МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

**«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ им. В. Г. ШУХОВА» (БГТУ им. В.Г. Шухова)**

Кафедра программного обеспечения вычислительной техники и автоматизированных систем

Лабораторная работа №4

по дисциплине: Компьютерная графика

тема: «Аффинные преобразования в пространстве»

Выполнил: ст. группы ПВ-223

Дмитриев Андрей Александрович

Проверил:

Осипов Олег Васильевич

Белгород 2024 г.

**Цель работы:** получение навыков использования аффинных преобразований в пространстве и создание графического приложения с использованием GDI в среде Visual Studio для визуализации простейших трѐхмерных объектов.

**Порядок выполнения работы**

1. Разработать алгоритм и составить программу для построения на

экране трѐхмерных изображений в соответствии с номером варианта.

В качестве исходных данных взять указанные в таблице №1.

**Требования к программе**

1. Программа должна быть написана на языке C++.

2. Окно поделить на 4 части одинаковые части:

2.1. В верхней левой части должна отображаться фронтальная

проекция (вид спереди);

2.2. Правая верхняя часть – профильная проекция (вид сбоку);

2.3. Левая нижняя часть должна отображать вид сверху

(горизонтальную проекцию);

2.4. На правой нижней части должна отображаться проекция, вид которой выбирает пользователь: центральная, ортографическая,

аксонометрическая: изометрическая, триметрическая.

3. Предусмотреть возможность выбора вида проекции пользователем,

например, с помощью выпадающего списка или клавишами.

4. Первые три проекции отобразить без перспективных искажений. Для

четвѐртой (справа-внизу) предусмотреть возможность отдаления/приближения и поворота фигуры клавишами или с помощью мыши.

5. Предусмотреть три режима отображения фигур: только каркас

(только рѐбра объектов), только грани (заполненные цветом), и каркас

и грани. Каркас фигуры нарисовать отрезками с использованием

отсечения по методу Сазерленда-Кохена [1, стр. 89].

6. Трѐхмерные объекты хранить в памяти как массив многоугольников

(не массив отрезков). Все грани фигуры в программе должны быть

сгенерированы процедурно.

**Содержание отчѐта**

1. Название темы.

2. Цель работы.

3. Постановка задачи.

4. Вывод необходимых формул для построения граней фигуры.

5. Вывод необходимых формул для построения всех проекций. Указать

какие матрицы используются для построения всех четырѐх проекций

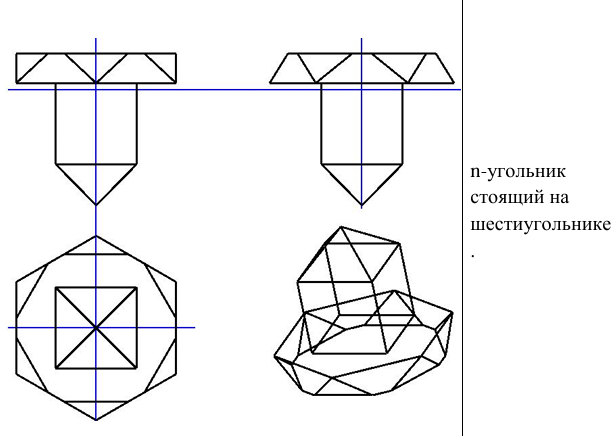
изображений и в какой последовательности они умножаются.

6. Текст программы.

7. Результат работы программы (снимки экрана).

8. Вывод о проделанной работе

Вариант 2:



Ход работы:

Пространство, в котором будет построена фигура – куб 1на1на1.

Начнём построение со «шляпки». Поворотом по оси Z отметим точки 6-угольника сверху, затем по среднему между полученными точками определим нижнюю часть 6-угольника.

Matrix rot\_z\_matrix = Matrix::RotationZ(60 \* M\_PI / 180);  
u\_6.emplace\_back(0.7, 0, 0.7);  
for (int i = 1; i < 6; i++) {  
 Vector temp = u\_6[i - 1] \* rot\_z\_matrix;  
 u\_6.emplace\_back(temp.x, temp.y, temp.z);  
}  
  
for (int i = 0; i < 6; i++) {  
 l\_6.emplace\_back((u\_6[i].x + u\_6[(i + 1) % 6].x) / 2,  
 (u\_6[i].y + u\_6[(i + 1) % 6].y) / 2,  
 (u\_6[i].z + u\_6[(i + 1) % 6].z) / 2 - 0.2);  
}

Парами точек определим стенки n-угольника.

rot\_z\_matrix = Matrix::RotationZ(360 / N \* M\_PI / 180);  
u\_N.emplace\_back(0.4, 0, 0.5);  
l\_N.emplace\_back(0.4, 0, -0.5);  
for (int i = 1; i < N; i++) {  
 Vector temp = u\_N[i - 1] \* rot\_z\_matrix;  
 u\_N.emplace\_back(temp.x, temp.y, temp.z);  
 l\_N.emplace\_back(temp.x, temp.y, temp.z - 1);  
}

И последним шагом определим центры.

cu\_6 = Vector(0, 0, 0.7);  
cl\_6 = Vector(0, 0, 0.5);  
c\_N = Vector(0, 0, -0.9);

Метод отрисовщика выглядит следующим образом:

void Draw(Frame &frame) {  
 int n = n\_angles;  
  
 Matrix projection\_matrix = Matrix::Ortho(-2.0 \* frame.width / frame.height, 2.0 \* frame.width / frame.height,  
 -2.0,  
 2.0, 1.0f, 100.0f);  
  
 Matrix front = Matrix::RotationX(-90 \* M\_PI / 180) \*  
 Matrix::Translation(0, 0, -4) \*  
 projection\_matrix \*  
 Matrix::Viewport(0, frame.height / 2, frame.width / 2, frame.height / 2);  
 Shape front\_shape(n, &frame);  
 front\_shape.draw\_polygon(front);  
  
 Matrix side = Matrix::RotationY(-90 \* M\_PI / 180) \*  
 Matrix::Translation(0, 0, -4) \*  
 projection\_matrix \*  
 Matrix::Viewport(frame.width / 2, frame.height / 2, frame.width / 2, frame.height / 2);  
 Shape side\_shape(n, &frame);  
 side\_shape.draw\_polygon(side);  
  
 Matrix top = Matrix::Translation(0, 0, -4) \*  
 projection\_matrix \*  
 Matrix::Viewport(0, 0, frame.width / 2, frame.height / 2);  
 Shape top\_shape(n, &frame);  
 top\_shape.draw\_polygon(top);  
  
 Matrix transform = Matrix::RotationZ(z\_rot \* M\_PI / 180) \*  
 Matrix::RotationX(((-90 + x\_rot) \* M\_PI / 180)) \*  
 Matrix::RotationY(y\_rot \* M\_PI / 180) \*  
 Matrix::Translation(0, 0, -4) \*  
 Matrix::Translation(x\_trsl, y\_trsl, z\_trsl);  
  
 // Выбор матрицы проектирования  
 Matrix main\_matrix;  
 switch (projection\_type % 2) {  
 case 0:  
 projection\_matrix = Matrix::Frustum(-0.5 \* frame.width / frame.height, 0.5 \* frame.width / frame.height,  
 -0.5, 0.5, 1.0, 20);  
 main\_matrix =  
 transform \*  
 projection\_matrix \*  
 Matrix::Viewport(frame.width / 2, 0, frame.width / 2, frame.height / 2);  
 break;  
 case 1:  
 projection\_matrix = Matrix::Ortho(-2.0 \* frame.width / frame.height, 2.0 \* frame.width / frame.height,  
 -2.0,  
 2.0, 1.0f, 100.0f);  
 main\_matrix =  
 transform \*  
 projection\_matrix \*  
 Matrix::Viewport(frame.width / 2, 0, frame.width / 2, frame.height / 2);  
 break;  
 }  
  
 Shape main\_shape(n, &frame);  
 switch (draw\_type % 3) {  
 case 0:  
 main\_shape.draw\_polygon(main\_matrix);  
 break;  
 case 1:  
 main\_shape.draw\_edges(main\_matrix);  
 break;  
 case 2:  
 main\_shape.draw\_full(main\_matrix);  
 break;  
 }  
  
}

**Вывод:** в ходе работы получены навыки использования аффинных преобразований в пространстве и создания графического приложения с использованием GDI в среде Visual Studio для визуализации простейших трѐхмерных объектов.