USING DYNAMIC TECHNIQUES IN COMPUTER GRAPHICS APPLICATIONS

Tănasie Răzvan, Tunaru Cristina

University of Craiova, Faculty of Automation, Computers and Electronics,
Software Engineering Department,
Bvd. Decebal, Nr. 107, Craiova, Romania,
E-mail: tanasie_razvan@software.ucv.ro
E-mail: krysmary8@yahoo.com

Abstract: This paper focuses on animation creation based on time dependent transformations in a real case and presents a solution that uses computer graphics techniques in interactive applications. The application is elaborated starting with a story theme and every character or element of the scene has a transformation matrix attached that defines the final movement of the object.

Keywords: computer graphics, rendering, DirectX, mesh, vision.

1. INTRODUCTION

Computer graphics is a discipline which is situated between the computer digital system programming and computation techniques that concern image design.

The object of computer graphics is the creation, the storage and the manipulation of the models of some objects and effects as well as the images attached to them, using the digital computer. A science connected with this is the computer image processing that deals with images using the digital computer.

The two disciplines have some common aspects, but they use different methods and techniques. The current vision is adjacency between them, especially after the appearance of the high resolution raster graphics. Even the mathematical instruments they use are complementary: the image design is based on geometry and algebra, while the image analysis is based on trigonometry and mathematical analysis.

In this paper one of the great aspects in computer graphics is discussed, namely, the creation of animation. It is true, the animation can be done in 3D Studio Max, but this manner is not our objective.

In DirectX9, the animation can be created using frames or by simple and complex transformations over the objects, transformations that are time dependent.

This paper debates about how to create animation with basic time dependent transformations in a real case: using computer graphics techniques in interactive applications. The application is elaborated starting with a story theme and every character or element of the scene has a transformation matrix attached that defines the final movement of the object.

Beside the transformation matrices modifying the form of the object that will be discussed, every component part of the 3D scene has a projection into the view space. For this operation two functions are used: the *WINAMPID3DXMatrixLookAtLH()* function, whose parameters specify the position where the camera is placed in the world, a point in the world where the camera aims and a vector that indicates which direction is “up” in the 3D world (usually this is the Oy axis). This function sets the view matrix.

WINAMPID3DXMatrixLookFovLH() function is used for setting the projection matrix. The projection

transformation defines our viewing volume (frustum) and is responsible for projecting the geometry in the frustum onto the projection window. The projection matrix is complex, using a view space (usually ¼ pi), distance to near and to far plane and aspect ratio (the geometry of the projection window is eventually transformed to screen space; the transformation from a square – projection window – to the screen, witch is a rectangle, causes a stretching distortion; the aspect ratio is simply the ratio between the screen's two dimensions and it is used to correct the distortion caused by mapping from a square to a rectangle; *aspectRatio = screenWidth/screenHeight*).

To make one or more transformations to an object the transformation matrix must be set, based on which are the transformations for the object and which is their application order. The application of a number of fundamental transformations to an object by modifying its position in time is made by modifying the object's initial coordinates. The most important thing in creating animation is to find the time dependence for the evolution of the coordinates of those objects.

2. GENERAL ASPECTS

In order to set different objects, either meshes or simple objects created in Direct3D, transformation matrices were used. To generate the animation both static and dynamic transformations were used. The matrices used to create all the operations are homogeneous matrices. The order of the fundamental transformations application it is very important, as well as the choice of time variables.

Depending on the order of the operations application, objects might have a different visualization. For instance, if an object, situated initially in the origin, is first rotated and then translated, the result will be a rotation around an axis, but the axis is out of origin). Instead, if first a translation is done and then a rotation (the rotation is time dependent) the result will not be a rotation around Oy axis (for example) where the object circulates round Oy axis like it is attached on this axis, but a rotation around Oy axis with a radius determined by the translation vector (see Fig. 1.).

Fig. 1. Different results for different order of transformation application – rotation and translation.

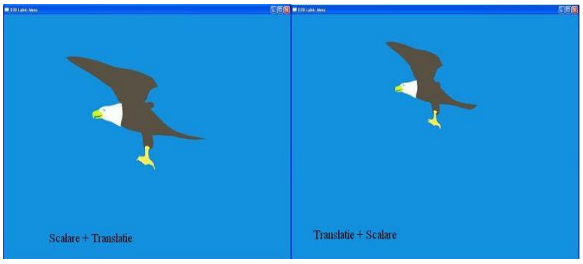

Fig. 2. Different results for different order of transformation application – scaling and translation.

Let's analyze the evolution of a 3D object when it is translated and scaled, in both situations (first a scaling then a translation; first a translation, then a scaling). The scaling factors and the point where the translation is made are the same in both situations. The difference is presented in Fig. 2.

The objects used for various transformations can be objects created with drawing primitives from the DirectX libraries or the meshes from 3D Studio Max loaded into Direct3D object. If we are dealing with a drawn object, this is seen like as a whole unit. In the case of meshes, to create animation can be more complex. A mesh consists of one or more subsets. A subset is a group of triangles in the mesh that can be all rendered using the same attributes. Let's take the following situation: a gremlin (every component part of the creature is a subset) that trembles across a fire hole and, during this time, it moves his tail. In order to realize this animation, both the gremlin's body (all that represents the body, except the tail) and his tail, will do a periodical translation movement on Oy axis, at the same time. More, over the tail will make a supplementary movement, a translation or a rotation (it depends what kind of movement is wanted for this object) generated by means of a different time dependency from the first used.

3. ANIMATION REALIZATION

These being said, let's suppose that a complete and permanent rotation to a mesh is needed. First, apply to the object o periodical rotation around Oy axis, then it will be translated into the point from where the rotation will start. Notice that the transformation that appears first in the line of multiplied transformations will actually be the last to be applied.

The translation matrix attached to this operation is:

$$T = \begin{bmatrix} 1 & 0 & 0 & 18 \\ 0 & 1 & 0 & 9 \\ 0 & 0 & 1 & -10 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (1)$$

and the rotation matrix around Oy axis:

$$R_y = \begin{bmatrix} \cos t & 0 & \sin t & 0 \\ 0 & 1 & 0 & 0 \\ -\sin t & 0 & \cos t & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (2)$$

where $t = cos\ \pi * timeGetTime()/1500$
$= -timeGetTime()/1500$

It is noticeable the way how the rotation period is chosen.

Then,

$$\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = R_y \cdot T \cdot \begin{bmatrix} x_0 \\ y_0 \\ z_0 \\ 1 \end{bmatrix} \quad (3)$$

That means we have the next relation after the determinations:

$$\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} \cos t & 0 & \sin t & 18\cos t - 10\sin t \\ 0 & 1 & 0 & 9 \\ -\sin t & 0 & \cos t & -18\sin t - 10\cos t \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x_0 \\ y_0 \\ z_0 \\ 1 \end{bmatrix} \quad (4)$$

Change $t = -t$, then:

$$\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} \cos t & 0 & \sin t & 18\cos t + 10\sin t \\ 0 & 1 & 0 & 9 \\ \sin t & 0 & \cos t & 18\sin t - 10\cos t \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_0 \\ y_0 \\ z_0 \\ 1 \end{bmatrix} \quad (5)$$

Then the final coordinates are some time dependent functions definited to R →R.

$$\begin{cases} x(t) = x_0\cos t + z_0\sin t + 18\cos t + 10\sin t \\ y(t) = y_0 + 9 \\ z(t) = x_0\sin t + z_0\cos t + 18\sin t - 10\cos t \end{cases} \quad (6)$$

or:

$$\begin{cases} x(t) = (x_0 + 18)\cos t + (z_0 + 10)\sin t \\ y(t) = y_0 + 9 \end{cases} \quad (7)$$

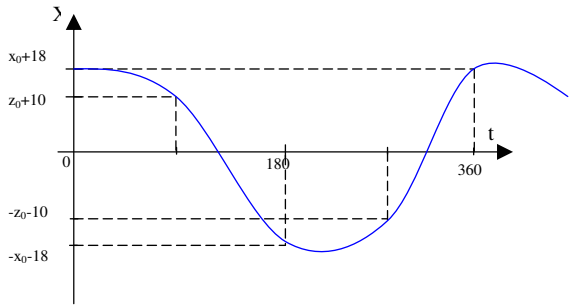

Fig. 3. Time dependency of the x coordinate for equation 6.

It can be observed that the initial value of y coordinate is modified only by translation, the rotation preserving it. The approximate graphic of the x(t) function can be seen in Fig. 3. The function repeats periodically as can be observed in the figure.

The graphic for z(t) function is similar (Fig. 4.).

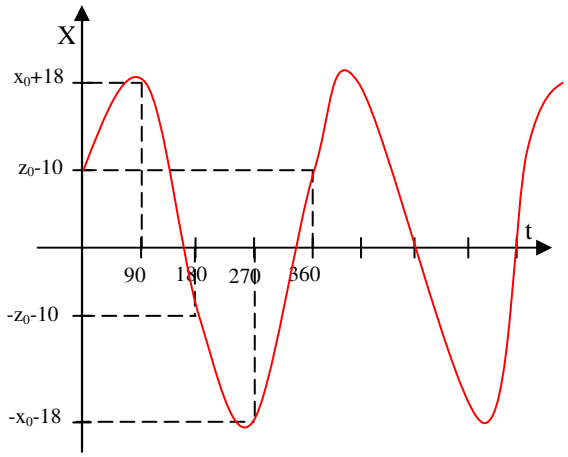

Fig. 4. Time dependency of the z coordinate for equation 6.

If a rotation around an axis different from the system axis, the transformation matrices, applied in the same order, look like this:

$$T = \begin{bmatrix} 1 & 0 & 0 & -18 \\ 0 & 1 & 0 & 3 \\ 0 & 0 & 1 & -10 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (8)$$

The rotation matrix around (2, 50, 1) axis is:

$$R_{axis} = \begin{bmatrix} (1-\cos t)x_0^2 + \cos t & (1-\cos t)x_0y_0 + z_0\sin t & (1-\cos t)x_0z_0 - y_0\sin t & 0 \\ (1-\cos t)x_0y_0 - z_0\sin t & (1-\cos t)y_0^2 + \cos t & (1-\cos t)y_0z_0 + x_0\sin t & 0 \\ (1-\cos t)x_0z_0 + y_0\sin t & (1-\cos t)y_0z_0 - x_0\sin t & (1-\cos t)z_0^2 + \cos t & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (9)$$

where $x_0=2$, $y_0=50$, $z_0=1$.

By replacing the values with the coordinates of the rotation axis, we obtain:

$$\begin{bmatrix} 4-3\cos t & (1-\cos t)100+\sin t & (1-\cos t)2-50\sin t & 0 \\ (1-\cos t)100-\sin t & 2500-2499\cos t & (1-\cos t)50+2\sin t & 0 \\ (1-\cos t)2+50\sin t & (1-\cos t)50-2\sin t & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (10)$$

where $t = timeGetTime()/1000$.

This means that after the calculations, the following expression is obtained:

$$\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} 4-3\cos t & (1-\cos t)100+\sin t & (1-\cos t)2-50\sin t & a \\ (1-\cos t)100-\sin t & 2500-2499\cos t & (1-\cos t)50+2\sin t & b \\ (1-\cos t)2+50\sin t & (1-\cos t)50-2\sin t & 1 & c \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_0 \\ y_0 \\ z_0 \\ 1 \end{bmatrix} \quad (11)$$

Where:

$$\begin{cases} a = -18*(4-3\cos t) + 3*(100 - 100\cos t + \sin t) + 2 - 2\cos t - 50\sin t = 230 - 248\cos t - 47\sin t \\ b = -18*(100 - 100\cos t - \sin t) + 3*(2500 - 2499\cos t) + 50 - 50\cos t + 2\sin t \\ \quad = 5750 - 5747\cos t + 20\sin t \\ c = 18*(2 - 2\cos t + 50\sin t) + 3*(50 - 50\cos t - 2\sin t) + 1 \\ \quad = 187 - 186\cos t + 894\sin t \end{cases} \quad (12)$$

Then, the final coordinates are some time functions on R→R:

$$\boldsymbol{x(t)} = x_0*(4-3\cos t) + y_0*[100*(1-\cos t)+\sin t] + z_0[2*(1-\cos t)-50\sin t] + 230 - 248\cos t - 47\sin t \quad (13)$$

$$\boldsymbol{y(t)} = x_0*[100(1-\cos t)-\sin t] + y_0*[2500-2499\cos t] + z_0[50(1-\cos t)+2\sin t] + 5750 - 5747\cos t + 20\sin t \quad (14)$$

$$\boldsymbol{z(t)} = x_0*[2*(1-\cos t)+50\sin t] + y_0*[50*(1-\cos t)-2\sin t] + z_0 + 187 - 186\cos t + 894\sin t \quad (15)$$

The approximate graphic of the x(t) function is presented in Fig. 5. The function repeats periodically as can be seen in the figure.

The graphic for z(t) coordinate is similar (Fig. 6.).

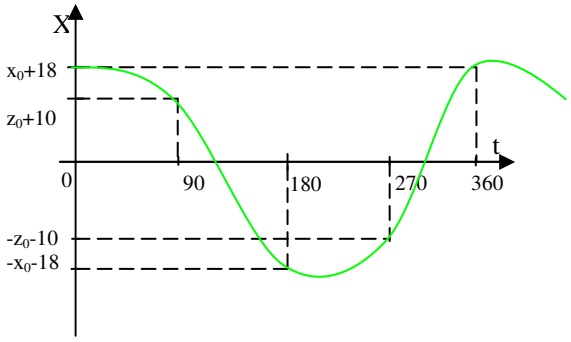

Fig. 5. Time dependency of the x coordinate for equation 13.

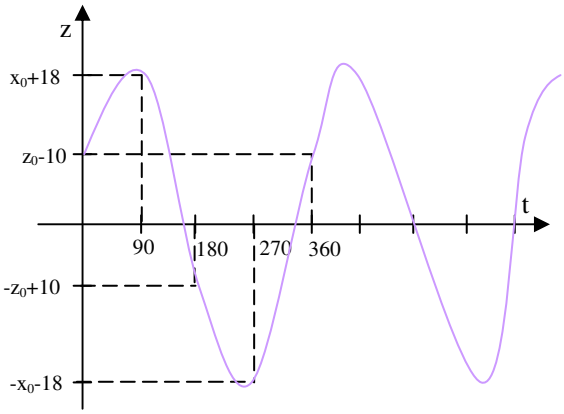

Fig. 6. Time dependency of the z coordinate for equation 15.

For a translation movement to an object for only a short time, the final coordinates of the objects are given by this relation:

$$\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = T \cdot \begin{bmatrix} x_0 \\ y_0 \\ z_0 \\ 1 \end{bmatrix} \quad (16)$$

$$T = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (17)$$

where $t = timeGetTime()\%32000/1000$

So,

$$\begin{cases} x(t) = x_0 \\ y(t) = y_0 + t \\ z(t) = z_0 \end{cases} \quad (18)$$

Then, the homologous values for x and z are some constants, namely, the first values of the object on these axis; only y(t) function depends linearly on time.

How can the definition field for y(t) function be calculated? First, it is known that *timeGetTime()%32000* has only integer values, in *{0, 1, 2 … 31999}*. Then, *(timeGetTime()%32000)/1000* can only have integer values in: *{0, 1, … 31}*. So, this function has only 32 values, but it uses only 29, because the object is translated just as long as *(timeGetTime()%32000)<29000,* otherwise, the object has a static translation.

Then, it can be written the following function in order to find the coordinates:

$$x(t) = x_0 \quad (19)$$

$$z(t) = z_0 \quad (20)$$

$$y(t) = \begin{cases} y_0 + t, \text{ if } t \in \{0, 1, \ldots, 28\} \\ y_0 + 29, \text{ if } t \in \{29, 30, 31\} \end{cases} \quad (21)$$

The graphic for x(t) is in Fig. 7.

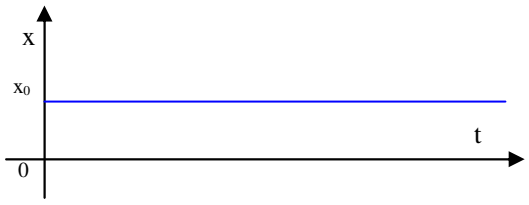

Fig. 7. The graphic for x(t) function for equation 19.

The graphic for z(t) is in Fig. 8.

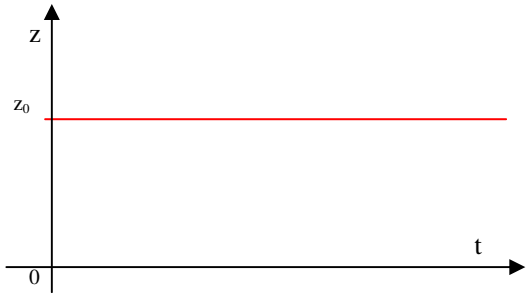

Fig. 8. The graphic for z(t) function for equation 20.

But, for y(t), the graphic is different as can be seen in Fig. 9.

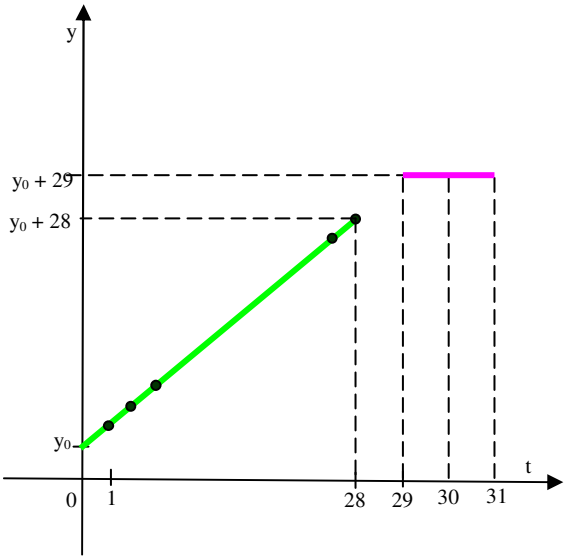

Fig. 9. The graphic for y(t) function for equation 21.

Other composition of fundamental transformations, more challenging, is when the movement of some wings is needed. But some wings beating into a point in space are not something spectacular. For realism, they will be translated together with a body where they are attached. Usually, or at least in this case, the wings and the body are parts of the same mesh. Next two time cycles will be chosen, the first one being used only for the wings, the other, for both the wings and the body.

The time is calculated as always, but a rotation in t_1=timeGetTime()%500/1500 and a translation in t_2=timeGetTime()%20000/1000 are used.

I-st case

The operations applied are a scaling, a rotation around Ox axis and a translation.

$$\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = [Trd\ ] \cdot [Trs\ ] \cdot [Rx\ ] \cdot [Sc\ ] \cdot \begin{bmatrix} x_0 \\ y_0 \\ z_0 \\ 1 \end{bmatrix} \quad (22)$$

The matrices are:

$$[Trd\ ] = \begin{bmatrix} 1 & 0 & 0 & t_2 + 1 \\ 0 & 1 & 0 & t_2 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (23)$$

Take into consideration that:

$t_2 = t \,\%\, 20000 / 1000$ (24)

$t_1 = t \,\%\, 500 / 1500$ (25)

$$[Trs\ ] = \begin{bmatrix} 1 & 0 & 0 & -10 \\ 0 & 1 & 0 & -10 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (26)$$

$$[Rx\ ] = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos t_1 & -\sin t_1 & 0 \\ 0 & \sin t_1 & \cos t_1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (27)$$

$$[Sc\ ] = \begin{bmatrix} 0.002 & 0 & 0 & 0 \\ 0 & 0.002 & 0 & 0 \\ 0 & 0 & 0.002 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (28)$$

Substitute the values and make all the calculations, and obtain:

$$\begin{cases} x(t) = 0.002\ x_0 + t_2 - 9 \\ y(t) = 0.002\ y_0 \cos t_1 - 0.002\ z_0 \sin t_1 + t_2 - 10 \\ z(t) = 0.002\ y_0 \sin t_1 + 0.002\ z_0 \cos t_1 - 9 \end{cases} \quad (29)$$

II-nd case

A rotation around Oy axis, a scaling, a rotation round Ox axis and a translation are applied. So, the final relation is:

$$\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = [Tr\ ] \cdot [Rx\ ] \cdot [Sc\ ] \cdot [Ry\ ] \cdot \begin{bmatrix} x_0 \\ y_0 \\ z_0 \\ 1 \end{bmatrix} \quad (30)$$

where

$$[Tr] = \begin{bmatrix} 1 & 0 & 0 & 10 \\ 0 & 1 & 0 & 10 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix} \quad (31)$$

$$[Rx] = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos t_1 & -\sin t_1 & 0 \\ 0 & \sin t_1 & \cos t_1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (32)$$

$$[Sc] = \begin{bmatrix} 0.002 & 0 & 0 & 0 \\ 0 & 0.002 & 0 & 0 \\ 0 & 0 & 0.002 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (33)$$

$$[Ry] = \begin{bmatrix} \cos a & 0 & \sin a & 0 \\ 0 & 1 & 0 & 0 \\ -\sin a & 0 & \cos a & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad (34)$$

where = $4\pi/3$.

Finally,

$x(t) = 0.002x_0 \cos a + 0.002\ z_0 \sin a + 10$ (35)
$y(t) = 0.002\ x_0 \sin a \sin t_1 + 0.002\ y_0 \cos t_1 - 0.002\ z_0 \cos a \sin t_1 + 10$ (36)
$z(t) = 0.002\ x_0 \sin a \cos t_1 + 0.002\ y_0 \sin t_1 + 0.002\ z_0 \cos a \cos t_1 + 1$ (37)

For the body of the flying man a scaling, a translation and a rotation are applied. The flying man's body and the wings are parts of the same mesh; as a result, both the rotation and the translation are made with the same time dependent function and the same position.

Time is kept timeGetTime()%5000/1000, the rotation is made with 100t and the translation is made with -16 on Oy.

For $t \in \{0\ldots3\}$ it results:

$$\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = [Ry] \cdot [T] \cdot [Sc] \cdot \begin{bmatrix} x_0 \\ y_0 \\ z_0 \\ 1 \end{bmatrix} \quad (38)$$

Then, after computing, the results are:

$x(t) = 0.002 \cos 100t\ x_0 + 0.002\ z_0 \sin 100t$
$y(t) = 0.002\ y_0 - 16$ (39)
$z(t) = -0.002\ x_0 \sin 100t + 0.002\ z_0 \cos 100t$

For $t \geq 4$, apply the same scaling and rotation matrix, but the rotation around Oy axis is made with 0 degrees, therefore the matrix has the principal diagonal 1.

Therefore, the second case has the following results:

$$\begin{cases} x(t) = 0.002*x_0 \\ y(t) = 0.002*y_0 - 16 \\ z(t) = 0.002*z_0 \end{cases} \quad (40)$$

The three functions are x,y,z : $\{0\ldots4\} \rightarrow R$.

$$x(t) = \begin{cases} 0.002\cos 100t \cdot x_0 + 0.002 z_0 \sin 100t, t \in \{0..3\} \\ 0.002 x_0, t \geq 4 \end{cases} \quad (41)$$

$$y(t) = \begin{cases} 0.002\ y_0 - 16 \\ 0.002\ y_0 - 16 \end{cases}, \quad (42)$$

$$z(t) = \begin{cases} -0.002 x_0 \sin 100t + 0.002 z_0 \cos 100t, t \in \{0..3\} \\ 0.002 z_0, t \geq 4 \end{cases} \quad (43)$$

4. CONCLUSIONS

Computer graphics and especially animation is a complex but also a very useful aspect of most of the present applications.

The application is implemented in Microsoft Visual C++ 7.0 (Microsoft Visual Studio .NET) and uses DirectX 9.0c libraries. The meshes are conceived in 3D Studio Max. The purpose of the application is to manipulate the meshes dinamycally in order to animate the scene. This is done by using the DirectX functions for basic transformations with time dependent parameters.

The project can be further developed by adding more complex and more artistic elements and stages created in 3D Studio Max or other modelling software. Also the optimization of the program can be moved to a superior level, thus giving the possibility to run complex animated software on a computer with medium resources.

5. REFERENCES

Adams, J. (2004), *Programming Role Playing Games with DirectX, Second Edition (Game Development Series),* Premier Press, USA.

Gray K. (2003), *The Microsoft DirectX 9 Programmable Graphics Pipeline*, Microsoft Press, USA.

Jones, W. (2004), *Beginning DirectX 9*, Premier Press, USA.

LaMothe A. (2003), *Tricks of the 3D Game Programming Gurus-Advanced 3D Graphics and Rasterization,* SAMS, USA.

Luna, F. D. (2003), *Introduction to 3D Game Programming with DirectX 9.0*, Wordware Publishing, Inc., USA.

Miller T. (2003), *Managed DirectX 9 Kick Start : Graphics and Game Programming*, SAMS, USA.

Snook G. (2003), *Real-Time 3D Terrain Engines Using C++ and DirectX 9 (Game Development Series)*, Charles River Media Inc., USA.