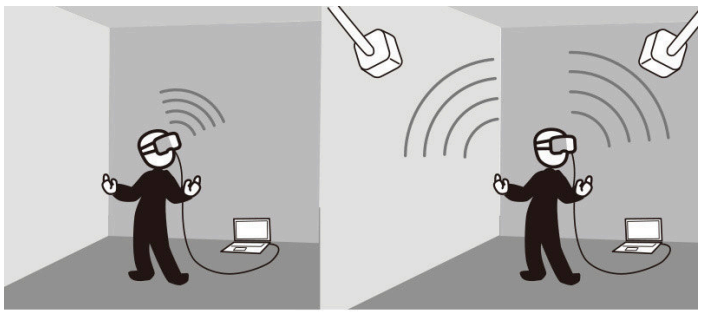
Project Design:

Hand-Controlled Environment Explorer

Background:

Hand tracking is a very large part of this project. Hand tracking in a computer context, surprisingly, refers to the virtual tracking of the position of a hand. This can be done in a variety of ways, such as outside-in (lighthouse) based tracking, using sensors in the environment to calculate positional data (used in the HTC Vive VR Headset[1]). Alternatively, inside-out tracking uses sensors positioned relative to the hands themselves, such as the Oculus Quest's camera based tracking (see picture below). These "sensors" could be a complicated system of photosensitive resistors, or simply a camera (or cameras) that track using computer vision.



Inside-Out Tracking Outside-In Tracking [2]

Computer vision refers to the "field of computer science that focuses on enabling computers to identify and understand objects and people in images and videos"[3]. This could be taken live through a camera

[1] https://hackaday.com/2016/12/21/alan-yates-why-valves-lighthouse-cant-work/

[2] https://www.google.com/url?sa=i&url=https%3A%2F%2Fwww.researchgate.net%2Ffigure%2FInside-out-versus-Outside-in-tracking-for-HMDs-Acer-2019_fig26_349734845&psig=AOvVaw0ORDXfTpXHo_QpPp4bWGII&ust=1686096220462000&source=images&cd=vfe&ved=0CA4QjRxqFwoTCJCK1YSsrf8CFQAAAAAdAAAAABAH

[3] https://azure.microsoft.com/en-ca/resources/cloud-computing-dictionary/what-is-computer-vision/#:~:text=Computer%20vision%20is%20a%20field,tasks%20that%20replicate%20human%20capabilities.

feed, or processed with pre-recorded footage. As it concerns motion tracking, Computer Vision is used to identify movement in a photo, involving techniques such as tracking the path of 'markers' out of regions of high contrast within the object, or predicting the future path of an object based on its current motion.

Augmented Reality (AR) is another thing that is conceptually relevant to this project. Microsoft describes it as being "designed to add digital elements over real-world views with limited interaction"[4]. This kind of thing would be like having a Pikachu from a mobile game be rendered over top of a camera feed, but can only be interacted with through the small touchscreen of the device. This project addresses the possibility of making something more interactive, using your actual physical hands to manipulate those superimposed elements. While ambiguous, this seems to be where the term *mixed reality* (MR) more closely fits.

Unicode is an international organization that aims to standardize text across computers, and its database contains every single text character you have ever seen. This document will refer to *ASCII,* a similar standard, but just know that any special characters will be taken from Unicode, not ASCII.

Note: This design doc represents the design of the current project. Any text in strikethrough is for outdated information that has been left in for reference purposes.

How is the Project Being Built?

This project aims to implement a computer-vision based hand-tracker that will be used to navigate a virtual environment. The user will position their hands in the view of a webcam, and the webcam will track the position of the players hands, and create corresponding movements ~~to pan around the image being displayed.~~ on a text based ASCII maze. The image, initially will be quite simple, but has a lot of room for improvement. ~~I imagine this will probably end up being a mosaic of smaller images that will trigger animations when hovered on.~~

The hand tracking component of this project will rely on various libraries (MediaPipe[5] for the bulk of the hand tracking, and OpenCV[6] mainly for image manipulation) that can track the X and Y, ~~(and possibly Z!)~~ position of the user's hands (and possible fingers depending on feasibility of implementation) and send that movement data off to the "environment".

---

[4] https://dynamics.microsoft.com/en-ca/mixed-reality/guides/what-is-augmented-reality-ar/#:~:text=Augmented%20reality%20(AR)%E2%80%94%20designed,headphones%20designed%20for%20such%20activities.

[5] https://developers.google.com/mediapipe

[6] https://docs.opencv.org/4.x/

The term “environment” is left ambiguous as a means to keep the project open to continual development and improvements. As it stands currently, this is a text based ASCII maze that will be displayed in an output window.

~~I use the term ‘environment’ to refer to whatever visual contraption I end up implementing. Initially, this will start off as just moving around a picture, but ideally will evolve into moving around an animated (possibly real-time rendered?) image with some interactive elements. While we’re *wishing for a unicorn*, a further development of this implementation could involve navigating a 3D environment, which would be contingent on z-axis tracking (a very difficult thing to do), and, of course, the rendering of an actual 3D environment.~~

Because I realized I didn’t put this anywhere, it’s going here! This project will be implemented in Python, and will likely use Google Collab, although a brief foray into other services may be done first.

Input:

- Footage from a WebCam or other camera device for tracking
- Possibly a pre-recorded video file for testing purposes

Output:

- ~~The visual representation of the ”environment”,~~ ASCII maze, controlled by hand movements

Modules:

As mentioned previously, this project will be broken down into ‘modules’ to iterate through the development. These effectively work as a schedule.

Much of this work was taken from these projects:

- mediapipe_hands.ipynb
- YOLOv4 Object Detection on Webcam In Google Colab
- MediaPipe Python Tutorial [How to Install + Real-Time Hand Tracking Example] | by Iqra Anwar
- How to Get all the Co-ordinates of Hand using Mediapipe Hand Solutions ? | by Abdul Rehman Kalsekar

1. Basic Hand Tracking - Output Movement

Focused on actually getting setup with hand tracking, and outputting the movement of the hand in a usable way.

Involves:

Importing libraires:

- from IPython.display import Image
- from IPython.display import display, Javascript
- from google.colab.output import eval_js

- from base64 import b64decode

WebCam Feed

- Creating WebCame Feed. This must be done with JavaScript as Google Collab does not allow you to make windows. Not sure how to do that exactly (involves using iPython's javascript class) but just copy the video_stream function from the mediapipe_hands.ipynb project.
- Function to convert JavaScript image into something that can be used by the rest of the code. Again, copy the js_to_image function from the same project
- Video_stream function - Gets the current frame of the video. Again, copy from project
- Call the function for the webcam feed in the main part of the code

import mediapipe as mp
import numpy as np
import time
import cv2

Process Camera Feed image

- Call the video_stream function and save its value to a variable
- Run the js_to_image function on variable above
- Use openCV to flip the image horizontally (not mirrored)

Tracking:

- Call mediapipe.hands with appropriate parameters (1 hand, 0.5 confidence, etc.)
- This part is unclear. It is missing from the current project but seems to be vital to operation. It seems some sort of iteration of the processed frame needs to occur, although I'm unsure as to what. See screenshot below taken from image:

```
if not results.multi_hand_landmarks:
  continue
# Draw hand landmarks of each hand.
print(f'Hand landmarks of {name}:')
image_hight, image_width, _ = image.shape
annotated_image = cv2.flip(image.copy(), 1)
for hand_landmarks in results.multi_hand_landmarks:
```

- Use hand_landmarks.landmark[mp_hands.HandLandmark.INDEX_FINGER_TIP].x and .y to get the x and y location of the index finger.
- Print x and y to console

- ~~Figuring out how to use WebCam (libraries? Mediapipe?)~~[7]
- ~~Rummaging through libraries to figure out how to track hands/fingers. Involves reading! Check out OpenCV's documentation, and start with *objdetect* and *tracking*~~

[7] https://mediapipe.readthedocs.io/en/latest/

- ~~Deciding on best output implementation. Options include:~~
    - ~~Left, Right, Up, Down~~
    - ~~Vectors?~~
    - ~~Magnitude and direction (vectors but like not formally)~~
    - ~~X and Y coordinates~~
- ~~Printing direction to console, as well as returning movement in some way.~~

2. Constructing Environment

Next step! Making something visual that can be navigated. Initially this will be a simple, low-resolution image, which can branch out as time goes on. At this time, the implementation is an ASCII maze.

```
import os
import random
```

In another code snippet above the main code snippet (beside where the other functions are):

- Global variables for keeping track of past x and y coordinates
- Making a grid from a 2 dimensional array (lists within lists). Populate each square with something like an "○"
- With Randint, randomly choose indices to replace the o with a barrier character,. Something like "▉".
- Function for updating grid (takes current x and y):
    - Clear the current board
    - Put an open square of the previous position (global variables)
    - Put a star for the new position "✦"
    - Sets global variables to current position
    - Print board
- Where the x and y coordinates were initially printed, call this function instead, placing the x and y arguments into the parameters

- ~~Outputting an image, but in a way that can be easily manipulated and moved around. Ideally this should be something that will scale very well, so SVG files may be ideal. Again, more reading! There's some GUI/image related libraries out there for python, and some stuff within OpenCV. Consider modules like *cudaimgproc*, *cvv*, and *highgui*. If it lends itself better, this may evolve into creating a navigable GUI that or may or not resemble this "maze"~~
- ~~Build navigation framework - how will this image be moved around? This will depend on implementation in module 1, but consider:~~
    - ~~Shifting by # of pixels?~~

    - Acceleration based on speed
    - Using x or y coordinates?
- Marrying movement logic w/ movement data from camera. Consider a function approach, in which position is obtained/updated by constantly calling a function that returns movement data (Δ movement?), which the rest of the code can use in something like a loop

3. Branch

Multiple options to pursue here. Subject to change. Branches include:

Animated Map

- Could look into making "environment" something more fluid, like a map that animates when you hover over it. I could see this being a mosaic of images, in which a given section could “update” into an animated version that moves (i.e. waves with dolphins), possible with sounds

AR-Like Interactions

- Consider placing these images over top of a camera feed. Again, this will vary wildly based on the approach.
- Figuring out z-axis tracking and a 3D "Environment"
- For a 3D environment, Could it be an SVG file? or some pre-rendered 3D structure? Or, and again while we’re wishing for a unicorn, something ray-traced?

Miscellaneous

- How feasible would it be to capture the 3D environment of a proper game, and navigate it? Hmmm