Delayed Feedback based Immersive Navigation Environment (DeFINE) -
User Manual

Kshitij Tiwari *

October 7, 2019

Contents

*Kshitij Tiwari is with the Department of Electrical Engineering and Automation, Aalto University, Espoo, 02150, Finland. Correspondence:kshitij.tiwari@aalto.fi

List of Figures

1 About

This document is intended for anyone using the Delayed Feedback based Immersive Navigation Environment (DeFINE). DeFINE is a system for creating immersive environments for the purpose of running human navigation experiments in virtual reality (VR). This document instructs the users in obtaining the software, installing of the required tools, and using the software.

In what follows, the software, hardware, and the physical requirements are explained. Then, this document provides an overview of the experiments one can perform using the environment. Next, there is a detailed explanation of the user interfaces and settings files used by the environment. Lastly, the data that is collected during the experiments is described.

2 Setting up the environment

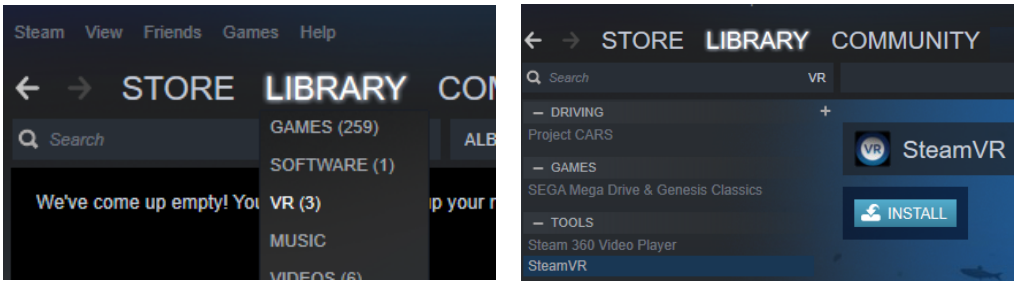
This section gives instructions on how to download and setup the environment. The installation and setup procedure explained here is current as of 18 September 2019, for Windows 10 version 1809 and SteamVR version 1.3.23.

2.1 Installing SteamVR and VR hardware

In order to use the environment with virtual reality (VR) equipment, SteamVR needs to be installed. SteamVR can be installed through Valve Corporation's videogame platform Steam. A free Steam account is needed for installing the SteamVR tool. SteamVR can be found in the Steam client by selecting Library > VR, as shown in Fig. 1a. This brings up a listing of all VR titles available to the account through Steam. SteamVR is installed by selecting it under the tools subcategory of the list of titles, and by clicking the *Install* button, as shown in Fig. 1b.

To install the VR hardware, the installation instructions provided by the manufacturer of the hardware should be followed. Supported hardware are hardware supported by SteamVR. As of 18^{th} September 2019, this list includes HTC Vive, Oculus Rift, Valve Index, and Windows Mixed Reality devices.

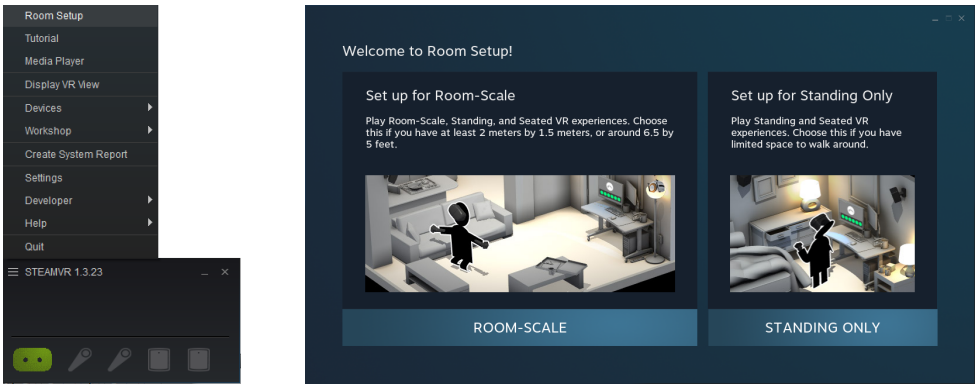
After connecting the hardware and installing all the potential drivers required by the hardware manufacturer, SteamVR play area needs to be setup. To do so, start SteamVR through the Steam platform. From the



(a) Library subcategory selection. (b) The SteamVR in the Library.

Figure 1: Finding and installing the SteamVR in Steam.

menus, of the window that comes up, select "Room setup", shown in Fig. 2a. A window like the one shown in Fig. 2b should appear. If there are plans of using physical walking as the locomotion method, the user should ensure that adequate physical area is cleared of obstacles, and select "Setup for Room-scale". Otherwise "Setup for standing only" is sufficient. The window will provide instructions until setup is complete.



(a) The SteamVR menu. (b) Room setup window.

Figure 2: Configuring the room boundaries via SteamVR.

2.2 Downloading the files from Git

The project is hosted on a git repository[1]. In the repository, there are two **branches**, *master* and *development*. The *master* branch contains a pre-built, stable version of the environment, ready to be used by experimenter. The *development* branch contains the source files of the currently developed version.

To download the files, one can press the small "download" icon, found above the file listing, on the right, as shown in Fig. 3. After downloading, extract the package. Alternatively, clone the git repository. Inside the extracted files of the master branch, one should find a file named "VR-environment.exe". Running this file starts the experiment environment.

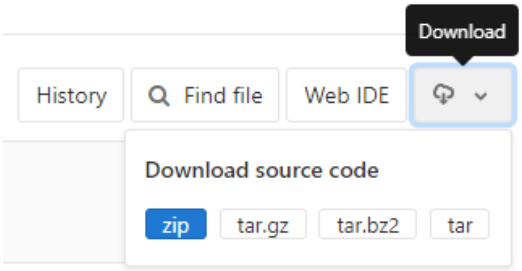


Figure 3: The download button in Git repository.

2.3 Configuring SteamVR Inputs

When starting the environment for the first time, SteamVR might ask the user to configure any inputs that the environment uses but have not been setup for your Virtual Reality equipment yet. Notification about this should be visible on the Head-Mounted Display (HMD). Typically, when there are menus or notifications displayed on the HMD, the environment will look out of focus or blurry on the monitor. If the locomotion methods do not work when starting the environment for the first time, or after changing the button configuration, please check that the inputs are properly assigned in SteamVR.

[1] Available at https://gitlab.com/aalto-qut/environment

3 Requirements

This section describes the hardware and software requirements to ensure a smooth execution of the VR environment. Setups not meeting these requirements might function slowly, and cause lag in the HMD, which is known to cause simulation sickness in some participants. In such cases, the experimenters should proceed with caution and at their own discretion.

3.1 Hardware requirements

Usage of the Virtual Reality environment requires an adequately high performance computer with at least one monitor, a keyboard, and a mouse connected, and the SteamVR-compatible Virtual Reality equipment(s).

	Recommended	**Used in development**
Graphics Card	GTX 1060 or better	GTX 1070
CPU	Intel Core i5-4590, equivalent or better	Intel Core i5-9600K
RAM	4GB or more	32GB
Motherboard	One with available PCIE port (For Vive Wireless Adapter)	Asus ROG STRIX Z390-H GAMING
VR Equipment	Any SteamVR compatible	HTC Vive Pro

3.2 Software requirements

The environment has some software dependencies, listed in the table below. These listed dependencies are based on the system used in development. Different versions, newer ones in particular, may work, but their correct functionality is not guaranteed.

	Recommended	**Used in development**
Operating System	Windows 10	Windows 10 Home Version 1809
SteamVR	$\geq$Version 1.3.23	Version 1.3.23

3.3 Physical requirements

For safe experimentation, a room with sufficient clearance around the participant wearing the VR HMD is required. If the chosen SteamVR-compatible Virtual Reality equipment has beacons or other trackers, they should be mounted according to the manufacturer's instructions to ensure error free tracking.

4 Overview of the environment

The experiment environment is intended for studying human navigation from a start position to a goal position. The predefined environment is intended for studying human navigation based on noisy visual cue, whereas the noisy cue is implemented by having a fly buzz around the otherwise invisible goal position. Other types of environments can be defined as needed by the experimenter.

The participant starts from starting position and their objective is to move as close to the goal as possible, as fast as possible. The trials are scored based on both speed and accuracy at which the participant finds the goal position.

4.1 Experiment scenarios

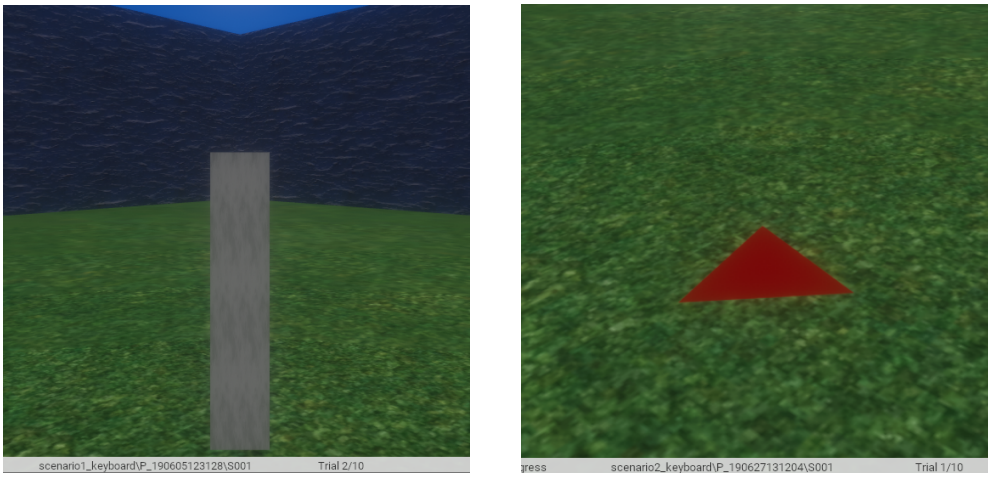
The experiment environment is designed with three navigation experiment scenarios in mind, all of which consists of practice and test phases. The purpose of the practice trials is to get the participant familiarized with the environment, locomotion method, and their objective in the environment. The test trials are intended for testing the participant's navigation performance.

4.1.1 Scenario 1

Scenario 1 has one starting position, with an oriented box in front of it in practice trials to allow orientation to the participant, as shown in Fig. 4a. There are two goal positions, and the participant has to get as close as possible to either one of them.

4.1.2 Scenario 2

Scenario 2 has two starting positions, and only one goal position. The starting position to be used is picked at random at the beginning of each trial. The starting position have an identifying icon on the floor in front of them, that can help the participant to understand the difference between them, as shown in Fig. 4b. Ideally the one goal position is to be located in such an asymmetric way, that the distances and directions to it are different between the two starting positions.



(a) Scenario 1, the oriented box visible. (b) Scenario 2, the unique icon visible.

Figure 4: The landmarks of the starting positions.

4.1.3 Scenario 3

Scenario 3 has only one start position and only one goal position, and no landmarks. It is intended for studying the effect of weighting of the score on the navigation speed and accuracy.

4.2 Experimenter view and participant's view

All the user interfaces of the experiment environment are visible on screen only: the participant wearing the VR HMD does not see them. On the screen, the experimenter can see everything the participant is seeing through the HMD, but also the status bar in the bottom of the screen.

The status bar contains information about the current status of the trial, the path where the experiment data is being stored inside the folder where the participant list was, the current trial number and the total number of trials, and the current block or experiment phase. If the experimenter presses the *Notes* button in the left corner of the status bar, a window for taking notes about the session and marking the session as bad shows up. These notes are stored in the *notes.json* in the experiment data. Marking a session as bad sets a boolean variable in the notes file to true, making it easy to check and ignore bad session while analysing the experiment data. An example of this note window is shown in Fig. 5.

The only interface visible to the participant during the trials is a small panel visible above their center of vision, as seen in Fig 5. This panel, shown in Fig. 6a shows them a timer, displaying the remaining time for that particular trial, and the score of the previous trial. This panel serves two purposes: it tells the participant how well they did in the previous trial by having the score visible, and it urges them to be as fast as possible by providing a sense of urgency with the timer visible and constantly ticking down.

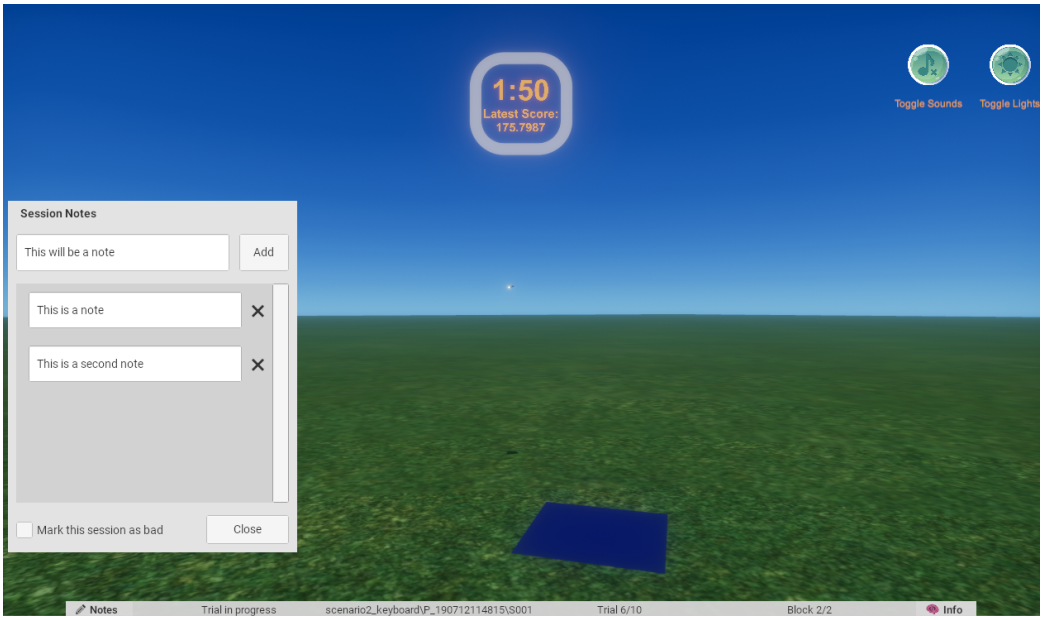


Figure 5: Experiment in progress, from experimenters point of view.

(a) The timer. (b) Buttons for controlling light and sounds.

Figure 6: Timer panel and environment buttons of DeFINE.

The environment has buttons, shown in Fig. 6b, for toggling all sounds and lights on or off. The status of the buttons are logged with the participant movements. These buttons are only visible to the experimenter on the computer monitor, and not in the HMD.

The experimenter has a number of buttons available during the experiments, as highlighted in red in Fig. 7.

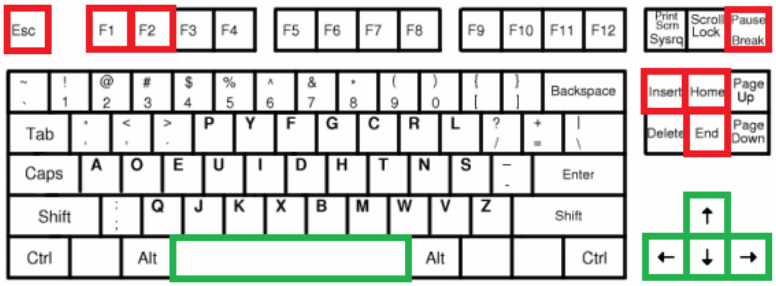


Figure 7: Keyboard inputs, red ones are for experimenter, green are for keyboard locomotion.

- Esc: Aborts current session, marking it as bad.
- F1: Displays / hides the notes window.
- F2: Displays / hides the experimenter button info window, that shows the same information as this list on the monitor.
- Pause: Pauses the current trial.

- Insert: Copy-pastes the current participant ID to the selected field in forms. Forms are explained in detail later in this section.
- Home: Activates second camera for screen rendering. The purpose of this is explained in detail in the implementation section of this thesis.
- End: End the current trial.

4.3 Locomotion in the environment

The experiment environment supports a variety of different locomotion methods which are explained below. All but the keyboard locomotion use the gaze direction of the participant measured via the HMD to determine the forward direction. The VR controllers are used by all locomotion methods, except keyboard locomotion. The common button used by all the locomotion methods using the controller is the *menu button*, which is used for ending the trials, skipping the scoreboard to move onto the next trial when it is visible, and for clicking on the pointed at location in the forms when they are visible. The *trackpad* is used for scrolling the forms up and down by pressing the upper or lower edge of the trackpad, respectively.

4.3.1 Keyboard locomotion

The keyboard locomotion is the only one of the supported locomotion methods that do not require the Virtual Reality hardware to use. In keyboard locomotion the participant is moved forward and rotated based on the presses of the arrow keys on the keyboard. Spacebar is used for ending trial when the participant is satisfied with their position. The buttons can be reconfigured through Unity's input configuration if needed. The default buttons are highlighted with green in Fig. 7.

4.3.2 Controller locomotion

In controller locomotion method, the participant uses the trigger of the controller to control the forward moving speed. The direction is always in the direction the HMD is pointing, but additionally using the left and right sides of the trackpad for turning counter-clockwise and clockwise respectively, can be turned on via the settings file. Holding the grip button while pulling the trigger will make the participant to move backwards instead of forwards.

Fig. 8 shows the HTC Vive Controller and its buttons as an example, though other VR controllers supported by SteamVR can be used. The HTC Vive comes with 2× controllers which are treated identically, i.e., the participant can use either of the controllers to press the buttons, and the experiment can be run using only one controller, with the exception of using the *armswing locomotion*.

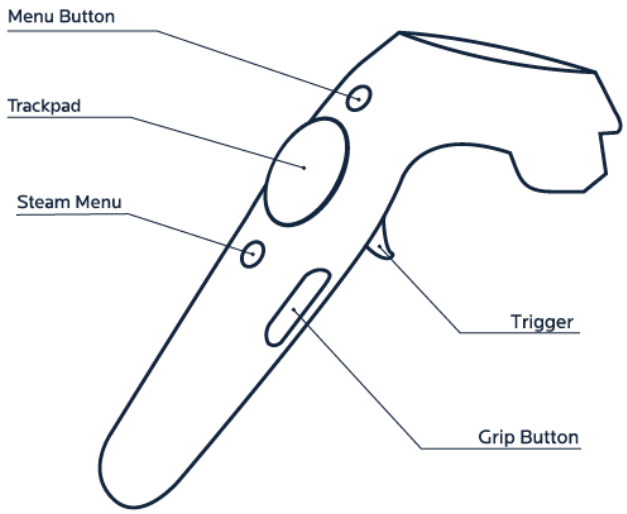


Figure 8: The HTC Vive Input buttons.

4.3.3 Teleport locomotion

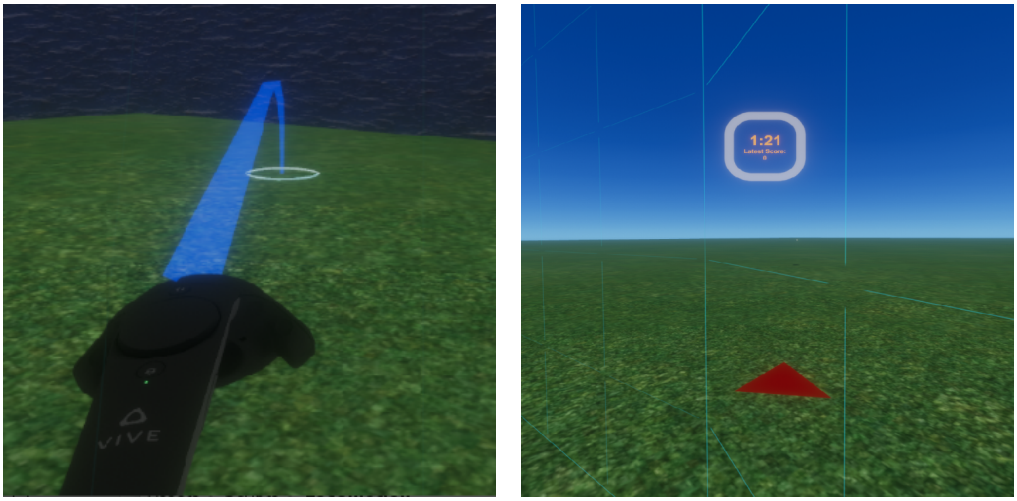
In teleport locomotion, the participant uses the controller to point to a desired direction while pulling the trigger of the controller. As shown in Fig. 9a, a visualization of the location to be teleported to is visible when the trackpad is being pressed, indicating with blue color if the pointed location is valid for teleportation, and with red if it is not. Once the participant releases the trigger, they are teleported to the marker, if the location is valid.

4.3.4 Physical walking locomotion

In physical walking locomotion, the participant physically walks in the desired direction. If the participant walks close to the boundary of the configured safe, clear area, SteamVR will make the edges of the area visible as

a warning to the participant, as shown in Fig. 9b. To continue in a direction despite the limitation of the available physical area, the participant can lock the virtual environment by holding the trigger of the controller. While locked, the participant can rotate themselves physically, while the rotation in the virtual environment remains locked. Locking also forces the configured safe area to be visible, allowing the participant to rotate themselves so that they face the safe area before continuing. Releasing the trigger releases the virtual environment to follow participant rotation as usual.

Using the locking method can have an impact on participant's perception of the environment. It is advisable to run experiments in large enough space that the need to use the feature can be avoided or at least minimized.



(a) The teleportation marker. (b) The border of the safe area.

Figure 9: The visualization markers in the HMD.

4.3.5 Armswing locomotion

In armswing locomotion the participant moves by holding both controllers, and walking in place, including swinging their arms. The movement is calculated based on the movement of the handheld controllers. The direction of the movement will always be the direction the participant is facing with the HMD.

4.3.6 Headbob locomotion

In headbob locomotion the participant moves by walking in place. The movement is calculated based on the vertical "bobbing" of the HMD. The direction of the movement will always be the direction the participant is facing with the HMD.

4.4 Forms in VR

The participants can be asked to fill some forms after each block. The forms are loaded from online sources, and shown in the virtual environment via an in-world web browser, in which the participant can input their answers to multiple choice questions using the VR controllers. For open questions, the participant can say their answer out loud, allowing the experimenter to type their answer using the keyboard. The typed text is visible to the participant, and so they can be certain that their answer was typed as they said it. When the forms are active, the controllers will project a blue laser, that the participant uses to point at the wanted position of the forms. The participant can click to the pointed at location on the form by pressing the menu button, which is the same button as is used for ending the trials. Pressing the upper or lower edge of the trackpad allows the participant to scroll up or down on the form, respectively. The participant ID can be entered into a selected text field in the forms by pressing the grip button of the controller. If there are multiple forms to be filled, the next form is automatically loaded when the previous one has been submitted.

By default, the participant is expected to fill the forms using the VR controllers. However, if it is necessary for the experimenter to be able to use the mouse to fill the forms on the screen, this can either be activated from the environment settings to be always on for the duration of the forms, or, it can be toggled on and off by pressing the *Home* button on the keyboard.

Warning: Enabling this feature activates a second camera to be rendered on screen, which is extremely resource heavy, and can cause lag. It is advisable to only use it when absolutely necessary.

4.5 Autopilot mode of experiments

The environment has an autopilot mode for running multiple sessions with different settings with the same participant in a pre-defined sequence. This

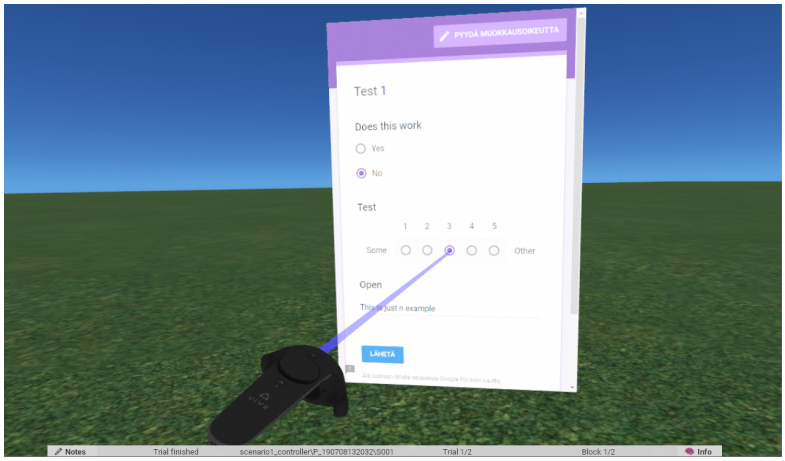


Figure 10: The in-world browser with a form loaded.

allows the experimenters to run the same sequence of sessions to each participant without having to enter the settings for each session manually. The autopilot mode can be enabled by checking the box labelled *Autopilot* in the settings window. The autopilot mode will take the entered participant information, while ignoring the scenario, locomotion, and environment settings selected in the window, using the ones selected in the *Autopilot.csv* file found in the settings folder. The settings in the *autopilot.csv* in order are: locomotion method, scoring method, scenario, environment settings, and session number. All columns in the file should be filled to contain the names of valid, existing settings, as shown in Fig. 11.

autopilot.csv

```
1  locomotion,scoring,scenario,environment,sessionNumber
2  keyboard,Method 1,scenario1,default,1
3  controller,Method 2,scenario2,default,2
4  headbob,Method 3,scenario3,default,3
5  teleport,Method 1,scenario2,default,4
```

Figure 11: An example of the autopilot file.

5 Running Trials

To use the experiment environment, run the *vr-environment.exe* found in the environment folder. In addition to this user manual, there are video guides

instructing about the usage [2] and customization [3] of DeFINE.

5.1 Graphical User Interface (GUI)

The first thing the user will see once the environment has started fully is the settings window, which allows changing the experiment parameters by selecting pre-defined settings files.

The experiment settings Graphical User Interface (GUI) window is shown in Fig. 12. The GUI and its functionality is provided by Unity Experiment Framework [1] with slight modifications.

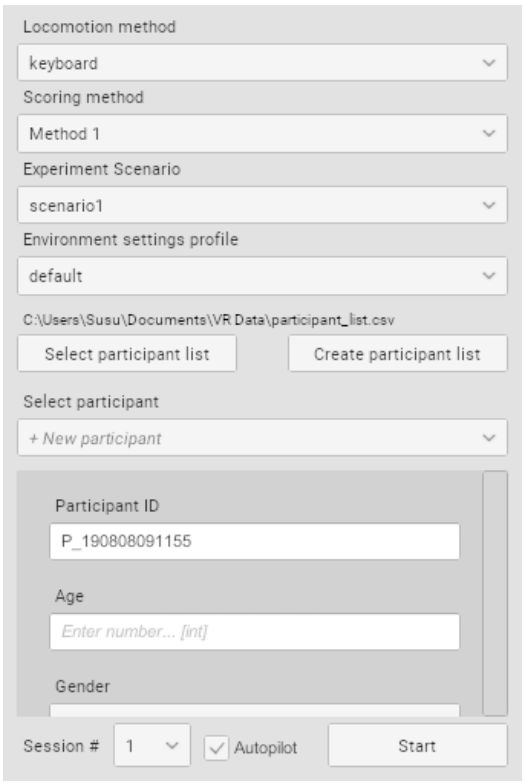


Figure 12: Screen capture of the experiment GUI.

[2] Available at https://youtu.be/OVYiSHygye0
[3] Available at https://youtu.be/smIp5n9kyAM

5.1.1 Locomotion method

The desired locomotion method settings can be selected from the dropdown menu. These settings affect the behaviour and method with which the experiment participant can move in the environment.

5.1.2 Scoring method

The desired scoring method can be selected from the dropdown menu. These settings affect the weights of the scoring function and the feedback shown to the participants.

5.1.3 Experiment Scenario

The desired experiment scenario settings can be selected from the dropdown menu. These setting affect the number of training and testing trials, locations of the start and goal positions, and the visibility of the start oriented blocks and icons.

If one wishes to only run, for instance, the training phase of a scenario, it is advisable to create a new scenario settings file, with the number of test trials set to zero. The same method can be used for only running the test phase. The scenario settings files are located in the "StreamingAssets/Scenario" folder of the software. In the file, modify the "n_practice_trials" and "n_main_trials" parameters in the file. An example of this is shown in Fig. 13.

```
{
    "n_practice_trials" : 5,
    "n_main_trials" : 5,
    "trial_time" : 120,
    "distance_deadzone" : 0.3f,
    "time_coeff" : 0.01,
    "distance_coeff" : 0.2,

    "use_start_blocks_in_training" : true,
    "use_start_blocks_in_testing" : false,
    "use_start_icons_in_training" : false,
    "use_start_icons_in_testing" : false,

    "start_1" : [-4.5,-4.5,45],
    "end_1" : [3,-4],
    "end_2" : [-4, 3]
}
```

Figure 13: An example scenario settings file.

5.1.4 Environment settings profile

The desired environment settings profile can be selected from the dropdown menu. These settings affect the behaviour of the fly circling the goal positions, the room size, and whether or not to hide the room and enabling of fog upon entering testing phase of the experiment to obscure visibility.

5.1.5 Participant list

Participant list is a CSV file containing all information about the participants of the experiment. By default the environment remembers the selected participant list, if one has been previously selected. If no participant list has been previously selected, select or create a **participant list** now. When selecting or creating a new participant list, keep in mind that all data to be collected during the experiments will be saved to the same folder with the participant list.

5.1.6 Select participant

After selecting a participant list, the user can use the dropdown menu to either select a new participant or a previously saved participant. Selecting a new participant will generate a new participant ID based on the system time at the moment in the participant information form below the dropdown menu. Selecting a previously saved participant will automatically fill the participant information form with the previously entered values.

5.1.7 Participant information

In the participant information form, the user should fill the participant ID, participant's age, gender, and highest achieved degree. For age and degree, applicable option from the respective dropdown menus is to be selected.

5.1.8 Session number

Lastly the user will need to specify **session number** for the selected participant. This allows saving multiple sessions with same participant easily.

5.1.9 Starting and ending session

When all information has been filled to the settings window, click the *start* button. This will immediately set all experiment objects and parameters to the ones specified in selected settings files, and attempt to start the first trial.

If there already is stored data for the given participant with the same scenario and locomotion settings and session number, the system will give a warning about this. Clicking *OK* on the warning will proceed, overwriting any previous data for the previous experiment. Clicking *Cancel* will return to the settings window, allowing changes to the settings in order to avoid conflicts with previous experiments.

Pressing the *Escape* button on the keyboard at any point during an experiment will abort the ongoing session and bring up the settings window. Data regarding the already finished trials at that point are stored as usual.

After finishing the last trial of the session, the settings window will become visible again.

5.2 Highscore board

The environment keeps track of the highscores obtained during the main trials. The scoreboard, shown in Fig. 14, is shown to the participant in between trials for an amount of time specified in the settings. Setting the time to zero in the settings results in the scoreboard not being shown. Scores from the practice trials are not counted in the highscores. If the latest trial made a new highscore onto the board, this new score is shown in red. Scores previously made by the same participant show ***YOU*** in place of the participant ID. If the latest trial did not make it to the board, the score is shown below the scoreboard. When the scoreboard is visible, the countdown timer indicates how long it will be visible. Upon the countdown timer reaching zero, the scoreboard is hidden and the next trial begins.

The settings files allow specifying the files where to load the scores and where to save changes separately. This adds the benefit of allowing the experimenter to use a fake, pre-prepared scoreboard in order to give all participants the same motivation and challenge at getting into the scoreboard. If the same participant is intended to do multiple sessions in a row, the scoreboard can be retained in memory, creating the illusion of the scores being saved even if they are not.

(a) A new highscore was made.

(b) Score didn't make it to board.

Figure 14: The highscore board shown in between trials

5.3 Settings files

The environment is provided with pre-defined setting files the experimenter can choose from using the user interface. Modifying and creating new settings files is not required for the running of experiments, provided that the default settings are suitable for the experimenters purposes.

The settings in the environment is divided into three parts: Locomotion settings, Scenario settings, and Environment settings. Each three settings are read from a respective JSON file at the beginning of the experiment (The moment the user clicks on start). Any setting not found from the settings files will use their default values, with the exception of start 1 and goal 1, which are always required to be specified in the settings files.

5.3.1 Scenario Settings

The scenario settings contains the relevant settings for adjusting the experiment scenario. The contents of the scenario settings are:

Name	Description	Data type	Default value
n_practice_trials	number of practice trials	int	5
n_main_trials	number of test trials	int	5
trial_time	max time per trial	float	120
use_start_blocks_in_training	oriented blocks for practise trials	bool	true
use_start_blocks_in_test	oriented blocks for test trials	bool	false
use_start_icons_in_training	start icons for practise trials	bool	true
use_start_icons_in_test	start icons for test trials	bool	false
start_1	the coordinates of start 1	list<float>	N/A
(optional)start_2	similar coordinates for start 2	list<float>	N/A
end_1	the coordinates of goal 1	list<float>	N/A
(optional)end_2	similar coordinates for end 2	list<float>	N/A

The coordinate settings need 4 values, for x,y,z, and rotation, respectively. The x,y, and z are in meters, while the rotation around y axis is given in degrees. Should the user provide 3 values, y (height) defaults to 1. Should the user provide only two values, the y defaults to 1, and rotation defaults to 0.

Care need to be taken when adjusting start and goal positions to prevent them being outside the room in the training phase. In particular, to ensure that the fly circling around the goal position does not clip through the walls, the goal should always be positioned at least a minimum distance of *fly_training_radius* + *fly_offset* from the room walls.

5.3.2 Locomotion Settings

The contents of the locomotion settings are used to adjust the locomotion method behaviour, and depend slightly on the desired locomotion method. The common settings are "locomotion", that specifies as a string the desired locomotion method.

The supported options are:

- "keyboard" for keyboard locomotion. This is the default.
- "controller" for VR controller locomotion
- "teleport" for teleportation locomotion
- "armswing" for walking in place based on the movement of the hand-held controllers
- "headbob" for walking in place based on the vertical movement of the Head Mounted Display

- "walking" for physical walking

Settings for keyboard locomotion:

Name	Description	Data type	Default value
speed	Speed multiplier for the locomotion	float	4
rotation_speed	rotation speed, degrees/s	float	85

Settings for Controller locomotion:

Name	Description	Data type	Default value
speed	Speed multiplier for the locomotion	float	2
use_for_rotation	Controls whether the controller is used for rotating	bool	false
rotate_speed	The max rotation speed, degrees/s	float	85

Armswing locomotion settings:

Name	Description	Data type	Default value
speed	Used for adjusting movement speed	float	21
require_moving_both	Specifies if both controllers need to move for the participant to "walk"	bool	true
min_movement_threshold	threshold for movement. Lower values are uncounted	float	0.01
max_movement_threshold	threshold for movement. Higher values are uncounted	float	0.5
max_swing_amount	The maximum total movement allowed, limits maximum velocity	float	0.5

For armswing locomotion, the movement thresholds are mainly intended for eliminating constant movement due to tracking errors and slight movements of the participant, and to prevent sudden jumps if the system loses tracking of one of the controllers.

Teleportation locomotion only has one setting, the float "speed", with default value of 10. For this locomotion the speed is in fact the speed of the "line" visualization of the teleport target. A higher value causes the line to bend in the gravity less, allowing longer teleportation distances.

Headbob locomotion settings:

Name	Description	Data type	Default value
speed	Used for adjusting movement speed	float	1
deadzone	The deadzone for movement, used for eliminating noise	float	0.001
threshold	The minimum vertical movement needed for walking	float	0.01
bob_hold	The time to keep walking after bob detection	float	0.3
rotation_threshold	vertical rotation threshold to distinguish looking down from bobs	float	0.1

Physical walking locomotion only has one setting, the float "speed", with default value of 1. It defines the mapping between participant's physical movement and movement in the virtual environment, value 1 providing one-to-one mapping between physical and virtual.

5.3.3 Environment Settings

Environment settings are used for adjusting settings separate from the locomotion and scenarios.

Name	Description	Data type	Default value
remove_walls_for_test	Remove the walls when entering test phase	bool	true
use_fog_in_test	Enable fog when entering test phase	bool	true
hide_inactive_starts	Have only the used start visible	bool	true
one_active_goal_in_test	Select only one active goal per trial in test phase	bool	true
texture_style	Use default or custom textures	string	"default"
fly_training_radius	The radius the fly circles the goal from during training	float	0.75
fly_test_radius	The radius the fly circles the goal from during test	float	1.5
fly_speed	speed of the fly	float	1
fly_offset	distance the fly is allowed to go off the specified radius	float	0.5
fly_min_height	The minimum height the fly flies at	float	0.75
fly_max_height	The maximum height the fly flies at	float	1.25
fly_offset_step_size	A step size for the offset	float	0.005
room_size	The size of the room	`list<float>`	[10,4,10]
highscore_retain"	Keep scoreboard in memory between sessions	bool	true
max_forward_speed	The global limit to any locomotion speed, m/s	float	2
enable_mouse_for_forms"	Enables the experimenter to use mouse for forms	bool	false
block_forms	List of urls of forms to display after block	`list<string>`	[ ]

5.3.4 Scoring Settings

Scoring settings are used for adjusting the scoring function and it's weighting of speed and accuracy.

Name	Description	Data type	Default value
keyword	A keyword to mark scoreboard files with	string	"None"
score_time	The time to show the scoreboard between trials	float	3
speed_coeff	coefficient for time for scoring	float	0.01
acc_coeff	coefficient for distance for scoring	float	0.1
scaling_factor	a factor to scale the scores with	float	1
fake_scoreboard	Use a fake scoreboard	bool	false
distance_deadzone	distance to goal to consider zero	float	0.3
round_score	Should the displayed score be rounded to integers	bool	true

It is important to note that using any other word than "None" as the keyword in the settings will result in the system using a custom method with the given name.

6 Data acquisition

All collected data will be stored in the same place as the selected or newly created participant list. The data is collected in hierarchical folder structure, as shown in Fig. 15. The highest level of this hierarchy is the experiment scenario-locomotion method folder, under which all experiments belonging to the respective pair is stored under separate folder for each participant. Session specific folders are found inside the participant specific folders. Inside the session folders are the generated data files of each session.

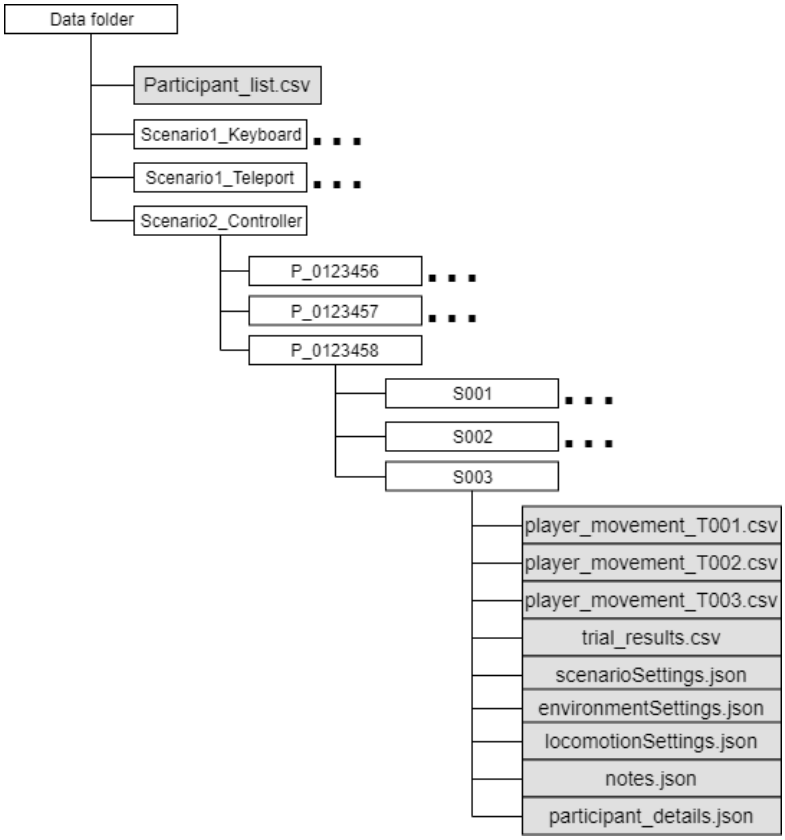


Figure 15: The file hierarchy of the gathered data.

The experiment environment creates the following files:

1. Highscore file, if specified in settings, in the same folder as the participant list. Contains top 10 scores of all test trials.

2. *player_movement_T###.csv* for each trial, containing the participant location data. ### will be the trial number. Contains timestamped data about the participants x and z position, rotation around y axis, and the status of audio and light in the environments.

3. trial_results.csv, that contains the starting time, ending time, total time taken, score, and distance to goal for all the trials in that session.

4. A copy of the used environmentSettings.json

5. A copy of the used locomotionSettings.json

6. A copy of the used scenarioSettings.json

7. A copy of the used scoreSettings.json

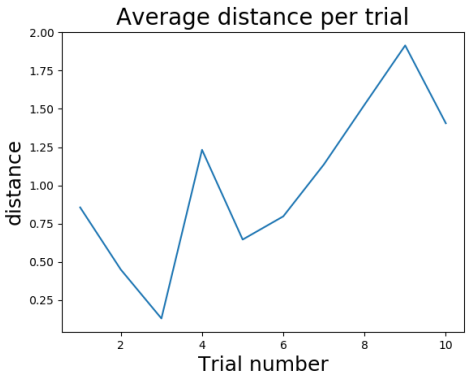
8. participant_details.json, that contains participant ID and any other collected data

9. notes.json, containing all the notes the experimenter gathered during the session

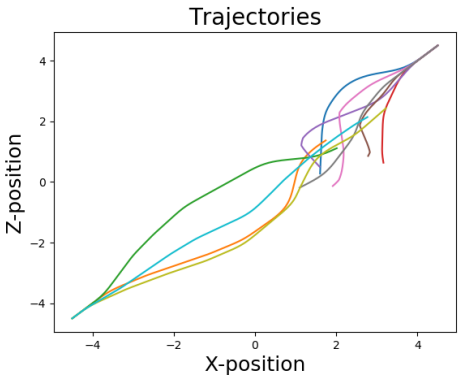
The continuous data is stored at every frame, which is when Unity updates all the elements, to avoid missing any movement. This makes the logging frequency dependent on the frame rate the environment is used. For the development hardware, the logging frequency is about 60Hz.

6.1 Plotter for the data

The environment includes a separate plotter script for creating plots of the data. The script has been written with python3. The plotter has requirements of having python3 installed as well as numpy, matplotlib, and pandas libraries for it. The plotter can automatically plot the average score, distance, or time taken by all the participants for each of the trials, as shown in Fig. 16a. The plotter takes three command line parameters: the path to the folder where to look for the experiment data, label ("score", "distance", or "time") depending which average plot is wanted, and a path to a folder where to save the created figures. If these are not given, the plotter will prompt for the values separately. The plotter also has functions for plotting the trajectories the participant took in each trial, as shown in Fig. 16b.



(a) Average distances per trial.



(b) Trajectories of a session.

Figure 16: An example of plots the plotter script can produce.

Acknowledgements

Thanks to Onur Sari and Ville Sinkkonen for their contributions to the development of this framework.

References

[1] J. Brookes, M. Warburton, M. Alghadier, M. Mon-Williams, and F. Mushtaq, "Studying human behavior with virtual reality: The unity experiment framework," *Behavior research methods*, pp. 1–9.