Magic Xroom

Data Collection guide
17.11.2023

Table of Contents

Introduction

This document contains important information regarding the Magic Xroom, its inner working and output, with the goal of facilitating its use for collecting data through external sensors.

Compatibility and software requirements

The Magic Xroom is compatible only with Windows 10+ x64 operating system.

To run the application, additional software must be installed and configured prior launching the Magic Xroom executable, depending on which external sensors will be used during the data collection.

The minimum requirements in terms of hardware sensors are a Virtual Reality headset and Virtual Reality controllers. For this the software required is:

- SteamVR (downloaded with Steam)
- VR headset linking software, i.e.
    - VIVE Focus 3 requires VIVE Business Streaming
    - Oculus Quest require the Oculus Desktop App

To use a Shimmer 3 GSR+ device:

- No additional software required, but the sensor must be configured with the *LogAndStream* firmware. Refer to the official documentation
- Bluetooth 5.1+ availability required
- Shimmer sensor must be paired with the computer prior to starting the application

To use face/eye tracking:

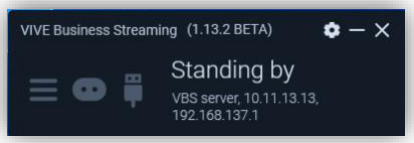
- SRanipal runtime (included in VIVE Console for SteamVR on Steam)
- The computer and the VR hardware must be connected on the same 5GHz WiFi network

Using the Magic Xroom

The following are the recommended steps to successfully launch and use the Magic Xroom with a VIVE Focus 3. Be advised that different methods might work but it is up to the user to choose which to follow and ensure the software works as intended.

Note: software versions shown in images might differ

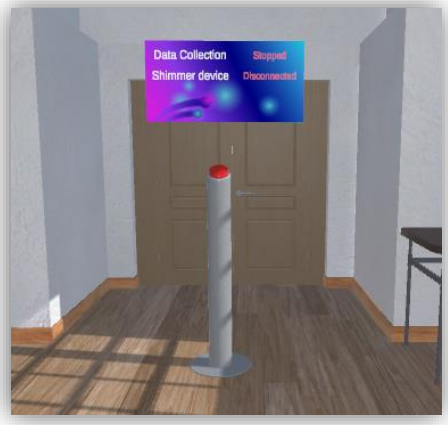
1. Download the required software and setup the hardware
    a. Link the VR headset and controllers, pair the Shimmer sensor, connect to a strong WiFi if eye/face tracking is enabled
2. Launch *VIVE Streaming Business*



3. Launch *VIVE SRanipal* (Steam version) if eye/face tracking is enabled (it launches automatically at point 5. but this step ensures the correct version is used in case multiple versions are installed on the computer)
    a. Default install directory is `<steam install directory>\steamapps\common\VIVEDriver\App\SRanipal`
    b. When successfully connected with the eye tracking accessory the tray icon should change color (same for mouth tracking)

4. Configure the user settings if necessary (see chapter User Settings configuration)
5. Launch the Magic Xroom executable (SteamVR starts automatically)

At this point the user should be within the virtual world of the Magic Xroom.
In order to **start the data collection** one last step is required:



The button shown in the above picture is used to start/stop the data collection. The panel above it shows the current state of the data collection [Stopped / Running] and the current state of the Shimmer device [Disconnected / Connecting / Connected / Streaming]. If the Shimmer device is used, wait until it shows a Connected state before starting the data collection and wait for it to turn into a Streaming state, otherwise no data will be collected for this device.
It is possible to work on multiple sessions without restarting the application by starting/stopping the data collection. Each time a new data collection is started a unique set of files is generated.

User Settings configuration

The Magic Xroom application can be configured with an external file to enable/disable some features or to tweak some parameters related to the data collection.

The configuration file can be found in the same directory as the executable
`<Magic Xroom directory>\xr2learn-magicxroom_Data\StreamingAssets\UserSettings.xml`

The `UserSettings.xml` file contents are the following:

```xml
<?xml version="1.0" encoding="utf-8" standalone="yes" ?>
<settings>
    <dataCollection autoStart="false" outputPath="" />
    <vr>
        <config samplingRate="10" samplingBufferSize="50" />
    </vr>
    <shimmer enabled="true" deviceName="Shimmer3">
        <config heartbeatsToAverage="1" trainingPeriodPPG="10" samplingRate="10" samplingBufferSize="50" />
        <sensors enableAccelerator="true" enableGSR="true" enablePPG="true" />
    </shimmer>
    <eyeTracking enabled="true">
        <config samplingRate="10" samplingBufferSize="50" />
    </eyeTracking>
    <faceTracking enabled="true">
        <config samplingRate="10" samplingBufferSize="50" />
    </faceTracking>
    <feedback enabled="true" afterScenario="false" afterLevel="true" />
</settings>
```

- **dataCollection**
  - **autoStart**[1] [true/false] => enables disables the automatic start of the data collection when the application starts.
  - **outputPath**[2] [Windows compatible directory] => specifies the location where the data collection files will be generated. If left empty the default location is the same as the executable file.
- **vr**
  - **config**
    - **samplingRate**[3] [integer] => the frequency in [Hz] at which the application will sample data from the VR headset and controllers
    - **samplingBufferSize**[3] [integer] => the number of samples buffered before writing to file.
- **shimmer**
  - **enabled** [true/false] => enable/disable the Shimmer sensor data collection
  - **deviceName** [any] => the Shimmer device internal name
  - **config**
    - **heartbeatsToAverage** [integer] => the number of heartbeat inputs used to calculate an average. Higher values tend to generate better results but sometimes break due to the volatility of the Bluetooth communication, i.e. if the value is set to 10 and one of the 10 inputs is corrupted or invalid, the overall average will be invalid for the next 10 calculations.
    - **trainingPeriodPPG** [integer] => the delay in [s] before the PPG signal is able to calculate a heartbeat
    - **samplingRate**[3] [integer] => the frequency in [Hz] at which the application will sample data from the Shimmer sensor
    - **samplingBufferSize**[3] [integer] => the number of samples buffered before writing to file.
  - **sensors**
    - **enableAccelerator** [true/false] => enables/disables the Shimmer internal accelerator sensor
    - **enableGSR**[true/false] => enables/disables the Shimmer internal GSR sensor
    - **enablePPG** [true/false] => enables/disables the Shimmer internal PPG sensor

- **eyeTracking**
  - **enabled** [true/false] => enables disables eye tracking
  - **config**
    - **samplingRate**[3] [integer] => the frequency in [Hz] at which the application will sample data from the eye tracking sensor
    - **samplingBufferSize**[3] [integer] => the number of samples buffered before writing to file.
  -
- **faceTracking**
  - **enabled** [true/false] => enables disables face tracking
  - **config**
    - **samplingRate**[3] [integer] => the frequency in [Hz] at which the application will sample data from the face tracking sensor
    - **samplingBufferSize**[3] [integer] => the number of samples buffered before writing to file.
  -
- **feedback**
  - **enabled** [true/false] => enables disables the feedback feature
  - **afterScenario** [true/false] => show the feedback panel at the end of a scenario
  - **afterLevel** [true/false] => show the feedback panel at the end of each level

***Important Note**: in case the `UserSettings.xml` file is not found in the specified directory or in case of any erroneous or missing data, the application will use any or all the default values shown in the above image and in original file provided with the executable*

[1] An interactable object that controls the start/stop of the data collection is present in the VR environment and it is the suggested way to control this feature.

[2] The Windows user **must** have permission to read/write in the specified directory

[3] To calculate the delay Δt in [s] between each write operation for a specific sensor use the following formula:

$$\Delta t = \frac{samplingBufferSize}{samplingRate}$$

Higher values will impact less on performance but will result in more data being lost in case of abrupt interrupts in the data collection.
Example: with a ***samplingBufferSize*** of 50 and a ***samplingRate*** of 10Hz the resulting delay will be 5s, meaning that at most 5s of data might be lost in case of unexpected errors or problems with the application/hardware/computer.

Magic Xroom output

The output of the Magic Xroom is a set of files containing the data collected by the sensors enabled prior to launching the application.

Each of the enabled sensors will generate a CSV file sharing a unique identifier for the session, using the following naming convention:

`data_collection_<session ID>_<sensor type>.csv`

The session ID is generated as a number referring to the number of ticks since midnight 01.01.0001. Each tick represents 100 nanoseconds and to retrieve the corresponding date, input the number in an epoch converter (compatible with C# `DateTime`) or use the following website datetimetoticks-converter.com.

VR

`data_collection_<session ID>_**VR**.csv`

Contains the data collected from the Virtual Reality Headset and Controllers.
The columns represent the position and rotation of the headset and controllers:

- `timestamp` => application timestamp
- `head_pos_x` => headset absolute position x axis
- `head_pos_y` => headset absolute position y axis
- `head_pos_z` => headset absolute position z axis
- `head_rot_x` => headset absolute rotation x (quaternion)
- `head_rot_y` => headset absolute rotation y (quaternion)
- `head_rot_z` => headset absolute rotation z (quaternion)
- `head_rot_w` => headset absolute rotation w (quaternion)
- `lcontroller_pos_x` => left controller absolute position x axis
- `lcontroller_pos_y` => left controller absolute position y axis
- `lcontroller_pos_z` => left controller absolute position z axis
- `lcontroller_rot_x` => left controller absolute rotation x (quaternion)
- `lcontroller_rot_y` => left controller absolute rotation y (quaternion)
- `lcontroller_rot_z` => left controller absolute rotation z (quaternion)
- `lcontroller_rot_w` => left controller absolute rotation w (quaternion)
- `rcontroller_pos_x` => right controller absolute position x axis
- `rcontroller_pos_y` => right controller absolute position y axis
- `rcontroller_pos_z` => right controller absolute position z axis
- `rcontroller_rot_x` => right controller absolute rotation x (quaternion)
- `rcontroller_rot_y` => right controller absolute rotation y (quaternion)
- `rcontroller_rot_z` => right controller absolute rotation z (quaternion)
- `rcontroller_rot_w` => right controller absolute rotation w (quaternion)

Shimmer

`data_collection_<session ID>_**SHIMMER**.csv`

Contains the data collected from the Shimmer device.
The columns represent the values captured by the Shimmer sensors

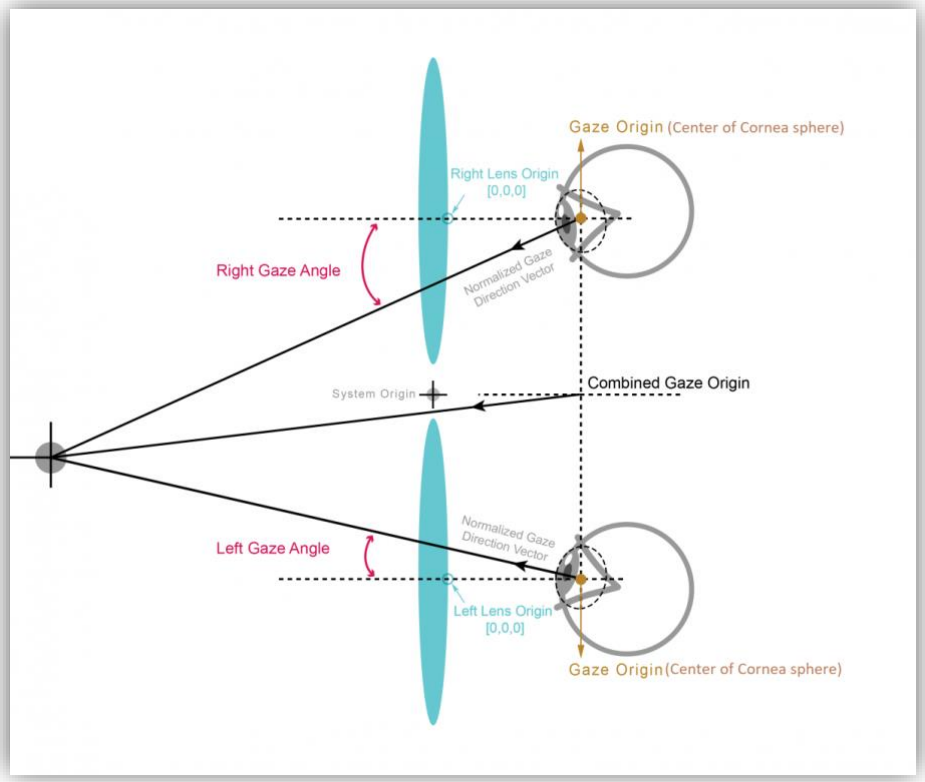
- `timestamp` => application timestamp
- `int_timestamp` => Shimmer internal timestamp
- `accel_x` => Shimmer accelerator x axis
- `accel_y` => Shimmer accelerator y axis
- `accel_z` => Shimmer accelerator z axis
- `gsr` => Shimmer Galvanic Skip Response (GSR) sensor
- `ppg` => Photoplethysmograph (PPG) sensor
- `hr` => hear rate computed from the PPG data

Eye Tracking

`data_collection_<session ID>_EYE.csv`

Contains the data collected from the Eye Tracking device (vectors are right-handed).
The columns represent the gaze, pupil and position of each eye

- `timestamp` => application timestamp
- `int_timestamp` => Eye Tracking device internal timestamp
- `left_gaze_origin_x` => left eye x cornea center relative to each lens center [mm]
- `left_gaze_origin_y` => left eye y cornea center relative to each lens center [mm]
- `left_gaze_origin_z` => left eye z cornea center relative to each lens center [mm]
- `left_gaze_dir_norm_x` => left eye gaze x direction normalized [0,1]
- `left_gaze_dir_norm_y` => left eye gaze y direction normalized [0,1]
- `left_gaze_dir_norm_z` => left eye gaze z direction normalized [0,1]
- `left_pupil_diameter` => left eye pupil diameter in [mm]
- `left_eye_openness` => left eye openness (0 closed, 1 open)
- `left_pos_norm_x` => normalized left eye pupil x pos relative to lenses (0.5,0.5 is center)
- `left_pos_norm_y` => normalized left eye pupil y pos relative to lenses (0.5,0.5 is center)
- `right_gaze_origin_x` => right eye x cornea center relative to each lens center [mm]
- `right_gaze_origin_y` => right eye y cornea center relative to each lens center [mm]
- `right_gaze_origin_z` => right eye z cornea center relative to each lens center [mm]
- `right_gaze_dir_norm_x` => right eye gaze x direction normalized [0,1]
- `right_gaze_dir_norm_y` => right eye gaze y direction normalized [0,1]
- `right_gaze_dir_norm_z` => right eye gaze z direction normalized [0,1]
- `right_pupil_diameter` => right eye pupil diameter in [mm]
- `right_eye_openness` => right openness (0 closed, 1 open)
- `right_pos_norm_x` => normalized right eye pupil x pos relative to lenses (0.5,0.5 is center)
- `right_pos_norm_y` => normalized right eye pupil y pos relative to lenses (0.5,0.5 is center)



Face Tracking

`data_collection_<session ID>_FACE.csv`

Contains the data collected from the Face Tracking device (vectors are right-handed).
The columns represent 27 facial points and how much these points are influencing the resulting facial expression

- `timestamp` => application timestamp
- `int_timestamp` => Face Tracking device internal timestamp
- `none` => no difference compared to the default shape
- `jaw_forward` => jaw position on the forward axis
- `jaw_right` => jaw position on the right side of the horizontal axis
- `jaw_left` => jaw position on the left side of the horizontal axis
- `jaw_open` => jaw openness
- `mouth_ape_shape` => mouth aperture shape
- `mouth_o_shape` => mouth O shape (i.e. while making an “O” sound)
- `mouth_pout` => mouth pouting shape
- `mouth_lower_right` => mouth lower right shift
- `mouth_lower_left` => mouth lower left shift
- `mouth_smile_right` => mouth smile shape right side
- `mouth_smile_left` => mouth smile shape left side
- `mouth_sad_right` => mouth sad shape right side
- `mouth_sad_left` => mouth sad shape left side
- `cheek_puff_right` => cheek puff shape right side
- `cheek_puff_left` => cheek puff shape left side
- `mouth_lower_inside` =>
- `mouth_upper_inside` =>
- `mouth_lower_overlay` =>
- `mouth_upper_overlay` =>
- `check_suck` => cheek “suck” expression
- `mouth_lower_right_down` =>
- `mouth_lower_left_down` =>
- `mouth_upper_right_up` =>
- `mouth_upper_left_up` =>
- `mouth_philtrum_right` =>
- `mouth_philtrum_left` =>

Progression Events

`data_collection_<session ID>_PROGRESS_EVENT.csv`

Contains the events representing the user interaction with the environment and the scenarios.
The columns represent the event type and the information relative to that event

- `timestamp` => application timestamp
- `event_type` => event type
- `info` => information relative to the event

The events generated fall in the following categories:

- **Scenario** => a scenario can either be started manually by the user, or is ended by completing all the levels or teleporting outside the scenario area. The `info` column represents the name of the scenario.
  - `SCENARIO_STARTED` => a scenario is manually started
  - `SCENARIO_ENDED` => a scenario is ended, either by competition or by leaving the scenario area

- **Level** => a scenario is composed of one or more levels. A level can be started, completed or failed. The `info` column represents the level difficulty as a number.
  - `LEVEL_STARTED` => a level is started
  - `LEVEL_FAILED` => a level is failed
  - `LEVEL_COMPLETED` => a level is completed successfully

- **Teleport** => the user uses a teleport feature to enter or exit a scenario area. The `info` column represents the name of the scenario.
  - `TELEPORT_IN` => the user was outside a scenario area and has teleported inside a scenario area
  - `TELEPORT_OUT` => the user was inside a scenario area and has teleported outside a scenario area

- **Feedback** => the user can provide feedback on his/her emotional state in the latest scenario or level. The `info` column represents the time in [ms] that the user waited before making a decision. The `SKIP` option is used for exceptional situations, i.e. the user entered a scenario area by mistake and exited it right away without starting a level.
  - `BORED` => the user was bored
  - `ENGAGED` => the user was engaged
  - `FRUSTRATED` => the user was frustrated
  - `SKIP` => the user skipped the selection

***Important Note**: The value of `timestamp` written in each of the files just mentioned is synchronized and taken from the same context. Although the application itself runs mainly on one thread, some of the sensors might have a delay in reading or providing data. When available, refer to both the `timestamp` and `int_timestamp` values to understand if the delay between the polling of data from a sensor and the writing to file operation should be taken into consideration or if it's negligible.*

Magic Xroom Scenarios

The Magic Xroom contains 4 scenarios meant to induce specific emotions and gathering data from external sensors. Each experience is composed of increasingly difficult tasks that require various skill levels to complete before a given time limit.
Some of the experiences are positioned on desks that can be adjusted in height with a handle.

Stacking Cubes

Stacking Cubes is an experience in which the user is tasked with positioning 4 cubes on top of each other within a time limit. The cubes properties vary between levels:

1. The cubes have no particular property and are the same size
2. The cubes are slippery
3. The cubes are bouncy
4. The cubes are of different size and must be stacked from small to big
5. The cubes are the same size but an external force (wind) is making it difficult to stack them straight

Color Words

Color Words is a fast-paced experience in which the user is tasked to select (touch) one of the cubes on the desk depending on its color. The correct color is shown on the screen as a word describing the color but colored with a different one. For example, if the word **`yellow`** appears the user must select a yellow cube, not a red one. Multiple cubes of the same color might appear. As the game progresses the number of cubes available increases and the time available to make a decision decreases, to the point where there isn't enough time to select a cube and the experience ends.

Canvas Painting

Canvas Painting is a painting minigame where the user is tasked with drawing a specific shape without exiting a given area.

Tower of Hanoi

Tower of Hanoi is a relatively famous game. The version used in the Magic Xroom is composed of 3 rods and 3 discs. The rods are of different size: the left-most can hold 3 disks at the same time, the middle one 2, and the right-most only 1.
The user is presented with two sets of rods and disks. The set closer to the user is the interactive one (interaction set), while the other one represents the target configuration (configuration set). At the beginning of each level the configuration set shows the target configuration and the user must try and match it with the disks in the interaction set. The moves available are limited (depending on the level and difficulty) and a time limit is displayed on the screen.