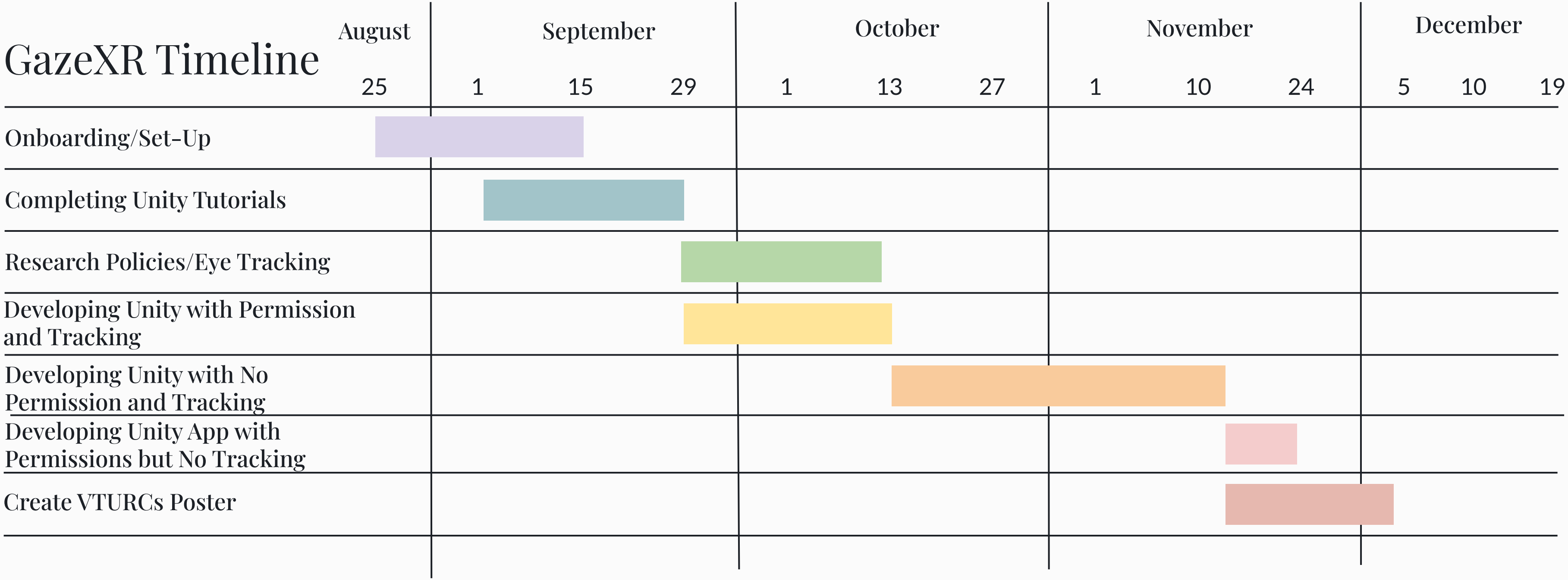
BURGS Weekly Presentation

Broadening Undergraduate Research Groups

10/10/2025

Allie, Casie, Gayatri, Kim

GazeXR: Updated Timeline



GazeXR Timeline

August

September

October

November

December

25 1 15 29 1 13 27 1 10 24 5 10 19

Onboarding/Set-Up

Completing Unity Tutorials

Research Policies/Eye Tracking

Developing Unity with Permission and Tracking

Developing Unity with No Permission and Tracking

Developing Unity App with Permissions but No Tracking

Create VTURCs Poster

Objectives

GazeXR: Game Design Idea

Purpose: Disguise as a fully playable game

01

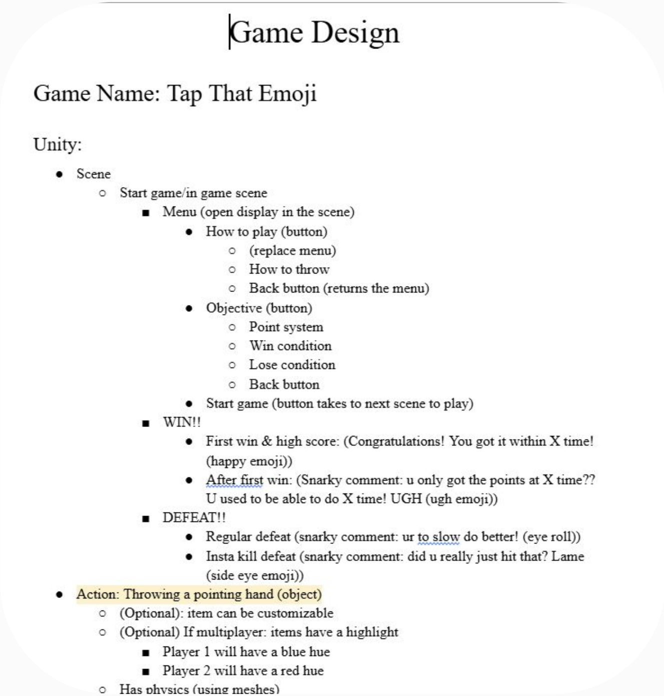
Functionality

- Objects & Scenes in Unity
- Behaviors of Objects
- Eye tracking background
  - Don’t need ray interactions

02

Playability

- Win & Defeat conditions
- Theme of game (Emojis :D)
- Purpose of game
- Fun factor (giving struggles)



Game Design

Game Name: Tap That Emoji

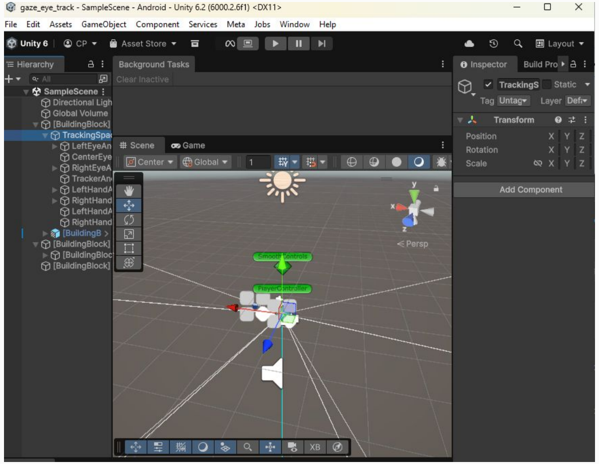
Unity:

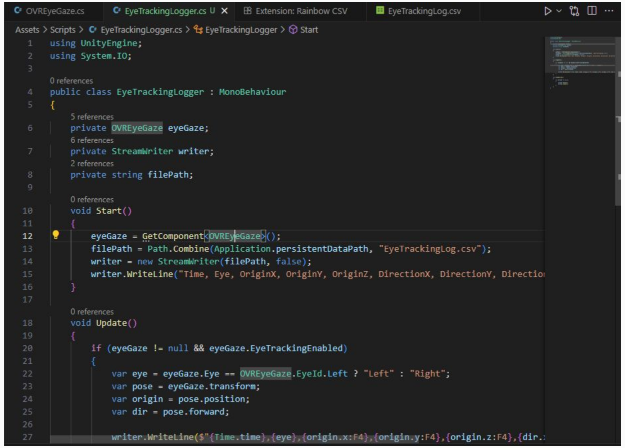
- Scene
  - Start game/in game scene
    - Menu (open display in the scene)
      - How to play (button)
        - (replace menu)
        - How to throw
        - Back button (returns the menu)
      - Objective (button)
        - Point system
        - Win condition
        - Lose condition
        - Back button
      - Start game (button takes to next scene to play)
    - WIN!!
      - First win & high score: (Congratulations! You got it within X time! (happy emoji))
      - After first win: (Snarky comment: u only got the points at X time?? U used to be able to do X time! UGH (ugh emoji))
    - DEFEAT!!
      - Regular defeat (snarky comment: ur to slow do better! (eye roll))
      - Insta kill defeat (snarky comment: did u really just hit that? Lame (side eye emoji))
- Action: Throwing a pointing hand (object)
  - (Optional): item can be customizable
  - (Optional) If multiplayer: items have a highlight
    - Player 1 will have a blue hue
    - Player 2 will have a red hue
  - Has physics (using meshes)

Objectives

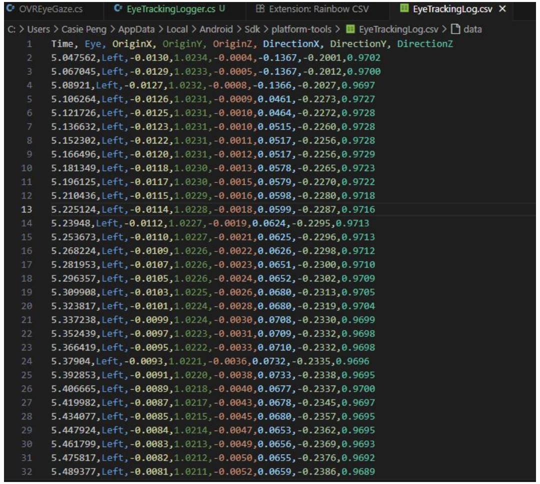
GazeXR: Eye Tracking with Permissions

Basic Functionality Finished





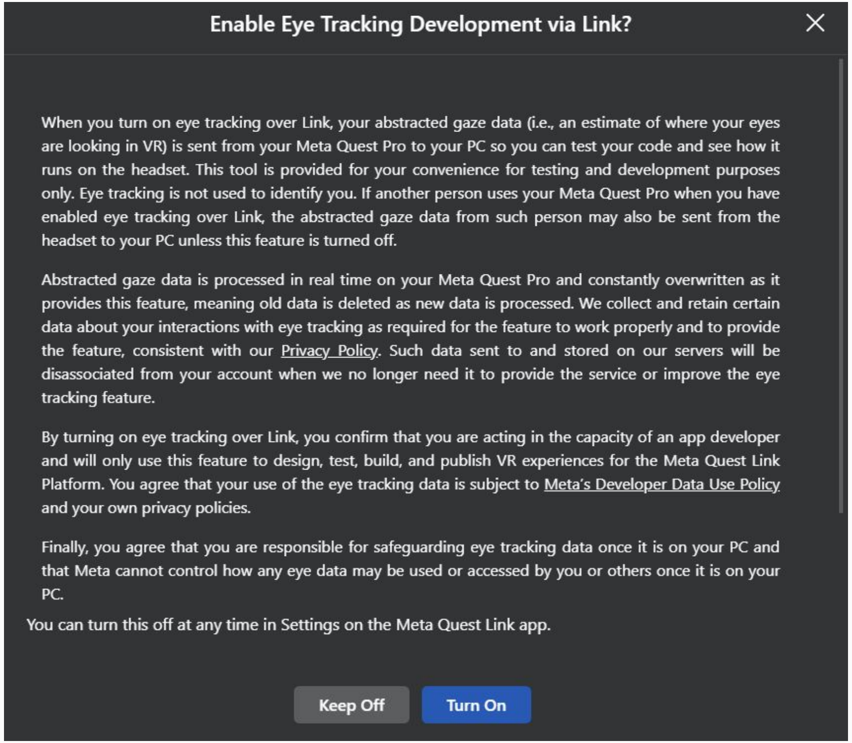
```
using UnityEngine;
using System.IO;

0 references
public class EyeTrackingLogger : MonoBehaviour
{
    5 references
    private OVREyeGaze eyeGaze;
    6 references
    private StreamWriter writer;
    2 references
    private string filePath;

    0 references
    void Start()
    {
        eyeGaze = GetComponent<OVREyeGaze>();
        filePath = Path.Combine(Application.persistentDataPath, "EyeTrackingLog.csv");
        writer = new StreamWriter(filePath, false);
        writer.WriteLine("Time, Eye, OriginX, OriginY, OriginZ, DirectionX, DirectionY, Directio
    }

    0 references
    void Update()
    {
        if (eyeGaze != null && eyeGaze.EyeTrackingEnabled)
        {
            var eye = eyeGaze.Eye == OVREyeGaze.EyeId.Left ? "Left" : "Right";
            var pose = eyeGaze.transform;
            var origin = pose.position;
            var dir = pose.forward;

            writer.WriteLine($"{Time.time},{eye},{origin.x:F4},{origin.y:F4},{origin.z:F4},{dir.
```


OVREyeGaze.cs
EyeTrackingLogger.cs U
Extension: Rainbow CSV
EyeTrackingLog.csv
C: > Users > Casie Peng > AppData > Local > Android > Sdk > platform-tools > EyeTrackingLog.csv > data

```
Time, Eye, OriginX, OriginY, OriginZ, DirectionX, DirectionY, DirectionZ
5.047562,Left,-0.0130,1.0234,-0.0004,-0.1367,-0.2001,0.9702
5.067045,Left,-0.0129,1.0233,-0.0005,-0.1367,-0.2012,0.9700
5.08921,Left,-0.0127,1.0232,-0.0008,-0.1366,-0.2027,0.9697
5.106264,Left,-0.0126,1.0231,-0.0009,0.0461,-0.2273,0.9727
5.121726,Left,-0.0125,1.0231,-0.0010,0.0464,-0.2272,0.9728
5.136632,Left,-0.0123,1.0231,-0.0010,0.0515,-0.2260,0.9728
5.152302,Left,-0.0122,1.0231,-0.0011,0.0517,-0.2256,0.9728
5.166496,Left,-0.0120,1.0231,-0.0012,0.0517,-0.2256,0.9729
5.181349,Left,-0.0118,1.0230,-0.0013,0.0578,-0.2265,0.9723
5.196125,Left,-0.0117,1.0230,-0.0015,0.0579,-0.2270,0.9722
5.210436,Left,-0.0115,1.0229,-0.0016,0.0598,-0.2280,0.9718
5.225124,Left,-0.0114,1.0228,-0.0018,0.0599,-0.2287,0.9716
5.23948,Left,-0.0112,1.0227,-0.0019,0.0624,-0.2295,0.9713
5.253673,Left,-0.0110,1.0227,-0.0021,0.0625,-0.2296,0.9713
5.268224,Left,-0.0109,1.0226,-0.0022,0.0626,-0.2298,0.9712
5.281953,Left,-0.0107,1.0226,-0.0023,0.0651,-0.2300,0.9710
5.296357,Left,-0.0105,1.0226,-0.0024,0.0652,-0.2302,0.9709
5.309908,Left,-0.0103,1.0225,-0.0026,0.0680,-0.2313,0.9705
5.323817,Left,-0.0101,1.0224,-0.0028,0.0680,-0.2319,0.9704
5.337238,Left,-0.0099,1.0224,-0.0030,0.0708,-0.2330,0.9699
5.352439,Left,-0.0097,1.0223,-0.0031,0.0709,-0.2332,0.9698
5.366419,Left,-0.0095,1.0222,-0.0033,0.0710,-0.2332,0.9698
5.37904,Left,-0.0093,1.0221,-0.0036,0.0732,-0.2335,0.9696
5.392853,Left,-0.0091,1.0220,-0.0038,0.0733,-0.2338,0.9695
5.406665,Left,-0.0089,1.0218,-0.0040,0.0677,-0.2337,0.9700
5.419982,Left,-0.0087,1.0217,-0.0043,0.0678,-0.2345,0.9697
5.434077,Left,-0.0085,1.0215,-0.0045,0.0680,-0.2357,0.9695
5.447924,Left,-0.0084,1.0214,-0.0047,0.0653,-0.2362,0.9695
5.461799,Left,-0.0083,1.0213,-0.0049,0.0656,-0.2369,0.9693
5.475817,Left,-0.0082,1.0212,-0.0050,0.0655,-0.2376,0.9692
5.489377,Left,-0.0081,1.0211,-0.0052,0.0659,-0.2386,0.9689
```

GazeXR: Developer Eye Tracking

Important?



Enable Eye Tracking Development via Link?

When you turn on eye tracking over Link, your abstracted gaze data (i.e., an estimate of where your eyes are looking in VR) is sent from your Meta Quest Pro to your PC so you can test your code and see how it runs on the headset. This tool is provided for your convenience for testing and development purposes only. Eye tracking is not used to identify you. If another person uses your Meta Quest Pro when you have enabled eye tracking over Link, the abstracted gaze data from such person may also be sent from the headset to your PC unless this feature is turned off.

Abstracted gaze data is processed in real time on your Meta Quest Pro and constantly overwritten as it provides this feature, meaning old data is deleted as new data is processed. We collect and retain certain data about your interactions with eye tracking as required for the feature to work properly and to provide the feature, consistent with our Privacy Policy. Such data sent to and stored on our servers will be disassociated from your account when we no longer need it to provide the service or improve the eye tracking feature.

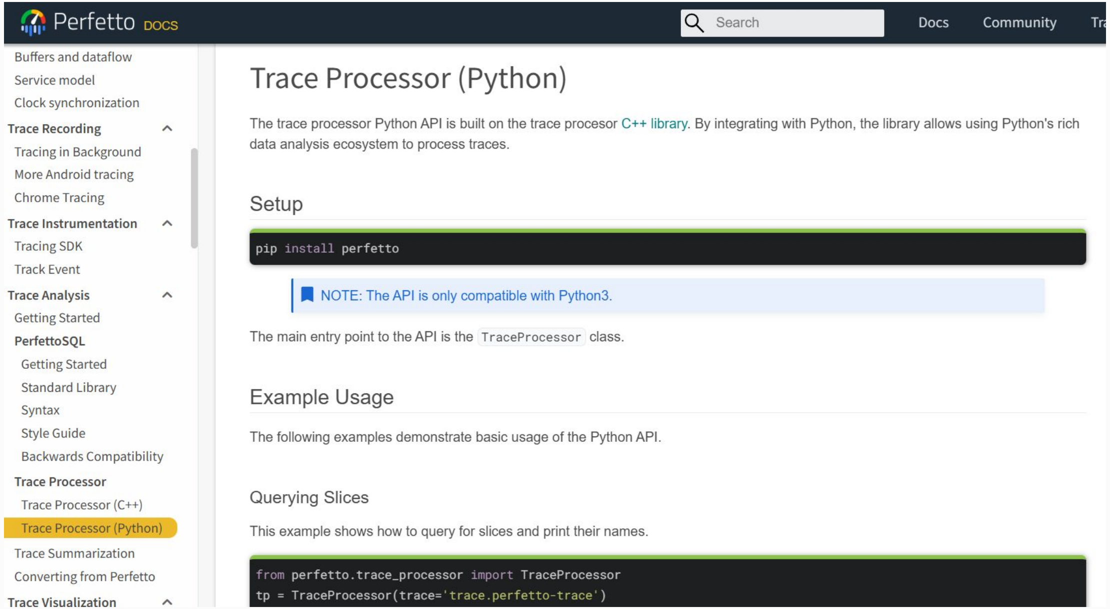
By turning on eye tracking over Link, you confirm that you are acting in the capacity of an app developer and will only use this feature to design, test, build, and publish VR experiences for the Meta Quest Link Platform. You agree that your use of the eye tracking data is subject to Meta's Developer Data Use Policy and your own privacy policies.

Finally, you agree that you are responsible for safeguarding eye tracking data once it is on your PC and that Meta cannot control how any eye data may be used or accessed by you or others once it is on your PC.

You can turn this off at any time in Settings on the Meta Quest Link app.

Perfetto Trace Processor API

Uses SQL queries to transform a .pftrace function into a time-series Pandas dataframe directly!



Trace Processor (Python)

The trace processor Python API is built on the trace procesor C++ library. By integrating with Python, the library allows using Python's rich data analysis ecosystem to process traces.

Setup

```
pip install perfetto
```

NOTE: The API is only compatible with Python3.

The main entry point to the API is the `TraceProcessor` class.

Example Usage

The following examples demonstrate basic usage of the Python API.

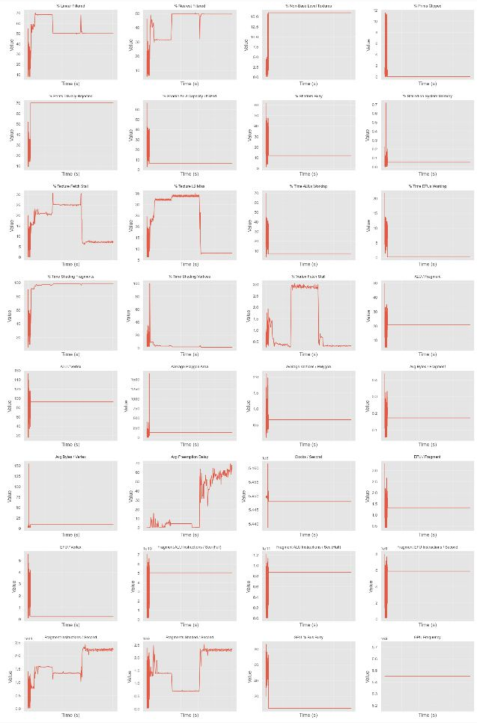
Querying Slices

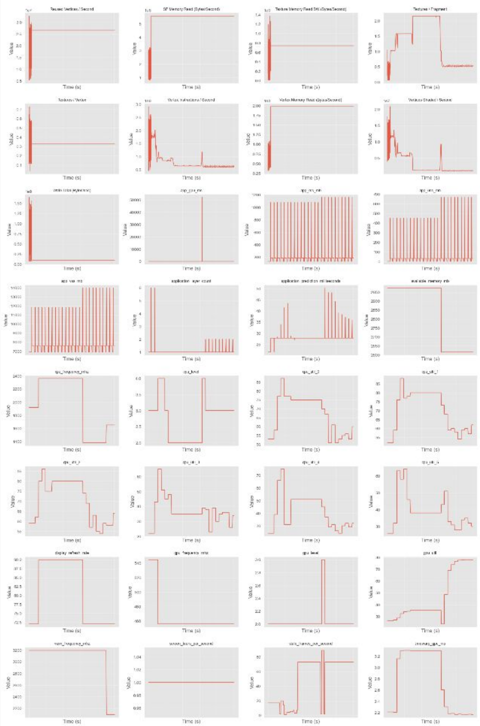
This example shows how to query for slices and print their names.

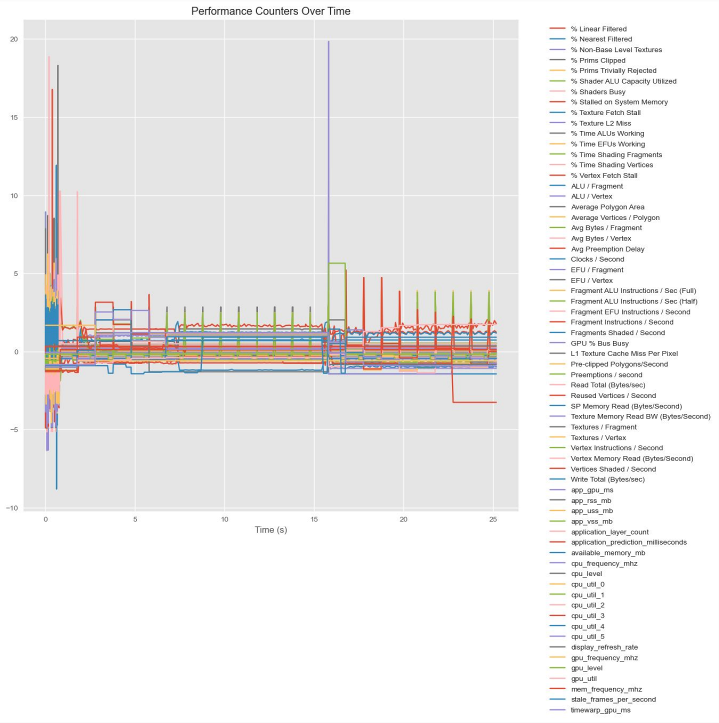
```
from perfetto.trace_processor import TraceProcessor
tp = TraceProcessor(trace='trace.perfetto-trace')
```

Objectives

Data Cleaned and Visualized







Next Steps

1. Top priority: collect more data
    a. Can’t start training a model until we have at least 3-5 data points per room across 3-5 room types
    b. Eventually want 15 data points per room
2. Model types I want to test:
    a. Random forest: basic but effective
    b. Dynamic time warping (DTW) with 1-NN: used to be the gold standard of time series classification
    c. ROCKET: comparable speed and accuracy to larger models, but requires significantly less computation

Questions

1. GazeXR: Is eye tracked data accurate? (Going to meet with Anish if possible)