3D Shapes and Time 2

2.1 Time

The introduction of time dependent movements brings always a new dimension in the attractiveness of computer graphics. Frequently for the purpose of producing movement, programmers use counters that increase distances, angles, or other parameters. Despite that, the achieved result depends on how fast is the computer or how heavy are the remaining computations, as the increments may be done faster or slower and similarly the same happening with the represented movements (or evolutions).

To avoid this, it is important to use real time measures from the underlying operating system via some function available. Consider the use of the following pseudocode

```
oldTime = getTime();
...
newtime = getTime();
deltaT = newtime-oldtime;
oldtime = newtime;

newposition.x = SPEED*deltaT;
...
```

Now you can replace getTime() by some function provided by your preferred programming environment.

For example in GLUT and SDL you can use respectively the number of milliseconds that passed since the initialisation of these toolkits

```
// GLUT
int currentTime=glutGet(GLUT_ELAPSED_TIME);
// SDL
int currentTime = SDL_GetTicks(); /
```

2.2 Transformations and Animations

The transformations are at the basis of the construction and manipulation of object models. These models are typically created centred on the origin of their own coordinate frame and frequently choosing one of the axes as its medial axis. For example, in classical OpenGL we may use GLU library that provides, among other things, the support for the generation of polygon meshes for several basic shapes. Lets consider a cylinder, it can be seen as a circumference on the plane x-y and extruded[1] along the z-axis.

```
void gluCylinder( GLUquadric* quad,
        GLdouble base,/* radius */
        GLdouble top, /* radius as it may also generate cones */
        GLdouble height,
        GLint slices, /* number of subdivisions around z axis */
        GLint stacks); /* number of subdivisions along z axis */
```

Exercise 2.1
Create a simple program, using GLUT, SDL or Python based on the previously provided skeletons, and use the following code snippet to draw the cylinder

```
GLUquadric* myQuad;
myQuadr = gluNewQuadric();
gluCylinder( myQuad, 1.0, 1.0, 2.0, 10, 10 );
```

Make sure the cylinder base is at point (0,0,-6) Test it and try to answer the question of why does it appear like that.

Exercise 2.2
Change the program so that when you press the following keys it will show the following behaviour:

x toggle a continuous rotation of 2 degrees/s about the x-axis

y toggle a continuous rotation of 2 degrees/s about the y-axis

z toggle a continuous rotation of 2 degrees/s about the z-axis

q toggle a pos-translation of 3 units along the x-axis

w toggle a pos-translation of 3 units along the y-axis

e toggle a pos-translation of 3 units along the z-axis

[1]Find the meaning of extrusion if you don't know it. Then check if in computer graphics it retains the same meaning.

a toggle a pre-translation of 3 units along the x-axis

s toggle a pre-translation of 3 units along the y-axis

d toggle a pre-translation of 3 units along the z-axis

Suggestion: Consider the user of boolean variables, where their values are changed using $a = !a$.

Exercise 2.3
Attach a blue sphere to the cylinder top, and a green disk to the base with the same radius so that they close the cylinder. Observe how the previous transformations affect the compound object.

Exercise 2.4
Attach to the cylinder another sphere of radius 1 unit, whose center is at 1 unit from the cylinder axis, and at a distance of 1 unit from the cylinder base. Observe how the previous transformations affect it.

2.3 Specifying Transformations via Homogeneous matrices

```
void glMultMatrixd(const GLdouble * m);
void glMultMatrixf(const GLfloat * m);
```

glMultMatrix multiplies the current matrix with the one specified using m, and replaces the current matrix with the product. Note that m points to m consecutive values used as elements of the 4×4 column-major matrix. Pay attention that C matrices are row-major.

Consider the definition of matrices like follows, without forgetting that they should be column major.

```
typedef GLfloat Matrix4x4 [4][4];
Matrix4x4 matTrans3D;
```

Exercise 2.5
Repeat some of the previous transformations by replacing glRotate and glTranslate by the application of the corresponding homogeneous matrices. Compare the results by switching from the use of GL transformation primitives to the matrix equivalents by pressing the **m** key.