FHIRWEAVR

API Documentation
Version 1.2.0

What's new in this version

- Updated FHIR references to use local paths
- Data transfer performance improvements
- Updated visuals for the demo game
- Amendment to previous documentation

What is FHIRWEAVR?

FHIRWEAVR is an API for the use of the Fast Healthcare Interoperability Resources (FHIR) specification with virtual reality software. FHIRWEAVR is short for FHIR Workbox for Exercise And Virtual Reality. Currently, the API is written for the development of games using the VirZOOM exercise bicycle in the Unity engine.

System Requirements

Unity 2017.1+
Visual Studio Version 2017+ (.NET Version 3.5)
VirZOOM Beta Bicycle with a compatible head-mounted display.
For writing to an external device, support for HTTP GET/POST and TLS 1.0 is required.
For reading FHIR documents, ability to read XML documents.

What does FHIRWEAVR give me access to?

FHIRWEAVR allows developers to access VirZOOM outputs in C#/Unity, to transform and store these in FHIR-compliant format, as well as send data to external systems intended for use by medical professionals.

Health and Safety

Consult a medical professional before using the VirZOOM. Set the system up on a level surface and clear the surrounding space. Do not exceed the weight limit of the Beta bicycle.

The author of FHIRWEAVR accepts no responsibility for damages caused by the use of the API or the VirZOOM bicycle. For full health and safety information, see www.virzoom.com.

API Structure

The download includes both FHIRWEAVR and an example endless runner game which makes use of it. The included game assists in rehabilitation of motor control, balance in particular, as well as assists in single-sided field-of-view perception therapy. Users of FHIRWEAVR are free to build on top of this example game should they choose to do so. If not, simply delete the DemoGame assets folder and any GameObjects you do not wish to use (for example, the Environment, EffectsCanvas or BikeMesh objects). The FHIRHUD object must remain attached to your **VZPlayer** object if you wish to display data.

The core scripts of FHIRWEAVR, listed below, can be found in the FHIRWEAVR assets folder.

DataHandler.cs. Scripts for reading metrics from the VirZOOM bicycle controller, generating session-average measurements, displaying bicycle outputs in-game and reading the current device's save path.
DateTimeFormats.cs. Scripts for getting FHIR-compatible date and time formats independent of system configuration.
ExampleUses.cs. An example setup for and use of FHIRWEAVR in Unity.
GenFHIR.cs. Scripts for generating and saving FHIR Documents for device profiles and metrics in XML format.
PushData.cs. Scripts for pushing data to external systems.

Data Catalogue

Available Metrics

Distance - total distance traveled from start of session.
Speed - speed of movement in meters per second.
Resistance - bicycle resistance setting.
Heartrate - user heart rate in beats per minute.
Rotation - yaw of head in radians (positive left/counterclockwise).
Lean - degree of lean left/right of user in meters (positive left).
Incline - degree of lean forward/back of user in meters (positive forward).

Available Types of Metric

Current - measurement at current frame.
Session - measurement average (or total in the case of distance) over a session, from start-up to when the function is called.

Output formats

Live individual metrics: **double**
Live all metrics: array of **double**
Stored metrics: **FHIR Documents** format, **XML**
Device profile: **FHIR Documents** format, **XML**

Available Functions

Before using any functions, you must instantiate FHIRWEAVR. In your Monobehaviour class, put the following at the start (name the instances whatever you'd like):
DataHandler myDataInstance;
PushData myUploaderInstance = new PushData();
In your **Start()** function, put
myDataInstance = DataHandler.Instance;

This is already set up for you in **ExampleUses.cs**.

Starting and Ending Sessions

Prefix the following functions with the name of your data instance and a period, e.g. "myDataInstance."

StartSession(). Starts the FHIRWEAVR session. If you want to use FHIRWEAVR from the very first frame, put this in your Start() function.

EndSession(). Ends the FHIRWEAVR session.

Getting Data

Prefix the following functions with the name of your data instance and a period, e.g. "myDataInstance."

GetAllData(string type). Returns a list of doubles of all metrics. Type may be current or session.

GetMetric(string type, string metric). Returns a double for the chosen metric. Type may be current or session. Metric may be distance, speed, resistance, heartrate, rotation, lean or incline.

Displaying Data

If you intend to use the built-in FHIR UI, you must keep the FHIRHUD object attached to your VZPlayer.

Prefix the following functions with the name of your data instance and a period, e.g. "myDataInstance."

DisplayAllData(string type, OPTIONAL double duration, OPTIONAL string additionalText). Displays all metrics on a diegetic UI element. Type may be current or session. Duration is the time to display the information in seconds, if no argument is entered, data will be displayed indefinitely. AdditionalText is for adding additional information after the displayed data. If you wish to specify only one optional parameter, do so using named parameters, for example **DisplayAllData("session", duration:5)**. For displaying continually updating data, place this function in **Update()** with no duration parameter specified.

DisplayMetric(string type, string metric, OPTIONAL double duration, OPTIONAL string additionalText). Displays a single metric on a diegetic UI element. Type may be current or session. Metric may be distance, speed, resistance, heartrate, rotation, lean or incline. Duration is the time to display the information in seconds, if no argument is entered, data will be displayed indefinitely. AdditionalText is for adding additional information after the displayed data. If you wish to specify only one optional parameter, do so using named parameters, for example **DisplayMetric("current", "heartrate", additionalText:"Target heartrate: 90 bpm")**. For displaying continually updating data, place this function in **Update()** with no duration parameter specified.

Hide(). Hides all currently displayed UI elements.

Saving Data

Prefix the following functions with "*GenFHIR.*"

Device(). Generates the VirZOOM device profile as a FHIR Document in XML format.

Document(string type, OPTIONAL string overwriteName). Generates FHIR Document for all metrics at a given time in XML format. Type may be current or session. If you wish to overwrite the same file, i.e. have a single file containing metrics which you want updated continually or at different points in time, specify the filename as the parameter overwriteName. If this is not specified, outputs will be saved individually as time-stamped files.

Sending Data

Prefix the following functions with "*PushData.*"

Upload(string fileToUpload, string host). Pushes saved data to external system. The device profile will be automatically pushed only once (requires TLS 1.0 support). FileToUpload is the name of the file to be pushed, if 'last' is entered this pushes the last saved file. Host is the address of the host.

ManualUpload(string fileToUpload, string filePath, string host). Function for manual upload operations. This functions as above but allows full file path specification and does not automatically push the device profile. Useful when using non-default save locations.

Additional Tools

DataHandler.path. Detects the current device configuration and returns a string of the path for save data on the local device. Intended for use by other parts of the API but potentially useful. The program must be run for this to be accessible as the path is generated on Unity Start().

DateTimeFormats.GetDT(string type). Gets FHIR-compliant date and time formats. Intended for use by other parts of the API but potentially useful, especially for consistency in medical software. Type may be date, time, full (both date and time) or filename (human-readable format for use in file names).

PushData.FileExistsAtURL(string url). Checks if a given file exists at the url. Intended for use by other parts of the API but potentially useful. Enter the full path of the file to look for including the file name as the parameter.

myDataInstance.SetDeviceActiveState(bool active). Sets the VirZOOM device's *status* property in FHIR logical tables. Use this if you intend to stop using the VirZOOM for an extended period of time.

Contact

For enquiries regarding FHIRWEAVR, please contact Theo Turner at zcabttu@ucl.ac.uk. For enquiries regarding the VirZOOM device, please contact VirZOOM support at info@virzoom.com.