

VrpnUT

Version 2.5.4, © PublicVR
(unedited first draft)

Introduction

VrpnUT is a modification to the game [Unreal Tournament 2004](#) (UT2004) which allows a remote (controller) process to move the viewpoint of a single player via a UDP connection. VrpnUT is packaged as a “mutator”, a standardized package used in UT2004 for modifications to the game. This mutator is freely available to the public (see below).

UT2004 is a multiplayer networked game where users connected via a standard LAN or the Internet can share a single virtual world. One way to do that is for one player to “host” everyone else by running a “Listen Server” instance of the game on their computer. They will have their own player on their own machine, and their machine will referee communication with all the others. This gives the hosting player certain administrator privileges over the game, but once play starts s/he is generally coequal with the other players.

When active, VrpnUT modifies the Listen Server mode by opening (for read only) a UDP port on the local machine. This allows an external process, either on the machine or elsewhere on the network, to write packet to the UDP port. The modified listen server picks up those packets and uses the instructions there to move the player’s avatar on that machine. For example, when VrpnUT reads a packet which tells it to turn the local player’s avatar 30° to the left, the local user will see their entire view turn 30°. Other players in the game see that player’s avatar turned 30° to the left. While VrpnUT is active, the game’s normal navigation controls still work. *(Side effect: If you use standard navigation while the current build of VrpnUT is active, you will find the player’s Ground Speed to be rather quick.)*

Compatibility

VrpnUT is only guaranteed to work with the 3369 version of the UE 2.5 engine which ships with most copies of UT2004. The VrpnUT code is small and simple, so a direct clash with other game modifications are unlikely. (We use "Unreal Tournament 2004, Editor's Choice" edition.) If you have a really old edition of UT2004, you can run the [3369 code patch](#) to upgrade it. We have not tested VrpnUT with UE 2.0, but there is a good chance it will work. Please let us know if you are interested in trying it and we will help where we can.



Figure 1: VrpnUT is used in this cave to drive the user's viewpoint through the scene.

VrpnUT was primarily developed for use with [CaveUT](#), another mutator which allows UT2004 to produce a single contiguous view across multiple screens. The screens can be in any orientation to each other, usually arranged as an enclosure around the user, similar to the [CAVE](#). VrpnUT also works well other software that projects the image into a dome display such as [DomeUT](#).

VrpnUT is intended to be the very beginning of a “client” (information receiver) for the open-source [Reality Peripherals Network](#) (VRPN). VRPN handles a range of VR interface devices, such as HMDs, tracking gloves, treadmills, and much more. It allows the client to communicate with all these devices via a generic format. VrpnUT's UDP packets are in a VRPN compatible format.

Installation

First, download VrpnUT.2.5.4.zip, and unpack the VrpnUT.2.5.4 folder onto your desktop or into some convenient directory. In the following instructions, all pathnames to files begin at the VrpnUT.2.5.4 folder as their root. So, Documentation/VrpnUT.pdf refers to a PDF file in the documentation subfolder under the VrpnUT.2.5.4 general folder.

If you are using VrpnUT **without** CaveUT, copy the contents of the install packet's “System” folder into the “System” folder for your UT2004 installation, which is C:\UT2004 by default, but you can name it anything you want. Allow windows to copy the files into the “C:\UT2004\System” subfolder. If you are using VrpnUT **with** CaveUT, you it's a little easier to use our combined CaveUT & VrpnUT installer.

Usage

To activate VrpnUT, start a listen server with the CaveUT and VrpnUT mutators active. Connect the spectator clients to the server in the normal way. To move the client using VrpnUT, send one or more UDP packets to the port (on the server machine) VrpnUT is listening to.

The UDP Packet is a space delimited string of 5 floating-point numbers in this format:

Linear_Velocity(m/s) Rotational_Velocity(deg/s) Head_delta-X(cm) Head_delta-Y(cm) Head_delta-Z(cm)

A sample packet with a linear velocity of 1.0 and everything else zero might look like this:

```
31 2E 30 30 30 30 30 30 20 30 2E 30 30 30 30 30 1.000000 0.000000
30 20 30 2E 30 30 20 30 2E 30 30 20 30 2E 30 30 0 0.00 0.00
```

Each row represents a 32-bit word. The first column is the packet coded in hexadecimal, the second column shows the numeric values in decimal.

Linear Movement and Rotational Movement continue at the specified velocity until the next packet is read. Linear Velocity is positive when moving forward and negative when moving in reverse. A value of zero stops the movement. Rotational Velocity is positive when rotating clockwise (right) and negative when rotating counter-clockwise (left). Again, a zero value stops rotation. The Head delta* parameters are currently unused. They are included for VRPN compatibility.

Do not have any trailing blanks or other characters at the end of your UDP packet, or VrpnUT will not interpret it correctly.

All parameters for VrpnUT are contained in the configuration file UT2004/System/VrpnUT.ini. They are:

```
[VrpnUT.VrpnUT_Link]
DegreesToUU=182.044444
CentimetersToUU=50.030621
```

The first one, “DegreesToUU” shows the (user-defined) number of unreal units per degree. So VrpnUT will take a rotation request in degrees, multiply it by this number, and give the result in Unreal Units to UT2004. You would change this ratio if you were not actually using degrees, but some other measure of angular distance, such as Radians.

The second parameter converts from centimeters to unreal units, a scaling factor which affects translation specified by VrpnUT. You should stay with the default if you can, but the actual conversion you want depends on the scale at which the virtual environment was built by its artist, the scale at which it is projected, and how you want the user to experience the virtual environment. This is part of a complex and interesting perceptual scaling challenge with immersive displays, which we will not go into here, though we would be happy to discuss it with you.

Please direct any questions you have to jeff@planetjeff.net

VrpnTester

The VrpnUT installation package contains a folder, “VrpnTester”, which contains `vrpn_tester.exe`, a simple Python program which will generate UDP packets VrpnUT can read. Copy the folder anywhere on your LAN or the PC where you will run your UT2004 Listen Server. To configure the tester, edit “VrpnTester/UdpConfig.ini”. Set “host = ..” to the IP address of the machine with the Listen server. If it is the same machine, leave “host = localhost” which is the default value. Similarly,

set “port = ...” to the same port VrpnUT is listening on. The default port for both VrpnUT and VrpnTester is “5000”.

To run the tester, double-click “vrpn_tester.exe” and a dialogue window will pop up. Along the top, you will see the UDP address and port the tester will write to. In the first textbox, enter the forward velocity you want specified in the next UDP packet. When you press the button UPDATE, the tester will send a UDP packet to VrpnUT, which will cause the player on the listen server to move forward (translate) at the specified speed, continuously, until a different speed is entered. A negative value moves the player backward.

Similarly, you can make the player turn by entering the rotational velocity desired. A positive value rotates the player to the right, while a negative value rotates him to the left. Of course, you can enter nonzero rotation and translation values in the same UDP packet.

Bugs and Fixes:

Credits:

Gerke “Max” Preusserner at Virtual Heroes (Programming, Primary Author)

[Jeffrey Jacobson](#) at [PublicVR](#) (Specifications, Testing, Documentation)

[James Cook](#) at the [Medical Virtual Reality Center](#) (VrpnTester, General Testing)

[Patrick Sparto](#) at the [Medical Virtual Reality Center](#) (Specifications)

[Medical Virtual Reality Center](#) (Funding)

[Jonathan Hagewood](#) at [Psyonix Inc.](#) (Programmed the previous version)

Extras:

<Still have to write this section.>

Distribution:

Copyright (c) 2005 PublicVR. All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
3. The end-user documentation included with the redistribution, if any, must include the following acknowledgment:

This product employs the software, VrpnUT(tm)© PublicVR, free to the public at "www.publicvr.org".

Alternately, this acknowledgment may appear in the software itself, if and wherever such third-party acknowledgments normally appear.

4. VrpnUT is a trademark of PublicVR. Accordingly, products derived from this software may not be called "VrpnUT", nor may "VrpnUT" appear in their name, without prior written permission from PublicVR.
5. All versions of VrpnUT v198 are the property of PublicVR. However, VrpnUT is a derivative work based on a Unreal Tournament 2004 by Epic Games. As such, it is subject to the End User License Agreement distributed with the installation package for Unreal Tournament 2004.

This software is provided "as is" and any expressed or implied warranties, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose are disclaimed. In no event shall PublicVR or their contributors be liable for any direct, indirect, incidental, special, exemplary, or consequential damages (including, but not limited to, procurement of substitute goods or services; loss of use, data, or profits; or business interruption) however caused and on any theory of liability, whether in contract, strict liability, or tort (including negligence or otherwise) arising in any way out of the use of this software, even if advised of the possibility of such damage.