# 4

# DLP and Virtual Worlds

## 4.1 VRML EAI AND DLP

VRML EAI stands for the external authoring interface of the virtual reality modeling language. The EAI allows developers to control the contents of a VRML world. A VRML specification can be loaded in a browser by an application, such as a Java applet. As mentioned, DLP programs are compiled to Java classes, which can be used as Java applets in Web Browsers. DLP has been extended with a VRML EAI library, called (bcilib). For instance, in the DLP VRML library, the predicate $getSFVec3f(Object, Field, X, Y, Z)$ gets the $SFVec3f$ value (which consists of three float numbers $X$, $Y$, and $Z$) of the $Field$ of object $Object$, and $setSFVec3f(Object, Field, X, Y, Z)$ assigns $X$, $Y$, and $Z$ values to the $SFVec3f$ $Field$ in $Object$, where $Object$ refers to an object in a 3D VRML world.

A DLP program can manipulate VRML virtual worlds by using the VRML EAI library predicates. Before we introduce more details of the DLP VRML predicates, we discuss first how to design VRML worlds that can be accessed from DLP programs.

## 4.2 DESIGN 3D VIRTUAL WORLDS FOR DLP

3D virtual worlds are implemented in VRML and the VRML EAI predicates refer to objects in the current virtual world which is loaded into a web browser. Each VRML file contains a scene graph hierarchy which consists of VRML

nodes. Node statements may contain SFNode or MFNode field statements that contain other node statements. Objects which can be manipulated by DLP programs are nodes which have names defined by DEF statements in VRML. We use DEF to define object names of VRML nodes, and use DLP VRML predicates to manipulate the values of the fields of the defined nodes.

For example, the following specification defines a yellow cylinder in VRML.

```
#VRML V2.0 utf8

Transform {
translation 0 0 0
rotation 0 1 0 0
children [Shape {appearance Appearance {
             material Material {diffuseColor 1.0 1.0 0.0}}
             geometry Cylinder { height 2.0 radius 1.0}}]}
```

In order to manipulate the values of the field 'translation' and the field 'rotation' of the cylinder, we have to define a name for the node 'Transform' as follows.

```
#VRML V2.0 utf8

DEF cylinder Transform {
translation 0 0 0
rotation 0 1 0 0
children [Shape {appearance Appearance {
             material Material {diffuseColor 1.0 1.0 0.0}}
             geometry Cylinder { height 2.0 radius 1.0}}]}
```

Note that we do not define the name of fields. More generally, we can define a prototype of a partial hierarchy first, so we can use such a prototype to create multiple instances of 3D objects. For example, the specification of a bus whose position and orientation can be controlled by DLP programs, consists of the following three steps:

1. Design a prototype for a bus;
2. Instantiate the bus prototype;
3. Use DEF to define the name of the bus.

The corresponding VRML description is as follows:

```
PROTO Bus [
      exposedField SFVec3f translation 0 0 0
      exposedField SFRotation rotation 0 1 0 0]
```

```
{
Transform {
      translation IS translation
      rotation IS rotation
      children [
         ......
      ]}}

Transform {children [ DEF bus1 Bus {
                        translation -5 0 -1.5
                        rotation 0 1 0 0} ] }
```

Thus, $setSFVec3f(bus1, translation, 15, 0, -1.5)$ sets $bus1$ to a new position $\langle -15, 0, -1.5\rangle$. Similarly, we can use the same method to manipulate a viewpoint in a VRML world, namely, we define a viewpoint first and then use the viewpoint predicates

$$getViewpointPosition(Viewpoint, X, Y, Z)$$

and

$$setViewpointPosition(Viewpoint, X, Y, Z)$$

to set and get the values of a viewpoint that's defined as:

```
DEF myviewpoint Viewpoint { position -10 1.75 0
                            orientation 0 1 0 -1.5708
                            set_bind TRUE}
```

Since VRML allows multiple viewpoints in a virtual world, we can only get the field values of the initial viewpoint unless we use explicitly the corresponding set-predicates. In the example above, calling $getViewpointPosition(myviewpoint, X, Y, Z)$, gets the result $X = -10.0$, $Y = 1.75$, and $Z = 0.0$ no matter how the user changes the viewpoint in the virtual world. In most applications, we want to get the current position of the user's viewpoint which may be changed by using the keyboard or mouse for navigation. This can be done by adding a proximity sensor to the virtual world:

```
DEF proxSensor ProximitySensor  {center 0 0 0
                                 size 1000 1000 1000
                                 enabled TRUE
                                 isActive TRUE}
```

Here we specify a proximity sensor with the size $(1000 \times 1000 \times 1000)$. Of course, the parameters should be changed for different applications. Getting the position and the rotation of the proximity sensor means getting the current values of the user's viewpoint. Therefore, we can define the extended get-viewpoint predicates as:

```
getViewpointPositionEx(_,X,Y,Z) :-
            getSFVec3f(proxSensor,position,X,Y,Z).

getViewpointOrientationEx(_,X,Y,Z,R):-
            getSFRotation(proxSensor,orientation,X,Y,Z,R).
```

## 4.3 LOADING 3D VIRTUAL WORLDS

3D virtual worlds have to be loaded in a web browser for program manipulation, which can be done as follows:

1. Virtual worlds embedded in html files:

   ```
   <html>
   <title> DLP-BCI example 1</title>
   <body bgcolor=white>
   <embed src="vrml/root.wrl" width="100%" height="80%">
   <applet codebase="classes/" archive="dlpsys.jar"
   code="dlpbrow.class" width=600 height=300 MAYSCRIPT>
   <param name="mayscript" value="true">
   <param name="cols" value="60">
   <param name="rows" value="10">
   <param name="object" value="example1">
   </applet>

   </body>
   </html>
   ```

   Where *root.wrl* is the initial VRML file which is stored in the directory "./*vrml*", DLP classes directory is "./*classes*", archive="dlpsys.jar" means that the DLP system library is "dlpsys.jar", code="dlpbrow.class" says that DLP is initialized by the class 'dlpbrow', which creates a text area in the browser that serves as a message output window for DLP. The statements $<$ *param name* = "*cols*" *value* = "60" $>$ and $<$ *param name* = "*rows*" *value* = "10" $>$ define the columns and rows of the text area. If 'dlpbrow.class' is replaced by another class 'dlpcons.class', this message window does not appear in the browser and all messages will be forwarded to the browser's built-in Java console. MAYSCRIPT states that java scripts are enabled. $<$ *param name* = "*object*" *value* = "*example*1" $>$ states that the DLP program which manipulates the virtual worlds is "example1".

2. Load virtual worlds for manipulation, by using the DLP VRML predicate $loadURL(URL)$. For example, $loadURL($"$example1.wrl$"$)$ loads the virtual world "$example1.wrl$" into the web browser.

## 4.4 VRML PREDICATES

We call VRML predicates which can be used for getting values *get-predicates*, and predicates for setting values *set-predicates*. The DLP VRML library (bcilib) offers a complete collection of get/set-predicates for all field types in VRML. Here are some VRML predicates from the DLP VRML EAI library:

- Single Field Predicates
  - $getSFBool(+Object, +Field, -Bool)$
  - $setSFBool(+Object, +Field, +Bool)$
  - $getSFFloat(+Object, +Field, -Float)$
  - $setSFFloat(+Object, +Field, +Float)$
  - $getSFInt32(+Object, +Field, -Int32)$
  - $setSFInt32(+Object, +Field, +Int32)$
  - $getSFString(+Object, +Field, -Atom)$
  - $setSFString(+Object, +Field, +Atom)$
  - $getSFTime(+Object, +Field, -Time)$
  - $setSFTime(+Object, +Field, +Time)$
  - $getSFColor(+Object, +Field, -R, -G, -B)$
  - $setSFColor(+Object, +Field, +R, +G, +B)$
  - $getSFVec2f(+Object, +Field, -X, -Y)$
  - $setSFVec2f(+Object, +Field, +X, +Y)$
  - $getSFVec3f(+Object, +Field, -X, -Y, -Z)$
  - $setSFVec3f(+Object, +Field, +X, +Y, +Z)$

  For instance, $setSFVec3f(OtherViewpointNode, position, X, Y, Z)$ sets the position of another viewpoint node with values $X, Y, Z$, and $getSFVec3f(OtherViewpointNode, position, X, Y, Z)$ returns the position of the other viewpoint node in the variables $X, Y$, and $Z$ respectively.

- Multi Field Predicates
  - $getMFFloat(+Object, +Field, -FloatList)$
  - $setMFFloat(+Object, +Field, +FloatList)$
  - $getMFInt32(+Object, +Field, -IntegerList)$
  - $setMFInt32(+Object, +Field, +IntegerList)$
  - $getMFString(+Object, +Field, -AtomList)$
  - $setMFString(+Object, +Field, +AtomList)$

  - $getMFColor(+Object, +Field, -RGBList)$
  - $setMFColor(+Object, +Field, +RGBList)$
    where $RGBList = [[R1, G1, B1], [R2, G2, B2], ....]$
  - $getMFVec2f(+Object, +Field, -XYList)$
  - $setMFVec2f(+Object, +Field, +XYList)$
  - $getMFVec3f(+Object, +Field, -XYZList)$
  - $setMFVec3f(+Object, +Field, +XYZList)$
    $XYList = [[X1, Y1], [X2, Y2], ....]$
    $XYZList = [[X1, Y1, Z1], [X2, Y2, Z2], ....]$

- Agent / Object Coordinates
  - $getPosition(+Object, -X, -Y, -Z)$
  - $setPosition(+Object, +X, +Y, +Z)$
  - $getRotation(+Object, -X, -Y, -Z, -R)$
  - $setRotation(+Object, +X, +Y, +Z, +R)$
  - $getViewpointPosition(+Viewpoint, -X, -Y, -Z)$
  - $setViewpointPosition(+Viewpoint, +X, +Y, +Z)$
  - $getViewpointOrientation(+Viewpoint, -X, -Y, -Z, -R)$
  - $setViewpointOrientation(+Viewpoint, +X, +Y, +Z, +R)$

In the VRML EAI library, the viewpoint predicates above manipulate a node $'Viewpoint'$, the default name of the viewpoint. However, in order to manipulate other viewpoints with non-default names, we can use generic predicates, like

$$getSFVec3f(Viewpoint, position, X, Y, Z)$$

and

$$setSFVec3f(Viewpoint, position, X, Y, Z).$$

Note that DLP programs which want to use VRML predicates to manipulate 3D objects should make the DLP VRML library *bcilib* available for it. This can be done by inheritance. Namely, the first line of DLP programs should be like:

```
:-object objectname : [bcilib].
```

## 4.5 MANIPULATING VRML WORLDS: EXAMPLES

In this section, we discuss how we can use DLP VRML predicates to control the contents of virtual worlds by a number of examples.

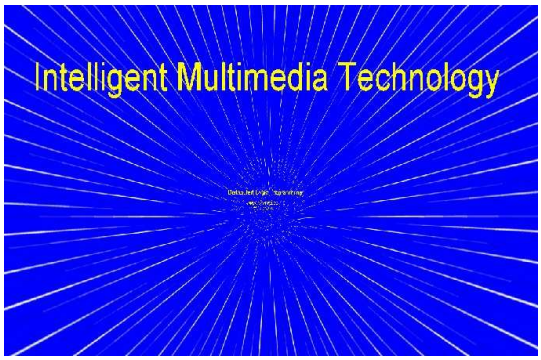


*Fig. 4.1* Screenshot of title moving

### 4.5.1 title moving

In this subsection, we discuss several examples to achieve the effect of title moving, like those at the beginning of TV programs and movies. A screenshot is shown in Figure 4.5.1.

First we use VRML to design a virtual world in which there is a tunnel, i.e., a big cylinder with no bottom, and with the texture 'star1.jpg' as its background to simulate a star sky. Moreover, we also specify several texts which are located at the tunnel, and a music file 'mozart38.wav' to add sound. The 3D virtual world can be specified as follows in a file 'title0.wrl'.

```
#VRML V2.0 utf8

Background {skyColor [0.0 0.0 1.0] groundColor [0.0 0.0 1.0]}

DEF myViewpoint Viewpoint { position 0 3 6 orientation 0 1 0 0}

Transform { translation 0 3 0 rotation 1 0 0  1.57
      children [Transform {translation 0 0 0
      rotation 0 1 0 0
      children  Shape {appearance Appearance {
        texture ImageTexture { url "star1.jpg" repeatS TRUE repeatT TRUE }
        textureTransform TextureTransform {scale 30 10}
        material Material {diffuseColor 0.0 0.0 1.0
             emissiveColor 1.0 1.0 1.0}}
        geometry Cylinder { height 2000 radius 2.5   top TRUE bottom FALSE}}
        }]}

Transform {translation -2 3 50 scale 0.3 0.3 0.3
children [Shape {appearance Appearance {
```

```
         material Material {diffuseColor 1 1 0
         emissiveColor 1 0 0}}}
       Text { string "Intelligent Multimedia Technology" }] }


Transform {translation -2 3 30 scale 0.3 0.3 0.3
children [Shape {appearance Appearance {
          material Material {diffuseColor 1 1 0
          emissiveColor 1 0 0}}}
       Text { string "Distributed Logic Programming" }] }


Transform {translation -2 3 10 scale 0.3 0.3 0.3
children [Shape {appearance Appearance {
            material Material {diffuseColor 1 1 0
            emissiveColor 1 0 0}}}
       Text { string "VRML+JAVA+PROLOG" }] }


Transform {translation -2 3 -10 scale 0.3 0.3 0.3
children [Shape {appearance Appearance {
            material Material {diffuseColor 1 1 0
                emissiveColor 1 0 0}}}
       Text { string "Multimedia Authoring" }] }


Transform {translation -2 3 -30 scale 0.3 0.3 0.3
children [Shape {appearance Appearance {
            material Material {diffuseColor 1 1 0
              emissiveColor 1 0 0}}}
       Text { string "Thank You Very Much!" }] }




Sound {maxBack 2000 maxFront 2000
minFront 1000 minBack 1000
 intensity 20 source AudioClip { loop TRUE
url ["mozart38.wav"]}}
```

In order to achieve an animation effect, we design a DLP program so that the viewpoint can be changed regularly. Thus, we add a defined name 'myViewpoint' to define the viewpoint in the virtual world. The DLP program 'titlemove0.pl' is shown below:

```
:-object titlemove0: [bcilib].

var count = 3000.
var increment = 0.15.
var url='./title/title0.wrl'.


main :- text_area(Browser),
        set_output(Browser),
loadURL(url),
        sleep(3000),
        move_title(count).

move_title(0):-!.

move_title(N):- N>0,
        N1 is N-1,
        getSFVec3f(myViewpoint,position, X,Y,Z),
        Znew is Z - increment,
        setSFVec3f(myViewpoint, position,X,Y,Znew),
        sleep(150),
        move_title(N1).

:-end_object titlemove0.
```

In order to let the format function in DLP programs to send its output to the web browser, we use the following clauses to set a text area as the output of the program:

```
 text_area(Browser),
 set_output(Browser),
```

However, note that if we set code="dlpcons.class" in the html file, the message window is not enabled in the browser, and the two lines above should not be used in the program.

In the program, the viewpoint's position is gradually moving to the negative Z-direction by decreasing the Z value 0.15 meter each time. This is a simple example that shows how to manipulate 3D objects to achieve animation in virtual worlds.

### 4.5.2 Bus Driving

In this example, we design a bus driving program. Assume we have designed a bus in a VRML world, whose url is:

```
./street1.wrl
```