VIRTUAL i·O

i-glasses!™ Developer Kit

Version 1.2

Revised 6/95

Table Of Contents

i-glasses!™ Tracker Software Development Toolkit
Version 1.2

Section A

Virtual i-O i-glasses! Tracker Software Development Toolkit

Copyright 1995 Virtual i-O
All Rights Reserved
Version 1.0

Installation

Make a directory on your hard drive called vio. Copy the directory viodkcd to vio. Hereafter, C: refers to the drive where you created vio.

Everything you need to know about programming the tracker is included in these libraries. To recompile and work with these files, set the environment variable SOFT to point to where these files reside. For example, if these files are located in C:\vio, then type: *set SOFT=C:\vio*.

If you include %SOFT%\batch in your path, you can have access to handy batch files to quickly move from one library directory to another. For example, if you type *imath*, you'll be placed in %SOFT%\src\imath1. This was done to make DOS more UNIX-like (aliases).

You can also cd to C:\vio and type *setenv*, which will setup the environment variable SOFT and add %SOFT%\batch to the path.

Usage

To make a library, simply type *m*, which calls *wmake -f watcom.mak*.
To update a library (make it and copy it to %SOFT%\libs) type *m update*.
To relink a higher level library tester or application, type *l*.
To clean a directory, type *m clean*.
To clean and update a library, type *m remake*.

Makefiles

This development system uses a generic makefile, which is located in %SOFT%\make. All frequently used information is placed in this file so individual library makefiles are trivially simple (they include %SOFT%\make\genwat.mak). If you use stack based arguments and/or don't have DOS/4GW Pro, you'll have to make changes to genwat.mak.

Stack based compilation for Watcom

To use stack based arguments with Watcom, uncomment the USE_STACK_ARGS line in SOFT%\make\genwat.mak (remove the #), then remake the entire library. Type *src*, then type *makeall* (calls batch files in %SOFT%\batch).

Building the libraries and testers with DOS/4GW Professional

If you have DOS/4GW Professional (from Tenberry Systems), you must either set the environment variable *use4gwpro* (*set use4gwpro=1*) or modify %SOFT%\make\genwat.mak by uncommenting the line USE_4GWPRO (remove the #).

Stereoscopic 3D rendering

Examine the code in src\grid1. Check out test.c: this code shows how to set up the tracker and draw stereo 3D in interleaved mode. The graphics code is located in src\vgfx1 for direct video memory rendering, and src\dramgfx1 for off screen rendering and interleaved copying. Press 'b' to toggle the background color. Press 's' to toggle stereo 3D and 'd' to toggle DRAM rendering (unless you're using a Pentium, DRAM rendering should be much slower than direct display memory rendering. That's *this* example only. Full texture mapped screens with transparency effects will probably run faster when rendering in DRAM).

SciTech's direct SVGA linear addressed graphics access should make this quite a bit easier (320x400, 256 to 64k color modes, addressed linearly, and not through slow VGA hardware (no outs, of course, either). Contact SciTech for more info.

Integer math

All math is now in 14 bit fixed point (1<<14 = 1.0). The yaw computation now does not require square root and has worst case error of 12 thousandths of a degree with 4 thousandths of a degree average case (when compared to floating point). If you do your own integer math, it is extremely important to have an accurate yaw computation. If you can do better, please let us know. The lookup tables used here are 256 entries for sine and 256 entries for arctanget. This uses 2048 bytes. The lookup tables are 28 bit fixed point and the entries are interpolated when looked up.

See src\convert1\convert.c for the math required to perform the angle computations (it's fast, short, and simple).

Filtering

The filtering method has changed to adaptive filtering (and will change again). We are currently experimenting with filtering in this library (you'll see it in src\convert1\convert.c. The filter code is in src\filter1\filter.c (currently working as a simple running average, although the code is a general Finite Impulse Response (FIR) filter library. An Infinite Impulse Response (IIR) interpolating filter is also included). *If your application does additional filtering (as in src\convert1\convert.c), you must have an option to turn it on and off.*

For now, don't set the tracker filter settings in your application. Use tcal to set the filters. We'll provide a user preferences tool with the tracker to set filter values (tracker.exe in the apps\trackman directory).

If you use tcal, *DO NOT ATTEMPT TO CALIBRATE THE TRACKER*. Factory calibration requires hardware jigs and cannot be done reliably by hand. We are providing this code so you can see how everything works (it's an open system). Once factory calibrated, the tracker should not need to be calibrated again.

16-bit DOS Compilation (Borland C/C++ compiler. Tested with version 3.1)

For the purpose of creating real-mode drivers, the following modules have been modified to allow compilation under Borland C/C++ 3.1 (or later) for 16-bit 386 DOS real-mode:

```
imath.c
vector.c
timer.c
filter.c
convert.c
serial.c
tracker.c
```

These are all the files that you need to write a real-mode driver. The vector1 and filter1 libraries are optional (vector1.h is needed by other libraries, though).

Examine %SOFT%\apps\simple for a single header and single library example that uses the bare essentials.

For a Borland 16-bit real-mode version, see %SOFT%\src\apps\bsimple. The library vstrack1.lib and header vstrack1.h can be used to create a real-mode driver.

To make a Borland `.obj` file, type *b*, which calls *make -f borland.mak*.
To remake all Borland files, type *bmakeall* (in %SOFT%\src).

Real-mode driver examples

Example real-mode drivers for Raven's Heretic and LucasArts' Dark Forces can be found in apps\htdrv and apps\dark respectively. A better way to handle drivers would be to use 32-bit protected mode DLLs. This is more elegant and will have higher performance as real/protected mode swapping won't be necessary. Contact Tenberry Systems for more information.

Notes

The tracker specification is finalized (for a 1.0 version). Thus, if you follow the spec, you don't need to use any of this code directly. If you find a difference in the spec and the code, please let us know.

Do not use any of the code that reads and writes to tracker memory. This will change and is not otherwise documented. To set filter values and mode settings, use tracker mode commands (not tracker read/write commands).

If you get an error communicating with the tracker, send a "!\r" string to the tracker until you get an 'O'.   If you keepsending just 'S', the tracker may never respond until reset with a "!\r" or other ! command. The examples call resetTracker(), which performs this operation.

i-glasses!™ Tracker Interface
Version 1.2

Copyright 1995, Virtual i-O
All Rights Reserved

Section B

Introduction

The **i-glasses!** ™ Tracker, available exclusively from Virtual i-O Corporation, sends yaw, pitch, and roll information to a host computer. This information is then available to application software for creating immersive, head-tracking, real-time stereoscopic 3D simulations called *virtual reality*.

Basic Tracker orientation

All orientation descriptions are from the perspective of someone actually wearing the Tracker:

- Positive yaw is defined as a left head rotation.
- Positive pitch is an upward head tilt.
- Positive roll is a left head tilt.

Y axis

Roll

Pitch

X axis

Yaw

Z axis
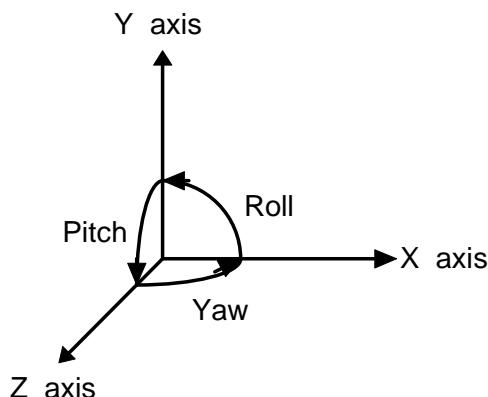
The coordinate system used is +Y up, +Z out, and +X right (The positive axes can be formed with the right-hand index, middle finger, and thumb: a right-handed coordinate system. See *Computer Graphics, principles and practice*, by Foley et al for more information on coordinate systems).

Communicating with the host computer

The Tracker communicates with the host computer via an RS-232C 3-wire serial interface (TXD, RXD, GND). The Tracker can run at 1200, 2400, 4800, 9600, and 19200 bps. The Tracker can be queried and tested using a standard ASCII terminal program.

All commands are printable ASCII strings, and provide feedback to tell the application if a command was successfully processed. All commands (except 'S') begin with a '!' (Attention) and end with a carriage return (Hex D). The Tracker responds with an 'O' for OK or an 'E' for an Error. The 'S' (Send data) command requests the Tracker to send data to the host. In this case, only the requested data is returned, and a time-out check must be used to determine if an error occurred.

Protected mode applications will need a bi-modal serial handler. A bi-modal serial handler for Watcom C/C++ is provided on the developer kit disk.

Using the i-glasses! Tracker

The Tracker must be initialized to a known state before an application can begin using the Tracker. Since the Tracker may be in a continuous streaming mode and at any of six supported bps rates, the host must send a reset command until successful. This will give the Tracker time to stop sending data and to change its communications rate if necessary. After the Tracker has been successfully reset, the host must put the Tracker into the appropriate mode for the application (polled, streaming, ASCII, binary, etc.). For emulation purposes, the Tracker retains its last operating modes (data mode, send mode, and send format) when it was powered off. All applications that directly support the Tracker must set and verify the operating modes on initial startup.

The i-glasses! Tracker command set

All commands to the Tracker are printable ASCII characters. Each command (except 'S') is terminated with a carriage return (Hex D). Result codes, 'O' and 'E' are ASCII 'O' and 'E'. Tracker orientation data is sent to the host in either ASCII or binary. The serial protocol is one start bit, 8 data bits, no parity, and one stop bit.

Command	Description
`!R`	Resets the Tracker to the default state: cooked, polled, binary mode. All applications should put the Tracker into Cooked, Euler, or an emulation mode before requesting orientation data.
`!V`	Get the Tracker revision string. This allows future revisions of the protocol to work as applications will know what version of the hardware they are talking to. The string format is: `M<16 chars>P<16 chars>T<8 chars>Hxxx.xxxFxxx.xxx` where M is for *Manufacturer* followed by a 16 character ID string, P is for *Product code* followed by a 16 character product code or serial number, T defines product *Type*, H is for *Hardware* revision followed by a C format "`%07.3f`" revision string, and F is for *Firmware* revision with the same C format string. An 'O' or an 'E' is appended to the end of the returned string to indicate whether an internal self test has passed or failed.
`!M`	`<data mode>,<send mode>,<send format>[,<magnetic filter>,<tilt filter>][,<Mouse sensitivity>,<Mouse threshold>]<CR>` Tells the Tracker to change data mode, send mode, send format, and filter modes, and is terminated with a carriage return (Hex D). Data modes are '0'-'4'. Send modes are 'P' for polled, 'C' for Continuous, and '0'-'1' in mouse mode. Send formats are 'A' for ASCII and 'B' for Binary. Filter ranges are '0' for none, and '7' for maximum. These parameters must be sent all at once, separated by commas, and in the defined order. Filter modes and mouse parameters are the only optional commands and shouldn't be set by the application, but rather by the Tracker manager software. If they are set by the application, the user must be able to change them. If the optional mouse parameters are set, the filter parameters must also be specified. The mouse sensitivity and threshold settings are fully described in the mouse emulation section. Only Tracker manager software should set mouse settings.
`S`	Tells the Tracker to send a packet of orientation data. In continuous modes, 'S' starts the stream. When a '!' is sent, the stream stops and the command is processed. '!' followed by a carriage return can be used to stop the stream.

Virtual i-O Tracker Modes

Mode 0: Raw data mode

The Tracker sends raw data readings from the sensors. The numeric format is 12 bits unsigned (0..4095) stored in 16-bits for all values. This mode is most useful for debugging the hardware.

Data packet format for mode 0 (binary)

The total packet size is 12 bytes. The byte format is:

Byte	Description
0	Header (always 255)
1	X-axis high byte
2	X-axis low byte
3	Y-axis high byte
4	Y-axis low byte
5	Z-axis high byte
6	Z-axis low byte
7	Pitch high byte
8	Pitch low byte
9	Roll high byte
10	Roll low byte
11	Arithmetic checksum (Bytes 0-10 added together)

Data packet format for mode 0 (ASCII)

The data is transmitted in the preceding form in ASCII hex with spaces separating each two byte ASCII hex value. This is for debugging only.

Send mode for data mode 0

The send modes for data mode 0 are 'P' for Polled and 'C' for Continuous. In continuous mode, a '!'<CR> command stops the stream and the 'S' command restarts it. To read data in continuous mode, the application searches for a start header (255). Once found, the rest of the packet must be read in, the checksum computed and compared to the packet's checksum. If the checksums don't match, the application must reread the data one byte beyond where it last found a 255 and start the process over again. Alternatively, the application can stop the stream with a '!'<CR>, pause a few character send times, flush its read buffers, then send an 'S' and begin reading the stream.

All raw modes should not be used for commercial applications. They are for factory debugging only.

Mode 1: Cooked data mode

The Tracker scales the magnetic vector, centering it about the zero, and linearizes the tilt sensor readings based on internal factory calibration constants. This is the mode to use when performing the angle computation on the host. The data format is signed 16-bit words. The x, y, and z magnetometer readings are approximately +/- 16384 (This varies with the Earth's magnetic field). The pitch and roll readings are converted to linear values where +16384 = 180 degrees and -16384 = - 180 degrees.

To convert to floating point: degrees = (float)reading/16384.0*180.0. A provided C library routine converts the cooked data values into yaw, pitch, and roll.

Data packet format for mode 1 (Binary)

The total packet size is 12 bytes. The byte format is:

Byte	Description
0	Header (always 255)
1	X-axis high byte
2	X-axis low byte
3	Y-axis high byte
4	Y-axis low byte
5	Z-axis high byte
6	Z-axis low byte
7	Pitch high byte
8	Pitch low byte
9	Roll high byte
10	Roll low byte
11	Arithmetic checksum (Bytes 0-10 added together)

Data packet format for mode 1 (ASCII)

The data is transmitted in the preceding form in ASCII hex with spaces separating each two byte ASCII hex value. This is for debugging only.

Send mode for data mode 1

The send modes for data mode 1 are 'P' for Polled and 'C' for Continuous. In continuous mode, a '!'<CR> command stops the stream and the 'S' command restarts it. To read data in continuous mode, the application searches for a start header (255). Once found, the rest of the packet must be read in, the checksum computed and compared to the packet's checksum. If the checksums don't match, the application must reread the data one byte beyond where it last found a 255 and start the process over again. Alternatively, the application can stop the stream with a '!'<CR>, pause a few character send times, flush its read buffers, then send an 'S' and begin reading the stream. Continuous mode is not recommended for commercial applications (error recovery is difficult and serial interrupts and CPU cycles are wasted).

Mode 2: Euler angles mode

The Tracker sends yaw, pitch, and roll angles. The data format for yaw, pitch, and roll is a signed 16-bit word, where +16384 = 180 degrees, and -16384 = -180 degrees.

Data packet format for mode 2 (Binary)

The total packet size is 8 bytes. The byte format is:

Byte	Description
0	Header (always 255)
1	Yaw high byte
2	Yaw low byte
3	Pitch high byte
4	Pitch low byte
5	Roll high byte
6	Roll low byte
7	Arithmetic checksum (Bytes 0-6 added together)

Data packet format for mode 2 (ASCII)

The data is transmitted in the preceding form in ASCII hex with spaces separating each two byte ASCII hex value. This is for debugging only.

Send mode for data mode 2

The send modes for data mode 2 are 'P' for Polled and 'C' for Continuous. In continuous mode, a '!' command stops the stream and 'S' command restarts it. To read data in continuous mode, the application searches for a start header (255). Once found, the rest of the packet must be read in, the checksum computed and compared to the packet's checksum. If the checksums don't match, the application must reread the data one byte beyond where it last found a 255 and start the process over again. Alternatively, the application can stop the stream with a '!'<CR>, pause a few character send times, flush its read buffers, then send an 'S' and begin reading the stream. Continuous mode is not recommended for commercial applications (error recovery is difficult and serial interrupts and CPU cycles are wasted).

Emulation Modes

Mode 3: Microsoft mouse emulation mode

When in binary mode and communicating with a mouse device driver, this mode operates at 1200 bps and simulates a 7 bit data byte (with 1 stop and 1 start) by always sending the last data bit as a simulated stop bit. The output format is defined by the Microsoft mouse data format (3 byte format).

While operating in mouse mode, X is determined by a scaled yaw angle calculation. Y is determined by a scaled pitch angle sensing.

Send mode 0 for mouse mode

This mode sends values like a mouse. Delta values are sent as long as the Tracker moves. The size of the deltas depends on how far the Tracker has moved.

Send mode 1 for mouse mode

When the Tracker is first initialized into this mode, a reference position is taken. Any movement away from this reference position results in deltas being continuously sent until the Tracker is moved back to within the threshold near the reference position. The size of the deltas depends on how far away the Tracker is moved from the reference position.

Sensitivity and mouse mode mickey values

One X mickey is 1/4 degree change in yaw and
one Y mickey is 1 degree change in pitch

for a sensitivity of 1. As sensitivity increases, the change per degree increases. Thus, a sensitivity of 2 represents 1/2 degree change in yaw, etc. A sensitivity of 0 can be used to disable X or Y mickeys. For example, DOOM works best with the Y axis disabled, where yaw causes the head to turn and pitch has no effect. The range for sensitivity is 0-9.

Mouse threshold settings

The threshold settings determine how far the Tracker has to move before a packet is sent. If the threshold is low, a movement in the Tracker will result in packets being sent frequently (small mickey counts sent frequently). If the threshold is high, the Tracker must move farther before a packet is sent (large mickey counts sent infrequently). The threshold is related to sensitivity in that the movement values are first adjusted by sensitivity before being compared to the threshold settings. The threshold range is 0-9.

Mode 4: CyberMaxx emulation mode [*Not yet implemented*]

Examples

Modes for terminal debugging

!M0,P,A,0,0   R+aw polled mode, ASCII, no filtering.

!M0,C,A,0,0   Raw continuous mode, ASCII, no filtering.

Modes useful for applications

!M1,P,B   Cooked polled mode, binary, filtering not changed.

!M1,P,B,0,0   Cooked polled mode, binary, no filtering.

!M1,P,B,3,3   Cooked polled mode, binary, medium filtering.

!M1,P,B,7,7   Cooked polled mode, binary, full filtering.

!M2,P,B,3,3   Euler polled mode, binary, medium filtering.

Emulation modes

Mouse mode

!M3,P,A,0,0,2,2 Mouse mode, polled, ASCII, no filtering, low threshold settings. This is for debugging only.

!M3,C,A,0,0,2,2 Mouse mode, continuous, ASCII, no filtering, low threshold settings. Transmits only when the mouse moves. For debugging only.

!M3,C,B,3,3,2,2 Mouse mode, continuous (whenever the mouse moves), binary, medium filtering, low threshold settings. For true mouse emulation, this command must be sent at 1200 bps to put the Tracker into 1200 bps mode.

Sample Code

```c
/* test.c: Simple program that prints data to the screen. */
/* Uses src\simple\vstrack1.lib. See the devkit disk for more info. */
/* Created 2/17/95 */
/* John Schultz */

#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
#include <math.h>
#include "vstrack1.h"

void main(int argc,char ** argv) {
TrackerData td;
TrackerStatus ts;

  ts = initTracker(&td,TP_COM1,9600,timerSecs(2));
  printf("%s.\n",trackerInfo(ts));
  if (ts != TS_OK) {
    closeTracker(&td);
    exit(0);
  } // if

  ts = sendTrackerCMD(&td,"!M1,P,B\r",timerSecs(2));
  if (ts != TS_OK) {
    printf("%s.\n",trackerInfo(ts));
    closeTracker(&td);
    exit(0);
  } // if

  requestTrackerData(&td);

  while (1) {
    float y,p,r;
    if (kbhit() && getch() == 'q') break;

    ts = readTracker(&td,timerSecs(1));
    if (ts != TS_OK) {
      printf("\n%s.\n",trackerInfo(ts));
      resetTracker(&td,timerTSecs(1));
    } // if
    requestTrackerData(&td);

    y = TOFLOAT(td.euler.y);
    p = TOFLOAT(td.euler.x);
    r = TOFLOAT(td.euler.z);

    printf("x %6ld y %6ld z %6ld y %6.2f p %6.2f r %6.2f\n",
      td.magnetic.x,td.magnetic.y,td.magnetic.z,y,p,r);

  } // while

  closeTracker(&td);

} // main

/* test.c */
```

Virtual i-O Head Tracker
Version Compatibility
5 June 1995

We at Virtual i-O are working constantly to improve the quality and ease of use of our Head Tracker and its accompanying software. During the early months of 1995, we released new and improved versions of the tracker as soon as they were ready for production. As a result, there are currently different versions of the Head Tracker in the field. To run your applications efficiently on past versions of the tracker, and to have them run well on future versions, your application program should tailor its use of the Head Tracker to each specific version.

The Virtual i-O Head Tracker makes this as easy by providing **functional compatibility** and **version information**. Each application must make proper use of <u>both</u> of these to maximize its performance.

Functional Compatibility

Virtual i-O provides functional compatibility in new versions of the Head Tracker. Version n of the Head Tracker contains all the functions in version $n - 1$. This guarantees that applications written for the functionality of today's tracker will work with tomorrow's tracker.

The Head Tracker has four data modes:

      0)   Raw
      1)   Cooked
      2)   Euler
      3)   Mouse
      4)   CyberMaxx™

Euler Mode is the standard application interface to the Head Tracker.

We recommend using Euler Mode to maximize functional compatibility. Euler Mode is the most device-independent mode. In Euler Mode, yaw, pitch, and roll in degrees are sent from the Head Tracker. Thus, the application need not know anything about the underlying technology of the Head Tracker.

Cooked Mode is a device-dependent mode. In Cooked Mode, filtered readings from the sensors in normalized (scaled and centered) volts are sent from the Head Tracker. Thus, the application must know about the underlying magnetic and gravometric technology of the Head Tracker, and must compute yaw, pitch, and roll itself. In early versions of the Head Tracker, Cooked Mode was faster than Euler Mode; so applications with excess computer power could trade device independence for speed by using Cooked Mode instead of Euler Mode. Even then, using Euler Mode enabled applications to run faster on slow computers. *In future versions of the Head Tracker, Cooked Mode may be slower than Euler Mode*; so applications should use Euler Mode if at all possible.

Raw Mode is the most device-dependent mode. In Raw Mode, unfiltered reading from the sensors in un-normalized volts are sent from the Head Tracker. *Raw Mode should only be used for troubleshooting the Head Tracker.*

Mouse Mode and CyberMaxx Mode are for use with applications <u>not</u> written for the Head Tracker. *Applications should not use Mouse Mode or CyberMaxx Mode.*
If an application written today wants to work with yesterday's tracker, it must do one of two things:

      1)   limit its use of functions to only those in older versions of the Head Tracker, or
      2)   check the version of the Head Tracker being used with the application and adjust your software accordingly.

Since limiting functionality is not desirable, version checking is highly recommended.

Version Information

An application can check which version of the Head Tracker is being used. The application sends a Version Command to the Head Tracker, and the Head Tracker send the version information back to the application. The application can then use the version information to decide which functions of the Head Tracker it will use.

The version command sent by the application to the Head Tracker is:

     `!V`

The version information sent by the Head Tracker to the application is:

     `MVirtual◊i-O.◊◊◊◊P00000001499.000,Ttracker◊H`*hhh.hhh*`F`*fff.fff*`O`

◊ is the space character, *hhh.hhh* is the hardware version number, and *fff.fff* is the firmware version number. The digits of these version numbers to the left of the decimal point indicate versions that changed the functionality of the Head Tracker. The digits to the right indicate versions that corrected problems in the functionality. The hardware and firmware versions released so far are shown in the following tables. 1499.000 is the Virtual i-O part number. This number may or may not change depending on version number. Virtual i-O will keep a current list of all tracker updates and will disseminate this information via mass mailings with names taken from our developer database. We will also provide this information in new versions of the Developer Kit manuals.

Hardware Version Number	Date	Interface Specification Numbers	Description
001.000	2/7/95	1.0	Initial release
001.001	4/25/95	1.0	Improved A/D converters

Firmware Version Number	Date	Interface Specification Numbers	Description
001.000	2/7/95	1.0	Initial release
001.001	2/15/95	1.0	Improved filtering; improved serial driver
001.002	3/13/95	1.0	Fixed mouse mode bug in Logitech support
001.003	4/25/95	1.0	Fixed Euler Mode bug in sign of yaw

All trackers released before version 001.003 need to have the sign (+ or -) for yaw switched in the application when using Euler mode. For example, if you are getting a "-12546" yaw reading from the tracker, switch it to "+12546" before processing.

Mouse Emulation Mode for *i-glasses!*™ Head Tracker
Version 1.2

Section C

Mouse Emulation Mode for Virtual i-O i-glasses!™ Head-tracker
Copyright 1995 Virtual i-O
All Rights Reserved

Mouse emulation

Mouse emulation allows PC game players to use the Virtual i-O Head-tracker (hereafter, "Tracker") on games that do not yet support the Tracker directly. Today, there are many games on the market that allow the player to either move or change their viewpoint using a mouse (hereafter, a mouse refers to any pointing device). The Tracker mouse emulation supports these types of mouse-controlled games. Games that *require* mouse button presses will work only on computers that can support two mice at the same time. Such computers will allow two mice to be physically connected and active at the same time, and will have a mouse driver that can support two mice at once (Such as Logitech's mouse driver, via the *mouse dual* command).

When two mice are connected at once (the Tracker and the pointing device), the Tracker can be used to change the player's viewpoint or position while the mouse's buttons can be used to activate features in the game. When set up this way, both mice are active. If both mice move at the same time, both actions take effect (they're summed up). Most games that support mice also have equivalent keyboard commands and won't require an additional mouse in addition to the Tracker.

In this manual, a serial mouse refers to any pointing device that plugs into a serial port. PS/2, Bus, and Inport mice are referred to as non-serial mice.

For more information on setting up and using a mouse and mouse drivers, see your mouse documentation.

How to use the Tracker in a mouse emulation mode

The Tracker is put into mouse emulation using Virtual i-O's TRACKER.EXE software.

If you have:		See section:
No serial port.		**1**
No mouse/pointing device.	**2**	
Serial port(s) present, but none free, non-serial mouse.		**3**
Serial port(s) present, but none free, serial mouse.		**4**
Serial port(s) present, one or more free, non-serial mouse.		**5**
Serial port(s) present, one or more free, serial mouse.		**6**

Section 1: No serial port.

Adding a serial port

If you do not have a serial port installed on your machine, you will need to add one. Serial cards are available at most computer stores. They usually come with 16550A UARTS (a more advanced UART than the older 8250/16450, and are capable of higher speed operation with less errors due to enhanced hardware buffering). These cards typically come with two serial ports, take up a very short 8-bit slot, and cost less than $40.

Some serial port cards allow non-standard port addresses and interrupts, allowing you to use more than two serial ports simultaneously. Application software must be able to support non-standard port addresses and interrupts. An example of an application that could use three serial ports at once would be a multi-player game that uses a mouse, a serial port connection (direct or modem), and the Tracker.

After a serial port card has been added to your computer, see section 8 or 9 below.

Section 2: No mouse/pointing device.

A mouse driver is required for applications that support mouse input. In order to use the Tracker in a mouse emulation mode, a mouse driver must be installed. Mouse drivers come with mice, trackballs, pointing sticks, touch pads, and drawing tablets, etc. The Tracker provides mouse x and y output only (no buttons), and cannot be used as a standalone mouse. Thus, to use the Tracker for mouse emulation, a mouse device and driver must be present. Mouse driver installation and setup is described in the mouse documentation. If a mouse driver is present, see section 9 below.

Section 3: Serial port(s) present, but none free, non-serial mouse.

An A/B switch box is required (see below), or an additional serial port must be added (see also section 1).

Adding an A/B (two position) RS-232 Serial Switch Box

An A/B serial switch box will allow you to connect two serial devices to the same serial port. By flipping the A/B switch, you can switch between two different devices without turning your computer off (Do not plug in or unplug anything while the computer is powered on). A 9 pin or 25 pin A/B switch box can be used. For example, if you are going to connect the switch box up to a 9 pin serial port, you will need a 9 pin A/B switch box. A switch box with all male connectors would be optimal, but a female switch box can be used with gender benders. You can also use a 25 pin switch box, but 9 pin to 25 pin adapters will be necessary. If your serial port is 25 pin, then a 25 pin switch box will be optimal. You will need a 9 to 25 pin adapter for the Tracker. A serial cable from the computer to the switch box will also be necessary. The gender and connector type of this cable will depend on the switch box chosen.

There are many different combinations of connectors and switch boxes possible. If you are unsure of what you need for your situation, write down the number of pins and gender of your serial port (9 or 25 pin, male), the number of pins and gender for the Tracker (9 pin, female), then tell your computer dealer that you want to connect two serial devices with these pinouts and genders to one serial port and switch between them using an A/B switch box.

After a switch box has been added, see section 8 below.

Section 4: Serial port(s) present, but none free, serial mouse.

An A/B switch box is required (see section 3 above), or an additional serial port must be added (see section 1). If an A/B switch box is used, the Tracker will share the same port as the mouse. See sections 7 and 8 below.

Section 5: Serial port(s) present, one or more free, non-serial mouse.

Plug the Tracker into an available serial port. If the port has a 25 pin connector, a 9 to 25 pin adapter will be required. A two-mouse mode capable mouse driver is required. See section 8 below.

Section 6: Serial port(s) present, one or more free, serial mouse.

Plug the Tracker into an available serial port. If the port has a 25 pin connector, a 9 to 25 pin adapter will be required. If a two-mouse mode driver is not available, an A/B switch box (see section 3) can be used with the Tracker sharing the same port as the mouse. See sections 7 and 8 below.

Running Applications with the Tracker in Mouse Emulation Mode

Section 7: Mouse and Tracker on the same serial port with a switch box

If you are using a switch box with the Tracker and mouse on the same port, put the switch on the Tracker and put the Tracker into the desired mouse emulation mode using TRACKER.EXE. Then, switch the box to the mouse and start your game. Once past all menus and in the game, flip the switch to the Tracker and begin playing. If you must access the user interface, flip the switch back to the mouse, perform the required user interface actions, then flip the switch to the Tracker when back in the game. When done playing the game, flip the switch back to the mouse for use in other mouse controlled applications.

Section 8: Mouse and Tracker on different ports running at the same time (Two-mouse mode)

If you have the Tracker plugged in and a mouse device available at the same time (Two serial ports present with a serial mouse on one port and the Tracker on another, or a PS/2, Bus, or Inport mouse with the Tracker on a serial port), you will need to use a two-mouse mode capable mouse driver. In this case, put the mouse into two-mouse mode (*mouse dual*, for the Logitech mouse driver). Keep the Tracker steady while you set up your game with the mouse (place the Tracker on a steady, stable surface). Once in the game, put the Tracker on and begin playing. You can use the mouse at the same time as the Tracker to control the game.

Section 9: No mouse installed and the Tracker on a serial port

If you are going to run a game that allows user interface and game control with the keyboard (In addition to a mouse, such as DOOM^{tm}), you can run the game with just the Tracker acting as a mouse. Put the Tracker into a mouse emulation mode using TRACKER.EXE, then run your mouse driver (set up for the serial port the Tracker is plugged into), then start the game. Use the keyboard to get set up, then begin playing the game. The Tracker can then be used for positional and directional control in addition to the keyboard.

Final Notes

The Tracker must be put into mouse emulation before running mouse controlled games that do not support the Tracker directly. Applications that support the Tracker directly will not need to use mouse emulation. But, if an application is run that supports the Tracker directly, a mouse emulation supported game will require the Tracker to be put into mouse emulation mode using TRACKER.EXE. This is due to the fact that the Tracker will stay in the last mode it was put into, even if it is powered off. If you only run mouse emulation supported games, you will only need to run TRACKER.EXE to change the way emulation works (such as sensitivity settings).

These mouse emulation mode setups are the most common and are easy to use. There are many different ways to use the Tracker in mouse emulation mode, so feel free to experiment. Have fun!

i-glasses!™ Stereoscopic 3D Graphics and Head Tracker Development Tools
Version 1.2

Section D

Virtual i-O i-glasses! Stereoscopic 3D Graphics and Head-tracker Development Tools

Software

The Virtual i-O i-glasses! development tools are based on IBM PC and compatible computers. The development tools include all source code necessary to create stereoscopic 3D head-tracking applications on any computer. The source code is ANSI-C and can easily be ported to other platforms. The serial library and graphics examples are designed specifically for the PC, but the concepts are easily ported to other architectures.

Hardware

Head-tracker (Tracker)

The i-glasses! Tracker can be interfaced to any device capable of communicating with a standard RS-232C serial 3-wire interface. The Tracker supports 1200, 2400, 4800, 9600, and 19200 bps. The higher communication rates (9600 and 19200) allow response times of less than 15ms with very little perceptible lag. The Tracker can auto-detect the communications rate and does not require any switches to be set by the user.

i-glasses! 3D Display System

Field and frame sequential stereoscopic 3D video

The i-glasses! can be used on any device capable of producing NTSC composite video. Stereoscopic 3D is supported by two methods. The first method is compatible with stereoscopic 3D videotape, and is called field-sequential stereoscopic 3D. This method uses the even scanlines for one eye, and the odd scanlines for the other eye. Since NTSC video is interlaced (where the even scanlines are displayed all at once, then the odd scanlines are displayed all at once), videotapes can be produced such that one eye's view is on the first field, and the other eye is on the second field. The i-glasses! video hardware then splits the images out to each eye. This is the industry standard way of producing stereoscopic 3D videotape. When computers create 3D images for this mode, they render two separate images into two separate frame buffers. The frames are then switched between at 60Hz to simulate an interlaced display. In this case, the 3D mode is frame-sequential, but to a video display, it appears just like an interlaced signal. If the computer is capable of generating interlaced NTSC composite video, then the images can be rendered in the same way as described below.

Interleaved stereoscopic 3D video

The second method is more suited to computer graphics and can use a non-interlaced display mode. This method is called interleaved stereoscopic 3D. Here, images are rendered into one frame buffer, where the first scanline represents the left eye, the second scanline the right eye, the third scanline the left eye, and so on for the entire display. Thus, a 320x400 frame buffer is split up into two 320x200 images (one for the left eye and one for the right eye) by the i-glasses! video hardware. For PC applications, the i-glasses! can take VGA (31.5kHz) compatible video signals directly (the 31.5kHz signal is converted to 15.75kHz). The current hardware requires VGA compatible input to support non-interlaced interleaved stereoscopic 3D.

Examples

A complete stereoscopic 3D application with head-tracking is included with the developer kit, and can run at up to 70 frames per second with negligible lag. The source code shows how to interface and communicate with the Tracker as well as how to produce stereoscopic 3D images. All examples are simple and concise, and can easily be applied to other computer architectures.

Interfacing to a Home Console Game System

Stereoscopic 3D (i-glasses!)

Since the many home console systems output NTSC composite video, they can be directly connected to the i-glasses! display system. If these systems can output interlaced video and can render left and right eye views to alternate scanlines (interleaved), then stereoscopic 3D can be easily supported. If the system you are working with can perform vertical retrace page-flipped programming, then it may be able to page flip with the vertical retrace interrupt. If this is the case, then frame-sequential stereoscopic 3D can be supported. For this mode, two images are rendered to two separate frame buffers (video pages), and the vertical retrace interrupt handler switches the display pages at 60Hz. To support this mode, a minimum of 4 video pages are needed (with 6 pages, triple buffering can be used for more speed). While the vertical retrace interrupt is constantly switching pages, the application renders to the other two off-screen pages. When both pages have been rendered, a flag is set to tell the vertical retrace interrupt routine to switch to the next two pages.

Head-tracking (Tracker)

In order to support the Tracker, a serial interface must be available. If the home system controllers are serial, then it may be possible to create a Y-adapter and allow one controller and the Tracker to be used simultaneously. If only one UART exists on the game port, then a serial port add-on will have to be used on one of the system's expansion ports.

While Virtual i-O is interested in having the widest range of supported systems availible to the end consumer, we do not officially support nor are we officially supported by any of the main home console game systems to date. When such support is available, Virtual i-O will include more indepth information for interfacing with these types of systems.

i-glasses!™ Video Specification
Version 1.2

Copyright 1995, Virtual i-O
All Rights Reserved

Section E

Virtual i-O i-glasses! Video Specification

Supported Video Modes

NTSC Composite
VGA modes 1h-13h, including unchained modes ('X')
Enigma 320x200

Video Timing Ranges

60 and 70 Hz vertical refresh (449 or 525 lines vertical total)
15.75-31.5kHz horizontal refresh

Resolution of 6000 series TFT LCD Panels

180,000 sub-pixels, for 256x230 triads (Computer addressable pixels). This resolution   comfortably supports 320x200 modes.

Supported Stereo-3D modes

Field Sequential

Field Sequential NTSC video (videotape). The left eye is defined as the first scanline in the first field (Closest to the top of the screen).

Frame Sequential

 VGA at 320x200, 320x400, 640x200, 640x400, and 640x480. The left eye is undefined.

Interleaved VGA

The left eye is the first scanline.
Interleaved 640x480: 640x240 per eye.
Interleaved 640x400: 640x200 per eye.
Interleaved 320x400: 320x200 per eye.

Interleaved Enigma

Interleaved 320x200: 320x100 per eye.

Programming Interleaved Stereoscopic 3D

A left eye and a right eye image must be rendered into an interleaved display format. The first scanline is the left eye, and the next scanline is the right eye (and so on, down the screen). The images can be rendered directly to video memory, or rendered off-screen and copied while interleaving to the video memory.

Source of Video Multiplexers

John Williamson
Stereoscopic 3D Video & Virtual Reality Technologist
Virtual i-O

Multiplexers, or 3D Converters, are systems which take two full video frames and combine them into a single stereoscopic, field sequential, 3D format which can then be viewed with a wide variety of 3D systems, including the Virtual i-O i-glasses. These systems will work only with video (NTSC or PAL), to date there does not exist a system which will multiplex for VGA (although there is a growing demand).

There are several applications which would cause someone to want this capability. These include having two, 2D cameras or inputs from either two computers or two outputs on a multi-channel option on an SGI.

The following is a short list of companies which are currently producing and selling these systems. *Inclusion on this list in no way constitutes an endorsement from Virtual i-O.*

Ikegami Electronics (U.S.A.), Inc.
37 Brook Avenue
Maywood, NJ 07607
Voice: 201-368-9171

Catalog Name: 3D Converter (Part of the LK-33 System) Price: $40,000

Comments: Have not reviewed it due to its high cost. Based on the quality of their camera, this is quite possibly the best system on the market, and possibly the only system to handle component input.

3D Video Plus
2 The Old House
36 Southend Road
Beckenham, Kent BR3 2AA
England
Voice/Fax: +44 181 650 4862

Catalog Name: 3D Video Encoder

Comments: Have not reviewed it. They do make a PAL version.

3D TV Corporation
P.O. Box Q
San Rafael, CA 94913-4316
Voice: 415-479-3516
Fax: 415-479-3316

Catalog Names: Model 100: $1,800
                              Model 200: $3,200
                              Model 300: $3,600

Comments: Have not reviewed. A PAL version is also available.

SOCS/3D
536 N. Santa Cruz Blvd., #201
Los Gatos, CA 95030
Voice: 408-354-9050
Fax: 408-354-0600

Catalog Name: S3D-1110 Video Field Multiplexer. Cost: $1,500

Comments: Seems to be a very good system for the money. Supports S-Video (slightly higher than NTSC, but lower than component).