БЕЛОРУССКИЙ НАЦИОНАЛЬНЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

ФАКУЛЬТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ И РОБОТОТЕХНИКИ

Кафедра “Системы автоматизированного проектирования”

Пояснительная записка к курсовой работе

по курсу “3D-моделирование инженерных конструкций”

«Получение 3D геометрической модели объекта и её реалистичного изображения»

**Исполнитель:** студент гр.10702319

Бородовский А.А.

**Руководитель:** доц. Полозков Ю. В.

Минск 2021

**СОДЕРЖАНИЕ**

ВВЕДЕНИЕ………………………………………………………………………………………2

1. ОПИСАНИЕ ОБЪЕКТА………………………………………………………………...4
2. ПОЛУЧЕНИЕ ГЕОМЕТРИЧЕСКОЙ МОДЕЛИ, ОПИСАНИЕ СТРУКТУРЫ ДАННЫХ………………………………………………………………..5
3. ГЕОМЕТРИЧЕСКИЕ ПРЕОБРАЗОВАНИЯ…………………………………….8
4. ПО СТРОЕНИЕ ВСЕХ ВИДОВ ПРОЕКЦИЙ…………………………………..13
5. УДАЛЕНИЕ НЕВИДИМЫХ ЛИНИЙ…………………………………………….21
6. ЗАКРАСКА ОБЪЕКТА С УЧЕТОМ ОСВЕЩЕНИЯ ТОЧЕЧНЫМ ИСТОЧНИКОМ СВЕТА………………………………………………………………..24

ЗАКЛЮЧЕНИЕ……………………………………………………………………………......27

# ВВЕДЕНИЕ

До недавнего времени компьютерная графика представляла собой весьма специфическое занятие, требующее дорогостоящей аппаратуры, значительных ресурсов памяти и своеобразного программного обеспечения.

В настоящее время сформировалась новая отрасль информатики – компьютерная графика. Ее можно определить как науку о математическом и геометрическом моделировании форм и размеров объектов, а также методов их визуализации.

Компьютерная графика – наиболее важная и наиболее динамично развивающаяся область информационных технологий. Она находит свое применение в различных сферах человеческой деятельности: от компьютерных игр и спецэффектов для кино до сложных научных симуляций.

По способам задания изображений компьютерную графику можно разделить на двумерную и трехмерную.

Двумерная компьютерная графика классифицируется по типу представления графической информации, и следующими из него алгоритмами обработки изображений. Обычно компьютерную графику разделяют на векторную и растровую.

Векторная графика — способ представления объектов и изображений в компьютерной графике, основанный на математическом описании элементарных геометрических объектов, обычно называемых примитивами, таких как: точки, линии, сплайны, круги и окружности, многоугольники. Объекты векторной графики являются графическими изображениями математических объектов. Термин «векторная графика» используется для пояснения различий от растровой графики, в которой изображение представлено в виде графической матрицы. При выводе на матричные устройства отображения (мониторы) векторная графика предварительно преобразуется в растровую графику, преобразование производится программно или аппаратно средствами современных видеокарт.

Растровая графика. Наиболее просто реализовать растровое представление изображения. Растр, или растровый массив (bitmap), представляет совокупность битов, расположенных на сетчатом поле-канве. Бит может быть включен (единичное состояние) или выключен (нулевое состояние). Состояния битов можно использовать для представления черного или белого цветов, так что, соединив на канве несколько битов, можно создать изображение из черных и белых точек.

Растровое изображение напоминает лист клетчатой бумаги, на котором каждая клеточка закрашена черным или белым цветом, в совокупности формируя рисунок.

Основным элементом растрового изображения является пиксел (рixel). Под этим термином часто понимают несколько различных понятий: отдельный элемент растрового изображения, отдельная точка на экране монитора, отдельная точка на изображении, напечатанном принтером.

В трёхмерной компьютерной графике все объекты обычно представляются как набор поверхностей или частиц. Минимальную поверхность называют полигоном.

Любой полигон можно представить в виде набора из координат его вершин. Координаты каждой вершины представляют собой вектор (x, y, z). Умножив вектор на соответствующую матрицу, мы получим новый вектор. Сделав такое преобразование со всеми вершинами полигона, получим новый полигон, а преобразовав все полигоны, получим новый объект, повёрнутый/сдвинутый/масштабированный относительно исходного.

Задача трехмерного моделирования состоит в том, что нужно определить структуру данных, описывающую трехмерное тело, на основе этой структуры получить возможность изменять объект, изменять его положение в пространстве. Также к задачам трехмерного моделирования относятся задачи проекционного черчения, реализации реалистического изображения объекта с учетом таких факторов как свет, материал, шероховатость поверхностей и т.д.

## 1. ОПИСАНИЕ ОБЪЕКТА

Объект состоит из конуса и соосной пирамиды с общим основанием, как показано на рисунке 1:

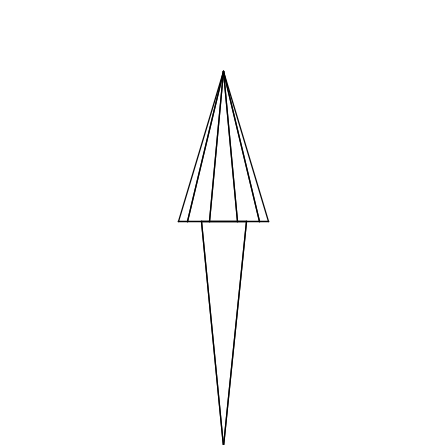


Рисунок 1 – Модель объекта

В данной курсовой работе, реализована возможность изменения геометрических параметров конуса и пирамиды: высоты (h1 и h2), радиуса (R2), стороны основания пирамиды (а) и степени аппроксимации (k), что показано на рисунке 2:

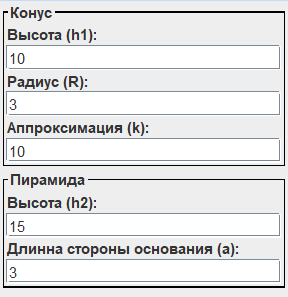


Рисунок 2 – Размеры фигур

## 2. ПОЛУЧЕНИЕ ГЕОМЕТРИЧЕСКОЙ МОДЕЛИ, ОПИСАНИЕ СТРУКТУРЫ ДАННЫХ

### **2.1 Выбор мировой системы координат**

Для удобства работы с координатами вершин, систему координат расположим в центре основания конуса и пирамиды:

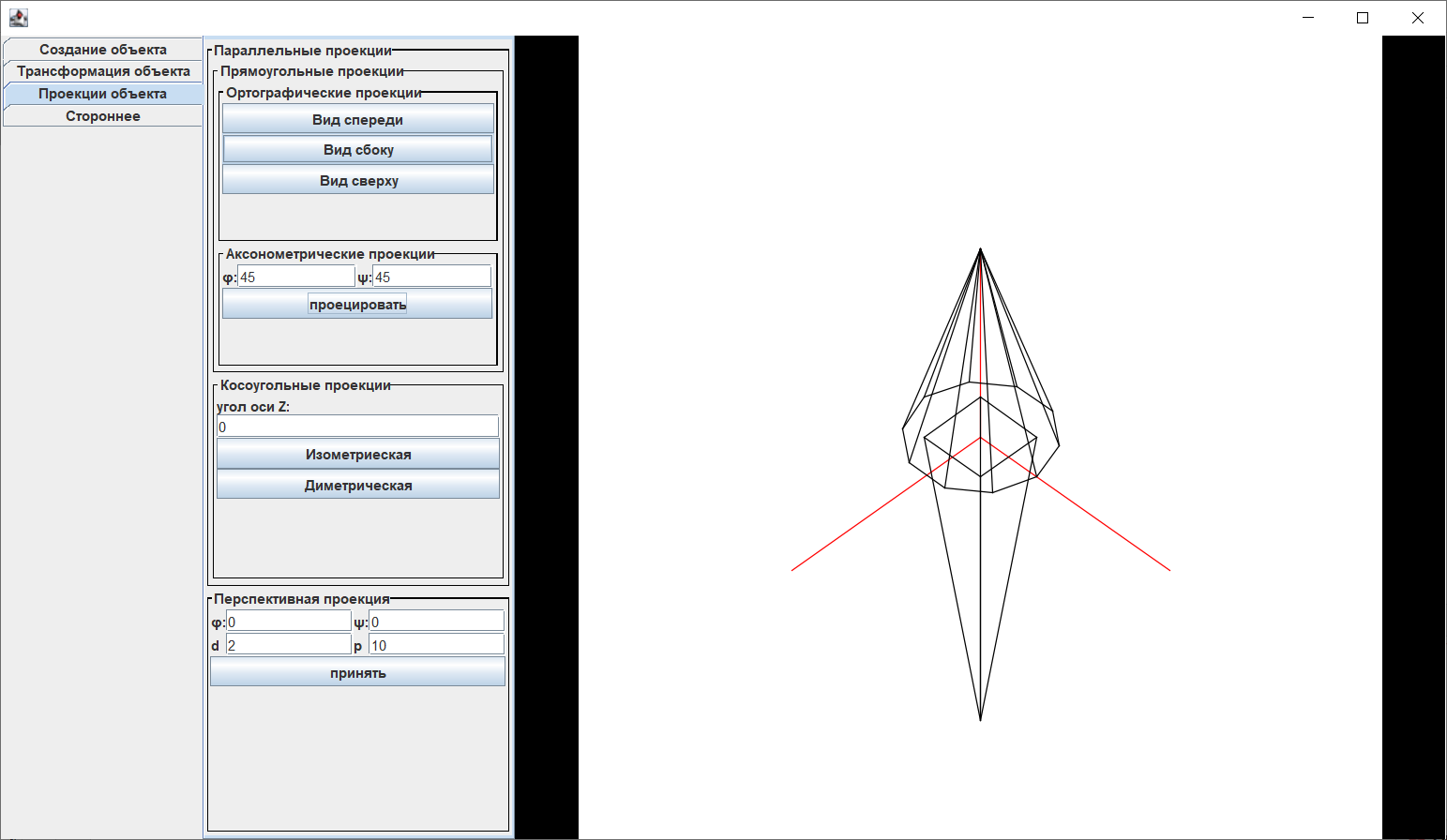


Рисунок 3 – Расположение системы координат

### **2.2 Расчёт координат**

Координаты оснований конуса рассчитываются по формулам:

Пример расчета координат основания конуса при степени аппроксимации k = 3 и R = 10:

[10, 0, 0, 1]

[5, 0, 8.66, 1]

Функция расчета координат основания конуса в программе реализовано следующим образом:

addPoint(**new** Point3D(**0**,h1,**0**));

**int**[] base = **new** **int**[k];

**double** angle = Math.toRadians(**360**f/k);

**for** (**int** i = **0**; i<k; i++){

addPoint(**new** Point3D(Math.cos(angle\*i)\*r,**0**,Math.sin(angle\*i)\*r));

addLine(**new** Line3D(**6**,**7**+i));

base[i] = **7**+i;

**if** (i>**0**){

addLine(**new** Line3D(**6**+i,**7**+i));

addPoly(**new** Poly3D((**7**+i),(**6**+i),(**6**)));

}

**if** (i == k-**1**){

addLine(**new** Line3D(**7**+i,**7**));

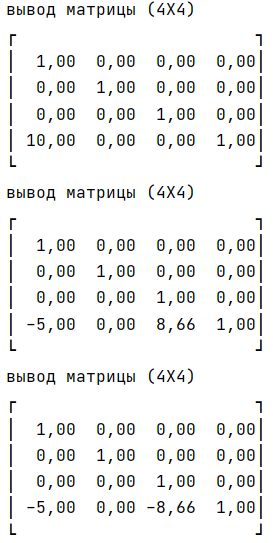
addPoly(**new** Poly3D((**7**),(**7**+i),(**6**)));

}

}

addPoly(**new** Poly3D(base));

Результат выполнения программы, точки описываются матрицей 4X4:



Главное окно программы с примером построения объекта представлено на рисунке 4:

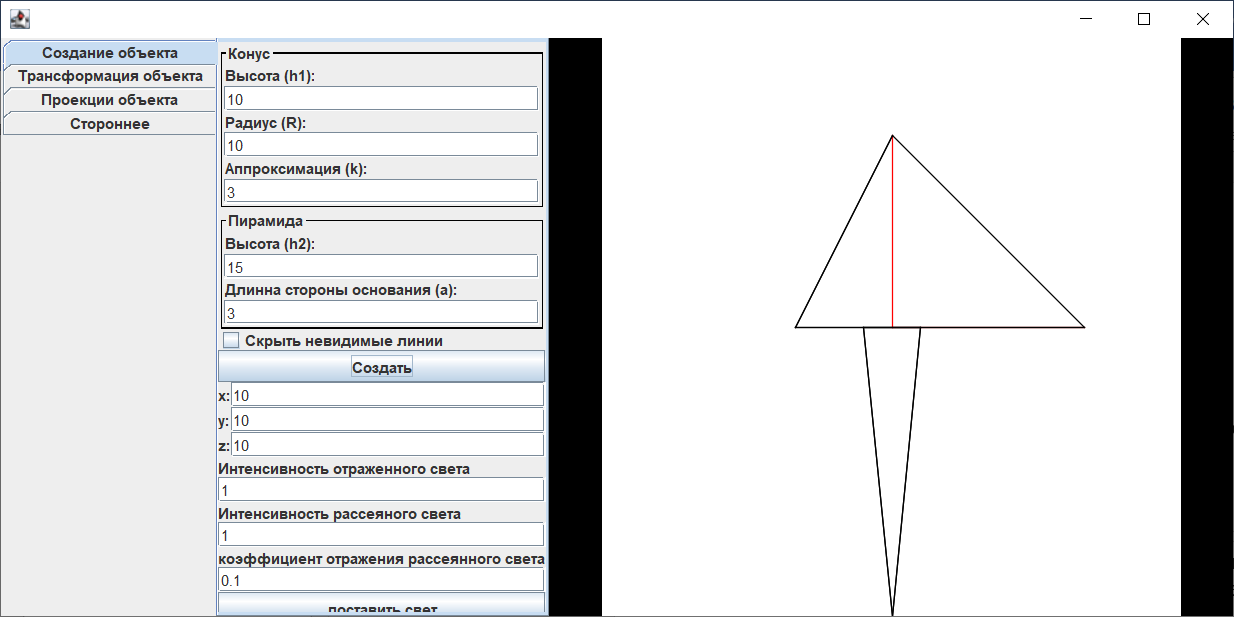


Рисунок 4 – Основное окно с исходным изображением объекта

# 3. ГЕОМЕТРИЧЕСКИЕ ПРЕОБРАЗОВАНИЯ

Для возможности геометрических преобразований модели объекта необходимо применять соответствующие матрицы преобразований. Для перемножения матриц пишем отдельную функцию:

public static Matrix multiplication(Matrix matrix1c, Matrix matrix2c){  
 Matrix matrix1 = new Matrix(matrix1c);  
 Matrix matrix2 = new Matrix(matrix2c);  
 if (matrix1.getColumns() != matrix2.getLines()){  
 throw new IllegalArgumentException("введены неумножаемые матрицы");  
 }  
 Matrix rez = new Matrix(matrix1.getLines(), matrix2.getColumns());  
 double element;  
 for (int i = 0; i < matrix1.getLines(); i++)  
 for (int j = 0; j < matrix2.getColumns(); j++){  
 element = 0;  
 for (int k = 0; k < matrix1.getColumns(); k++){  
 element = element + matrix1.getValue(i, k) \* matrix2.getValue(k, j);  
 }  
 rez.setValue(i,j,element);  
 }  
 return rez;  
}

## Перенос

Матрица переноса выглядит следующим образом:





.

где Dx - значение переноса относительно оси *OX*, Dy - значение переноса относительно оси *OY*, Dz - значение переноса относительно оси *OZ*.

Пример расчета координат большего основания при степени аппроксимации k = 3 при сдвигах по осям Dx = 5, Dy = 10, Dz = 15.

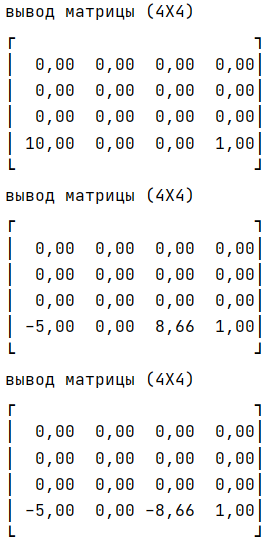
Исходные координаты 3-ех точек основания до перемещения:

В результате перемножения матриц получаем координаты после перемещения:

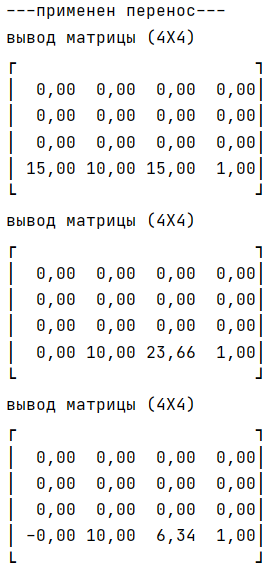
В программе данная матрица реализована следующим образом:

public RepositionMatrix(double x, double y, double z){  
 super(4,4,  
 1, 0, 0, 0,  
 0, 1, 0, 0,  
 0, 0, 1, 0,  
 x, y, z, 0);  
}

Пример выполнения программы. Координаты основания конуса при степени аппроксимации k = 3 до перемещения:



После перемещения:



Результат выполнения представлен на рисунке 5:

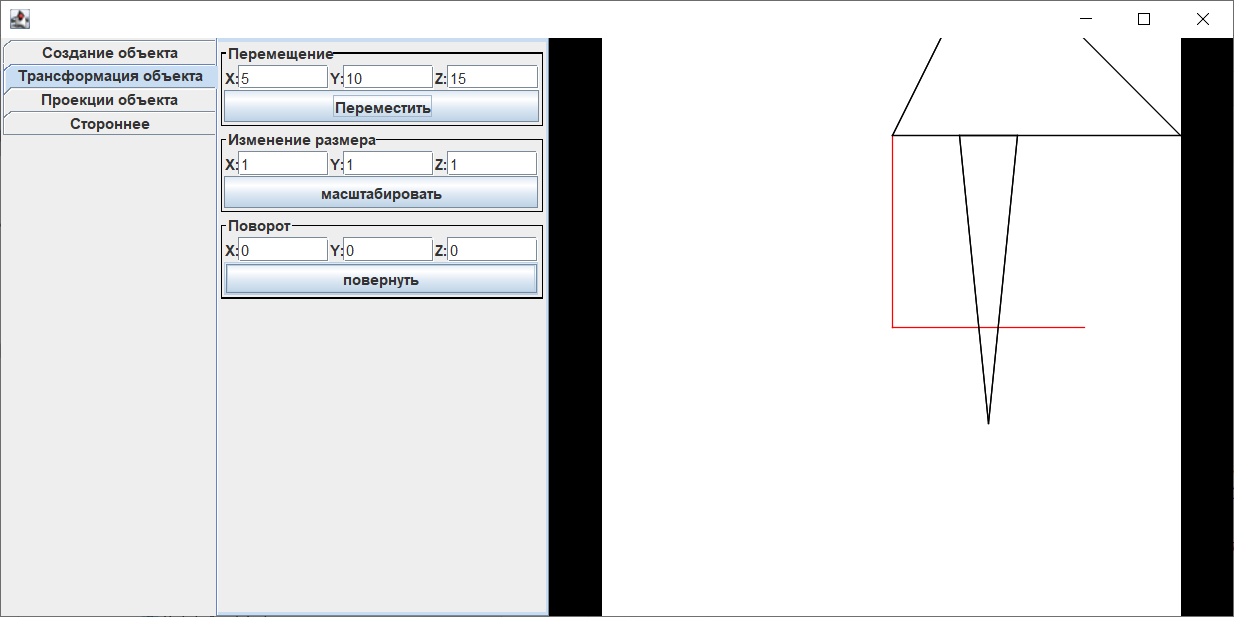


Рисунок 5 – Пример переноса объекта

## 3.2 Масштабирование

Матрица масштабирования выглядит следующим образом:





или

.

где double Sx - коэффициент масштабирования относительно оси *OX*, double Sy - коэффициент масштабирования относительно оси *OY*, double Sz - коэффициент масштабирования относительно оси *OZ*.

Пример расчета координат большего основания при степени аппроксимации k = 3 при масштабировании по осям на Sx = 1.5, Sy = 2, Sz = 0.5.

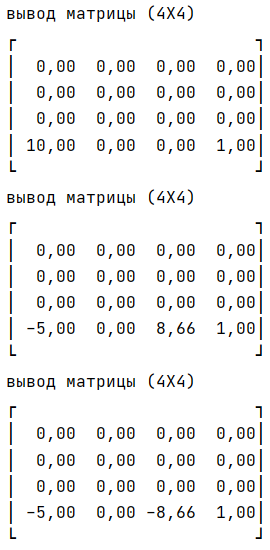
Исходные координаты 3-ех точек основания до масштабирования:

В результате перемножения матриц получаем координаты после масштабирования:

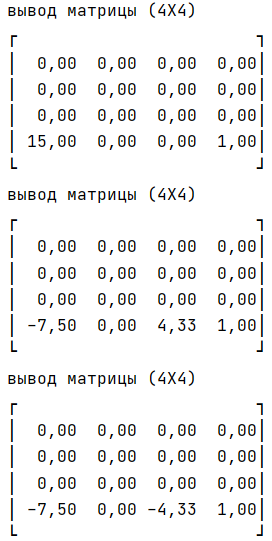
В программе данная матрица реализована следующим образом:

public SizeMatrix(double x, double y, double z){  
 super(4,4,  
 x, 0, 0, 0,  
 0, y, 0, 0,  
 0, 0, z, 0,  
 0, 0, 0, 1);  
}

Пример выполнения программы. Координаты большего основания при степени аппроксимации k = 3 до масштабирования:



После масштабирования:



Результат выполнения представлен на рисунке 6:

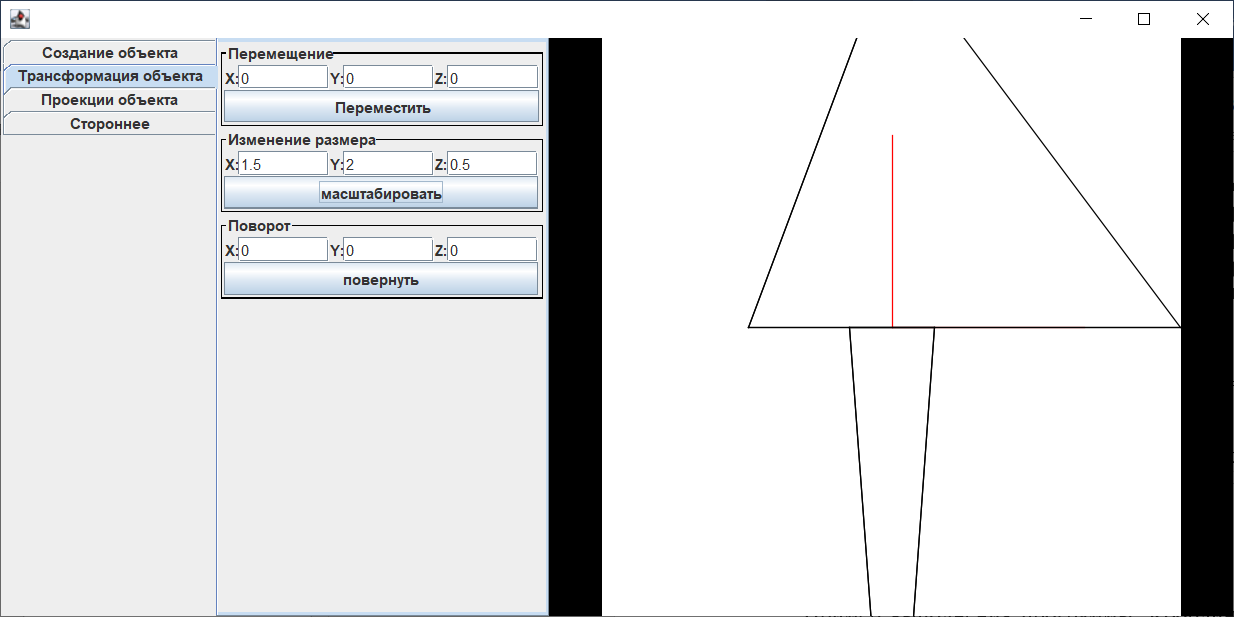


Рисунок 6 – Пример масштабирования объекта

## 3.3 Поворот

Поворот вокруг осей осуществляется с помощью алгоритма написсаного в слушателе кнопки rotate:

@Override  
public void actionPerformed(ActionEvent e) {  
 try {  
 double x = Double.*parseDouble*(rotX.getText());  
 double y = Double.*parseDouble*(rotY.getText());  
 double z = Double.*parseDouble*(rotZ.getText());  
 RotateMatrix matrix= new RotateMatrix(x,y,z);  
 Obj3D obj3D = SimpleStore.*getObjByName*("MainObj");  
 assert obj3D != null;  
 double centerX = obj3D.getCenter().getX();  
 double centerY = obj3D.getCenter().getY();  
 double centerZ = obj3D.getCenter().getZ();  
 RepositionMatrix repositionMatrix = new RepositionMatrix(-centerX,-centerY,-centerZ);  
 obj3D.matrixSum(repositionMatrix);  
 obj3D.matrixMulty(matrix);  
 repositionMatrix = new RepositionMatrix(centerX, centerY ,centerZ);  
 obj3D.matrixSum(repositionMatrix);  
 paintPanel.repaint();  
 } catch (Exception exception) {  
 System.*out*.print(exception);  
 }  
}

Матрица поворота вокруг оси :

,

где θ - угол поворота относительно оси *OX*

Матрица поворота вокруг оси :

,

где θ - угол поворота относительно оси *OY.*

Матрица поворота вокруг оси Z:

,

где θ - угол поворота относительно оси *OZ.*

Пример расчета координат большего основания при степени аппроксимации k = 3 при поворотах относительно осей *OX, OY* и *OZ* на 45.

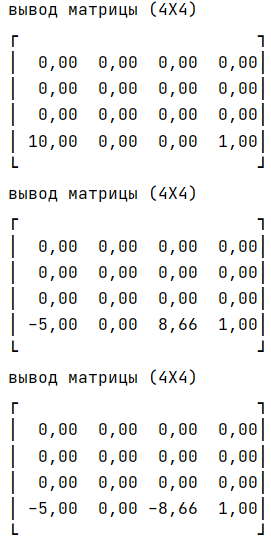
Координаты до поворота:

В результате перемножения матриц получаем координаты после поворота:

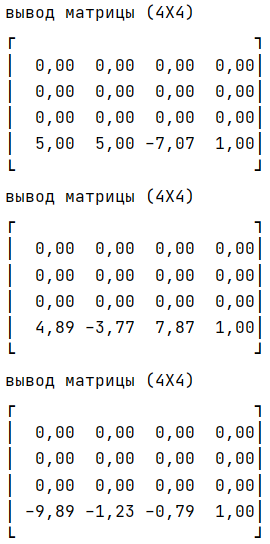
В программе данные матрицы реализованы следующим образом:

public RotateMatrix(double x, double y, double z){  
 super(4,4);  
 double grX = Math.*toRadians*(x);  
 double grY = Math.*toRadians*(y);  
 double grZ = Math.*toRadians*(z);  
 Matrix rotX = new Matrix(4,4,  
 1,0,0,0,  
 0,Math.*cos*(grX),Math.*sin*(grX),0,  
 0,-Math.*sin*(grX),Math.*cos*(grX),0,  
 0,0,0,1);  
 Matrix rotY = new Matrix(4,4,  
 Math.*cos*(grY),0,-Math.*sin*(grY),0,  
 0,1,0,0,  
 Math.*sin*(grY),0,Math.*cos*(grY),0,  
 0,0,0,1);  
 Matrix rotZ = new Matrix(4,4,  
 Math.*cos*(grZ),Math.*sin*(grZ),0,0,  
 -Math.*sin*(grZ),Math.*cos*(grZ),0,0,  
 0,0,1,0,  
 0,0,0,1);  
 Matrix rez;  
 rez = Matrix.*multiplication*(rotX,rotY);  
 rez = Matrix.*multiplication*(rez,rotZ);  
 setValues(rez.getValues());  
}

Пример выполнения программы. Координаты большего основания при степени аппроксимации k = 3 до поворота:



После поворота:



Результат выполнения представлени на Рисунке 7:

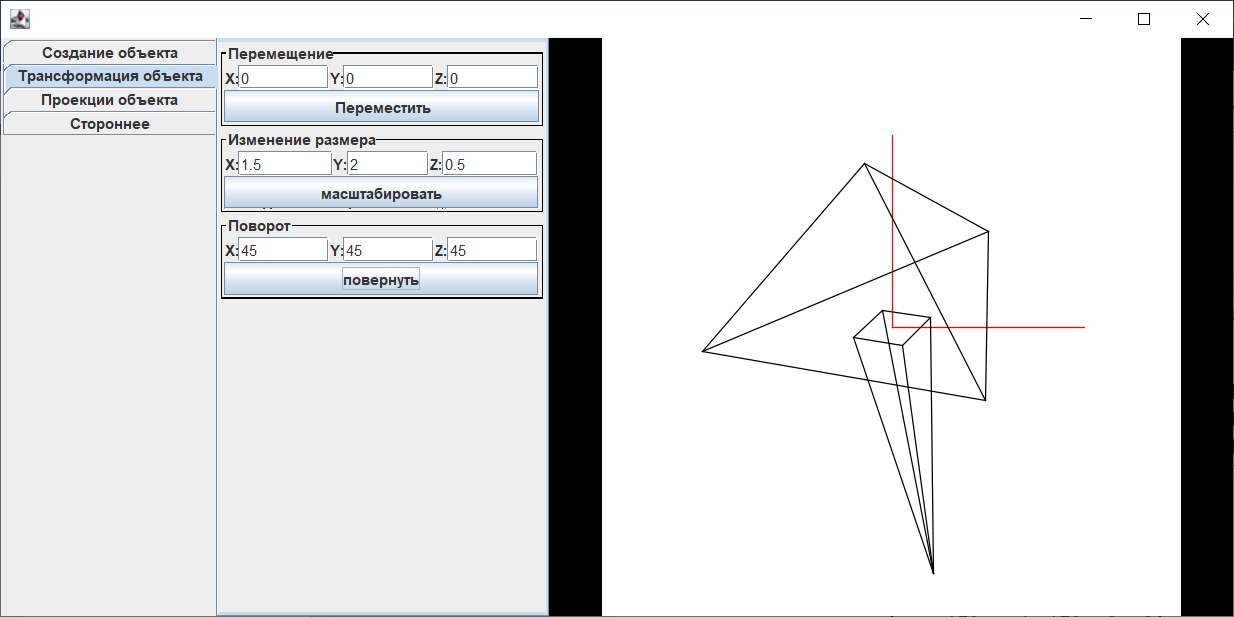


Рисунок 7 – Пример поворота объекта

# ПОЛУЧЕНИЕ ВСЕХ ВИДОВ ПРОЕКЦИЙ

ПРОЕКИИ

ПАРАЛЛЕЛЬНЫЕ

ЦЕНТРАЛЬНЫЕ

ПРЯМОУГОЛЬНЫЕ

КОСОУГОЛЬНЫЕ

ОРТОГРАФИЧЕСКИЕ

АКСОНОМЕТРИ-ЧЕСКИЕ

ВИД СПЕРЕДИ

ВИД СВЕРХУ

ВИД СБОКУ

ИЗОМЕТРИЧЕСКАЯ

ПРЯМОУГ. ДИМЕ-ТРИЧЕСКАЯ

ТРИМЕТРИЧЕСКАЯ

КОСОУГ. ДИМЕТРИ-ЧЕСКАЯ (cabinet)

КОСОУГ. ИЗОМЕТРИЯ (cavalier)

В данном проекте реализована возможность получения следующих видов проекций: фронтальной, горизонтальной, профильной, аксонометрической, косоугольной и перспективной проекций.

## Аксонометрическая проекция

Начать стоит с аксонометрической проекции, т.к. от нее мы в будущем еще будем от нее отталкиваться.

Аксонометрическая проекция получается при повороте на угол относительно оси Y и на угол  относительно оси Х с последующим проецированием вдоль оси Z.

Матрицы точек объекта надо умножить на матрицу аксонометрической проекции:

При выборе аксонометрической проекции необходимо ввести углы  и .

Пример расчета координат большего основания при степени аппроксимации n = 3 для аксонометрической проекции при углах и :

В программе построение фронтальной проекции имеет вид:

public AcsonomProjection(double fi, double psi) {  
 super(4,4);  
 RotateMatrix rotateMatrix1 = new RotateMatrix(0,-psi,0);  
 RotateMatrix rotateMatrix2 = new RotateMatrix(fi,0,0);  
 Matrix matrix = new Matrix(Matrix.*multiplication*(rotateMatrix1,rotateMatrix2));  
 setValues(matrix.getValues());  
}

Пример выполнения программы. Координаты большего основания при степени аппроксимации k = 3 в аксонометрической проекции:

Результат выполнения представлен на рисунке 11:

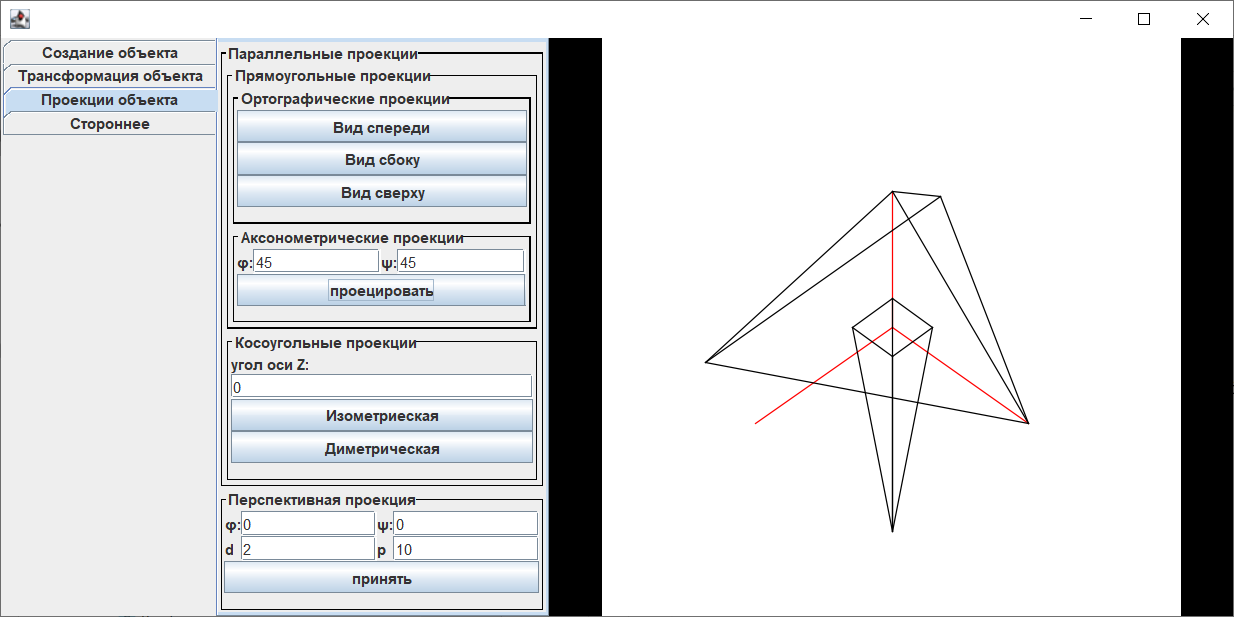


Рисунок 11 – Аксонометрическая проекция объекта

## 4.1 Фронтальная проекция

При построении фронтальной проекции принимаем, что наша проекционная плоскость – Z=0, а направление проецирования совпадает с направлением оси Z (  ). И для получения **фронтальной** проекции объекта необходимо строить объект по координатам X, Y. Ортографические проекции можно получить путем применения аксонометрической, как это показанно в программе (применение проекции закреплено за кнопкой):

@Override  
public void actionPerformed(ActionEvent e) {  
 Painter.*setOblique*(false);  
 Painter.*setModMatrix*(new AcsonomProjection(0,0));  
 paintPanel.repaint();  
}

Примерно такой же код прикреплен и к другим кнопкам с различными значениями в функцию AcsonomProjection

Результат построения фронтальной проекции при степени аппроксимации k = 3 после поворота объекта на по всем осям изображен на рисунке 8:

## 

Рисунок 8 – Фронтальная проекция объекта

## 4.2 Профильная проекция

Для получения **профильной** проекции объекта необходимо строить объект по координатам Z, Y (за проекционную плоскость X=0, направление проецирования – направление оси X). В программе построение фронтальной проекции глобального поворота сцены:

Результат построения профильной проекции при степени аппроксимации k = 3 после поворота объекта на по всем осям изображен на рисунке 9:

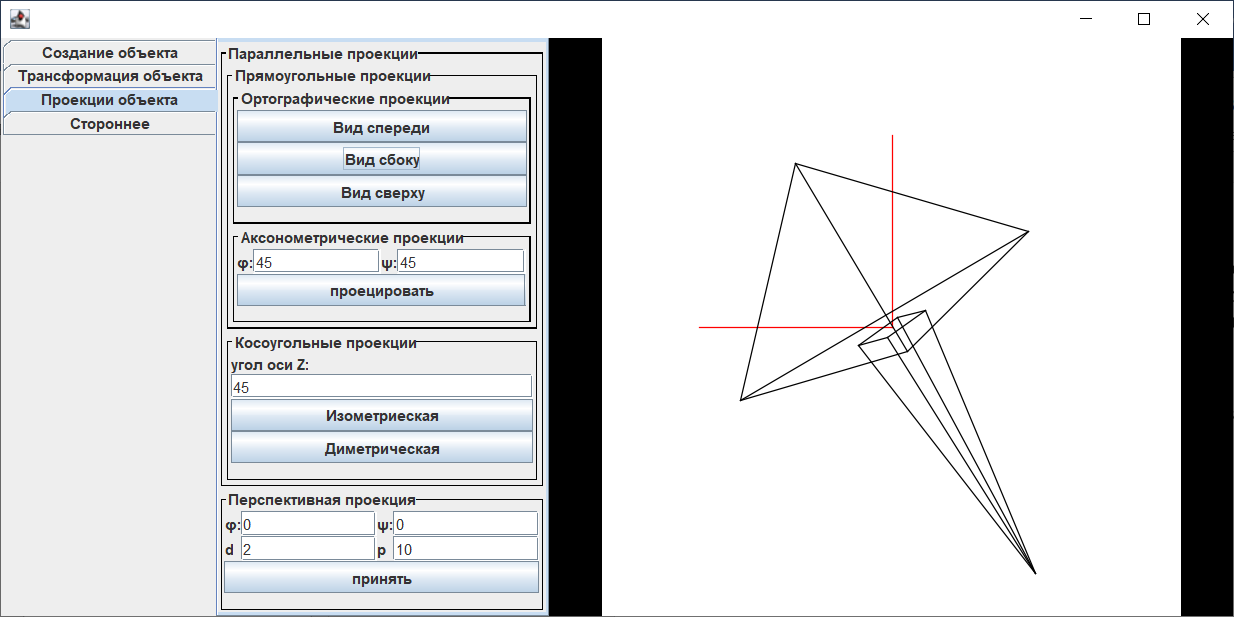


Рисунок 9 – Профильная проекция объекта

## 4.3 Горизонтальная проекция

Для получения **горизонтальной** проекции объекта необходимо строить объект по координатам X, Z (в данном случае мы принимаем за проекционную плоскость Y=0, а за направление проецирования – направление оси Y). В программе построение фронтальной так же получается путем поворота сцены:

Результат построения горизонтальной проекции при степени аппроксимации k = 3 после поворота объекта на по всем осям изображен на рисунке 10:

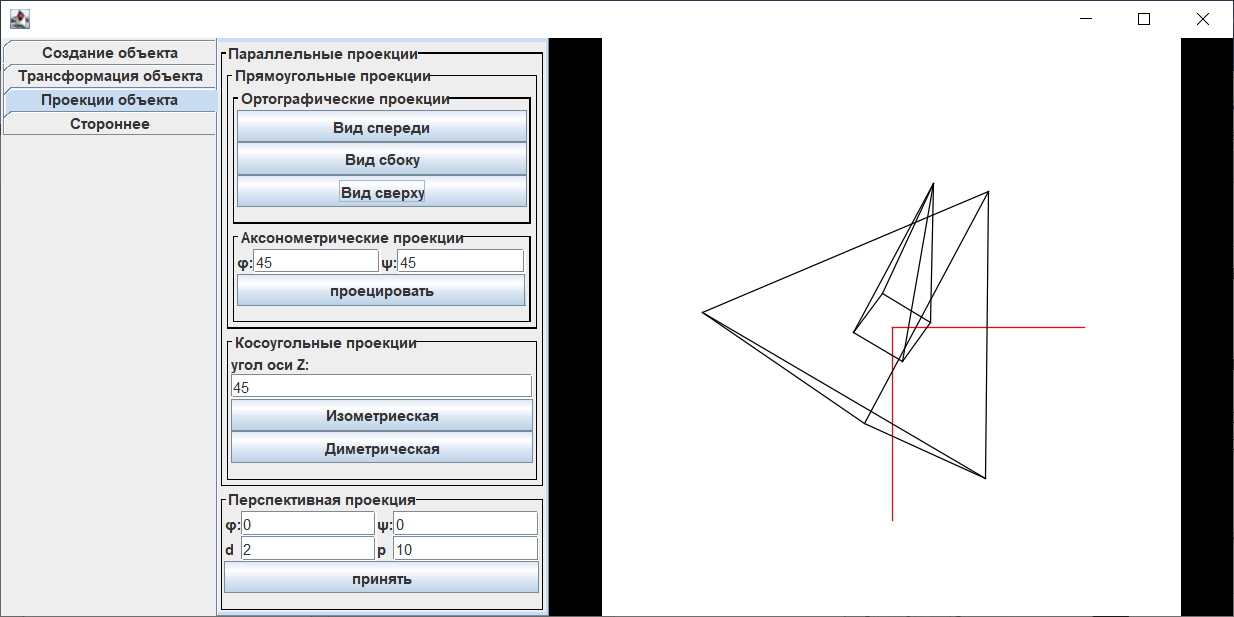


Рисунок 10 – Горизонтальная проекция объекта

## Косоугольная проекция

Для построения косоугольной проекции матрицу точек нашего объекта надо умножить на матрицу косоугольной проекции:

.

При этом необходимо ввести параметры L и угол α. Значение данных примеров видно из данного рисунка.

27

Применение матрицы  приводит к сдвигу и последующему проецированию объекта. Плоскости с постоянной координатой *z=z1* переносятся в направлении *х* на *,* в направлении *y* — на ** и затем проецируется на плоскость *z=0,* однако мы сохраним значения z для вохможности в последующем удалять невидимые линии*.* Сдвиг сохраняет параллельность прямых, а также углы и расстояния в плоскостях, параллельны оси *z*.

Пример расчета координат большего основания при степени аппроксимации k = 3 для косоугольной проекции при углах и :

Пример выполнения программы. Координатыоснования при степени аппроксимации n = 3 в косоугольной проекции:

Результат выполнения представлен на рисунке 12:

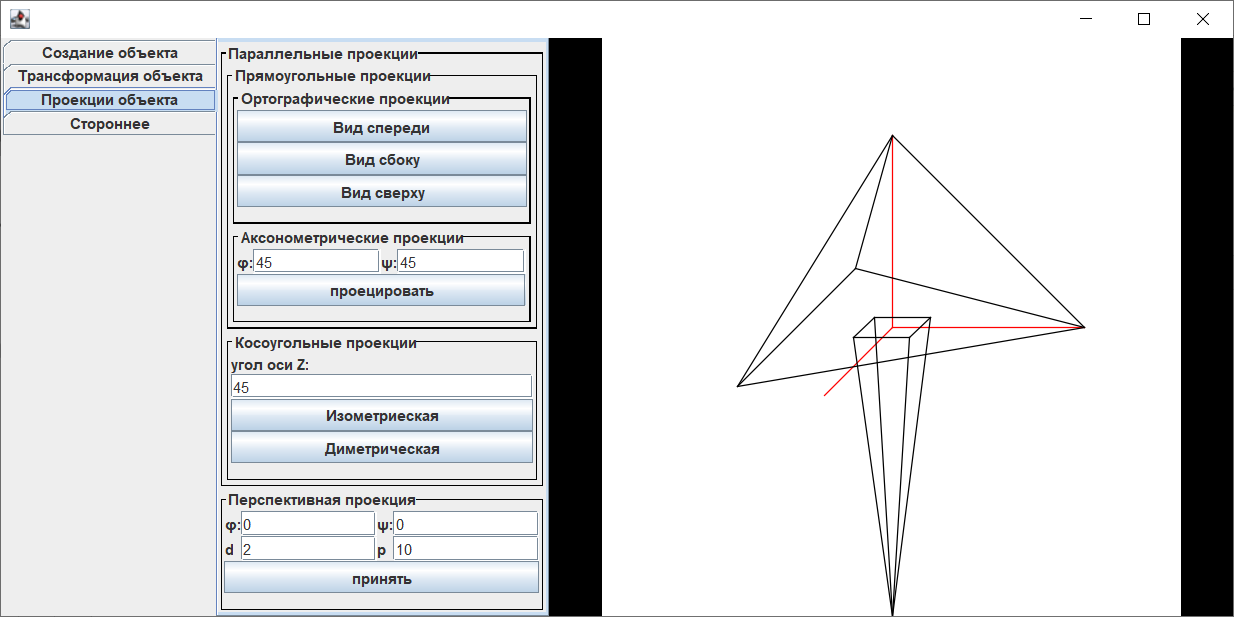


Рисунок 12 – Косоугольная проекция объекта

## 4.6 Перспективная проекция

Для получения перспективной проекции необходимо разделить на *Z* все точки объекта и умножить их на *d,* где *d –* своеобразный масштабный коэффициент, расстояние между точкой наблюдения и экраном. Это расстояние является фактором, приводящим к тому, что удаленные объекты выглядят мельче. Допустимы все значения *z*, кроме *z=0*.

Но существует еще так называемое видовое преобразование. Для этого мы переносим точку наблюдении и задаем ее в сферических координатах. Далее необходимо преобразовать мировую систему координат в видовую, то есть задача сводится к преобразованию СК.

Результирующая матрица преобразования имеет следующий вид:



*ϕ* - угол поворота относительно оси *OX*

*θ* - угол поворота относительно оси *OZ*

*ρ*- расстояние от центра мировой системы координат до точки наблюдения

d- масштабный коэффициент (расстояние между точкой наблюдения и экраном).

После применения видового преобразования применяем матрицу аксонометрической проекции, которая в программе имеет вид:

public PerspectiveProjection (double d, double p){  
 super(4,4,  
 1,0,0,0,  
 0,1,0,0,  
 0,0,1,-1/d,  
 0,0,0,1);  
}

Пример расчета координат основания при степени аппроксимации k = 3 для перспективной проекции после поворота по всем осям на при расстоянии от центра мировой системы координат до точки наблюдения ρ = 15*,* масштабном коэффициенте d = 15, угле поворота относительно оси *OZ θ =* и угле поворота относительно оси *OX ϕ = 30*.

Начальные координаты:

В результате перемножения матриц получаем координаты после видового преобразования, а после и самой проекции:

Пример выполнения программы. Координаты узловых точек большего основания при степени аппроксимации n = 3 в перспективной проекции после поворота по всем осям на и сдвига по оси *OZ* на 4 после видового преобразования при расстоянии от центра мировой системы координат до точки наблюдения ρ = 15*,* масштабном коэффициенте d = 15, угле поворота относительно оси *OZ θ =* и угле поворота относительно оси *OX ϕ = 30*:

Результат выполнения представлен на рисунке 13:

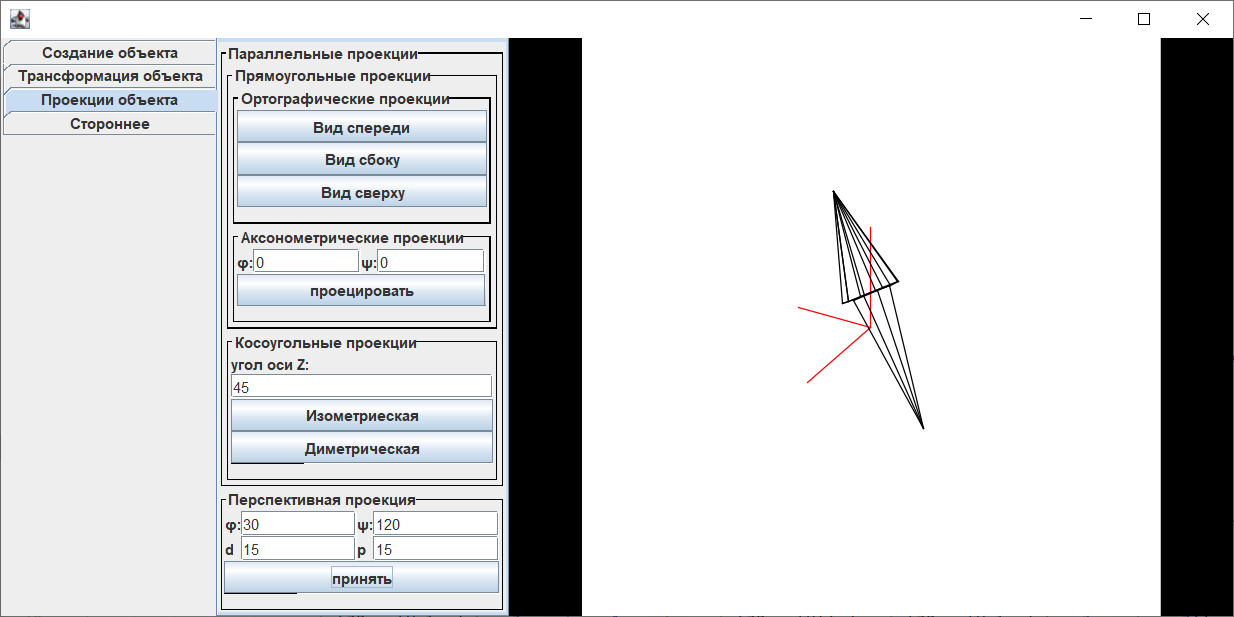


Рисунок 13 – Перспективная проекция объекта

# УДАЛЕНИЕ НЕВИДИМЫХ ЛИНИЙ

В курсовой работе для удаления невидимых линий осуществлялось прорисовывая сначала самые дальние грани.

Алгоритм сортировки по глубине

(список приоритетов, алгоритм художника)

Основная идея алгоритма художника: отсортировать многоугольники (грани объектов) в соответствии с их удаленностью от точки наблюдения, затем разместить многоугольники (раскладывая их в растр) в порядке убывания расстояния, т.е. ближние многоугольники помещаются в буфер кадра последними и закрашивают дальние.

Аналогичен способу создания картины художником: рисуется фон, предметы дальнего, среднего и, наконец, переднего плана. Простой, но ресурсоемкий алгоритм с избыточным выводом графической информации.

Алгоритм Ньюэл М., Ньюэл Р., Санча:

1. Упорядочить все многоугольники в соответствии с min Z-координатами.

2. Разрешить все неопределенности с Z-оболочками и сформировать окончательный список (из которого будут исключены невидимые многоугольники).

3. Преобразовать каждый из многоугольников в растровую форму в порядке убывания Z-координаты (можно методами заполнения) в соответствии с правилами закраски.

При сложных пересечениях многоугольников возможны неопределенности.

Алгоритм проверки: для простоты будем считать, что рассматривается па­раллельное проектирование вдоль оси Oz, грань Р находится в конце списка (самая удаленная). Перед выводом грани Р следует убедиться, что никакая другая грань Q, проекция которой на ось Oz пересекается с проекцией грани Р, не может закрываться гранью Р. И если это условие выполнено, то грань Р должна быть выведена раньше.

Предлагаются следующие 5 тестов в порядке возрастания сложности проверки:

1. Пересекаются ли проекции этих граней на ось Ох?
2. Пересекаются ли их проекции на ось Оу?
3. Находится ли грань Р по другую сторону от плоскости, проходящей через грань Q, чем начало координат (наблюдатель)?
4. Находится ли грань Q по ту же сторону от плоскости, проходящей через грань Р, что и начало координат (наблюдатель)?
5. Пересекаются ли проекции этих граней на картинной плоскости?

Если хотя бы на один из этих вопросов получен отрицательный ответ, то считается что эти две грани (Р и Q) упорядочены верно и сравниваем Р со следующей гранью. В противном случае считаем, что эти грани необходимо поменять местами, для чего проверяются следующие тесты:

3'. Находится ли грань Q по другую сторону от плоскости, проходящей через грань Р, чем начало координат?

4'. Находится ли грань Р по ту же сторону от плоскости, проходящей через грань Q, что и начало координат?

В случае если ни один из этих тестов не позволяет с уверенностью решить, какую из этих двух граней нужно выводить раньше, то одна из них разбивается на две грани плоскостью, проходящей через другую грань. В этом случае вопрос об упорядочении оставшейся грани и частей разбитой грани легко решается.

Результат выполнения программы после поворота объекта по всем осями на угол при степени аппроксимации k = 3 представлен на рисунке 14:

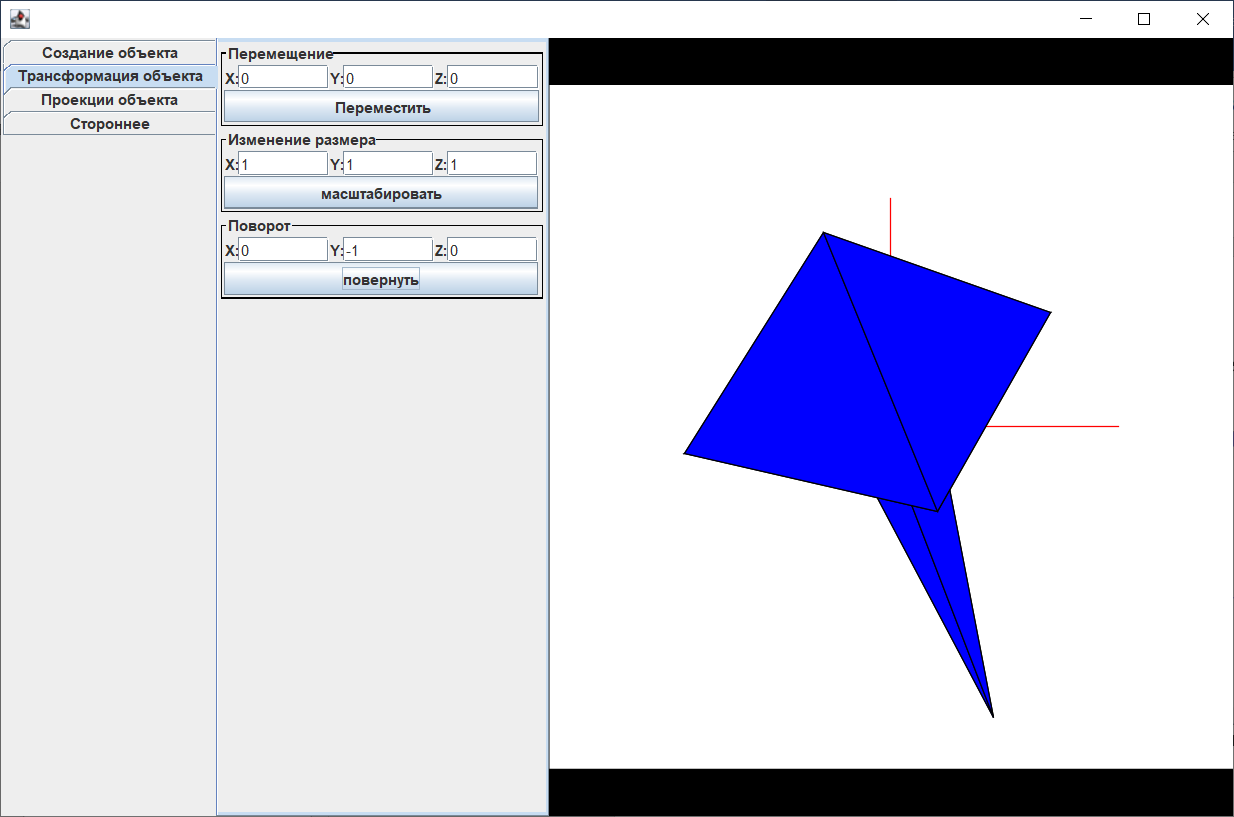


Рисунок 14 – Результат удаления невидимых линий

# ЗАКРАСКА ОБЪЕКТА С УЧЕТОМ ОСВЕЩЕНИЯ

Модель освещения представлена диффузным отражением, при котором поверхности имеют одинаковую яркость независимо от угла обзора.

Свет точечного источника отражается от поверхности по закону Ламберта рисунок 7.1:

83

Рисунок 7.1 Диффузное отражение

**

где  - интенсивность отраженного света,  - интенсивность точечного источника,  - коэффициент диффузного отражения(),  - угол между направлением света и нормалью к поверхности.

В реальной ситуации на объекты падает еще и рассеянный свет, отраженный от окружающей обстановки, например, от стен комнаты, других предметов.

Тогда интенсивность с учетом рассеянного света и диффузного отражения равна:



где  - интенсивность рассеянного света,  - коэффициент отражения рассеянного света .

Алгоритм работает на основе вычисления угла между нормалью грани и лучом источника света. Косинус данного угла рассчитывается следующим образом:

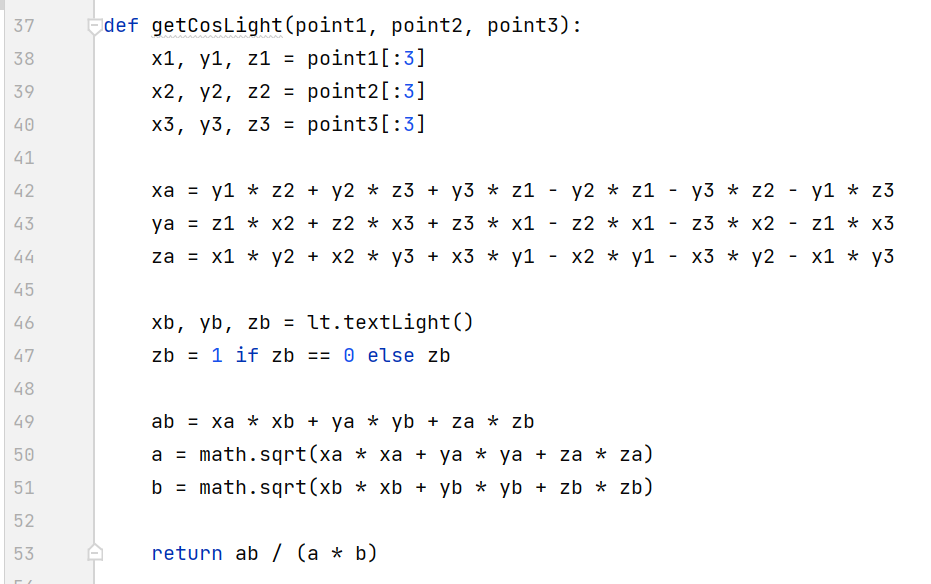
где - вектор к источнику света, – вектор нормали к грани.

где x1, y1, z1-координаты вектора , x2, y2, z2-координаты вектора .

Грань считается видимой, если угол между нормалью грани и вектором источника света находится в диапазоне [0°; 90°].Расчет интенсивностей для каждой грани проводится перед отрисовкойcучетом этого угла.

Координаты расположения источника света, интенсивность отражённого света и рассеянного, коэффициенты диффузного отражения и рассеянного, задаёт сам пользователь

В программе метод закраски граней с учетом освещения реализован следующим образом. Вычисление косинуса угла между нормалью грани и лучом источника света:



Результат выполнения программы после поворота объекта по оси OY на угол при степени аппроксимации k = 12 и расположении источника света в точке с координатами (10, 10, 5) представлен на рисунке 15:

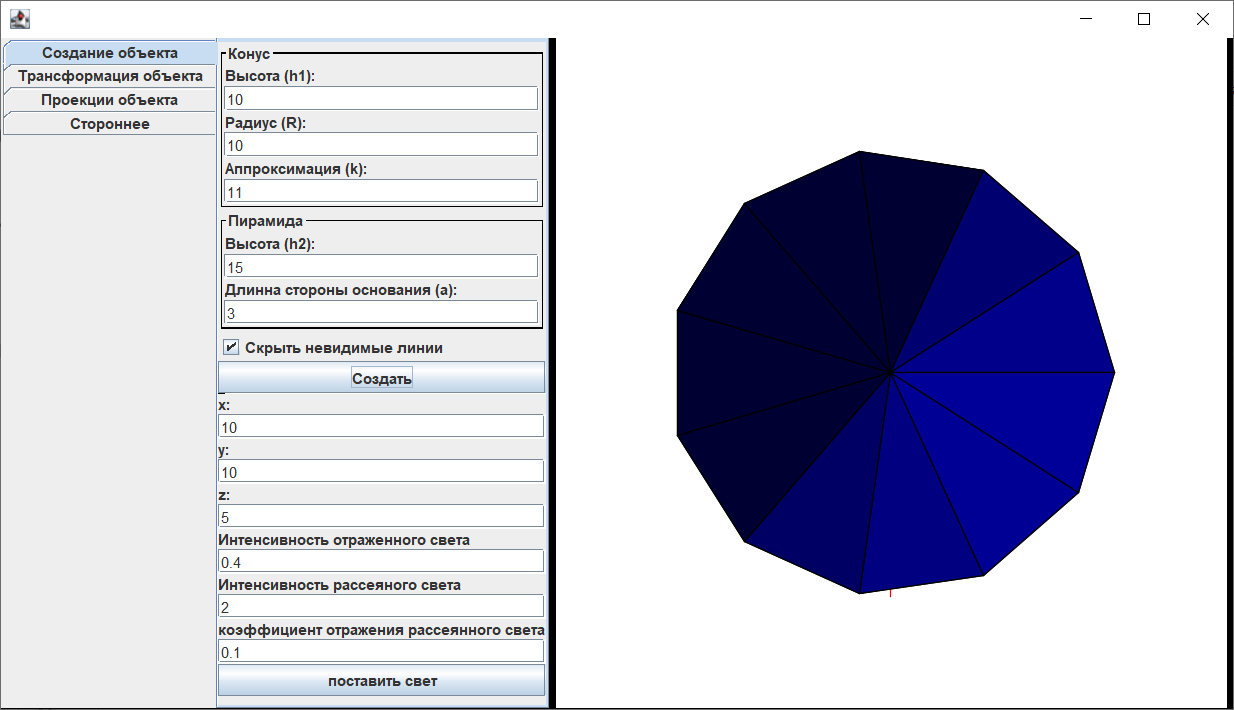


Рисунок 15 – Пример закраски объекта с учетом освещения

# ЗАКЛЮЧЕНИЕ

В результате выполнения данного курсового проекта была решена основная задача трехмерного моделирования, которая состоит в определении структуры данных, описывающей трехмерное тело, на основе этой структуры получении возможности изменять объект, изменять его положение в пространстве.

В проекте была получена трехмерная геометрическая модель конуса и пирамиды с общей поверхностью, ее отображение с учётом невидимых линий и поверхностей. В программе реализован алгоритм освещения объекта, предусмотрен режим редактирования полученного изображения и режим получения различных проекций с изменяемыми параметрами.

# СПИСОК ЛИТЕРАТУРЫ

1. Сиденко Л.А. Компьютерная графика и геометрическое моделирование. Питер – 2009, 224 с.
2. Д. Рихтер. CLR via C#. Программирование на платформе Microsoft .Net Framework 4.5 на языке C#. Питер – 2017, 896 с.
3. Носкова Л.А. Конспект лекций по дисциплине «3D-моделирование инженерны х конструкций» - Мн.: БНТУ, 2018г.