**ОТЧЕТ**

**по лабораторной работе №3**

Выполлнила

Ли Ицзя

3530904.90102

**Цель работы**

Цель задания - отобразить изменение формы объекта, т.е. осуществить преобразование одного трехмерного объекта в другой. Изменение должно быть плавным, пошаговым, предусмотреть не менее 8 шагов морфирования. Задание выполняется при помощи библиотек OpenGL или DirectX

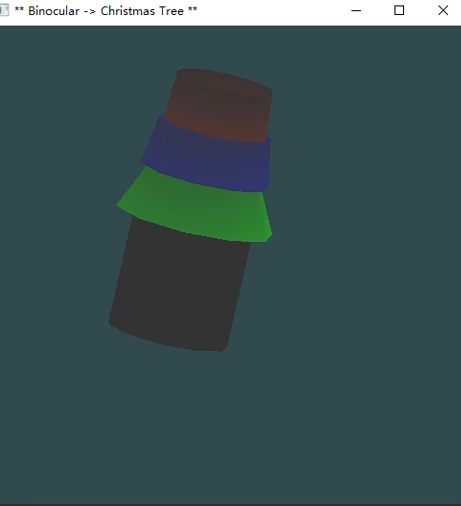
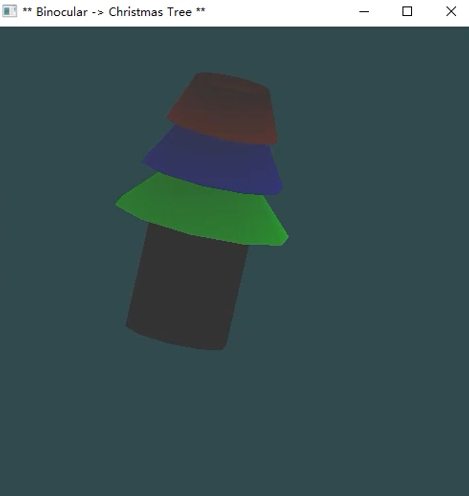
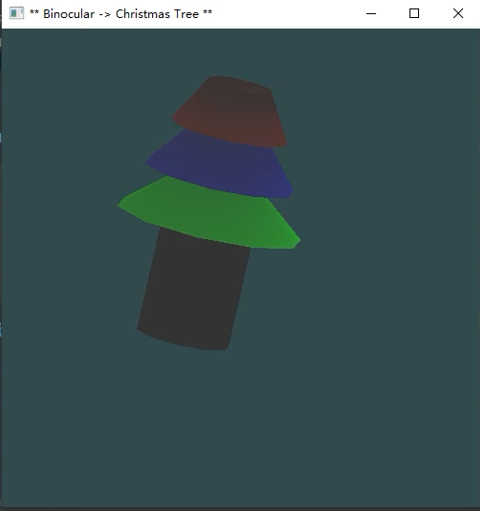
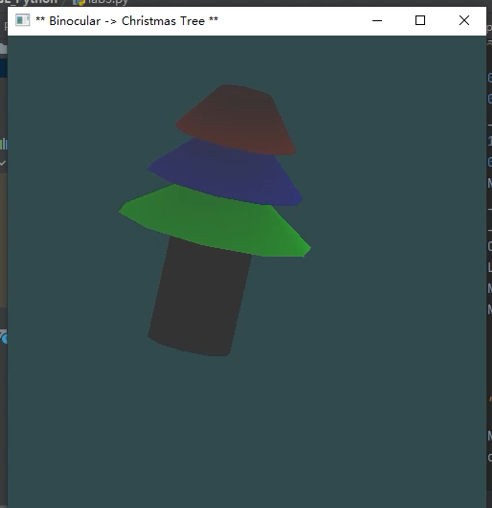
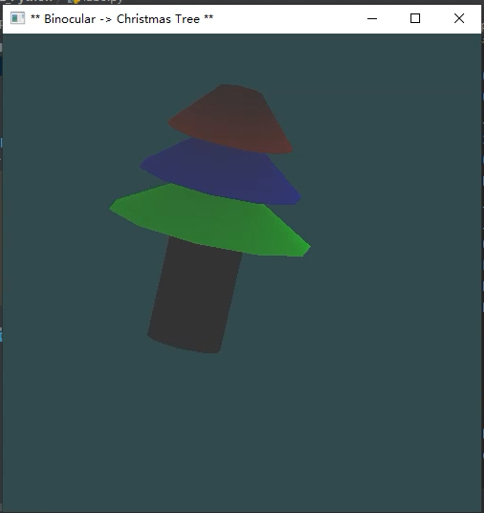
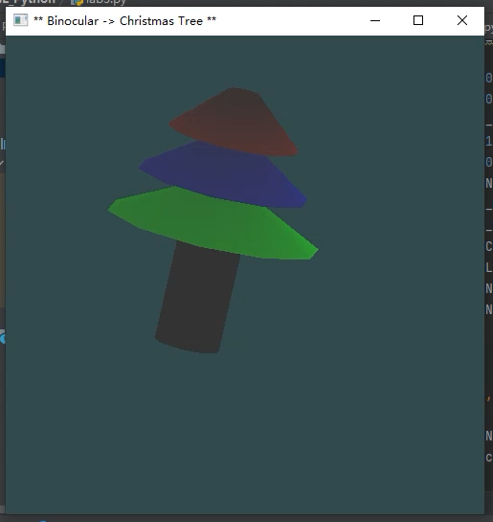
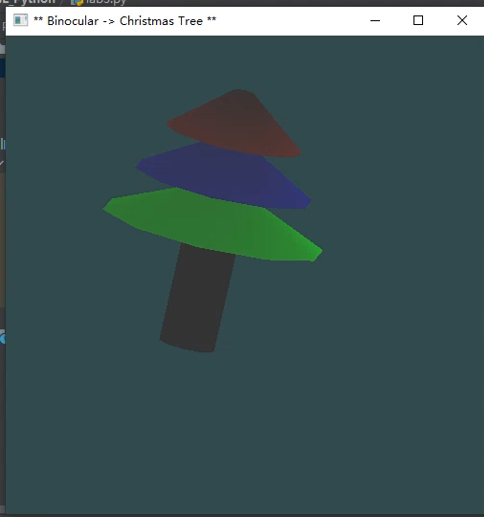
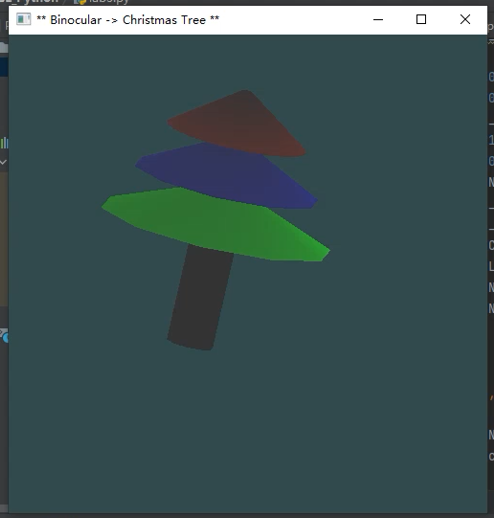
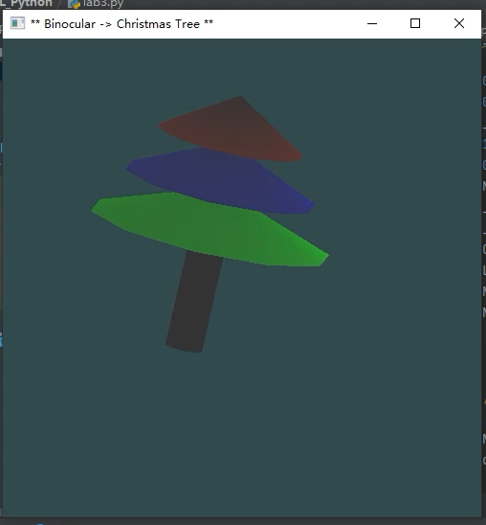
Объекты должны изображать узнаваемые предметы, однако могут при этом быть комбинацией примитивных форм. В сцену должен быть включен источник освещения.

Материалы объектов, определяющие отражение света поверхностью объекта, требуется задать самостоятельно.

**Результат**

Было решено преобразовать монокуляр в елку.



**Код Реальзации**

import time  
from OpenGL.GL import \*  
from OpenGL.GLU import \*  
from OpenGL.GLUT import \*  
import sys  
import math  
  
STEP\_TIME = 2  
  
  
def init():  
 global ambient\_  
 global lightpos\_  
 global increase  
 global color\_white  
 global color\_darkRed  
 global color\_black  
  
 ambient\_ = (1.0, 1.0, 1.0, 1)  
 color\_white = (0.0, 0.0, 0.0, 0.0)  
 lightpos\_ = (1.0, 1.0, 1.0)  
 color\_darkRed = (0.1, 0.0, 0.0, 0.0)  
 color\_black = (0.0, 0.0, 0.0, 0.0)  
  
 glClearColor(0.2, 0.3, 0.3, 1.0)  
 gluOrtho2D(-1.0, 1.0, -1.0, 1.0)  
 glRotatef(-90, 1.0, 0.0, 0.0)  
 glLightModelfv(GL\_LIGHT\_MODEL\_AMBIENT, ambient\_)  
 glEnable(GL\_LIGHTING)  
 glEnable(GL\_LIGHT0)  
 glLightfv(GL\_LIGHT0, GL\_POSITION, lightpos\_)  
  
  
def drawShape(R, r, h, num):  
 num = float(num)  
 c = []  
  
 for i in range(int(num) + 1):  
 angle = 2 \* math.pi \* (i / num)  
 x = R \* math.cos(angle)  
 y = R \* math.sin(angle)  
 x2 = r \* math.cos(angle)  
 y2 = r \* math.sin(angle)  
 pt = ((x), (y), (x2), (y2))  
 c.append(pt)  
  
 glBegin(GL\_TRIANGLE\_FAN)  
 glVertex(0, 0, h / 2.0)  
 for (x, y, x2, y2) in c:  
 z = h / 2.0  
 glVertex(x2, y2, z)  
 glEnd()  
 glBegin(GL\_TRIANGLE\_FAN)  
 for (x, y, x2, y2) in c:  
 z = -h / 2.0  
 glVertex(x, y, z)  
 glEnd()  
 glBegin(GL\_TRIANGLE\_STRIP)  
 for (x, y, x2, y2) in c:  
 for i in range(int(num)):  
 z = h / 2.0  
 glVertex(x, y, -z)  
 glVertex(x2, y2, z)  
 glEnd()  
  
  
def timer(value):  
 glutPostRedisplay()  
 glutTimerFunc(150, timer, 0)  
  
  
def draw():  
 global lightpos\_  
 global color\_white  
 global increase  
 global color\_darkRed  
  
 time.sleep(STEP\_TIME)  
 glClear(GL\_COLOR\_BUFFER\_BIT)  
  
 glPushMatrix()  
  
 glLightfv(GL\_LIGHT0, GL\_POSITION, lightpos\_)  
 lpos = [0.8, 0.8, 0.8, 1.0]  
 lc = [0.8, 1.0, 0.8, 1.0]  
 glMaterialfv(GL\_FRONT\_AND\_BACK, GL\_DIFFUSE, color\_white)  
 glLightfv(GL\_LIGHT0, GL\_POSITION, lpos)  
 glLightfv(GL\_LIGHT0, GL\_DIFFUSE, lc)  
 glLightf(GL\_LIGHT0, GL\_CONSTANT\_ATTENUATION, 0.1)  
 glLightf(GL\_LIGHT0, GL\_LINEAR\_ATTENUATION, 0.01)  
 glMaterialfv(GL\_FRONT\_AND\_BACK, GL\_DIFFUSE, color\_white)  
 glMaterialfv(GL\_FRONT\_AND\_BACK, GL\_SHININESS, color\_white)  
 glEnable(GL\_LIGHT0)  
 glTranslatef(-0.2, 0.0, -0.1)  
 glRotatef(20, 1.0, 1.0, 1.0)  
 color\_green = (0.0, 0.1, 0.0, 0.0)  
  
 glMaterialfv(GL\_FRONT\_AND\_BACK, GL\_SHININESS, color\_black)  
 drawShape(0.3 - 0.2 \* increase, 0.3 - 0.2 \* increase, 0.2, 8 + 10 \* increase) # 6  
 drawShape(0.3 - 0.2 \* increase, 0.3 - 0.2 \* increase, 0.4, 8 + 10 \* increase) # 5  
 glTranslatef(0.0, 0.0, 0.2)  
 glMaterialfv(GL\_FRONT\_AND\_BACK, GL\_DIFFUSE, color\_black)  
 drawShape(0.3 - 0.2 \* increase, 0.3 - 0.2 \* increase, 0.2, 8 + 10 \* increase) # 4  
 glTranslatef(0.0, 0.0, 0.2)  
 glMaterialfv(GL\_FRONT\_AND\_BACK, GL\_DIFFUSE, color\_green)  
 drawShape(0.3 + 0.2 \* increase, 0.3 - 0.28 \* increase, 0.2, 8 + 10 \* increase) # 3  
 glTranslatef(0.0, 0.0, 0.2)  
 color\_blue = (0.0, 0.0, 0.1, 0.0)  
 glMaterialfv(GL\_FRONT\_AND\_BACK, GL\_DIFFUSE, color\_blue)  
 drawShape(0.25 + 0.15 \* increase, 0.3 - 0.28 \* increase, 0.2, 8 + 10 \* increase) # 2  
 glMaterialfv(GL\_FRONT\_AND\_BACK, GL\_DIFFUSE, color\_darkRed)  
 glTranslatef(0.0, 0.0, 0.2)  
 drawShape(0.2 + 0.1 \* increase, 0.25 - 0.23 \* increase, 0.2, 20) # 1  
  
 glPopMatrix()  
  
 if increase < 1: increase += 0.1  
 glFlush()  
 glutSwapBuffers()  
  
  
# Run the script  
if \_\_name\_\_ == '\_\_main\_\_':  
 STEP\_TIME = 0.01  
 increase = 0  
  
 glutInitDisplayMode(GLUT\_DOUBLE | GLUT\_RGB)  
 glutInitWindowSize(800, 800)  
 glutInitWindowPosition(50, 50)  
 glutInit(sys.argv)  
  
 glutCreateWindow(b"Binocular -> Christmas Tree")  
 glutDisplayFunc(draw)  
 glutTimerFunc(150, timer, 0)  
  
 init()  
 glutMainLoop()