I.J. Image, Graphics and Signal Processing, **2011, 5, 44-50**
Published Online August 2011 in MECS (http://www.mecs-press.org/)

A Rotation Transformation Method of 3D Object in WPF by Modifying Camera Attributes

YU Ren
Navy Engineering University
Wuhan, China
yurenarticle@hotmail.com

Mao Wei
Navy Engineering University
Wuhan, China
weimao@yahoo.com.cn

LU Gubin
Navy Engineering University
Wuhan, China
lugubing@yahoo.com.cn

Lu Feng
XINXIANG Technology Co. Lited.
Wuhan, China
lfausten@hotmail.com

Abstract— **Based on the .NET Framework 2.0, the new generation of the windows software development framework .NET Framework 3.0 has four new components, in which the most attractive one is WPF(Windows Presentation Foundation). WPF is a new GUI engine that can provide uniform descriptions and methods for user interface, 2D/3D graphic, document and media. In WPF, the commonly used rotation transformation method for 3D object is RotateTransform3D. This method needs to calculate the state of the camera carefully, otherwise, the 3D object may move anomaly on the screen. To solve this kind of problem, another method is brought forward in the paper, which realizes the rotation transformation by calaulating and changing the attributes (Position, LookDirection, UpDirection) of the camera directly. The method can exhibite the rotation of 3D object distortion-freely, without the anomal movement on the screen. The calculation and the program of the method is simple.**

Index Terms-WPF; Three Dimention Rotation; GUI Engine; .NET Framework

I. INTRODUCTION

In an information management system which needs to exhibit a 3D object, such as an equipment or facility, the rotation transformation of the 3D object is a key function when developing the software. Usually, this is carried out by Direct3D[1] or OpenGL[2], which require the developers have abundant special knowledges, and the workload of program is high. Another way is to utilize the specialized 3D graphical engine, which is generally expensive, and has many functions unnecessary.

The release of Windows Vista brings forward a new generation of Windows software development framework, that is, the .NET Framework 3.0, which brings four new groupwares. One of them is the WPF (Windows Presentation Foundation)[3], which is a new GUI engine, can provide uniform interface technique for different applications. It is based on the DirectX 9/10, and makes the graphic disposal of 3D object easy. The WPF provides abundant .NET UI framework, integrates vector graphic, flow text support, 3D vision effect, and powerful controls model framework.

The rotation transformation methods for 3D object in WPF are[4]:

- By using RotateTransform3D method.
- By calculating and modifying the camera attributes: Position, LookDirection, UpDirection.

When rotating the 3D object with RotateTransform3D method, it needs to calculate the state of the camera carefully, otherwise, the 3D object may move anomaly on the screen, and puzzle the user.

To avoid this problem, another method which realizes the rotation transform by calculating and modifying the camera's attributes directly is brought forward in the paper. The way to describe the 3D object in WPF is explained at first in section 2, with some instances in section 4 about the rotation of 3D object in 3D scene when the mouse is moving. In section 5, the calculation of the two key parameters is illustrated, that is: the coordinates of the rotation point and the radial of the camera rotation orbit. In section 6, the procedures and methods to calculate the attributes of the camera are discussed in detail. The method for mapping the track of the mouse on the screen to the three dimension coordinates is given in Section 7, so as to realize the control of the rotation by moving the mouse on the screen.

II. THE DESCRIPTION OF 3D OBJECT IN WPF

In WPF, the 3D graphic object is resided in the Viewport3D control. The ViewPort3D is required to host any 3D models, serves as a container in three dimentions scene, and is the projection of a 3D object in three dimentions scene to the two dimentions plane[5].

a. The describtion of 3D graphic

As generally, all the 3D graphics in WPF are described as a set of a series of triangle, as shown in Figue 1. That is to say, the triangle is the smallest geometric solid to describe a plane. In the sence of the WPF, the color of each triangle can be calculated with the render engine, according to its texture and the angle between the plane and the light. The reason of utilizing triangle is that all the points in a triangle can be ensured to be in a same plane. Thus, the calculation and rendering will be simple.

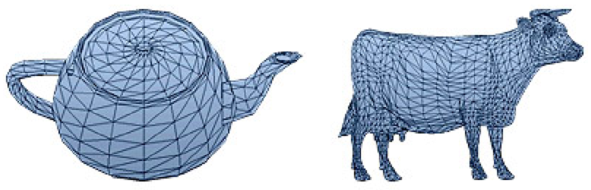


Figure 1. Triangles to describe 3D object

A plane on a 3D object is called a mesh. A mesh is defined with a series of 3D points. These points are called vertices. All these vertices are connected with an encirclement pattern to form many triangles. Each triangle has its obverse side and reverse side, and only obverse side will be rendered. The obverse side of a triangle is confirmed by the order of encirclement of the vertices. In WPF, anticlockwise encirclement pattern is utilized. The obverse side can be judged with "right hand rule", which means, close your right hand, stick up your thumb, encircle other fingers with anticlockwise pattern, then the direction of your thumb is the obverse side of the triangle. The "right hand rule" is illustrated in Figure 2.

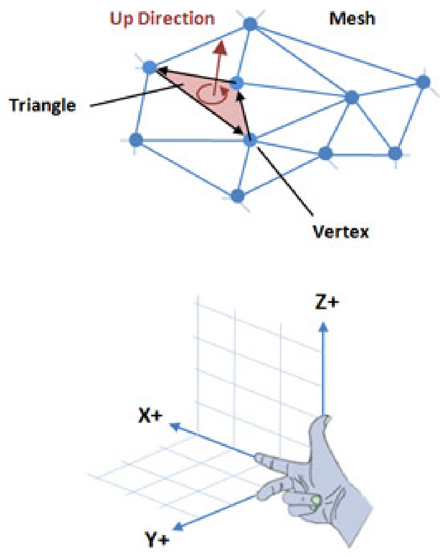


Figure 2. "Right Hand Rule" to judge the frontispiece of a Triangles

In WPF, the three demention graphic is defined with the following parts:

a. Three demention attibutes

Two kinds of 3D point structure are defined in WPF: Point3D and Point4D. The Point3D structure defines X, Y, Z coordinates in rectangular coordinates system in 3D space. The Point4D structure defines X, Y, Z and W coordinates in homogeneous coordinates system. The homogeneous coordinates system is used for nonaffine transform of the 3D matrix. The Vector3D structure defines the X, Y, Z components in 3D space. A vector in Vector3D is a row array formed with three elements, that is, the X, Y, Z coordinate.

The Vector3D has five public attributes:

Length - get the length of a Vector3D object.

LengthSquared – get the square of the length of a Vector3D object.

X – get or set the X component of a Vector3D object.

Y - get or set the Y component of a Vector3D object.

Z - get or set the Z component of a Vector3D object.

b. Three demention scene

A three demention scence contains these important elements:

- Camera and coordinates system

There is a virtual camera in Viewport3D scene. Its position, origin and other attributes determine the angle of the scene. The coordinates system in which the camera placed is a world coordinates system.

- Transpform3D

The class Transform3D is used for the zoom, rotation and translation of a 3D object within the 3D space. It can be used on the attributes of the objects Model3D, ModelVisual3D, Camera, et al.. When this attribute is set, the 3D object will be transformed to the new corresponding position.

- Model3D

The class Model3D is similar to class Drawing in 2D scene. It is a fundamental component to construct 3D model in the scene. It has three subclass:

➢ Light：to calculate automatically the light and shade effect in 3D scene according to the distance of the light source. If there is no light source, the scene will be dark.

➢ GeometryModel3D：to frame the 3D geometric object. Together with Geometry and Material attributes, it can construct a colored 3D geometric object.

➢ TextureCoordinates：provide the mapping of 3D to 2D for each Material.

- Visual3D

The class Visual3D is used for displaying the 3D object. Only when a 3D object is put into the Visual 3D, can it be displayed. The Visual3D has only one subclass ModelVisual3D.

c. Three demention transformation

There are two methods for three **demention transformation** in WPF:

- Using RotateTransform3D method. The status of the camera should be treated carefully when this method is used. Otherwise, the 3D object may move abnormally.
- Calculating and modifying the camera attributes: Position, LookDirection, UpDirection.

In WPF, there are two kinds of camera: orthographical camera and perspective camera. In the paper, the perspective camera is utilized. The attributes of a camera are:

- Position： defines the position of the camera. Different views of the scene can be established when moving Camera. The Position is the style of Point3D, and has the values X, Y and Z. It can be calculate simply: the coordinates of a point, such as the origin (0,0,0), minus the coordinates of the Camera, and the results are the values of the Position.
- LookDirection：defines the visual angle vector of the camera, that is, the direction of the Camera. It is the style of Vector3D.
- UpDirection：defines the upward vector of the camera. It is the style of Vector3D, with default value (0, 1, 0).
- FieldOfView： defines the scare of the visual angle of the camera.

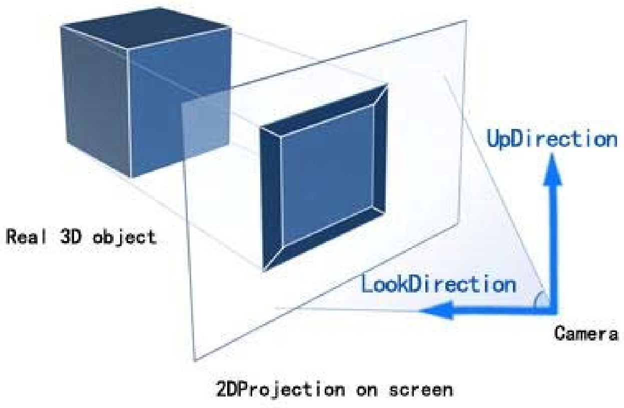


Figure 3. The relationship of the camera's attributes and the 3D object

The relationship of the camera's attributes and the 3D object is illustrated in Figure 3.

In the paper, the method of modifying the camera attributes is adopted to perform the rotation of 3D object.

III. THE INSTANCE OF THE ROTATION OF 3D OBJECT

When the camera is moving on a sphere, the 3D object will rotate in the 3D scene. The relationships of the mouse and the 3D object from the observer point of view are shown in Figure 4 and 5.

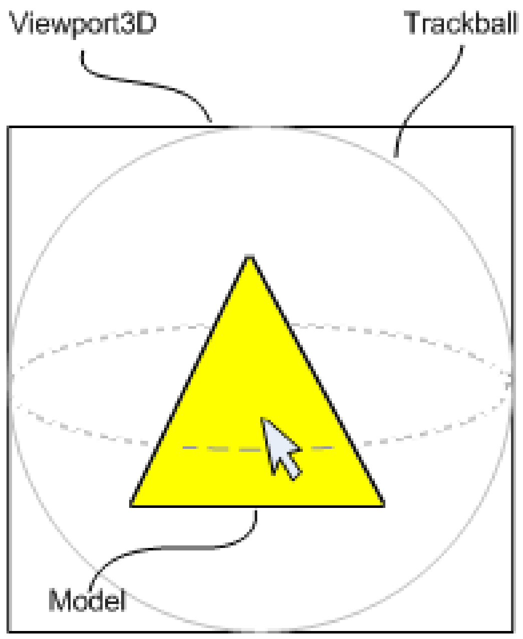


Figure 4. The relationship of the mouse and the 3D object from the observer point of view, front side.

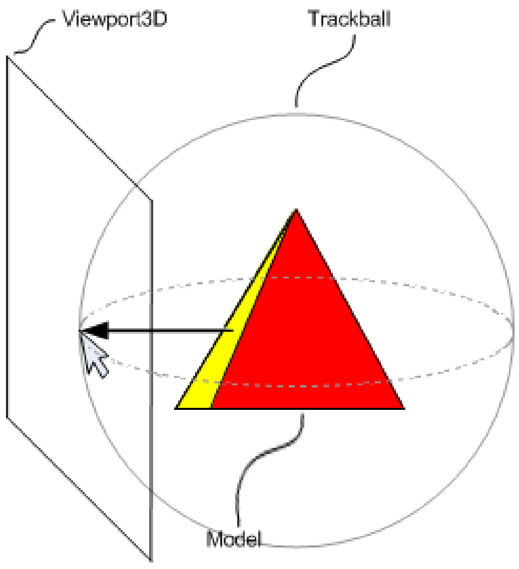


Figure 5. The relationship of the mouse and the 3D object from the observer point of view, flank side.

When the mouse is moving horizontally, the rotation of the Y axis must be maintained at the same point of the mouse pointer, as shown in Figure 6.

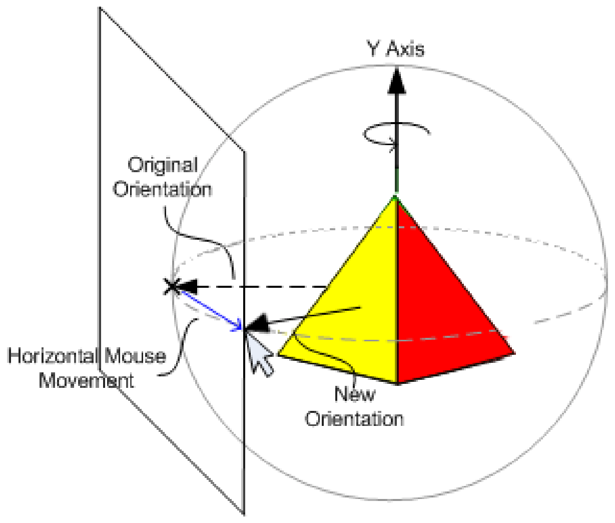


Figure 6. The relationship of the mouse and the 3D object when the mouse is moving horizontally.

When the mouse is moving vertically, the rotation of the Y axis must be maintained at the same point of the mouse pointer, as shown in Figure 7.

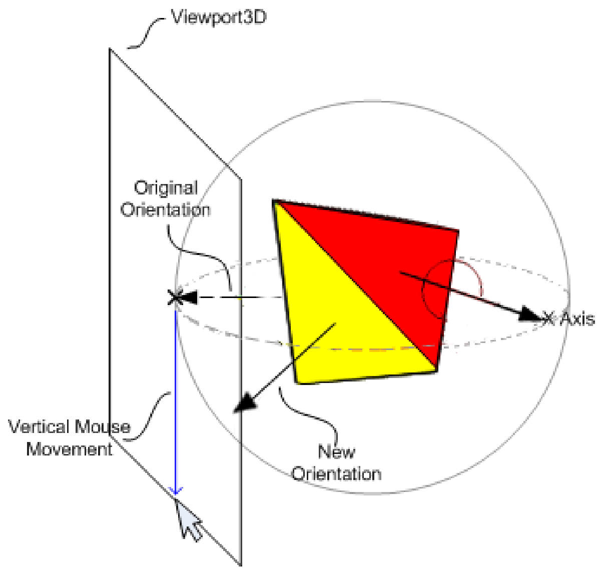
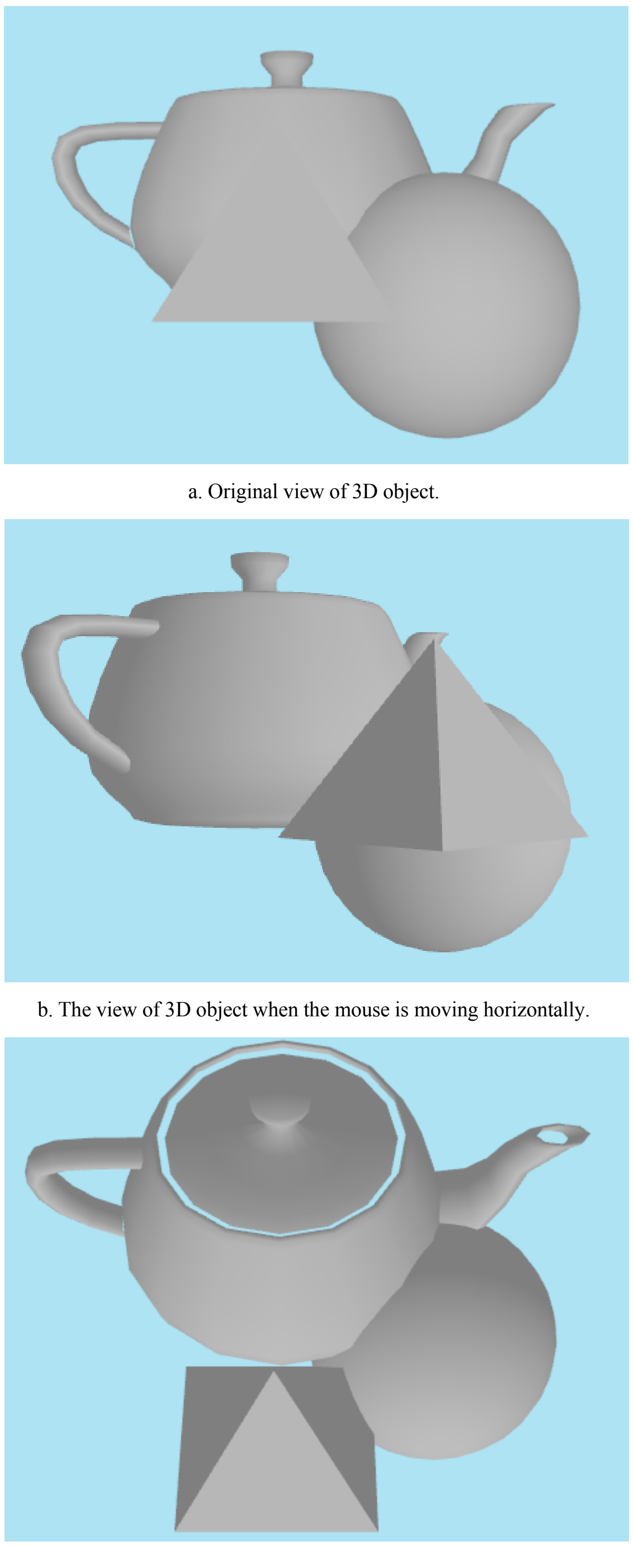


Figure 7. The relationship of the mouse and the 3D object when the mouse is moving vertically.

An instance of the rotation of a 3D object in 3D scene is shown in Figure 8.



a. Original view of 3D object.

b. The view of 3D object when the mouse is moving horizontally.

c. The view of 3D object when the mouse is moving vertically.

Figure 8. An instance of the rotation of a 3D with mouse moving.

IV. CALCULATION OF THE ROTATION POINT COORDINATES AND THE RADIAL OF THE CAMERA ROTATION ORBIT

Before the rotation transformation of the 3D object, two key parameters should be calculated. They are: the coordinates of the rotation point and the radial of the camera rotation orbit.

A. *The Coordinates of the Rotation Point*

The rotation point means the point in the three dimention rectangular coordinates, arroud which the camera rotates. This point is fixed to a 3D object, and the vision effect with this rotation transformation method should be that the 3D object rotates at this fixed point on the screen. The projection of a 3D object to the three axes is shown in Figure 9.

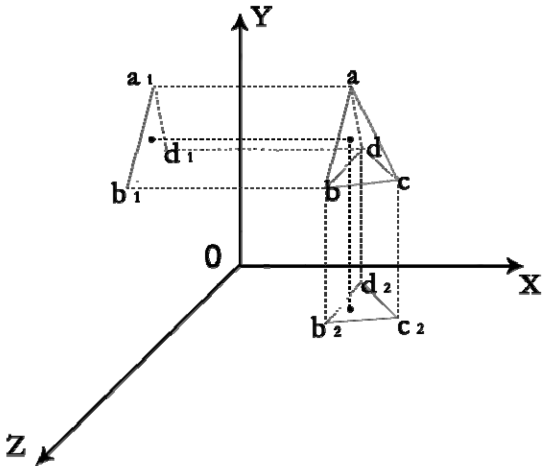


Figure 9. The projection of a 3D object to the three axes

In this paper, the rotation point is defined as the midpoint of the 3D object. That means, each of its coordinates is the median of the coordinates of the corresponding maximum and minimum points of the 3D object in the three dimention rectangular coordinates. In WPF, the boundary rectangle of the 3D object can be obtained easily with the method VisualTreeHelper.GetDescendantBounds, and the coordinates of the midpoint of this rectangle are the coordinates of the rotation point.

```
Rect3D bounds =
VisualTreeHelper.GetDescendantBounds(modelvisual3D);
Point3D p3d = new Point3D(bounds.X + (bounds.SizeX / 2), bounds.Y +
(bounds.SizeY / 2), bounds.Z + (bounds.SizeZ / 2));
```

B. *The Radial of the Camera Rotation Orbit*

The radial of the camera rotation orbit means the distance from the camera to the rotation point. When the camera moves on the spherical surface with this radial, the 3D object seems to rotate at a fixed position on the screen.

The radial can be gained by a vector, which equals to the vector of the camera position minus the vector of the rotation point. The radial is the length of this vector.

```
Vector3D v = new Vector3D(P.X, P.Y, P.Z);
Vector3D v1 = new Vector3D(camP.X, camP.Y, camP.Z);
double r = (v - v1).Length;
```

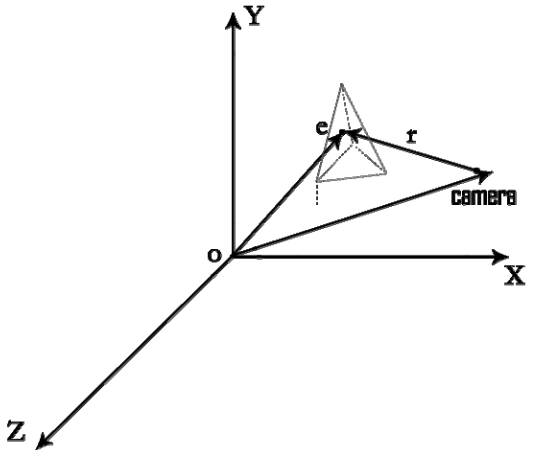


Figure 10. The relationship of the camera posision, the rotaion point and the radial

The relationship of the camera posision, the rotaion point and the radial is shown in Figure 10.

V. THE CALCULATION OF THE CAMERA ATTRIBUTES WHEN THE 3D OBJECT ROTATES

Let's image the camera moves on the surface of the sphere whose center point is the rotation point and has the radial defined previously. To rotate the 3D object, the attributes LookDirection and UpDirection should be modified to ensure the line of the sight of the camera can always fell on the 3D object. The attribute FieldOfView, that is, the visual angle of the camera, will not change before and after rotation.

In order to calculate the attributes LookDirection and UpDirection, the concept of the Sphere Coordinates System is introduced in this method, as shown in Figure 11. The sphere coordinates of a point A are :

α: the angle between the projection of the vector A on the X-Y plane and the X axis;

β: the angle between the the projection of the vector A on the X-Y plane and the vector A;

r: the length of the vector A.

In order to rotate the 3D object at a fixed point on the screen, the differences of the angle values α, β of the camera position point and the angle values α, β of the vertex of the LookDirection vector should be consistent respectively before and after the movement of the camera. So do the corresponding value differences between the camera position and the UpDirection vector. The calculation of the attributes LookDirection and UpDirection when the camera is at new position is based on these requirements.

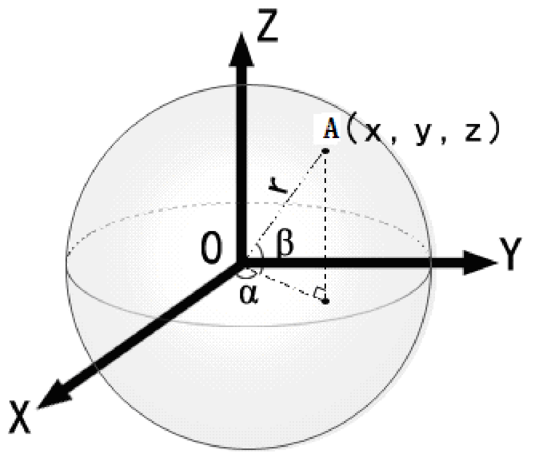


Figure 11. Sphere Coordinates System

A. *Calculation Procedure*

- In order to simplify the calculation, take the rotation point as new origin, establish a new parallel three dimension rectangular coordinates system and sphere coordinates system.
- Calculate the distances from the vertexes of the vectors of the LookDirection and the UpDirection to the rotation point.
- Calculate the angle differences from the vector vertexes of the LookDirection and UpDirection respectively to the camera position before the movement of the camera.
- Calculate the sphere coordinates of the vertexes of the LookDirection and the UpDirection at new position.
- Calculate the rectangular coordinates of the vertexes of the LookDirection and the UpDirection respectively at new position.
- Convert the rectangular coordinates of the two vectors' vertexes to the original rectangular coordinates.
- Modify the attributes of the camera in new position.

B. *Calculation Method*

- To the camera original position, the coordinates of the camera in a new rectangular coordinates system is (x_1,y_1,z_1), the corresponding sphere coordinates is (α_1, β_1 , r_1). The rectangular coordinates of the vertex of the LookDirection in new rectangular coordinates system is (x_2,y_2,z_2), the corresponding sphere coordinates is (α_2,β_2,r_2). The rectangular coordinates of the vertex of the LookDirection in new rectangular coordinates system is (x_3,y_3,z_3), the corresponding sphere coordinates is (α_3,β_3,r_3).
- To the camera new position, the rectangular and sphere coordinates of the camera at the new position and the vector vertexes of the LookDirection and the UpDirection in new rectangular and sphere coordinates system are (x_1',y_1',z_1'), (α_1', β_1', r_1'), (x_2',y_2',z_2'), (α_2', β_2', r_2'), (x_3',y_2',z_2'), (α_3', β_2', r_2'), respectively.
- The radial of the camera rotation orbit keeps unchanged before and after the movement of the camera.
- The formulas to calculate the sphere coordinates are:

$$\alpha_i = \arctan\frac{y_i}{x_i} \tag{1}$$

$$\beta_i = \arctan\frac{z_i}{\sqrt{x_i^2 + y_i^2}} \tag{2}$$

- The formulas to calculate the angle differences are:

$$\Delta\alpha_2 = \alpha_2 - \alpha_1 \qquad \Delta\beta_2 = \beta_2 - \beta_1 \tag{3}$$

$$\Delta\alpha_3 = \alpha_3 - \alpha_1 \qquad \Delta\beta_3 = \beta_3 - \beta_1 \tag{4}$$

- The formulas to convert the sphere coordinates of the two vectors' vertexes to the rectangular coordinates:

$$\begin{aligned} z_2' &= r_2' \times \sin(\beta_2' + \Delta\beta_2) \\ x_2' &= \sqrt{r_2'^2 - z_2'^2} \times \cos(\alpha_2' + \Delta\alpha_2) \\ y_2' &= \sqrt{r_2'^2 - z_2'^2} \times \sin(\alpha_2' + \Delta\alpha_2) \end{aligned} \tag{5}$$

$$\begin{aligned} z_3' &= r_3' \times \sin(\beta_3' + \Delta\beta_3) \\ x_3' &= \sqrt{r_3'^2 - z_3'^2} \times \cos(\alpha_3' + \Delta\alpha_3) \\ y_2' &= \sqrt{r_3'^2 - z_3'^2} \times \sin(\alpha_3' + \Delta\alpha_3) \end{aligned} \tag{6}$$

C. *The key codes*

- Code for parallel movement conversion of the coordinates.

```
// Rotation Point
Point3D RevolvePoint = ComputeRevolvePoint3D();
// Orijinal Camera Position
Point3D CameraP = new Point3D(pcamera.Position.X -
RevolvePoint.X, pcamera.Position.Y - RevolvePoint.Y,
pcamera.Position.Z - RevolvePoint.Z);
//Vertex of the LookDirection
Point3D LookDirectionP = new
Point3D(pcamera.LookDirection.X - RevolvePoint.X,
pcamera.LookDirection.Y - RevolvePoint.Y,
pcamera.LookDirection.Z - RevolvePoint.Z);
//Vertex of the UpDirection
Point3D UpDirectionP = new Point3D(pcamera.UpDirection.X -
RevolvePoint.X, pcamera.UpDirection.Y - RevolvePoint.Y,
pcamera.UpDirection.Z - RevolvePoint.Z);
// New Position
Point3D ObjectPoint = new Point3D(objectPoint.X -
RevolvePoint.X, objectPoint.Y - RevolvePoint.Y,
objectPoint.Z - RevolvePoint.Z);
```

- Code for the radial calculation

```
//旋转点向量 Vector of the rotation point
Vector3D RevolveV = new Vector3D(RevolvePoint.X,
RevolvePoint.Y, RevolvePoint.Z);
// Vector of the LookDirection
double r1 = (RevolveV - pcamera.LookDirection).Length;
```

- Code for the angles and the angle differences calaulation:

```
double a=Math.Atan(CameraP.Y / CameraP.X);
double b=Math.Asin(CameraP.Z / r);
double ap1=a1 – a+ ap;
double bp1=b1 – b+ bp;
```

- Code for the new coordinates calculation

```
double z1 = r1 * Math.Sin(bp1);
double x1 = Math.Sqrt(r1 * r1 - z1 * z1) * Math.Cos(ap1);
double y1 = Math.Sqrt(r1 * r1 - z1 * z1) * Math.Sin(ap1);
```

- Movement of the camera

```
// Convert the new coordinates to the original coordinates
system
Vector3D LookV = new Vector3D(x1 + RevolvePoint.X, y1 +
RevolvePoint.Y, z1 + RevolvePoint.Z);
Vector3D UpV = new Vector3D(x2 + RevolvePoint.X, y2 +
RevolvePoint.Y, z2 + RevolvePoint.Z);
// Modify the attributes of the camera, make it move.
pcamera.Position = objectPoint;
pcamera.LookDirection = LookV;
pcamera.UpDirection = UpV;
```

VI. CONTROL THE ROTATION OF 3D OBJECT WITH MOUSE

By now, the attributes of the camera at new position are calculated. The next problem is to get the movement track of the camera on the spherical surface from the track of the mouse on the two dimension screen. The mapping of the track in two dimension coordinates system to the spherical surface in three dimension coordinates system is shown in Figure 12.

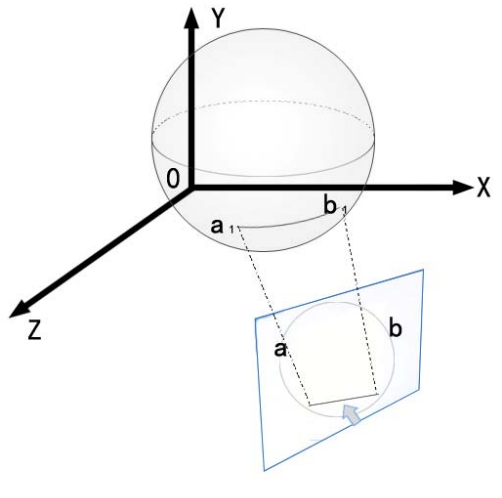


Figure 12. The mapping of the track in two dimension coordinates system to the sphere surface in three dimension coordinates system.

In fact, the track on a spherical surface has its projection on a two dimension screen. Contrarily, the two dimension coordinates of a point on a screen can be converted into the three dimension coordinates on the sphere surface. This can be easily done with the GeneralTransform2DTo3D method in WPF. Thus, the GeneralTransform2DTo3D method is used here to map the track of the mouse in two dimension coordinates system to the sphere surface in three dimension coordinates system.

Calculation Procedure

- Take the rotation point as the center point, the distance between the camera and the rotation point as the radial, build a virtual sphere.
- Convert the coordinates of the mouse track on the screen to the three dimension coordinates on the spherical surface with the GeneralTransform2DTo3D method.

```
GeneralTransform2DTo3D
gt1=viewport3d.TransformToAncestor(model);
Point3D point = gt.Transform(mousePoint);
```

- Take this three dimension coordinates as the new position of the camera, practise the rotation transformation with the described method.

VII. CONCLUSION

A new method for the rotation transformation of 3D object is illustrated in detail in the paper. The method utilizes the new GUI engine WPF from the Microsoft, and rotates the 3D object by calculating and modifying the attributes of the camera directly. The method can avoid the abnormal movement of the 3D object on the screen that may be caused by the RotateTransform3D method, exhibite the rotation distortion-freely. Compared with the Direct3D and OpenGL, the method has great pacticability, demands less calculation load and the program is simple.

REFERENCES

[1] Clayton Walnum, “Direct3D Programming Kick Start”, SAMS, ISBN 978-0672324987,2003

[2] Dave Shreiner, Mason Woo , Jackie Neider, Tom Davis , ”OpenGL(R) Programming Guide: The Official Guide to Learning OpenGL(R)”, Addison-Wesley Professional, 6th Edition, ISBN 978-0321481009, 2007

[3] Microsoft Developer Network for Visual Studio 2008

[4] Matthew MacDonald. “Pro WPF in C# 2008: Windows Presentation Foundation with .NET 3.5”, Second Edition. Apress, ISBN 978-1590599556, 2008

[5] http://viewport3d.com/trackball.htm