**Московский авиационный институт**

**(Национальный исследовательский университет)**

Институт: «Информационные технологии и прикладная математика»

Кафедра: 806 «Вычислительная математика и программирование»

Дисциплина: «Компьютерная графика»

**Лабораторная работа № 4**

Тема: Ознакомление с технологией OpenGL

Студент: Попова Наталья Сергеевна

Группа: 08-305

Преподаватель: Чернышов Л.Н.

Дата: 09.11.2022

Оценка:

Москва, 2022

1. **Постановка задачи**

Создать графическое приложение с использованием OpenGL. Используя результаты Л.Р.№3, изобразить

заданное тело (то же, что и в л.р. №3) с использованием средств OpenGL 2.1. Использовать буфер вершин. Точность

аппроксимации тела задается пользователем. Обеспечить возможность вращения и масштабирования многогранника

и удаление невидимых линий и поверхностей. Реализовать простую модель освещения на GLSL.

Параметры освещения и отражающие свойства материала задаются пользователем в диалоговом режиме.

**Вариант №15**: Сектор эллипсоида.

1. Описание программы

**Язык программирования**: Python

**Используемые библиотеки**: pygame, numpy, OpenGl.GL, Open.GLU

**Используемая среда программирования**: VS Code

Программа строит сектор эллипсоида в зависимости от переданных на вход значений a, b, c; окрашивает грани фигуры в зависимости от поворота фигуры. Также в ней можно поворачивать фигуры вдоль осей OX, OY, OZ; менять коэффициенты аппроксимации и масштаб.

Функции getPoints, getFaces находят соответственно точки сектора эллипсоида по формуле эллипсоида с заданными a, b, c и грани по точкам.

Функция ellipsoid\_sector строит фигуры с помощью двух выше описанных функций, а также с помощью функций OpenGl.

В главной функции main происходят построения окна и тела. В ней реализуются повороты тела вокруг осей, масштабирование, увеличение или уменьшение коэффициента аппроксимации путем нажатия клавиш «X», «Y», «Z»; «up», «down», «left», «right»; «+», «-» соответственно.

1. **Набор тестов**

**Тест №1**

Точность аппроксимации = 5

Параметры эллипсоида: A = 2 , B = 1.5 , C = 1.6

Повороты: X = 0 , Y = 0, Z = 0

**Тест №2**

Точность аппроксимации = 8

Параметры эллипсоида: A = 2 , B = 1.5 , C = 1.6

Повороты: X = 0 , Y = 0, Z = 0

**Тест №3**

Точность аппроксимации = 8

Параметры эллипсоида: A = 2 , B = 1.5 , C = 1.6

Повороты: X = 270 , Y = 0, Z = 60

**Тест №4**

Точность аппроксимации = 5

Параметры эллипсоида: A = 2 , B = 1.5 , C = 1.6

Повороты: X = 60 , Y = 60, Z = 0

**Тест №5**

Точность аппроксимации = 3

Параметры эллипсоида: A = 3 , B = 2 , C = 1

Повороты: X = 60 , Y = 120, Z = 60

1. **Результаты выполнения тестов**

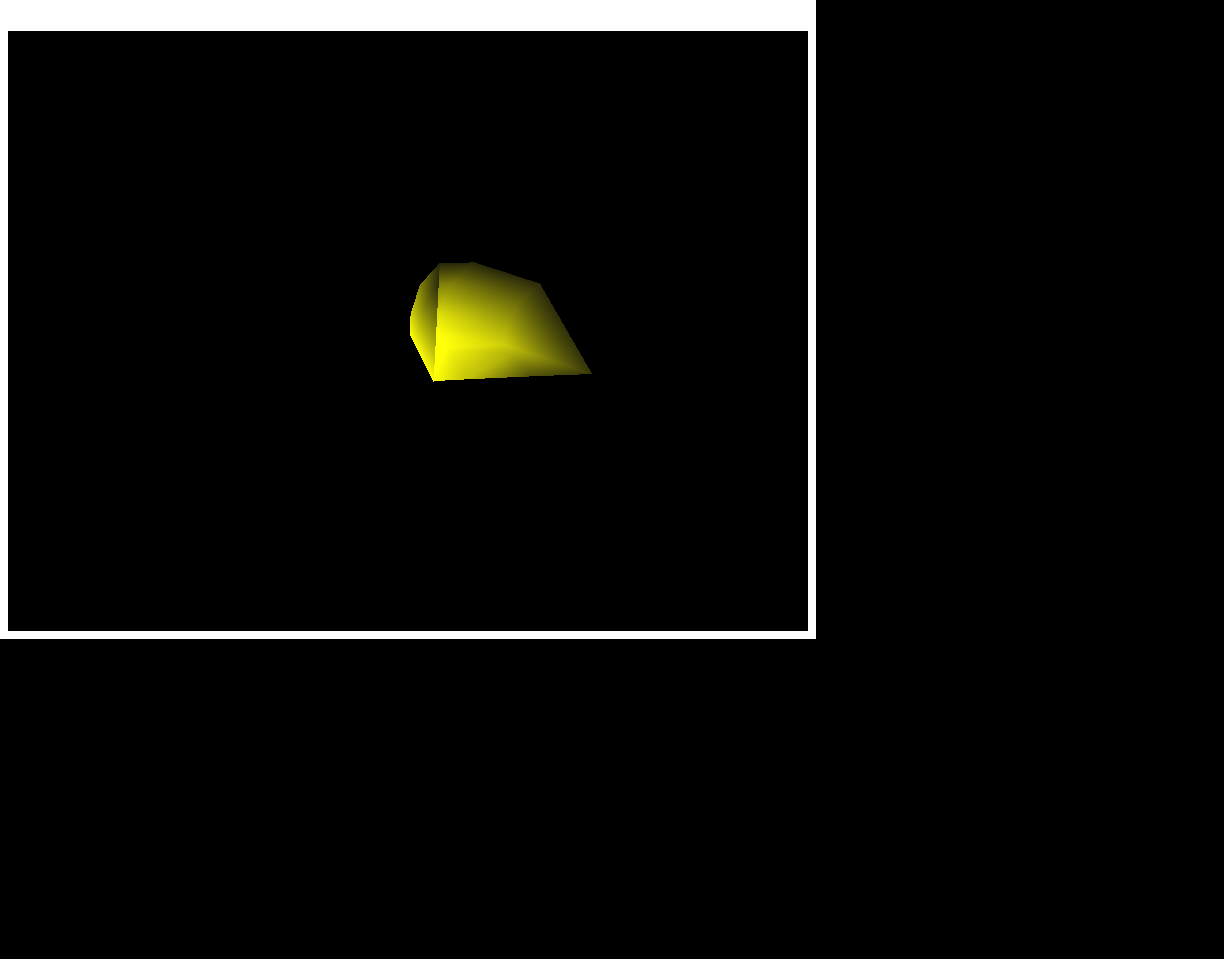


Рисунок 1 - Тест №1

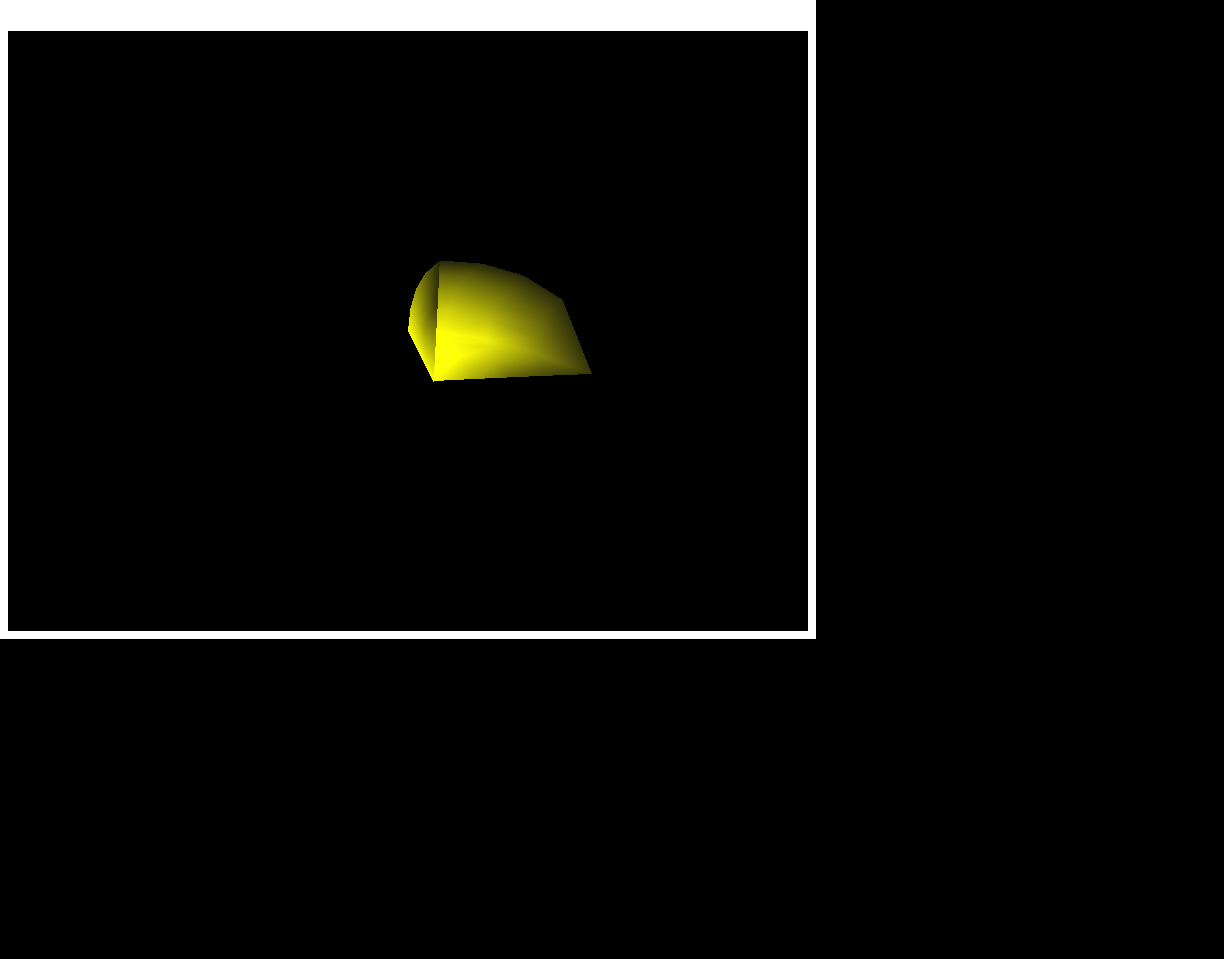


Рисунок 2 - Тест №2

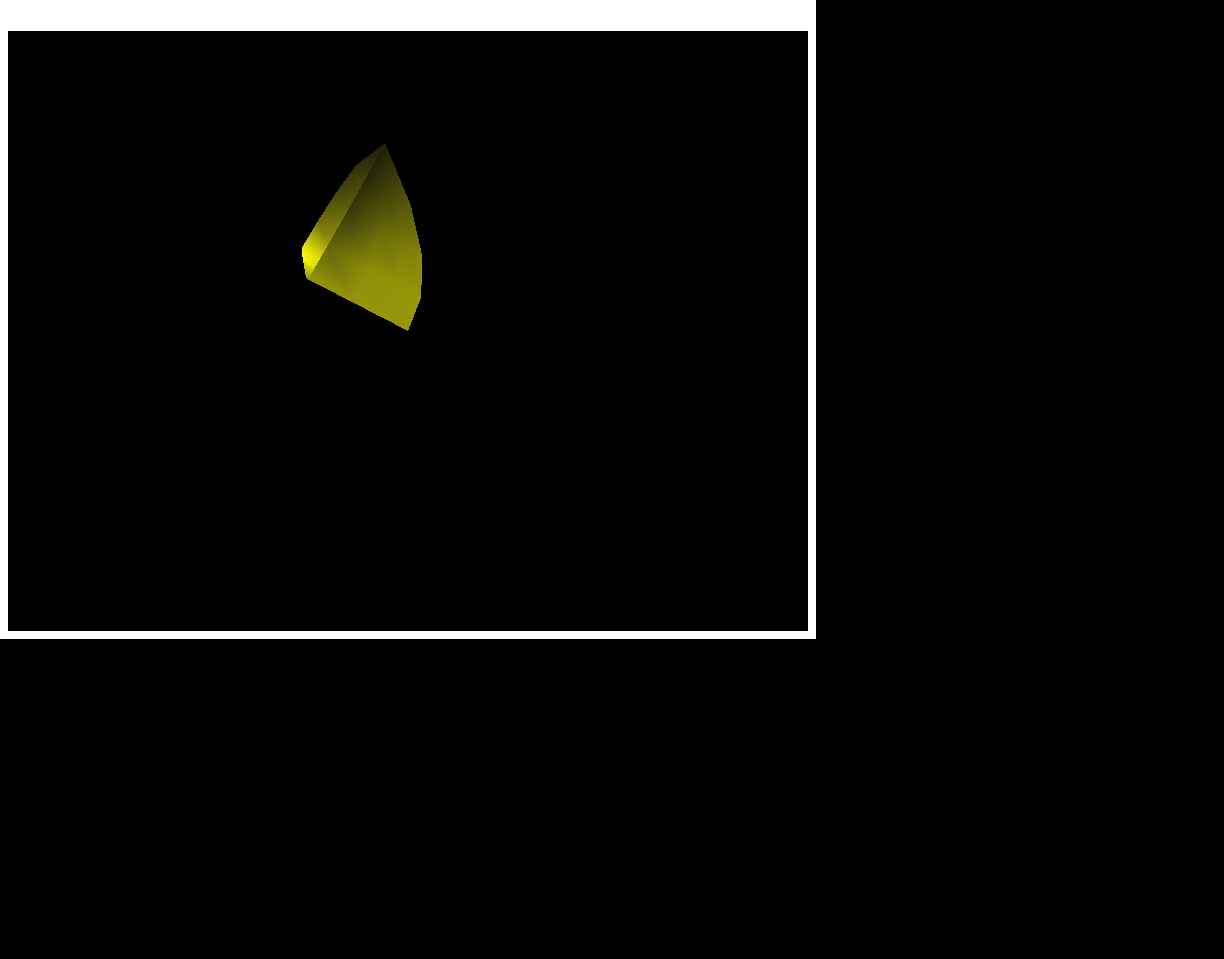


Рисунок 3 - Тест №3

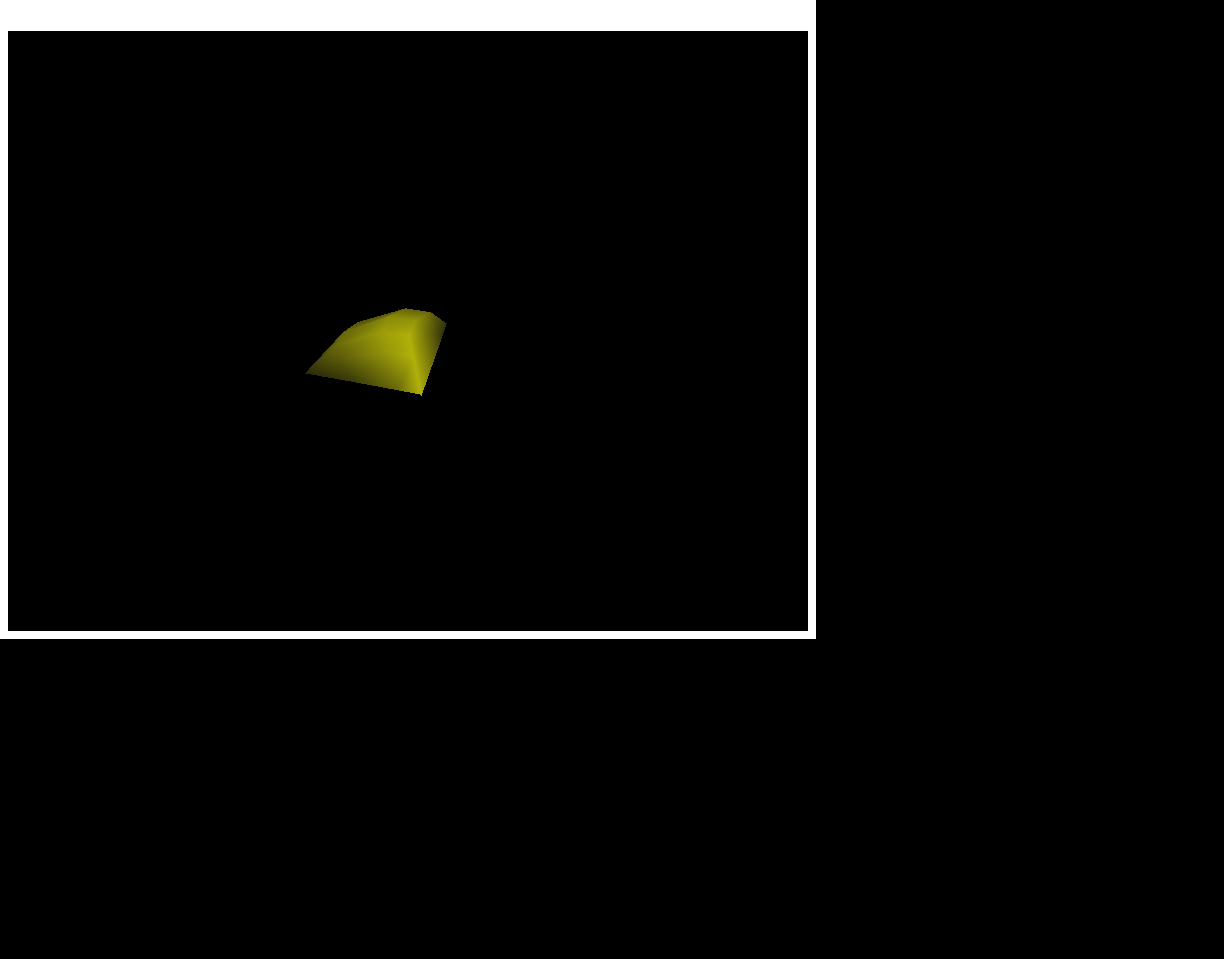


Рисунок 4 - Тест №4

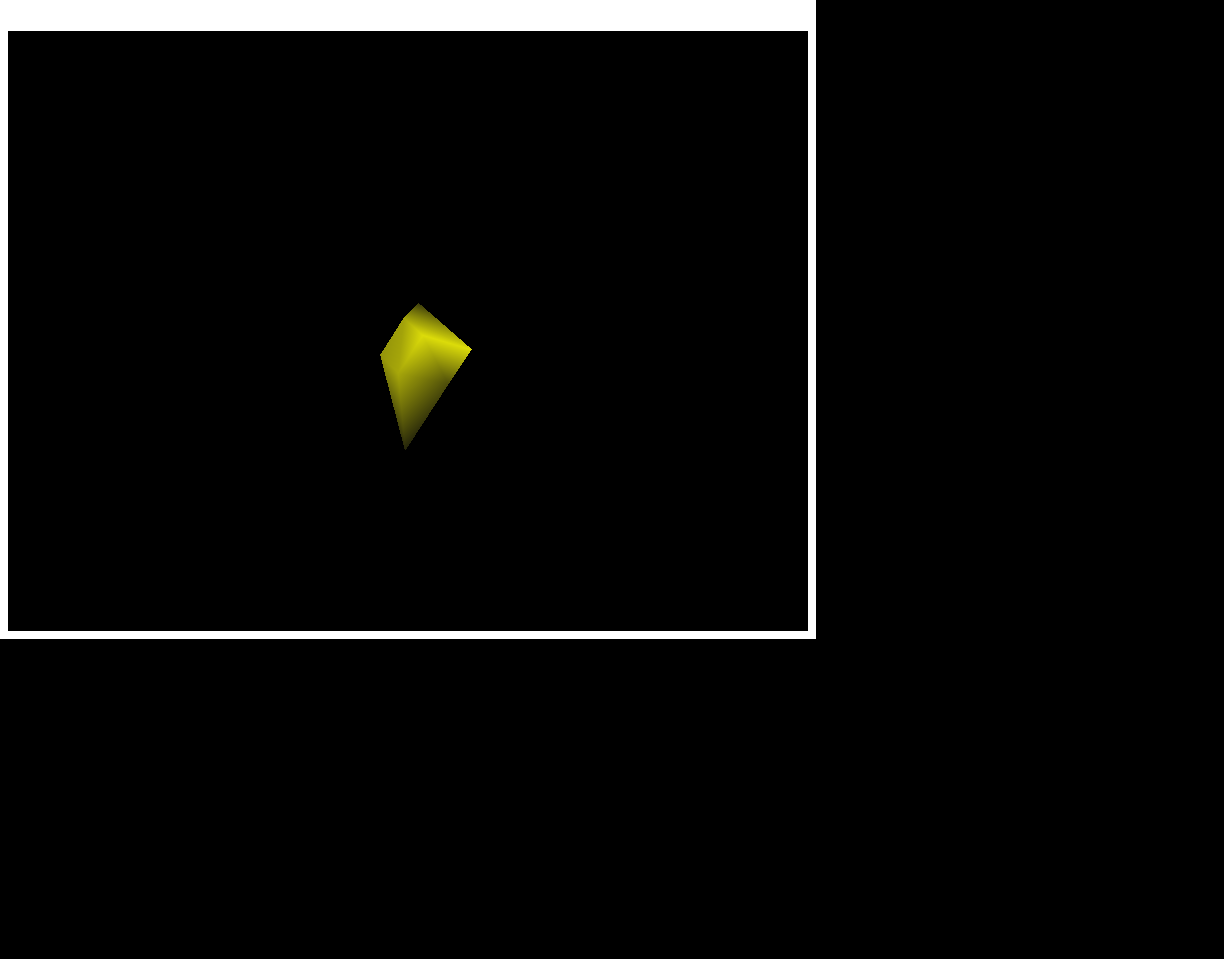


Рисунок 5 - Тест №5

1. **Листинг программы**

*# Попова Наталья, М8О-305Б-20, Варинт 15 - сектор эллипсоида***import** pygame  
**import** numpy **as** np  
**from** pygame.locals **import** \*  
**from** OpenGL.GL **import** \*  
**from** OpenGL.GLU **import** \*  
  
  
**def** getPoints(a=3, b=2, c=1, precision=5):  
 points = []  
 maxX = 100000 \* a / 100001  
 stepX = 2 \* maxX / precision  
 x = -maxX  
 **while** x <= (maxX + maxX / 100000):  
 maxY = b \* np.sqrt(1 - (x / a) \* (x / a))  
 maxY = 100 \* maxY / 101  
 stepY = 2 \* maxY / precision  
 currentPoints = []  
 y = -maxY  
 **while** y <= (maxY + maxY / 100):  
 z = c \* ((x / a) \* (x / a) + (y / b) \* (y / b))  
 **if** z < 0:  
 **if** x < 0:  
 **if** y < 0:  
 currentPoints.append([0, 0, 0])  
 **else**:  
 currentPoints.append([0, y, 0])  
 **else**:  
 currentPoints.append([x, y, 0])  
 **elif** x < 0:  
 **if** y < 0:  
 currentPoints.append([0, 0, z])  
 **else**:  
 currentPoints.append([0, y, z])  
 **elif** (y < 0):  
 currentPoints.append([x, 0, z])  
 **else**:  
 currentPoints.append([x, y, z])  
 y += stepY  
 points.append(currentPoints)  
 x += stepX  
 lowPoints = []  
 **for** i **in** range(0, len(points)):  
 currentPoints = []  
 **for** j **in** range(len(points[i]) - 1, -1, -1):  
 currentPoints.append([points[i][j][0], points[i][j][1], points[i][j][2]])  
 lowPoints.append(currentPoints)  
 **for** i **in** range(0, len(points)):  
 points[i] = points[i] + lowPoints[i]  
 **return** points  
  
  
**def** getFaces(points):  
 k = len(points)  
 n = len(points[0])  
 faces = []  
 **for** i **in** range(0, k - 1):  
 **for** j **in** range(0, n - 1):  
 temp = [[i, j], [i + 1, j], [i + 1, j + 1], [i, j + 1]]  
 faces.append(temp)  
 faces.append([[i, n - 1], [i + 1, n - 1], [i + 1, 0], [i, 0]])  
 **return** faces  
  
  
**def** ellipsoid\_sector(a, b, c, precision):  
 points = getPoints(a=a, b=b, c=c, precision=precision)  
 faces = getFaces(points)  
 glBegin(GL\_QUADS)  
 **for** face **in** faces:  
 **for** point **in** face:  
 COLOR = ((1, 1, 0))  
 glMaterial(GL\_FRONT\_AND\_BACK, GL\_DIFFUSE, COLOR)  
 glVertex3fv(points[point[0]][point[1]])  
 glEnd()  
  
  
**def** main():  
 pygame.init()  
 display = (800, 600)  
 pygame.display.set\_mode(display, DOUBLEBUF | OPENGL)  
  
 gluPerspective(45, (display[0] / display[1]), 0.1, 50.0)  
 glTranslatef(0, 0, -10)  
 glRotatef(25, 2, 1, 0)  
  
 run = **True** a = 2  
 b = 1.5  
 c = 1.6  
 precision = 5  
  
 **while** run:  
 events = pygame.event.get()  
 **for** event **in** events:  
 **if** event.type == pygame.QUIT:  
 pygame.quit()  
 run = **False** quit()  
  
 **if** event.type == pygame.KEYDOWN:  
 **if** event.key == pygame.K\_LEFT:  
 glTranslatef(-0.5, 0, 0)  
 **if** event.key == pygame.K\_RIGHT:  
 glTranslatef(0.5, 0, 0)  
  
 **if** event.key == pygame.K\_UP:  
 glTranslatef(0, 1, 0)  
 **if** event.key == pygame.K\_DOWN:  
 glTranslatef(0, -1, 0)  
 **if** event.key == pygame.K\_x:  
 glRotatef(30, 1, 0, 0)  
 **if** event.key == pygame.K\_y:  
 glRotatef(30, 0, 1, 0)  
 **if** event.key == pygame.K\_z:  
 glRotatef(30, 0, 0, 1)  
 **if** event.key == pygame.K\_ESCAPE:  
 pygame.quit()  
 **if** event.key == pygame.K\_EQUALS:  
 precision += 1  
 **if** event.key == pygame.K\_MINUS:  
 **if** precision > 3:  
 precision -= 1  
 **if** event.key == pygame.K\_r:  
 a += 0.1  
 **if** event.key == pygame.K\_t:  
 **if** a > 0.1:  
 a -= 0.1  
 **if** event.key == pygame.K\_f:  
 b += 0.1  
 **if** event.key == pygame.K\_g:  
 **if** b > 0.1:  
 b -= 0.1  
 **if** event.key == pygame.K\_a:  
 c += 0.1  
 **if** event.key == pygame.K\_s:  
 **if** a > 0.1:  
 a -= 0.1  
  
 **if** event.type == pygame.MOUSEBUTTONDOWN:  
 **if** event.button == 4:  
 glTranslatef(0, 0, 1.0)  
  
 **if** event.button == 5:  
 glTranslatef(0, 0, -1.0)  
  
 glClear(GL\_COLOR\_BUFFER\_BIT | GL\_DEPTH\_BUFFER\_BIT)  
  
 glEnable(GL\_CULL\_FACE)  
 glCullFace(GL\_FRONT)  
  
 glLightfv(GL\_LIGHT0, GL\_POSITION, (0, 0, 2, 1))  
 glLightfv(GL\_LIGHT0, GL\_AMBIENT, (0, 0, 0, 1))  
 glLightfv(GL\_LIGHT0, GL\_DIFFUSE, (1, 1, 1, 1))  
 glEnable(GL\_DEPTH\_TEST)  
 glEnable(GL\_LIGHTING)  
 glEnable(GL\_LIGHT0)  
  
 ellipsoid\_sector(a=a, b=b, c=c, precision=precision)  
  
 pygame.display.flip()  
 pygame.time.wait(10)  
  
  
**if** \_\_name\_\_ == **'\_\_main\_\_'**:  
 main()

**ЛИТЕРАТУРА**

1. Документация к библиотеке WebGL [Электронный ресурс].

URL: <https://ru.wikipedia.org/wiki/WebGL> (дата обращения 08.11.22)

1. Руководство openGL [Электронный ресурс].

URL: <https://www.opengl.org.ru/docs/pg/index.html> (дата обращения 08.11.22)

1. Статьи по OpenGL [Электронный ресурс].

URL: <http://steps3d.narod.ru/tutorials/pyopengl-tutorial.html> (дата обращения 09.11.22)

1. OpenGL на Python [Электронный ресурс].

URL: <https://habr.com/ru/post/246625/> (дата обращения 09.11.22)