ГУАП

КАФЕДРА № 14

ОТЧЕТ   
ЗАЩИЩЕН С ОЦЕНКОЙ

ПРЕПОДАВАТЕЛЬ

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| доцент, канд. техн. наук |  |  |  | А.В. Шахомиров |
| должность, уч. степень, звание |  | подпись, дата |  | инициалы, фамилия |

|  |
| --- |
| ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ |
| 3D-ФИГУРЫ, ПОСТРОЕНИЕ ПРОЕКЦИЙ |
| по курсу: КОМПЬЮТЕРНАЯ ГРАФИКА |
|  |
|  |

РАБОТУ ВЫПОЛНИЛ

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| СТУДЕНТ ГР. № | 1041 |  |  |  | Ю.В. Ахромова |
|  |  |  | подпись, дата |  | инициалы, фамилия |

Санкт-Петербург 2022

1. **Цель работы**

Изучение методов работы с библиотеками для графики, рисования и изменения проекций трёхмерных фигур.

1. **Постановка задачи**

С помощью библиотеки для графики нарисовать трёхгранную пирамиду, подписать её точки, а также реализовать методы работы с ней:

- перемещение;

- вращение;

- масштабирование.

1. **Формализация задачи**

При запуске программы создаётся окно размерами 1400 на 700 функцией initwindow. Для создания и работы с точками используется класс Point, хранящий координаты x, y и z точки её имя и метод для отображения её имени. Линии между точками отрисовываются при помощи функции line\_DDA алгоритмом рисования DDA-линии. В классе Piramid содержатся данные для пирамиды: четыре её точки, их имена, цвет линий. Конструктор класса задаёт начальные координаты точек, их имена и отрисовывает треугольник на экране при помощи метода класса drawPiramid. Для задания цветов используются следующие обозначения: TEXTCOL – цвет текста, MAINCOL – цвет границ треугольника и WHITE – белый цвет.

Перемещение осуществляется тремя методами: moveX, moveY и moveZ, сдвигающим треугольник вертикально, горизонтально, на нас и от нас соответственно. Методы rotateZ, rotateY и rotateX поворачивают фигуру по или против часовой стрелки вокруг осей z, y и x соответственно. Метод scale масштабирует фигуру. По завершении каждой из функций экран отчищается и выводится новая отредактированная пирамида. Управление производится при помощи следующих клавиш:

W – перемещение вверх;

A – перемещение влево;

S – перемещение вниз;

D – перемещение вправо;

Z – перемещение вперёд;

X – перемещение назад;

Q – поворот против часовой стрелки вокруг оси z;

E – поворот по часовой стрелке вокруг оси z;

R – поворот против часовой стрелки вокруг оси y;

T – поворот по часовой стрелке вокруг оси y;

F – поворот против часовой стрелки вокруг оси x;

G – поворот по часовой стрелке вокруг оси x;

- – уменьшение в масштабе;

+ – увеличение в масштабе.

Нажатие любой другой клавиши приводит к завершению программы.

По завершении программы функция closegraph освобождает всю память, выделенную под графическую систему, затем восстанавливает экран в режим, который был до вызова initwindow.

Использованное ПО:

Microsoft Visual Studio Enterprise 2019.

Версия компилятора: 16.11.8

Библиотека graphics.h (https://github.com/ahuynh359/Graphics).

1. **Тестовый пример**

Точка A (20; 250; 0).

Точка B (400; 50; -20).

Точка C (100; 50; 50).

Точка D (10; 5; 10).

1. **Листинг программы**

#include <iostream>

#include <math.h>

#include "graphics.h"

#pragma comment(lib,"graphics.lib")

#define P 3.14

#define TEXTCOL 13

#define MAINCOL 11

#define WHITE 15

using namespace std;

// отрисовка линии попиксельно

void line\_DDA(float x1, float y1, float x2, float y2, COLORREF cColor) {

// Целочисленные значения координат начала и конца отрезка, округлённые до ближайшего целого

int iX1 = roundf(x1);

int iY1 = roundf(y1);

int iX2 = roundf(x2);

int iY2 = roundf(y2);

// Длина и высота линии

int deltaX = abs(iX1 - iX2);

int deltaY = abs(iY1 - iY2);

// Считаем минимальное количество итераций, необходимое для отрисовки отрезка

// Выбирая максимум из длины и высоты линии, обеспечиваем связность линии

int length = max(deltaX, deltaY);

// особый случай, на экране закрашивается ровно один пиксель

if (length == 0) {

putpixel(iX1, iY1, cColor);

return;

}

// Вычисляем приращения на каждом шаге по осям абсцисс и ординат

double dX = (x2 - x1) / length;

double dY = (y2 - y1) / length;

// Начальные значения

double x = x1;

double y = y1;

// Основной цикл

length++;

while (length--) {

x += dX;

y += dY;

putpixel(roundf(x), roundf(y), cColor);

}

}

// класс для точек

class Point {

public:

float x;

float y;

float z; // z = (0.5x; 0.5y) для угла в 45. Проблемы с округлением? не делать короче 2х пикселей тогда??

char\* name;

void namePoint(char\* name) {

outtextxy(x, y, name);

}

};

// класс трёхгранной пирамиды

class Piramid {

public:

Point A;

Point B;

Point C;

Point D;

char name\_A[2] = "A";

char name\_B[2] = "B";

char name\_C[2] = "C";

char name\_D[2] = "D";

int col = MAINCOL;

// конструктор

Piramid() {

A.x = 20; A.y = 250; A.z = 0;

A.name = name\_A;

B.x = 400; B.y = 50; B.z = -20;

B.name = name\_B;

C.x = 100; C.y = 50; C.z = 50;

C.name = name\_C;

D.x = 10; D.y = 5; D.z = 10;

D.name = name\_D;

drawPiramid();

}

// отрисовка

void drawPiramid() {

// оси

setcolor(WHITE);

line(700, 350, 730, 350);

line(700, 350, 700, 320);

line(700, 350, 685, 365);

char w[20] = "W - Up";

char a[20] = "A - Left";

char s[20] = "S - Down";

char d[20] = "D - Right";

char z[20] = "Z - Forward";

char x[20] = "X - Backward";

char qrf[40] = "Q, R, F - Rotations counter clockwise";

char etg[30] = "E, T, G - Rotations clockwise";

char plus[20] = "+ - Bigger scale";

char minus[20] = "- - Smaller scale";

char other[20] = "Any other - Exit";

outtextxy(1140, 20, w);

outtextxy(1140, 40, a);

outtextxy(1140, 60, s);

outtextxy(1140, 80, d);

outtextxy(1140, 100, z);

outtextxy(1140, 120, x);

outtextxy(1140, 140, qrf);

outtextxy(1140, 160, etg);

outtextxy(1140, 180, plus);

outtextxy(1140, 200, minus);

outtextxy(1140, 220, other);

setcolor(TEXTCOL);

A.namePoint(A.name);

B.namePoint(B.name);

C.namePoint(C.name);

D.namePoint(D.name);

line\_DDA(A.x, A.y, B.x, B.y, col); // линия 1

line\_DDA(B.x, B.y, C.x, C.y, col+1); // линия 2

line\_DDA(A.x, A.y, C.x, C.y, col+2); // линия 3

line\_DDA(A.x, A.y, D.x, D.y, col+3); // линия 4

line\_DDA(B.x, B.y, D.x, D.y, col+4); // линия 5

line\_DDA(C.x, C.y, D.x, D.y, col-1); // линия 6

}

// пермещение

void moveX(float amt) {

A.x += amt;

B.x += amt;

C.x += amt;

D.x += amt;

}

void moveY(float amt) {

A.y += amt;

B.y += amt;

C.y += amt;

D.y += amt;

}

void moveZ(float amt) {

A.x += 0.5 \* amt;

B.x += 0.5 \* amt;

C.x += 0.5 \* amt;

D.x += 0.5 \* amt;

A.y -= 0.5 \* amt;

B.y -= 0.5 \* amt;

C.y -= 0.5 \* amt;

D.y -= 0.5 \* amt;

}

// поворот вокруг z

void rotateZ(int u) { // u = -1 по часовой, u = 1 против

Point tmpA = A, tmpB = B, tmpC = C, tmpD = D;

float ang = u \* 0.05; // угол поворота

Point Cen; // точка центра треугольника

Cen.x = (A.x + B.x + C.x + D.x) / 4;

Cen.y = (A.y + B.y + C.y + D.y) / 4;

A.x = A.x - Cen.x; // расстояние от а до центра по х

A.y = A.y - Cen.y; // по у

B.x = B.x - Cen.x; // расстояние от b до центра по х

B.y = B.y - Cen.y; // по у

C.x = C.x - Cen.x; // расстояние от c до центра по х

C.y = C.y - Cen.y; // по у

D.x = D.x - Cen.x; // расстояние от d до центра по х

D.y = D.y - Cen.y; // по у

// поворот вокруг нк

// A

float tmpX = A.x \* cos(ang) + A.y \* sin(ang);

float tmpY = -A.x \* sin(ang) + A.y \* cos(ang);

A.x = tmpX + Cen.x;

A.y = tmpY + Cen.y;

// B

tmpX = B.x \* cos(ang) + B.y \* sin(ang);

tmpY = -B.x \* sin(ang) + B.y \* cos(ang);

B.x = tmpX + Cen.x;

B.y = tmpY + Cen.y;

// C

tmpX = C.x \* cos(ang) + C.y \* sin(ang);

tmpY = -C.x \* sin(ang) + C.y \* cos(ang);

C.x = tmpX + Cen.x;

C.y = tmpY + Cen.y;

// D

tmpX = D.x \* cos(ang) + D.y \* sin(ang);

tmpY = -D.x \* sin(ang) + D.y \* cos(ang);

D.x = tmpX + Cen.x;

D.y = tmpY + Cen.y;

}

// поворот вокруг y

void rotateY(int u) { // u = -1 по часовой, u = 1 против

Point tmpA = A, tmpB = B, tmpC = C, tmpD = D;

float ang = u \* 0.05; // угол поворота

Point Cen; // точка центра треугольника

Cen.x = (A.x + B.x + C.x + D.x) / 4;

Cen.z = (A.z + B.z + C.z + D.z) / 4;

A.x = A.x - Cen.x; // расстояние от а до центра по х

A.z = A.z - Cen.z; // по z

B.x = B.x - Cen.x; // расстояние от b до центра по х

B.z = B.z - Cen.z; // по z

C.x = C.x - Cen.x; // расстояние от c до центра по х

C.z = C.z - Cen.z; // по z

D.x = D.x - Cen.x; // расстояние от d до центра по х

D.z = D.z - Cen.z; // по z

// поворот вокруг нк

// A

float tmpX = A.x \* cos(ang) + A.z \* sin(ang);

float tmpZ = -A.x \* sin(ang) + A.z \* cos(ang);

A.x = tmpX + Cen.x;

A.z = tmpZ + Cen.z;

// B

tmpX = B.x \* cos(ang) + B.z \* sin(ang);

tmpZ = -B.x \* sin(ang) + B.z \* cos(ang);

B.x = tmpX + Cen.x;

B.z = tmpZ + Cen.z;

// C

tmpX = C.x \* cos(ang) + C.z \* sin(ang);

tmpZ = -C.x \* sin(ang) + C.z \* cos(ang);

C.x = tmpX + Cen.x;

C.z = tmpZ + Cen.z;

// D

tmpX = D.x \* cos(ang) + D.z \* sin(ang);

tmpZ = -D.x \* sin(ang) + D.z \* cos(ang);

D.x = tmpX + Cen.x;

D.z = tmpZ + Cen.z;

}

// поворот вокруг x

void rotateX(int u) { // u = -1 по часовой, u = 1 против

Point tmpA = A, tmpB = B, tmpC = C, tmpD = D;

float ang = u \* 0.05; // угол поворота

Point Cen; // точка центра треугольника

Cen.y = (A.y + B.y + C.y + D.y) / 4;

Cen.z = (A.z + B.z + C.z + D.z) / 4;

A.y = A.y - Cen.y; // расстояние от а до центра по y

A.z = A.z - Cen.z; // по z

B.y = B.y - Cen.y; // расстояние от b до центра по y

B.z = B.z - Cen.z; // по z

C.y = C.y - Cen.y; // расстояние от c до центра по y

C.z = C.z - Cen.z; // по z

D.y = D.y - Cen.y; // расстояние от d до центра по y

D.z = D.z - Cen.z; // по z

// поворот вокруг нк

// A

float tmpY = A.y \* cos(ang) + A.z \* sin(ang);

float tmpZ = -A.y \* sin(ang) + A.z \* cos(ang);

A.y = tmpY + Cen.y;

A.z = tmpZ + Cen.z;

// B

tmpY = B.y \* cos(ang) + B.z \* sin(ang);

tmpZ = -B.y \* sin(ang) + B.z \* cos(ang);

B.y = tmpY + Cen.y;

B.z = tmpZ + Cen.z;

// C

tmpY = C.y \* cos(ang) + C.z \* sin(ang);

tmpZ = -C.y \* sin(ang) + C.z \* cos(ang);

C.y = tmpY + Cen.y;

C.z = tmpZ + Cen.z;

// D

tmpY = D.y \* cos(ang) + D.z \* sin(ang);

tmpZ = -D.y \* sin(ang) + D.z \* cos(ang);

D.y = tmpY + Cen.y;

D.z = tmpZ + Cen.z;

}

// масштабирование

void scale(float e) {

Point Cen; // точка центра треугольника

Cen.x = (A.x + B.x + C.x + D.x) / 4;

Cen.y = (A.y + B.y + C.y + D.y) / 4;

if ((abs(A.x - Cen.x) >= 1 && abs(A.y - Cen.y) >= 1) && (abs(B.x - Cen.x) >= 1 && abs(B.y - Cen.y) >= 1) &&

(abs(C.x - Cen.x) >= 1 && abs(C.y - Cen.y) >= 1) && (abs(D.x - Cen.x) >= 1 && abs(D.y - Cen.y) >= 1) ||

e > 1) { // предотвращение сжатия в точку

// A.x

float xe = (Cen.x + A.x) / 2;

float lx = Cen.x - A.x;

lx = lx \* e;

A.x = xe - lx / 2;

// A.y

float ye = (Cen.y + A.y) / 2;

float ly = Cen.y - A.y;

ly = ly \* e;

A.y = ye - ly / 2;

// B.x

xe = (Cen.x + B.x) / 2;

lx = Cen.x - B.x;

lx = lx \* e;

B.x = xe - lx / 2;

// B.y

ye = (Cen.y + B.y) / 2;

ly = Cen.y - B.y;

ly = ly \* e;

B.y = ye - ly / 2;

// C.x

xe = (Cen.x + C.x) / 2;

lx = Cen.x - C.x;

lx = lx \* e;

C.x = xe - lx / 2;

// C.y

ye = (Cen.y + C.y) / 2;

ly = Cen.y - C.y;

ly = ly \* e;

C.y = ye - ly / 2;

// D.x

xe = (Cen.x + D.x) / 2;

lx = Cen.x - D.x;

lx = lx \* e;

D.x = xe - lx / 2;

// D.y

ye = (Cen.y + D.y) / 2;

ly = Cen.y - D.y;

ly = ly \* e;

D.y = ye - ly / 2;

}

}

};

int main() {

initwindow(1400, 700); // создаём консольное окно 1400 на 700

Piramid Tri; // создание трёхгранной пирамиды

// управление

int i = 1; // условие выхода

while (i) {

switch (getch()) {

case 'w':

cout << 'w' << endl;

Tri.moveY(-10); // вверх

break;

case 'a':

cout << 'a' << endl;

Tri.moveX(-10); // влево

break;

case 's':

cout << 's' << endl;

Tri.moveY(10); // вниз

break;

case 'd':

cout << 'd' << endl;

Tri.moveX(10); // вправо

break;

case 'x':

cout << 'x' << endl;

Tri.moveZ(10); // назад

Tri.scale(0.9);

break;

case 'z':

cout << 'z' << endl;

Tri.moveZ(-10); // вперёд

Tri.scale(1.1);

break;

// вокруг z

case 'q':

cout << 'q' << endl;

Tri.rotateZ(1); // против часовой

break;

case 'e':

cout << 'e' << endl;

Tri.rotateZ(-1); // по часовой

break;

// вокруг y

case 'r':

cout << 'r' << endl;

Tri.rotateY(1); // против часовой

break;

case 't':

cout << 't' << endl;

Tri.rotateY(-1); // по часовой

break;

// вокруг x

case 'f':

cout << 'f' << endl;

Tri.rotateX(1); // против часовой

break;

case 'g':

cout << 'g' << endl;

Tri.rotateX(-1); // по часовой

break;

case '=':

cout << '+' << endl;

Tri.scale(1.5); // увеличение

break;

case '-':

cout << '-' << endl;

Tri.scale(0.5); // уменьшение

break;

default:

cout << "default -> exit" << endl;

i = 0;

break;

}

cleardevice(); // отичстка экрана

Tri.drawPiramid(); // перерисовка трёхгранной пирамиды

}

getch(); // чтение одного символа с клавиатуры

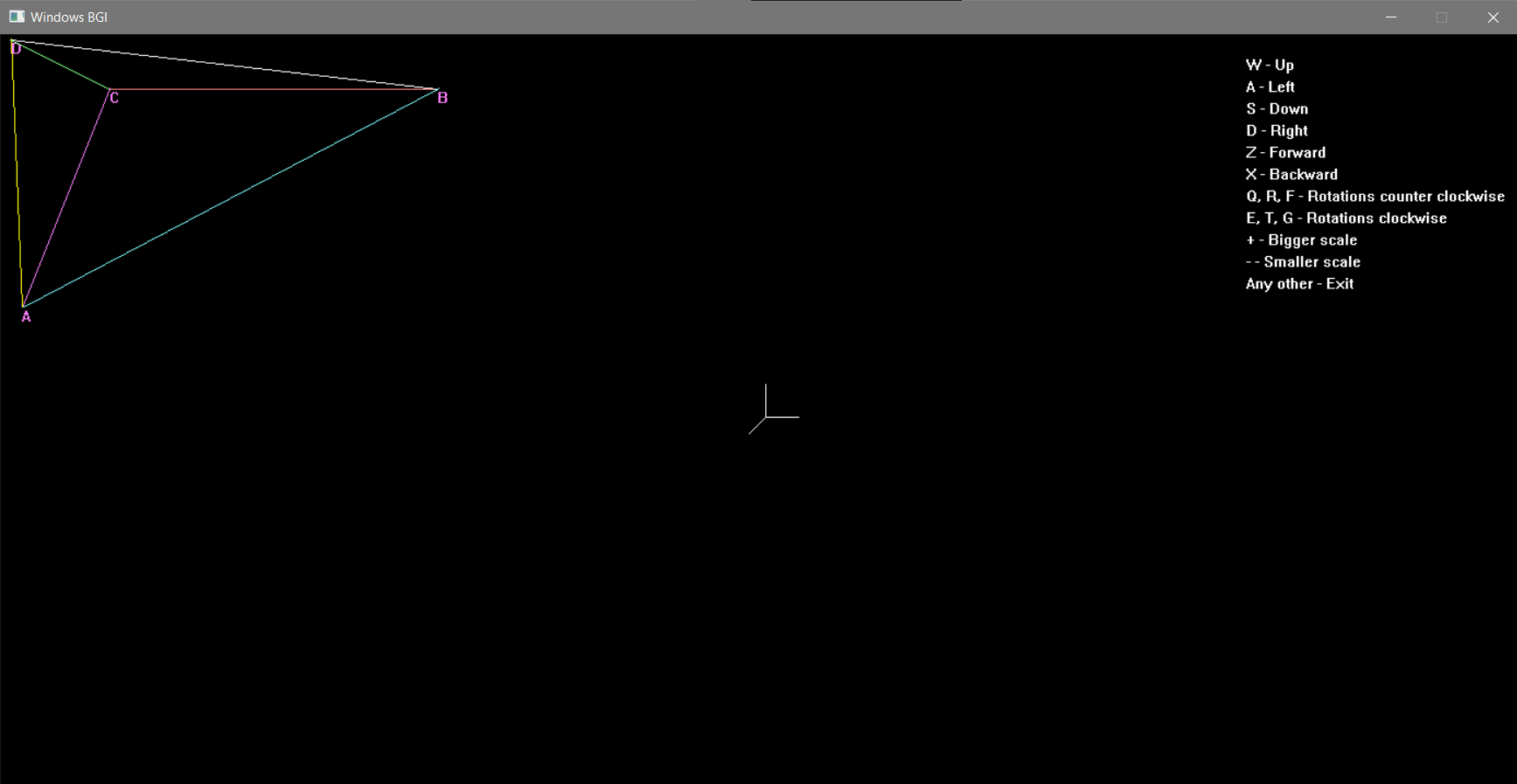
closegraph(); // освобождает всю память, выделенную под графическую систему, затем восстанавливает экран в режим, который был до вызова initwindow

return 0;

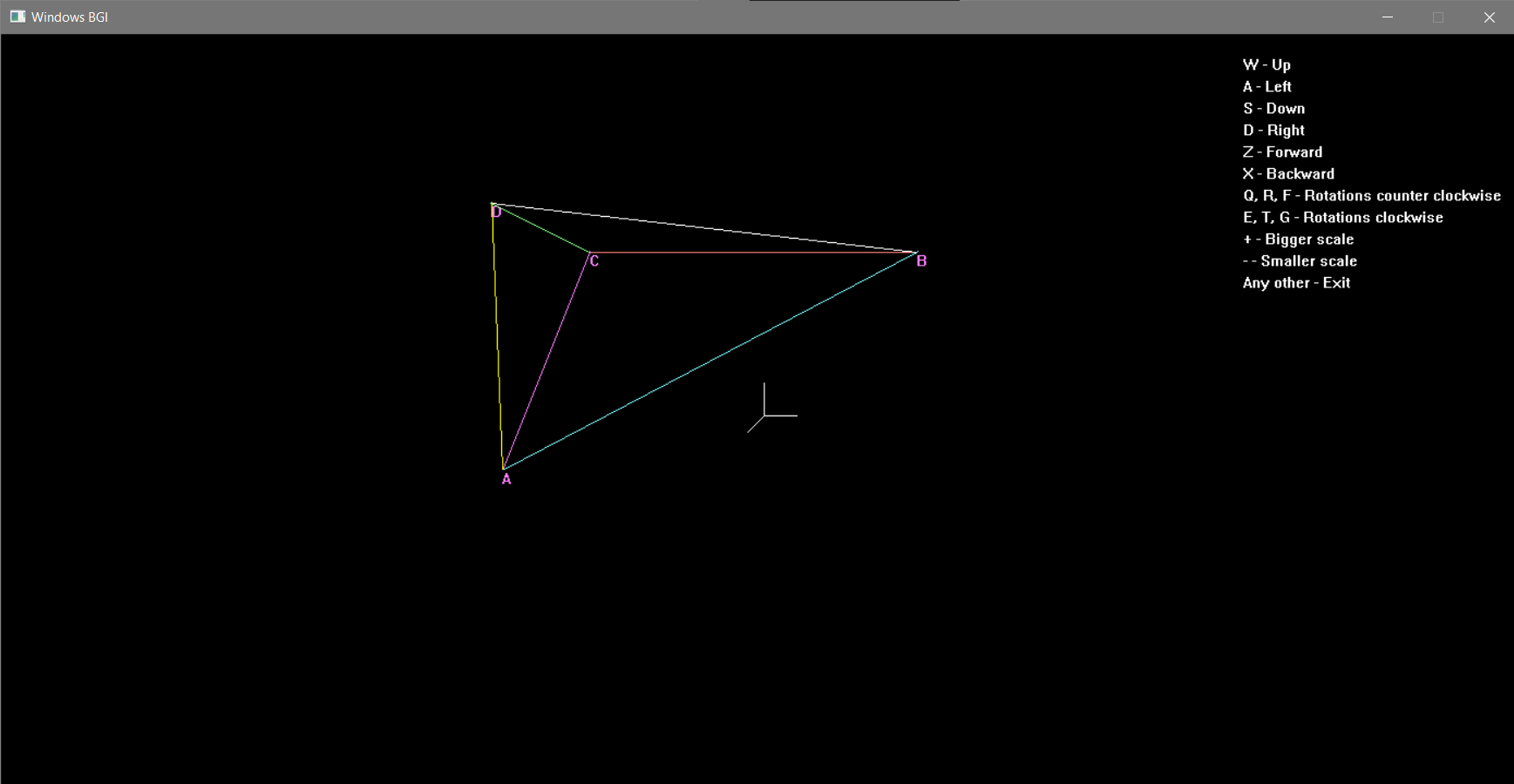
}

1. **Результаты работы программы**

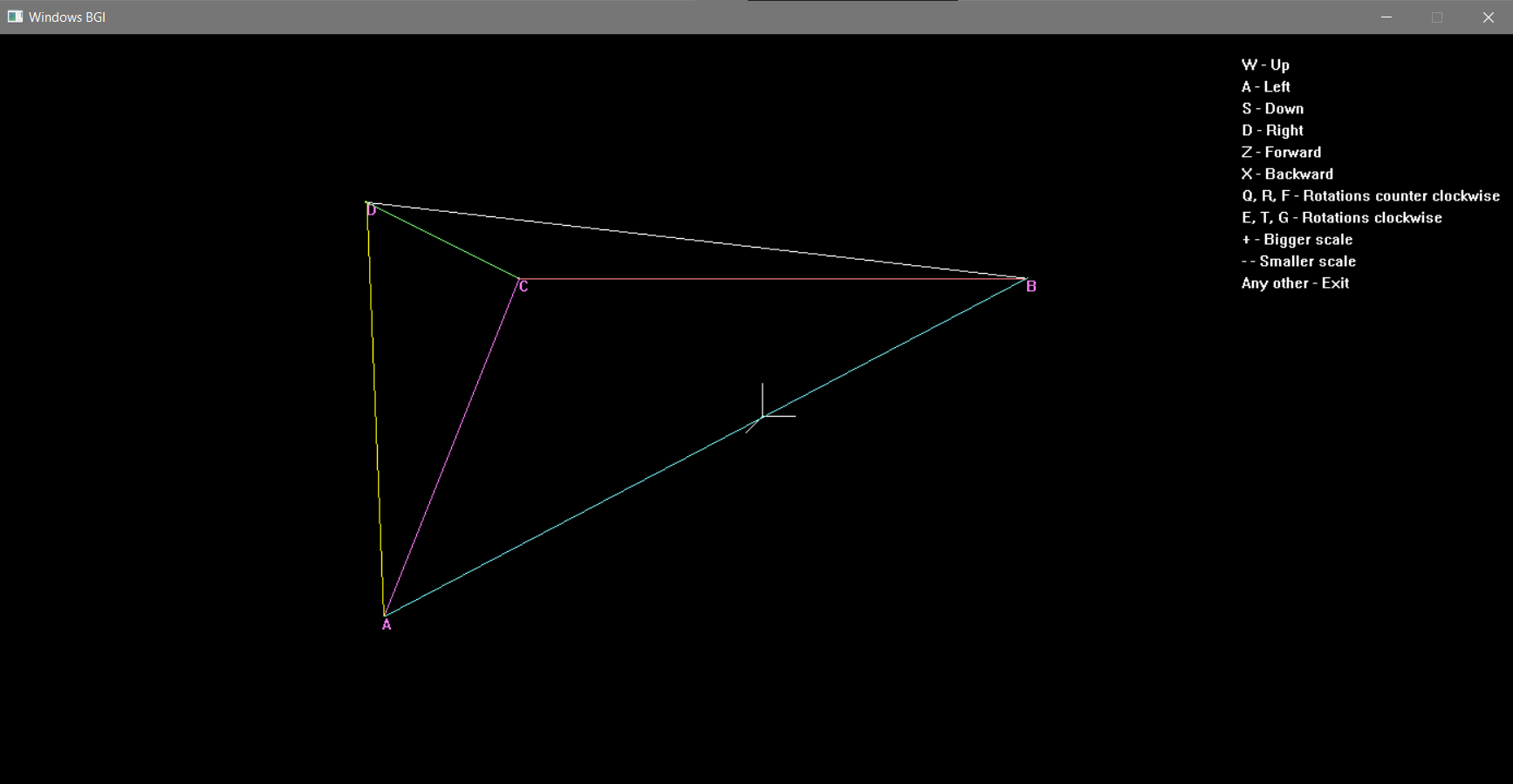
Результаты работы программы представлены на рисунках 1-11.



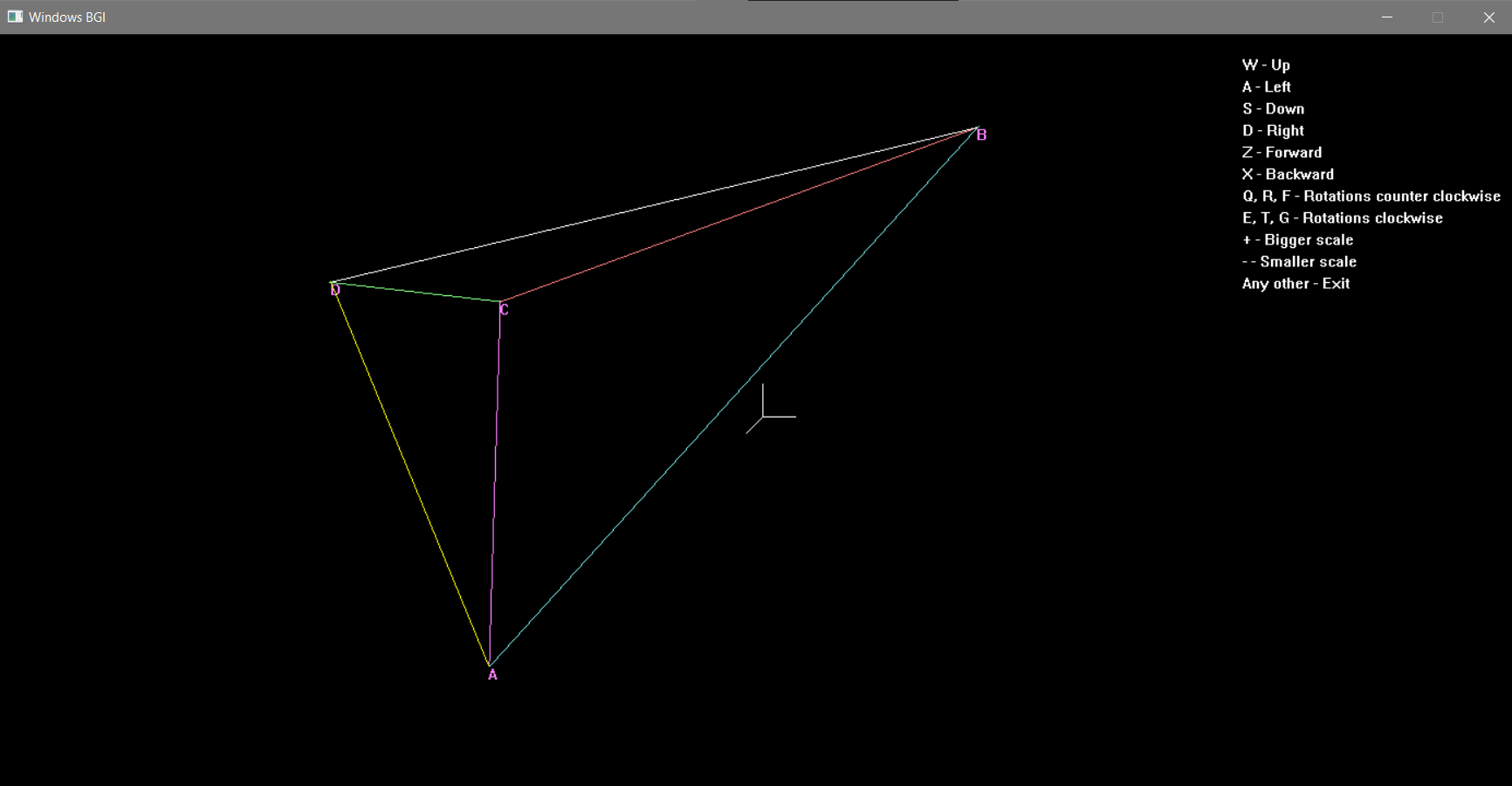
*Рисунок 1 – Отрисовка фигуры*



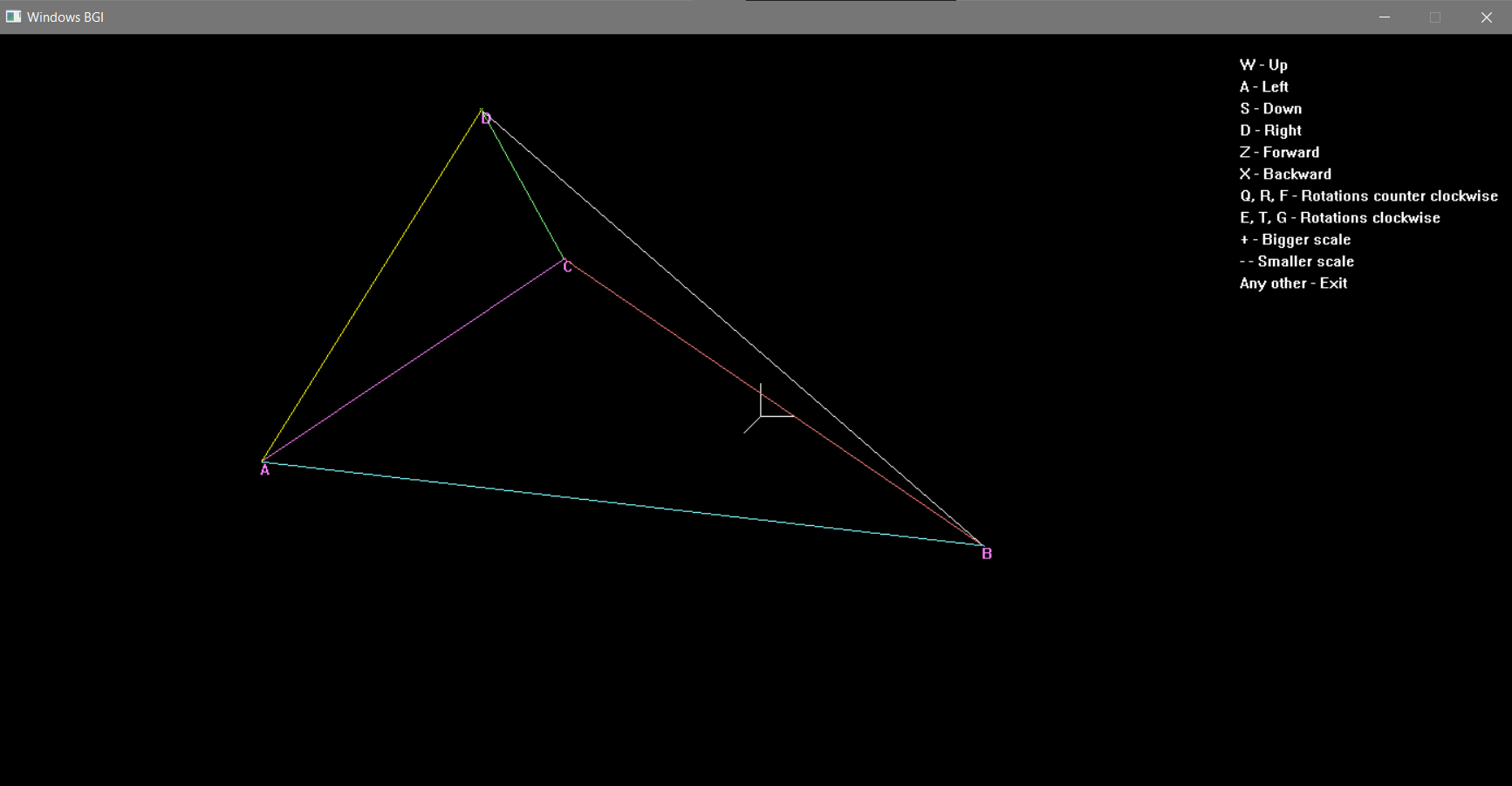
*Рисунок 2 – Перемещение фигуры вдоль осей x и y*



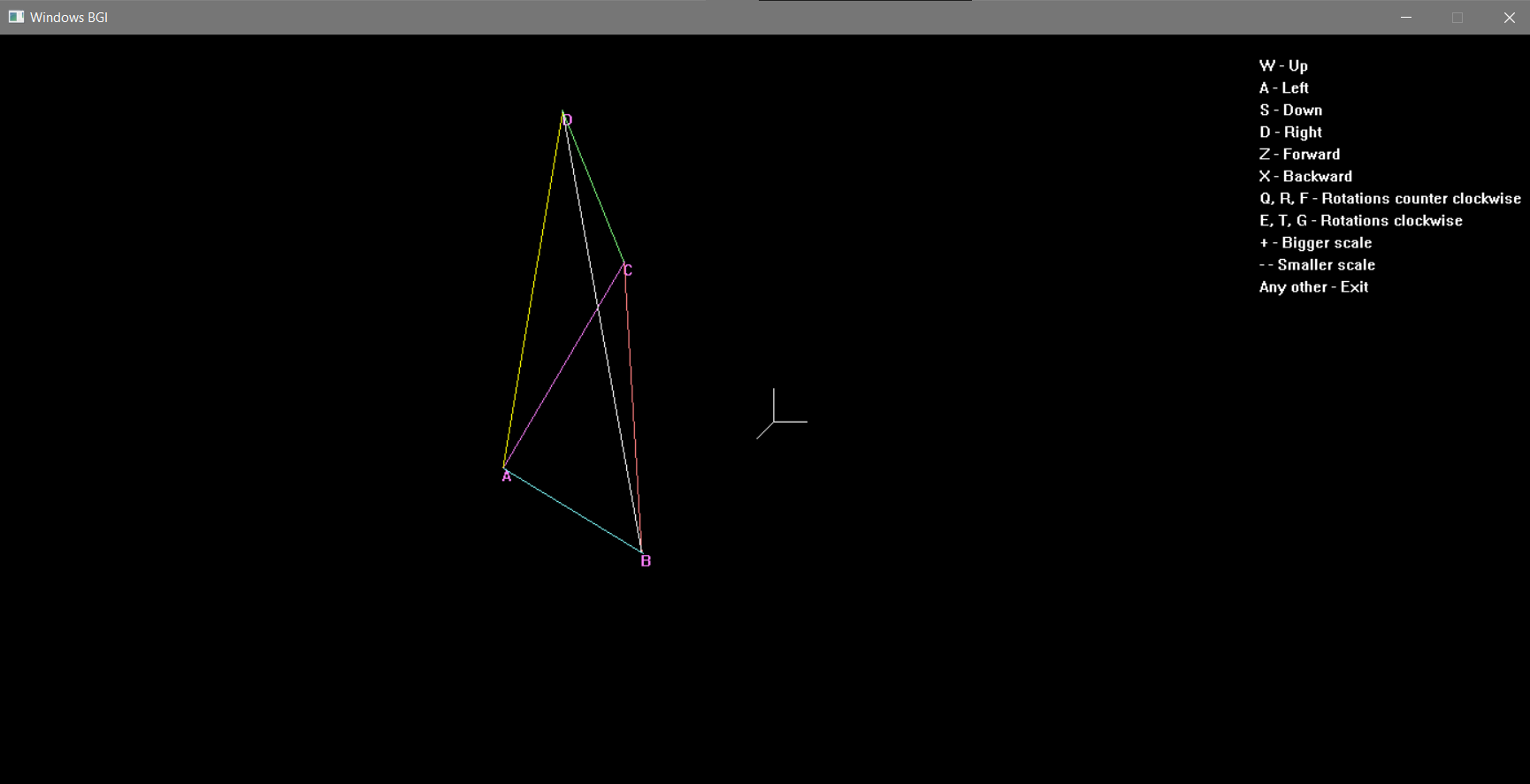
*Рисунок 3 – Перемещение фигуры вдоль оси z*

**

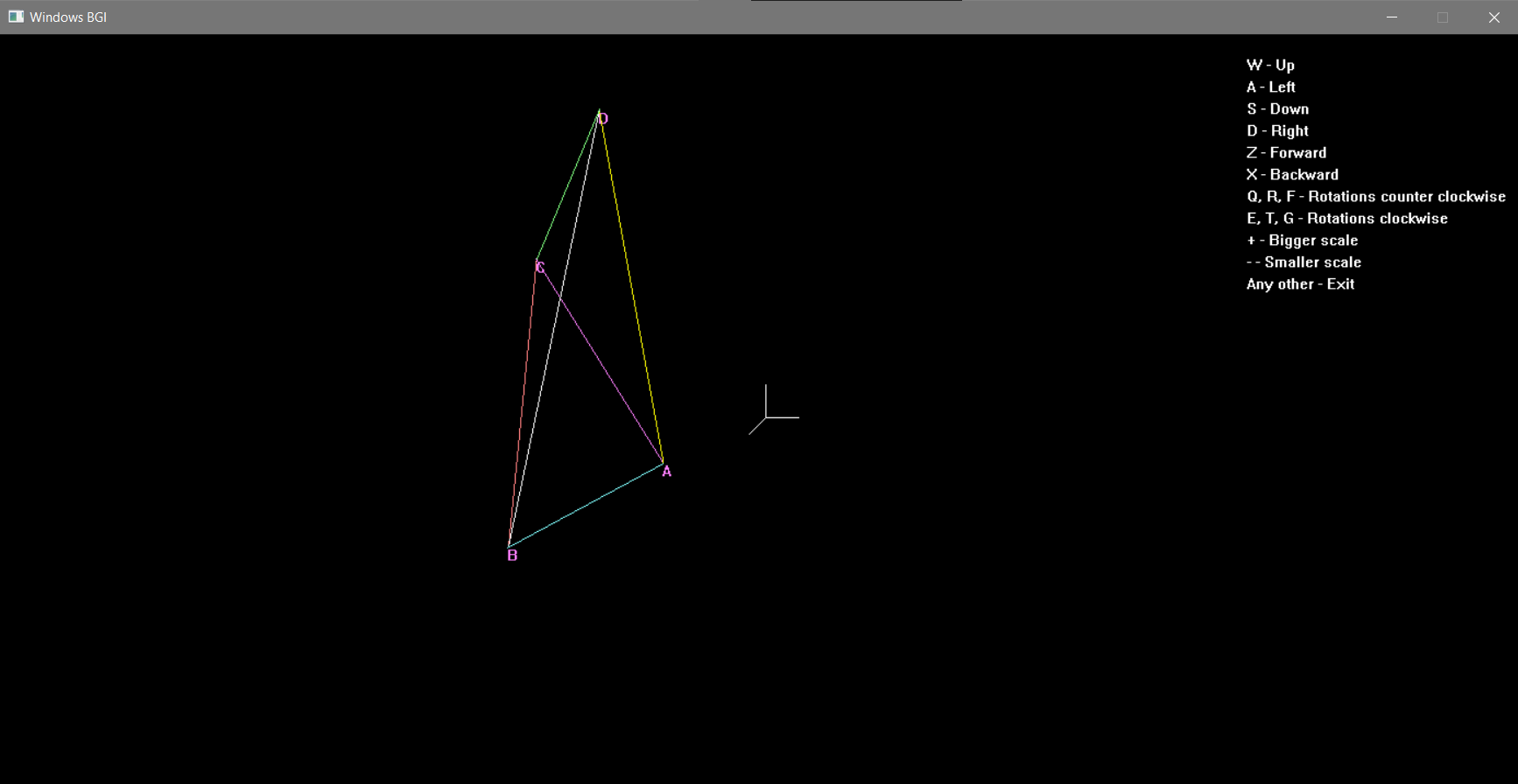
*Рисунок 4 – Поворот фигуры против часовой стрелки (ось z)*



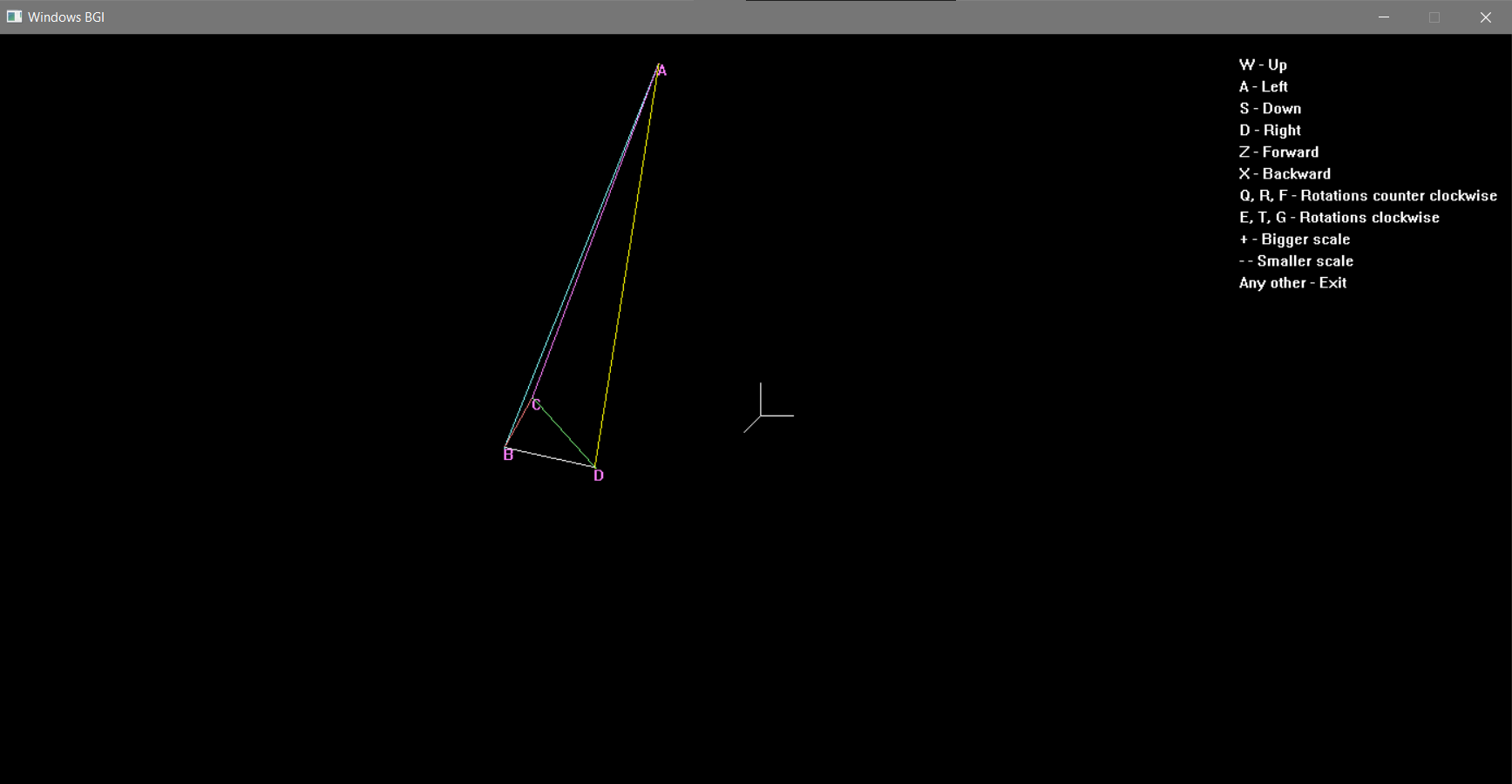
*Рисунок 5 – Поворот фигуры по часовой стрелке (ось z)*

**

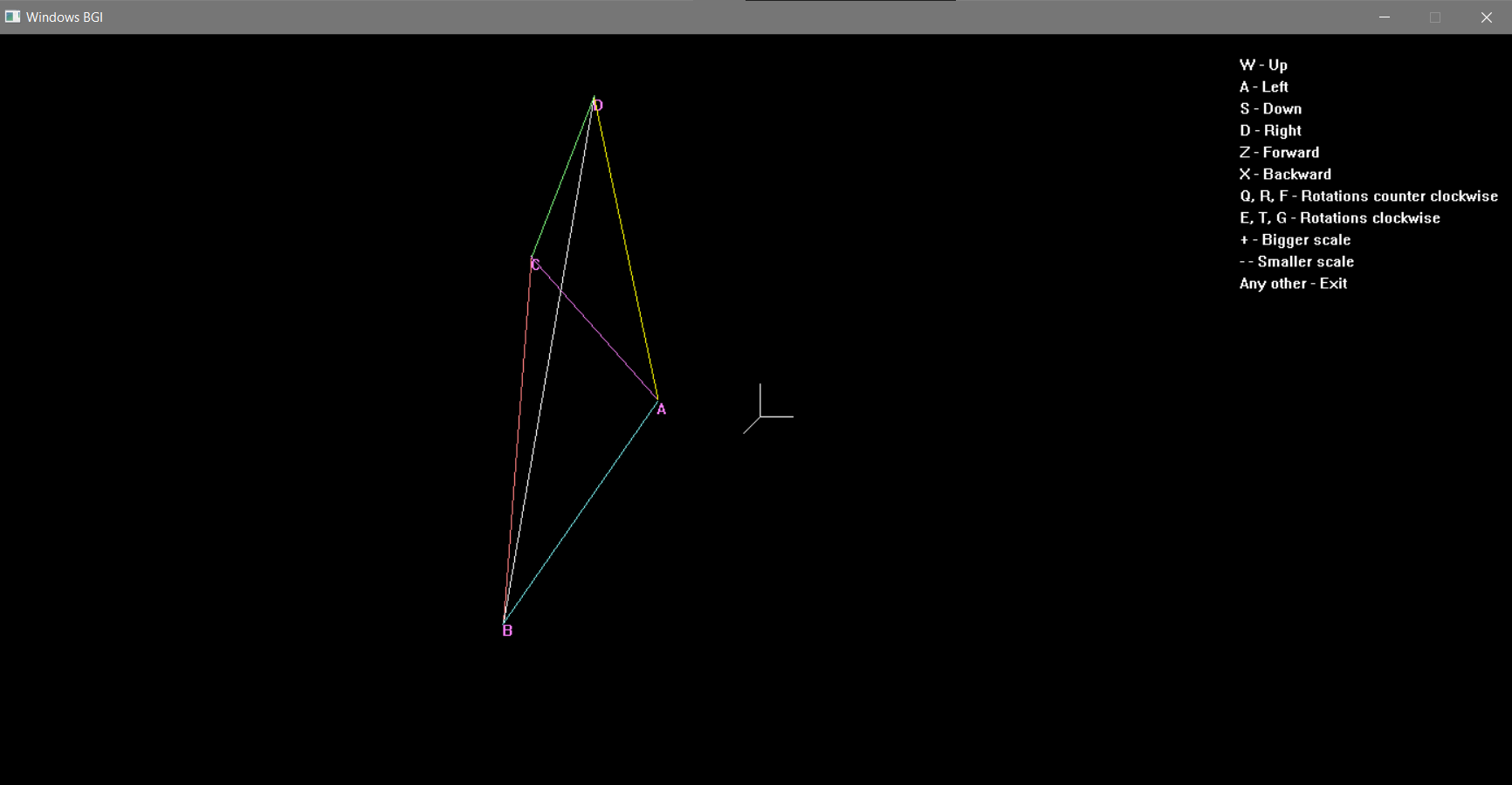
*Рисунок 6 – Поворот фигуры против часовой стрелки (ось y)*



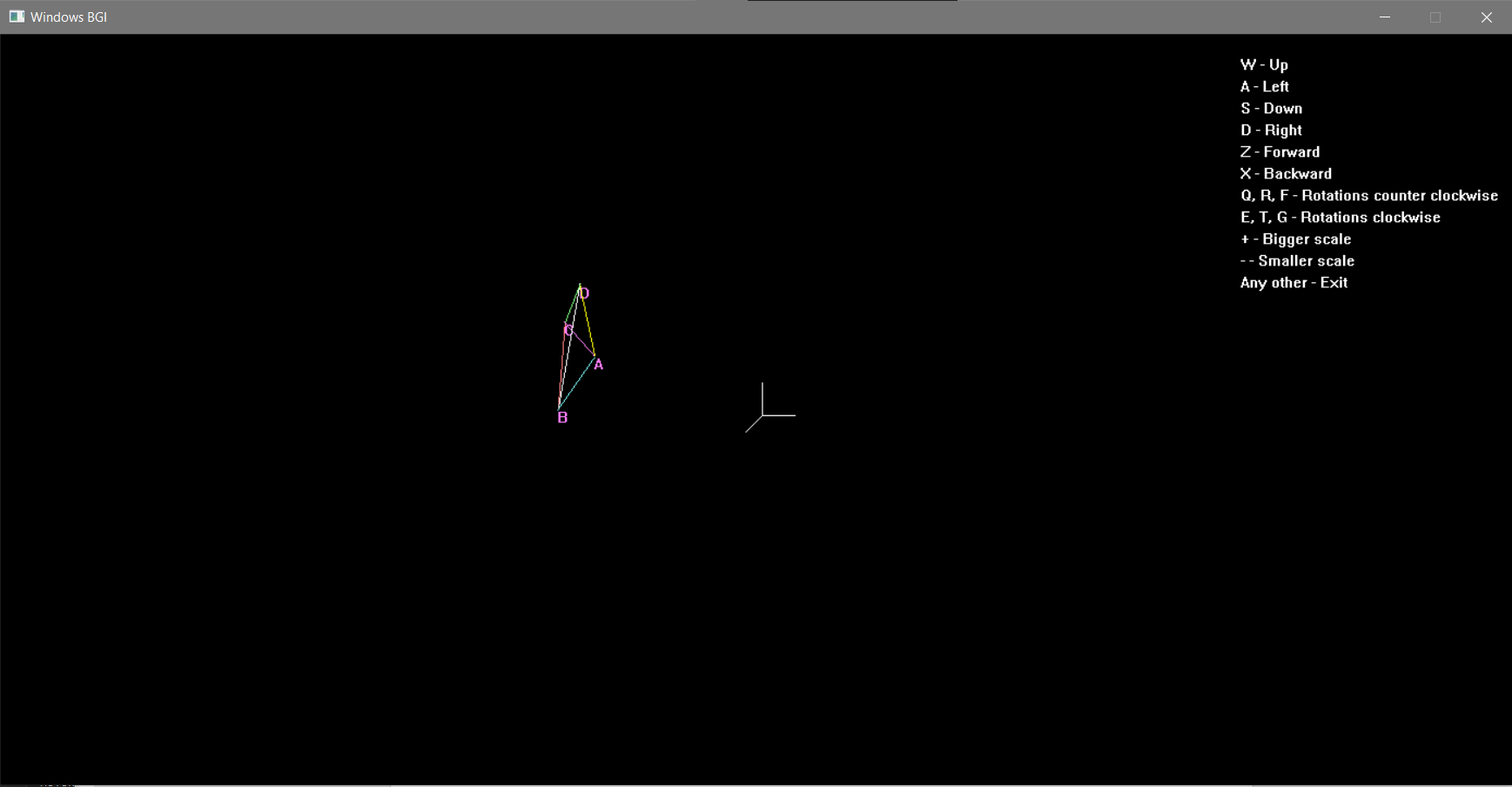
*Рисунок 7 – Поворот фигуры по часовой стрелке (ось y)*

**

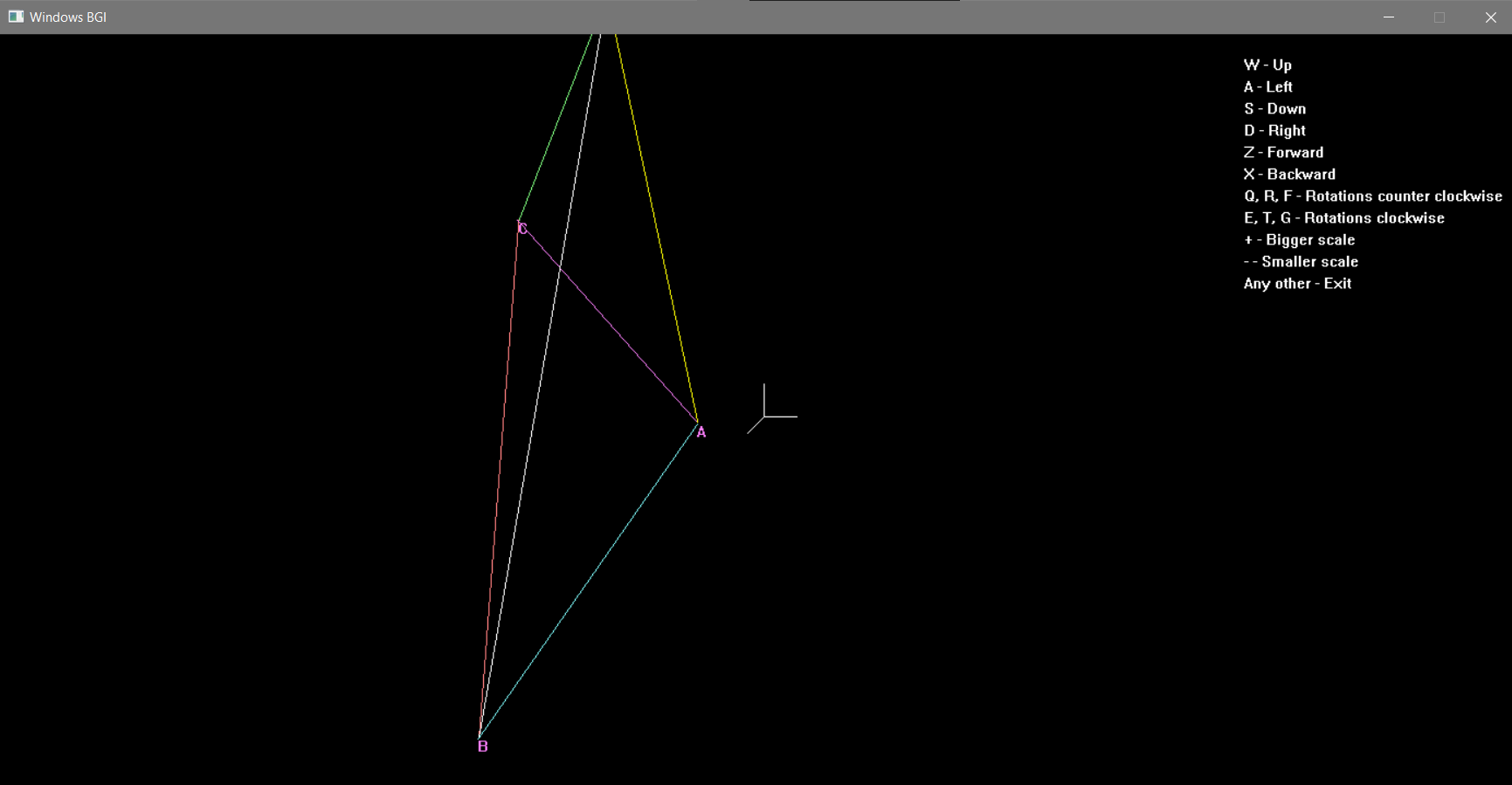
*Рисунок 8 – Поворот фигуры против часовой стрелки (ось x)*



*Рисунок 9 – Поворот фигуры по часовой стрелке (ось x)*



*Рисунок 10 – Уменьшение фигуры*



*Рисунок 11 – Увеличение фигуры*

1. **Выводы**

В ходе выполнения лабораторной работы были получены навыки работы с библиотекой для графики graphics.h, рисования, закрашивания и изменения проекции трёхмерной фигуры. Результат программы корректен.