

# UniVRPNity Documentation

Quentin Petit and Clément Grellier

## 1 Features

- Manage Analog, Button and Tracker remote.
- A syntax very close to C++ native client and Vrpnet client.
- Provide Unity MonoBehaviour scripts and editors for Analog, Button and Tracker remote.
- Provide concrete examples with these scripts (Translation, Rotation, Scale, Animation)

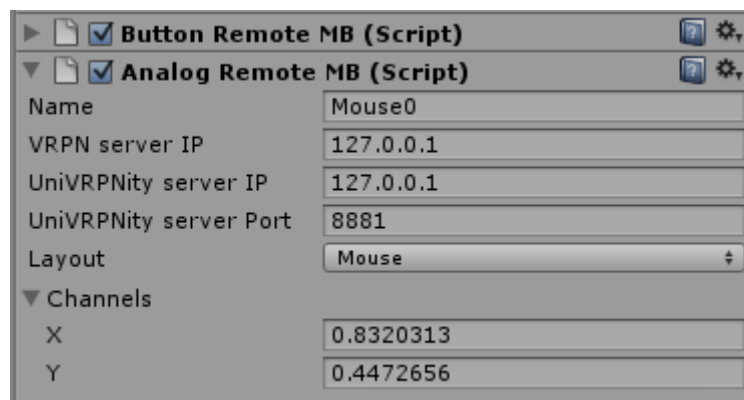


Figure 1: MonoBehaviour remote of the mouse during execution.

## 2 Why using UniVRPNity?

### 2.1 Vrpnet

VRPN is coding in C++ but free version of Unity only accept C# , a modified JavaScript and Boo language. After a few research we hopefully found Vrpnet a VRPN wrapper in C#. But when we try to incorporate it in Unity it throw a native DLL error. Indeed! Free Unity version do not accept C++ wrapper.

## 2.2 UIVA

Then we found UIVA which create an intermediary between the VRPN wrapper and a full C# client which is accepted by Unity. However UIVA does not fit our needs because it is peripheral dependent. For instance to recover DTrack information, we had to modify the UIVA server and created a particular case for the DTrack data.

## 2.3 VRPNWrapper

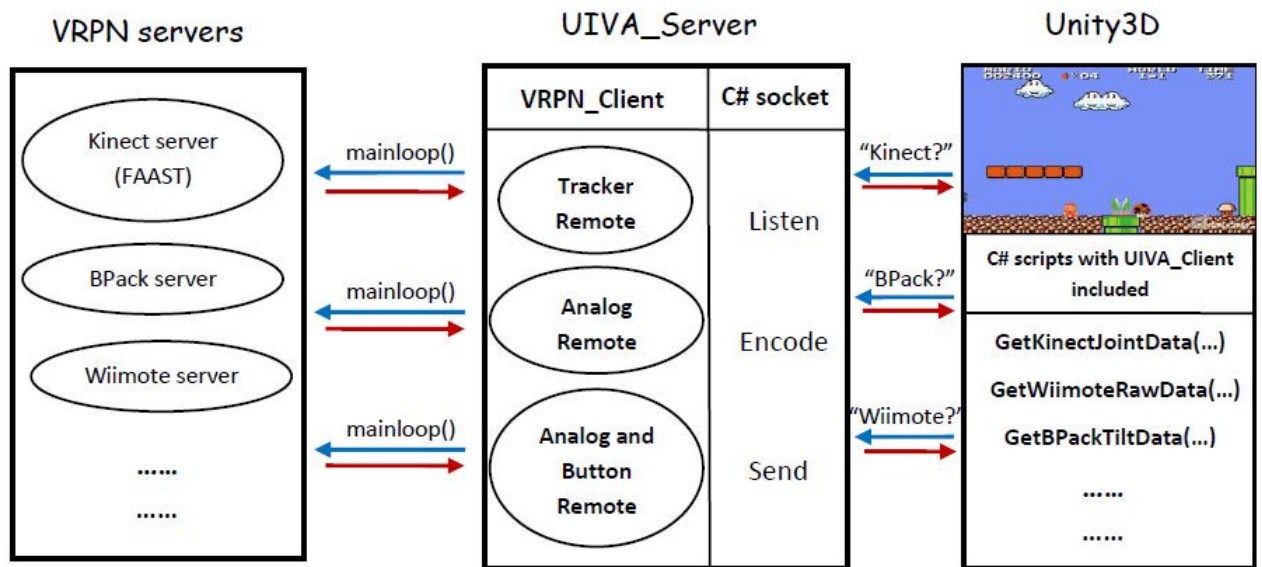


Figure 2: UIVA architecture.

## 2.4 Full C# client

We also tried to write a full C# from scratch but communication with a VRPN server is very hard :

- There is no network specification of VRPN
- VRPN event are binary serialize with C++ type and depend of the VRPN event type (button, analog, tracker)

But we found a way and it worked for button (keyboard), analog (mouse) et tracker (DTrack) but this method wasn't safe and reliable.

- VRPN server send us all information from all peripheral with no easy way to distinguished them
- Parse an unknown frame except for it size is awkward

## 2.5 UniVRPNity

The only reliable solution was UIVA. So we start from UIVA and rewrite a full middle server and a full C# client. It became a true intermediary, because VRPN event are sent directly to client (with a C# conversion).

## 3 UniVRPNity architecture

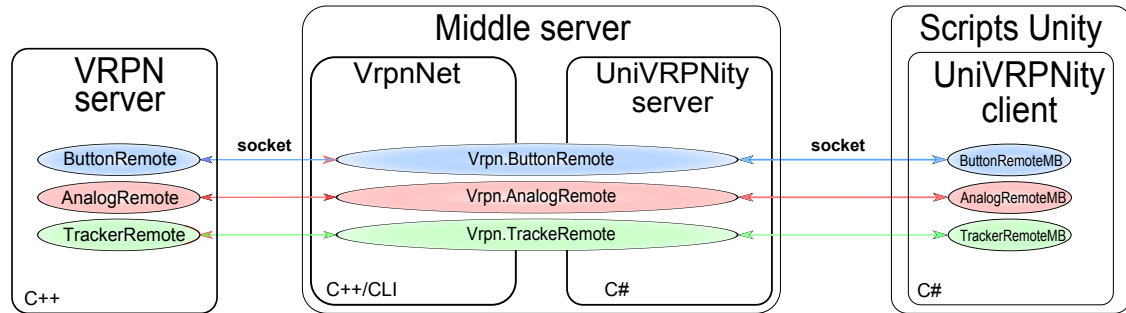


Figure 3: UniVRPNity architecture.

## 4 How UniVRPNity work?

### 4.1 Requirement

- A running VRPN server which listening peripherals
- A running UniVRPNity server

### 4.2 Initialization stage

- A remote client is created in a `RemoteMB Start()` method
- Client ask to the UniVRPNity server to pass through VRPN event of a named and typed device ("Mouse0@localhost#Analog" are sent as handshake.)
- UniVRPNity server ask the same thing to the VRPN server

### 4.3 Iterative stage : For each detected event

- VRPN server sent to UniVRPNity server all data of the event
- UniVRPNity server convert these data in full C# event and transfer them to the concerned client
- Client receive these data from socket and directly store it into a buffer (asynchronous)

- On next Update(), buffer are flushed into client callback (synchronous)

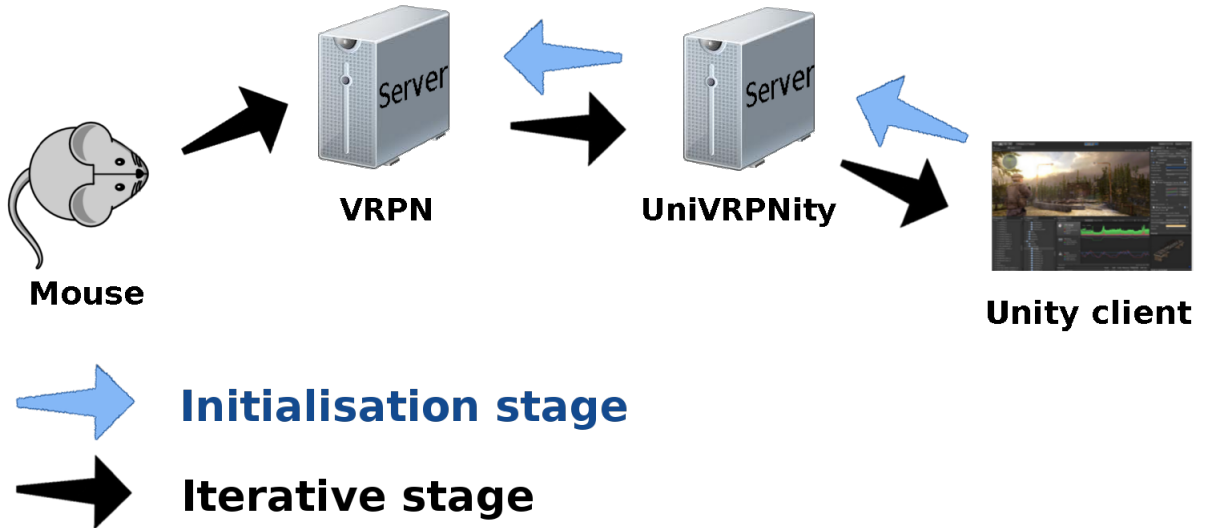


Figure 4: UniVRPNity architecture.

## 5 How to use UniVRPNity?

### 5.1 Compile projects

- UniVRPNityCommon.dll
- UniVRPNityServer.dll (Import directly given VrpNet.dll)
- UniVRPNityClient.dll

### 5.2 Copy

- Copy UniVRPNityCommon.dll in Assets
- Copy UniVRPNityClient.dll in Assets
- Copy scripts into Assets and Editor file into Assets/Editor

### 5.3 Run & Check

- Allow VRPN on your firewall
- Edit VRPN configuration file (vrpn.cfg)

- Launch VRPN server
- Check with print device executable
- Launch UniVRPNityServer.exe with it port in parameter (default:8881)
- Drag and drop MonoBehaviour remotes and configure them on Unity
- Run Unity

## 6 UniVRPNity Server

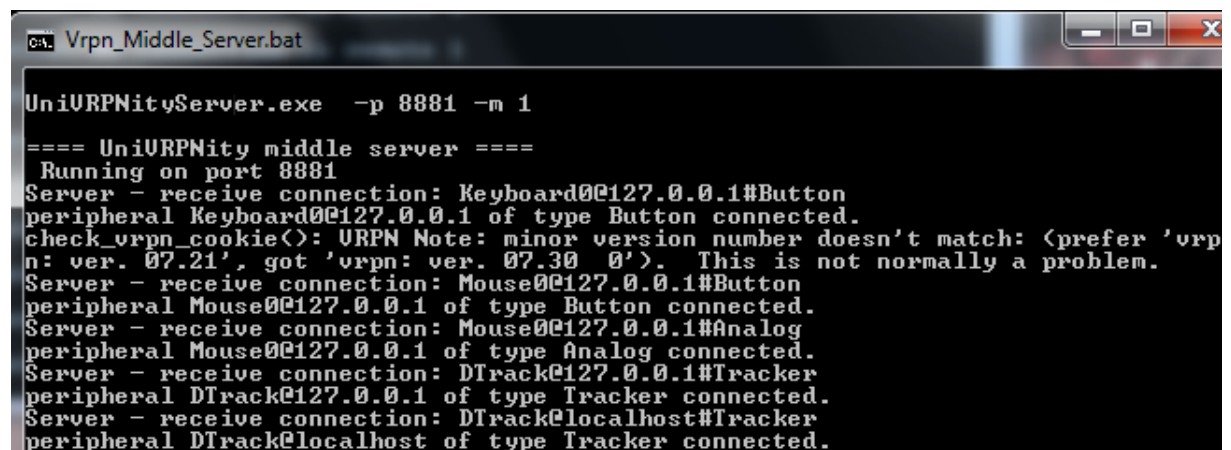
This server can be define as a middleware between VRPN server and an Unity client.

### 6.1 Execution conditions

- Firewall must accept this application
- This server must be running before launching Unity
- It can be run before VRPN server start
- Server port must be unused (Cannot be 2 server with the same port)

### 6.2 Start

UniVRPNity server waiting an handshake from clients. This handshake is a string which contains the name and the address of the device (Analog, Button, Tracker) separate by a '#' character. The server is limited to 32 remotes to listen at the same moment. It create a Vrpnet remote object with the given information.



```

Vrpn_Middle_Server.bat
UniVRPNityServer.exe -p 8881 -m 1
==== UniVRPNity middle server ====
Running on port 8881
Server - receive connection: Keyboard0@127.0.0.1#Button
peripheral Keyboard0@127.0.0.1 of type Button connected.
check_vrpn_cookie(): VRPN Note: minor version number doesn't match: (prefer 'vrpn: ver. 07.21', got 'vrpn: ver. 07.30 0'). This is not normally a problem.
Server - receive connection: Mouse0@127.0.0.1#Button
peripheral Mouse0@127.0.0.1 of type Button connected.
Server - receive connection: Mouse0@127.0.0.1#Analog
peripheral Mouse0@127.0.0.1 of type Analog connected.
Server - receive connection: DTrack@127.0.0.1#Tracker
peripheral DTrack@127.0.0.1 of type Tracker connected.
Server - receive connection: DTrack@localhost#Tracker
peripheral DTrack@localhost of type Tracker connected.

```

Figure 5: UniVRPNity server at launch.

### 6.3 Communication

Events are immediately binary serialize and transfer to client. However to avoid CPU eating, there is a sleep between 2 mainloop method call.

### 6.4 End

It automatically close and kill a remote device when the associate socket is closed.

### 6.5 Arguments of the server

- -h (-help) : Display help. Do not run the server
- -v (-verbose) : Display all peripheral event receive from VRPN server and send to the client
- -p (-port) : Specify the port number of the server. Default port is 8881
- -m (-millisleep) : Time between each internal mainloop in millisecond. Default time is 1ms

Example : `UniVRPNityServer.exe -v -p 8881 -m 1`

## 7 UniVRPNity Client

### 7.1 MonoBehaviour Scripts

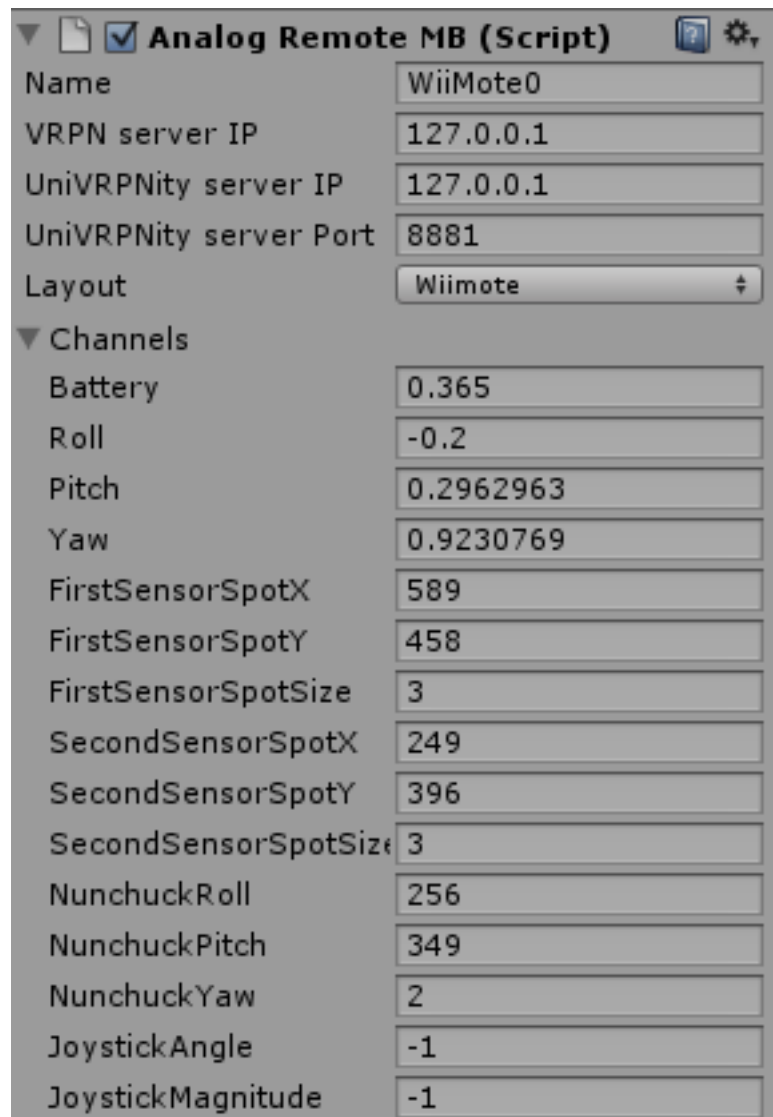
These scripts are ready to use. You have to fill in network parameters and run Unity to recover peripheral data.

- Name : Device name which are specified in `vrpn.cfg`
- VRPN server IP : Host IP where are running a VRPN server
- UniVRPNity server IP : Host IP where are running a UniVRPNity server
- UniVRPNity server port : The port listened by UniVRPNity server

There are 3 specialized MonoBehaviour scripts :

- AnalogRemoteMB for analog device composed of channels
- ButtonRemoteMB for button device composed of states
- TrackerRemoteMB for tracker composed of position (Vector3) and orientation (Quaternion)

### 7.1.1 AnalogRemoteMB example



The screenshot shows a configuration window titled "Analog Remote MB (Script)". It contains several input fields for configuring a remote control device. The fields are organized into sections: general settings, channels, and sensor data. The "Channels" section is expanded, showing a list of sensors and their current values.

Field	Value
Name	WiiMote0
VRPN server IP	127.0.0.1
UniVRPNity server IP	127.0.0.1
UniVRPNity server Port	8881
Layout	Wiimote
<b>Channels</b>	
Battery	0.365
Roll	-0.2
Pitch	0.2962963
Yaw	0.9230769
FirstSensorSpotX	589
FirstSensorSpotY	458
FirstSensorSpotSize	3
SecondSensorSpotX	249
SecondSensorSpotY	396
SecondSensorSpotSize	3
NunchuckRoll	256
NunchuckPitch	349
NunchuckYaw	2
JoystickAngle	-1
JoystickMagnitude	-1

Figure 6: Monobehaviour remote of the Wiimote during execution.

### 7.1.2 ButtonRemoteMB example

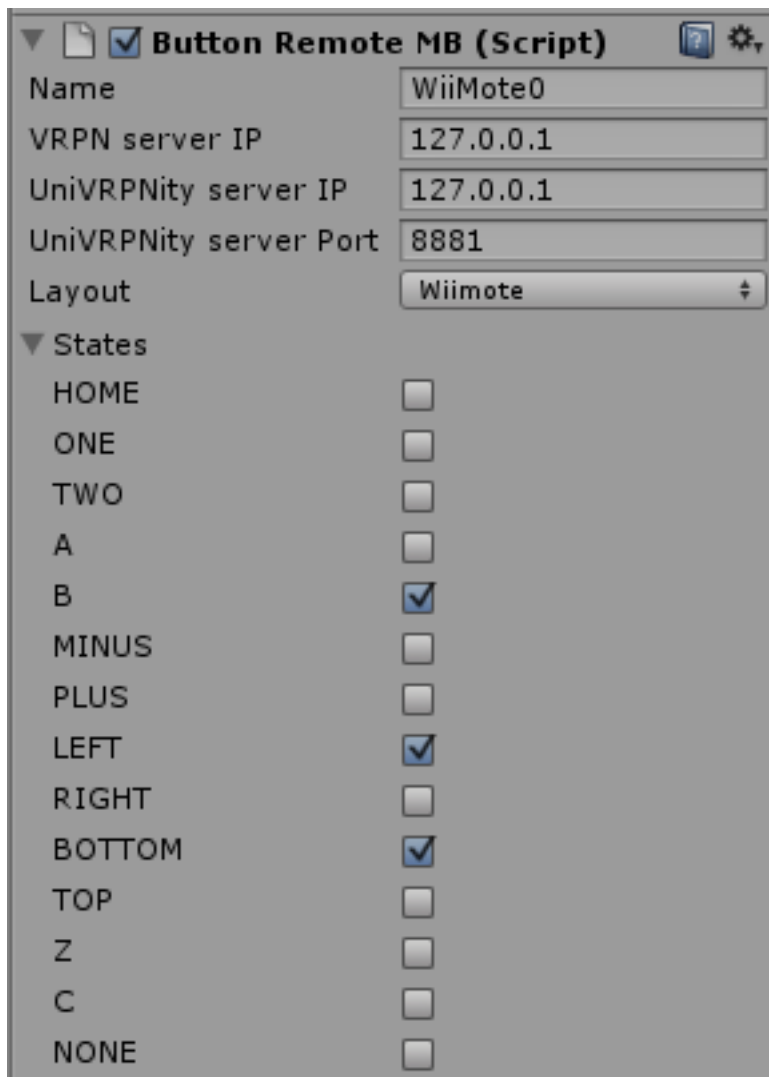





Figure 7: Monobehaviour remote of the Wiimote during execution.



### 7.1.3 TrackerRemoteMB example

▼  **Tracker Remote MB (Script)**  

Name

VRPN server IP

UniVRPNity server IP

UniVRPNity server Port

▼ Trackers

▼ Tracker 0

Position

X  Y  Z

Orientation

X  Y  Z  W

▶ Tracker 1

▼ Tracker 2

Position

X  Y  Z

Orientation

X  Y  Z  W

▼ Tracker 3

Position

X  Y  Z

Orientation

X  Y  Z  W

▶ Tracker 4

▶ Tracker 5

▶ Tracker 6

▶ Tracker 7

▼ Tracker 8

Position

X  Y  Z

Orientation

X  Y  Z  W

▼ Tracker 9

Position

X  Y  Z

Orientation

X  Y  Z  W

▶ Tracker 10

▼ Tracker 11

Position

## 8 UniVRPNity Common

This library contains all server and client common elements :

- Events which are sent over network
- Serializable Vector3 and Quaternion (conversion method are included)
- Some network default values

## 9 Known issues

Unity freezes at the beginning and throw lot of exceptions?

Client have a long timeout for the server connection. It's happen when you forgot to run the UniVRPNity Server. If it's running, check address and port of the UniVRPNity server.

Input values are not recovered?

Check the VRPN server state. Is it running? Is the device enabled in VRPN configuration file. Use print device executable to test it. If it's correct, check the VRPN device name and it IP.

Script stop recording information?

If it happen, you can disable and enable the script to reconnect peripheral.

UniVRPNity Server does not want to run

Check if the used port is correct and available.

## 10 Known bug

- Date associate to event are wrong (wrong hour of the day).