OPEN INVENTOR CORNER

Fun With Draggers: Building an Interactive Track Light

by Paul Isaacs, Silicon Graphics

In the May/June issue of *Developer News*, we discussed *engines*, *animating nodes*, and the *field-to-field connections* provided with Open Inventor 2.0. In this issue we discuss *draggers*, Inventor objects which you can move by click-dragging with the mouse. In particular, we show how the combination of draggers and field-to-field connections make it easy to create scenes with built-in user interfaces.

This article is a case study. It presents a standard Inventor file of a room with two track lights. Each track light is built of moveable draggers that allow you to control the parameters of a **SpotLight**. The file may be read into any Open Inventor application such as ivview or SceneViewer.

Everything needed to create this file comes with the inventor_eoe subsystem of IRIX 5.2. The developer's option is not required. Many of the techniques used in this article are described more fully in *The Inventor Mentor*, published by Addison-Wesley (ISBN 0-201-62495-8).

An Interactive Track Light

Figure 1 shows the device we are going to build. The *plank* can be click-dragged left and right, carrying the cylindrical *socket* and conical *bulb* with it. The socket can be moved forward and back along the plank (the bulb will follow). Click on the bulb and you can rotate it to point the SpotLight in any direction.

In addition, two panels are mounted on the rear wall, one for each track light. The *dimmer* (a small cube which slides up and down) controls the intensity of the spotlight. Similarly, the *spreader* (a small cone to the right of the dimmer) changes the beam spread.

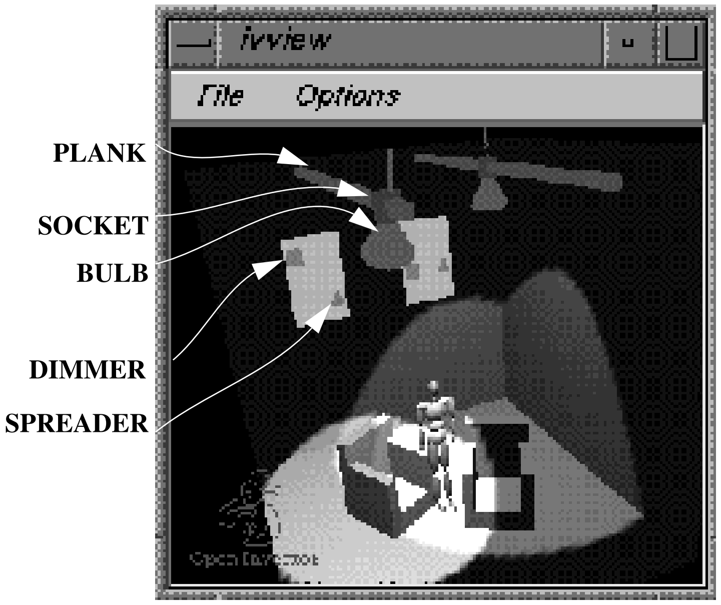

FIGURE 1. **Two Track Lights In a Room**

What's a Dragger Do?

Draggers are nodes you click and drag with the mouse. Each class of dragger has a shape you can pick, a unique way of moving, and a field (or fields) that reflects that motion. To build the track light, only two classes of dragger are needed, **Translate1-Dragger** and **RotateSphericalDragger**.

Translate1Dragger translates in a line along the X direction of its local space. It has a *translation* field that is constantly updated whenever you move the dragger. **Rotate-SphericalDragger** rotates freely in all directions. Its *rotation* field always reflects its current orientation.

Beauty Makeovers -- Give Your Dragger A New Look!

It's easy to change the shape of a dragger. That's why four different looking parts of the track light (plank, socket, dimmer, and spreader) can all be **Translate1Draggers**.

All draggers are *nodekits*. To change a part of a nodekit in an Inventor file, type the part name followed by the scene graph you want to use. To change the shape of a **Translat-e1Dragger**, you need to change two parts, "*translator*", and "*translatorActive*." The first part, "translator", is the geometry you select to translate the dragger. The second, “translatorActive”, is the geometry displayed while the dragger is in motion. Compare the following two files. If you have IRIX 5.2 installed on your system, just type:

```
ivview -q filename
```

and take a look. (If you leave out the -q option, the draggers will not work). The first file shows a regular **Translate1Dragger**. It looks like a double-headed arrow.

```
#Inventor V2.0 ascii
#This is a default dragger
Translate1Dragger { }
```

Notice that it turns yellow when you move it. This is because different scene graphs are used for the “translator” and “translatorActive.” The file below describes the plank of our track light. The plank does not change color because we use identical scene graphs for the two parts. The scene graph defined as PLANK_GEOM (a Separator with a cyan stick inside it) appear on screen as the new shape, which we can click and drag in the X direction. The plank begins at a height of 20 units because the translation field begins at (0,20,0). These elevates the plank relative to the floor of the room.

```
 #Inventor V2.0 ascii
DEF PLANK_DRAGGER Translate1Dragger {
    translation 0 20 0
    translator DEF PLANK_GEOM Separator {
        Material { emissiveColor .1 .3 .3 }
        Cube { width .5 height .5 depth 20 }
    }
    translatorActive USE PLANK_GEOM
}
```

At the conclusion of this article, the entire file TrackLight.iv is listed. In this file each dragger has its own customized geometry. Elsewhere in this article, for the sake of brevity, we will abbreviate the customization of geometry as follows:

```
#Inventor V2.0 ascii
DEF PLANK_DRAGGER Translate1Dragger {
    translation 0 20 0
    # custom geometry - see TrackLight.iv
}
```

Follow Me! -- Dragger/Transform Pairs Allow You To Move Other Objects

Draggers do not normally affect other nodes in a scene. They just scoot around where you push them. But if you connect the motion field into the corresponding field of a **Transform** node, this pair of nodes creates a powerful combination. By moving the dragger, you also edit the **Transform** node. This causes other shapes to follow the dragger. Figure 2 shows a dragger/transform pair that will translate the **Cube** whenever you use the dragger. Without the field-to-field connection, the cube would stay still even if you moved the dragger.

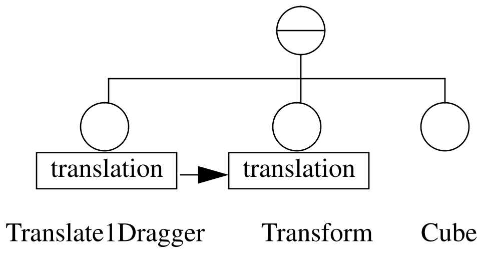

FIGURE 2. Connected Dragger/Transform Pair Moves the Cube Too

The following file shows how the plank's translation field is connected into a **Translation** node. Since the light's socket (also a dragger) follows it in the file, the socket moves along the X axis in unison with the plank. The **RotationXYZ** node rotates the socket perpendicular to the plank. This also means that the socket dragger's line of motion is rotated. So while the plank moves in the world's X direction, the socket moves in the world's Z direction. (Note: When you read the following file into ivview, the geometry will revert back to the default arrows since we've commented out the custom geometry. But the mechanism will still be correct).

```
#Inventor V2.0 ascii
# This dragger/Translation pair moves the SOCKET along with it.
DEF PLANK_DRAGGER Translate1Dragger {
    translation 0 20 0
    # custom geometry -- see TrackLight.iv
}
Translation { translation = USE PLANK_DRAGGER . translation }

# This rotates the SOCKET so it drags along the world's Z axis
RotationXYZ { axis Y angle 1.57079 }

DEF SOCKET_DRAGGER Translate1Dragger {
    # custom geometry -- see TrackLight.iv
}
```

The entire assembly of the track light's plank, socket, and bulb and **SpotLight** is built using these dragger/transform pairs. In TrackLight.iv, the socket dragger is paired with a **Translation** that serves to move the bulb with it. The bulb in turn is paired with a **Rotation** node that rotates the **SpotLight** itself.

TransformSeparator -- Moving the Light Without Moving the Room

When we put a **SpotLight** at the end of the "chain" of draggers we've built, we complete an interface that serves to translate and rotate the **SpotLight** exactly where we want it.. However, we do not want these transformations to affect other objects in the scene. The natural choice might be to put the whole mechanism underneath a **Separator** node, but this causes a serious problem: the **Separator** blocks the **SpotLight** from affecting objects outside the **Separator**. The room will not be lit!

Fortunately, Open Inventor 2.0 has a special node that solves this problem. **Transform-Separator** pushes and pops only the transformation in the state, but no other properties. Hence, the following scene structure gives us what we need:

```
TransformSeparator {
    # plank/translation combo
    # socket/translation combo
    # bulb/rotation combo
    # SpotLight
}
Separator {
    # All the other stuff in the room.
}
```

Remote Control -- The Dimmer and the Spreader

The *intensity* and *cutOffAngle* fields of the **SpotLight** are controlled by draggers that are mounted on the wall of the room -- the *dimmer* and the *spreader*. Here is a case where we don't want a dragger/transform combination. We just want the draggers to stay on the wall and feed values into a **SpotLight** located elsewhere.

In this case, we need to extract a single float value (intensity or cutOffAngle) from a vector field (the translation of a **Translate1Dragger**). To do so, we need to employ a **DecomposeVec3f** engine. This will remove the X value from the translation of the dragger and feed it into the desired **SpotLight** field. Here is how this looks in file:

```
#Inventor V2.0 ascii
SpotLight {
    intensity = DecomposeVec3f {
        vector = DEF INTENSITY_DIMMER Translate1Dragger {
            translation .5 0 0
            # custom geometry - see TrackLight.iv
        } . translation
    } . x
    cutOffAngle = DecomposeVec3f {
        vector = DEF ANGLE_DIMMER Translate1Dragger {
            translation .5 0 0
            # custom geometry - see TrackLight.iv
        } . translation
    } . x
}
```

Each of the draggers begins with a translation of (.5,0,0) so that the SpotLight will have an intensity and cutoffAngle of .5. The draggers are defined in-line, so if you try to render the above file, you'll get a SpotLight but no draggers will show up. If you look in TrackLight.iv you will see that later in the file, after the light has been set up, the remote control panel is created. Within this panel, the commands USE INTENSITY_DIMMER and USE ANGLE_DIMMER display these draggers in their proper locations.

The Files

Here are the complete files used to create Figure 1. If you run

```
ivview -q AllRoom.iv
```

you will be able to play with the interface described herein.

AllRoom.iv
Room.iv
TrackLight.iv

(see the .iv files in this directory)