MERCURY LISP INTERFACE

Andrew MacDonald
February 1993

This is preliminary documentation for an xlisp interface to Mercury.

Mercury is designed to run with a front end which communicates between the native environment of the virtual world and the Mercury Participant System software. This document describes an xlisp front end tailored for used with FERN II. As such, many of the commands depend on knowing entity identifiers, and all functions expect data to be in FERN standard formats.

Every Mercury function can be identified by the prefix hg-; Hg is the symbol for atomic mercury.

Currently, Mercury is mostly an information consumer. It produces only one set of information: values of sensor data, both raw and processed. Mercury sends this data back to the front end periodically, and the front end must read this data by calling hg-get-data. More frequent calls to this function will cause Mercury to send data to the front end more often, to a maximum rate of 30 updates a second.

Every function but hg-init and hg-shutdown rely on Mercury being initialized by a call to hg-init, but currently they do no checking to see that Mercury is indeed initialized. If these functions are called without Mercury having been started, results are unpredictable. Please initialize before running.

Functions are broken into three categories: commands, attribute settings, and data receivers. Commands are functions which tell Mercury to change participant or environment state and are prefixed by "hg-poke-" (some have other prefixes). Attribute settings operate specifically on attributes of FERN II entities and are prefixed by "hg-set-". Currently there is only one data receiver function, hg-get-data. (hg-add-processor also returns data, but is a special case, see below.)

All entity identifiers passed to Mercury are assumed to be valid. Mercury has no access to FERN II databases except through this lisp interface, so it cannot check whether things which claim to be entities actually are. Passing phony Eids may result in erratic behavior.

Most functions return T if they complete successfully, nil otherwise; exceptions are noted below.

These are the **commands**:

```
(hg-init <run-host>
      :display-host <display-host>
      :binary <binary>
      :sound-file <sound-file>
      :config-file <config-file>
      :participant <participant-number>)
```

Start and establish communications with the Mercury Participant Entity on <run-host>. The display is normally on the invoking host, but can be changed with the :display-host key. The binary is normally /home/mercury/mercury, but can be changed with the :binary key. Descriptions of what sound samples are available and which key triggers them is normally found in the file "/home/mercury/sound.data"; this filename can be changed with the :sound-file key. Mercury sensor configurations are described in the configurations file, which is selected with the :config-file key. See /home/mercury/standard.config on the iris for more detail. The :participant key selects which machine the sound renderer will run on; this should be set to 0 for the current configuration. The hg-init function can be called multiple times, but only the first call from any node has any effect.

Hg-init returns configuration data from Mercury. This consists of a vector of two elements, the raw sensors, and the sensor processors. The raw sensors are a vector of triples #(0 :TYPE <index>) where :TYPE is either :6D or :SWITCH, and index is a unique index into the raw sensors. #(0 :6D 0) is the position of the sensor source in the world. The sensor processors are a vector of triples #(:TYPE #(consumption counts) #(production counts)), where :TYPE is the type of processors (currently only :DISPLACER), and the consumption and production counts specify how many of a given sensor type are consumed and produced by this processor. Example of a return value:

```
  #( #( #(0 :6D 0) #(0 :6D 1) #(0 :6D 2) #(0 :SWITCH 0) #(0 :SWITCH 1))
     #( #(:DISPLACER #(2 0) #(1 0))))
```

(hg-shutdown)

    Close communications to and terminate current Mercury Human Interface. If called when there is no active Mercury, there is no effect. This is called automatically by the xlisp (exit) function.

These next three commands manipulate the **Movement Pyramid** of the sensor source. For a full explanation of Movement Pyramid concepts, see the file movement.pyramid.

(hg-poke-source-velocity <velocity>)

    Set the velocity of the participant's sensor source. Velocity is a vector triple #(dx dy dz) of speed in each of the x, y and z directions. Calling this function with a velocity of nil is equivalent to, but of higher performance than, calling with a velocity of #(0.0 0.0 0.0).

(hg-poke-source-origin <Eid>)

    Make the position of the entity indicated by the argument be the origin about which the sensor source moves. If the argument is nil, set the sensor source to move relative to the world origin.

(hg-poke-void-color <color>)

    Set the void color (i.e., the background color of the screen) to be the indicated red-green-blue triple #(r g b). All values should be normalized to [0.0 - 1.0].

(hg-poke-source-delta <6D>)

    Use the indicated point-quaternion <6D> to compute a new position and orientation for the sensor source by composing the position and orientation of the sensor source's current origin with <6D>.

(hg-poke-fogginess <density> <color>)

    If <density> is greater than 0.0, turn on fog with indicated density and color, setting the void color to be the fog color for image consistency. If <density> equals 0.0, turn off fog and restore original void color.

(hg-bind-entity <Eid> <sensor triple>)

Associate the sensor triple with the entity specified. Currently this only works for :6D sensors. Binding a :6D sensor to an entity causes the entity to be moved to the position specified by the sensor. Example: (hg-bind-entity 3 #(1 :6D 0)) would move entity 3 to the position indicated by the first output of the first sensor processor.

(hg-bind-head <Eid>)

Specify which entities position and orientation to use for moving the participant's head in the virtual world.

(hg-add-processor <processor-type> #(<sensor0> <sensor1> ... <sensor-n>))

Add a new processor of the given type to Mercury. The sensors specified are the ones to be used to feed the processor. This function returns a vector of new sensor triples which identify the sensor streams which have just been created. Example, assuming no other processors have been specified:

```
(hg-add-processor :DISPLACER #( #(0 :6D 0) #(0 :6D 1)))
```

returns:

```
#( #(1 :6D 0))
```

(hg-specify-packet #(<sensor0> <sensor1> ... <sensor-n>))

Tell Mercury what data to send to the front end. The data cooresponding to the sensors specified is sent back in that order. This function returns a packet, which is a lisp vector for holding the data when it is retrieved with hg-get-data (see below).

(hg-statistics <flag>)

If flag is t, turn on reporting of frames-per-second update rate. If flag is nil, turn off such reporting.

(hg-reset)

Currently does nothing.

(hg-calibrate)

    When this call is issued, Mercury immediately calibrates all the sensors. Specifically, it sets the current sensor orientations to be the zero-orientation, i.e., Mercury assumes all sensors are facing negative Z axis.

(hg-delete-entity <Eid>)

    Remove this entity and any pictures, sounds or other attributes of it from Mercury's environment.

(hg-poke-delta-limit <bounding box>)

    Constrain the source's position delta to remain bounded by two points, a minimum and a maximum. <bounding box> format is #(<minimum triple> <maximum triple>). <bounding box> value of nil removes the constraint.

(hg-poke-detach-radius <distance>)

    Set a distance beyond which the source ceases to follow the object to which it is currently origined. The effect of this is to reset the source's origin attribute to nil and the source's delta attribute to be current world position. Setting <distance> to nil disables detaching.

(hg-poke-aural-control <midi command>)

    Send a MIDI command to the sound renderer. A <midi command> is a vector of four integer values in the range [0, 127] which encode the standard MIDI protocol.

(hg-poke-aural-ambience <channel> <id> <value>)

    This command is provided for backward compatibility and should not be used.

These are the **attribute setters**. Each takes the entity id of the entity to work on and a value to set. If the value is the skull string ("%"), the attribute is considered deleted, and Mercury resets its internal value for the attribute to the default.

(hg-set-6d <Eid> <6D>)

    Default: #(#(0.0 0.0 0.0) #(1.0 #(0.0 0.0 0.0)))
    Move picture and sound of entity <Eid> to the position and
    orientation indicated by the point-quaternion <6D>.

(hg-set-origin <Eid> <Oid>)

    Default: nil
    Make the position and orientation of the entity <Eid> to be
    perpetually that of the entity <Oid>. If <Oid> is nil, cancel the
    current origin, if any.

(hg-set-6d-delta <Eid> <6D>)

    Default: #(#(0.0 0.0 0.0) #(1.0 #(0.0 0.0 0.0)))
    Offset the picture and sound of the entity from the object it is
    origined to (or the world origin of the entity's origin is nil) by
    the given position and orientation deltas. Note that having both
    a 6D-delta and a 6D may cause the entity to seem to jump around as
    Mercury renders the object first based on one position and then
    another. Please be careful to have only one position specifier
    active at a given time.

(hg-set-picture <Eid> <Dog Description>)

    Default: nil
    Give the entity <Eid> the picture indicated by the <Dog Description>.
    The picture description should be a string, either a pointer to a
    file on the Mercury Participant Entity host, or a direct object
    describer. Currently Mercury can only handle descriptions 1000
    bytes long or shorter.

(hg-set-color <Eid> <RGB>)

    Default: #(1.0 1.0 1.0)
    Set the color of the picture of the entity <Eid> to be <RGB>. The
    color should be a red-green-blue vector triple, with all values
    normalized to [0.0 - 1.0].

(hg-set-visible <Eid> <visible-flag>)

    Default: T
    If <visible-flag> is t, make the picture of the entity <Eid>
    visible. If <visible-flag> is nil, make the picture of the entity
    <Eid> invisible.

(hg-set-wireframe <Eid> <wireframe flag>)

    Default: nil
    If <wireframe flag> is t, the picture of entity <Eid> is rendered
    in lines. If <wireframe flag> is nil, the picture of entity <Eid>
    is rendered in polygons.

(hg-set-scale <Eid> <scale-triple>)

    Default: #(1.0 1.0 1.0)
    Change the size of the picture of the entity by multiplying each
    of the vertices of the object by the given factor in the <x y z>
    triple.

(hg-set-texture <Eid> <Texture Name> &optional <Alpha Name>)

    Default: nil
    To the picture of the entity <Eid> attach the texture indicated
    by <Texture Name>, with an optional transparency component
    indicated by <Alpha Name>. Both texture and alpha arguments
    must be strings naming files residing on the Mercury Participant
    Entity host.

(hg-set-tex-map <Eid> <Texture Mapping>)

    Default: #(#(0.0 0.0 0.0) #(1.0 0.0 0.0) #(0.0 0.0 1.0))
    Map any textures applied to the picture of entity <Eid> according
    to <Texture Mapping>. A texture map is a vector triple of three
    vector triples, the first of which is the origin point for the
    texture (relative to object origin), the second is the S vector
    direction, and the third is the T vector direction.

(hg-set-tex-scale <Eid> <Texture Scale>)

    Default: #(1.0 1.0)
    Scale any textures applied to the picture of entity <Eid> according
    to <Texture Scale>. A texture scale is a vector pair of floats, the
    first being the S scale, and the second is the T scale.

(hg-set-sound <Eid> <Sound Name>)

    Default: nil
    Give the entity <Eid> the sound named by <Sound Name>.

(hg-set-loudness <Eid> <Loudness>)

    Default: 1.0
    Set the loudness of the sound of the entity <Eid> to be <Loudness>.
    A loudness is a float in the range [0.0 - infinity). In practice,
    the upper limit is about 1.3, 1.0 being "normal" loudness (that at
    which the sound was sampled). Loudness of 0.0 turns the sound off.

(hg-set-audible <Eid> <audible flag>)

    Default: T
    If <audible flag> is t, the sound of entity <Eid> is made audible.
    If <audible flag> is nil, the sound of entity <Eid> is made inaudible.

(hg-set-source <Eid> <source number>)

    Default: nil
    Associate a sound source with this entity. Four sources are
    currently supported, labeled 0 through 3. Any sound whose key
    ties it to the given source will be heard to play from the place
    where the entity is.

(hg-set-doppler <Eid> <doppler flag>)

    Default: nil
    If <doppler flag> is t, all sounds played through the source
    associated with this entity will bend their pitch based on
    their velocity relative to the ears to produce doppler shifts.
    Set <doppler flag> to nil to turn off pitch bending.

(hg-set-rolloff <Eid> <rolloff>)

    Default: 0.65
    Set the distance attenuation exponent for the source associate
    with this entity. Larger numbers produce faster rolloff.

This is the **data receiver**:

(hg-get-data <packet>)

    Fill <packet> with new position and button data from the Mercury
    Participant System. This function should be called as often as
    possible for smooth operation, but Mercury has a built-in pacing
    mechanism which allows the front end to slow down considerably
    without being clogged with data.

    <packet> is the vector returned by hg-specify-packet.

    hg-get-data returns the packet if new data is successfully
    acquired. If there is a transmission error, or no new data is
    available, nil is returned.