A Self-Training Tool for Learning 3D Geometrical Transformations

José Ribelles[1], **Ángeles López**[2]
[1]Departamento de Lenguajes y Sistemas Informáticos
[2]Departamento de Ingeniería y Ciencia de los Computadores
Universitat Jaume I
[1]ribelles@lsi.uji.es [2]lopeza@icc.uji.es

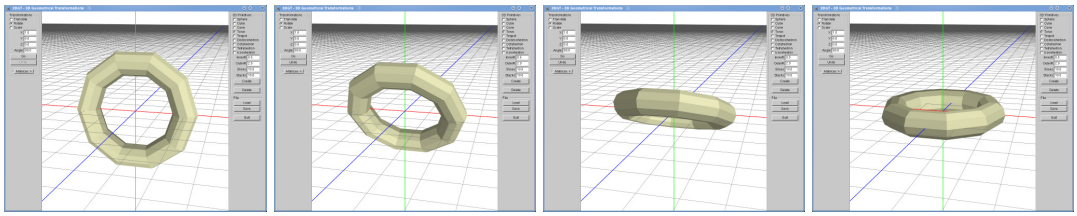


Figure 1: Four frames of the animation corresponding to a 90 degree rotation of a torus around the X axis.

Abstract: *This paper presents a self-training tool for learning 3D geometric transformations, i.e. translation, scaling and rotation. Our aim is to provide students with a tool they can use to practise these transformations by themselves so that they can understand and learn how to use them. Its main feature is to show an animation of the geometrical transformation applied on a geometric primitive. This animation helps to comprehend the transformation more easily, as the student sees how the primitive changes its size, position and orientation. Furthermore, the teacher can also use it as a teaching aid to explain and solve exercises in class. The geometric primitives as well as the transformation parameters used in this tool are those provided by the GLUT library and the OpenGL standard. Moreover, one of the results the tool can also provide is the C source code of the scene for OpenGL programming, which is very useful if the student needs to learn to program OpenGL transformations. Finally, this tool has been made available to our students for three academic years, and they found it very useful.*

Keywords: geometrical transformation, geometric objects, OpenGL transformation.

1 Introduction

Three-dimensional geometrical transformations are a central topic in the contents of any basic subject about computer graphics. The use of these transformations allows us to locate, orientate and change the size of objects in 3D space. First of all, both the teacher (using the blackboard or slides) and the student (with a sheet of paper) use 2D material to learn this 3D topic. Therefore, the ability of the student to represent different 2D views of 3D objects and to obtain the result of the transformations applied to these objects largely determines the effort that will be required to learn.

Secondly, in order to validate the result of a problem, the student often programs the sequence of OpenGL transformations and obtains different views of the result. However, this mechanism only shows the final result, which increments the student's frustration should they be faced with an unexpected outcome. Finally, students often perform a "trial and error" process: they change values randomly until a more or less satisfactory result is obtained.

The third motivation to develop a specific tool is that, during several years of teaching, we have observed that students had trouble in understanding a series of issues, such as the following:

- How a primitive centred on the origin is affected by a negative scaling.
- How a primitive is affected by a symmetry transformation respect to a plane.
- Why a scaling operation can produce a translation of the object.

- How to define the direction of rotation, that is, the sign of the angle.
- How to determine the appropriate rotation axis.
- How to determine the correct order in a sequence of transformations.
- How to determine the correct values for the parameters of transformations.
- Which OpenGL code is associated to a sequence of transformations.

In this paper we present a tool addressed to students, but which can also be useful for teachers:

- The student can train individually, for as much time as needed, until he or she acquires the ability to pose the problem and solve it on paper.
- The teacher can use this tool to illustrate the solution to a problem by using a video projector.

This paper is organised as follows. Section 2 describes the features considered for the development of such a tool. Section 3 describes the tool in detail, and section 4 shows some examples of use.

2 Features of the 3DGT tool

As the aim of this tool is to allow students to practise and understand the application of 3D geometrical transformations, its main characteristic resides in displaying an animation of the object during the transformation (see Figure 1). This characteristic can be described in more detail as follows:

- Each 3D geometrical transformation is displayed as a movement from the initial state of the object, until the final state.
- The point of view can be modified interactively, even during the animation.
- Each transformation can be undone, and this transformation is animated too.
- Each animation can be repeated as many times as desired.
- A sequence of transformations can be applied successively to any object.

Another important feature of this tool consists in the use of the OpenGL graphic standard. The user can thus learn how OpenGL performs the transformations:

- The geometric primitives used in the tool are those provided by the GLUT library. The parameters of these primitives are also those required by the GLUT library.
- The parameters of the transformations are those required by OpenGL.
- The tool provides the C source code of the final scene for OpenGL programming.

Finally, let us point out that this tool is not aimed to be a modelling tool. As the 3DGT tool does not provide complex operations typical of modelling tools, such as union, copy, etc., the user is dissuaded from using it for modelling complex scenes.

3 Description of the 3DGT tool

The interface of this application (see Figure 2) has three parts:

- In the middle, the 3D scene is displayed. The user can interact and change the point of view.
- On the right, there are two menus: a menu for creating and removing 3D primitives, and the file menu.
- On the left, the transformations menu, for doing and undoing transformations.

The 3D scene can be displayed through a variety of options. The user can use the mouse and the keyboard to interact with the scene:

- **Mouse Left** button for rotating the scene
- **Ctrl + mouse left** button for zooming
- **Shift + mouse left** button for panning

- **Ctrl + Shift + mouse left** button for selecting a primitive of the scene (which will be displayed in light yellow)
- **Mouse Right** button for a popup menu with the following options:
  - Polygon mode: the user can choose between fill or line
  - Shade model: the user can choose between smooth or flat
  - Show axis
  - Show floor
  - Background colour: the user can choose between blue, black and white

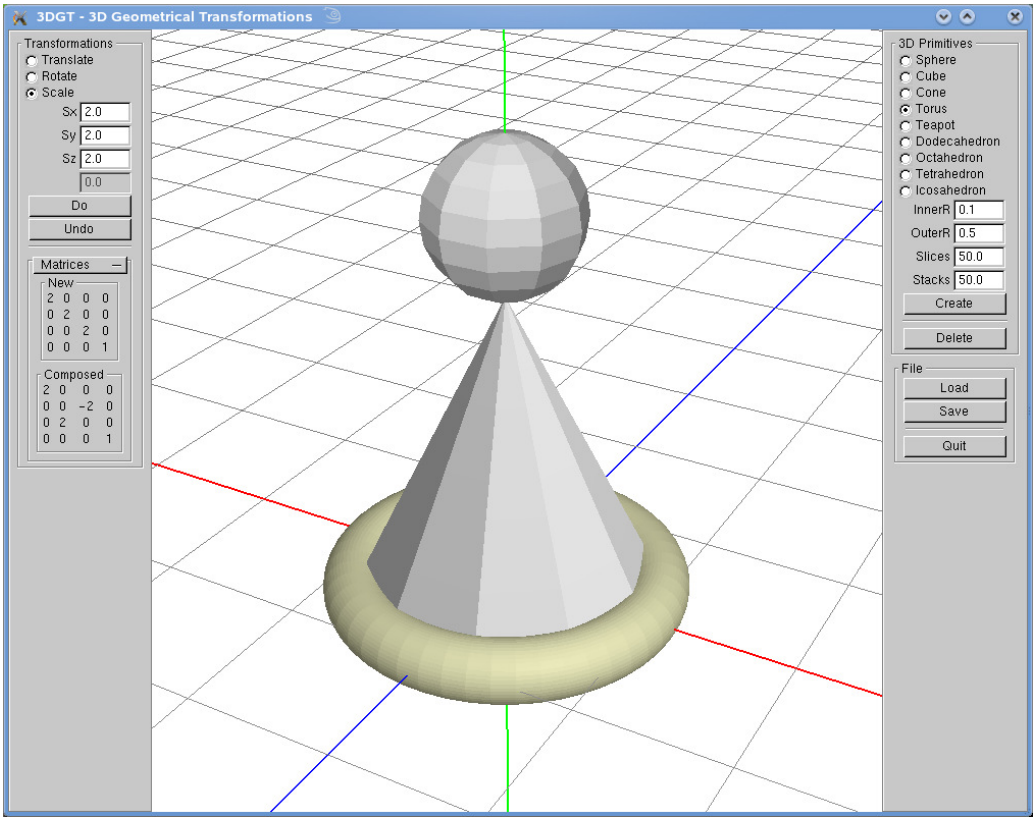


Figure 2. The interface of the 3DGT tool.

All the geometrical transformations, as well as undo and delete operations, are applied on the selected primitive (**Ctrl + Shift + mouse left** button). When a primitive is added to the scene it is automatically selected.

The menu for creating and removing 3D primitives consists of a primitive selector and buttons for creating and deleting the primitive (see Figure 3). To create a primitive, the user must choose the primitive type in the selector. Then, the primitive parameters appear below so that the user can introduce the desired values. Finally, the user must press the **Create** button. To delete a primitive, the user must select the primitive interactively in the scene and press the **Delete** button.

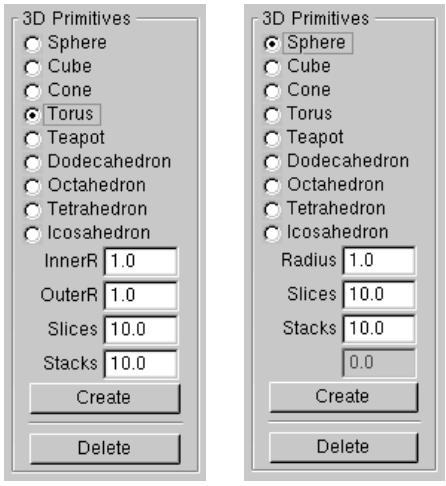


Figure 3. Left: When the torus is selected, the parameters required by GLUT are: inner and outer radius, slices and stacks; Right: when the sphere is selected, the parameters required by GLUT are: radius, slices and stacks.

Only the geometric primitives of the GLUT library are provided: sphere, cube, cone, torus, teapot, dodecahedron, octahedron, tetrahedron, icosahedron. The parameters of each primitive are those required by the GLUT library.

The transformation menu consists of a selector of the geometrical transformation: translation, scaling and rotation. The parameters of each transformation are those required by the OpenGL standard (see Figure 4). The transformation is performed when the user presses the **Do** button. The transformation is undone when the user presses the **Undo** button. Moreover, two matrices are shown under a rollout called **Matrices**. The first matrix is called **New** and it is the transformation matrix corresponding to the transformation selected by the user and its parameters. The second matrix is called **Composed**, and it is the composed transformation matrix for the selected object in the scene (see Figure 5).

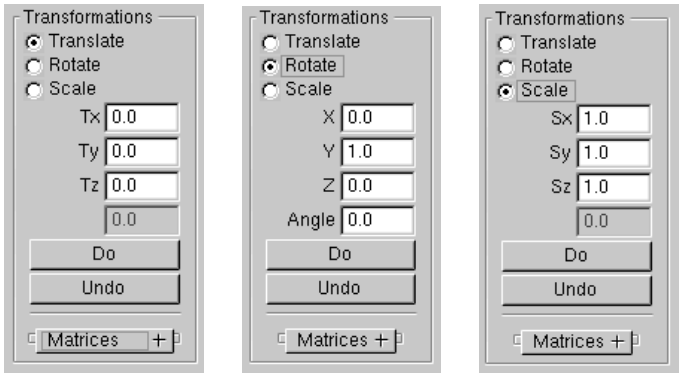


Figure 4. The transformation menu: translation (left), rotation (middle), scaling (right). The user can select a transformation and its parameters, which are those required by OpenGL.

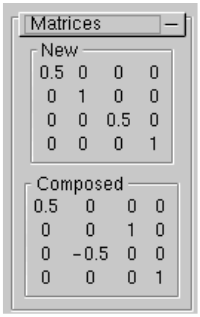


Figure 5. In this example, the user has selected a scale transformation, with Sx and Sz to 0.5 and Sy to 1.0, and the transformation matrix is shown in New. The Composed matrix shows the transformation matrix of the selected object that, in this example, has been first rotated -90 degrees around the X axis and scaled after as shown in the New matrix.

The **File** menu (see Figure 6) allows the scene that has been created to be saved and loaded. Two files are created when the **Save** button is pressed: "data.c" is the C source code of the scene; "data.save" is the file needed by 3DGT to load a scene. A **Quit** button to leave the application is also provided.

Figure 6. The File menu, which includes Load, Save and Quit options.

4 Examples

In the first example (see Figure 7a), the user created a cube and a sphere. Then, the user created two tori and finally rotated and moved both of them. In the second example (see Figure 7b), a larger number of primitives were created, and some of them underwent a transformation.

5 Methodology

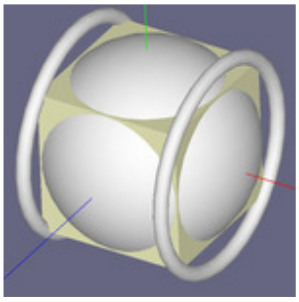
The classes are divided into three parts: lectures, exercises and laboratory.

For lectures, we use the slides provided by the author of the textbook [Angel, E., 2006] on its website, under the item "PPT Lectures". The file "AngelCG12.ppt" contains the basic concepts about transformations, and the file "AngelCG13.ppt" shows how to carry out transformations and how to manage matrix modes in OpenGL.

During the timetabled classes, students are given several exercises to solve on paper, so that they can practice the theoretical concepts through practical examples. Appendix A contains a list of such exercises. After the students have tried to solve each problem, the teacher provides the solution and uses the 3DGT tool to show them the correct sequence of transformations, as well as perhaps the undesirable effects of other wrong sequences of transformations. Some of these exercises can be solved by the students as homework.

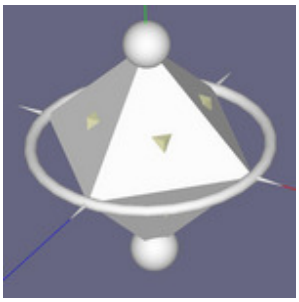
Finally, in the laboratory, two major problems are proposed (see Appendix B). For each problem, the students must write a *C* program that performs the correct OpenGL transformations to solve the problem. They can use the 3DGT tool to test the sequence of transformations and their parameters, as well as to obtain the *C* source code for OpenGL programming.

```
void scene(void) {

  glutSolidCube(1.000);
  glutSolidSphere(0.700,30,30);
  glPushMatrix();
    glTranslatef(-0.500,0.000,0.000);
    glRotatef (90.000,0.000,1.000,0.000);
    glutSolidTorus(0.050,0.700,10,30);
  glPopMatrix();
  glPushMatrix();
    glTranslatef(0.500,0.000,0.000);
    glRotatef (90.000,0.000,1.000,0.000);
    glutSolidTorus(0.050,0.700,10,30);
  glPopMatrix();
}
```



(a)

```
void scene(void) {

  glutSolidOctahedron();
  glPushMatrix();
    glTranslatef(0.000,1.000,0.000);
    glutSolidSphere(0.200,30,30);
  glPopMatrix();
  glPushMatrix();
    glTranslatef(0.000,-1.000,0.000);
    glutSolidSphere(0.200,30,30);
  glPopMatrix();
  glPushMatrix();
    glRotatef (90.000,1.000,0.000,0.000);
    glutSolidTorus(0.050,1.000,30,30);
  glPopMatrix();
  glutSolidCone(0.100,1.300,10,10);
  glPushMatrix();
    glRotatef (90.000,0.000,1.000,0.000);
    glutSolidCone(0.100,1.300,10,10);
  glPopMatrix();
  glPushMatrix();
    glRotatef (180.000,0.000,1.000,0.000);
    glutSolidCone(0.100,1.300,10,10);
  glPopMatrix();
  glPushMatrix();
    glRotatef (-90.000,0.000,1.000,0.000);
    glutSolidCone(0.100,1.300,10,10);
  glPopMatrix();
  glPushMatrix();
    glScalef (0.750,0.750,0.750);
    glutSolidCube(1.000);
  glPopMatrix();
}
```



(b)

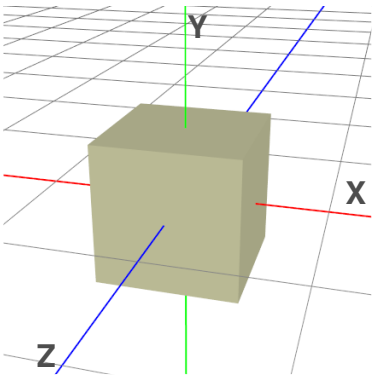
Figure 7. Examples created with the 3DGT tool.

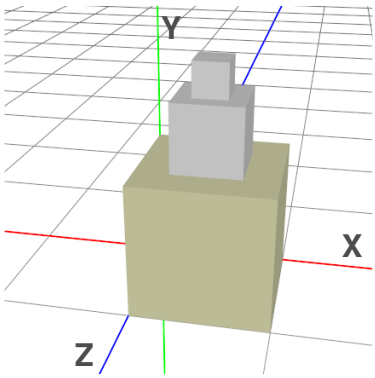
References

- Shreiner, D., Woo, M., Neider, J., Davis, T., 2006. OpenGL Programming Guide. Addison-Wesley. http://www.glprogramming.com/red
- The freeglut programming consortium 2003. The Open-Source OpenGL Utility Toolkit: Application Programming Interface. http://freeglut.sourceforge.net/docs/api.php
- Angel, E., 2006. Interactive Computer Graphics. Addison-Wesley. http://www.cs.unm.edu/~angel/BOOK/INTERACTIVE_COMPUTER_GRAPHICS/FOURTH_EDITION/

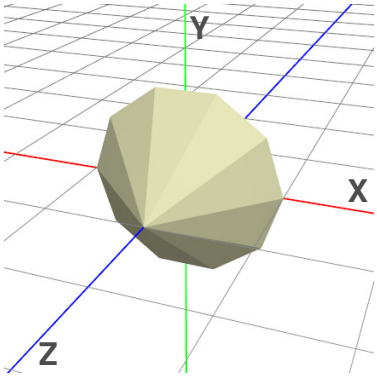
Appendix A

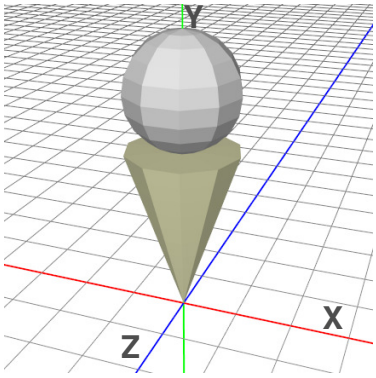
1. Start with a unit cube centred at the origin. Use two more cubes like this one and obtain the model shown in the right image, where each new cube is half the size of the previous one.



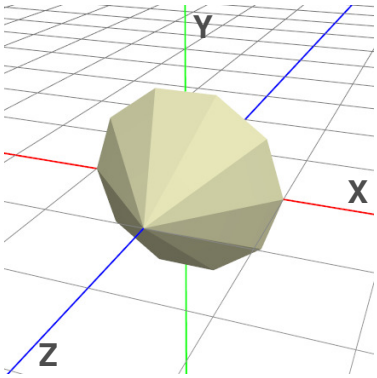


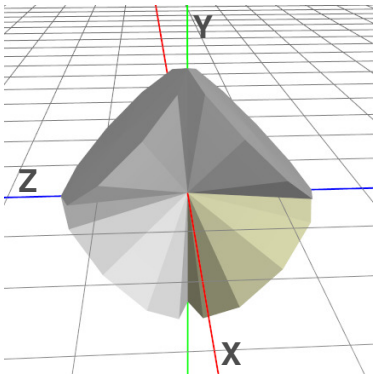
2. Start with a cone of base radius 1 and height 1, as shown in the left image. The right image shows the transformed cone of height 3 and a sphere of radius 1. Determine the transformations.



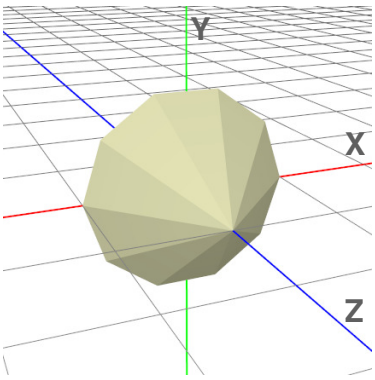


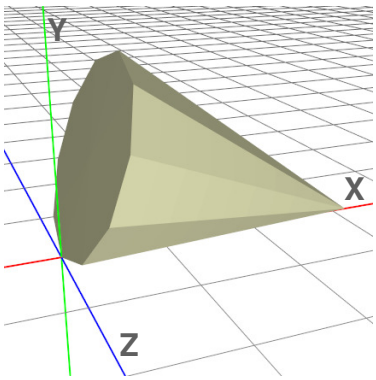
3. The left image shows a cone of base radius 1 and height 1. List the transformations that are necessary to convert the cone in the left image into each cone in the right image.



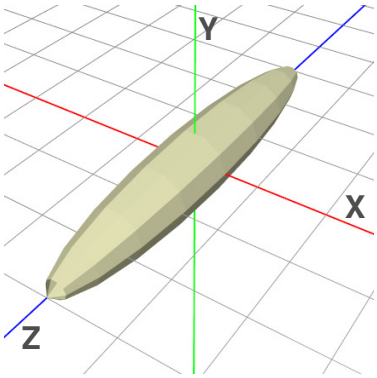


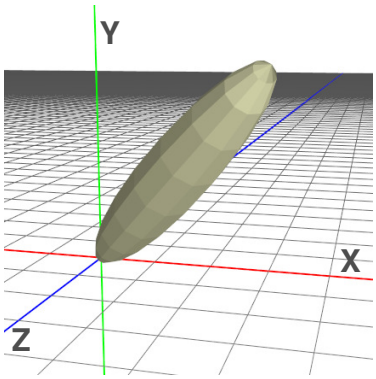
4. Determine the transformations that are necessary to situate the cone shown in the left image (base radius 1 and height 1) in the position of the one shown in the right image (base radius 1 and height 3).



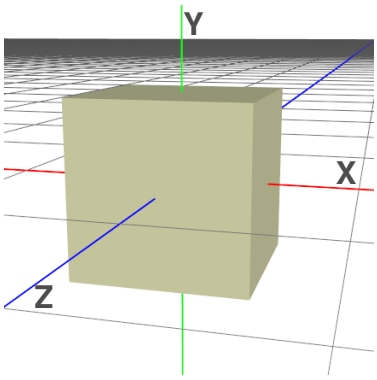


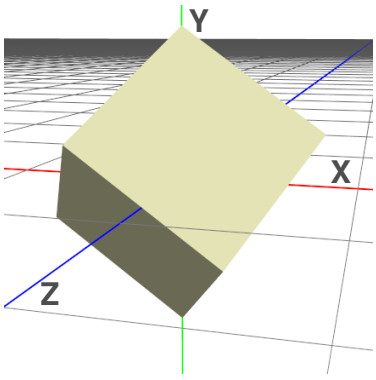
5. Start with a sphere of radius 1 centred at the origin. The left image shows the sphere transformed with factor scales *Sx* and *Sy* 0.5 and *Sz* 3. Determine the transformations that are necessary to situate the sphere as shown in the right image, where one end point is at the origin and the other belongs to the line $x=y=z$.





6. Start with a unit cube centred at the origin, as shown in the left image. The right image shows the desired orientation, where one of the diagonals of the cube is on the Y axis. Determine the transformations that are needed.

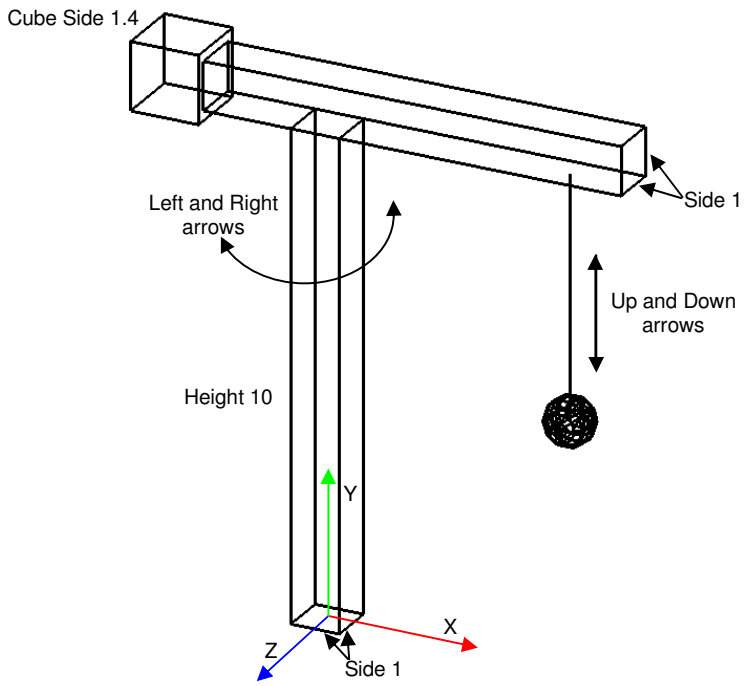




7. Start with a sphere of radius 1 centred at the origin. Move the sphere along the Z axis and then use rotations to situate the sphere at the point (3, 3, 3). In order to validate your solution, move another sphere directly to that point. It should coincide with the first one.

Appendix B

1. Write the OpenGL code to model the crane shown in the figure below. Start with unit cubes. Allow interaction using the keyboard as shown in the figure.



2. Write the OpenGL code to model the table lamp shown in the figure below. Choose your own measures. Allow interaction using the keyboard or the mouse as shown in the figure. Add the spotlight.