**Documentation for Fove Unity Visual Stimulation and Eye Tracking Project**

**Background & Research**

A Unity-based application that delivers dynamic visual stimuli using the Fove eye-tracking headset. It presents chromatic light stimuli (red and blue spheres) in various visual field positions while recording pupil data and event markers via the Lab Streaming Layer (LSL).

**Problem Statement**

This project enables controlled visual stimulation in virtual reality using the **Fove** headset and **LSL (Lab Streaming Layer)** to capture:

* **Visual Stimuli Events** – Red/Blue sphere presentations at different locations.
* **Eye Tracking Data** – Real-time pupil radius data from the Fove headset.

Useful for psychophysics experiments, visual neuroscience, and eye-tracking studies.

**Goals**

* Dynamic 3D visual stimuli (colored spheres).
* Eye position–dependent stimulus rendering.
* Pupil size collection using Fove SDK.
* Event and data streaming via LSL.
* Fixation light for gaze stability.
* Adjustable luminance and positions.Hypothesis

**Scope**

1. Open the project in Unity.
2. Set up LSL:
   1. Import the LSL C# bindings into the Unity project (liblsl-Csharp).
   2. Ensure LSL runtime is installed and available on your system path.
3. Set up Fove SDK:
   1. Install Fove Unity SDK.
   2. Add the FoveInterface prefab to your scene.
4. Scene Setup:
   1. Add the StimuTest and FoveBackgroundController scripts to an empty GameObject.
   2. Assign necessary parameters in the Inspector (or use defaults).
5. Play the scene in the Unity Editor.
6. After the initial delay (WaitForSec), stimulus presentation will begin.
7. Real-time pupil data and event markers will be sent via LSL.

Or

1. From Unity: file open Scene select pupilometerTest with all the above already configured.

**Requirements:**

* Unity 2020+ (recommended)
* [Fove Unity SDK](https://developer.fove.io/)
* [Lab Streaming Layer (LSL)](<https://github.com/sccn/labstreaminglayer>) (LabRecorder.exe)
* Fove VR headset (with runtime installed) and updated license

**Scripts Breakdown:**

FoveBackgroundController.cs

Background Controller

* foveCamera.backgroundColor = backgroundColor \* luminance;

Parameters (Editable in Inspector – under Fove Interface)

| **Parameter** | **Description** | **Default** |
| --- | --- | --- |
| backgroundColor | Background color | blue |
| Luminance | Light level | 0.3 |

StimuTest.cs

Stimuli Presentation

* Red and blue spheres are presented at 5 default locations:
  + Center, Left, Right, Up, Down.
* Red/Blue spheres: 0.5s on, 3s off.
* Long blue: 8s on/off, repeated multiple times.

Fixation Light

* White sphere centered 8.5 units forward from Fove headset.

Eye Tracking

* Captures pupil radius from left and right eyes.
* Sends data every frame using LSL.StreamOutlet.

Event Markers

* Start/Stop events for stimuli are timestamped and streamed via LSL.

Parameters (Editable in Inspector – under Stimulation)

| **Parameter** | **Description** | **Default** |
| --- | --- | --- |
| stimulationDurationRedBlue | Duration (sec) red/blue spheres | 0.5 |
| intervalDurationRedBlue | Delay between red/blue spheres | 3.0 |
| stimulationDurationLongBlue | Long blue duration (sec) | 8.0 |
| intervalDurationLongBlue | Delay between long blues | 8.0 |
| focalBlueLightIterations | Repeats of long blue | 3 |
| WaitForSec | Initial wait before first stimulus | 30 |
| blueCircleSize | Blue circle size | 0.1 |
| redCircleSize | Red circle size | 0.1 |
| blueLuminance | Blue luminance | 0.3 |
| redLuminance | Red luminance | 0.3 |
| LongBlueLuminance | Long Blue Luminance | 6000.0 |
| fixationLightSize | Fixation Light Size | 0.09 |
| fixationLightLuminance | Fixation Light Luminance | 0.5 |
| WaitForSec | Seconds to wait before starting the stimulation | 30 |
| vectorPositions | Default lights positions | (0,0,2),(-0.73,0,2),  (0.73,-,2),(0,0.73,2), (0,-0.73,2) |

**LSL Streams**

* Stream 1: UnityPupilData (float32[2])
  + Left and right pupil radius in mm.
* Stream 2: UnityEventStream (string)
  + Event markers: e.g., "Start Red sphere for 0.5 seconds in (0, 0, 2)"

**Stimulation session setup instructions:**

1. open unity hab

2. open pupilometer

3. go to lab recorder under - C:\Users\GuestVR\Downloads\LabRecorder-1.16.4-Win\_amd64\LabRecorder

4. open LabRecorder.exe

5. Change study root to D:\Pupilometer Data

6. Update Participant number

7. Click start

8. go to the unity project and press the start bottom

9. go back to LabRecorder and press update

10. mark all the streams

11. stop the LabRecorder after the stimulation is over

There is a recording with the instructions here - D:\Pupilometer Data\Guides\ Pupilometer\_video\_guide.mp4

**Future Evolutions:**

Real-Time Dashboard - Add a real-time UI showing:

* Current stimulus
* Eye status (validity)
* Retry in case of eye tracking failure
* Pupil size live values
* Control left or right eye
* Export eye's data as CSV file according to the current Shiba format

**Post-Experiment Data Viewer (Python Script)**

This Python utility allows researchers to visualize and inspect pupil data and event markers recorded during a stimulation session. It loads XDF files recorded via LabRecorder, extracts relevant streams, and plots the left and right pupil size over time, with event markers clearly overlaid for reference.

**Features**

* Loads .xdf recording files using a file dialog.
* Extracts and plots left/right pupil size data (from UnityPupilData stream).
* Overlays event markers (from UnityEventStream) as vertical lines.
* Interactive hover tool shows marker labels (e.g., “Start Red sphere…”).
* Useful for data analysis.

**Requirements:**

* Python 3.13
* pyxdf, numpy, matplotlib, mplcursors, tkinter (built-in)
* Install dependencies (if needed): "pip install pyxdf matplotlib mplcursors"

**File Selection:**

A file dialog will open upon script execution — select your .xdf file (e.g., from D:\Pupilometer Data).

**Output:**

* Interactive plot of pupil size over time.
* Vertical dashed lines for stimulus events.
* Cursor hover reveals the label of each event marker.
* תמונה שמכילה טקסט, קו, עלילה, תרשים

  תוכן בינה מלאכותית גנרטיבית עשוי להיות שגוי.

**Example Use Cases:**

* Verify pupil response according to different stimulations.
* Check data quality and stream presence.
* Confirm correct stimulus timing and recording start/stop points.

**Usage Flow**:

1. Run the script (python resultCalculator.py)
2. Select an .xdf file from the dialog
3. Plot will automatically render with pupil traces and events

**Author**

* Name: *Yovel Pando and Roni Sova*
* GitHub: https://github.com/yovelP/Popilometer-VR