**Министерство образования Республики Беларусь**

**Учреждение образования**

**«Белорусский государственный университет информатики   
и радиоэлектроники»**

Кафедра интеллектуальных

информационных технологий

**Отчёт**

по лабораторной работе № 4

по курсу «Графический интерфейс интеллектуальных систем»

Выполнили:

студенты гр. 821701

Булова М.И.

Титенков П.В

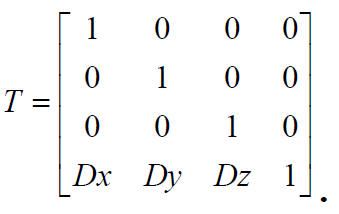
Минск 2011

**Цель**

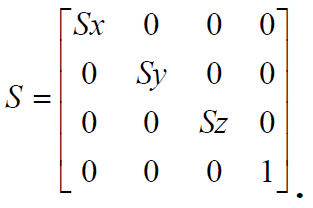
Изучить базовые геометрические операции (поворот, перемещение, скалирование), их математическое и векторное представление. Реализовать функции геометрических преобразований в двумерном и трехмерном пространстве, а также перспективные преобразования.

**Теоритические сведения**

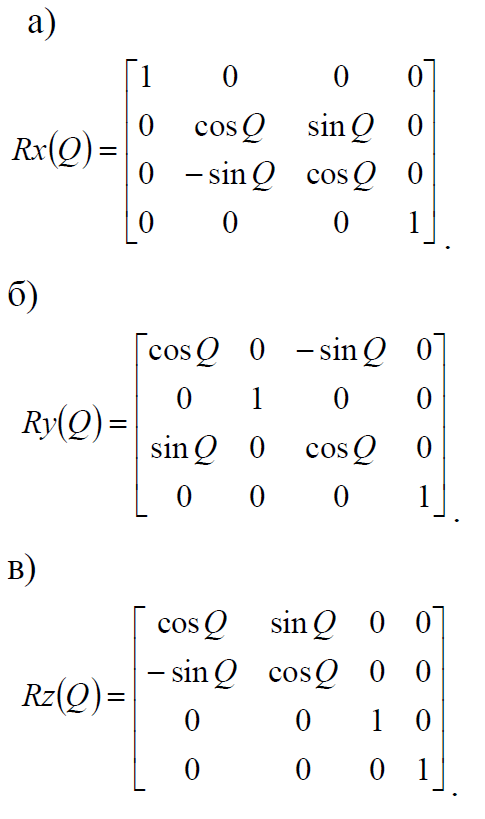
В трехмерном случае есть пять базовых операций:  
1) Перемещение



2) Скалирование



3) Вращение вокруг оси



**Реализация**

public Matrix moving**(**Matrix matrix**,** String courseMoving**)** **{**

**if** **(**courseMoving**.**equals**(**"right"**))** **{**

**for** **(**int i **=** 0**;** i **<** matrix**.**getHeight**();** i**++)** **{**

matrix**.**setElementAt**(**i**,** 0**,** matrix**.**getElementAt**(**i**,** 0**)** **+** 3.0**);**

**}**

**}**

**if** **(**courseMoving**.**equals**(**"left"**))** **{**

**for** **(**int i **=** 0**;** i **<** matrix**.**getHeight**();** i**++)** **{**

matrix**.**setElementAt**(**i**,** 0**,** matrix**.**getElementAt**(**i**,** 0**)** **-** 3.0**);**

**}**

**}**

**if** **(**courseMoving**.**equals**(**"up"**))** **{**

**for** **(**int i **=** 0**;** i **<** matrix**.**getHeight**();** i**++)** **{**

matrix**.**setElementAt**(**i**,** 1**,** matrix**.**getElementAt**(**i**,** 1**)** **-** 3.0**);**

**}**

**}**

**if** **(**courseMoving**.**equals**(**"down"**))** **{**

**for** **(**int i **=** 0**;** i **<** matrix**.**getHeight**();** i**++)** **{**

matrix**.**setElementAt**(**i**,** 1**,** matrix**.**getElementAt**(**i**,** 1**)** **+** 3.0**);**

**}**

**}**

**if** **(**courseMoving**.**equals**(**"z up"**))** **{**

**for** **(**int i **=** 0**;** i **<** matrix**.**getHeight**();** i**++)** **{**

matrix**.**setElementAt**(**i**,** 2**,** matrix**.**getElementAt**(**i**,** 2**)** **+** 3.0**);**

**}**

**}**

**if** **(**courseMoving**.**equals**(**"z down"**))** **{**

**for** **(**int i **=** 0**;** i **<** matrix**.**getHeight**();** i**++)** **{**

matrix**.**setElementAt**(**i**,** 2**,** matrix**.**getElementAt**(**i**,** 2**)** **-** 3.0**);**

**}**

**}**

**return** matrix**;**

**}**

public Matrix rotate**(**Matrix matrix**,** String courseRotating**)** **{**

setMatrixRotate**();**

**if** **(**angle **>=** Math**.**PI**)** **{**

angle **=** 0**;**

**}**

**if** **(**courseRotating**.**equals**(**"right"**))** **{**

angle **=** 10 **/** 180. **\*** Math**.**PI**;**

matrixRotate**.**setElementAt**(**0**,** 0**,** Math**.**cos**(**angle**));**

matrixRotate**.**setElementAt**(**0**,** 2**,** **-**Math**.**sin**(**angle**));**

matrixRotate**.**setElementAt**(**2**,** 0**,** Math**.**sin**(**angle**));**

matrixRotate**.**setElementAt**(**2**,** 2**,** Math**.**cos**(**angle**));**

matrixRotate**.**setElementAt**(**1**,** 1**,** 1.0**);**

matrixRotate**.**setElementAt**(**3**,** 3**,** 1.0**);**

**}**

**if** **(**courseRotating**.**equals**(**"left"**))** **{**

angle **=** **-(**10 **/** 180. **\*** Math**.**PI**);**

matrixRotate**.**setElementAt**(**0**,** 0**,** Math**.**cos**(**angle**));**

matrixRotate**.**setElementAt**(**0**,** 2**,** **-**Math**.**sin**(**angle**));**

matrixRotate**.**setElementAt**(**2**,** 0**,** Math**.**sin**(**angle**));**

matrixRotate**.**setElementAt**(**2**,** 2**,** Math**.**cos**(**angle**));**

matrixRotate**.**setElementAt**(**1**,** 1**,** 1.0**);**

matrixRotate**.**setElementAt**(**3**,** 3**,** 1.0**);**

**}**

**if** **(**courseRotating**.**equals**(**"up"**))** **{**

angle **=** 10 **/** 180. **\*** Math**.**PI**;**

matrixRotate**.**setElementAt**(**1**,** 1**,** Math**.**cos**(**angle**));**

matrixRotate**.**setElementAt**(**1**,** 2**,** Math**.**sin**(**angle**));**

matrixRotate**.**setElementAt**(**2**,** 1**,** **-**Math**.**sin**(**angle**));**

matrixRotate**.**setElementAt**(**2**,** 2**,** Math**.**cos**(**angle**));**

matrixRotate**.**setElementAt**(**0**,** 0**,** 1.0**);**

matrixRotate**.**setElementAt**(**3**,** 3**,** 1.0**);**

**}**

**if** **(**courseRotating**.**equals**(**"down"**))** **{**

angle **=** **-(**10 **/** 180. **\*** Math**.**PI**);**

matrixRotate**.**setElementAt**(**1**,** 1**,** Math**.**cos**(**angle**));**

matrixRotate**.**setElementAt**(**1**,** 2**,** Math**.**sin**(**angle**));**

matrixRotate**.**setElementAt**(**2**,** 1**,** **-**Math**.**sin**(**angle**));**

matrixRotate**.**setElementAt**(**2**,** 2**,** Math**.**cos**(**angle**));**

matrixRotate**.**setElementAt**(**0**,** 0**,** 1.0**);**

matrixRotate**.**setElementAt**(**3**,** 3**,** 1.0**);**

**}**

**if** **(**courseRotating**.**equals**(**"z up"**))** **{**

angle **=** **-(**10 **/** 180. **\*** Math**.**PI**);**

matrixRotate**.**setElementAt**(**0**,** 0**,** Math**.**cos**(**angle**));**

matrixRotate**.**setElementAt**(**0**,** 1**,** Math**.**sin**(**angle**));**

matrixRotate**.**setElementAt**(**1**,** 0**,** **-**Math**.**sin**(**angle**));**

matrixRotate**.**setElementAt**(**1**,** 1**,** Math**.**cos**(**angle**));**

matrixRotate**.**setElementAt**(**2**,** 2**,** 1.0**);**

matrixRotate**.**setElementAt**(**3**,** 3**,** 1.0**);**

**}**

**if** **(**courseRotating**.**equals**(**"z down"**))** **{**

angle **=** 10 **/** 180. **\*** Math**.**PI**;**

matrixRotate**.**setElementAt**(**0**,** 0**,** Math**.**cos**(**angle**));**

matrixRotate**.**setElementAt**(**0**,** 1**,** Math**.**sin**(**angle**));**

matrixRotate**.**setElementAt**(**1**,** 0**,** **-**Math**.**sin**(**angle**));**

matrixRotate**.**setElementAt**(**1**,** 1**,** Math**.**cos**(**angle**));**

matrixRotate**.**setElementAt**(**2**,** 2**,** 1.0**);**

matrixRotate**.**setElementAt**(**3**,** 3**,** 1.0**);**

**}**

matrix **=** multiplication**(**matrix**,** matrixRotate**);**

**return** matrix**;**

**}**

public Matrix perspective**(**Matrix matrix**)** **{**

recountMatrixPerspective**(**matrix**);**

matrix **=** multiplication**(**matrix**,** matrixPerspective**);**

**return** matrix**;**

**}**

public Matrix scaling**(**Matrix matrix**)** **{**

matrix **=** multiplication**(**matrix**,** multiplication**(**matrixMoving**,** matrixRotate**));**

**return** matrix**;**

**}**

private final Matrix matrixPerspective **=** **new** Matrix**(new** Double**[][]{**

**{**1.0**,** 0.0**,** 0.0**,** 0.0**},**

**{**0.0**,** 1.0**,** 0.0**,** 0.0**},**

**{**0.0**,** 0.0**,** 1.0**,** 0.0**},**

**{**0.0**,** 0.0**,** 0.0**,** 0.0**}**

**});**

private Matrix matrixMoving **=** **new** Matrix**(new** Double**[][]{**

**{**1.0**,** 0.0**,** 0.0**,** 0.0**},**

**{**0.0**,** 1.0**,** 0.0**,** 0.0**},**

**{**0.0**,** 0.0**,** 1.0**,** 0.0**},**

**{**0.0**,** 0.0**,** 0.0**,** 1.0**}**

**});**

private Matrix matrixRotate **=** **new** Matrix**(new** Double**[][]{**

**{**1.0**,** 0.0**,** 0.0**,** 0.0**},**

**{**0.0**,** 1.0**,** 0.0**,** 0.0**},**

**{**0.0**,** 0.0**,** 1.0**,** 0.0**},**

**{**0.0**,** 0.0**,** 0.0**,** 1.0**}**

**});**

private Matrix matrixReflect **=** **new** Matrix**(new** Double**[][]{**

**{**1.0**,** 0.0**,** 0.0**,** 0.0**},**

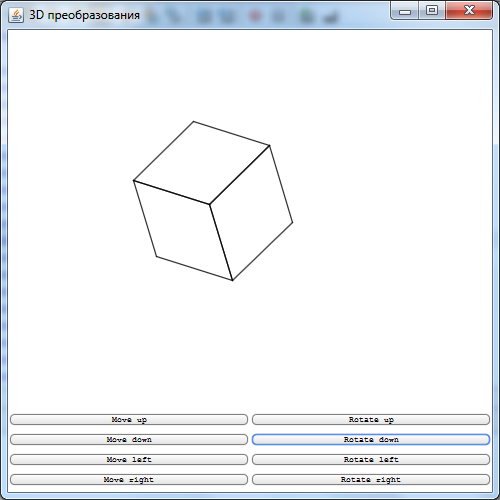
**{**0.0**,** 1.0**,** 0.0**,** 0.0**},**

**{**0.0**,** 0.0**,** 1.0**,** 0.0**},**

**{**0.0**,** 0.0**,** 0.0**,** 1.0**}**

**});**

**Пример работы**



**Выводы**

В ходе лабораторной работы были изучены основные механизмы реализации базовых геометрических операций, были реализованы методы для работы с матрицами, константные множители для каждого вида преобразований. Был реализован графический интерфейс, который позволяет производить все выше описанные операции с трехмерным кубом.