

Configuring BodyExample-based User Clients for MASSIVE-3

Chris Greenhalgh
31st May 2000. Version 3.

Introduction

The class BodyExample provides a basis for the standard user client programs for MASSIVE-3 (including bodyExampleClient, SofaClient, hiveClient and MuseumClient). Run-time configuration of BodyExample is achieved through two means: environment variables; and a configuration file. These are described in this document. Use of the configuration file is the main and preferred mechanism in most cases.

Environment Variables

Specific Environment Variables

BodyExample depends upon the following environment variables:

- HOME: the directory in which BodyExample looks to find the user's configuration file.
- USER: the name to be used for the user within the virtual world. Setting this environment variable overrides the configuration parameter "UserName" (which has default value "X").
- TEST_CONSTRAINT: if set, the value of this environment variable is applied to the user as a movement constraint. This is normally only used for testing. The format of a constraint is described in the MgmtServer API document.
- MOVE_RATE_S: the time, in seconds, between movement updates being sent to remote processes. If set, this environment variable overrides the configuration parameter "MoveRateS" (which normally has the default value "0.5", although this may be over-riden by particular client programs).

General Environment Variables

In addition, it also makes use of the following environment variables used by all MASSIVE-3 programs:

- CRGPATH: the directory path (separated by ':' on UNIX or ';' on Windows), which is searched to locate files including a default configuration file, geometries, and textures.
- APPLOGFILE: the file to which APPLOG logging output should be directed. Special values "stderr" and "stdout" identify the standard error and output streams, respectively. The default is not to output any APPLOG logging output.
- APPLOGOPTS: a search path which determines which APPLOG logging output is actually allowed, and which suppressed. Entries are separated by ':' or ';' and processed in order; the first matching entry determines is applied. Entries are positive (allowing), or negative (disallowing). Negative entries begin with a '-' character. Each entry is a dot-separated sequence of names. Each entry implicitly ends in a ".*" wildcard, matching all logging names have this as a prefix. Most MASSIVE-3 logging names begin "HIVEK.".
- TTL: the TTL used for multicast communication (if used), default "1".
- HIVE_MIN_PORT: the smallest port number to be used for communication, default "0". Useful when configuring for use with partial firewalls.
- HIVE_MAX_PORT: the largest port number to be used for communication, default "65535". Useful when configuring for use with partial firewalls.
- HIVE_MC_BASE_ADDR: the lowest multicast address to be allocated to MASSIVE-3 processes. Default "239.0.0.0".
- HIVE_MC_ADDR_MASK: bit-mask determining the range of multicast addresses to be allocated. A random number is bit-wise logical ANDed with this, and added to the value of HIVE_MC_BASE_ADDR to determine a multicast group. Default "0.255.255.255".
- LOCALHOST: dotted decimal form IP address to use for the local machine. Useful for multi-homed machines, and for machines which are running standalone (when it should normally be set to "127.0.0.1"). Default is the value returned by the system call "gethostname", normally the main IP address allocated to the machine.
- IGNORE_SIGINT: which causes the application to ignore the INT signal. In some cases (on some platforms, e.g. IRIX) this is necessary to allow the program to be interrupted and resumed which debugging, e.g. using dbx (otherwise the program catches the INT signal and subsequently terminates).

Configuration File Options

The Configuration File Format

BodyExample (and its derivatives) use a configuration file called “massive3.config”. To locate this file the program first searches in the directory specified by the “HOME” environment variable. If not file of this name is found (or if it is unreadable) then the program searches the elements of the “CRGPATH” environment variable in turn, until a file named “massive3.config” is located, and this is used.

The Configuration File Format

This file consists of a sequence of lines, each beginning with a name or tag (which is a single word), followed by a colon (‘:’), and a value. There are several standard value types:

- **Boolean:** the value true is indicated by the first character of the value being one of the following characters: ‘y’, ‘Y’, ‘t’, ‘T’, or ‘1’. All other values (including an empty value) are interpreted as false.
- **Integer:** the value is parsed as a decimal integer, with optional leading minus sign. Only simple values are accepted (e.g. no expressions, scientific notation or exponential notation).
- **Float:** the value is parsed as a decimal floating point number, with optional leading minus sign and optional scientific/exponential notation.
- **String:** the value is taken from the first non-whitespace character to the last non-whitespace character on the line. Only single line values are allowed. Any quote characters are considered to be part of the value.
- **Point3D:** the value has the form of three comma-separated floats, which are the X, Y and Z components of the point.
- **Quaternion3D:** the value has the form of three comma-separated floats, which are Euler angles, representing consecutive rotations about the X, Y and Z axes.

There is no standard way of including comments in the config file, and nor should it include blank lines. These problems may be remedied at some point; in the mean time entries with the name “Comment” can be used for annotations.

Configuration Domains

The configuration file is used for the following purposes, describe in detail in the following sections:

- Defining user identity.
- Defining the size and appearance of the user’s avatar, including “breathing” and bobbing up and down while moving.
- Defining if and how used activity/inactivity is conveyed via the user’s avatar.
- Defining the components present in the visor.
- Defining the display and rendering characteristics.
- Controlling visual feedback of audio activity and awareness.
- Controlling aspects of navigation.
- Controlling object manipulation.
- Tracker configuration and calibration, for immersiveClient only.

User Identity

- **UserName:** specifies the name given to the user’s avatar. This is the value given to attributes on the user’s avatar, with names “PLAYER” and “USER”.

Avatar Appearance

Body Geometries:

- **UserTopGeometry:** a geometry which is positioned at the “top” of the user’s avatar. This is actually normally on the ground plane. The orientation of “top” is the normal direction of motion. In the case of an immersive user, “top” is the virtual equivalent of the actual physical space in which the user is tracked, i.e. tracker movements relative to the tracker reference become virtual movements relative to “top”. UserTopGeometry can be used to place a virtual “shadow” for ground-locked avatars. The default is to omit this geometry. Forward (the direction in which the user is nominally facing) is defined to be along the negative Z axis.
- **LocalUserTopGeometry:** this is a second geometry, equivalent to UserTopGeometry except that only the user will see, and not other observing processes (it is a local data item). This is useful to give immersive

users a graphical representation of the useful tracking volume which is not visible to other users. The default is to omit this geometry.

- **UserBodyGeometry:** a geometry which represents the user's torso and pelvis. This is normally positioned directly below the head geometry (e.g. in the case of an immersive user with tracked head but no tracking for the body). The front of the body is in the negative Z direction, positive Y is up, and positive X is to the right. The default value is "example_body.dgl". Nominal sizes and positions are given below. If alternate appearances are being specified then "UserBodyGeometry2" denotes the first alternate appearance, "UserBodyGeometry3" the next, and so on.
- **UserHeadGeometry:** a geometry which represents the user's head. The user looks along the negative Z axis. The default value is "example_head.dgl". Nominal sizes and positions are given below. If alternate appearances are being specified then "UserHeadGeometry2" denotes the first alternate appearance, "UserHeadGeometry3" the next, and so on.
- **LeftHandGeometry:** a geometry which represents the user's left hand. For desktop clients this is essentially decorative, or may be able to perform simple scripted motions. The nominal orientation of this geometry is with the finger extended along the negative Z axis, with the thumb extended along the positive Y axis. The default value is "example_left_hand.dgl". Nominal sizes and positions are given below. If alternate appearances are being specified then "LeftHandGeometry2" denotes the first alternate appearance, "LeftHandGeometry3" the next, and so on.
- **RightHandGeometry:** a geometry which represents the user's right hand. For desktop clients this is essentially decorative, or may be able to perform simple scripted motions. The nominal orientation is as for the left hand. The default value is "example_right_hand.dgl". Nominal sizes and positions are given below. If alternate appearances are being specified then "RightHandGeometry2" denotes the first alternate appearance, "RightHandGeometry3" the next, and so on.
- **UserEyeGeometry:** a geometry which (on the desktop clients) tracks the mouse's motion within the graphical scene. The default is to omit this geometry.
- **LocalUserEyeGeometry:** a geometry which (on the desktop clients) tracks the mouse's motion within the graphical scene but is only visible to the user. The default is "example_crosshair.dgl". Note that the locking of head position during navigation can lead to a mismatch between mouse position and eye position which may hinder use if not represented.
- **UserEyelidGeometry:** a geometry which overlays the eye geometry, and blinks occasionally. It can also be used to convey inactivity (in which case it closes progressively); configuration for this is described in a later section. The default is to omit this geometry.
- **ConstraintGeometry:** a geometry which is made visible when the user is actively constrained (by a visible constraint). The default is to omit this geometry.
- **ConstraintGeometry2:** a geometry which is made visible, as an alternative to the above, when the user is constrained (by a visible constraint) but not currently pushing against that constraint. The default is to omit this geometry.
- **UserHandGeometry:** a geometry which is made visible when the user attempts to pick up an object. For the desktop client its parent is the user's eye (since the mouse moves within the graphical view). For an immersive client its parent is normally one of the immersive user's hands. The default value for this geometry is "example_hand.dgl", and is normally a line.
- **UserHandGeometry2:** an alternative geometry which is visible while the user is not attempting to pick up an object. This can be especially helpful for immersive users attempting to determine when they are correctly pointing at an object in order to pick it up. The default is to omit this geometry (which can be rather visually distracting to bystanders).

Body Size and Positioning:

- **UserScale:** an overall scale-factor applied to the user's embodiment. This scale affects all geometries and positions, and need not be taken into account when initially sizing geometries and positions. The default value is "1.0". If alternate appearances are being specified then "UserScale2" denotes the first alternate appearance, "UserScale3" the next, and so on.
- **UserBodyPos:** a Point3D which specifies the position of the user's body relative to the user's "top" (see UserTopGeometry, above). This point nominally corresponds to the centre of the user's pelvis. The default value is actually "0,0,0", but a good working value would be "0,0.9,0", i.e. a "leg" length of approximately 0.9 metres. This value is ignored by the immersive client, which takes the user's head position from the tracker input, and uses UserHeadPos (below) to calculate an appropriate body position, moment by moment.
- **UserHeadPos:** a Point3D which specifies the position of the user's head relative to the user's body. This point nominally corresponds to the centre of the spine around the base of the skull, i.e. slightly behind and below the centre of the head. The default value is "0,0,0", but a good working value would be "0,0.65,0".

- **UserEyePos:** a Point3D which specifies the position of the user's eye relative to the user's head. The default value is "0,0,0", but this should normally be over-riden to move the eye point (used for rendering) out of the head geometry. In particular, some -ve Z component is likely to be required.
- **UserLeftHandPos:** a Point3D which specifies the resting position of the user's left hand geometry. Default value is "0,0,0", but a better starting value would be "-0.3,0,0".
- **UserLeftHandRot:** a Quaternion3D which specifies the resting orientation of the user's left hand geometry. The default value is the identity quaternion, but the value "-90,0,0" will rotate a reference hand geometry (fingers extended along the negative Z axis) to a more natural resting position.
- **UserRightHandPos:** a Point3D which specifies the resting position of the user's right hand geometry. Default value is "0,0,0", but a better starting value would be "0.3,0,0".
- **UserRightHandRot:** a Quaternion3D which specifies the resting orientation of the user's right hand geometry. The default value is the identity quaternion, but the value "-90,0,0" will rotate a reference hand geometry (fingers extended along the negative Z axis) to a more natural resting position.

The actual sizes of geometries can vary, however immersive users will find accurate geometry sizes easier to work with, e.g. body approximately 0.6 metres above origin, approx. 0.4 m wide, approx 0.2 m deep, head approx. 0.2 m diameter, hand approx. 0.2 long and 0.1 m wide.

Appearance control:

- **Visible:** flag determines whether the user's embodiment is visible (or invisible). Default value is true.
- **UserControlledVisible:** flag determines whether the user can make their own avatar visible and invisible (pressing 'v'). Default is true.
- **UserControlledAppearance:** flag determines whether the user can change their own avatar's appearance between a pre-set range of alternatives (set using UserHeadGeometry2, etc.). Default value is true.

Activity Indication

There are two standard ways for BodyExample to indicate that it has not detected any recent activity from the local user: by "slumping" the avatar's head, or by closing the avatar's eye (note that no eye geometry is specified, by default). Configuration is as follows:

- **ShowActivityFlag:** determines whether either mechanism is used. Default is true.
- **SlumpBodyFlag:** determines whether (in)activity is indicated by the head slumping. Default is true.
- **UserEyelidGeometry:** if specified (see above), implicitly enables this method of showing activity, provided that ShowActivity is also true.
- **SlumpStart:** specifies time in seconds before slumping/eye closing begins (floating point value). Default value is "10.0".
- **SlumpEnd:** specifies time in seconds before slumping/eye closing is complete (floating point value). Default value is "30.0".
- **SlumpStep:** specifies time in seconds between increments in degree of slumping/eye closing (floating point value). Default value is "2.0".
- **EyelidOpenAngle:** specifies the angle in degrees about the positive X axis that the eyelid geometry should be rotated in order to convey activity (i.e. to be open). The default value is "60.0".
- **BlinkInterval:** specifies the interval, in seconds, between normal blinks of the avatar's eyelid geometry (if present). Default value "10.0". Pressing the spacebar may also force an eye blink.
- **BlinkDuration:** specifies the duration, in seconds, for which the eyelid is "closed" during a blink. Default value is "0.1".

Visor

The visor is visible only to the user, and can contain indicators and other widgets for interaction. It is configured as follows:

- **VisorShowConstraint:** whether the presence/activity of constraints on user movement are displayed on the visor. Default is true.
- **VisorShowVehicle:** whether the visor includes some navigational widgets allowing flying and other movement. Default is true.
- **VisorShowCompass:** whether the visor shows a simple compass indicating the direction of the negative Z axis (nominal "north"). Default is true.
- **VisorShowDropIcon:** whether the visor shows a button for explicitly dropping carried items. This is not required in some interaction modes (some clients). Default is true.

Display and Rendering

- **ReplicationPolicy**: specifies the replication policy to use with regard to locales and aspects. Default value is “0-0-0=STEP1”, i.e. all aspects up to topological distance 1. Is over-riden by an attribute of this name in the avatar’s home aspect (i.e. in which their embodiment is currently created).
- **RenderPolicy**: specifies the rendering selection policy to use with regard to locales and aspects. Default value is “0-0-0=STEP1”, i.e. all aspects up to topological distance 1. Is over-riden by an attribute of this name in the avatar’s home aspect (i.e. in which their embodiment is currently created). Is reset whenever the user crosses a boundary into a new locale.
- **RenderWidth**: width of the 3D graphical view. Default value is “640”.
- **RenderHeight**: height of the 3D graphical view. Default value is “480”.
- **RenderNearClip**: near clipping distance for the 3D graphical view. Default value is “0.1”.
- **RenderFarClip**: far clipping distance for the 3D graphical view. Default value is “1000”.
- **RenderVertFov**: vertical field of view of the 3D graphical view, in degrees. Default value is “60”.
- **PrintFrameRate**: flag determining whether frame rate is printed continuously (can be printed once by pressing ‘F’). Default is false.
- **FrameTimeAlpha**: geometric smoothing constraint used to smooth frame times to determine average rate. Default value is “0.1”.
- **MarkupSolids**: subjectively display nearby solid objects. Default is false. Can be toggled using key F8.
- **SolidAuraSize**: how far away the program should look for solid objects, for markup and avoidance (according to MarkupSolids and AvoidSolids). Default value is “10.0” (metres).
- **BoundaryMarkup**: subjectively display boundaries in the current aspect. Default value is false. Can be toggled using key F3.

The viewpoint can be configured as follows:

- **ViewpointPos0, ViewpointPos1, ViewpointPos2**: offset of the eye from head position for viewpoint 0, 1 and 2. Default value is nominal eye position (UserEyePos) (viewpoints 0 and 2), and nominal eye position (UserEyePos) + (0,1,3) (viewpoint 1). Viewpoint are selected using F5 (viewpoint 0), F6 (viewpoint 1) and F7 (viewpoint 2).
- **ViewPos0, ViewPos1, ViewPos2**: a further offset from the eye of the rendering origin for viewpoint 0, 1 and 2. Default value is “0,0,0” (viewpoints 0 and 1) and “0,1.5,2” (viewpoint 2).
- **ViewRot0, ViewRot1, ViewRot2**: a rotation applied from the eye to the rendering origin for viewpoint 0, 1 and 2. Default value is “0,0,0” (viewpoints 0 and 1) and “0,180,0” (viewpoint 2).
- **ViewFromTop0, ViewFromTop1, ViewFromTop2**: a flag indicating whether rendered view should be relative to eye (false) or “top” (true). Default value is false (viewpoints 0 and 1) and true (viewpoint 2).
- **InitialViewpoint**: the index of the viewpoint to be used on starting the client. Default value is “0”.
- **RestrictedViewpoints**: flag indicating whether the user is allowed to change their own viewpoints (using F5, F6, F7). Default value is “false”.

Audio Activity and Awareness

- **UserMouthGeometry**: a geometry which appears when the user’s audio is active (as determined by the audio server’s silence detection algorithm). The size of the geometry is determined by the following parameters: MouthGeomScale. Default value is “example_mouth.dgl”.
- **MouthGeomScale**: specifies a scale-factor for the UserMouthGeometry as its largest size. Default value is “1.0”.
- **AudioNimbus**: specifies an audio nimbus to be used for this user, which affects how they are heard in the current environment. The default is not to specify an audio nimbus. Audio nimbus is represented by an attribute with name “AUDIO_NIMBUS”. Allowed values (in C-style printf format) are: “set-cos-theta value=%f”, which fixes the left/right pan of the item (1.0 places it on the right, -1.0 places it on the left); “level value=%f”, which gives a global level; “distance-pwl n=%d distances=%f[,%f...] values=%f[,%f...]”, which specifies a piece-wise linear function of distance (distances and values are comma-separated lists, each with ‘n’ elements); “local-box value-in=%f value-out=%f size=%f,%f,%f offset=%f,%f,%f”, which specifies a box, centred on the user (subject to the specified offset), with given size, inside which audio nimbus is ‘value-in’ and outside which it is ‘value-out’.
- **hearLevelShow**: determines whether a subjective display is used to show the user how well other user can head them, according to audio awareness. This decorates the other users with an additional locla geometry, which is resized to represent the level of audio awareness which they should have of this user. Default is true.
- **hearLevelGeometry**: specifies the geometry to be used for this. Default value is “hear_level.dgl”.

- `hearLevelMinScale`: specifies the minimum scale to be used, to represent zero awareness. Default value is “0.0”.
- `hearLevelMaxScale`: specifies the maximum scale to be used, to represent full (1.0) awareness. Default value is “1.0”.
- `hearLevelScaleX`: flag specifying whether the scale should be applied in the X axis. Default value is true.
- `hearLevelScaleY`: flag specifying whether the scale should be applied in the Y axis. Default value is true.
- `hearLevelScaleZ`: flag specifying whether the scale should be applied in the Z axis. Default value is true.
- `hearLevelYOffset`: specifies an offset to be applied to the hear level geometry when placing it relative to the audio source/sink geometry (i.e. the other user’s head, normally).

Navigation

Navigation includes response to mouse movement, visual “bobbing” while moving, terrain following and subjective grid display. First, mouse input:

- `maxMouseHeadXRotate`: when the mouse is moved in the graphical view, the head can be set to exaggerate this motion. This can allow the user to look around a larger field of view, just by moving the mouse. This can be effective at high frame rates, but can become awkward or uncomfortable at lower frame rates (e.g. below 10-15 Hz). The default value is “70.0” (degrees).
- `maxMouseHeadYRotate`: this is the Y (vertical) correspondent to `maxMouseHeadXRotate`. The default value is “60.0” (degrees).
- `EyeMaxXRotate`: the eye geometry follows the mouse in the graphical view. The head geometry can be constrained to follow the eye to a lesser extent, i.e. the eye is allowed a certain freedom of orientation, but at larger angles the head will also turn. This degree of eye freedom before the head turns is specified (in the X/horizontal direction) by this parameter. The default value is “20.0” (degrees).
- `EyeMaxYRotate`: this is the Y/vertical correspondent to `EyeMaxXRotate`.
- `MouseTurnLimit`: if the mouse is close to the edge of the screen with any button pressed (e.g. picking up an object) then it can cause the user to begin to turn in that direction (so that the user can move the object to a position currently out of view). This parameter determines how close to the edge of the screen the mouse has to be before this function occurs. The default value is “0.85” (fraction of the distance from the centre of the screen to the edge).
- `MouseTurnRate`: determines how quickly the user turns after exceeding the `MouseTurnLimit`, above. Default value is “5.0”.
- `MouseDollyRate`: determines how quickly the user flies into the air or slides sideways when Control and left mouse button are pressed together while moving the mouse in the graphical view. Default value is “0.3”.
- `MaximumSpeed`: determines the maximum rate of allowed forwards/backwards movement. A value of 0 indicates no limit. Default value is “0.0” (metres/second).

Visual “bobbing”. As the user moves, their body may be made to bob up and down. This can give a greater appearance of activity in movement, and may also provide additional parallax-based distance queues to the user while moving. This is controlled as follows:

- `BobVariation`: total range of Y offset comprising bobbing motion. Half is positive, and half is negative, applied in a sinusoidal variation. Default value is “0.05” (metres).
- `BobPeriod`: minimum duration for single cycle of the bobbing motion, irrespective of speed. Default value is “0.4” (seconds).
- `BobStride`: distance over which a single bobbing motion will occur at slower speeds, equivalent to a virtual stride. Default value is “1.0” (metres).
- `BobHoldTime`: time for which non-zero height offset will be maintained when the user has stopped moving (avoid an immediate jump back to reference height among a sequence of small movements). Default value is “1.0” (seconds).
- `BobDropRate`: maximum rate at which offset will return to zero after holding time. Default value is “0.05” (metres per second).

Others:

- `MoveRateS`: the time, in seconds, between movement updates being sent to remote processes. This parameter is over-riden by the environment variable `MOVE_RATE_S`, if set. The default value is normally “0.5”, although this may be different for different client programs.
- `SubjectiveGrid`: show a 1 metre grid around the user, which is only visible to them. May help in determining distance and detailed movement. Default value is “false”. Can be toggled using key F4.
- `TerrainFollow`: determine whether user normally follows any specified terrain (ground), or just floats. Default is true. Terrain is an entity which has an attribute as a child with name “IS_GROUND”. Any

geometry which is also a child of the entity is treated as defining a ground level on which the user should normally travel.

- FlyingHeight: the up and down arrows move the user to and from a fixed flying height, specified by this parameter. The default value is “5.0”. This ability is subject to parameter UserControlledFlying.
- UserControlledFlying: determines whether the user is allowed to fly, as in FlyingHeight, above. Default is true.
- AvoidSolids: prevent user passing through solid objects. Default is true. Can be toggled using key F9.
- SolidDistance: minimum distance that user is allowed to approach a solid object (if AvoidSolids is true). Default is “0.4” (metres).
- RestrictedControls: flag disabling certain keyboard operations, e.g F1-F4, F8-F9. Default value is “false”. Used for restricted capability clients (e.g. avoid “cheating”).

Object Manipulation

- PickYOffset: when a user picks up an object it is moved upwards by this distance as a visual feedback device. The default value is “0.0” (i.e. no feedback).
- MouseDwellFlag: attempt to obtain ownership on object(s) that mouse pauses over, in anticipation of manipulation. Default value is true.
- ControlledUpdateOnly: force all object manipulation to use update requests (no ownership transfer). Default is false.
- ControlledFirstUpdate: make first object manipulation request use an update request, immediately followed by an ownership request, as in the CIAO system. Default is false.
- UseDropIcon: does the user have to select the drop icon to drop a held object, or will any mouse press of the relevant button drop it. Default value is true (press icon to drop).

Immersive Tracker

The immersiveClient supports a Fastrak 3D tracker. It is configured as follows:

- FastrakDevice: the device which is the serial line to which the fastrak is connected. The default value is “/dev/ttyd1”. At present, on Windows, only “COM1:” is tried and this cannot be over-riden.
- FastrakBaudrate: the baud rate to be used when accessing the fastrak. The default value is “115200”. Only certain values are allowed, typically: 115200, 57600, 38400, 19200, 9600, 4800, 2400, or 1200.
- FastrakOriginPos: an offset (Point3D) applied to all returned sensor values. The default value is “0,0,0”.
- FastrakOriginPos/<IP-address>: if specified, this property over-rides FastrakOriginPos for the specified machine, only. The <IP-address> must be in dotted decimal notation (e.g. “127.0.0.1”), and is default IP address of the machine (or the value of the environment variable LOCALHOST, if it is defined).
- FastrakOriginRot: a rotation (Quaternion3D) applied to all returned sensor orientations. The default value is no rotation (“0,0,0”).
- FastrakOriginRot/<IP-address>: if specified, this overrides FastrakOriginRot for the specified machine.
- FastrakNumUnits: the number of sensors configured for the given fastrak device. The default value is “0”. Over-riden by “FastrakNumUnits/<IP-address>”.
- FastrakUnit0Pos: an offset applied to the position returned by the first sensor. This offset is in local coordinates for the sensor, i.e. it will be transformed according to the orientation of the sensor before being returned. This can be used to compensate for the sensor not being in the place which is nominally being tracked (e.g. the base of the skull). The default value is “0,0,0”. Over-riden by “FastrakUnit0Pos/<IP-address>”.
- FastrakUnit1Pos: as above for the second sensor. Also FastrakUnit1Pos/<IP-address>, FastrakUnit2Pos, FastrakUnit2Pos/<IP-address>, etc.
- FastrakUnit0Rot: a rotation applied to the orientation returned by the first sensor. This can be used to compensate for the sensor not being in the orientation which is nominally being tracked. The default value is “0,0,0” (no rotation). Over-riden by “FastrakUnit0Rot/<IP-address>”.
- FastrakUnit1Rot: as above for the second sensor. Also FastrakUnit1Rot/<IP-address>, FastrakUnit2Rot, FastrakUnit2Rot/<IP-address>, etc.
- FastrakReset: determines whether a reset signal is sent to the fastrak on initialisation. Does not seem to work with some fastraks. Default is false. Over-riden by “FastrakReset/<IP-address>”.
- VisorYOffset: offset of the visor relative to the body in the Y axis. Default value “0.0”.
- VisorZOffset: offset of the visor relative to the body in the Z axis. Default value “-1.0”.
- VisorXRotate: rotation of the visor about the X axis. Default value “-45.0” (degrees).
- FlySpeed: speed of movement when the user presses the left mouse button. Default value “2.0”.

- DivmouseDevice: the device which is the serial line to which the Division flying mouse is connected. The default value is NULL (i.e. unset), indicating that there is no division mouse available. A UNIX a typical value might be “/dev/ttyd0”, while on Windows a typical value might be “COM2:”.
- DivmouseUnit: which division flying mouse unit to use to control movement. Possibilities are “0” or “1”. The default value is “0”.
- DivmouseBaudrate: the baud rate to be used when accessing the division mouse (if any). The default value is “9600”. Only certain values are allowed, typically: 115200, 57600, 38400, 19200, 9600, 4800, 2400, or 1200.
- DivmouseFlyUpButton: identity of the button used to simulate the up arrow to fly up. A negative value disables this function. The default value is “2”. The identification of buttons is believed to be: top-left is button 3; top-middle is button 0; top-right is button 2; upper-trigger is button 1; and lower-trigger is button 4.
- DivmouseFlyDownButton: identity of the button used to simulate the down arrow to fly down. A negative value disables this function. The default value is “3”. The identification of buttons is as above.
- DivmouseFlyButton: identity of the button used to move forwards (also left mouse button). A negative value disables this function. The default value is “1”. The identification of buttons is as above.
- DivmouseFlyBackButton: identity of the button used to move backwards. A negative value disables this function. The default value is “0”. The identification of buttons is as above.
- DivmouseHandButton: identity of the button used to grasp objects (also right mouse button). A negative value disables this function. The default value is “4”. The identification of buttons is as above.
- DivmouseNavigateButton: identity of the button use for head-movement navigation. A negative value disables this function. The default value is “-1”. This option is not yet supported (30/5/00). The identification of buttons is as above.