МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

Отчет
по лабораторной работе №2
по дисциплине «КОМПЬЮТЕРНАЯ 3D-ГРАФИКА»
Тема: 3D трансформации.

Студентка гр. 5381 _____ Буздина М.А.

Преподаватель _____ Герасимова Т.В.

Санкт-Петербург
2019

Цель работы.

Представить 3D сцену и в ней один или несколько объектов из вашего задания (стараться представить все).

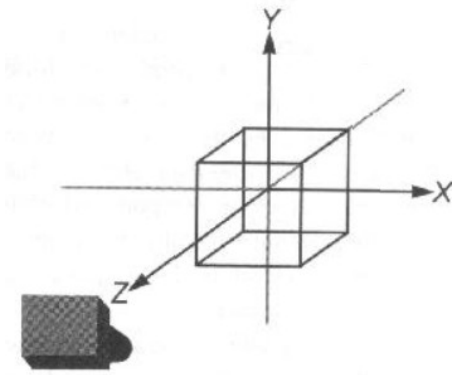
Задание.

Освоить использование стандартных матричных преобразований над этими объектами и осуществить:

1. Просмотр трансформаций: через lookAt или эквивалентные модельные трансформации;
2. Трансформацию проекции: через perspective и ortho;
3. Моделирование трансформации rotate, translate, scale и матричный стек.

Основные теоретические сведения.

Предположим, что мы хотим повернуть куб на 30 градусов и поместить его на 5 единиц от камеры для рисования. Вы можете написать программу интуитивно, как показано ниже:

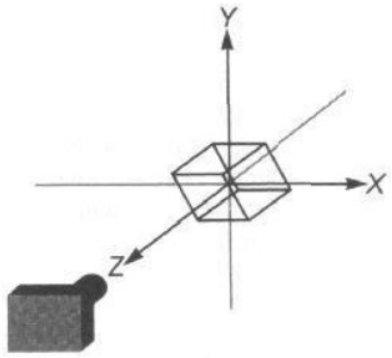



```
// сначала вращаемся вокруг оси x на 30 градусов
    mv = mult (mv, rotateX (30));
    // затем переносим обратно 5
    mv = mult (mv, translate (0, 0, -5));
    // Копирование mv в шейдер
    gl.uniformMatrix4fv (mvLoc, gl.FALSE, flatten (mv));
    // рисуем модель куба с центром в начале координат
 gl.drawArrays (gl.LINE_STRIP, wireCubeStart, wireCubeVertices);
```

Рис. 3

На рис. 3. показан эффект этих преобразований: результат последовательного изменения матрицы поворота за которой следует перенос. Но мы видим, что после запуска программы ничего не изменилось. Почему?

Если мы немного изменим программу, как показано ниже:




```
/// первый перевод назад 5
    mv = mult (mv, translate (0, 0, -5));
    // затем поворачиваем вокруг оси x на 30 градусов
    mv = mult (mv, rotateX (30));
    // Копируем mv в шейдер
    gl.uniformMatrix4fv (mvLoc, gl.FALSE, flatten (mv));
  // Нарисовать модель куба с центром в начале координат
   gl.drawArrays (gl.LINE_STRIP, wireCubeStart, wireCubeVertices);
```

Рис.3.4

На рис.3.4. показан новый результат.

Ход работы.

Результатом работы данного курса представлен на рис.1:

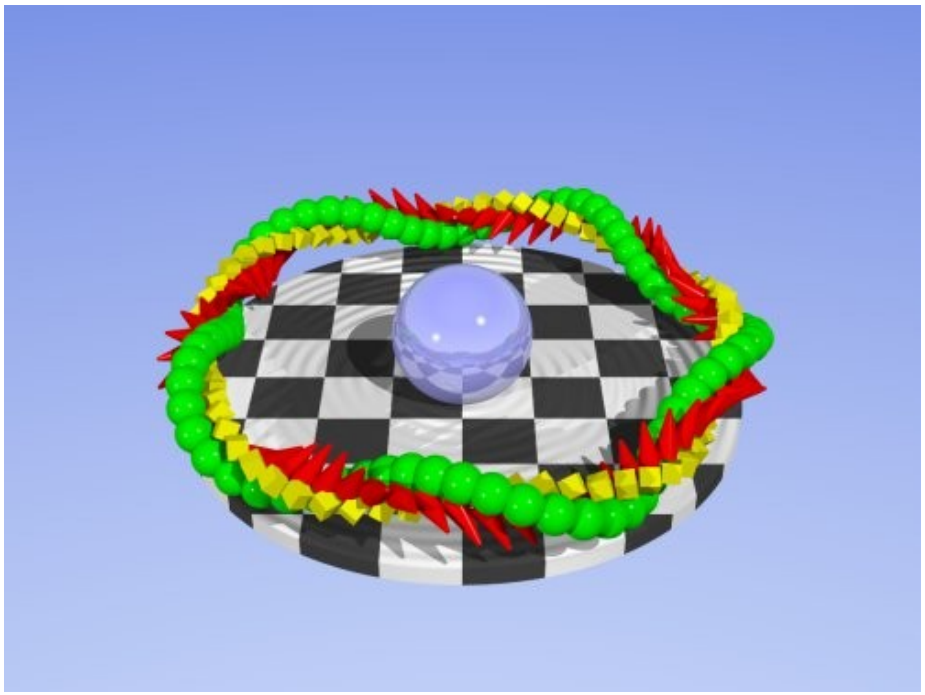


Рисунок 1 — Финальная сцена

Все перемещения и изменения размеров на рисунках 2-5 сделаны при помощи трансформаций rotate, translate, scale.

```
//Draw Rectangle
mvPushMatrix();
mat4.translate(mvMatrix, [0.0, 0.0, 0.0]); //перемещение
mat4.scale(mvMatrix, [1, 1, 1]); //масштабирование
mat4.rotate(mvMatrix, degToRad(-270), [1, 1, 1]); // поворот
setBuffersToShaders(figures[0].getPositionBuffer(),
figures[0].getColorBuffer());
gl.drawArrays(gl.TRIANGLE_STRIP, 0, figures[0].getPositionBuffer().numItems);
mvPopMatrix();
//Draw Triangle
mvPushMatrix();
mat4.translate(mvMatrix, [-8.0, -3.0, -5.0]); //перемещение
mat4.scale(mvMatrix, [5, 1, 5]);//масштабирование
mat4.rotate(mvMatrix, degToRad(-90), [1, 1, 1]); // поворот
setBuffersToShaders(figures[1].getPositionBuffer(),
figures[1].getColorBuffer());
gl.drawArrays(gl.TRIANGLES, 0, figures[1].getPositionBuffer().numItems);
mvPopMatrix();
//Draw Circle
mvPushMatrix();
mat4.translate(mvMatrix, [4.0, 1.0, -20.0]); //перемещение
mat4.scale(mvMatrix, [2, 2, 1]);//масштабирование
mat4.rotate(mvMatrix, degToRad(45), [1, 1, 1]); // поворот
setBuffersToShaders(figures[2].getPositionBuffer(),
figures[2].getColorBuffer());
gl.drawArrays(gl.TRIANGLE_FAN, 0, figures[2].getPositionBuffer().numItems);
mvPopMatrix();
```

Трансформации: через lookAt. В данной работе обеспечено изменение положения камеры с использованием lookAt при нажатии клавиш стрелок и для изменения z координаты pgup и pgdn. На рисунках 2-3 представлены различные углы обзора одной композиции.

```
mat4.lookAt([xCameraPos, yCameraPos, zCameraPos], [0, 0, 0], [0, 1, 0],
mvMatrix);
```

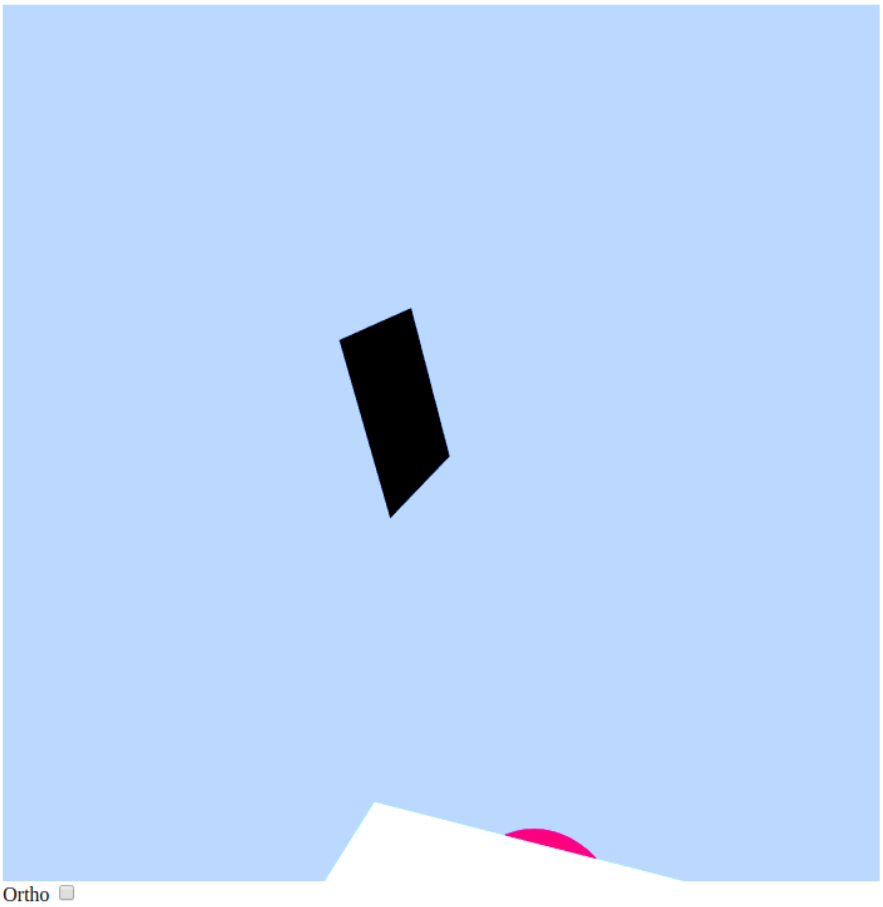


Рисунок 2 — Вид снизу

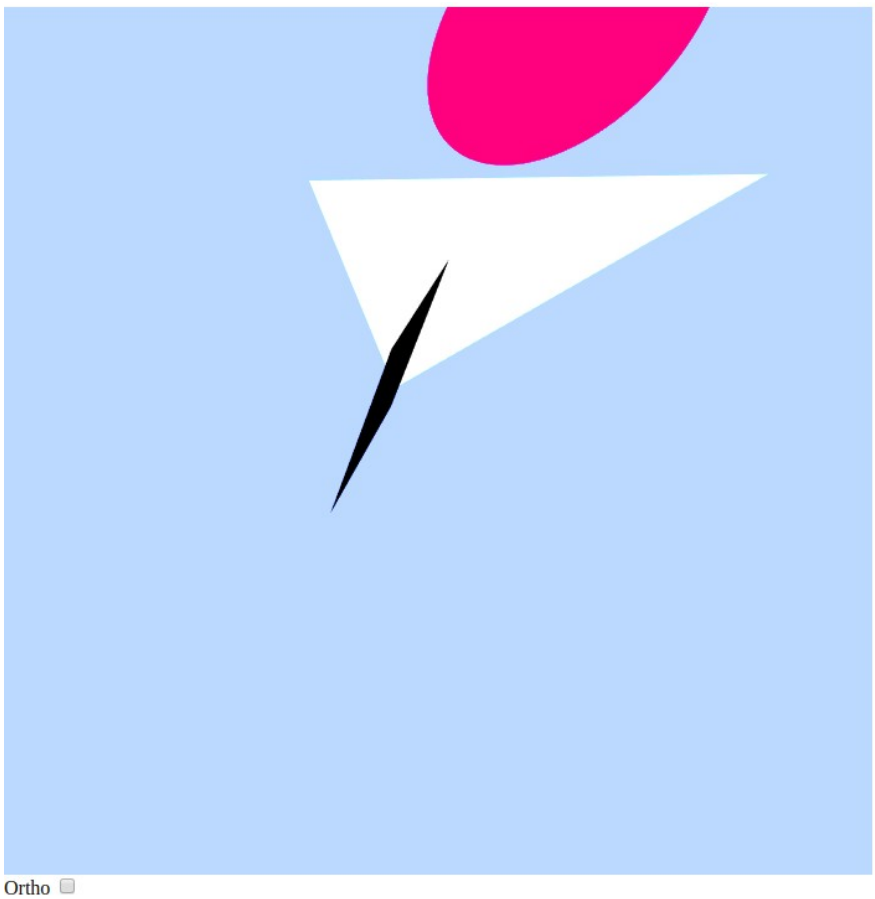


Рисунок 3 — Вид сверху

Реализация ортогональной и перспективной проекций: через perspective и ortho. На рисунке 4 представлена перспективная проекция, на рисунке 5 — ортогональная.

```
if (document.getElementById("ortho").checked){
    mat4.ortho(-gl.viewportWidth / 100, gl.viewportWidth/100, -
gl.viewportHeight/100, gl.viewportHeight/100, 0.1, 100.0, pMatrix);
}
else {
    mat4.perspective(60, gl.viewportWidth / gl.viewportHeight, 0.1, 100.0,
pMatrix);
}
```

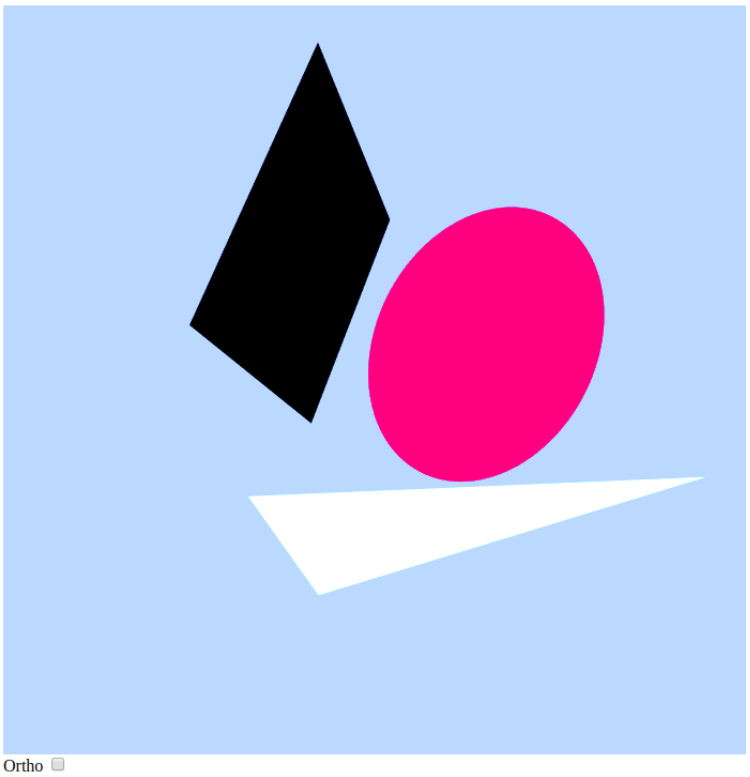


Рисунок 4 — Перспективная проекция

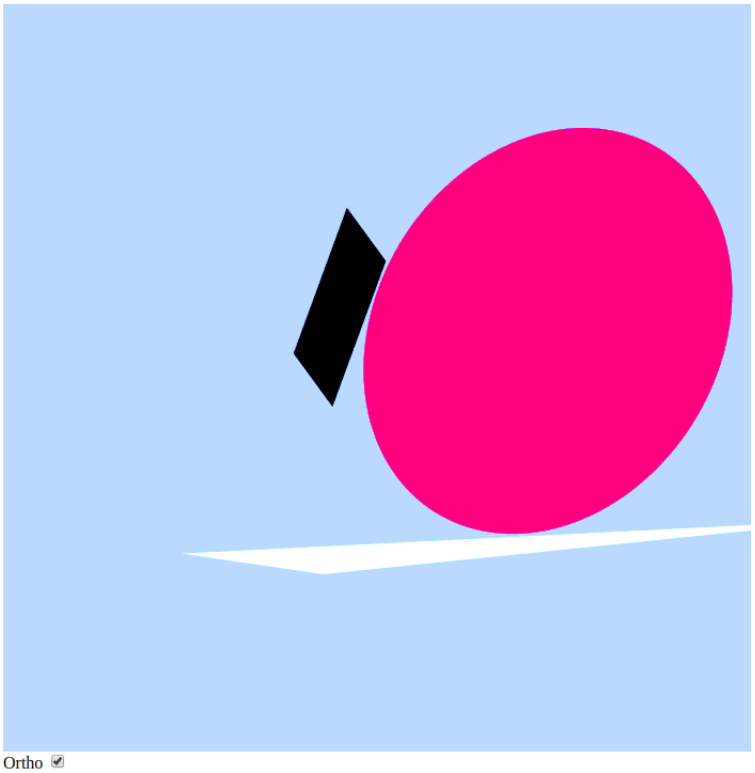


Рисунок 5 — Ортогональная проекция

Выводы.

В ходе данной работы была представлена 3D сцена с объектами из задания и освоены стандартные матричные преобразования над ними: просмотр трансформаций: через lookAt или эквивалентные модельные трансформации, трансформацию проекции: через perspective и ortho, моделирование трансформации rotate, translate, scale и матричный стек.