System Manual

0.1 Overview

There are two separate branches of this system. CAVE and HMD. the CAVE branch is more developed, consisting of two main components: a Unity 3D project written in C#, and a TCP client program written in C++.

The code is located publicly on a GitHub repository[1]containing the three programs outlined bellow, as well as supporting material such as EyeLink documentation and the required library and header files for interacting with the EyeLink.

0.2 CAVE System

The CAVE system incorporates full visual correction for a diplopia suffer using a Cave Automatic Virtual Environment and the EyeLink II eye tracker. The CAVE uses a combination of Unity 3D (Ver. 5.3.4) & MiddleVR (Ver. 1.6.1) on one PC to render the environment, and DTrack 2 & VRPN on another to handle the head tracking of the user. The EyeLink II tracks the gaze of the user which is sent to the Unity application. The gaze of the user is modelled, and the rotation correction required is calculated and applied to the set of View Planes that project to the misaligned eye within the CAVE, correcting the user's vision within the virtual environment.

0.2.1 Technical Specification

The following is the technical specification used in the creation of the system. It is the recommend, but not minimum specification needed.

[1]https://github.com/SnubbleJr/DiplopiaCorectionViaVE

Rendering PC

- Intel Core i7-3930K CPU, 3.2GHz
- 32Gb RAM
- Nvidia quadro K5000 GPU

Projectors

- Graphics setup is for 5600x1050 display with 96Hz refresh rate
- Display is divided into three 1400x1050 projections for each of the vertical walls, and 1100x1050 for the floor (the floor projection is clipped at the sides so that it is approximately square)
- Shutterglasses used are Volfoni ActivEyes

ART tracking

- Trackpack4 (6 cameras) optical system

EyeLink II

- An EyeLink II eye tracking head set and accompanying host PC & IR markers are required.

(See accompanying EyeLink user guide for more information [1].)

Experiment computer

- A computer with two network cards. This is needed for the transferring of data from the EyeLink II host computer to the computer used for rendering inside the CAVE.

Misc

- A sever on which VRPN and DTrack is needed to run

0.2.2 Set Up

Further details for setting up a CAVE will not be given; access to a CAVE is assumed henceforth.

Experiment & host PC

The computer used to forward data between the host PC and render PC - referred to as the experiment PC by the EyeLink - requires its monitor to be outlined with the supplied IR markers. This is for the calibration and validation of the eye tracker. This computer also requires two network cards. Ensure one is set to the static IP address 100.1.1.2, and subnet mask 255.255.255.0. Connect the host PC via Ethernet cable to static networking card of the experiment PC.

Connect the EyeLink II with the host PC as per the instructions of the user manual (a copy of which is in the EyeLink Supporting Material folder).

Two separate programs are run on experiment PC, these are located in the 'EyeTracker for CAVE' directory. The directory labelled 'Calibration and Validation' contains the example program 'simple.exe', supplied by SR Research used for the calibration and validation of the eye tracker, and not extended upon.

Within the directory 'Send data to cave' is the Visual Studios solution 'simpleexample.sln'. This solution forwards data from the host PC to the Unity project running on the rendering PC. The EyeLink API [2] can be found in the supporting materials folder.

Note: errors about the dependencies of the header and lib files may occur, make sure the relative file paths correctly link to the 'includes' and 'libs' directories supplied in the 'EyeTracker for CAVE' directory. Copies are also included in the supporting material folder.

Unity Project

Located within the 'CAVE' directory, the unity project root is labelled 'EyeDisplacement'. The MiddleVR asset packed must be included within the Unity project. Opening the scene 'scene.unity' will present several game objects.

The game objects 'VRManager', 'GazeCaster' and 'Shift' must be kept in the scene. Make sure the correct configuration file ('CAVE/config.vrx') is selected by the VR Manager.

GazeCaster is the main object used for receiving EyeLink data and simulating the user's gaze. The GazeCasterManager script is used to customise the experiment

to be run. Here, the dominant eye of the participant can be set, as well as the option to log their gaze during the experiment. This data is contained within the 'Logs' folder of the build's data directory, containing the rotation of the user's eyes as well as the correctional rotation needed for an object at a specified depth.

The type of drift correction can be set to reflect the type of experiment that is to be run. Binoculars should be used for fully sighted participants, while monocular is intended for the visually impaired. The (x, y) offset of the correction marker can is modifiable in the inspector.

The use of test inputs or outputs can also be set; mimicking the gaze input that would be sent from the EyeLink with a controller, or visualising the resultant screen rotation that would be output.

Note: make sure the IP addresses set in the TCPListener script on GazeCaster and in simpleexample.sln on the experiment PC are set to the address of the rendering PC.

0.2.3 Run Order

Outlined is the basic operational order of an experiment. This is further outlined in the User Manual

1. Activate DTrack and VRPN server - MiddleVR should be able to pick up head tracker movement of glasses

2. Run simple.exe, calibrate and validate EyeLink. Exit after good calibration achieved

3. configure experiment within Unity, build and run

4. Run simplExample, data should now be sending, gaze should be tracked by yellow sphere

5. Perform experiment. Data can be logged once wand button 2 (defined in MiddleVR) is pressed

6. Perform drift correction if needed

7. Once finished, press space bar on experiment PC to stop sending to unity, stop recording EyeLink data

0.3 HMD portion

Located with the 'HMD' directory is the Head Mounted Display branch of the system. This system is capable of view plane translation and independent rotation and translation of each stereo half camera. Although not implemented, the addition of view plane orientation is well within scope.

0.3.1 Technical Specifications

HMD

- An Oculus Rift DK2 was used in the development of this system. As Unity 3D (Ver. 5.3.4) was used, this specific HMD is not required, and any compatible display should be adequate.

Computer

A powerful enough computer is required to run VR based Unity applications, the following is the recommenced, but not the minimum specification to run tests on

- Intel Core i7-4790k CPU, 4.4Ghz
- 16 GB DDR3 2133MHz RAM
- Nvidia GTX 970 4GB GPU

0.3.2 Set Up

The set up for this system is straightforward.

Set up the HMD to such that it can be recognised with Unity. Two scenes are supplied. 'Experiment.unity' is used for running experiments. 'Normal cross scene.unity' is used for testing and checking to see if the individual camera set up generates the same images as unity's built in stereo camera.

Within the scene Experiment are two game objects that are required to run an experiment: 'ModePicker' and 'Camera Rig'. One of two experiments can be run

with this system - the flag 'Shift Texture' on ModePicker indicatedswhether view port translation or camera translation & rotation is performed.

The experiment can be run directly within the editor. Operational instructions for the user are located in the User Manual.

The texture shift experiment will yield data of the correctional shift required to align objects in a CSV format. The camera shift will not. The code could be easily extended to implement this however.