THE FERN II/MERCURY VIRTUAL BODY

Max Minkoff
September 1992

(in other words, the FERN II _interface_ to Mercury)

This is a set of entities that know how to talk to mercury, and will hopefully provide all the functionality you'll need to talk to and from the virtual body. If this does not meet your needs, please let me know. This is a first version that I'm hoping will spark some discussion and lead to a virtual body that will provide all the functionality we'll need.

There are 4 basic entities included with the virtual body:

body:
This is the entity that is the communications hub between body parts and other entities. It makes setup and ambience calls to mercury. It only puts its name in its boundary.

head:
This is the entity that will be perceiving other world entities and communicating with mercury. It puts its name and its current position in its boundary.

hand:
This is the entity that loads any wand tools you'd like to use and manages communication with them. It also provides such happy information as button ups and button downs, as well as the current state of the buttons. It puts its name and its current position in its boundary.

source:
This entity represents the virtual position tracker source. For more information about the source, and how movement is accomplished, please refer to /home/mercury/movement.docs.

To use the virtual body, all you need to do is to make an entity with /home/mercury/body/merc-body.fent and tell it to enter whatever space you want it to participate in. After that, all you'll _need_ to do is just have your other entities put the proper attribute information in their boundaries. You'll also probably want to start up at least the "fly" tool.

A good example of how to use the virtual body is the metro demo. To run it, log on to frabjous, go to /home/mercury/demos/metro/ directory, and type "veos train-console.lsp". You may need to plug in and boresight the fastrack. You probably should get Andy or me to help you get things going the first time. I'll also be very happy to help you get your code running with the virtual body.

For more advanced interactions, you'll probably want to design your own wand tool. If you do, please talk to me and/or use /home/mercury/wand-tools/fly.fent on the suns as an example. The virtual body requires that you put any wand you make into this directory.

The body has the following methods defined:

Fern:

"enter"
-- takes an eid and makes the whole body enter it

Setup:

"load-tools"
-- takes a list of wand-tool names and makes the hand load them.

"set-reset-aural-ambience"
-- takes a triple and stores it for resetting aural ambience.

"set-reset-delta"
-- takes a 6D. Stores this 6D as the reset-delta. Reset-delta defaults to the origin.

"set-reset-limit"
-- takes a bounding box to be used as the reset-limit. Reset-limit defaults to
#(#(-1000.0 -1000.0 -1000.0) #(1000.0 1000.0 1000.0))

Resets:

"reset-source"
-- takes no arguments. resets the delta to either the reset delta or the origin, sets the origin to nil, and resets the delta limit if set.

"reset-aural-ambience"
-- resets aural ambience

Poke the source:

"set-delta"
-- takes a 6D and gives it to the source

"set-origin"
-- takes an eid or a nil and gives it to the source

```
     "ride"
          -- another name for set-origin

     "set-velocity"
          -- takes a triple and gives it to the source

     "set-limit"
          -- takes a bounding box and gives it to the source.  The limit
               restrains the source's delta.
```

Ambience:

```
     "set-void-color"
          -- takes a triple, sets void color

     "set-aural-ambience"
          -- takes a triple and issues a midi control command with those
               parameters
```

mercury specific:

```
     "merc-boresight"
          -- boresights the fastrack.  The fastrack must be boresighted every
               time you turn it on.

     "statistics"
          -- takes t/nil, turns on/off merc statistics
```

The head will perceive the following attributes:

Position (default):

```
     "6D"                    #(#(0.0 0.0 0.0) #(1.0 #(0.0 0.0 0.0)))
     * "origin"              nil
```

Visual:

```
     "picture-desc"          n/a
     "color"                 #(1.0 1.0 1.0)
     "visible"               t
     "scale"                 #(1.0 1.0 1.0)
     "wireframe"             nil
     "texture-desc"          n/a
     "tex-map"               #(#(0.0 0.0 0.0) #(1.0 0.0 0.0) #(0.0 0.0 1.0))
     "tex-scale"             #(1.0 1.0)
```

Aural:

"sound-desc"	n/a
"loudness"	1.0
"audible"	t
"sound-source"	nil
"doppler"	nil

Note that if you get an attribute instead of copying it, that attribute will be displayed at its default value.

* "origin" is an attribute that will allow entities to "attach" to each other. If you make an entity have origin attribute with a value of another entity's eid, mercury will move that first entity with the second. This let's you do things like ride the train and attach the wand tool to the wand handle cleanly. Again, see /home/mercury/movement.docs for more info.

A NEW VERSION OF THE MERCURY VIRTUAL BODY

There are many changes, though most of them are additions, so everything *should* be fine. This is a somewhat long list, but please take a look through the whole thing if you'll be using the body any time soon.

1) PICTURE-DESC: picture-desc will now tack on ".dog" to the end of your attribute value if it's not already there. I.e. if you say ("picture-desc" "cube") it will be sent to the renderer as "/home/veos/dogs/cube.dog". You can send the whole thing if you like also. Please note that currently it only supports "/" as the first character. Anything starting with "." or "~" will not work. Please let me know if you feel that this is a problem.

2) BODY PART 6Ds: All body parts now put out specific 6Ds in addition to the general ones. That means that the hand puts out both "6D" and "hand-6D", the head does "6D" and "head-6d" and the source does "6D" and "source-6D". So if you only want to see where the hand is but don't care about 6Ds in general, just perceive "hand-6D" instead of "6D".

3) WIREFRAMING: Wireframing now works. Again, ("wireframe" t) will make your object wireframe.

4) FOG: You can now send to the body "set-fogginess" with a single argument of the form (vector density rgb) where density is the density of the fog and rgb is a triple representing the color of the fog.

E.G. #(0.03 #(0.7 0.7 0.7)) will give a nice light-grey fog.

5) AMBIENCE ENTITY: The body will now perceive two attributes - "fogginess" and "void-color" and will do the right thing for each. This means that you can have an entity in your space that manages these values, and any body that you put in it will get those values. Of course individual values may still be set by sending "set-void-color" and "set-fogginess" to the body, but this gives you a different option. I'm hoping that this concept will evolve into something more, so please let me know what you think about this, and if you do more with it (or want more done with it).

6) BODY PARTS: FERN entity body parts are now matched full with Mercury body parts. This means that you can set a picture for the hand, head, or source and have it be managed properly. You can also origin an entity to a body part - for instance we'll want to origin the body entity to the head.

7) AURAL CONTROL: You can now send "send-aural-control" to the body with a quadruple representing a MIDI event and it will be put out on the MIDI network that the MIDIator is attached to.

8) ORIGINING:

A. SET:

Now when you send "set-origin" to the body to have it attach to another entity (like the train), it won't move from its current position (unless you move or the origin entity moves) Previously you'd jump to the entity). If you want to change your position to be near the new entity, send "set-delta" to the body. Alternatively, if you are using the teleport wand, just make an attribute of "jack-delta" whose value is the new delta you should get when you jack to that object. Also, you can make an attribute of "delta-limit" whose value is the bounding box around the origin entity you want to limit your flight to. For example, when you jack to the train in the metro demo, the train has an attribute of "jack-delta" with a value of

```
#(#(0.0 1.8 0.0) #(1.0 #(0.0 0.0 0.0)))
```

which is the level of the real source off the floor, so you end up riding the train at the correct height. Also, the train has an attribute of "delta-limit" with a value of

```
#(#(0.0 1.8 -2.5) #(0.0 1.8 2.5))
```

which means that your flight is limited to -2.5 to 2.5 along the z-axis, locked to 0 on the x axis and 1.8 (normal height) on the y-axis. Please note that 1.8 is the height of the _physical_source_ off the ground, not any attempt at normal human height. How tall you are won't make any difference here.

B. DETACH:

You can now send "set-detach-radius" to the body with a float to set the distance at which the body will detach from an object that it is origined to. In other words, if you are riding some object and fly a certain distance away, you'll probably want to detach from it. On objects like the train where you've set a flying limit, you probably won't want to do this, but if you want to stop riding an object by just flying away from it, this is for you! Alternatively, as above, you can put an attribute called "detach-radius" in the boundary of an entity that you are going to jack to, and the right thing will happen. Please refer to the teleport wand as an example.