浙江大学实验报告

课程名称：____计算机图形学___ 指导老师：_____成绩：_____
实验名称：____OpenGL 显示列表_____ 实验类型：_____基础实验_____同组学生姓名：_____

装

订

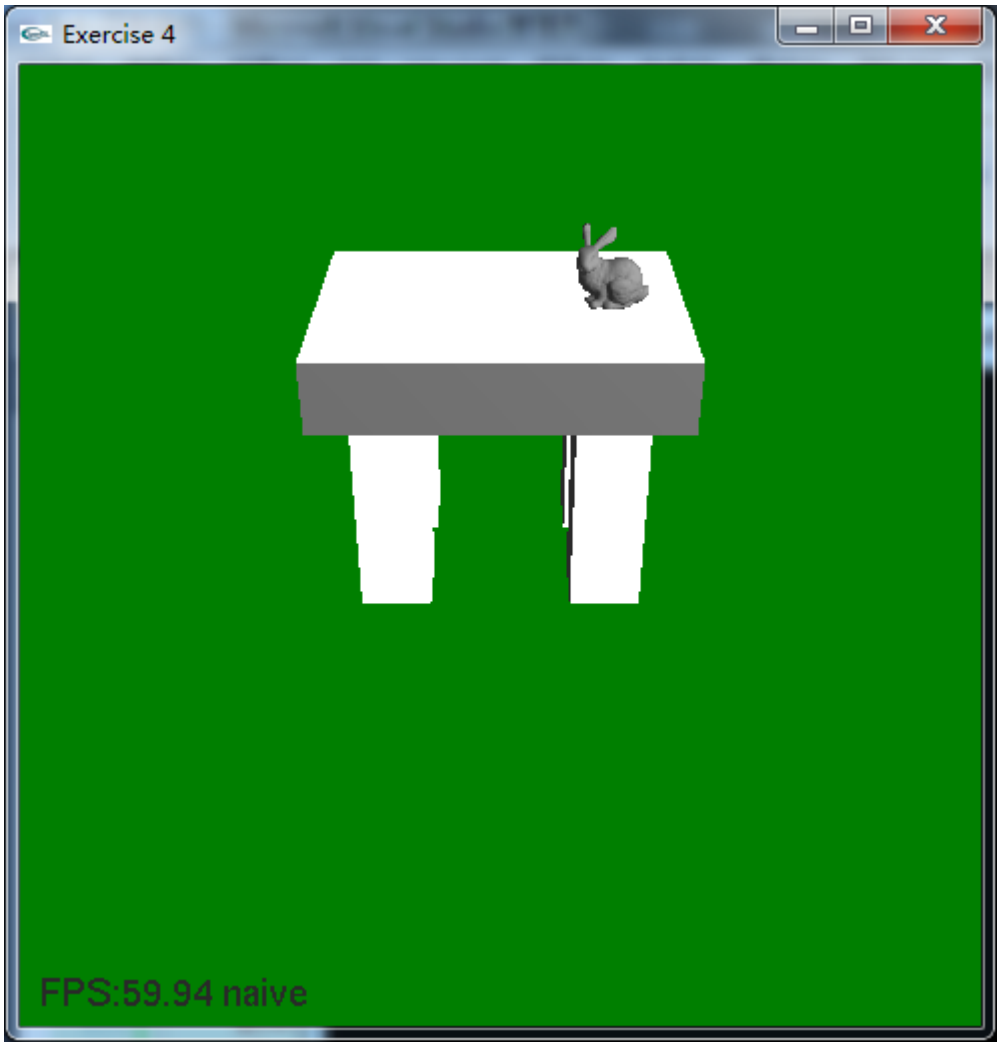
线

一、实验目的和要求

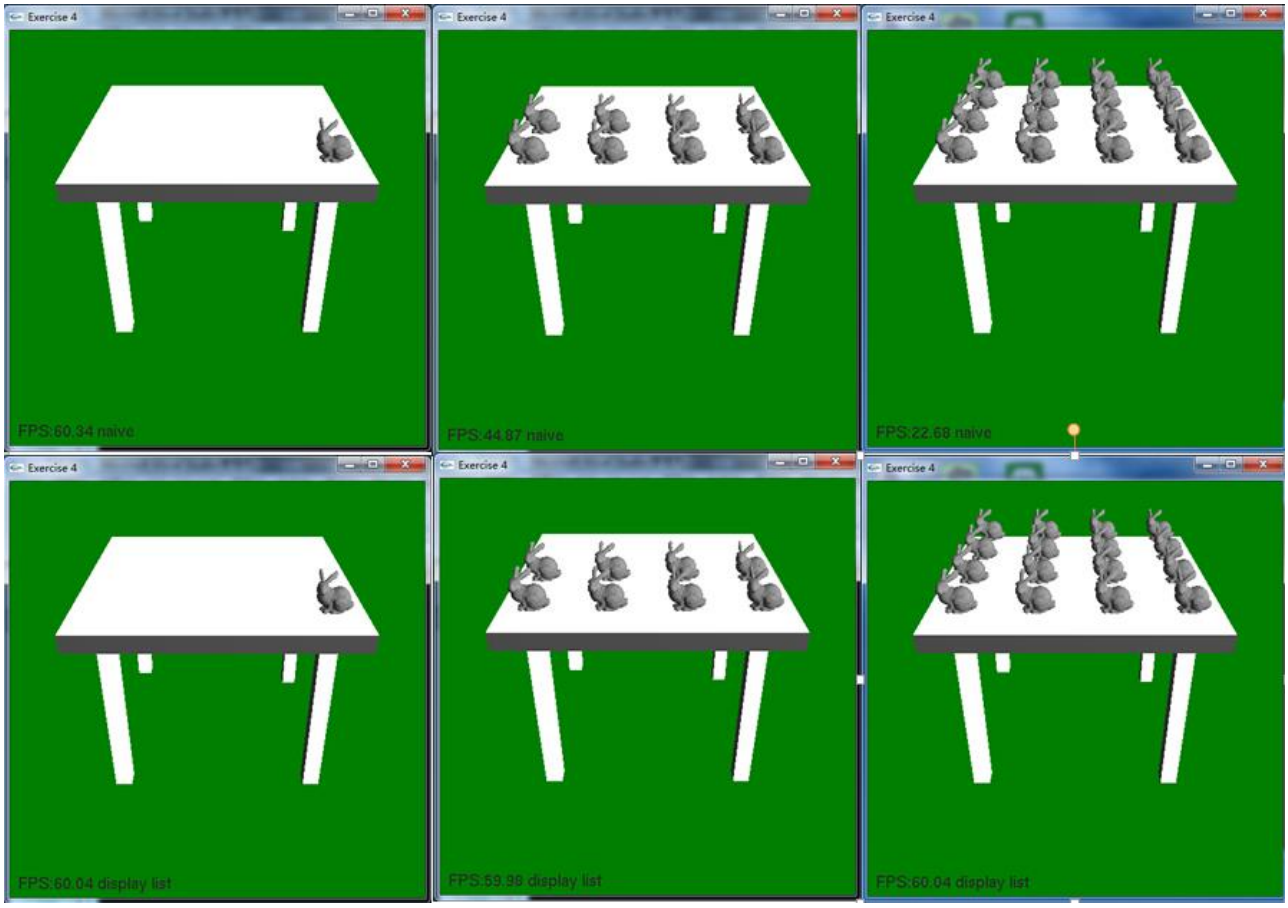
在三维观察实验的基础上，通过实现下述实验内容，掌握 OpenGL 中显示列表的作用和使用方法。

二、实验内容和原理

使用 Visual Studio C++编译已有项目工程，



修改代码，通过**键盘按键**，控制兔子的数量（1 至 16 个）以及整个场景的渲染模式，生成以下图形：



用按键 I、K 添加兔子数量增减（所有兔子均摆放着在桌面上，兔子间不要有交叉，桌面不够大可自行调整），按键 L 来切换显示列表和非显示列表绘制方式。WASDZC 控制上下左右前后移动,空格键控制整体旋转。

通过动画以及对 FPS 的理解和分析显示列表对程序绘制性能的影响。

三、主要仪器设备

Visual Studio C++

glut.zip

Ex4-vs2010 工程

四、实验原理

1. 显示列表

OpenGL 在即时模式（Immediate Mode）下绘图时，程序中每条语句产生的图形对象被直接送进绘图流水线，在显示终端立即绘制出来。当需要在程序中多次绘制同一个复杂的图像对象时，这种即时模式会消耗大量的系统资源，降低程序的运行效率。而显示列表是一种更有效组织 OpenGL 语句的形式，OpenGL 显示列表（Display List）是由一组预先存储起来的留待以后调用的 OpenGL 函数语句组成的，当调用这张显示列表时就依次执行表中所列出的函数语句。显示列表是最快的一种绘制静态数据的方式，因为顶点数据和 OpenGL 命令被缓冲在

服务器端的显示列表中，这样减少了从客户端到服务器段的数据传输。当一个显示列表被创建之后创建显示列表以 glNewList 开始，以 glEndList 结束。通过 glCallList 可以调用显示列表。其缺点在于当一个显示列表被编译后，它不能被改变。

2.FPS

FPS 指每秒传输帧数，是渲染效率的一种衡量，其大小反映绘制的流畅性，反映了整个程序在当前的一个渲染状态下平均每秒所能容纳的“渲染循环”执行次数，也表征了平均每个“渲染循环”所用的时间。它的大小可以直观反映绘制的流畅性。我们可以调用 glutGet(GLUT_ELAPSED_TIME) 函数来计算 FPS 。该函数返回两次调用 glutGet(GLUT_ELAPSED_TIME)的时间间隔,单位为毫秒。通过得到两次调用的间隔时间可以计算绘制图像的刷新帧率。在本程序中用 frame 变量存储帧数，当两次间隔调用时间差大于 1000ms 时用 FPS = 帧数/时间更新 FPS 的值。

3.位图显示

FPS 的显示可以通过 glutBitmapCharacter(GLUT_BITMAP_HELVETICA_18,*c)把字符变成位图显示在窗口中。该函数用于在 glut 窗口某位置显示字符。由于其是 glut 内置函数，功能有所限制，只能显示英文字符，而且字体、大小都不能任意设置。

五、实验过程分析

1.显示列表

（1）分配显示列表编号，我们用到的函数为 GLuint glGenLists (GLsizei range)，其中参数 range 指定要分配几个显示列表，这里我们用到的是 GLint lid = glGenLists(2)，即分配了两个空显示列表。创建显示列表，我们用到的函数是 void glNewList (GLuint list, GLenum mode)，其中第一个参数标示当前正在操作的显示列表号，第二个参数有两种取值--GL COMPILE 和 GL COMPILE AND EXECUTE，前者声明当前显示列表只是装入相应 OpenGL 语句，不执行；后者表示在装入的同时，执行一遍当前显示列表，这里我们用到的是 glNewList(lid, GL_COMPILE)和 glNewList(lid + 1, GL_COMPILE)。结束显示列表，这里我们用到的是 void glEndLists()函数。

创建显示列表的相关源代码如下：

```
1.  GLint GenTableList()
2.  {
3.      GLint lid = glGenLists(2);  //  Generate two empty display lists
4.
5.      //  Draw the table
6.      glNewList(lid, GL_COMPILE);  //Used to create and replace a display list function pro
    totype. Specify the name of the display list and the compile mode is compile only. Store
    first, not execute
7.      DrawTable();
8.      glEndList();
```

```
9.
10.     //  Draw the rabbits
11.     glNewList(lid + 1, GL_COMPILE);
12.     DrawBunny();
13.     glEndList();
14.
15.     return lid;  //  Return the display list number
16. }
```

（2）调用显示列表需要用到的函数为 glCallList(id)，入参 id 表示了要调用的显示列表的编号。另外也可以使用 glCallLists 一次性调用一组显示列表。这里我们用到的是 glCallList(rabbitList)和 glCallList(tableList)。

（3）使用显示列表绘制桌子和兔子的相关源代码如下：。

```
1.  void Draw_Table_List()  //  Draw the scene( new way )
2.  {
3.      glPushMatrix();
4.
5.      //  Translate and scale
6.      glTranslatef(2.0, 4.5, 1.5);
7.      glScalef(2, 2, 2);
8.
9.      //  Draw the rabbits
10.     for (int i = 1; i <= rabbitNumber; i++)
11.     {
12.         glCallList(rabbitList);//  If it is a list mode, it will be called by callList.
13.         if (i % 4 == 0)  //  If it is needed to wrap here
14.             glTranslatef(2.0f, 0.0f, -0.5f);
15.         else
16.             glTranslatef(-0.66f, 0.0f, 0.0f);
17.     }
18.
19.     glPopMatrix();
20.
21.     glCallList(tableList);  //  Call the display list to draw the table
22. }
```

2. 兔子数量和位置的改变

（1）兔子数量

兔子的数量在 1-16 之间，兔子数量不能小于 1 或者大于 16，对此需要进行控制，与其相关的代码如下：

```
case'i':
{
    if (rabbitNumber < 16) rabbitNumber++;  //  When the number of rabbits is less than 16, the number of rabbits increases
    break;
}
case'k':
{
    if (rabbitNumber > 1) rabbitNumber--;  //  When the number of rabbits is greater than 1, the number of rabbits is reduced
    break;
}
```

（2）兔子位置

每一行的兔子只能有四个，每行兔子与兔子之间的间隔相同，同样的，每列兔子与兔子之间的间隔也相同。并且当兔子数量增长到 5、9、13 个时需要换行，换行有关的代码如下：

即时模式：

```
for (int i = 1; i <= rabbitNumber; i++)
{
    DrawBunny();  //  Draw rabbit directly
    if (i % 4 == 0)  //  If it is needed to wrap here
        glTranslatef(2.0f, 0.0f, -0.5f);
    else
        glTranslatef(-0.66f, 0.0f, 0.0f);
}
```

显示列表：

```
for (int i = 1; i <= rabbitNumber; i++)
{
    glCallList(rabbitList);//  If it is a list mode, it will be called by callList.
    if (i % 4 == 0)  //  If it is needed to wrap here
        glTranslatef(2.0f, 0.0f, -0.5f);
    else
        glTranslatef(-0.66f, 0.0f, 0.0f);
}
```

3. FPS

在这里调用 glutGet(GLUT_ELAPSED_TIME)函数来计算 FPS。该函数返回两次调用glutGet(GLUT_ELAPSED_TIME)的时间间隔,单位为毫秒。通过得到两次调用的间隔时间可以计算绘制图像的刷新帧率。在本程序中用 frame 变量存储帧数，当两次间隔调用时间差大于1000ms 时用 FPS = 帧数/时间更新 FPS 的值。

部分相关代码如下：

```
frame++;
time = glutGet(GLUT_ELAPSED_TIME);

//  Returns the time interval in which glutGet(GLUT_ELAPSED_TIME) is called twice in mill
iseconds
if (time - timebase > 1000) {  //  When the time interval is greater than 1000ms
    sprintf_s(buffer, "FPS:%4.2f %s",
        frame*1000.0 / (time - timebase), mode);  //  Write into buffer
    timebase = time;  //  Last time's time interval
    frame = 0;
}
```

4. 位图显示

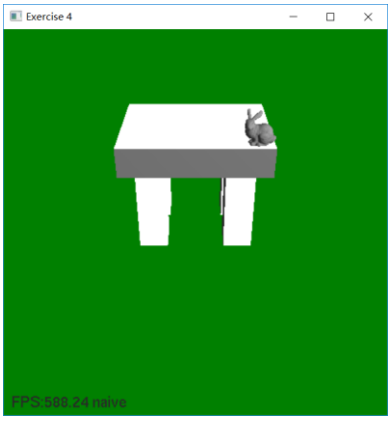
FPS 的显示可以通过 glutBitmapCharacter(GLUT_BITMAP_HELVETICA_18,*c)把字符变成位图显示在窗口中。该函数用于在 glut 窗口某位置显示字符。

相关代码如下：

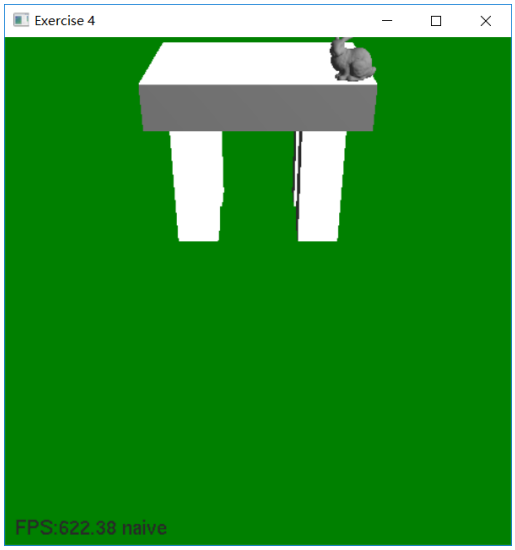
```
for (c = buffer; *c != '\0'; c++) {
    glutBitmapCharacter(GLUT_BITMAP_HELVETICA_18, *c);
}
```

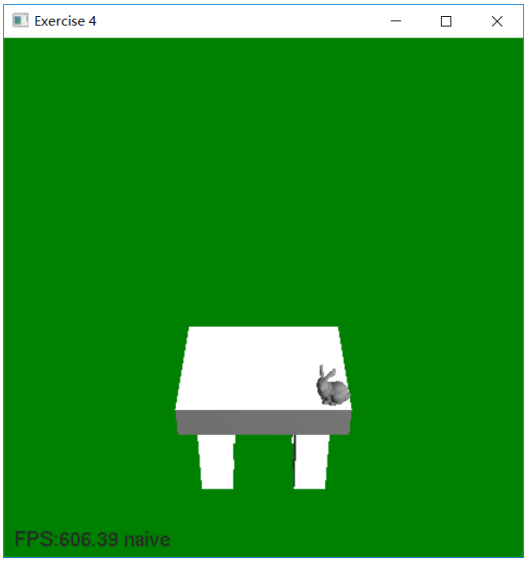
六、实验结果与分析

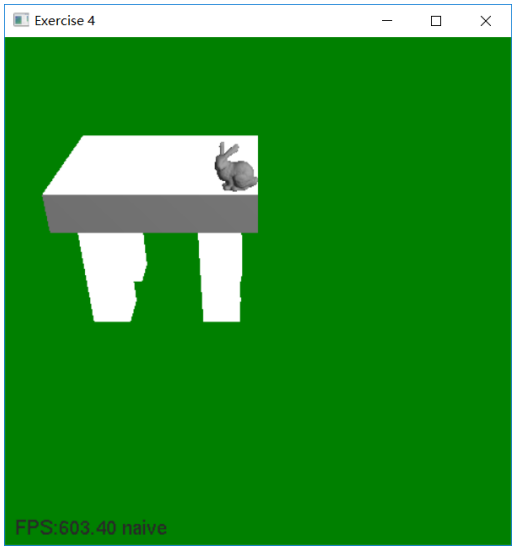
1. 桌子和兔子的原始位置图像：

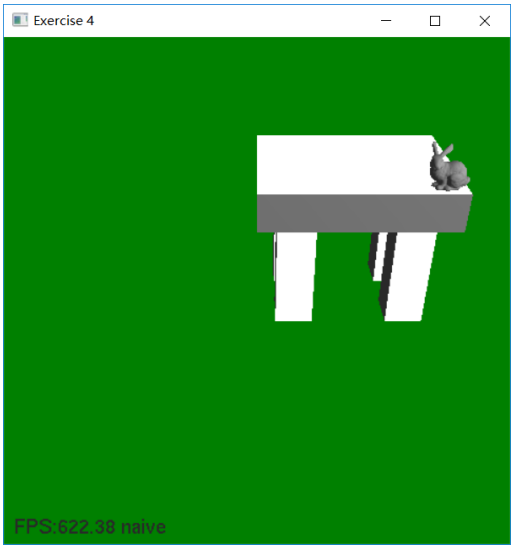


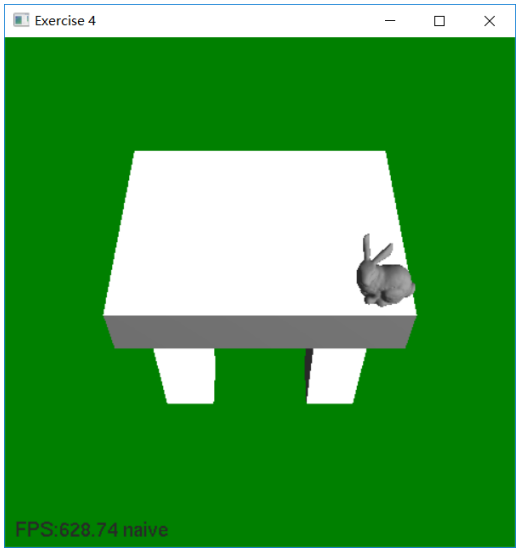
2 上下左右前后移动后的位置图像

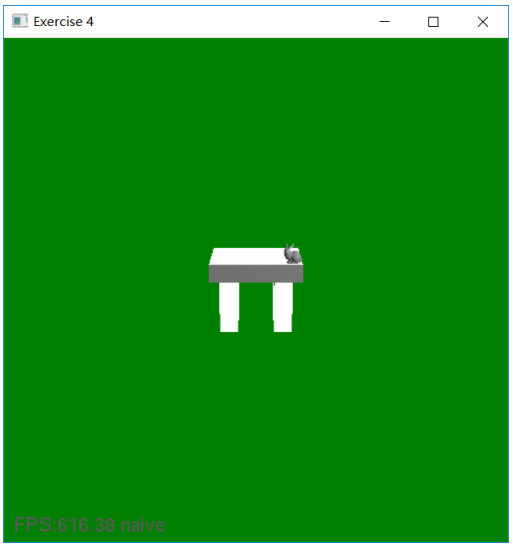




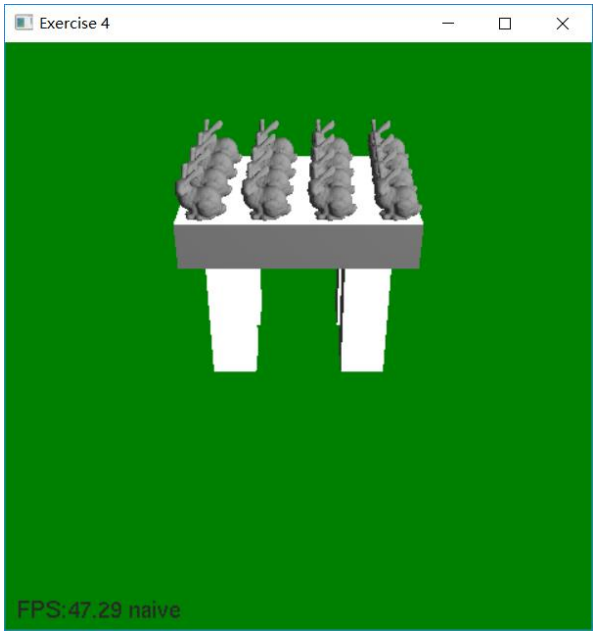


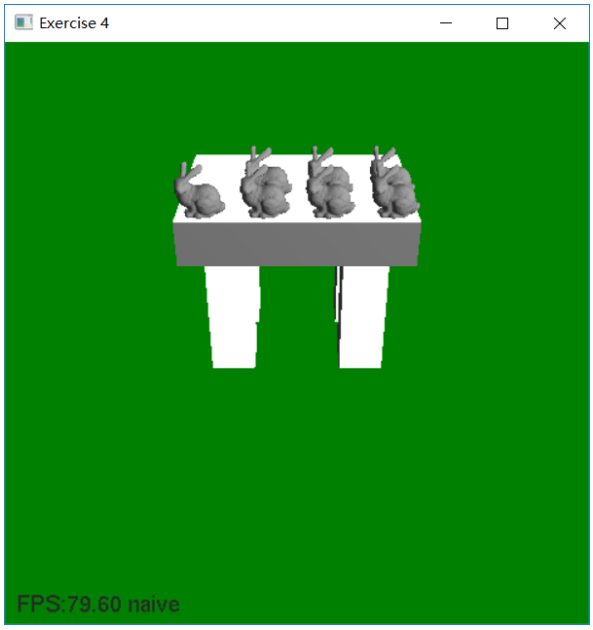






3.兔子增多与减少



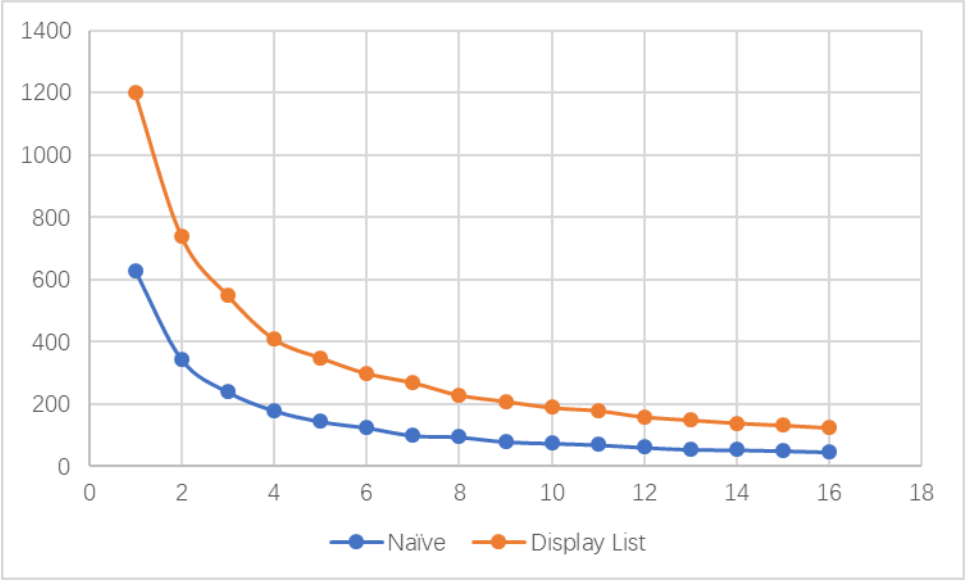


4.兔子旋转

5.两种模式切换

6.FPS 与兔子数量的关系

兔子数量	Naïve	Display List
1	630	1200
2	345	740
3	240	550
4	180	410
5	145	350
6	125	300
7	100	270
8	95	230
9	80	210
10	75	190
11	70	180
12	62	160
13	55	150
14	54	140
15	51	133
16	47	125



当兔子的数量改变时，可以记录下 FPS 的近似值，将数据绘制折线图，可以看出随着兔子数量的增加，FPS 减小。显示列表模式下，FPS 一直比 Naive 模式要高，每秒刷新的帧率更高，说明使用显示列表比直接绘制更加高效。

七、源代码

```
#include <stdlib.h>
#include "GL/glut.h"
#include <stdio.h>
#include <string.h>

#include "stanford_bunny.h"

float eye[] = { 0, 4, 6 }; //  the place of the eyes
float center[] = { 0, 0, 0 };  //  the place of the viewpoint
float fDistance = 0.2f;  //  distance factor
float fRotate = 0;  //  rotation factor
bool bAnim = false;  //  whether to rotate

bool bDrawList = false;  //  Whether to use the display list
GLint tableList = 0;  //  table list
GLint rabbitList = 0;  //  rabbit list
int rabbitNumber = 1;  // the number of rabbits

void Draw_Leg() // Draw the leg
{
    glScalef(1, 3, 1);  // Stretch the model three times in the y direction
    glutSolidCube(1.0);  // Draw a cube with a side length of one
}

void DrawTable()  //  Draw the table
{
```

```
27.     // Draw the cube
28.     glPushMatrix();
29.     glTranslatef(0, 3.5, 0);
30.     glScalef(5, 1, 4);  //  Stretch the model five times in the x direction and four time
    s in the z direction
31.     glutSolidCube(1.0);  //  Draw a cube with a side length of one
32.     glPopMatrix();
33.
34.     //  Draw the four legs
35.     glPushMatrix();
36.     glTranslatef(1.5, 1.5, 1);  //  Move the leg to its position
37.     Draw_Leg();  // Draw a leg
38.     glPopMatrix();
39.
40.     glPushMatrix();
41.     glTranslatef(-1.5, 1.5, 1);  //  Move the leg to its position
42.     Draw_Leg();  // Draw a leg
43.     glPopMatrix();
44.
45.     glPushMatrix();
46.     glTranslatef(1.5, 1.5, -1);  //  Move the leg to its position
47.     Draw_Leg();  // Draw a leg
48.     glPopMatrix();
49.
50.     glPushMatrix();
51.     glTranslatef(-1.5, 1.5, -1);  //  Move the leg to its position
52.     Draw_Leg();  // Draw a leg
53.     glPopMatrix();
54. }
55.
56.
57. GLint GenTableList()
58. {
59.     GLint lid = glGenLists(2);  //  Generate two empty display lists
60.
61.     //  Draw the table
62.     glNewList(lid, GL_COMPILE);  //Used to create and replace a display list function pro
    totype. Specify the name of the display list and the compile mode is compile only. Store
    first, not execute
63.     DrawTable();
64.     glEndList();
65.
66.     //  Draw the rabbits
67.     glNewList(lid + 1, GL_COMPILE);
```

```
68.     DrawBunny();
69.     glEndList();
70.
71.     return lid;  //  Return the display list number
72. }
73.
74. void Draw_Table_List()  //  Draw the scene( new way )
75. {
76.     glPushMatrix();
77.
78.     //  Translate and scale
79.     glTranslatef(2.0, 4.5, 1.5);
80.     glScalef(2, 2, 2);
81.
82.     //  Draw the rabbits
83.     for (int i = 1; i <= rabbitNumber; i++)
84.     {
85.         glCallList(rabbitList);//  If it is a list mode, it will be called by callList.
86.         if (i % 4 == 0)  //  If it is needed to wrap here
87.             glTranslatef(2.0f, 0.0f, -0.5f);
88.         else
89.             glTranslatef(-0.66f, 0.0f, 0.0f);
90.     }
91.
92.     glPopMatrix();
93.
94.     glCallList(tableList);  //  Call the display list to draw the table
95. }
96.
97. void DrawScene()  //  Draw the scene( old way )
98. {
99.     glPushMatrix();
100.
101.     //  Translate and scale
102.     glTranslatef(2.0, 4.5, 1.5);
103.     glScalef(2, 2, 2);
104.
105.     // Draw the rabbits
106.     for (int i = 1; i <= rabbitNumber; i++)
107.     {
108.         DrawBunny();  //  Draw rabbit directly
109.         if (i % 4 == 0)  //  If it is needed to wrap here
110.             glTranslatef(2.0f, 0.0f, -0.5f);
111.         else
```

```
112.             glTranslatef(-0.66f, 0.0f, 0.0f);
113.     }
114.     glPopMatrix();
115.
116.     DrawTable();  //  Draw the table directly
117. }
118.
119. void reshape(int width, int height)
120. {
121.     if (height == 0)                                    //  Prevent A divide by zero
122.     {
123.         height = 1;                                     //  Make height equal one
124.     }
125.
126.     glViewport(0, 0, width, height);                    //  Reset the current viewpo
    rt
127.
128.     glMatrixMode(GL_PROJECTION);                        //  Select the projection matrix
129.     glLoadIdentity();                                   //  Reset the projection matrix
130.
131.     float whRatio = (GLfloat)width / (GLfloat)height;
132.     gluPerspective(45, whRatio, 1, 1000);              //  Set projection orientation
133.
134.     glMatrixMode(GL_MODELVIEW);                         //  Select the modelview matrix
135. }
136.
137. void idle()
138. {
139.     glutPostRedisplay();  //  Call the current drawing function
140. }
141.
142. void key(unsigned char k, int x, int y)
143. {
144.     switch (k)
145.     {
146.     case 27:
147.     case 'q': {exit(0); break; }
148.
149.     case 'a':  //  The object moves to the left
150.     {
```

```
151.         eye[0] += fDistance;
152.         center[0] += fDistance;
153.         break;
154.     }
155.     case 'd':  //  The object moves to the right
156.     {
157.         eye[0] -= fDistance;
158.         center[0] -= fDistance;
159.         break;
160.     }
161.     case 'w':  //  The object moves up
162.     {
163.         eye[1] -= fDistance;
164.         center[1] -= fDistance;
165.         break;
166.     }
167.     case 's':  //  The object moves down
168.     {
169.         eye[1] += fDistance;
170.         center[1] += fDistance;
171.         break;
172.     }
173.     case 'z':  //  Move forward
174.     {
175.         eye[2] *= 0.95;
176.         break;
177.     }
178.     case 'c':  //  Move backwards
179.     {
180.         eye[2] *= 1.05;
181.         break;
182.     }
183.     case 'l':  //  Switch display list and non-display list drawing mode
184.     {
185.         bDrawList = !bDrawList;
186.         break;
187.     }
188.     case ' ':  //  Rotate
189.     {
190.         bAnim = !bAnim;
191.         break;
192.     }
193.     case'i':
194.     {
```

```
195.         if (rabbitNumber < 16) rabbitNumber++;
196.         break;
197.     }
198.     case'k':
199.     {
200.         if (rabbitNumber > 1) rabbitNumber--;
201.         break;
202.     }
203.     default: break;
204.     }
205. }
206.
207. void getFPS()
208. {
209.     static int frame = 0, time, timebase = 0;
210.     static char buffer[256];  // String buffer
211.
212.     char mode[64];  //  the mode
213.     if (bDrawList)  //  Whether to draw with the display list
214.         strcpy_s(mode, "display list");
215.     else
216.         strcpy_s(mode, "naive");
217.
218.     frame++;
219.     time = glutGet(GLUT_ELAPSED_TIME);
220.
221.     //  Returns the time interval in which glutGet(GLUT_ELAPSED_TIME) is called twice in
    milliseconds
222.     if (time - timebase > 1000) {  //  When the time interval is greater than 1000ms
223.         sprintf_s(buffer, "FPS:%4.2f %s",
224.             frame*1000.0 / (time - timebase), mode);  //  Write into buffer
225.         timebase = time;  //  Last time's time interval
226.         frame = 0;
227.     }
228.
229.     char *c;
230.     glDisable(GL_DEPTH_TEST);  //  Prohibit depth testing
231.     glMatrixMode(GL_PROJECTION);  //  Select projection matrix
232.     glPushMatrix();               //  Save the original matrix
233.     glLoadIdentity();             //  Load unit matrix
234.     glOrtho(0, 480, 0, 480, -1, 1);    //  Position orthographic projection
235.     glMatrixMode(GL_MODELVIEW);   //  Select Modelview Matrix
236.     glPushMatrix();               //  Save the original matrix
237.     glLoadIdentity();             //  Load unit matrix
```

```
238.     glRasterPos2f(10, 10);
239.     for (c = buffer; *c != '\0'; c++) {
240.         glutBitmapCharacter(GLUT_BITMAP_HELVETICA_18, *c);
241.     }
242.     glMatrixMode(GL_PROJECTION);  //  Select projection matrix
243.     glPopMatrix();                //  Reset to original save matrix
244.     glMatrixMode(GL_MODELVIEW);   //  Select Modelview Matrix
245.     glPopMatrix();                //  Reset to original save matrix
246.     glEnable(GL_DEPTH_TEST);      //  Open depth test
247. }
248.
249. void redraw()
250. {
251.     glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
252.     glClearColor(0, 0.5, 0, 1);
253.     glLoadIdentity();                                   // Reset The Current Modelview M
    atrix
254.
255.     // The place of the camera,
256.     // the place of the object
257.     // and the observation direction
258.     gluLookAt(eye[0], eye[1], eye[2],
259.         center[0], center[1], center[2],
260.         0, 1, 0);
261.
262.     glEnable(GL_DEPTH_TEST);  //  Open depth test
263.     glEnable(GL_LIGHTING);  //  Turn on lighting
264.     GLfloat gray[] = { 0.4, 0.4, 0.4, 1.0 };  //  Set gray
265.     GLfloat light_pos[] = { 10, 10, 10, 1 };  //  Set the light source position
266.     glLightModelfv(GL_LIGHT_MODEL_AMBIENT, gray);  //  Specify the ambient light intensi
    ty of the entire scene
267.     glLightfv(GL_LIGHT0, GL_POSITION, light_pos);  //  Set the illumination position of
    the 0th light source
268.     glLightfv(GL_LIGHT0, GL_AMBIENT, gray);  //  Set the illumination color after the mu
    ltiple reflection of the 0th light source (ambient light color)
269.     glEnable(GL_LIGHT0);  //  Turn on the 0th light source
270.
271.     if (bAnim)
272.         fRotate += 0.5f;  //  Change the rotation factor
273.     glRotatef(fRotate, 0, 1.0f, 0);         // Rotate around Y axis
274.
275.     glScalef(0.4, 0.4, 0.4);  //  Scale
276.     if (!bDrawList)
277.         DrawScene();                        // old way ( ordinary drawing )
```

```
278.     else
279.         Draw_Table_List();                  // new way( display list drawing )
280.
281.     getFPS();  //  Get frames per second
282.     glutSwapBuffers();  //  Swap buffer
283. }
284.
285. int main(int argc, char *argv[])
286. {
287.     glutInit(&argc, argv);  //  Initialize the glut library
288.     glutInitDisplayMode(GLUT_RGBA | GLUT_DEPTH | GLUT_DOUBLE);  //  Specify the window d
    isplay mode that the function glutCreateWindow will create. RGB mode Double buffering
289.     glutInitWindowSize(480, 480);  //  Set the window position, which is the position of
     the top left corner of the window relative to the entire screen
290.     int windowHandle = glutCreateWindow("Exercise 4");  //  Set the window title
291.
292.     glutDisplayFunc(redraw);  //  Register a draw callback function that specifies the f
    unction to call when the window content needs to be redrawn
293.     glutReshapeFunc(reshape);  //  The callback function when the registration window si
    ze changes.
294.     glutKeyboardFunc(key);  //  Register key callback function
295.     glutIdleFunc(idle);  //  Register global callback function : call when idle
296.
297.     tableList = GenTableList();  // initialize the display list
298.     rabbitList = tableList + 1;
299.
300.     glutMainLoop();  //  Glut event processing loop
301.     return 0;
302. }
```