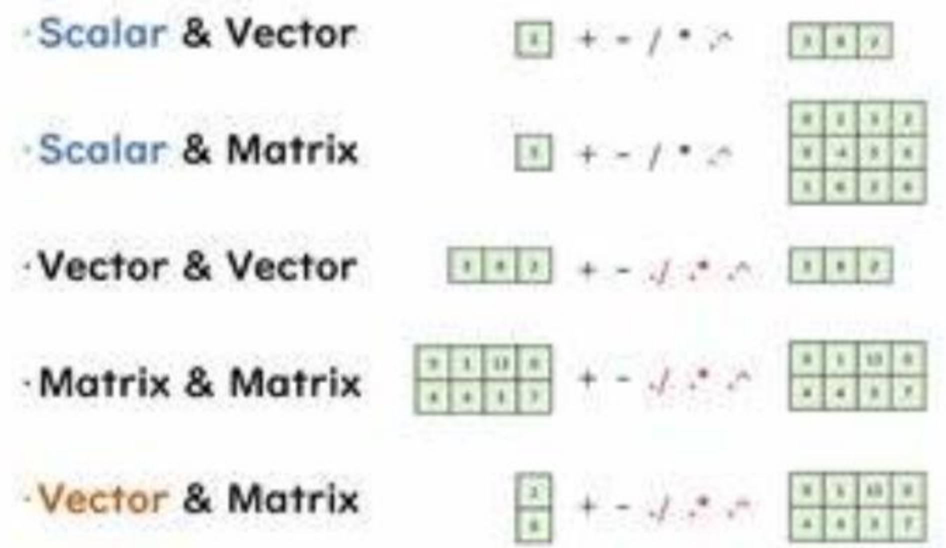
- Fused AI Accelerator Architecture
- Qualcomm® Hexagon™ Vector eXtensions
- Qualcomm® Hexagon™ Scalar Accelerator
- Qualcomm® Hexagon™ Matrix eXtensions



### Unleashed on-device AI

Our 8x more performant AI[2], support for 12+ concurrent cameras, and a dedicated XR acceleration block enable unprecedented interaction concurrency essential for MR experiences.

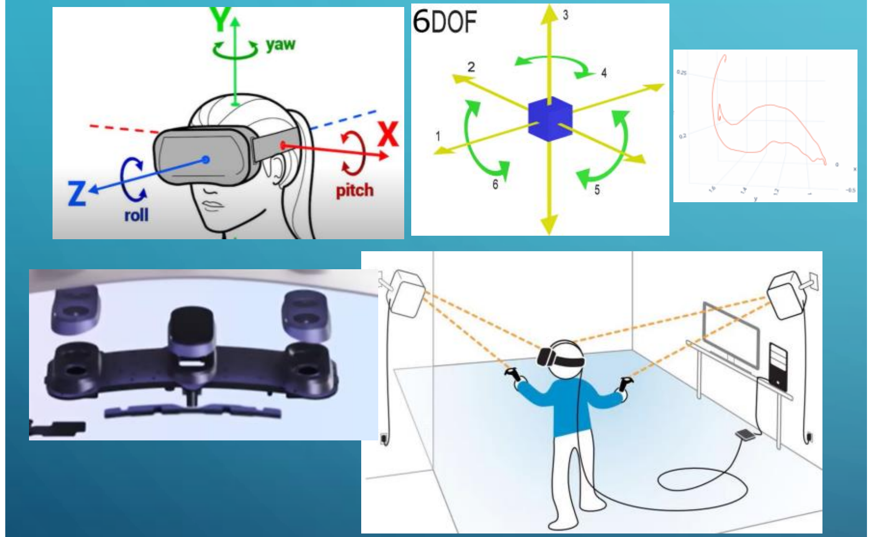
- Support for 12+ concurrent cameras for video see-through; controller, eye, hand, head, and face tracking; and depth sensing
- Dedicated hardware-accelerated computer vision block for enhanced, low-power, and high-accuracy perception algorithms, including head tracking, 3D reconstruction, and more
- Improved INT8 acceleration, thanks to fused microarchitecture and larger on-device memory
- Perception concurrency includes hand, head, controller, facial expression, depth estimation, and 3D reconstruction for intuitive interactions



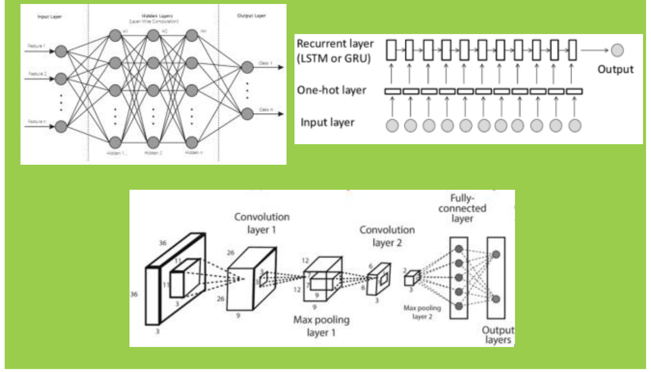
# HOW DOES IT WORKS

!!! All the process should be faster than 50ms max !!!

Collect sensor data



Estimate the position of eyes, head and joints (AI)



Generate the scene image

VR headset SPECS and TERMINOLOGY explained! (youtube.com)

# POSSIBILITIES

- VR /AR support
- Upper body tracking (IK)
- Legs estimation (AI)

Quest 3 Now Has Inside-Out Body Tracking & MR Occlusion (uploadvr.com)

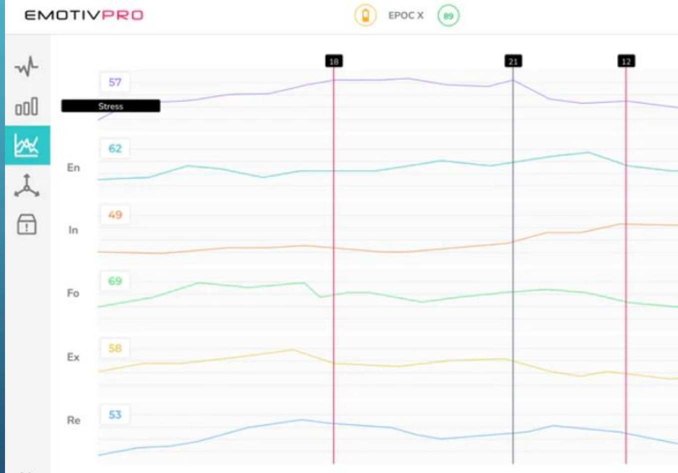
# IMMERSION REQUIREMENTS

Michael Abrash on VR at Steam Dev Days in 2014 based on VR research at Valve define minimal standard:

- A wide field of view (80 degrees or better)
- Adequate resolution (1080p or better)
- Low pixel persistence (3 ms or less)
- A high enough refresh rate (>60 Hz, 95 Hz is enough but less may be adequate)
- Global display where all pixels are illuminated simultaneously (rolling display may work with eye tracking.)
- Optics (at most two lenses per eye with trade-offs, ideal optics not practical using current technology)
- Rock-solid tracking – translation with millimeter accuracy or better, orientation with quarter degree accuracy or better, and volume of 1.5 meter or more on a side
- Low latency (20 ms motion to last photon, 25 ms may be good enough)

# EEG READING

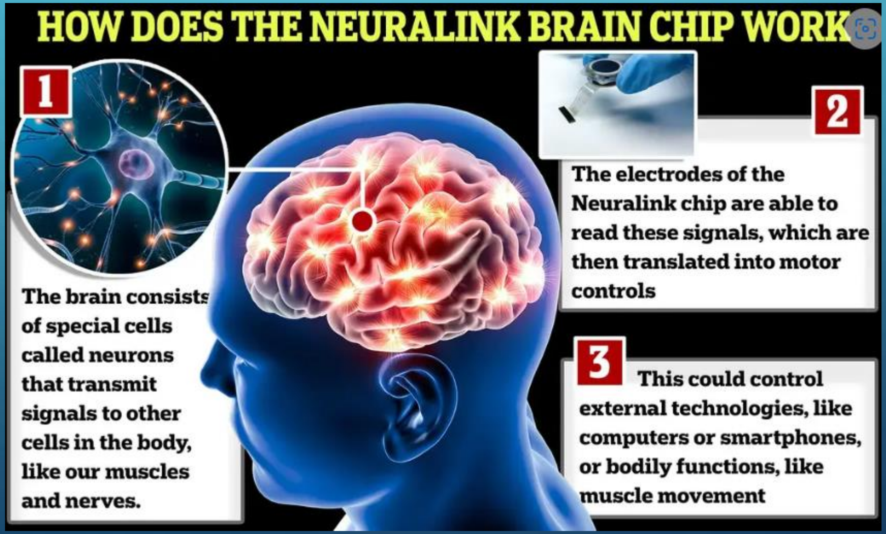
- Accelerate Your Human Behavior Research with EMOTIV's Revolutionary Performance Metrics.
- Driven by our proprietary machine-learning algorithms, AI, and the world's largest EEG data set, we've been building and perfecting our Performance Metrics for over a decade. In fact, we're the first neurotech company in the world to build them.



EMOTIV | Brain Data Measuring Hardware and Software Solutions

# NEURAL LINKS (2024)

- Elon Musk, neurotechnology company Neuralink, informed that first human received an implant from the brain-chip startup and is recovering well.
- The surgery is not a surprise: the US Food and Drug Administration (FDA) had given the company clearance in September to carry out the first trial of its implant on humans.

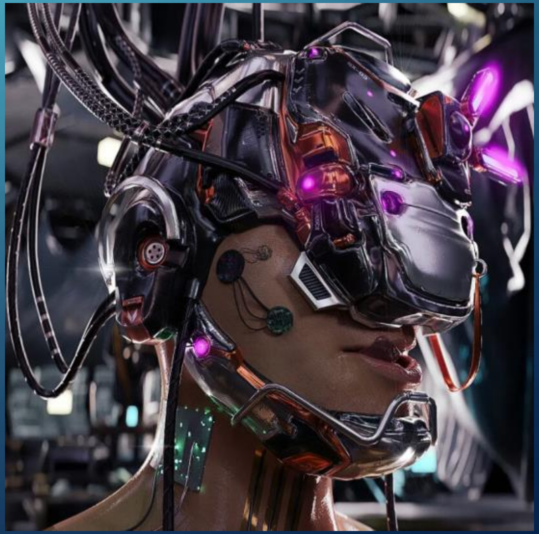


HOW DOES THE NEURALINK BRAIN CHIP WORK

1 The brain consists of special cells called neurons that transmit signals to other cells in the body, like our muscles and nerves.

2 The electrodes of the Neuralink chip are able to read these signals, which are then translated into motor controls

3 This could control external technologies, like computers or smartphones, or bodily functions, like muscle movement



ArtStation - Cyberpunk Head

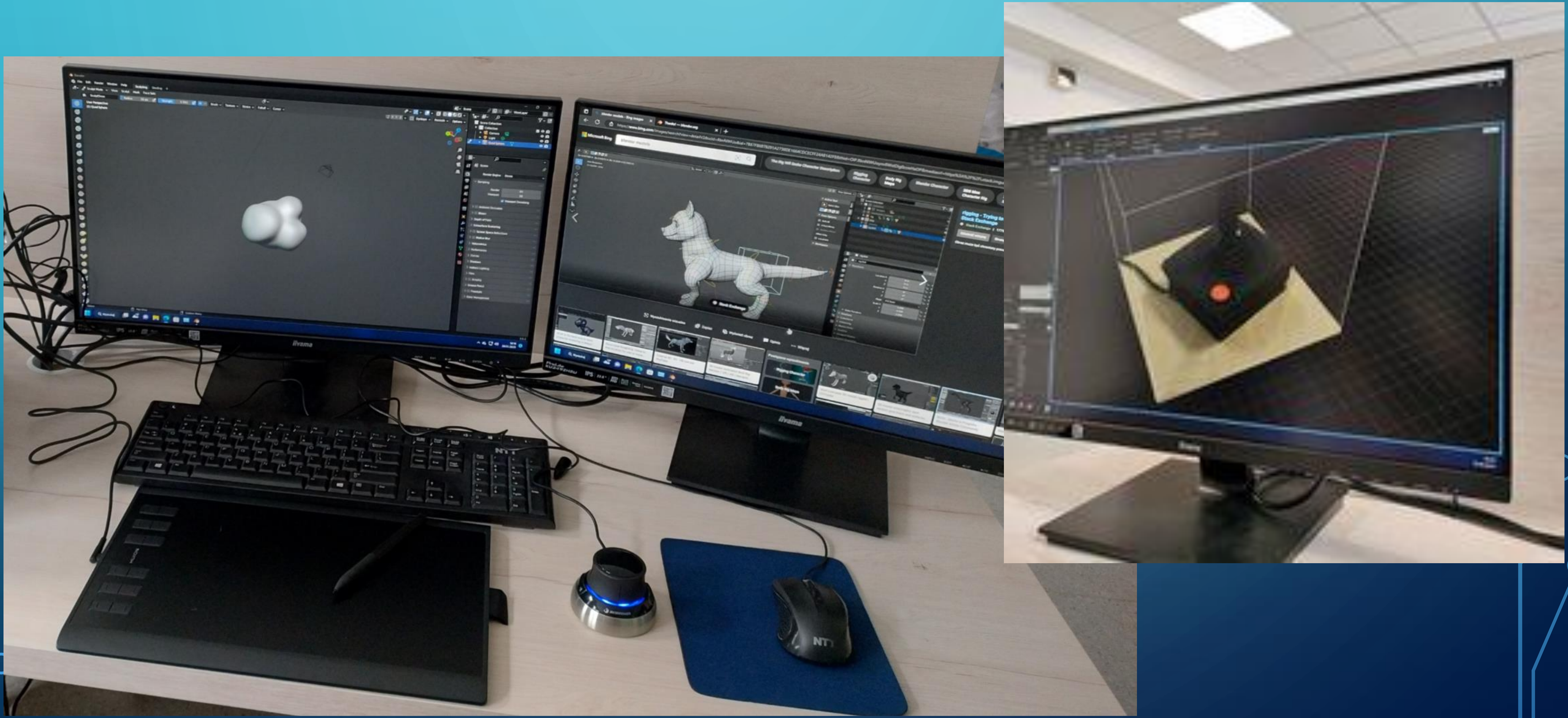
# GAME DEV LABORATORY

15 computers with Ryzen 5500 GPU NVIDIA RTX 3060Ti allows:

- ✓ Creating an d tuning AI models
- ✓ Modelling generating 3D objects
- ✓ Project of 2D, 3D Worlds in Unity, Unreal or Godot

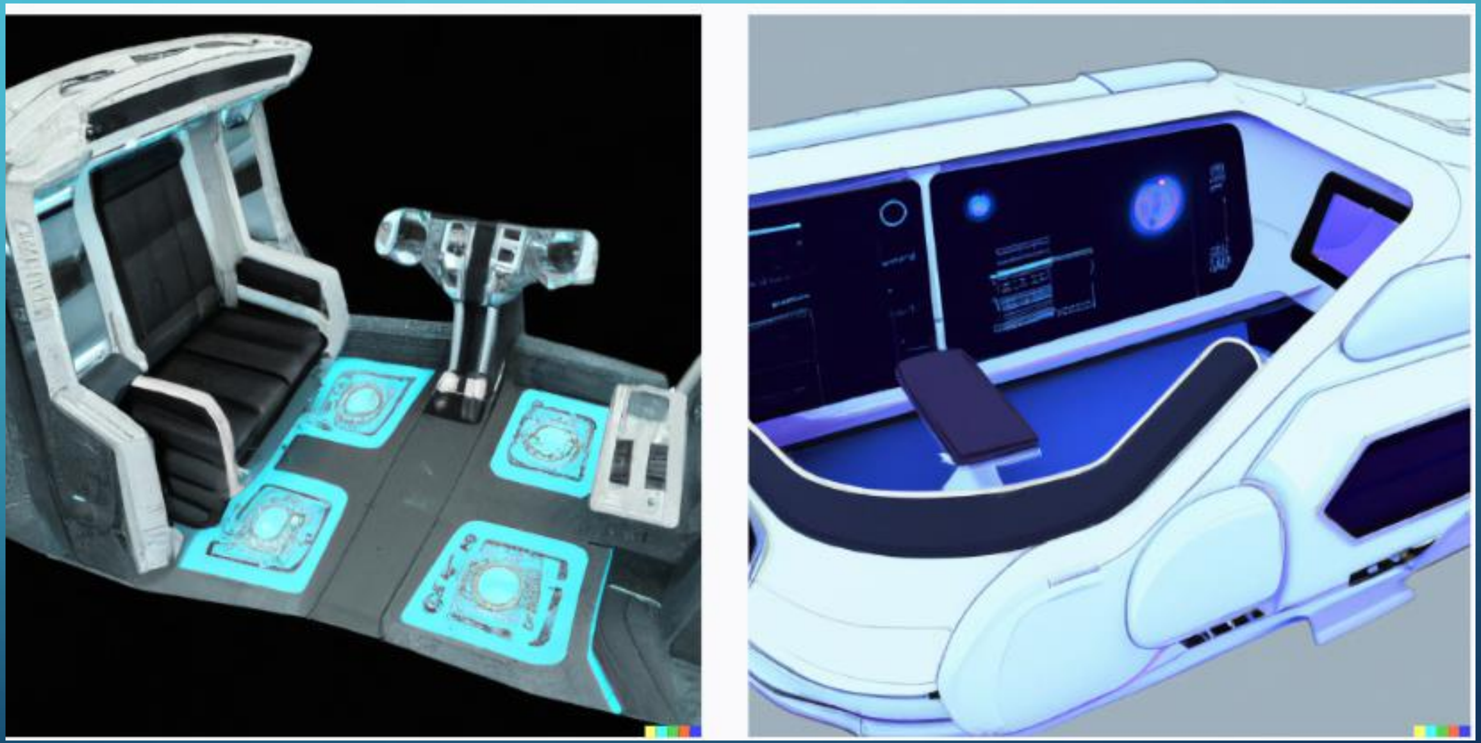
# OBJECT CREATION

- The models are created with suport of AI tools like photogrammetry.

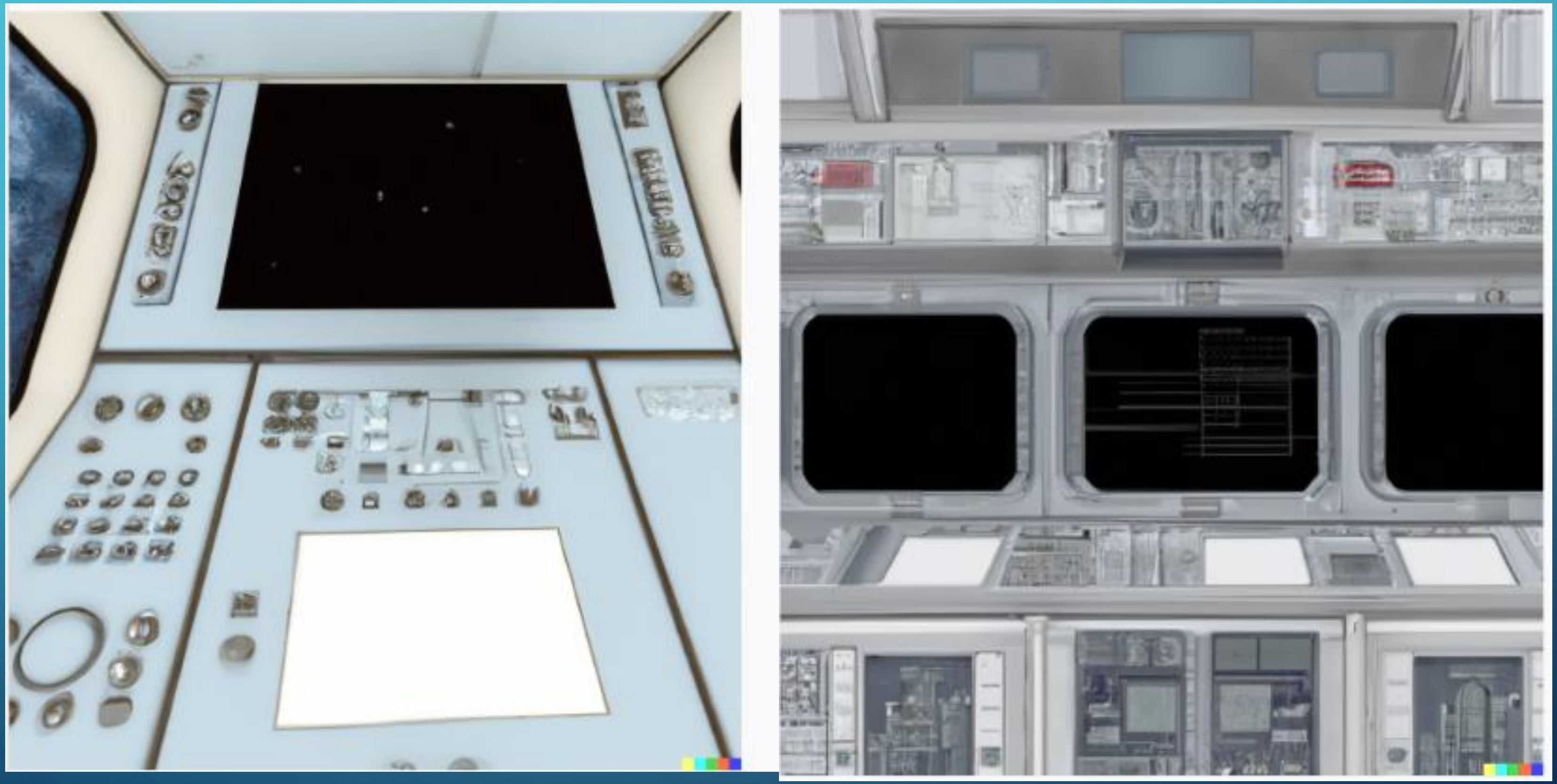


# CREATE CONCETPS FOR PROJECTS (DALL-E)

**Generate the space shuttle cabin from 23 century. It should be simple**

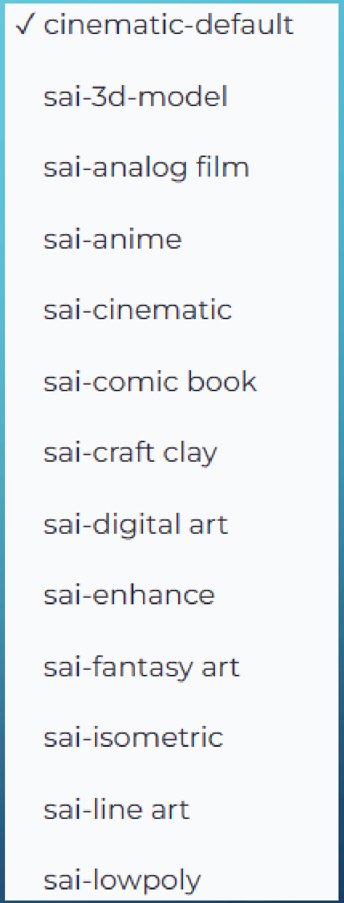


# GENERATE TEXTURE ON WALL

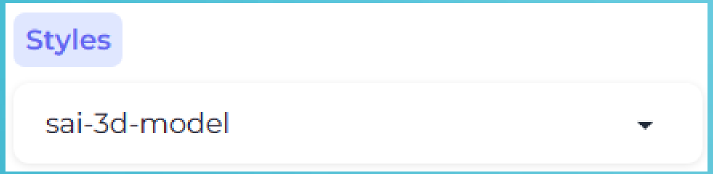


# CONCEPT (STABLE DIFFUSION)

https://stablediffusionweb.com

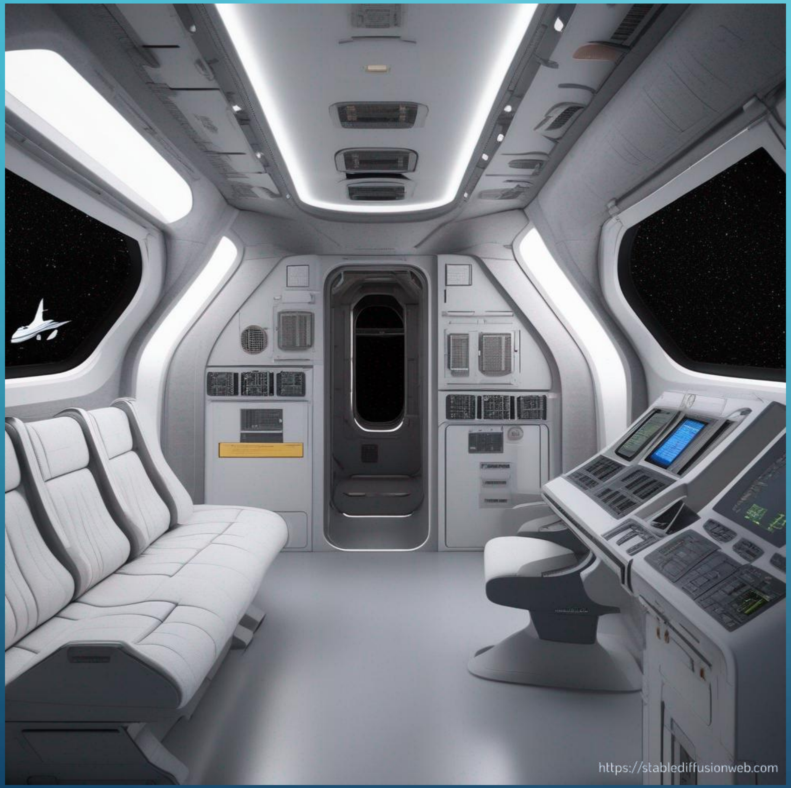


# WAY OF TUNE MODEL

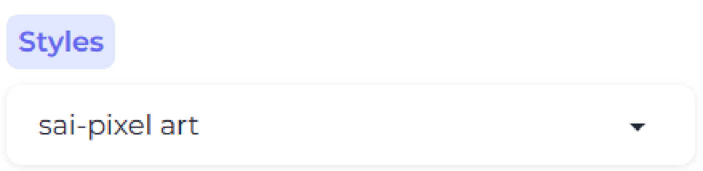


Styles

sai-3d-model

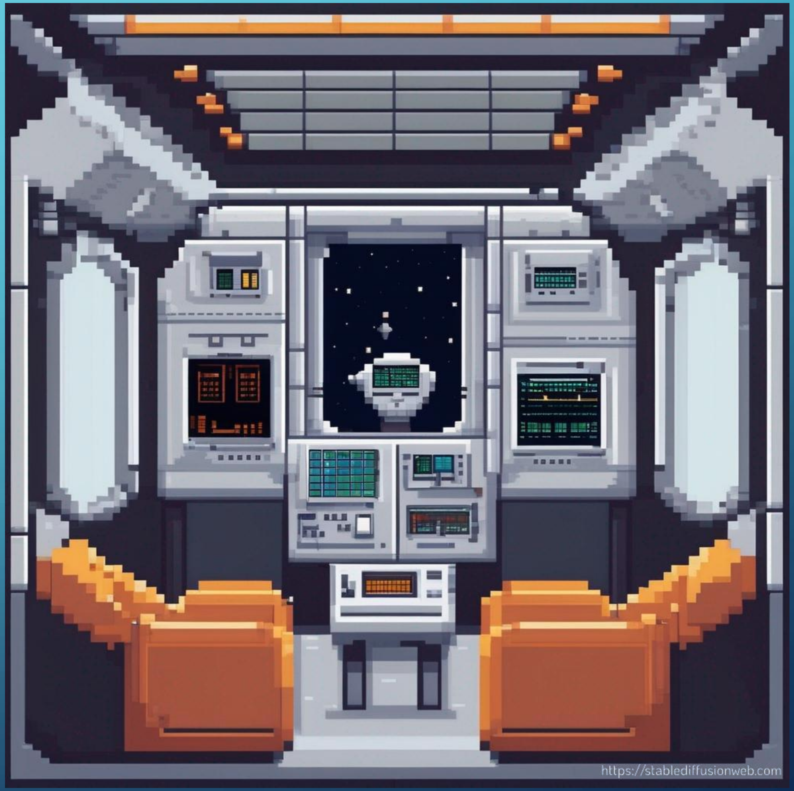


https://stablediffusionweb.com



Styles

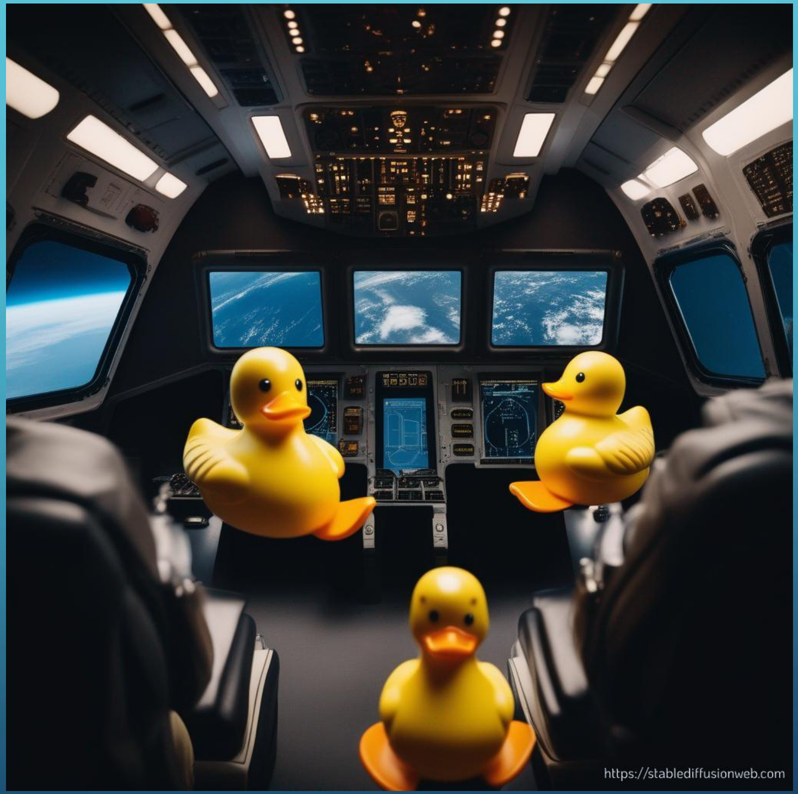
sai-pixel art



https://stablediffusionweb.com

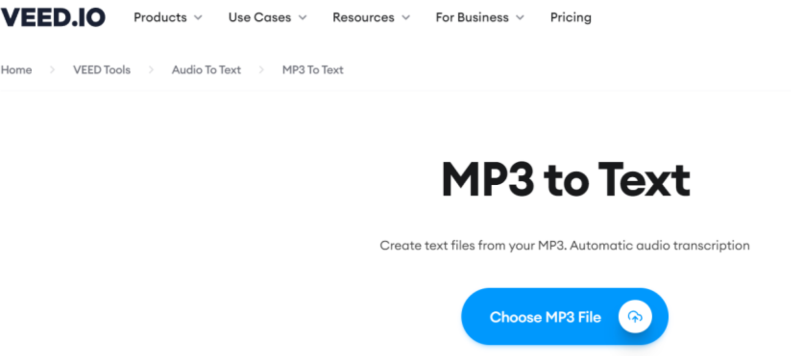
# PRECISION

- Generate the space shuttle cabin from 23 century. Inside sits 2 pilots. On the floor lays yellow rubber duck. From ceiling sticks lots of cables.



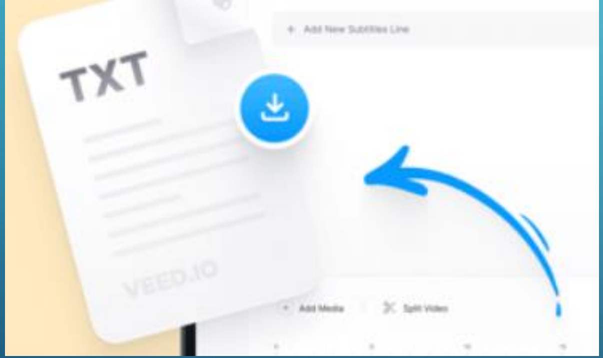
https://stablediffusionweb.com

# SPEECH TO TEXT CONVERSION



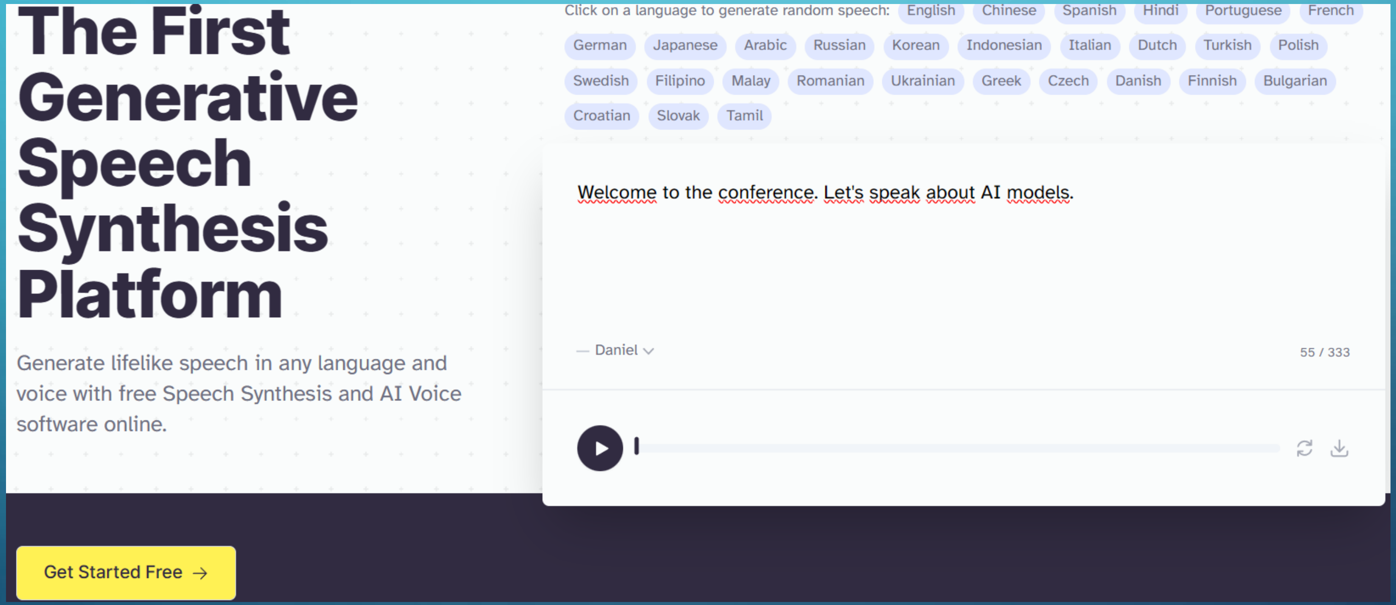
MP3 to Text

Create text files from your MP3. Automatic audio transcription



# ALTERNATIVE

- Speech Synthesis: Generate AI Audio & Voiceovers (elevenlabs.io)



**The First Generative Speech Synthesis Platform**

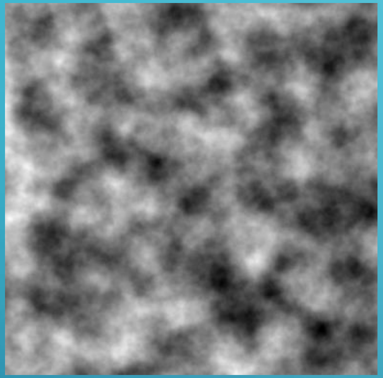
Generate lifelike speech in any language and voice with free Speech Synthesis and AI Voice software online.

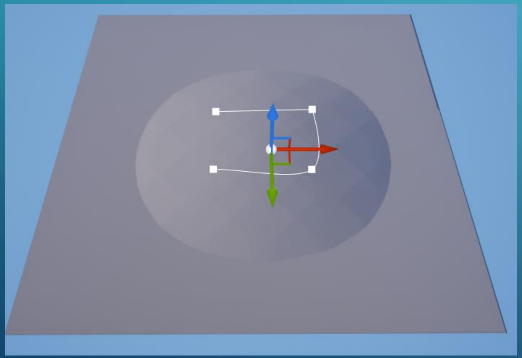
Click on a language to generate random speech:

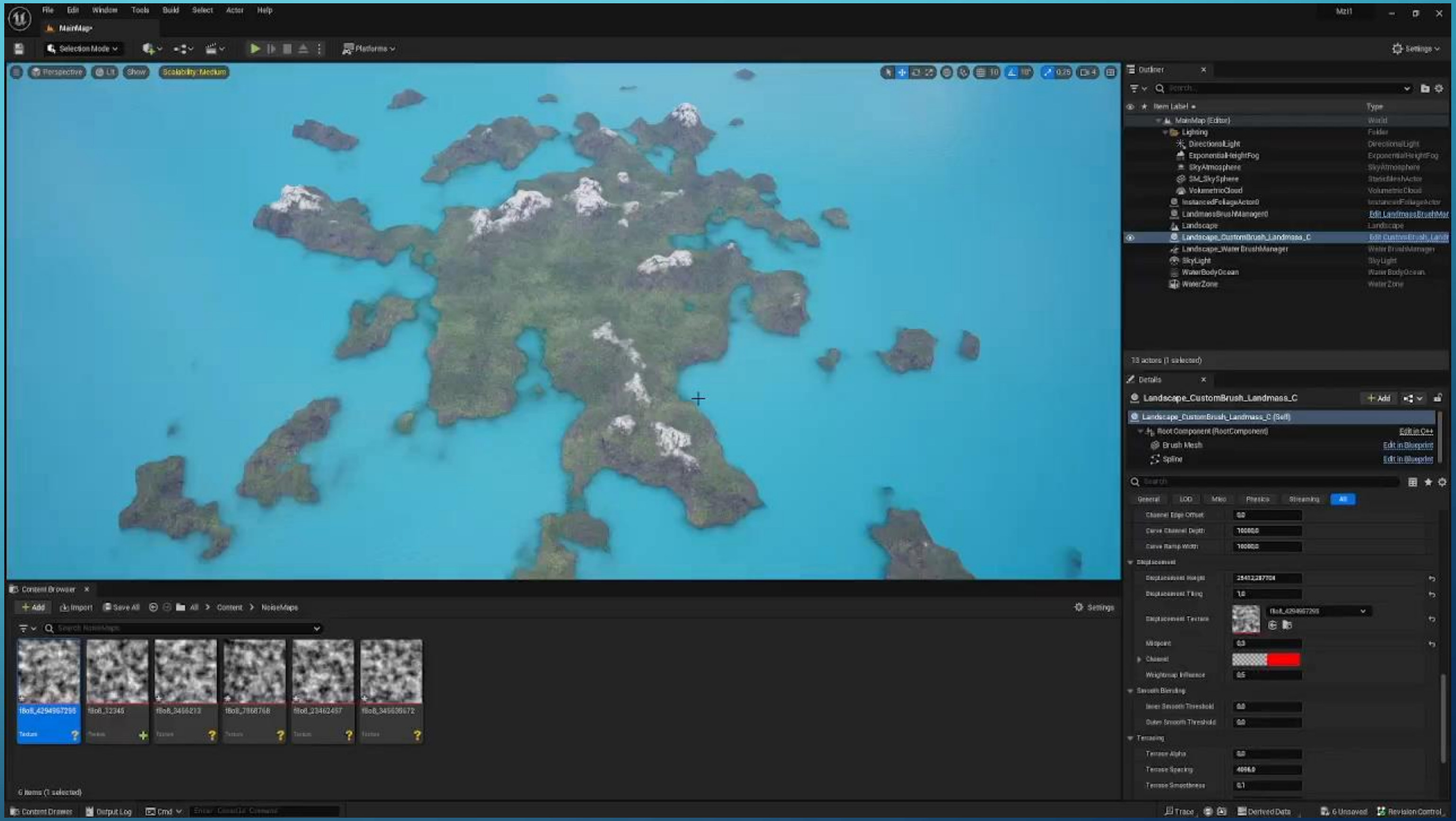
Welcome to the conference. Let's speak about AI models.

# WORLD CREATION (UNREAL)

- Perlin Noise

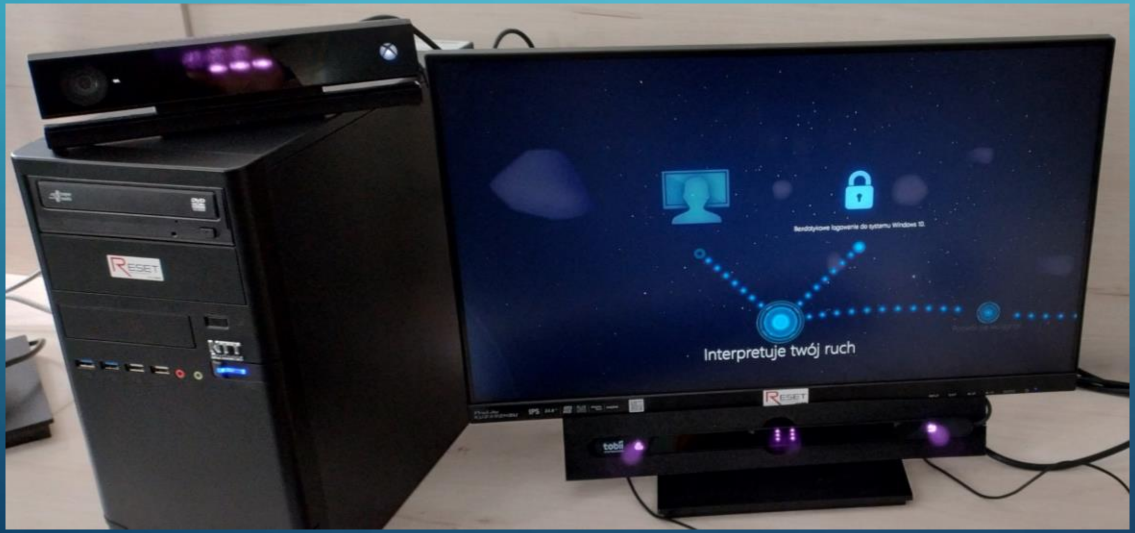


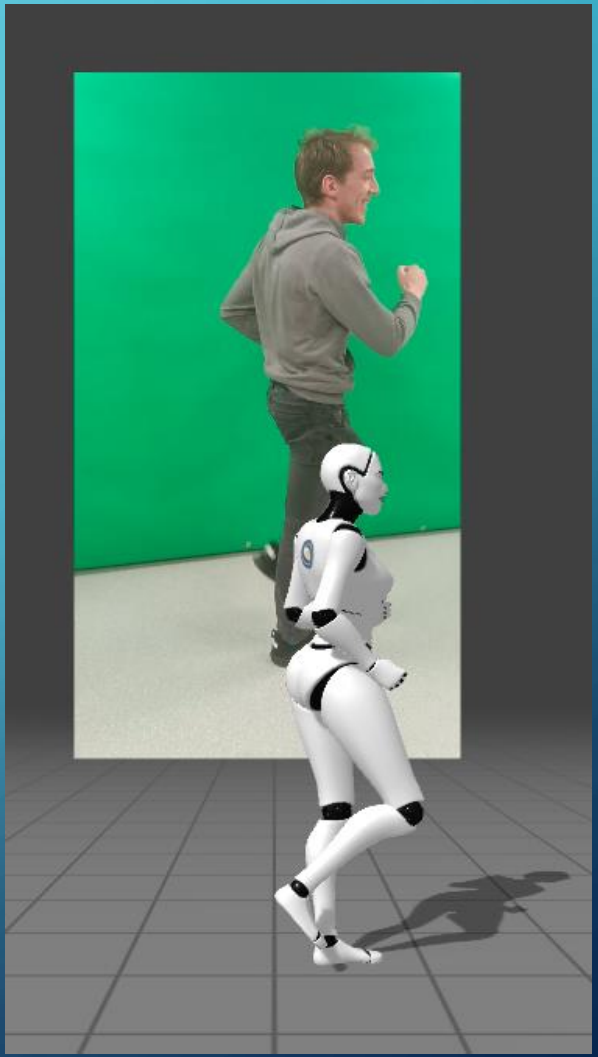




# MOVEMENT TRACKING

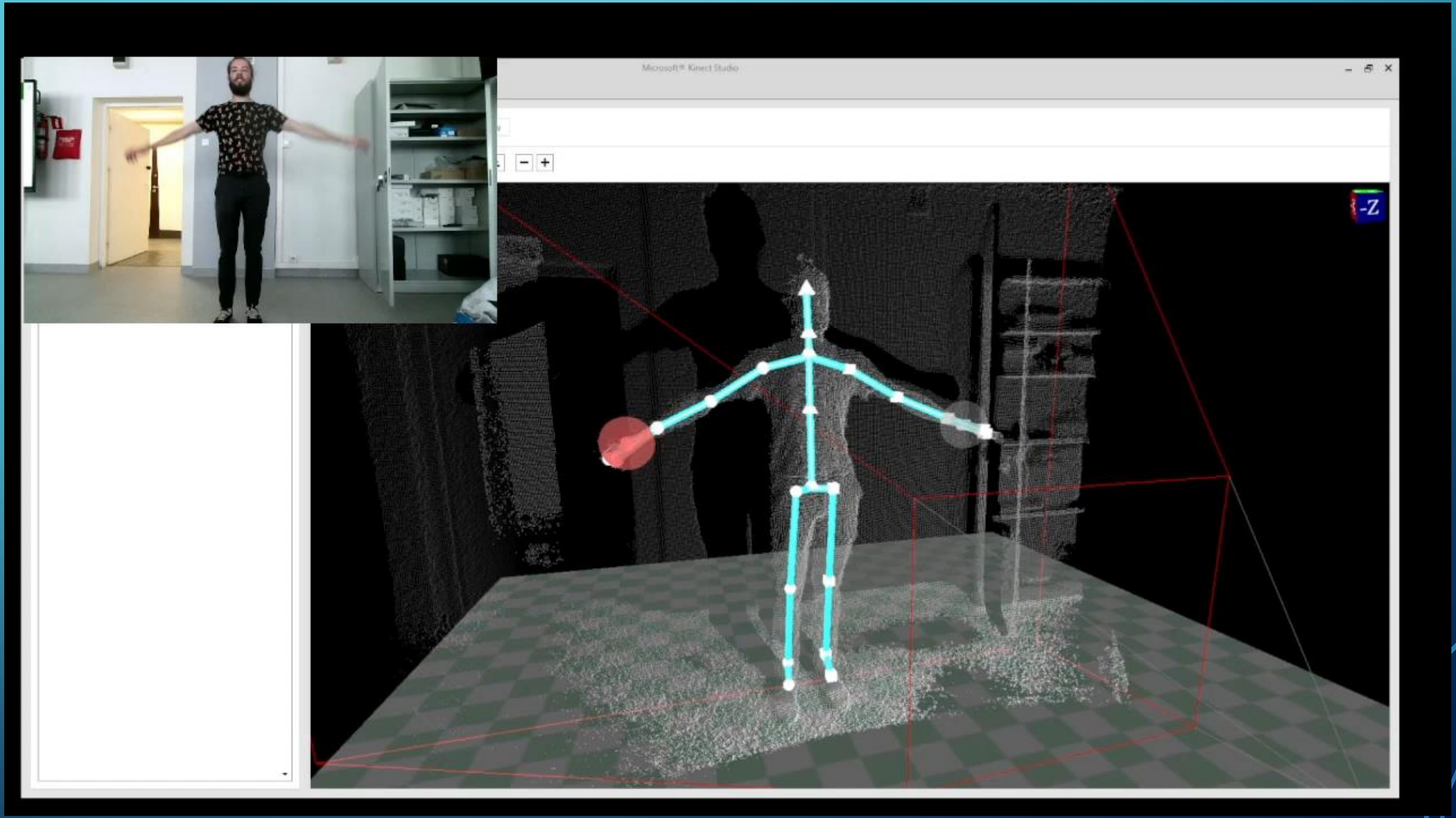
- use of "green screen„
- character and face tracking (Kinect)
- eye tracking (Eyetracker)
- using the character database + movement sequences





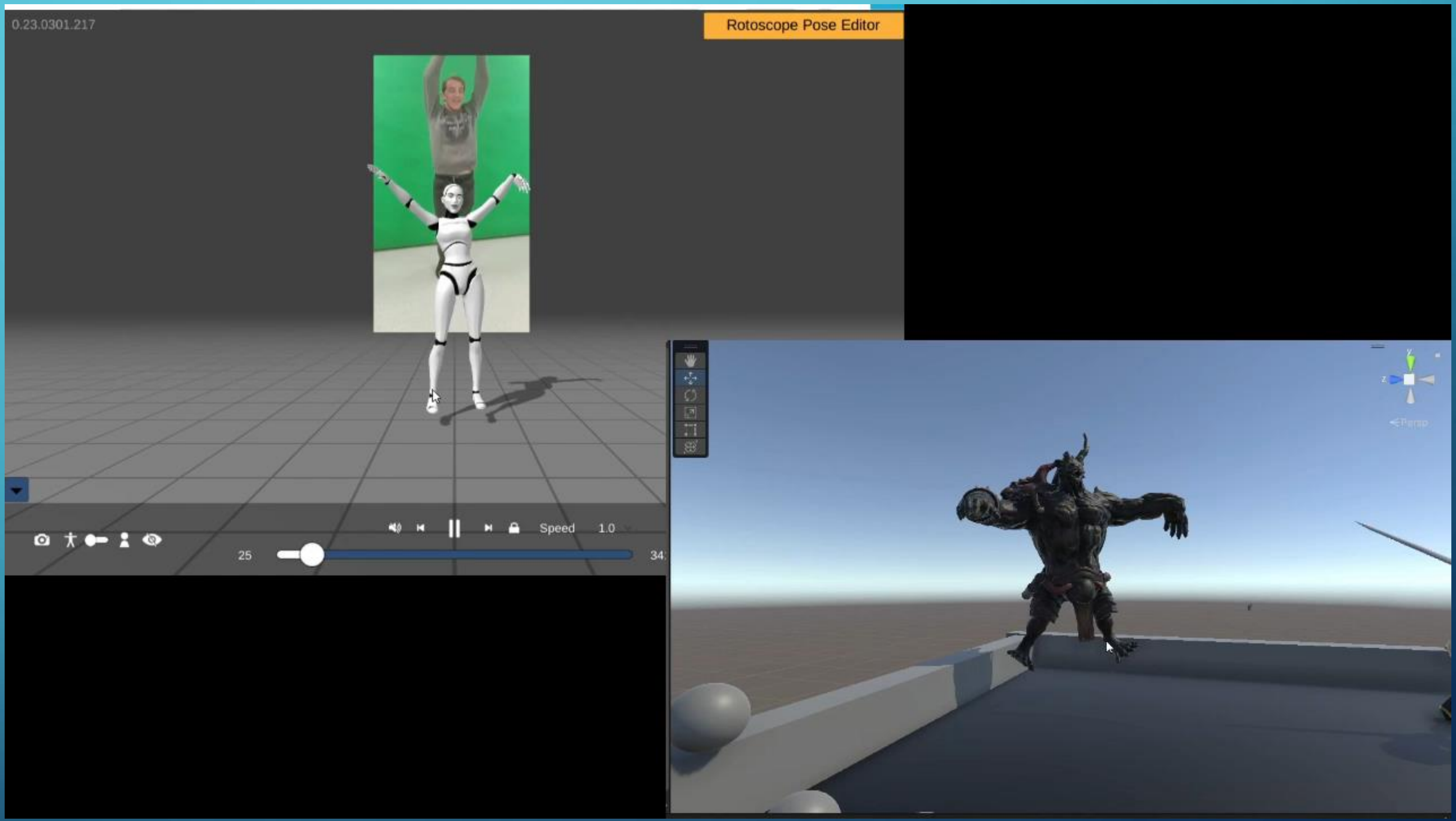
# AI MODEL IN KINECT 2

- body tracking
- face tracking
- depth tracking
- voice recognition
- object tracking



# AI SUPORT (BY DEEP MOTION)

- camera movement
- motion mapping to the bone system + DNN
- mapping motion animation to the selected Avatar



# HOLO-DECK ?

- The connection of virtual and real environment

# GOGLE VR ORAZ AR

- designing and optimizing solutions in VR and AR technology is possible using:
- Oculus Quest 2/3

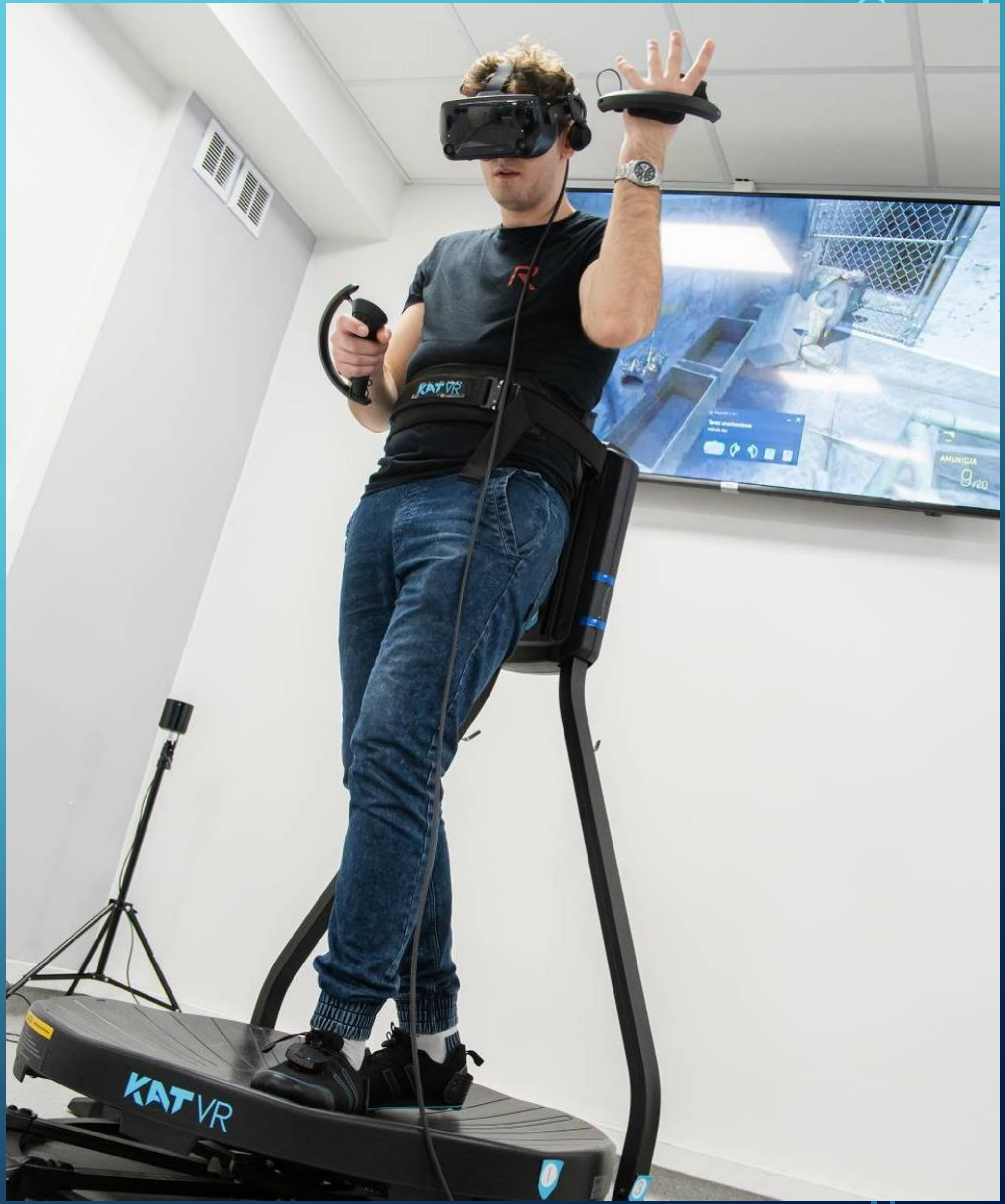
✓ HP Reverb

✓ Valve Index

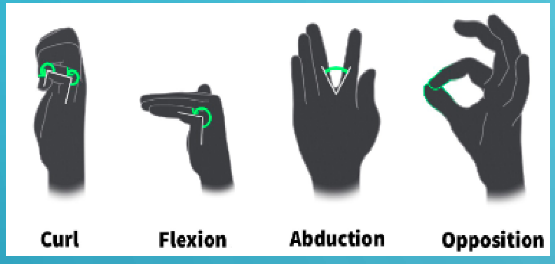
✓ Holo Lens 2

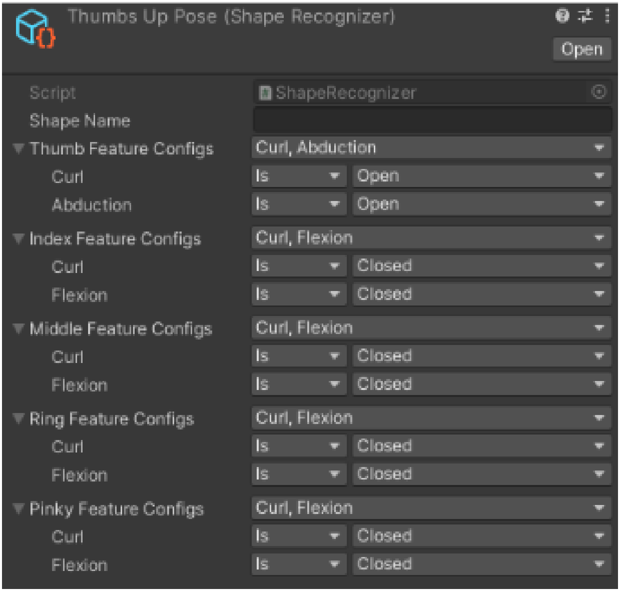
# INTERACTION

- VR controllers (tracking body and hand movements)
- movement of lower limbs on the KatVR treadmill
- Detection using cameras



# GESTURE RECOGNITION (QUEST 2)



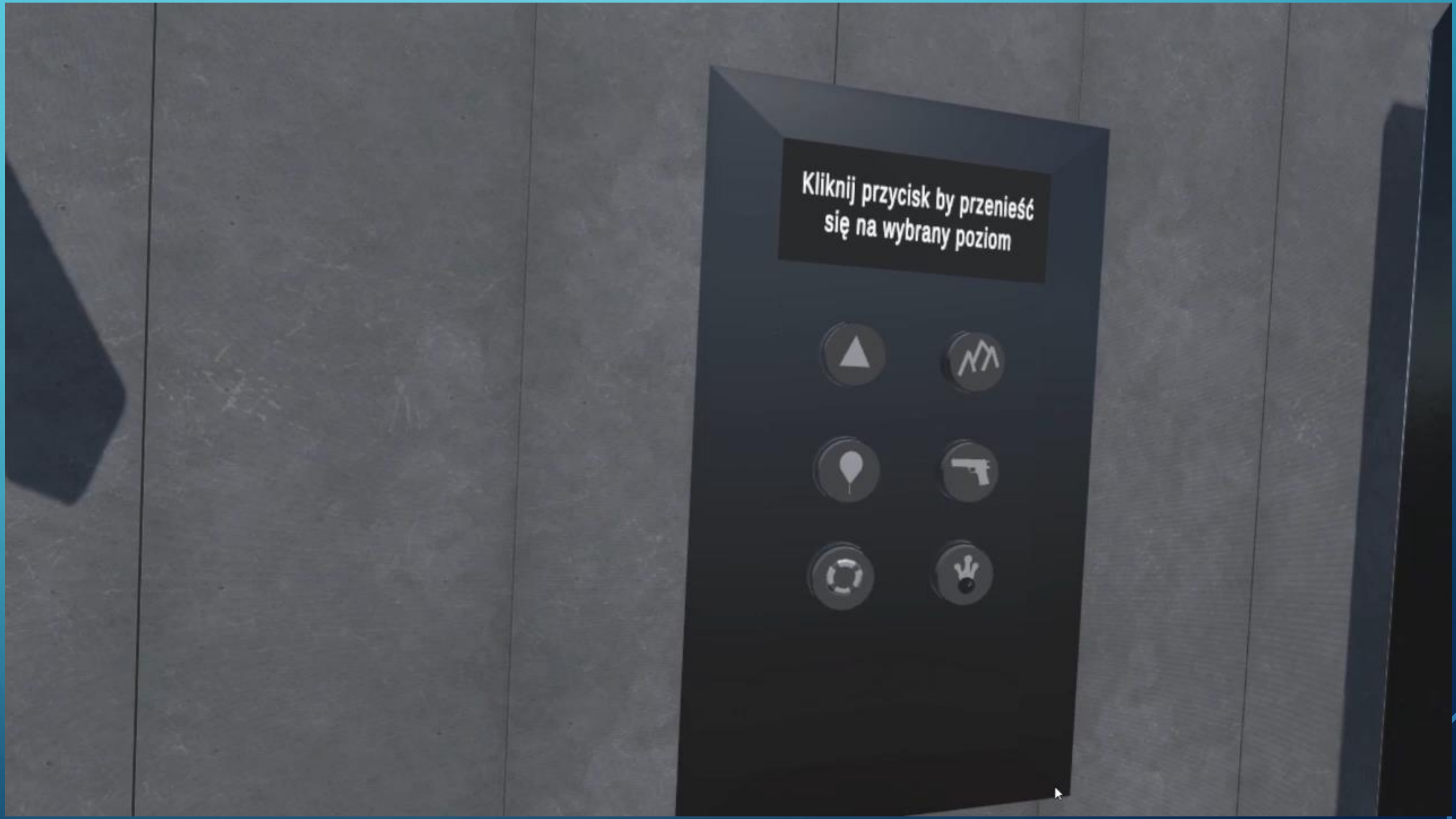


Thumbs Up Pose (Shape Recognizer)

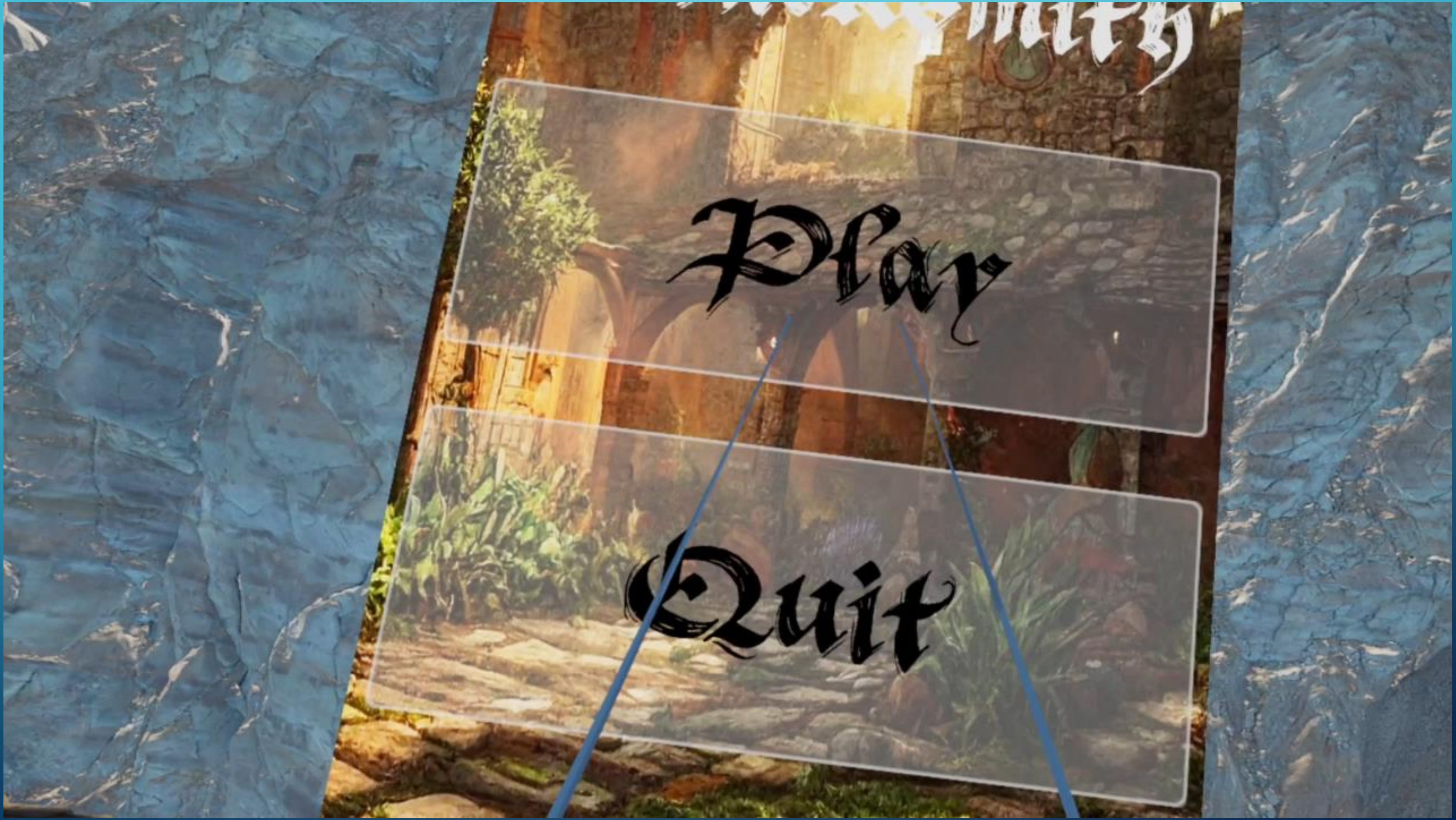
Script: ShapeRecognizer
Shape Name:
Thumb Feature Configs: Curl, Abduction
  Curl: Is Open
  Abduction: Is Open
Index Feature Configs: Curl, Flexion
  Curl: Is Closed
  Flexion: Is Closed
Middle Feature Configs: Curl, Flexion
  Curl: Is Closed
  Flexion: Is Closed
Ring Feature Configs: Curl, Flexion
  Curl: Is Closed
  Flexion: Is Closed
Pinky Feature Configs: Curl, Flexion
  Curl: Is Closed
  Flexion: Is Closed

# PADS TRACKING (CONTROLER)

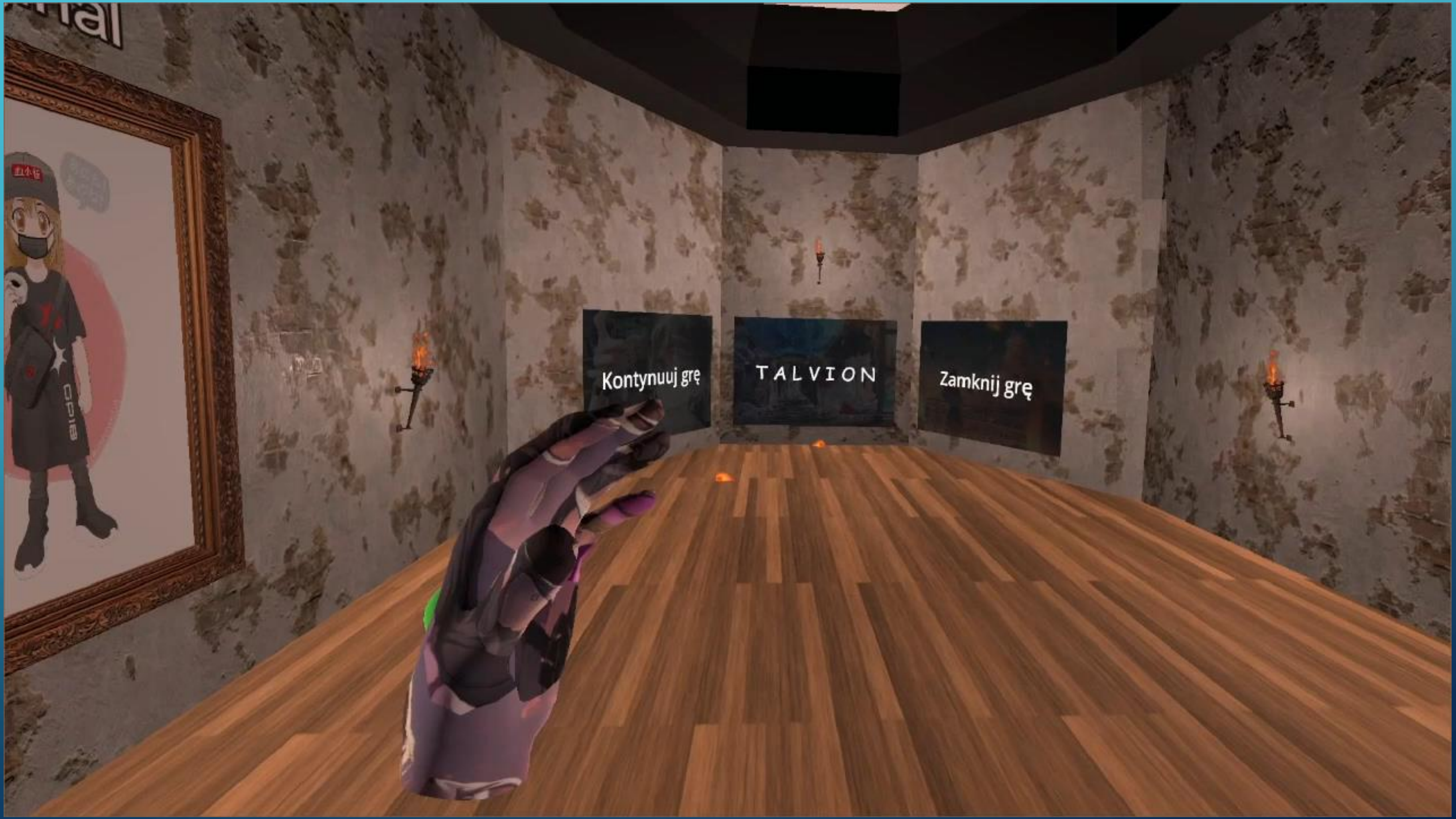
- support for the Open XR standard
- original implementation of holds



# MOVEMENT ACTIONS (UNREAL)

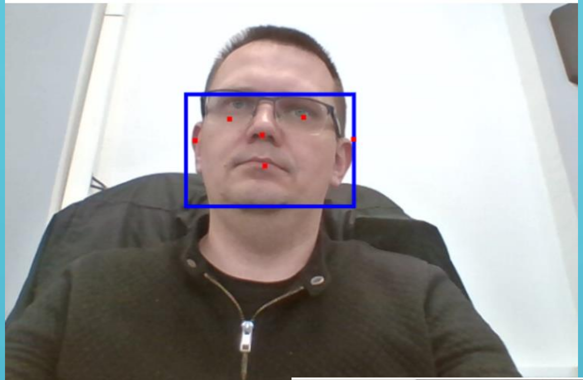


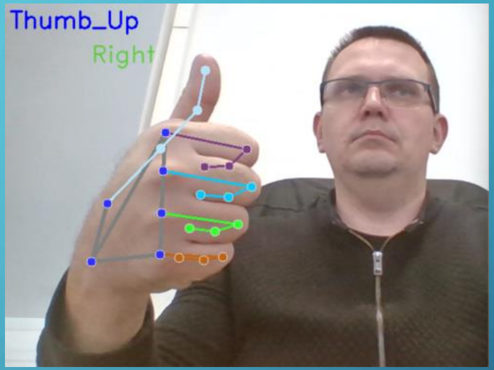
# NEW GESTURES CREATION (GODOT)

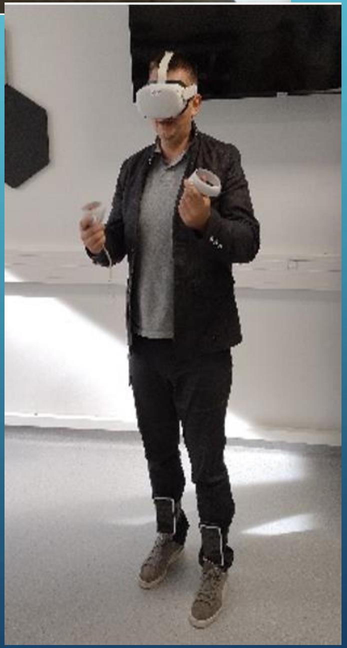


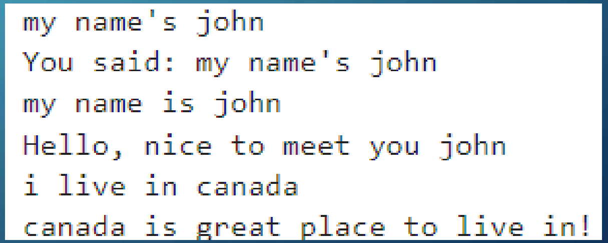
# EXAMPLES OF AI MODELS

- Face tracking
- Hands tracking
- Legs analysis
- Audio to text / language model





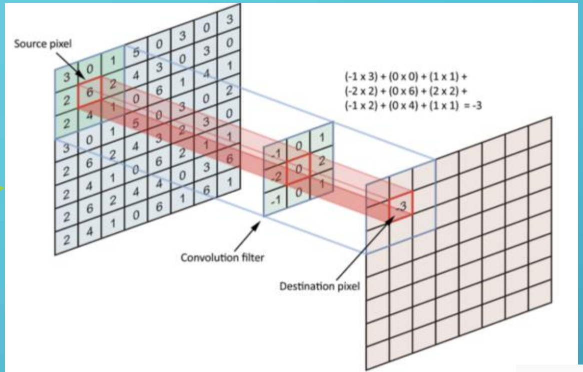


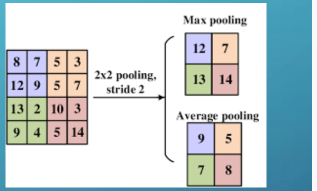


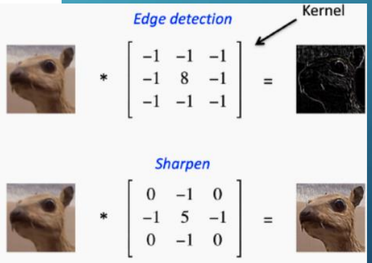
```
my name's john
You said: my name's john
my name is john
Hello, nice to meet you john
i live in canada
canada is great place to live in!
```

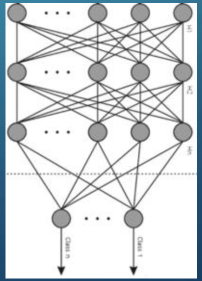
# CONVOLUTION NN (ALREADY DONE?)

- There are **three types** of layers:
  - convolutional layer (catches features),
  - pooling layer (reduce size),
  - fully connected layer (makes classification).


Source pixel
(-1 x 3) + (0 x 0) + (1 x 1) +
(-2 x 2) + (0 x 6) + (2 x 2) +
(-1 x 2) + (0 x 4) + (1 x 1) = -3
Convolution filter
Destination pixel







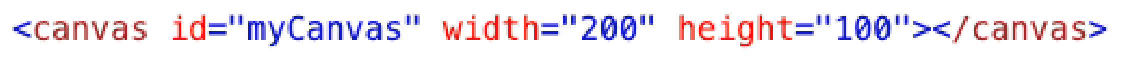
# USE CASE 1 – FACE DETECTION

- The most AI tasks (due to processing power requirements) are done on server side. However some tasks can be succesfully used on client side.
- The example is using:
  - Tensorflow library
  - HTML Video and Canvas tags.

# WHAT IS CANVAS

- It is element that allows to draw graphics on webpages

- Using WebGL it is posible to render 2D and 3D graphics in Canvas [Intro]

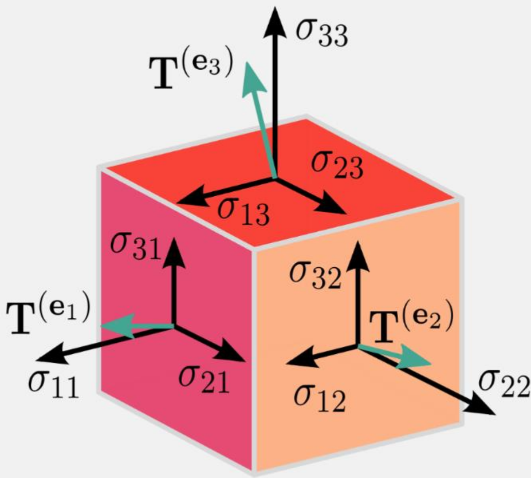


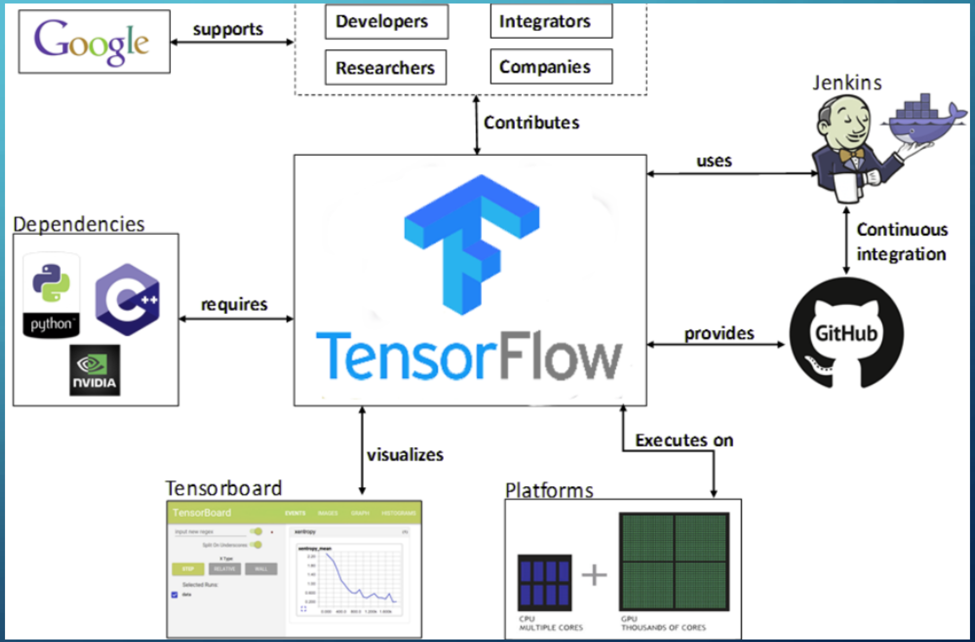
```
<canvas id="myCanvas" width="200" height="100"></canvas>
```

Online repository: https://www.w3schools.com/html/html5_canvas.asp

# TENSORFLOW

- It is a library to perform machine learning tasks. It is also useful to run already trained models in many languages. One of those languages are Java script used in browsers.
- From 2015, when it was introduced for Python language it become one of most commonly used AI framework.





Online repository: [https://www.tensorflow.org/js?hl=pl](https://www.tensorflow.org/js?hl=pl)

# BLAZEFACE MODEL

- One of lightweight model created in Tensorflow trained to detect face elements in realtime.
- This model can be used to detect based elements of human face.

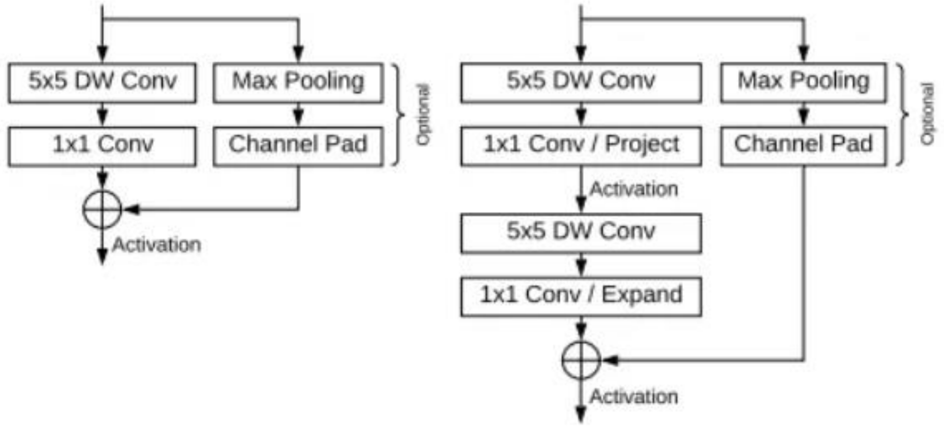


Figure 1. BlazeBlock (left) and double BlazeBlock

(Source: https://arxiv.org/abs/1907.05047)

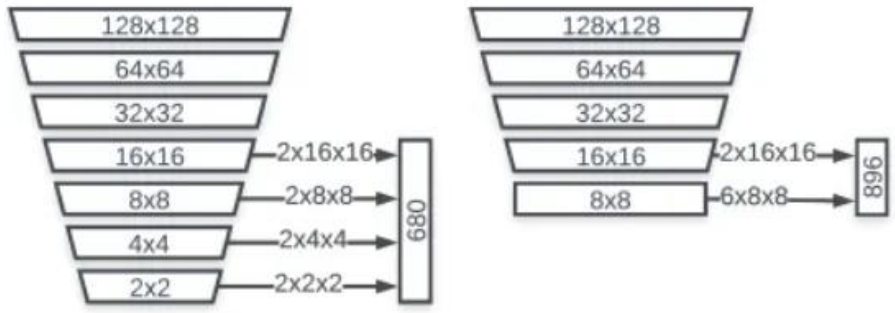


Figure 2. Anchor computation: SSD (left) vs. BlazeFace

(Source: https://arxiv.org/abs/1907.05047)

Online repository:
Face detection guide | MediaPipe | Google for Developers

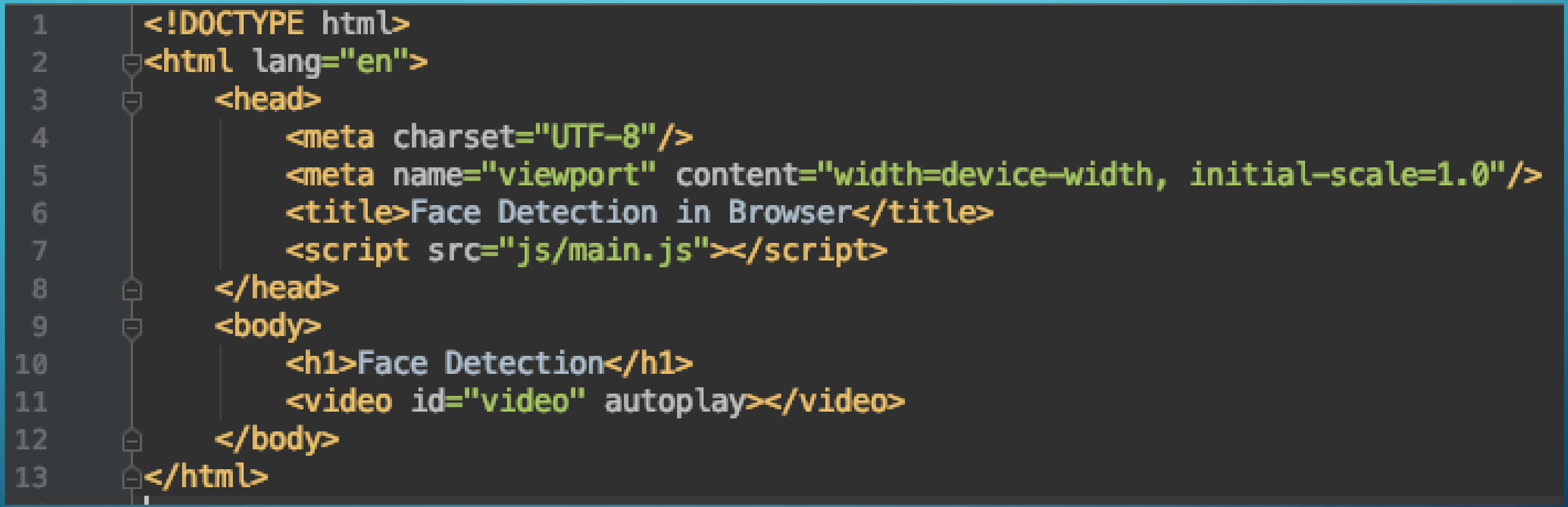
# IMPLEMENTATION OF A MODEL

- To use a model in the client application:
  - Obtain a data in application
  - Load AI model
  - Process the data
  - Fetch and proces the result

- The example will be presented using Javascript language

# APPLICATION

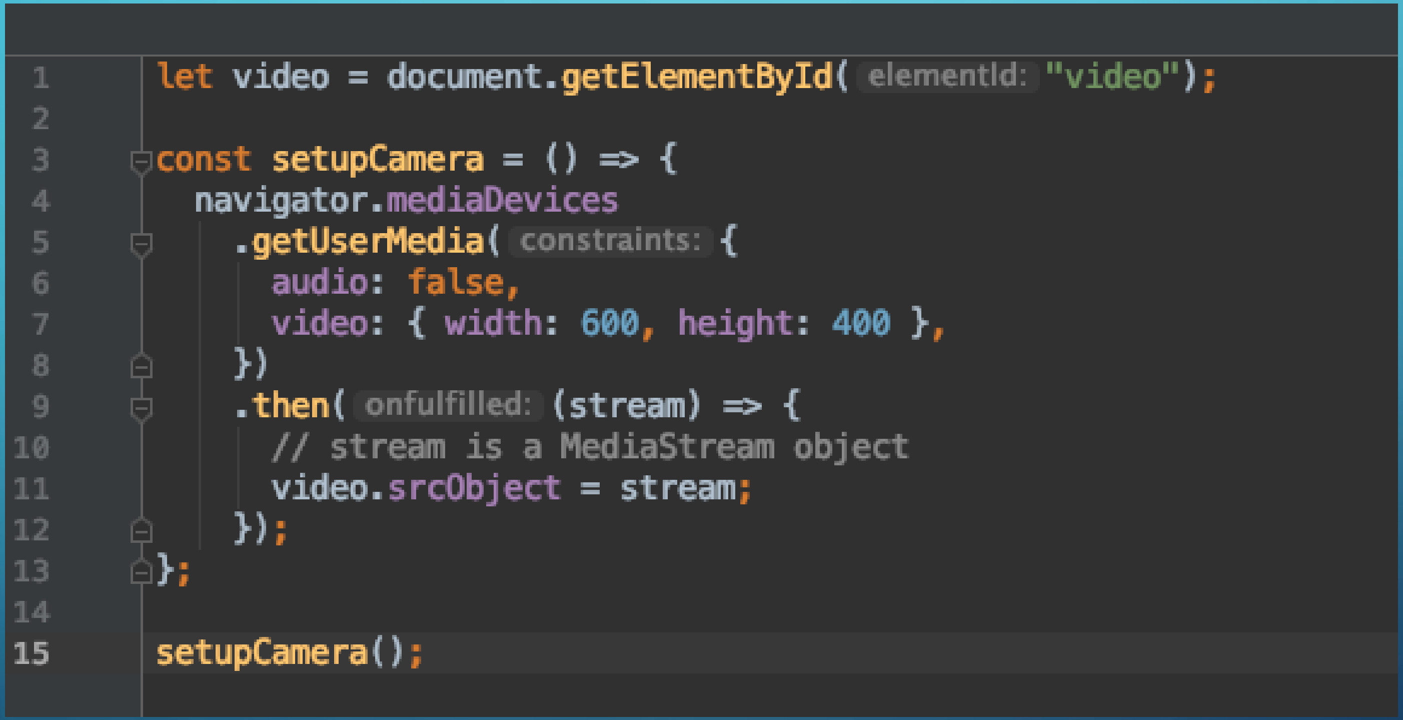
- The application is defined as Web page in HTML ver. 5



```html
<!DOCTYPE html>
<html lang="en">
    <head>
        <meta charset="UTF-8"/>
        <meta name="viewport" content="width=device-width, initial-scale=1.0"/>
        <title>Face Detection in Browser</title>
        <script src="js/main.js"></script>
    </head>
    <body>
        <h1>Face Detection</h1>
        <video id="video" autoplay></video>
    </body>
</html>
```

# FETCH DATA

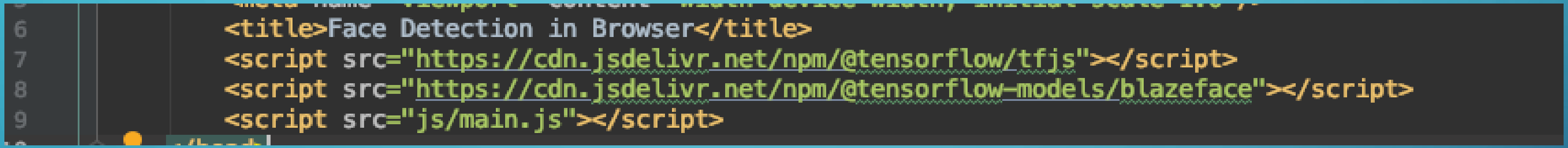
- Data is obtained from user. In this case is a video stream from camera.



```
let video = document.getElementById( elementId: "video");

const setupCamera = () => {
  navigator.mediaDevices
    .getUserMedia( constraints: {
      audio: false,
      video: { width: 600, height: 400 },
    })
    .then( onfulfilled: (stream) => {
      // stream is a MediaStream object
      video.srcObject = stream;
    });
};

setupCamera();
```

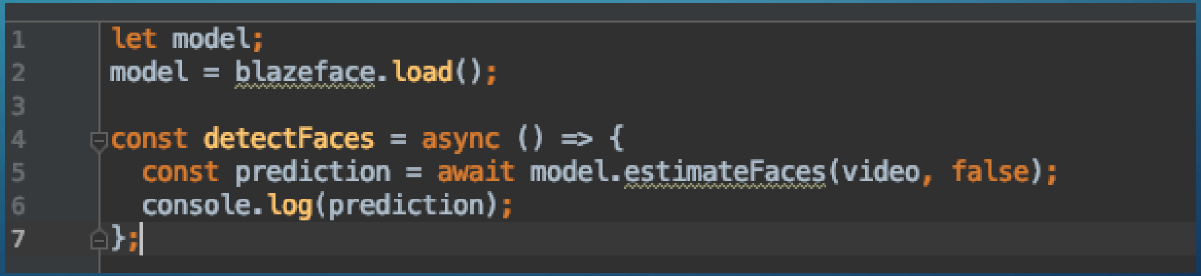
# TENSORFLOW ENVIRONMENT

- The tensorflow and blazeface is added as library to application



```
<title>Face Detection in Browser</title>
<script src="https://cdn.jsdelivr.net/npm/@tensorflow/tfjs"></script>
<script src="https://cdn.jsdelivr.net/npm/@tensorflow-models/blazeface"></script>
<script src="js/main.js"></script>
```

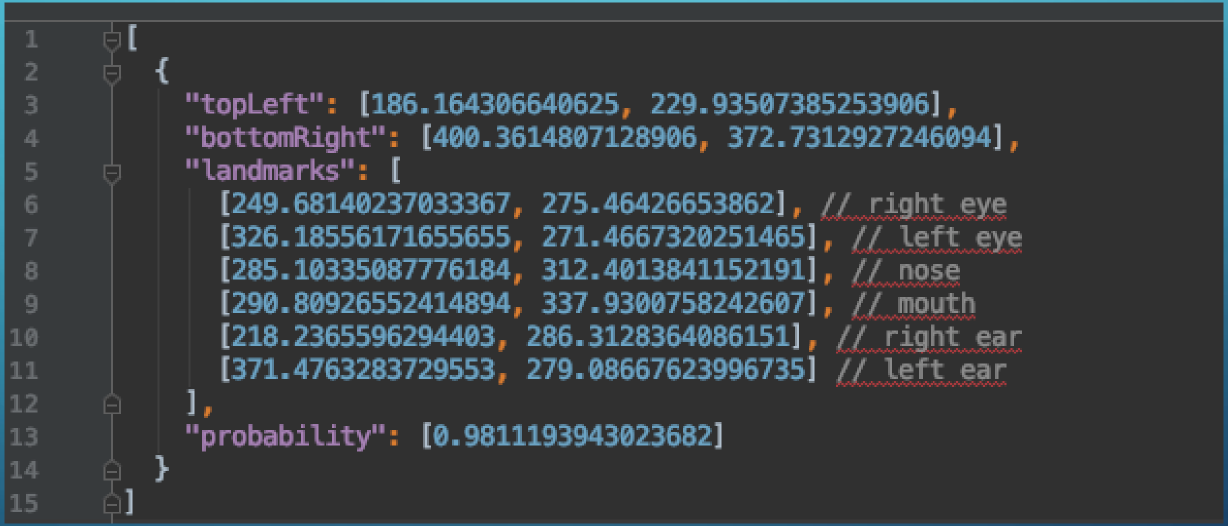
- Next step is to load a model and use it on data stream



```
let model;
model = blazeface.load();

const detectFaces = async () => {
  const prediction = await model.estimateFaces(video, false);
  console.log(prediction);
};
```

# MODEL RESULTS

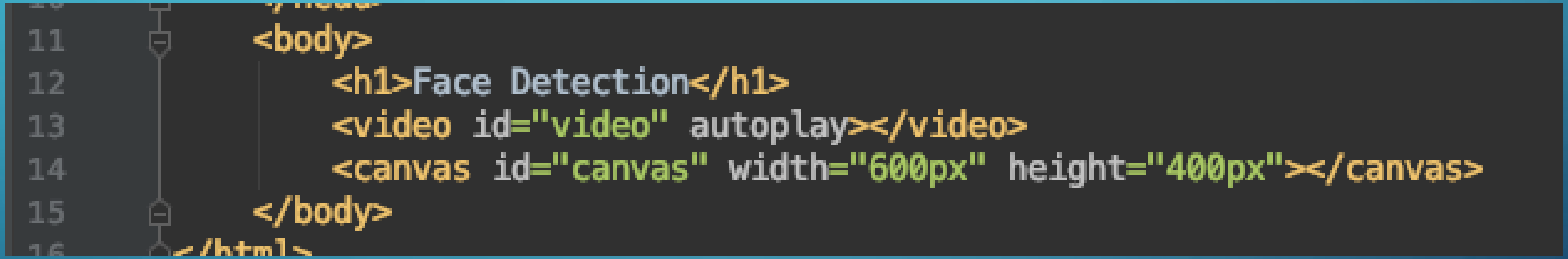
- The model results are generated as the JSON file, which describes the user face.
- File contain position of a face and its markers:



```
[
  {
    "topLeft": [186.164306640625, 229.93507385253906],
    "bottomRight": [400.3614807128906, 372.7312927246094],
    "landmarks": [
      [249.68140237033367, 275.46426653862], // right eye
      [326.18556171655655, 271.4667320251465], // left eye
      [285.10335087776184, 312.4013841152191], // nose
      [290.80926552414894, 337.9300758242607], // mouth
      [218.2365596294403, 286.3128364086151], // right ear
      [371.4763283729553, 279.08667623996735] // left ear
    ],
    "probability": [0.9811193943023682]
  }
]
```

# DATA UTILISATION

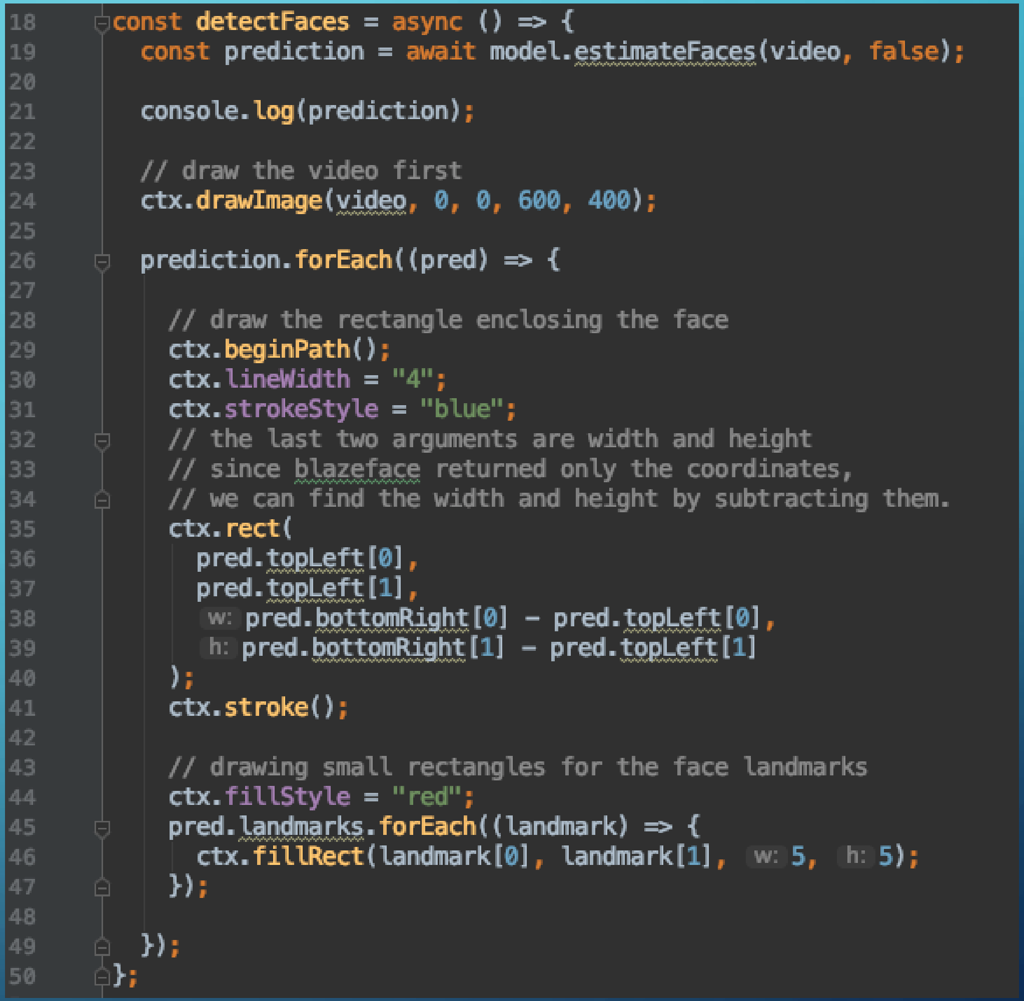
- The data can be used in user verification proces or to measure its Focus on the application.
- In this usecase we presets result in graphic form using Canvas.



```
    </head>
    <body>
        <h1>Face Detection</h1>
        <video id="video" autoplay></video>
        <canvas id="canvas" width="600px" height="400px"></canvas>
    </body>
</html>
```

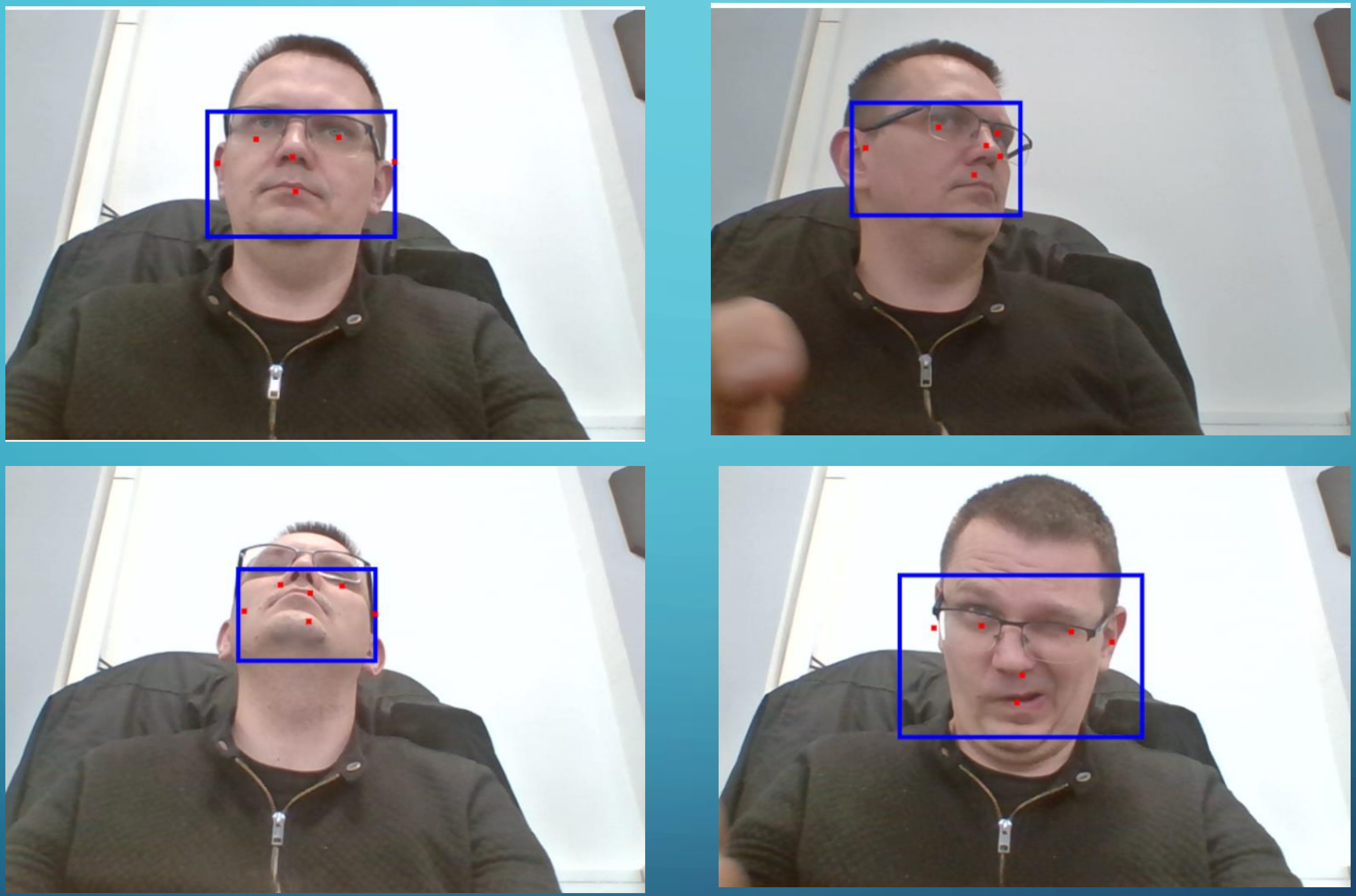
# DRAWING PROCEDURE

- The result is presented in graphical form
- All landmarks was presented as small red rectangles



```
const detectFaces = async () => {
  const prediction = await model.estimateFaces(video, false);

  console.log(prediction);

  // draw the video first
  ctx.drawImage(video, 0, 0, 600, 400);

  prediction.forEach((pred) => {

    // draw the rectangle enclosing the face
    ctx.beginPath();
    ctx.lineWidth = "4";
    ctx.strokeStyle = "blue";
    // the last two arguments are width and height
    // since blazeface returned only the coordinates,
    // we can find the width and height by subtracting them.
    ctx.rect(
      pred.topLeft[0],
      pred.topLeft[1],
      w: pred.bottomRight[0] - pred.topLeft[0],
      h: pred.bottomRight[1] - pred.topLeft[1]
    );
    ctx.stroke();

    // drawing small rectangles for the face landmarks
    ctx.fillStyle = "red";
    pred.landmarks.forEach((landmark) => {
      ctx.fillRect(landmark[0], landmark[1], w: 5, h: 5);
    });

  });
};
```

# EXAMPLE



Special thanks to Paweł Płoneczka IT&D Senior Product Manager in Reckitt complany who helps to create this usecase

# USE CASE 2

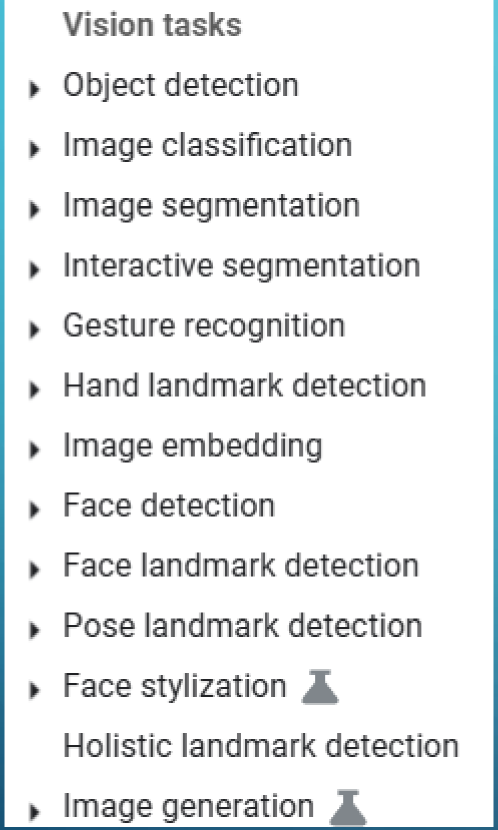
- The gesture tracking can find applications in many fields:
  - using immersive virtual reality
  - accesing user interface elements
  - in healthcare using precise movement monitoring
  - gaming while natural hand interaction
- In this example:
  - module aims to return position of hands and detect set of gestures
  - it will be used to open and close applications.

# MEDIAPIPE

- Solution simplifying deploying machine learning models on devices, making them production-ready and accessible across various platforms.
- The solution:
  - Contin lightweight models with high accuracy
  - Supports vision, text, and audio processing
  - Is accelerates on both CPU and GPU
  - Accessible on Android, iOS and web.

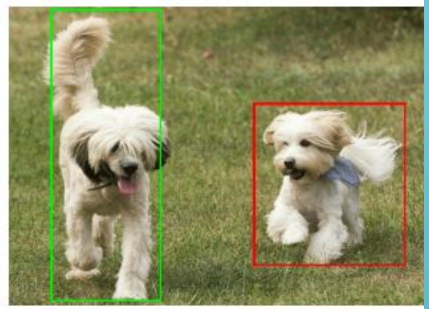
Online repository:
MediaPipe | Google for Developers

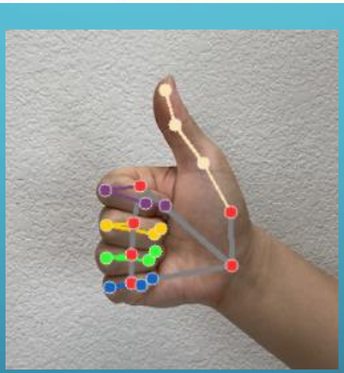
# MEDIAPIPE



**Vision tasks**

- Object detection
- Image classification
- Image segmentation
- Interactive segmentation
- Gesture recognition
- Hand landmark detection
- Image embedding
- Face detection
- Face landmark detection
- Pose landmark detection
- Face stylization
- Holistic landmark detection
- Image generation





[https://developers.google.com/mediapipe/solutions/vision]

# MODELS

- Mediapipe offers build in models for image, sound and text processing. In this case the hand tracking will be used thus following models will be implemented:
  - gesture_recognizer – model which tracks pretrained hand gesture – in total 8 gestures are suported
  - hand_landmarker – allows to describe hand as set of descriptors (including each hand bone)

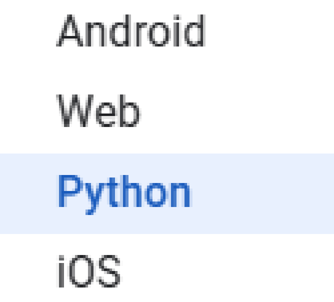
# IMPLEMENTATION OF A MODEL

- To use a AI module the following elements will be implemented:
  - Obtain a data in module
  - Load AI model
  - Process the data
  - Fetch and process the result

- The example will be presented using Python language

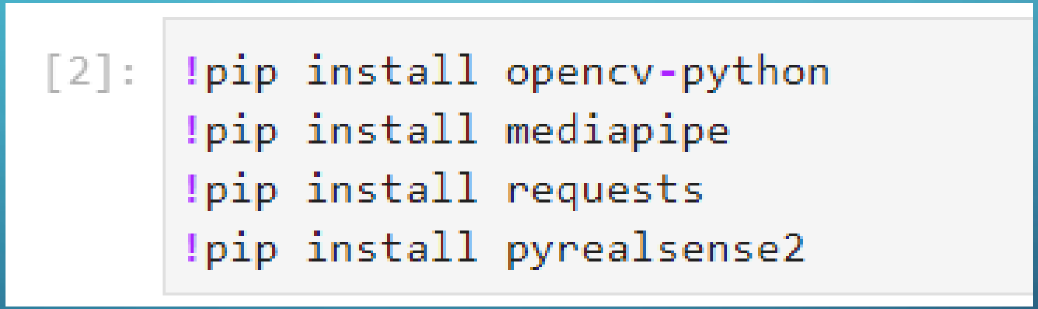
# ENVIRONMENT SETTINGS

- The script can be applied in environment, where the access to camera and application is possible
- Therefore, the example can be execute using Anaconda environment (access to local computer processes).
- The processing algorithm can also work on servers solutions.



# IMPORTING LIBRARIES

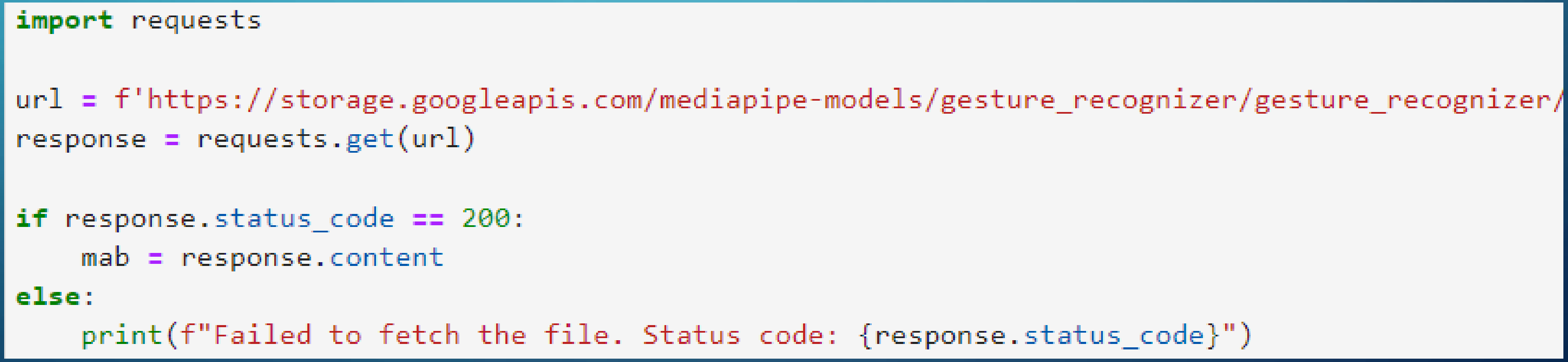
- The usecase is build based on mediapipe library. However the picture manipulation is based on opencv library. Additionally, suporting libraries were used.



```
[2]: !pip install opencv-python
     !pip install mediapipe
     !pip install requests
     !pip install pyrealsense2
```

# AI MODELS

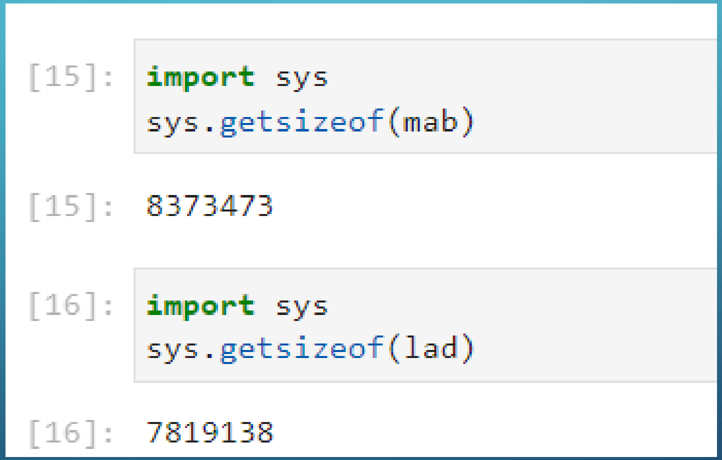
- The models for mediapipe are created as routines or tasks that can be easily used after configuration.
- The models in example are downloaded directly from Google server.



```
import requests

url = f'https://storage.googleapis.com/mediapipe-models/gesture_recognizer/gesture_recognizer/
response = requests.get(url)

if response.status_code == 200:
    mab = response.content
else:
    print(f"Failed to fetch the file. Status code: {response.status_code}")
```

# MODEL VERIFICATION

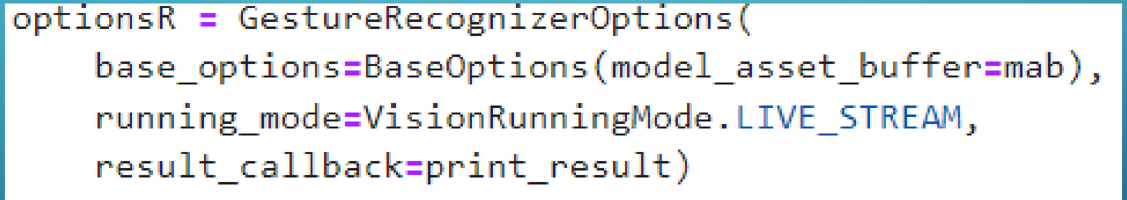
- The should be verify by checking its size of check sum
- Due to used download method the size verification is sufficient



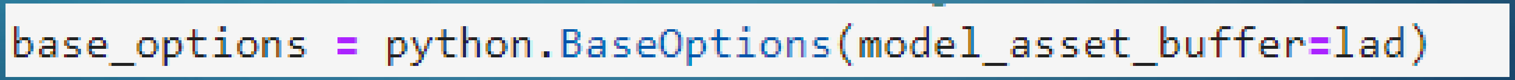
```
[15]: import sys
      sys.getsizeof(mab)

[15]: 8373473

[16]: import sys
      sys.getsizeof(lad)

[16]: 7819138
```

# SUPORT FUNCTIONS

- To create the recognizer using mediapipe the model should be configured.
- The configuration is made by seting the options.



```
optionsR = GestureRecognizerOptions(
    base_options=BaseOptions(model_asset_buffer=mab),
    running_mode=VisionRunningMode.LIVE_STREAM,
    result_callback=print_result)
```



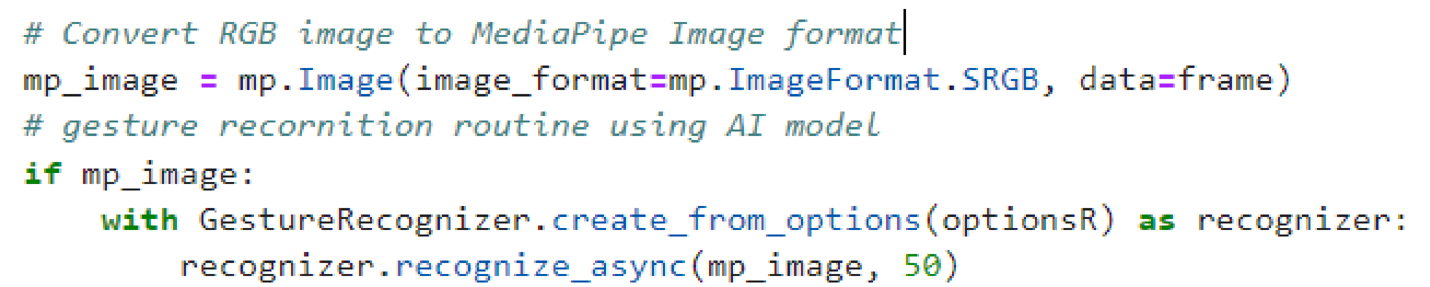
```
base_options = python.BaseOptions(model_asset_buffer=lad)
```

# CREATING AI MODULE

- AI module follows the simple routine for each image obtained from camera:
  - Send image to hand recognition routine
  - Send image to gesture recognition routine
  - Obtain results from routines and display it as image

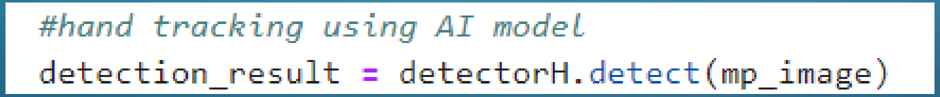
# PROCESING USING AI MODELS

- Gesture recognition:



```
# Convert RGB image to MediaPipe Image format
mp_image = mp.Image(image_format=mp.ImageFormat.SRGB, data=frame)
# gesture recornition routine using AI model
if mp_image:
    with GestureRecognizer.create_from_options(optionsR) as recognizer:
        recognizer.recognize_async(mp_image, 50)
```

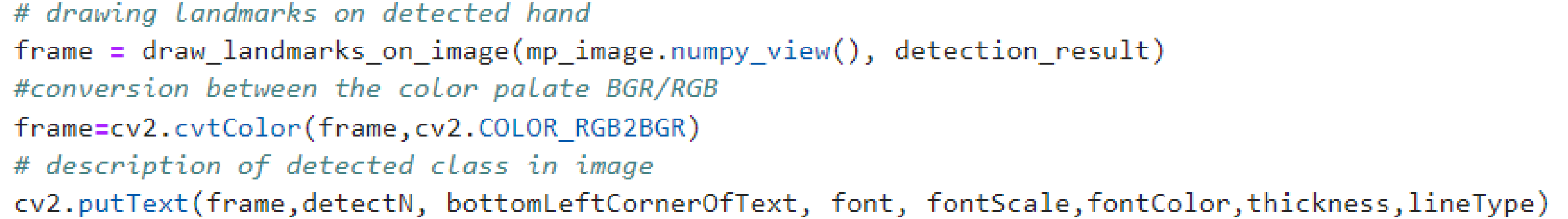
- Hand tracking:



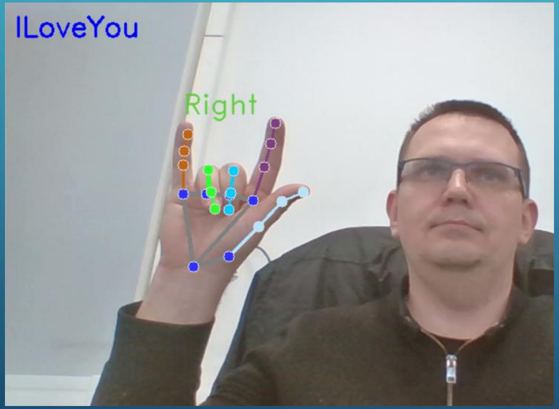
```
#hand tracking using AI model
detection_result = detectorH.detect(mp_image)
```

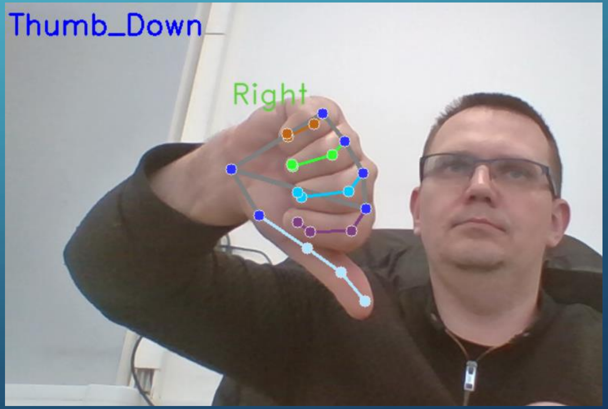
# PROCESS A RESULTS

- The result as a detection class and set of landmarks can be displayed as an image:



```
# drawing landmarks on detected hand
frame = draw_landmarks_on_image(mp_image.numpy_view(), detection_result)
#conversion between the color palate BGR/RGB
frame=cv2.cvtColor(frame,cv2.COLOR_RGB2BGR)
# description of detected class in image
cv2.putText(frame,detectN, bottomLeftCornerOfText, font, fontScale,fontColor,thickness,lineType)
```





# USE CASE 3 – BUILD OWN MODEL

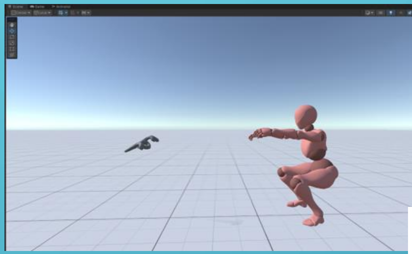
- Virtual Reality (VR) is a technology that creates three-dimensional virtual environments for users to explore and interact with objects using special devices like VR goggles and controllers.
- It finds applications in various fields such as military and civilian pilot training, surgical simulation, and education and entertainment.
- VR offers increased immersion by engaging users physically, enhancing their sense of presence in the virtual environment.
- Most VR systems focus primarily on tracking hands and head movements, neglecting lower limb tracking.
- Current systems supporting these features often come with a high price tag.

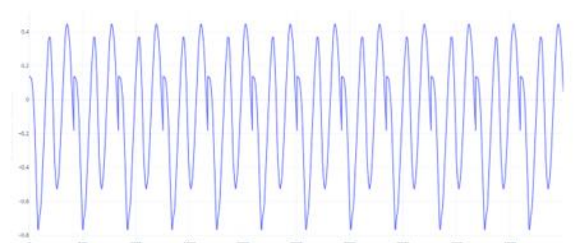
# LEG TRACKING FOR VR

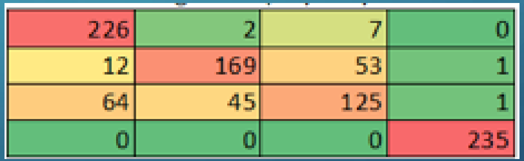
- To enhance game immersion, additional sensors and detectors can be used, but acquiring them entails extra costs.
- This use case proposes utilizing budget smartphones as sensors during gameplay.

# SENSOR TRACKING

- Attach sensors
- Create routine
- Generate results
- Analyse results



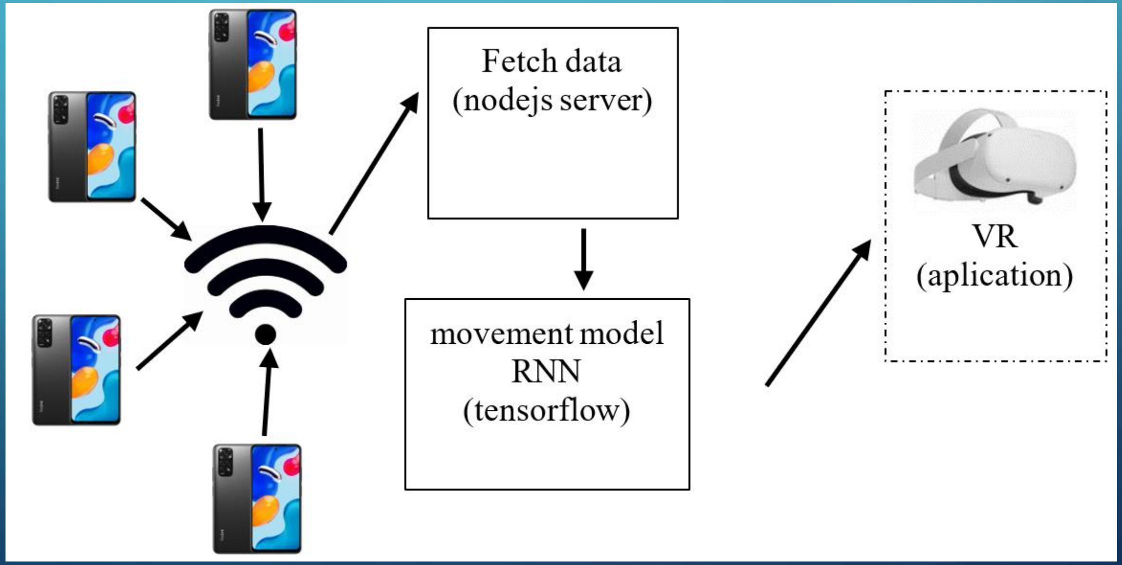




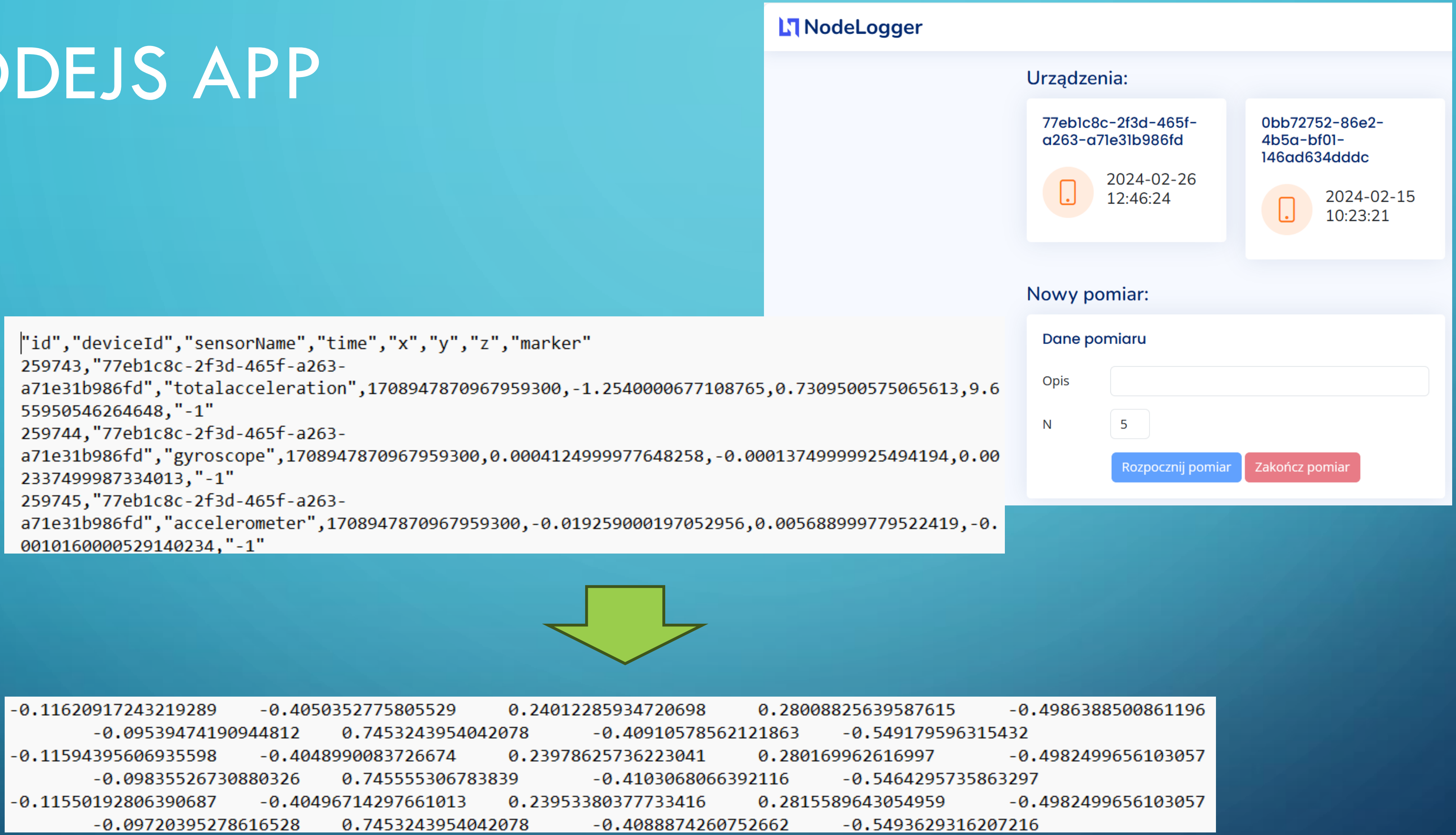
| | | | |
|---|---|---|---|
| 226 | 2 | 7 | 0 |
| 12 | 169 | 53 | 1 |
| 64 | 45 | 125 | 1 |
| 0 | 0 | 0 | 235 |

# OBTAINING DATA

- The solution is using smartphones as sensors for monitoring limb motion.
- data packets are collected upon application launch and transmitted to a destination for synchronization.



# NODEJS APP



**NodeLogger**

Urządzenia:

77eb1c8c-2f3d-465f-a263-a71e31b986fd
2024-02-26 12:46:24

0bb72752-86e2-4b5a-bf01-146ad634dddc
2024-02-15 10:23:21

Nowy pomiar:

**Dane pomiaru**

Opis:

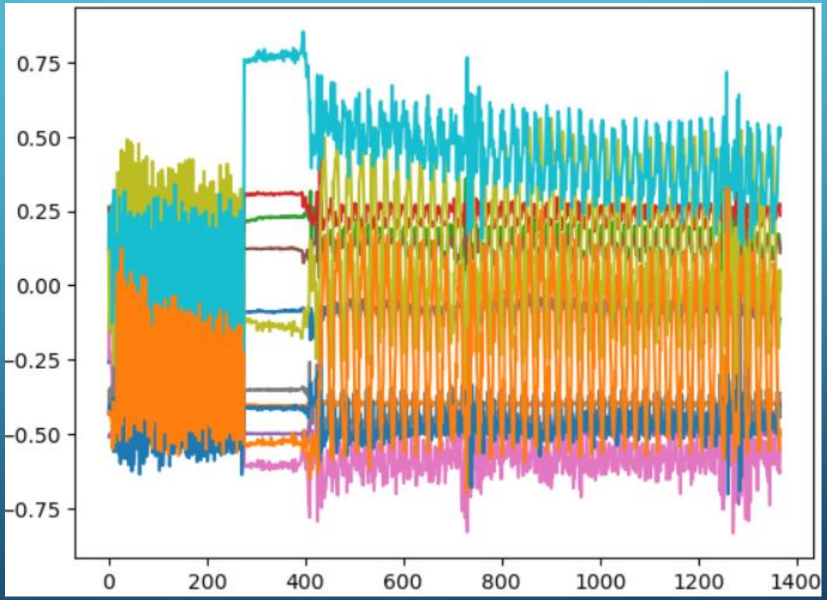
N: 5

```
"id","deviceId","sensorName","time","x","y","z","marker"
259743,"77eb1c8c-2f3d-465f-a263-
a71e31b986fd","totalacceleration",1708947870967959300,-1.2540000677108765,0.7309500575065613,9.6
55950546264648,"-1"
259744,"77eb1c8c-2f3d-465f-a263-
a71e31b986fd","gyroscope",1708947870967959300,0.0004124999977648258,-0.00013749999925494194,0.00
2337499987334013,"-1"
259745,"77eb1c8c-2f3d-465f-a263-
a71e31b986fd","accelerometer",1708947870967959300,-0.019259000197052956,0.005688999779522419,-0.
0010160000529140234,"-1"
```

```
-0.11620917243219289    -0.4050352775805529     0.24012285934720698     0.28008825639587615     -0.4986388500861196
        -0.09539474190944812    0.7453243954042078      -0.40910578562121863    -0.549179596315432
-0.11594395606935598    -0.4048990083726674     0.23978625736223041     0.280169962616997       -0.4982499656103057
        -0.09835526730880326    0.745555306783839       -0.4103068066392116     -0.5464295735863297
-0.11550192806390687    -0.40496714297661013    0.23953380377733416     0.2815589643054959      -0.4982499656103057
        -0.09720395278616528    0.7453243954042078      -0.4088874260752662     -0.5493629316207216
```

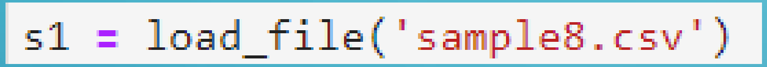
# DATA PREPROCESSING

- Data from smartphone sensors is sampled at frequencies, with 10 Hz being adequate for basic activities and 20 Hz for complex movements. The time-series data is then sent to the inference module.

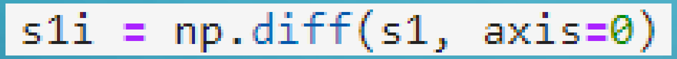


# FORMATING DATA FOR TENSORFLOW

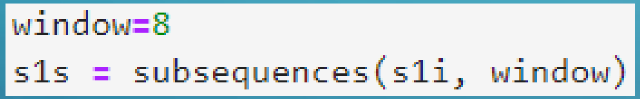
- The data has to be formatted for tensor flow library. The require training data as sequence of samples for each attribute. If model is sequential the each sample has time window.



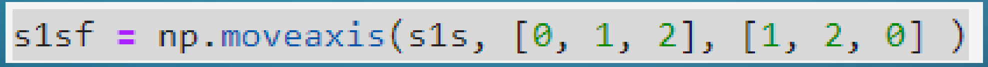
```
s1 = load_file('sample8.csv')
```



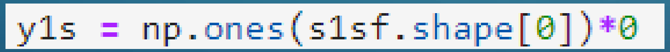
```
s1i = np.diff(s1, axis=0)
```



```
window=8
s1s = subsequences(s1i, window)
```



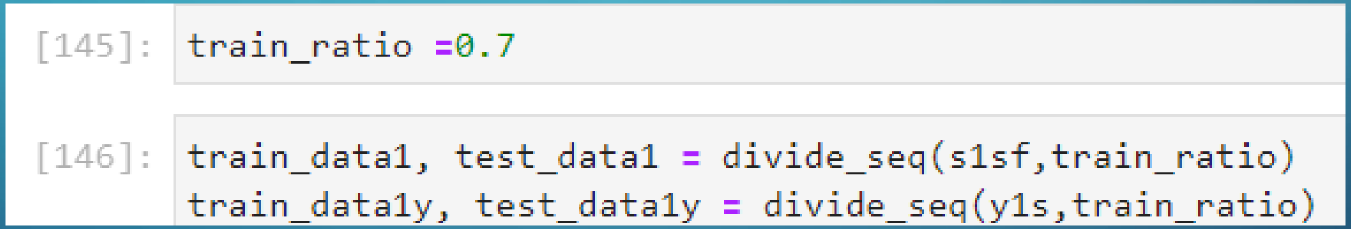
```
s1sf = np.moveaxis(s1s, [0, 1, 2], [1, 2, 0] )
```



```
y1s = np.ones(s1sf.shape[0])*0
```

# TRAINING / VERIFICATION SET

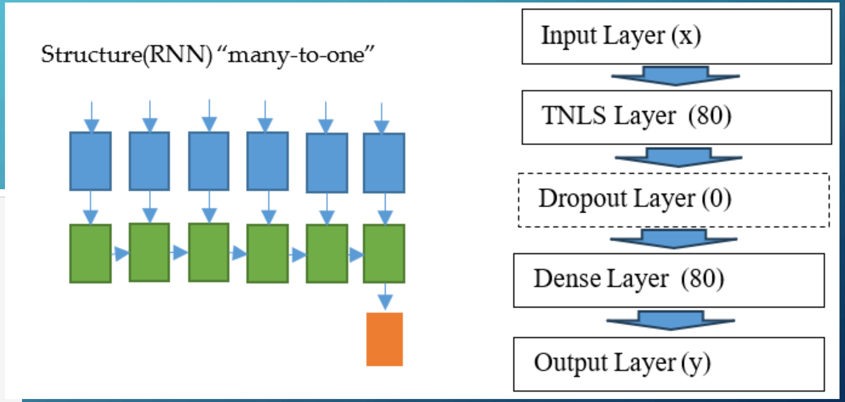
- The data is divided into two subsets for training and verification.
- The training set should be significantly bigger than verification set

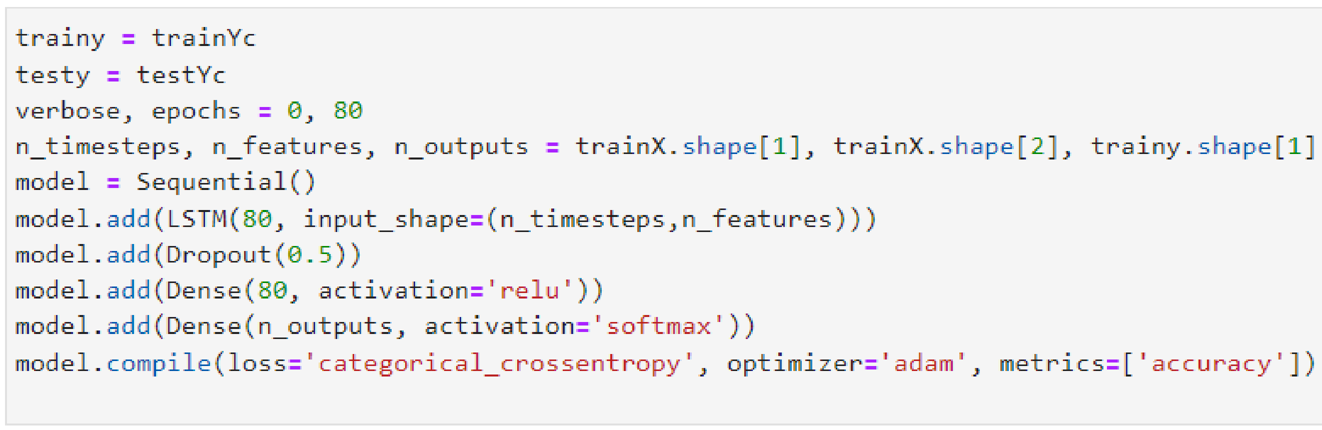


```
[145]: train_ratio =0.7

[146]: train_data1, test_data1 = divide_seq(s1sf,train_ratio)
       train_data1y, test_data1y = divide_seq(y1s,train_ratio)
```

# RNN LAYER

- The model adjusting for input sequence and class count while reducing neurons for speed, with epochs tuned via learning curve analysis and dropout layers to prevent overfitting.

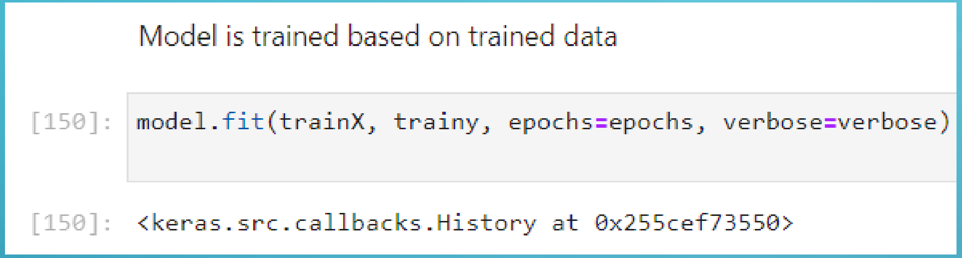




```
trainy = trainYc
testy = testYc
verbose, epochs = 0, 80
n_timesteps, n_features, n_outputs = trainX.shape[1], trainX.shape[2], trainy.shape[1]
model = Sequential()
model.add(LSTM(80, input_shape=(n_timesteps,n_features)))
model.add(Dropout(0.5))
model.add(Dense(80, activation='relu'))
model.add(Dense(n_outputs, activation='softmax'))
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
```

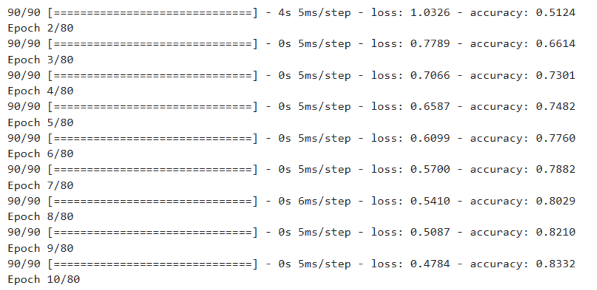
# MODEL TRAINING AND VERIFICATION

- Training of model is made by feeding it with data



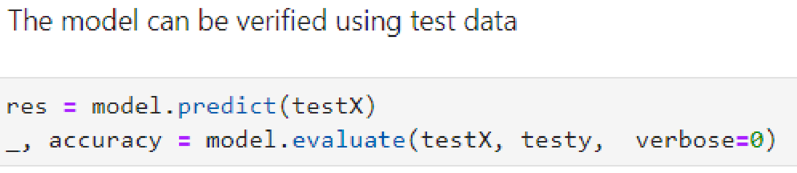
Model is trained based on trained data

```
[150]: model.fit(trainX, trainy, epochs=epochs, verbose=verbose)

[150]: <keras.src.callbacks.History at 0x255cef73550>
```



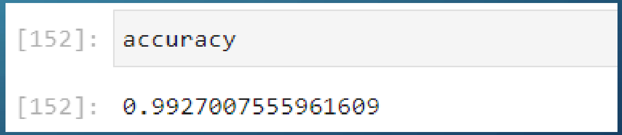
```
90/90 [==============================] - 4s 5ms/step - loss: 1.0326 - accuracy: 0.5124
Epoch 2/80
90/90 [==============================] - 0s 5ms/step - loss: 0.7789 - accuracy: 0.6614
Epoch 3/80
90/90 [==============================] - 0s 5ms/step - loss: 0.7066 - accuracy: 0.7301
Epoch 4/80
90/90 [==============================] - 0s 5ms/step - loss: 0.6587 - accuracy: 0.7482
Epoch 5/80
90/90 [==============================] - 0s 5ms/step - loss: 0.6099 - accuracy: 0.7760
Epoch 6/80
90/90 [==============================] - 0s 5ms/step - loss: 0.5700 - accuracy: 0.7882
Epoch 7/80
90/90 [==============================] - 0s 6ms/step - loss: 0.5410 - accuracy: 0.8029
Epoch 8/80
90/90 [==============================] - 0s 5ms/step - loss: 0.5087 - accuracy: 0.8210
Epoch 9/80
90/90 [==============================] - 0s 5ms/step - loss: 0.4784 - accuracy: 0.8332
Epoch 10/80
90/90 [==============================] - 0s 5ms/step - loss: 0.4701 - accuracy: 0.8394
```

- Verification of model is made be making prediction with test data and checking result



The model can be verified using test data

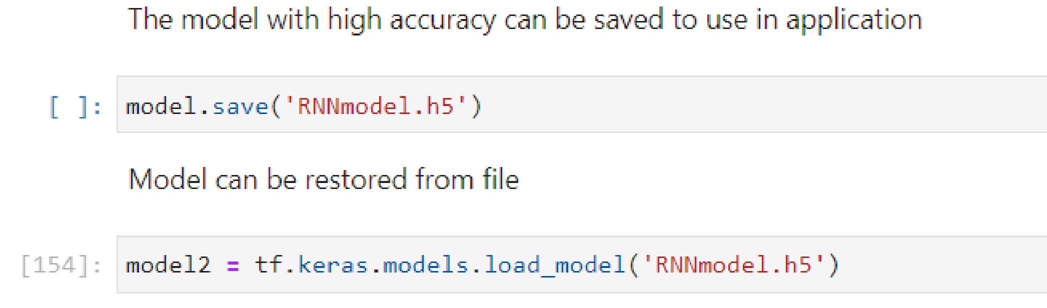
```
res = model.predict(testX)
_, accuracy = model.evaluate(testX, testy,  verbose=0)
```



```
[152]: accuracy

[152]: 0.9927007555961609
```

# SAVING / RESTORING MODELS

- Created models can be saved / restored to used in other applications
- The tuning process allows to look for better models with all data or part of data



The model with high accuracy can be saved to use in application

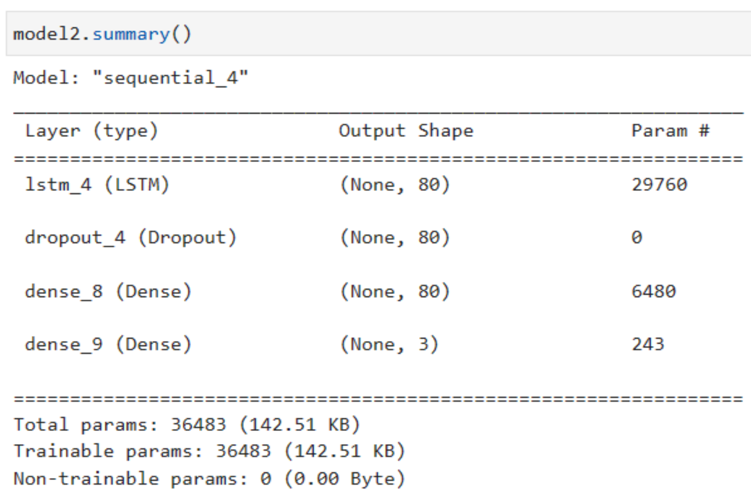
```
[ ]: model.save('RNNmodel.h5')
```

Model can be restored from file

```
[154]: model2 = tf.keras.models.load_model('RNNmodel.h5')
```

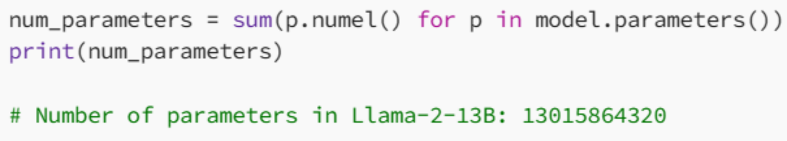
# SIZE OF MODEL

- The size of model determine its training time and execution time
- fit time >> prediction time



```
model2.summary()
Model: "sequential_4"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm_4 (LSTM)               (None, 80)                29760

 dropout_4 (Dropout)         (None, 80)                0

 dense_8 (Dense)             (None, 80)                6480

 dense_9 (Dense)             (None, 3)                 243

=================================================================
Total params: 36483 (142.51 KB)
Trainable params: 36483 (142.51 KB)
Non-trainable params: 0 (0.00 Byte)
```

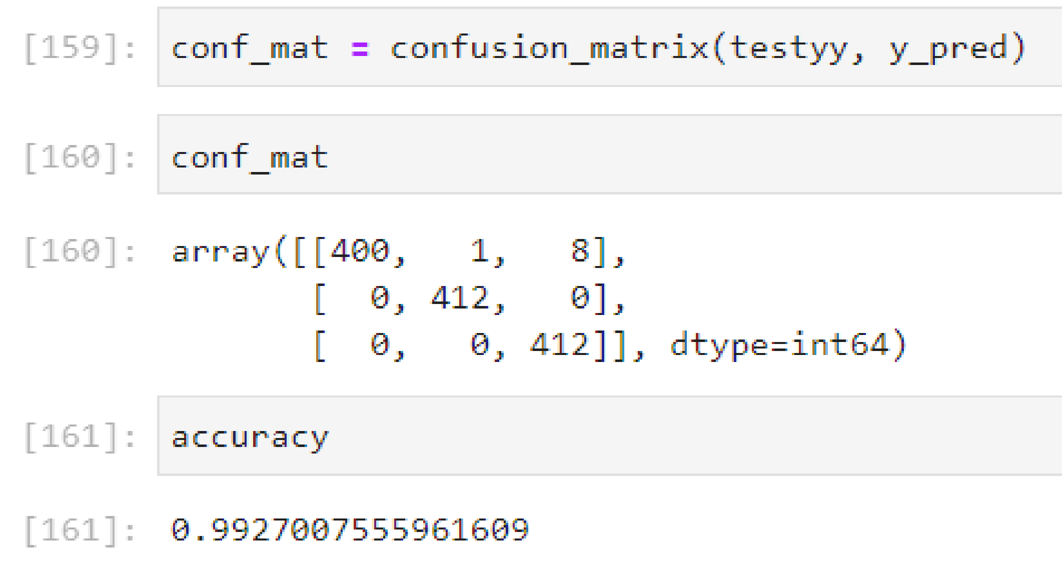
```
Yolo3 -image classification
Total params: 62001757 (236.52 MB)
Trainable params: 61949149 (236.32 MB)
Non-trainable params: 52608 (205.50 KB)
```



```
num_parameters = sum(p.numel() for p in model.parameters())
print(num_parameters)

# Number of parameters in Llama-2-13B: 13015864320
```

# MULTIPLE CLASS CLASSIFICATION

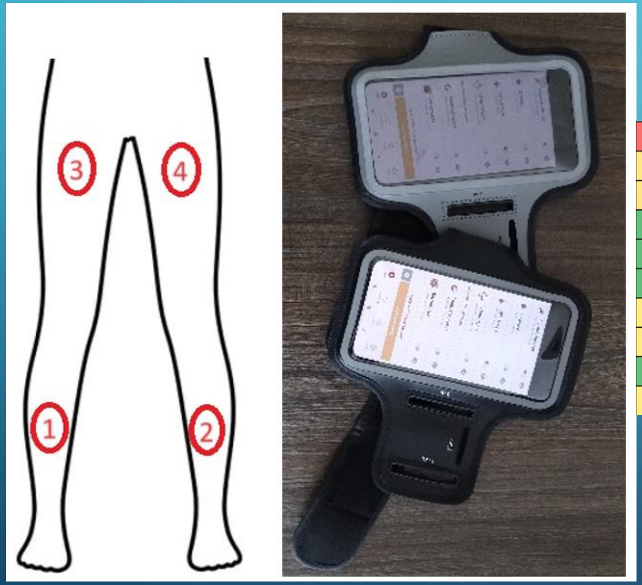
- In case of multiple class classification the analysis of each class results can be useful to find class at which a additional data are needed

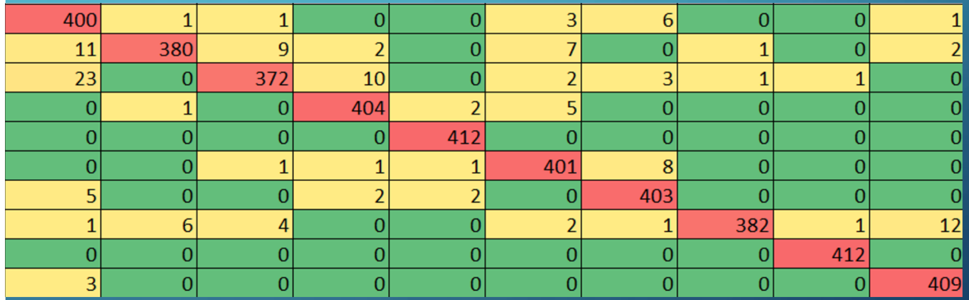


```
[159]:  conf_mat = confusion_matrix(testyy, y_pred)

[160]:  conf_mat

[160]:  array([[400,   1,   8],
               [  0, 412,   0],
               [  0,   0, 412]], dtype=int64)

[161]:  accuracy

[161]:  0.9927007555961609
```

# FINAL EXPERIMENTS AND RESULTS

- solution enables activity tracking with 99% accuracy and player activity detection for ten activity classes within 0.5s.
- detection time to 0.2 seconds by sacrificing detection accuracy to 95% through a window reduction to 5.





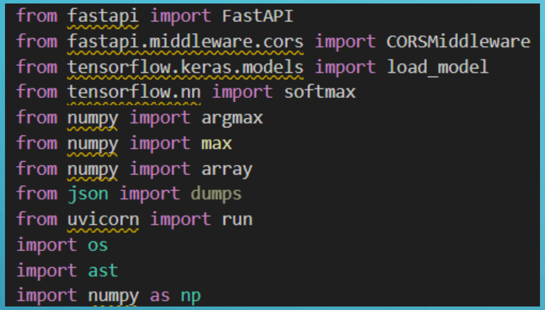
| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 400 | 1 | 1 | 0 | 0 | 3 | 6 | 0 | 0 | 1 |
| 11 | 380 | 9 | 2 | 0 | 7 | 0 | 1 | 0 | 2 |
| 23 | 0 | 372 | 10 | 0 | 2 | 3 | 1 | 1 | 0 |
| 0 | 1 | 0 | 404 | 2 | 5 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 412 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 1 | 401 | 8 | 0 | 0 | 0 |
| 5 | 0 | 0 | 2 | 2 | 0 | 403 | 0 | 0 | 0 |
| 1 | 6 | 4 | 0 | 0 | 2 | 1 | 382 | 1 | 12 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 412 | 0 |
| 3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 409 |

# CREATING EXTERNAL MODULE

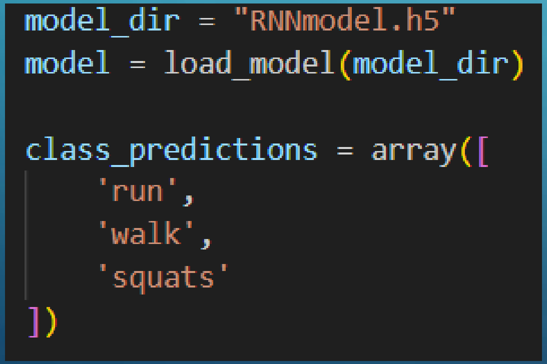
- The AI model can be integrateg with the external module
- The module in case of big model can be hosted by service
- The service interface should be provided e.g. API

# API IMPLEMENTATION

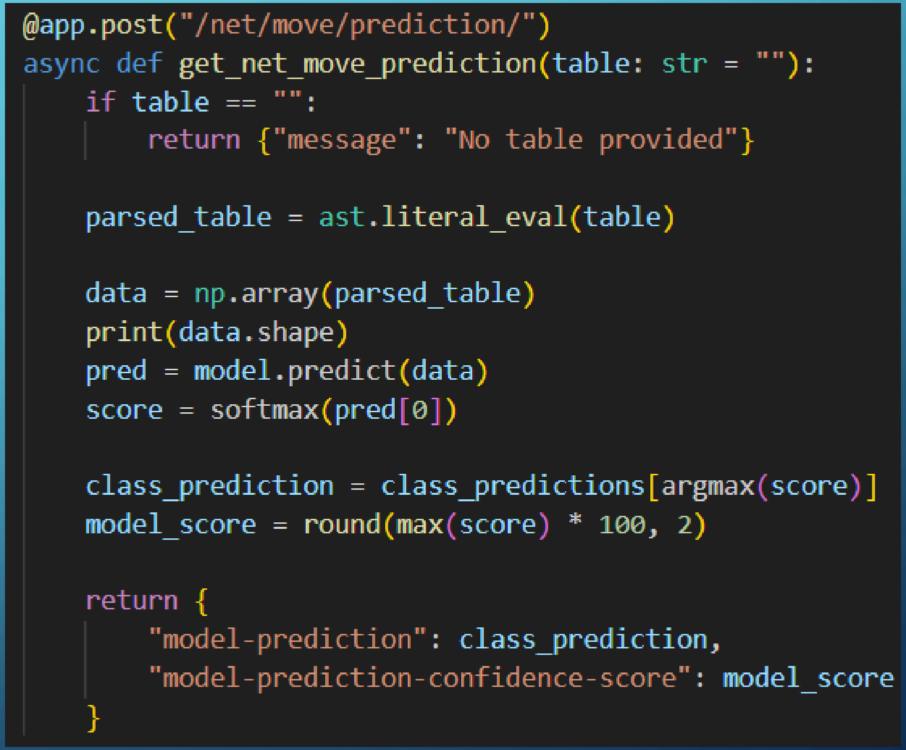
- The fastapi + tensorflow library



```python
from fastapi import FastAPI
from fastapi.middleware.cors import CORSMiddleware
from tensorflow.keras.models import load_model
from tensorflow.nn import softmax
from numpy import argmax
from numpy import max
from numpy import array
from json import dumps
from uvicorn import run
import os
import ast
import numpy as np
```

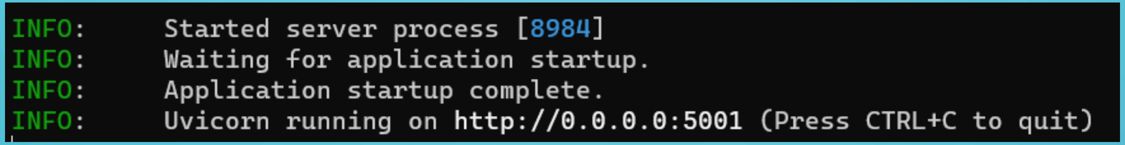


```python
model_dir = "RNNmodel.h5"
model = load_model(model_dir)

class_predictions = array([
    'run',
    'walk',
    'squats'
])
```

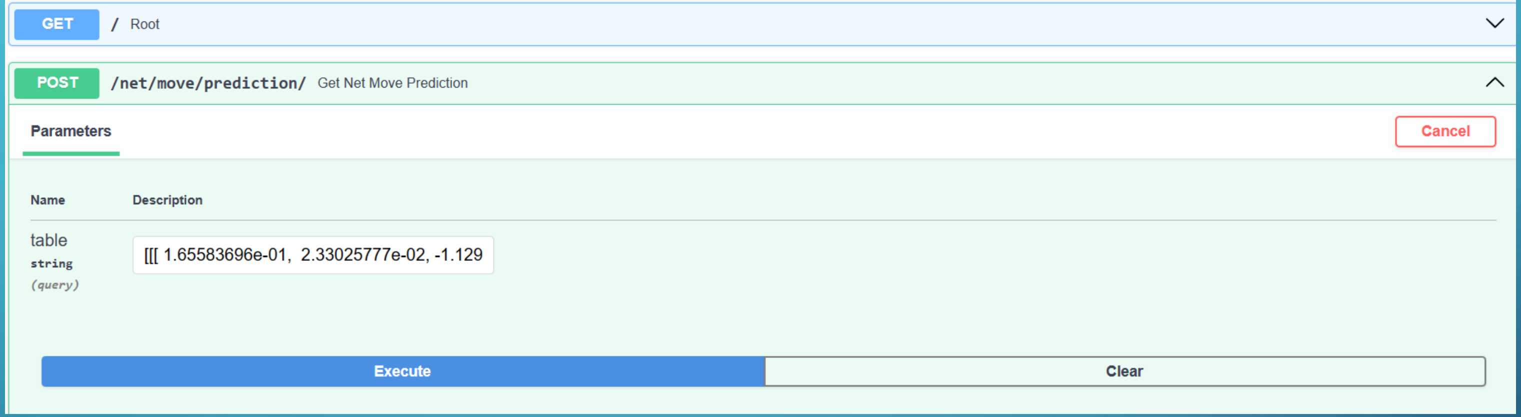


```python
@app.post("/net/move/prediction/")
async def get_net_move_prediction(table: str = ""):
    if table == "":
        return {"message": "No table provided"}

    parsed_table = ast.literal_eval(table)

    data = np.array(parsed_table)
    print(data.shape)
    pred = model.predict(data)
    score = softmax(pred[0])

    class_prediction = class_predictions[argmax(score)]
    model_score = round(max(score) * 100, 2)

    return {
        "model-prediction": class_prediction,
        "model-prediction-confidence-score": model_score
    }
```

# API EXAMPLE

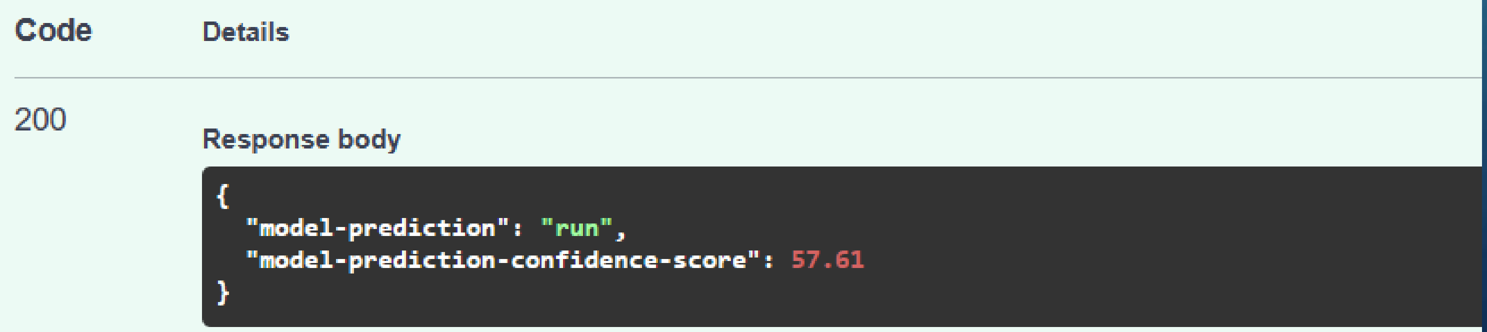


```
INFO:     Started server process [8984]
INFO:     Waiting for application startup.
INFO:     Application startup complete.
INFO:     Uvicorn running on http://0.0.0.0:5001 (Press CTRL+C to quit)
```



`/net/move/prediction/`

| Name | Description |
|---|---|
| table<br>string<br>*(query)* | [[[ 1.65583696e-01,  2.33025777e-02, -1.129 |

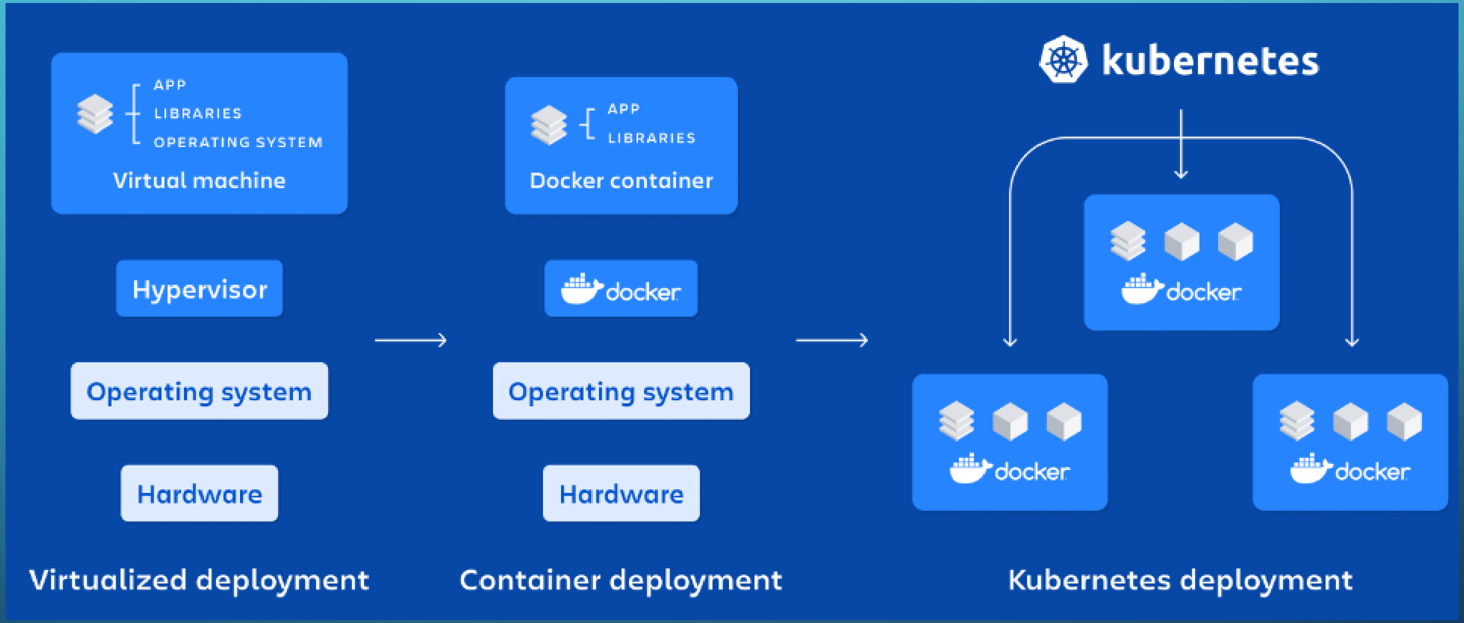


| Code | Details |
|---|---|
| 200 | Response body |

```
{
  "model-prediction": "run",
  "model-prediction-confidence-score": 57.61
}
```

# NEXT STEP

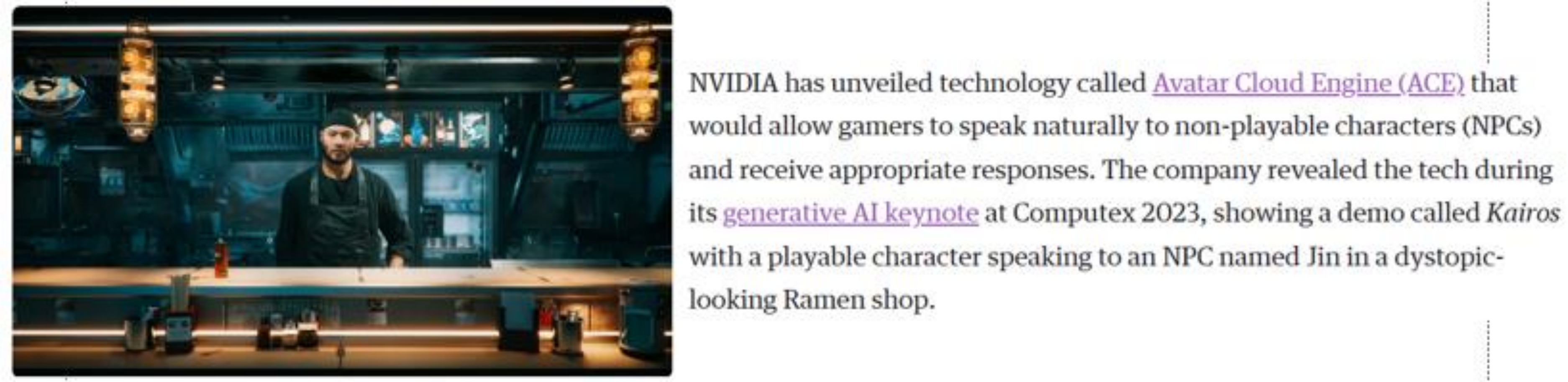
- Creating image using docker
- Load balancing using Kubernetes



[ https://www.atlassian.com/microservices/microservices-architecture/kubernetes-vs-docker ]

# USE CASE 4 – AI BASED NPC

- Using AI it is easier than ever to create the bots that can gather data, respond to our needs and join into conversation. This use case present the implementation of interactive chatbot:
    - Obtain voice input and convert it to text
    - Analyse text and produce response
    - The response is converted to voice



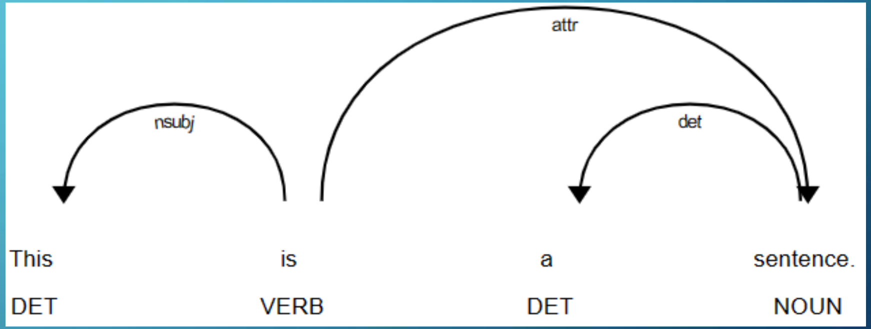
NVIDIA has unveiled technology called Avatar Cloud Engine (ACE) that would allow gamers to speak naturally to non-playable characters (NPCs) and receive appropriate responses. The company revealed the tech during its generative AI keynote at Computex 2023, showing a demo called *Kairos* with a playable character speaking to an NPC named Jin in a dystopic-looking Ramen shop.

# LIBRARIES

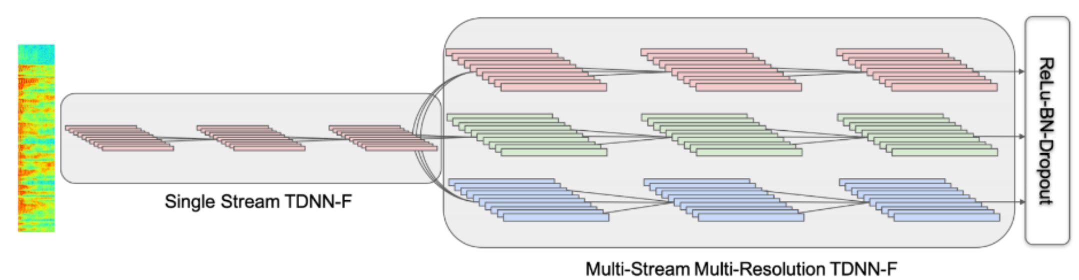
- To create an assistant three AI models will be used for an application. The following libraries were selected:
  - vosk - speech recognition library
  - spacy - natural language processing library
  - pyttsx3 - speak model for python
- The libraries was selected based on open policy and python language support

# MODELS

- Each library has various pretrained models, which can be used.
  - pyttsx3 models are build in library ().
  - vosk models have to be download manually ().
  - Spacy models have build-in downloader.



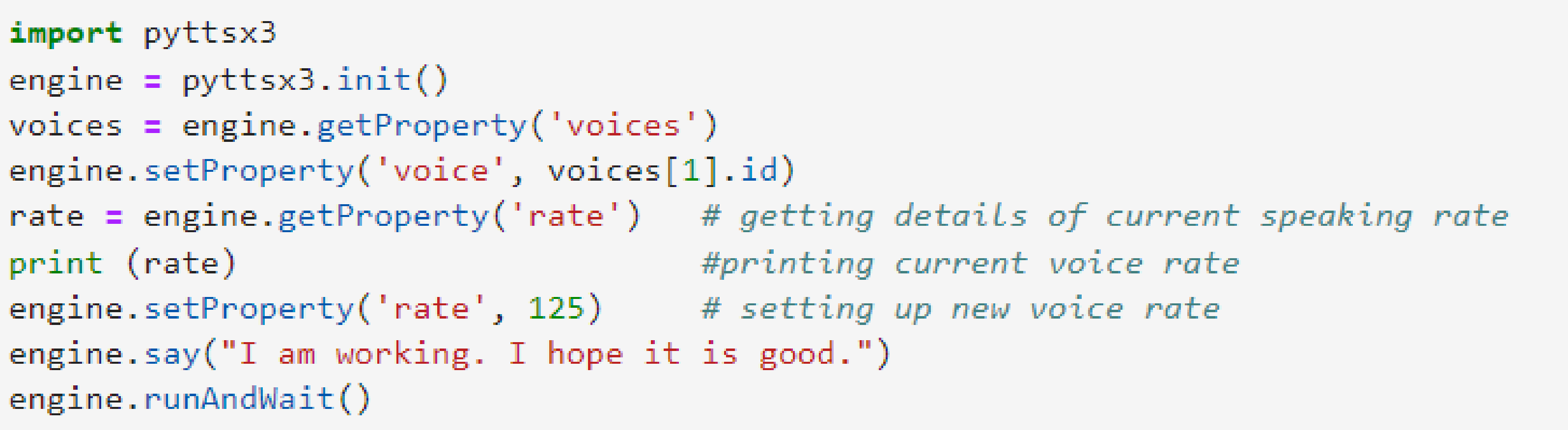
https://spacy.io/



[Multistream TDNN and new Vosk model | Speech Recognition With Vosk (alphacephei.com)]

# BUILD-IN MODELS

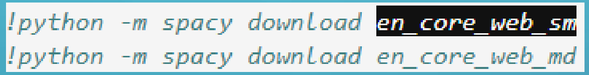
- Build in models are integrated with library. We can select a type of model used or its parameters.



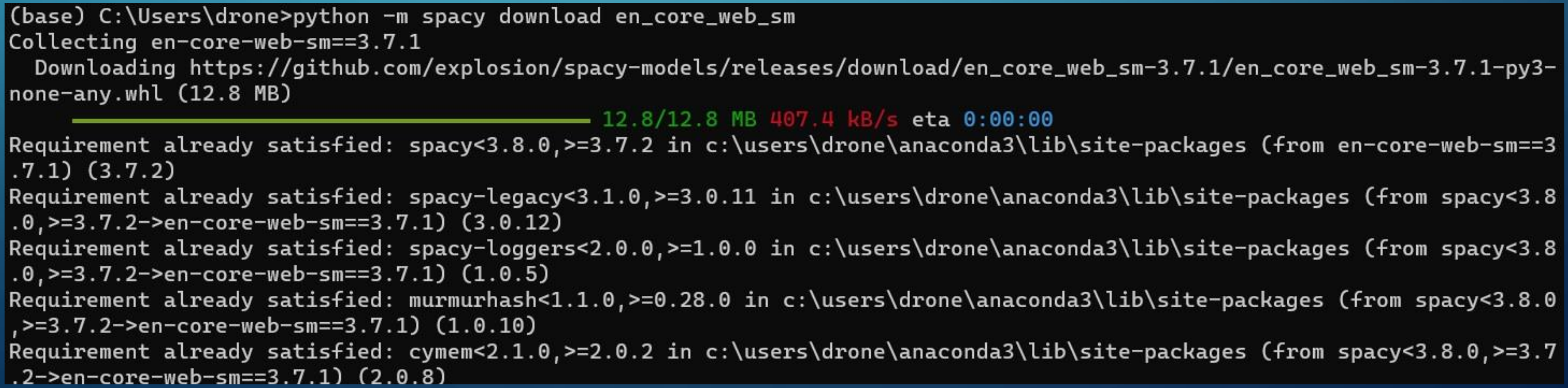
```python
import pyttsx3
engine = pyttsx3.init()
voices = engine.getProperty('voices')
engine.setProperty('voice', voices[1].id)
rate = engine.getProperty('rate')   # getting details of current speaking rate
print (rate)                        #printing current voice rate
engine.setProperty('rate', 125)     # setting up new voice rate
engine.say("I am working. I hope it is good.")
engine.runAndWait()
```

# INTEGRATED DOWNLOADER

- Integrated downloader allows to check the model version and download the corect version of it.
- In case of spacy it can be done be script of command line.



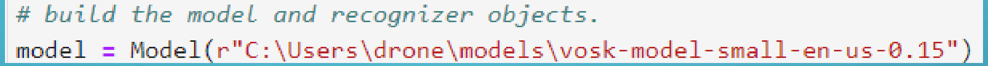
```
!python -m spacy download en_core_web_sm
!python -m spacy download en_core_web_md
```



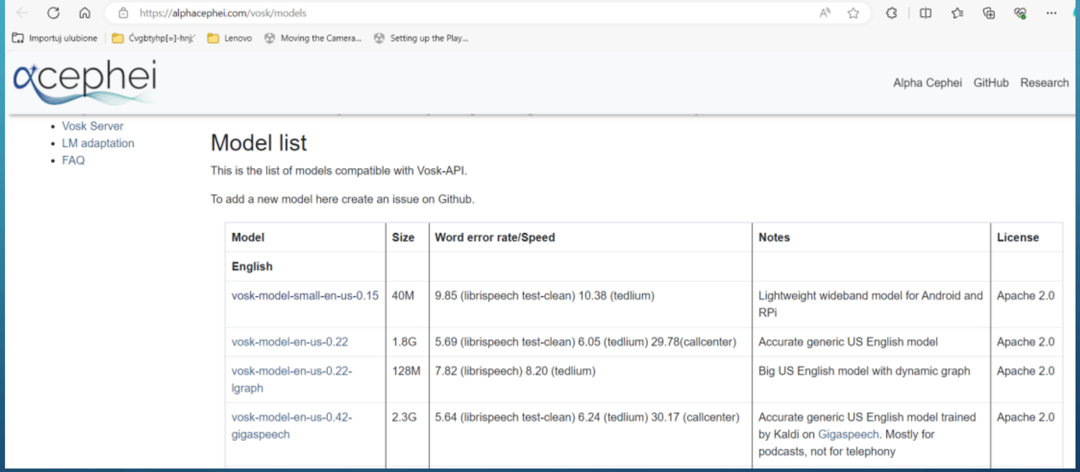
```
(base) C:\Users\drone>python -m spacy download en_core_web_sm
Collecting en-core-web-sm==3.7.1
  Downloading https://github.com/explosion/spacy-models/releases/download/en_core_web_sm-3.7.1/en_core_web_sm-3.7.1-py3-
none-any.whl (12.8 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 12.8/12.8 MB 407.4 kB/s eta 0:00:00
Requirement already satisfied: spacy<3.8.0,>=3.7.2 in c:\users\drone\anaconda3\lib\site-packages (from en-core-web-sm==3
.7.1) (3.7.2)
Requirement already satisfied: spacy-legacy<3.1.0,>=3.0.11 in c:\users\drone\anaconda3\lib\site-packages (from spacy<3.8
.0,>=3.7.2->en-core-web-sm==3.7.1) (3.0.12)
Requirement already satisfied: spacy-loggers<2.0.0,>=1.0.0 in c:\users\drone\anaconda3\lib\site-packages (from spacy<3.8
.0,>=3.7.2->en-core-web-sm==3.7.1) (1.0.5)
Requirement already satisfied: murmurhash<1.1.0,>=0.28.0 in c:\users\drone\anaconda3\lib\site-packages (from spacy<3.8.0
,>=3.7.2->en-core-web-sm==3.7.1) (1.0.10)
Requirement already satisfied: cymem<2.1.0,>=2.0.2 in c:\users\drone\anaconda3\lib\site-packages (from spacy<3.8.0,>=3.7
.2->en-core-web-sm==3.7.1) (2.0.8)
```

# MANUAL DOWNLOAD

- In case of manual download the version and library versions have to be verified manually.
- In this case we find a models site: VOSK Models (alphacephei.com) and download a model.
- The path to a model must be added in a script



```
# build the model and recognizer objects.
model = Model(r"C:\Users\drone\models\vosk-model-small-en-us-0.15")
```



Model list

This is the list of models compatible with Vosk-API.

To add a new model here create an issue on Github.

| Model | Size | Word error rate/Speed | Notes | License |
|---|---|---|---|---|
| **English** | | | | |
| vosk-model-small-en-us-0.15 | 40M | 9.85 (librispeech test-clean) 10.38 (tedlium) | Lightweight wideband model for Android and RPi | Apache 2.0 |
| vosk-model-en-us-0.22 | 1.8G | 5.69 (librispeech test-clean) 6.05 (tedlium) 29.78(callcenter) | Accurate generic US English model | Apache 2.0 |
| vosk-model-en-us-0.22-lgraph | 128M | 7.82 (librispeech) 8.20 (tedlium) | Big US English model with dynamic graph | Apache 2.0 |
| vosk-model-en-us-0.42-gigaspeech | 2.3G | 5.64 (librispeech test-clean) 6.24 (tedlium) 30.17 (callcenter) | Accurate generic US English model trained by Kaldi on Gigaspeech. Mostly for podcasts, not for telephony | Apache 2.0 |

# IMPLEMENTATION OF A MODEL

- To use a AI module the following elements will be implemented:
  - Fetch the sound stream
  - Convert sound stream to text (first AI model)
  - Analyse text using NLP model
  - Return result as voice (third AI model)

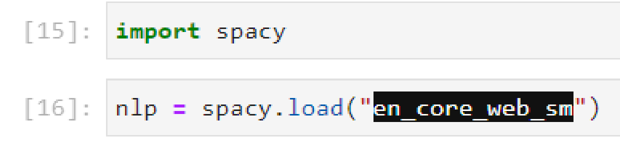
- The example will be presented using Python language

# ENVIRONMENT SETTINGS

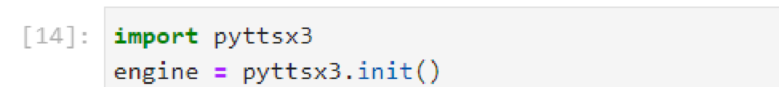
- The script can be applied in environment, where the access to microphone and speakers is possible
- Therefore, the example can be execute using Anaconda environment (access to local computer processes).
- The processing algorithm can also work on servers solutions.

# AI MODELS

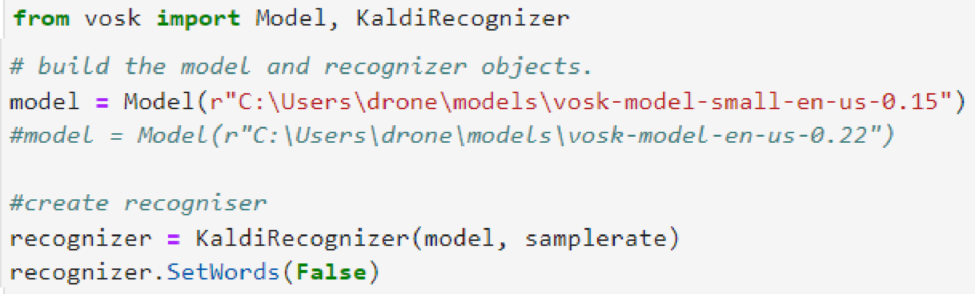
- The three models are imported in three ways.
- The lightweight models was used in example



```
[15]: import spacy

[16]: nlp = spacy.load("en_core_web_sm")
```



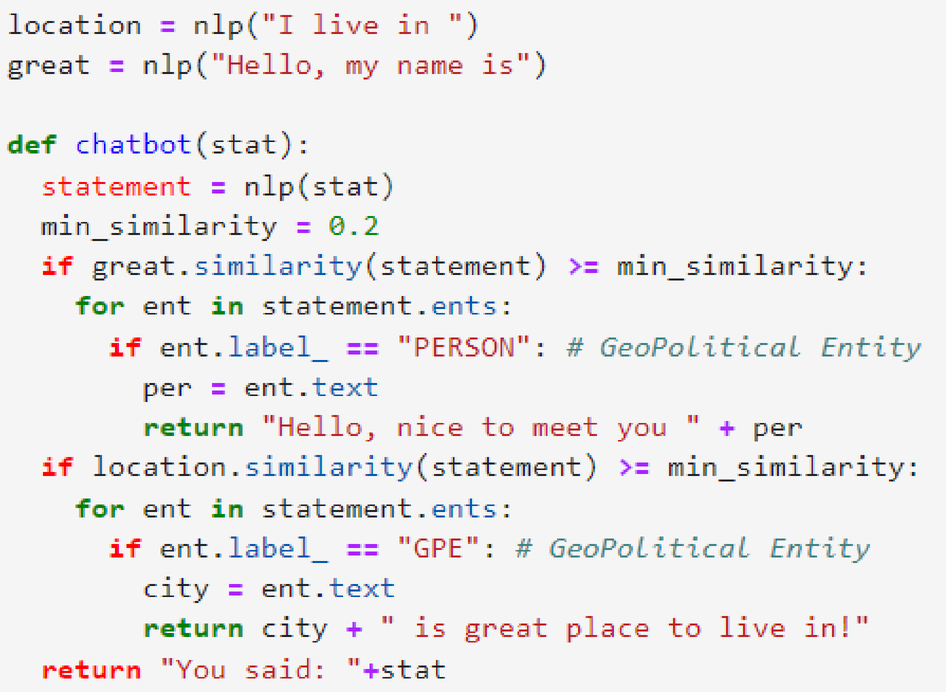
```
[14]: import pyttsx3
      engine = pyttsx3.init()
```



```
from vosk import Model, KaldiRecognizer

# build the model and recognizer objects.
model = Model(r"C:\Users\drone\models\vosk-model-small-en-us-0.15")
#model = Model(r"C:\Users\drone\models\vosk-model-en-us-0.22")

#create recogniser
recognizer = KaldiRecognizer(model, samplerate)
recognizer.SetWords(False)
```

# IMPLEMENTATION - CHATBOT

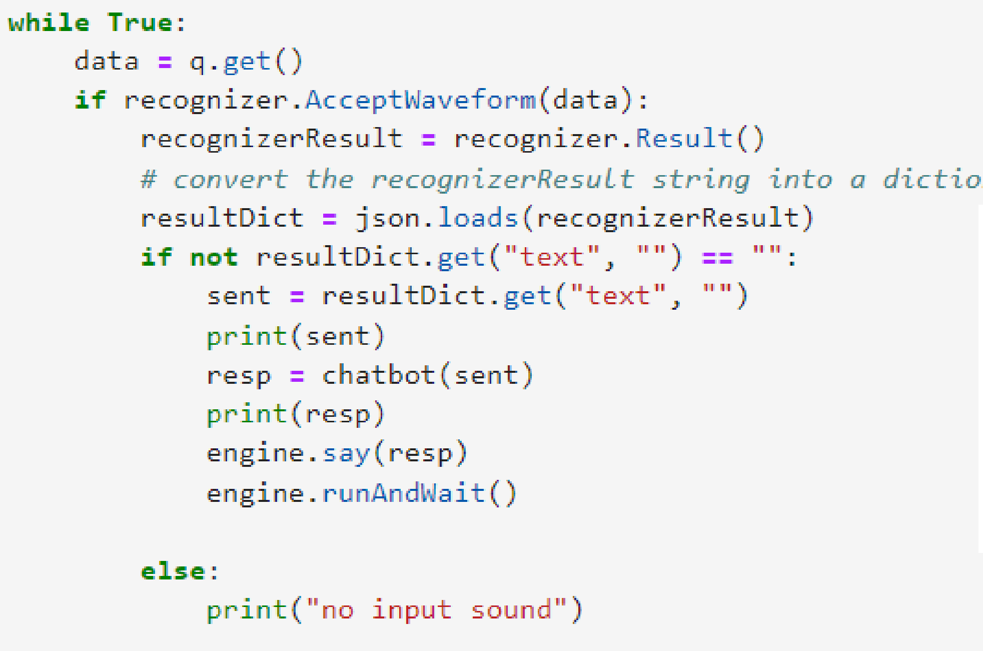
- The chatbot obtain the text information
- It convert in to NLP statement
- Then the statement are compared based of similarity
- The response is given taking under consideration words tags like location or person



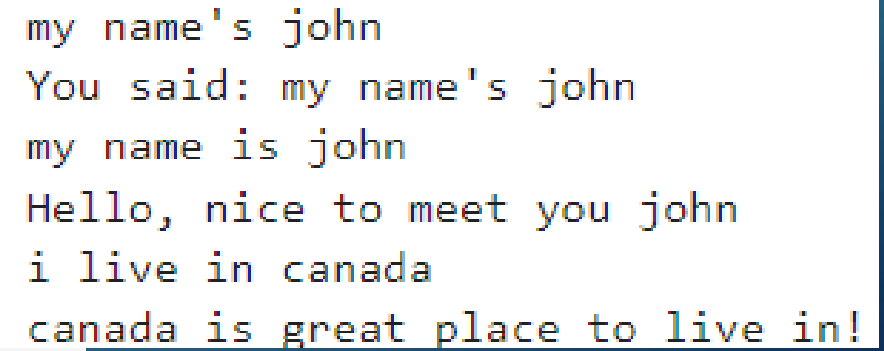
```
location = nlp("I live in ")
great = nlp("Hello, my name is")

def chatbot(stat):
  statement = nlp(stat)
  min_similarity = 0.2
  if great.similarity(statement) >= min_similarity:
    for ent in statement.ents:
      if ent.label_ == "PERSON": # GeoPolitical Entity
        per = ent.text
        return "Hello, nice to meet you " + per
  if location.similarity(statement) >= min_similarity:
    for ent in statement.ents:
      if ent.label_ == "GPE": # GeoPolitical Entity
        city = ent.text
        return city + " is great place to live in!"
  return "You said: "+stat
```

# PROCESS A RESULTS

- The application understand English language and respond in the same language.
- It can repeat sentences and answer simple statements



```
while True:
    data = q.get()
    if recognizer.AcceptWaveform(data):
        recognizerResult = recognizer.Result()
        # convert the recognizerResult string into a dictionary
        resultDict = json.loads(recognizerResult)
        if not resultDict.get("text", "") == "":
            sent = resultDict.get("text", "")
            print(sent)
            resp = chatbot(sent)
            print(resp)
            engine.say(resp)
            engine.runAndWait()

        else:
            print("no input sound")
```



```
my name's john
You said: my name's john
my name is john
Hello, nice to meet you john
i live in canada
canada is great place to live in!
```

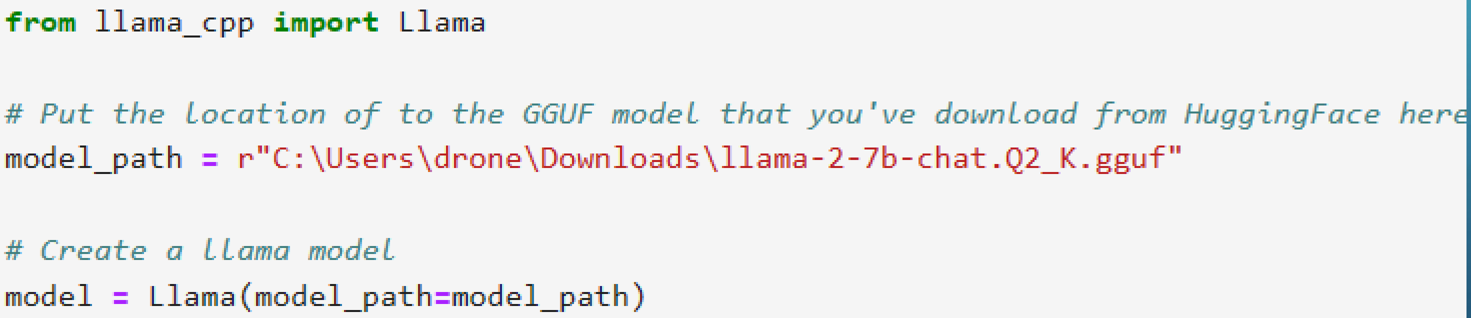
# LLAMA: OPEN AND EFFICIENT FOUNDATION LANGUAGE MODELS

- LLaMA - a collection of foundation language models ranging from 7B to 65B parameters.
- models was trained on trillions of tokens, and show that it is possible to train state-of-the-art models using publicly available datasets exclusively, without resorting to proprietary and inaccessible datasets.
- In particular, LLaMA-13B outperforms GPT-3 (175B) on most benchmarks

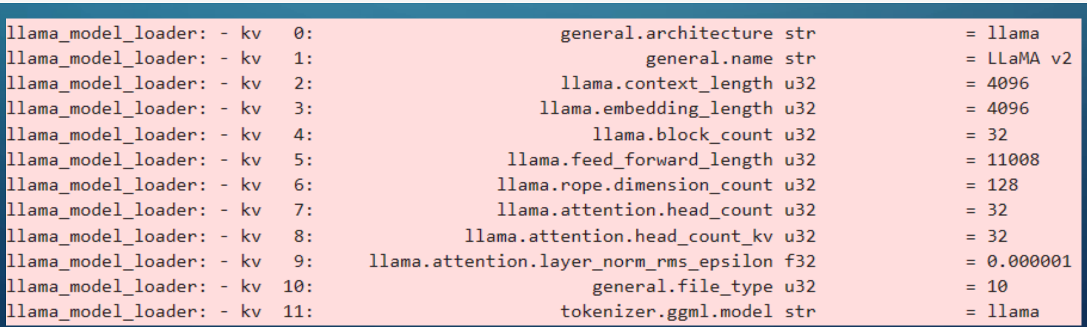
[2302.13971] LLaMA: Open and Efficient Foundation Language Models (arxiv.org)

# USING MODEL

- The model can be obtained via Meta platform or its modified clone: [TheBloke/Llama-2-7B-Chat-GGUF · Hugging Face]()
- The model then can be uploaded to environment:



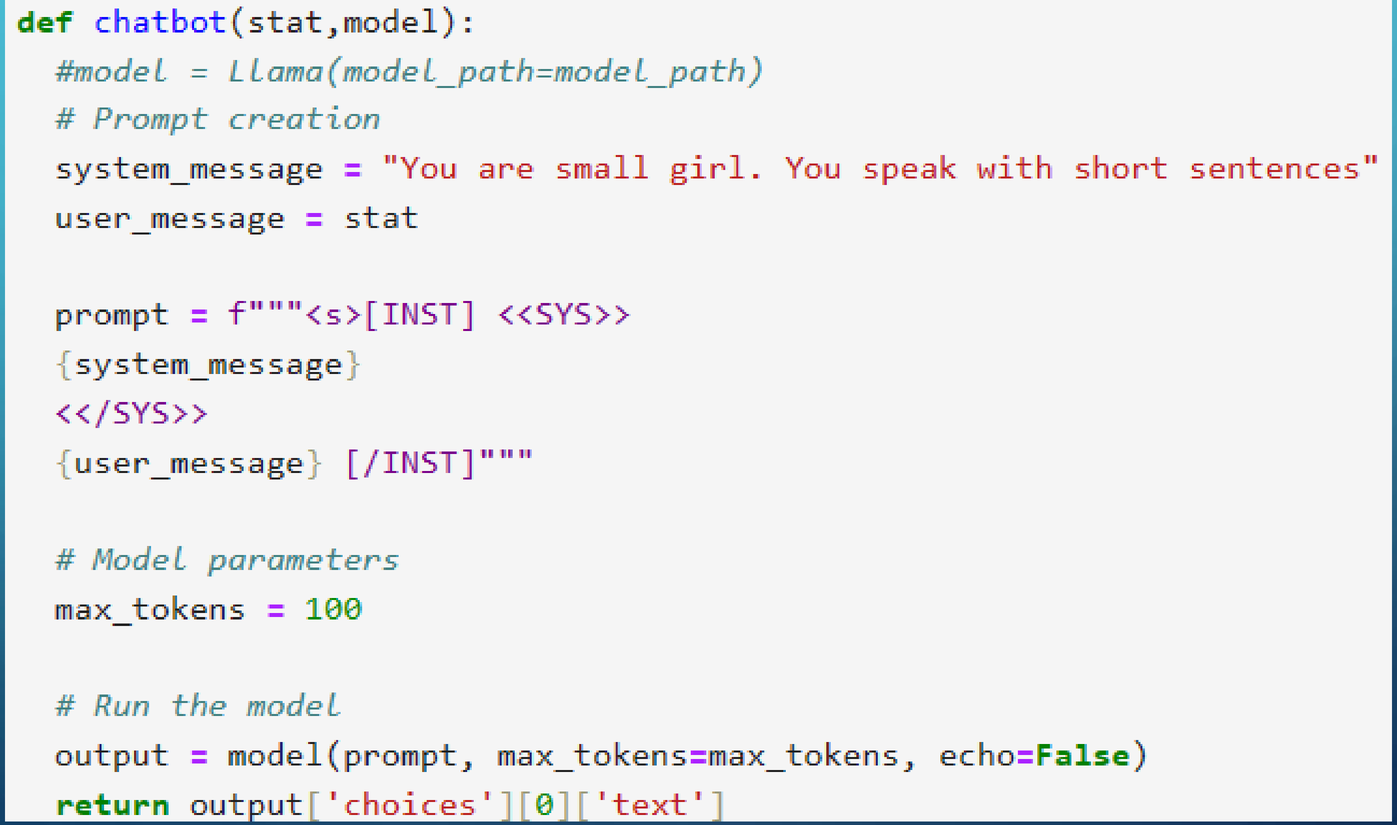
```python
from llama_cpp import Llama

# Put the location of to the GGUF model that you've download from HuggingFace here
model_path = r"C:\Users\drone\Downloads\llama-2-7b-chat.Q2_K.gguf"

# Create a llama model
model = Llama(model_path=model_path)
```



```
llama_model_loader: - kv   0:                       general.architecture str              = llama
llama_model_loader: - kv   1:                               general.name str              = LLaMA v2
llama_model_loader: - kv   2:                       llama.context_length u32              = 4096
llama_model_loader: - kv   3:                     llama.embedding_length u32              = 4096
llama_model_loader: - kv   4:                          llama.block_count u32              = 32
llama_model_loader: - kv   5:                  llama.feed_forward_length u32              = 11008
llama_model_loader: - kv   6:                 llama.rope.dimension_count u32              = 128
llama_model_loader: - kv   7:                 llama.attention.head_count u32              = 32
llama_model_loader: - kv   8:              llama.attention.head_count_kv u32              = 32
llama_model_loader: - kv   9:     llama.attention.layer_norm_rms_epsilon f32              = 0.000001
llama_model_loader: - kv  10:                          general.file_type u32              = 10
llama_model_loader: - kv  11:                       tokenizer.ggml.model str              = llama
```

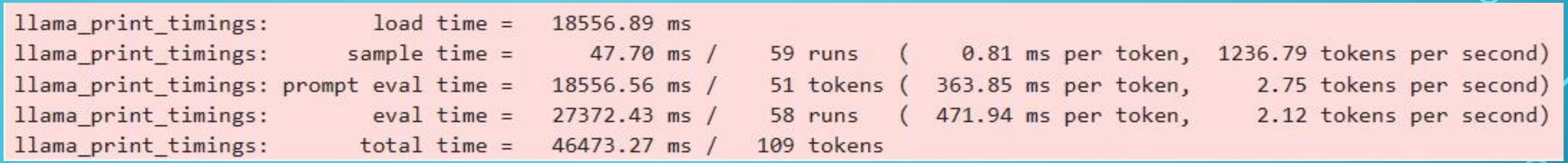
# CHATBOT MODIFICATION

- The context and the question is forwarded to a model by function:

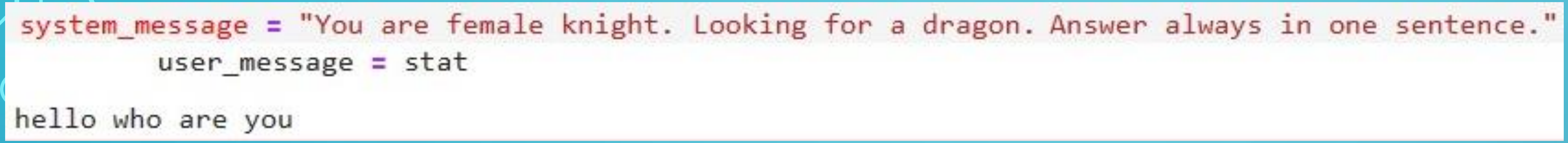


```
def chatbot(stat,model):
  #model = Llama(model_path=model_path)
  # Prompt creation
  system_message = "You are small girl. You speak with short sentences"
  user_message = stat

  prompt = f"""<s>[INST] <<SYS>>
  {system_message}
  <</SYS>>
  {user_message} [/INST]"""

  # Model parameters
  max_tokens = 100

  # Run the model
  output = model(prompt, max_tokens=max_tokens, echo=False)
  return output['choices'][0]['text']
```

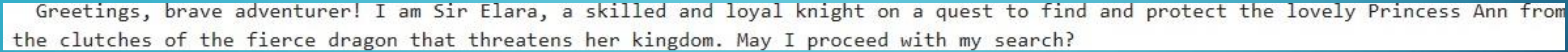
# RESULT



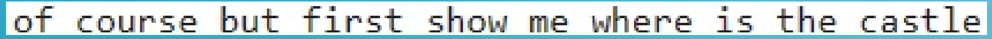
```
llama_print_timings:        load time =  18556.89 ms
llama_print_timings:      sample time =     47.70 ms /    59 runs   (    0.81 ms per token,  1236.79 tokens per second)
llama_print_timings: prompt eval time =  18556.56 ms /    51 tokens (  363.85 ms per token,     2.75 tokens per second)
llama_print_timings:        eval time =  27372.43 ms /    58 runs   (  471.94 ms per token,     2.12 tokens per second)
llama_print_timings:       total time =  46473.27 ms /   109 tokens
```



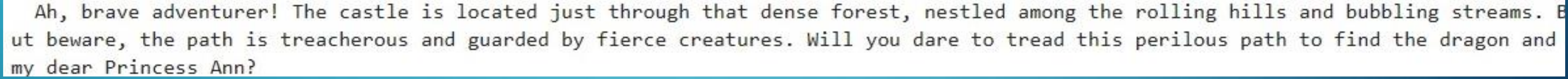
```
system_message = "You are female knight. Looking for a dragon. Answer always in one sentence."
        user_message = stat

hello who are you
```



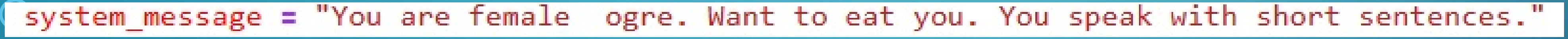
```
  Greetings, brave adventurer! I am Sir Elara, a skilled and loyal knight on a quest to find and protect the lovely Princess Ann from
 the clutches of the fierce dragon that threatens her kingdom. May I proceed with my search?
```



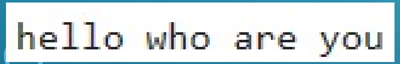
```
of course but first show me where is the castle
```



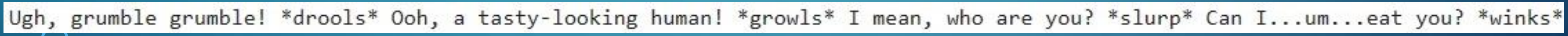
```
  Ah, brave adventurer! The castle is located just through that dense forest, nestled among the rolling hills and bubbling streams. B
ut beware, the path is treacherous and guarded by fierce creatures. Will you dare to tread this perilous path to find the dragon and
my dear Princess Ann?
```



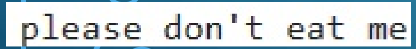
```
system_message = "You are female  ogre. Want to eat you. You speak with short sentences."
```



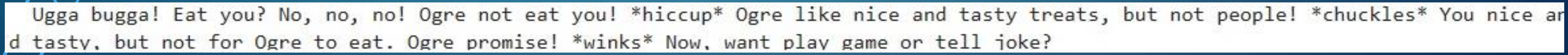
```
hello who are you
```



```
Ugh, grumble grumble! *drools* Ooh, a tasty-looking human! *growls* I mean, who are you? *slurp* Can I...um...eat you? *winks*
```

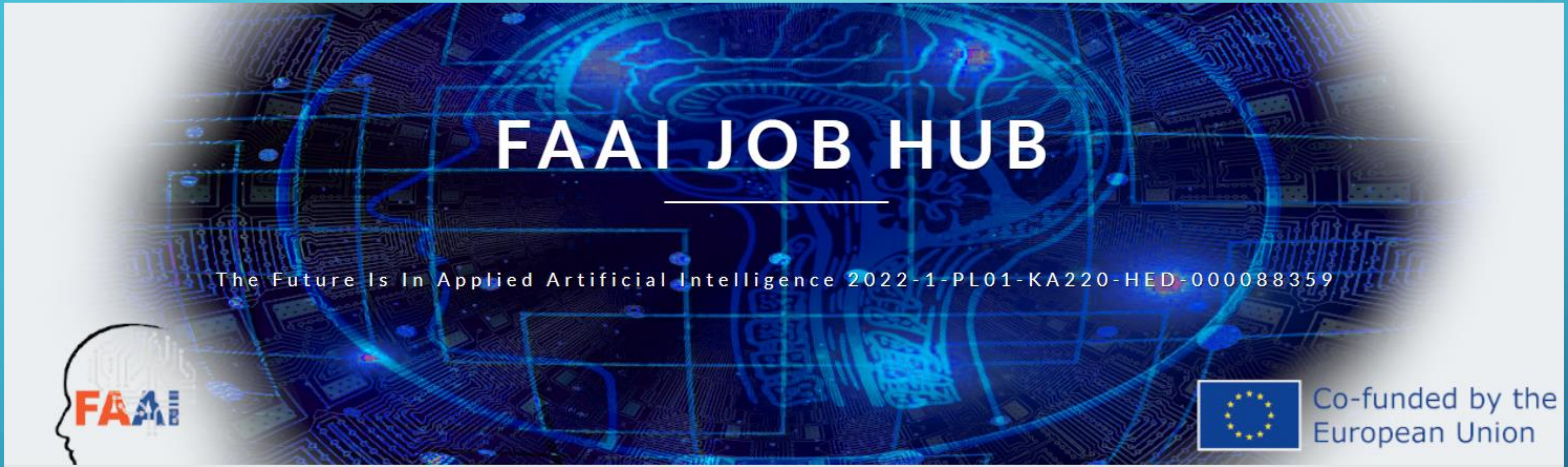


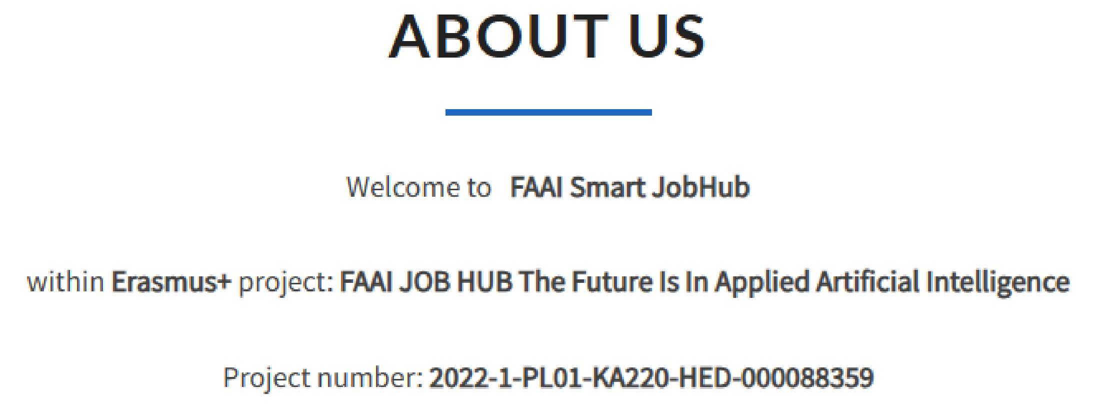
```
please don't eat me
```



```
  Ugga bugga! Eat you? No, no, no! Ogre not eat you! *hiccup* Ogre like nice and tasty treats, but not people! *chuckles* You nice ar
d tasty, but not for Ogre to eat. Ogre promise! *winks* Now, want play game or tell joke?
```

# THANK YOU FOR ATTENTION!


FAAI JOB HUB
The Future Is In Applied Artificial Intelligence 2022-1-PL01-KA220-HED-000088359
FAAI
Co-funded by the European Union



## ABOUT US

Welcome to **FAAI Smart JobHub**

within **Erasmus+** project: **FAAI JOB HUB The Future Is In Applied Artificial Intelligence**

Project number: **2022-1-PL01-KA220-HED-000088359**

**https://faai.ath.edu.pl/**

# REFERENCES

- https://www.researchgate.net/figure/Reinforcement-Learning-Agent-and-Environment_fig2_323867253
- Machine Learning: Algorithms, Real-World Applications and Research Directions. Iqbal H. Sarker. SN Computer Science volume 2, Article number: 160 (2021
- Breiman L, Friedman J, Stone CJ, Olshen RA. Classification and regression trees. CRC Press; 1984.
- Keerthi SS, Shevade SK, Bhattacharyya C, Radha Krishna MK. Improvements to platt's smo algorithm for svm classifier design. Neural Comput. 2001;13(3):637–49.
- Krizhevsky A, Sutskever I, Hinton GE. Imagenet classification with deep convolutional neural networks. In: Advances in neural information processing systems, 2012: 1097–1105
- Machine Learning: Algorithms, Real-World Applications and Research Directions. Iqbal H. Sarker. SN Computer Science volume 2, Article number: 160 (2021)
- https://towardsdatascience.com/simple-introduction-to-convolutional-neural-networks-cdf8d3077bac
- History of VR – Timeline of Events and Tech Development – VirtualSpeech