Computer and Information Science

www.ccsenet.org/journal.html

Vol. 2, No. 2

May 2009

# Study on the Virtual Simulation of Flight Environment Based on OpenGL

Ruixian Li

School of Transportation and Vehicle Engineering, Shandong University of Technology, Zibo 255049, China

E-mail: lrx@sdut.edu.cn

**Abstract**

According to the flight dynamic mathematical model and the kinematic mathematical model, this article adopts the Visual C++6.0 visualization development environment and the OpenGL programming technology to simulate the flight rule of the aircraft. The flight environment virtual simulation system can simulate the flight rules in 3D space and users can control the 3D aircraft real time by the keyboards, and the main parameters of the aircraft can dynamically change with it, and the interface of the software is friendly and easy to operate.

**Keywords:** VC++, OpenGL, Simulation

## 1. Introduction

The visual simulation of flight environment is the representative man-in-the-loop real-time simulation system, and it is a sort of practical application of the visual reality technology. It adopts the computer-graphics technology and the multimedia technology to establish the dynamical and kinematic mathematical models and evaluate the whole flight quality of the aircraft. The visual simulation of flight environment is mainly used to analyze the study the flight control, guidance and navigation system of the aircraft, which include the confirmations of the control rule, guidance rule and navigation mode, the selection of the parameters, the matching of the system, the analysis and the evaluations about the dynamic characters, flight quality, guidance and navigation precisions of the aircraft. In addition, the visual simulation of flight environment also could be used in flight training and employee training.

Because the visual simulation of fight environment possesses many advantages such as cheap costs, reliable technology, flexibility and convenience which can not be unexampled by general technologies, it has been increasingly emphasized in domestic and foreign markets, and the research and application domains about relative technologies are continually extended from initial single-flat simulation such as flight performance and aviation electron to present simulator network and distributed interactive simulation, from initial simulated flight training to present product research project argumentation, design analysis, production manufacturing, experiment evaluation and maintenance training. As a sort of new technology, the visual simulation of flight environment is gradually going to be mature.

The visual simulation of flight environment is a complex system, and it needs a great lot modeling works of subjects. Taking Visual C++6.0 and OpenGL as the development tools, this article will simulate the scene of the flight process according to the flight kinematics, and users can use the keyboard to control the aircraft.

## 2. Introduction of OpenGL

OpenGL is a sort of interface about graphics and hardware in fact, and it includes numerous graphic operation functions which can be utilized to establish 3D models to implement real-time interaction by developers. Different with other graphic program designs, it offers very clear functions, and it is a software package with high performances. In addition, the powerful graphic function of OpenGL doesn't require developers to write the 3D object model as fixed data format, and it can utilize the data sources with different formats flexibly.

The plotting mode of OpenGL is different with the plotting mode of Windows, and it uses the special pixel format and the rendering contexts to plot the graphics. And the plotting process can be described as follows.

(1) Over load the object model by the geometric primitive and establish the mathematical description of the object.

(2) Select proper position to place the object in the 3D space and confirm the optimal viewpoint to see the scene.

(3) Compute the color of the object in the scene.

(4) Transform the mathematical description of the object model in the scene into the pixel points in the screen, and this process is called as the rasterization.

The plotting flow is seen in Figure 1.

OpenGL could produce the 3D scene with true color which includes simple 2D object plotting and interactive 3D animation scenes with different effect and it can help developers to complete these plotting works higher effectively.

## 3. OpenGL Plotting

### *3.1 Establishing 3D scene*

Before plotting, we should implement the initialization operation. And the usual clear commands include followings.

(1) Void glClearColor (GLfloat rad, GLfloat green, GLfloat blue, GLfloat alpha). It is used to confirm the clearing value of the color buffer.

(2) Void glClearDepth (GLfloat depth). It is used to appoint the clearing value of the deep buffer.

(3) Void glClear (GLbitfield mask). It is used to clear one sort or several sorts of buffer in the vision region.

In OpenGL, all geometric objects should be described by many ordered peak sets, i.e. OpenGL adopts ordered peak sets to establish the geometric primitives. All defined peaks in one geometric primitive are placed between the function glBegin() and the function glEnd(), and the relationships among these peaks are defined.

We can also appoint the color, normal, texture coordinate of the peak between the function glBegin() and the function glEnd(), which only needs to transfer relative functions before defining the space position of the peak.

But it is very difficult to plot complex 3D object by above functions, and the convenient method is to utilize existing plotting tools to plot the model and use proper transformation software to transform the model into the files which are supported by the language of C++ or directly compile the read-in programs, and then obtain relative data of the object model.

### *3.2 Coordinates transformation*

The coordinate transformation includes modeling transformation, view transformation, projection transformation and viewport transformation. Two current transformation orders include the function glMatrixMode (Glenum mode) which is used to appoint the present matrix and the function Void glLoadIdentity which is used to endow the present matrix to the unit matrix.

The modeling transformation can change the size, position and direction of the object in the scene, and it includes following three OpenGL orders.

(1) glTranslate{fd}(TYPE x, TYPE y, TYPE z). This order is used for the matrix translation which moves the origin of the coordinate system to the point (x, y, z).

(2) glRotate{fd}(TYPE angle, TYPE x, type y, TYPE z). It is used in matrix rotation, i.e. the negative vector rotation surrounding the origin, where the parameter "angle" appoints the rotation angle, and the parameter "x, y, z" appoints the vector from the origin to the point (x, y, z), and the vector is the rotation axis and the rotation direction is the negative direction.

(3) glScale{fd}(TYPE x, TYPE y, TYPE z). This order is used to change the actual size of the object.

The view transformation changes the position and the direction of the viewpoint, and it likes the fixed tripod for the camera to appoint the lens to the scene needed. Here, we only introduce the function gluLookAt() which is offered by the utility application library, i.e. "void gluLookAt (GLdouble eyex, GLdouble eyey, GLdouble eyez, GLdouble center, GLdouble centery, GLdouble centerz, GLdouble upx, GLdouble upy, GLdouble upz)". The parameters in the function are respectively to appoint the position of the viewpoint, the position of the reference point and the upward direction of the direction.

The projection transformation is used to define the body of the view, and the remnant parts are cut. The projection transformation includes two sorts, i.e. the perspective projection and the orthogonal projection. The most obvious character of the perspective project is to zoom according to the perspective rate. When the object is more near to the camera, the imaging is larger. The usual function of the perspective projection is Void gluPerspective (GLdouble fovy, GLgouble Aspect, GLdouble zNear, GLdouble zFar), where, Fovy denots the angle of the vision region in the direction of Y, Aspect is the appointed width and height rate, Znear is the distance from the appointed viewpoint to the near cutting face, and Zfar is the distance from the viewpoint to the far cutting face.

The viewport transformation is used to appoint the size of the viewport, and its order is "void glViewport (Glint x, GLinty, GLsizei width, GLsizei height)", where, "x, y" appoints the bottom left corner of the viewport, and "width" and "height" respectively appoint the width and the height of the viewport. The initial window value is "(0, 0, Win Width, Win Height)" by default, where the Win Width and the Win Height denote the size of the window.

There is still an important operation, i.e. the stack operation. And it is a sort of mechanism to implant the saving and recovering of the matrix, and it utilizes the stack operation order "glPushMatrix()" to save the present matrix on the top of the stack, and uses "glPopMaterix()" to move the matrix on the top of the stack and recover the original matrix.

### *3.3 Texture management*

The texture projection is one important part of the third dimension graphic making, and it can be utilized to more conveniently make graphics with more third dimension when we need not expend too much time to consider the surface details of the object. In OpenGL, the basic approaches of the texture projection includes following aspects.

3.3.1 Texture definition

We can transfer the following function to define the 2D texture projection in programs.

void glTexImage2D (GLenum target, GLint level, GLint components, GLsizei width, Glsizi height, GLint boder, Glenum format, GLenum type, Const GLvoid*pixels)

3.3.2 Texture control

The texture control of OpenGL is to define how the texture packs the surface of the object because the appearance of the texture will not always be identical with the shape of the object. The function controlling the texture in OpenGL is "void glTexParameter {if}[v] (GLenum target, GLenum pname, TYPE param)", and the parameter of "target" is to confirm whether the control parameters are used in 1D or 2D texture, and the parameter of "pname" and the parameter of "param" are used to zoom in and zoon out the filters.

3.3.3 Mapping mode

The function in OpenGL to control the mode of texture projection is "void glTexEnv{if}[v] (GLenum target, GLenum pname, TYPE param)".

3.3.4 Texture coordinates

When plotting the scene of the texture projection, we should not only give the geometric coordinates of each peak, but define the texture coordinates, and through several times transformation, the geometric coordinates of the peak is in the plotting of the screen, but the confirmation of the texture coordinates is decided by which element is endowed to the peak. The texture image is the form of square array, and the texture coordinates can usually be defined by the form of 1D, 2D, 3D or 4D, which are respectively called as s coordinate, t coordinate, r coordinate and q coordinate. The function to set up the present texture coordinate in OpenGL is "void glTexCoord {1 2 3 4} {s I f d} [v] (TYPE coord)".

The OpenGL provides the function which can automatically produce the texture coordinates, i.e. "void glTexGen{i}[v] (GLenum coord, GLenum pname, TYPE param)".

*3.4 Frame buffer and animation*

3.4.1 Composing of frame buffer

The graphics plotted on the screen are composed by pixels, and every pixel has one fixed color or other information such as depth of the corresponding point. When plotting, the memory must save data for each pixel, and the memory region which saves data for all pixels is called as the buffer. All buffers in the system are called as the frame buffer, and these different buffers can be used to set up color, eliminate hidden face, anti-alias scene and stencil scene.

OpenGL is composed by four sorts of buffers such as color buffer, depth buffer, stencil buffer and accumulation buffer.

3.4.2 Buffer clearing

The process that OpenGL clears buffer includes giving the clearing value of every buffer which is wrote in, executing the operation by the single function order, transferring the buffer table which should be cleared, and judging whether the hardware can clear simultaneously, or else, executing every operation in turn.

The following functions set up clearing values for every buffer.

void glClearColor (GLclampf red, GLclampf green, GJclampf blue, GKclampf alpha)

void glClearIndex (GLfloat index)

void glClearDepth (GLclampd depth)

void glClearStencil (GKint s)

void glClearAccum (GKfloat red, GLfloat green, GLfolat blue, GLfloat alpha)

Above functions respectively provide present clearing values for four sorts of buffer under the mode of RGBA.

After selecting the clearing buffers and their clearing values, we can transfer glClear() to complete the operation of clearing. The clearing function is "void glClear (Glbitfield mask)".

3.4.3 OpenGL animation

The dual-buffer is offered by OpenGL, and it can be used to make animation, i.e. when displaying a frame of graphic in the foreground buffer contents, the background buffer is plotting the next frame of graphic, and when the plotting is completed, and contents of background buffer will be display on the screen, but the foreground buffer is plotting the next frame of graphic. In that way back and forth, the screen will always display the plotted graphic, so all graphics seems continually. In OpenGL, the function of "void auxSwapBuffers (void)" is used to set up the interactive buffers, i.e. when executing once plotting process, the foreground buffer and the background will change to plot the next frame of graphic back the screen.

**4. Modeling of 3D scene in the flight environment**

*4.1 Simulations of 3D landform*

The simulation of 3D landform is one of the most important technologies in the development of visualization system, and the landform simulation includes two sorts such as the real landform and the simulation landform.

The real landform is the reappearance of the real landform in the real world, and it must adopt concrete data to build. In that situation, the method of the digital elevation model is usually adopted, and the data structure of the method is complex, but its precision is very high, and the graphic creation speed is slow.

When we only emphasize the requirement of sense without considering the landform creation in the visualization process, we can adopt the simulation landform which usually includes the random elevation data creation and the fractal elevation data creation. Though the landform created by these methods is very visual and the graphic creation speed is very quick, but it can not correspond with the real world. In the system, we adopt the method of the fractal elevation data creation.

*4.2 Simulations of sky and cloud*

The sky is only to strengthen the visual effect. The main sceneries in the sky are clouds, but the shapes of cloud include infinite details. The usual method is the method of box (implementing texturing in the plotting polygons) or the method of roundness (plotting multiple peaks, texture projections and pulverizations to make them even). In the article, we adopt the method of box to simulate the sky through the texture plotting of the cloud graphics in the quadrangle and transferring glTranslate() and glRotate() to realize the translation and rotation of the texture coordinates.

```
……
glColor4f (1,1,1,1)
glBindTexture (GL_TEXTURE_2D, texture[1])    //establish an appointed texture
glTranslatef (-xtrans,-ytrans-MAX*48,-ztrans)
glRotatef (90,1,0,1)
gluSphere (quadratic,MAX*50,20,20)
```

*4.3 Simulation of 3D aircraft*

First, establish the geometric model of the aircraft. Because the structure of the aircraft is complex, so in the article, we adopt the plotting software of MS 3D (MilkShape 3D) to establish the model of the aircraft, and compile the model read-in program to read the 3D file into the application. The MS 3D is a very cabinet 3D making tool, and it has better model compress packets which include the introduction of file formats that can easily analyze the read the model.

(1) To plot complex graphic by MS 3D is to synthesize some simple geometric models through link, extrusion, cover and other measures. So a simple aircraft model must be completed by the synthesis of many simple models.

(2) Compiling the mode read-in program (or using the transformation software). First, we should establish a corresponding model data structure with the file format of MilkShape 3D, then save the data of MilkShape 3D into the data structure. In the programs, first define one class of Model, and derive a new class of “MilkShapeModel” from the class of Model. The concrete approaches include the definition of model data structure and the read of MilkShap 3D data. Because the read-in programs of most models are almost same and the compiling methods are fixed, so we don’t give unnecessary details any more.

The real-time requirement which reduce the transect points when the precision is not high in the simulation process can largely reduce the operation consumption and enhance the running speed. So in the modeling, we should try to optimize the model structure and the pixel size and color of the maps.

(3) Establishing the aircraft pose model. In the simulation process of the aircraft, we can compute various sorts of data and implement relative codes design according to different situations of the simulation.

**5. Keyboard control**

When simulating the flight dynamics system, we add the human-computation interaction interface and users can control the aircraft by the keyboard.

*5.1 Response of keyboard information*

Keyboard is one of main input equipments of the computer, and users can input data through the keyboard. When user presses certain key on the keyboard, the keyboard information will produce. There are three pieces of keyboard information in common use.

(1) WM_KEYDOWN. It is the information produced when the key is pressed.

(2) WM_KEYUP. It is the information produced when the key is loosened.

(3) WM_CHAR. It is the character information.

The keys on the keyboard can be divided into the system keys and the non-system keys. All non-system keys will produce WM_KEYDOWN and WM_KEYUP, and the WM_CHAE are not produced by all non-system keys.

### *5.2 Key-press response function*

In the article, we design the press of 0-9 to control the flight speed of the aircraft, and the presses of the direction keys are used to control the pitching and yawing of the aircraft, and the key of “W” to decide whether displaying the scene by adopting the line frame mode. And we design the concrete codes.

## 6. Software result test

The software test is the important part of the software development, and it can help us obtain the dynamic performance, flight parameters and other important information about the aircraft. Users can control the 3D aircraft through the keyboard, and main parameters can dynamically be displayed with it. The performance test results of the software are seen in Figure 2 to Figure 5.

## 7. Conclusions

Under many situations, the flight environment visual simulation based on OpenGL is universally applied in many subsystem simulations about the flight environment such as flight performance, aviation electron and other single platform simulation. And in the development tools, because of powerful function and flexible applications, Visual C++6.0 and OpenGL with the good supports of the Microsoft system are the first choice for foreign and domestic relative industries.

## References

Chen, Yantao. (2001). *Graphics Program Design Guide of OpenGL.* Beijing: China Waterpower Press.

Li, Ying. (2002). *Examples of OpenGL Technical Application.* Beijing: National Defense Industry Press.

Qiaolin. (2000). *Program Design of OpenGL.* Beijing: Tsinghua University Press.

Wang, Lanbo. (1999). *Easy Study of Visual C++6.0.* Beijing: Electronic Industry Press.

Wang, Xingren. (1998). *Flight Real-time Simulation System and Technology.* Beijing: Beijing Aeronautical and Astronautical University Press.

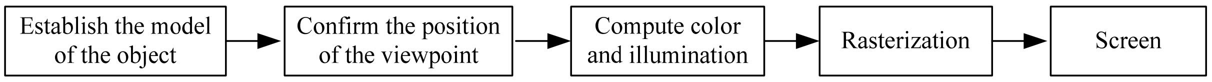
Establish the model of the object
Confirm the position of the viewpoint
Compute color and illumination
Rasterization
Screen

Figure 1. Graphics Plotting Flow

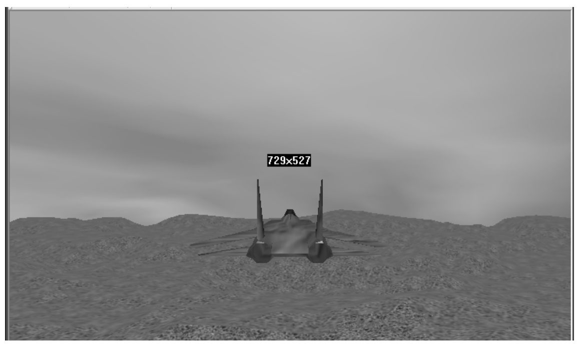

Figure 2. Real-time Flight of Aircraft

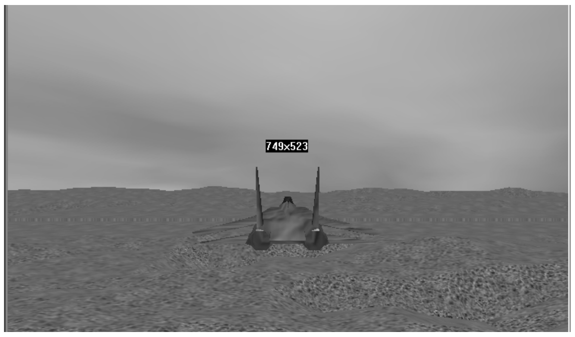

Figure 3. Adjusting the Flight Speed by 0-9 Keys

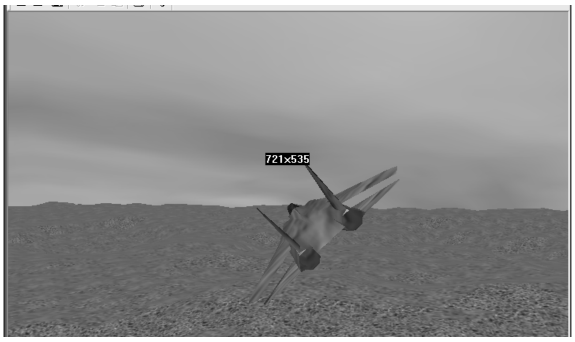

Figure 4. Controlling the Pitching of Aircraft by the Directional Keys

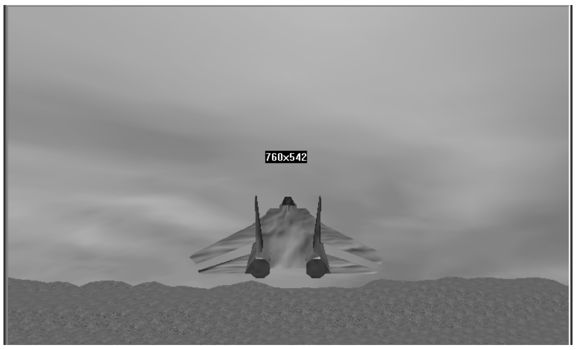

Figure 5. Controlling the Yawing of Aircraft by the Directional Keys