Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное
учреждение высшего образования
«Южно-Уральский государственный университет
(национальный исследовательский университет)»
Институт естественных и точных наук
Кафедра прикладной математики и программирования

ОТЧЕТ

о выполнении лабораторной работы № 6 по дисциплине
«Математические основы компьютерной графики»

Автор работы,
студент группы ЕТ-212
_____ Мухутдинов Б.А.
«____» _____ 2022 г.

Руководитель работы,
старший преподаватель
_____ Шелудько А.С.
«____» _____ 2022 г.

Челябинск 2022

1 ЗАДАНИЕ

Написать программу для выполнения аффинных преобразований многоугольника на плоскости. Предварительно определить структуру данных (класс) и разработать соответствующие подпрограммы (методы). Число и координаты вершин многоугольника считать из файла. Интерфейс программы должен содержать следующие элементы управления:

- перемещение фигуры;
- поворот фигуры (относительно центра фигуры);
- растяжение/сжатие фигуры;
- сохранение результата в файл;
- выход из программы.

2 ОПИСАНИЕ КЛАССА HEXAGON

Есть структура coords для хранения координат. В ней находятся следующие объекты:

- double x;
- double y.

В классе hexagon находятся объекты:

- coords center;
- coords external[6];
- int r;
- int rot=0.

В center хранится координаты центра шестиугольника, в массиве структур external хранится координаты вершин фигуры, в r радиус фигуры и в rot хранится как повернута фигура.

Методы класса:

- hexagon(double x, double y, int rad);
- void hexagon_remove();
- void hexagon_draw();
- void move(int direction);
- void rotate(int direction, int type);
- void change_size(int direction);
- void help(int type);
- void error();
- bool in_field(int dx);
- void calc_external_coords().

hexagon(double x, double y, int rad) - конструктор. hexagon_remove() - для удаления шестиугольника с экрана. hexagon_draw() - для рисования шестиугольника. move(int direction) - движение шестиугольника в определенном направлении. rotate(int direction, int type) - поворот шестиугольника влево или вправо. change_size(int direction) - для изменения размера шестиугольника. help(int type) - для вывода инструкции. error() - для вывода ошибки в случае выхода за границы экрана. bool in_field(int dx) - для проверки нахождения шестиугольника внутри экрана. void calc_external_coords() - приватная функция для вычисления вершин шестиугольника.

Реализация в task.cpp

Файл main.cpp

```
#include <iostream>
#include <graphics.h>
#include <task.hpp>
#include <control.h>
#include <fstream>

using namespace std;

int main(){
   int type = 0, radius;//type: 0- , 1-move, 2-rotate, 3-resize
   double x0, y0;
   ifstream file;
   file.open("info_for_figure.txt");
   file >> radius >> x0 >> y0;
   hexagon a = hexagon(x0, y0, radius);
   initwindow(WIDTH, HEIGHT, "well...");
   create_bg("bg.jpg");
   create_button(MOVE, 657, 378);
   create_button(ROTATE, 657, 463);
   create_button(RESIZE, 657, 547);
   create_button(SAVE, 657, 632);
   create_button(EXIT, 657, 718);
   setbkcolor(COLOR(235, 245, 238));
   setcolor(COLOR(70, 35, 122));
   setfillstyle(SOLID_FILL, COLOR(61, 220, 151));
   a.hexagon_draw();
   a.help(0);
   while(1){
      while(mousebuttons() != 1){
         switch(type){
            case 1:
               switch(getch(kbhit())){
                  case KEY_UP:
                     a.move(0);
                     break;
                  case KEY_DOWN:
                     a.move(1);
                     break;
                  case KEY_LEFT:
                     a.move(2);
                     break;
                  case KEY_RIGHT:
                     a.move(3);
                     break;
               }
               break;
            case 2:
               switch(getch(kbhit())){
```

```
                    case KEY_LEFT:
                        a.rotate(0, 0);
                        break;
                    case KEY_RIGHT:
                        a.rotate(1, 0);
                        break;
                }
                break;
            case 3:
                switch(getch(kbhit())){
                    case KEY_LEFT:
                        a.change_size(0);
                        break;
                    case KEY_RIGHT:
                        a.change_size(1);
                        break;
                }
                break;
        }
    }
    switch(select_control()){
        case MOVE:
            a.help(1);
            type = 1;
            break;
        case ROTATE:
            a.help(2);
            type = 2;
            break;
        case RESIZE:
            a.help(3);
            type = 3;
            break;
        case SAVE:
            save();
            break;
        case EXIT:
            return 0;
    }
  }
}
```

Файл task.hpp

```
#ifndef TASK_HPP
#define TASK_HPP
#define HEIGHT 800
#define WIDTH 1000

struct coords{
   double x;
   double y;
};

class hexagon{
   private:
      coords center;
      coords external[6];
      int r;
      int rot=0;
      void calc_external_coords();
   public:
      hexagon(double x, double y, int rad);
      void hexagon_remove();
      void hexagon_draw();
      void move(int direction);
      void rotate(int direction, int type);
      void change_size(int direction);
      void help(int type);
      void error();
      bool in_field(int dx);
};

void save();

#endif
```

Файл task.cpp

```
#define _USE_MATH_DEFINES
#include <graphics.h>
#include <task.hpp>
#include <math.h>

hexagon::hexagon(double x, double y, int rad){
   center.x = x;
   center.y = y;
   r = rad;
   calc_external_coords();
}

void hexagon::calc_external_coords(){
```

```
    double theta = 0.0;
    for(int i = 0; i<6; i++){
        external[i].x = center.x + r*cos(theta);
        external[i].y = center.y + r*sin(theta);
        theta+=M_PI/3;
    }
}

void hexagon::hexagon_draw(){
    for(int i = 0; i<6; i++){
        line(external[i].x, external[i].y, external[(i+1)%6].x, externa
    }
}

void hexagon::hexagon_remove(){
    setcolor(COLOR(235, 245, 238));
    for(int i = 0; i<6; i++){
        line(external[i].x, external[i].y, external[(i+1)%6].x, externa
    }
    setcolor(COLOR(70, 35, 122));
}

void hexagon::move(int direction){//direction: 0-up, 1-down equals fo
    int d[2] = {-2, 2};
    if(in_field(d[direction%2])){
        hexagon_remove();
        direction<2?center.y+=d[direction%2]:center.x+=d[direction%2];
        for(int i = 0; i<6; i++){
            direction<2?external[i].y+=d[direction%2]:external[i].x+=d[d
        }
        hexagon_draw();
    }else{
        error();
    }
}

void hexagon::rotate(int direction, int type){//direction: 0 - rotate
    int d[2] = {1, -1};
    hexagon_remove();
    coords temp;
    int reduce = 0;
    rot>0?reduce = -1:reduce=1;
    if(type == 0){
        rot += d[direction];
        for(int i=0; i < 6; i++){
            temp.x = center.x+(-center.x+external[i].x)*cos(d[direction]
            temp.y = center.y+(-center.x+external[i].x)*sin(d[direction]
            external[i] = temp;
        }
    }else{
        int temp_rot = rot;
        while(temp_rot != 0){
```

```
            for(int i=0; i < 6; i++){
               temp.x = center.x+(-center.x+external[i].x)*cos(-reduce*M
               temp.y = center.y+(-center.x+external[i].x)*sin(-reduce*M
               external[i] = temp;
            }
            temp_rot += reduce;
         }
      }
      hexagon_draw();
   }

   void hexagon::change_size(int direction){
      int d[2] = {-2, 2};
      hexagon_remove();
      if(in_field(d[direction])){
         r+=d[direction];
         calc_external_coords();
         rotate(0, 1);
      }else{
         error();
      }
   }

   void hexagon::help(int type){//type: 0-about program, 1-move, 2-rotat
      setbkcolor(COLOR(61, 220, 151));
      bar(670, 45, 1000, 370);
      switch(type){
         case 0:
            outtextxy(675, 50, " !");
            outtextxy(675, 50+textheight("), "   ");
            outtextxy(675, 50+textheight(")*2, "");
            outtextxy(675, 50+textheight(")*3, ");
            break;
         case 1:
            outtextxy(675, 50, "  );
            outtextxy(675, 50+textheight("), "   ");
            outtextxy(675, 50+textheight(")*2, ");
            break;
         case 2:
            outtextxy(675, 50, "  );
            outtextxy(675, 50+textheight("), "   ;
            outtextxy(675, 50+textheight(")*2, "<-  ->) ");
            break;
         case 3:
            outtextxy(675, 50, "  );
            outtextxy(675, 50+textheight("), "   ;
            outtextxy(675, 50+textheight(")*2, "<-  ->) ");
            break;
      }
      setbkcolor(COLOR(235, 245, 238));
   }
```

```
void hexagon::error(){
   setcolor(RED);
   outtextxy(675, 5, "ERROR!!");
   delay(150);
   bar(674, 4, 676+textwidth("ERROR!!"), 6+textheight("ERROR!!"));
   setcolor(COLOR(70, 35, 122));
}

bool hexagon::in_field(int dl){
   for(int i=0; i<6; i++){
      if(external[i].x + dl*10 >= 638) return false;
      if(external[i].x - dl*10 < 0) return false;
   }
   return true;
}

void save(){
   int width, height;
   IMAGE *output;

   width  = getmaxx() + 1;
   height = getmaxy() + 1;
   output = createimage(width, height);

   getimage(0, 0, width - 1, height - 1, output);
   saveBMP("output.jpg", output);
   freeimage(output);
}
```

Файл control.h

```
#ifndef CONTROL_H
#define CONTROL_H
#define dx 327
#define dy 62

enum CONTROLS {MOVE, ROTATE, RESIZE, SAVE, EXIT, N_CONTROLS};

struct button{
   int x0;
   int y0;
};

void create_button(int, int, int);
void create_bg(const char*);
int select_control();

#endif
```

Файл control.cpp

```
#include <graphics.h>
#include <control.h>

button Buttons[N_CONTROLS];

void create_button(int i, int x, int y){
   Buttons[i].x0 = x;
   Buttons[i].y0 = y;
}

void create_bg(const char *file_name){
   IMAGE *image;
   image = loadBMP(file_name);
   putimage(0, 0, image, COPY_PUT);
   freeimage(image);
}

int select_control(){
   int x, y;

   x = mousex();
   y = mousey();

   for (int i = 0; i < N_CONTROLS; i++)
   {
      if (x > Buttons[i].x0 && x < Buttons[i].x0 + dx &&
          y > Buttons[i].y0 && y < Buttons[i].y0 + dy)
      {
         return i;
      }
   }

   return -1;
}
```

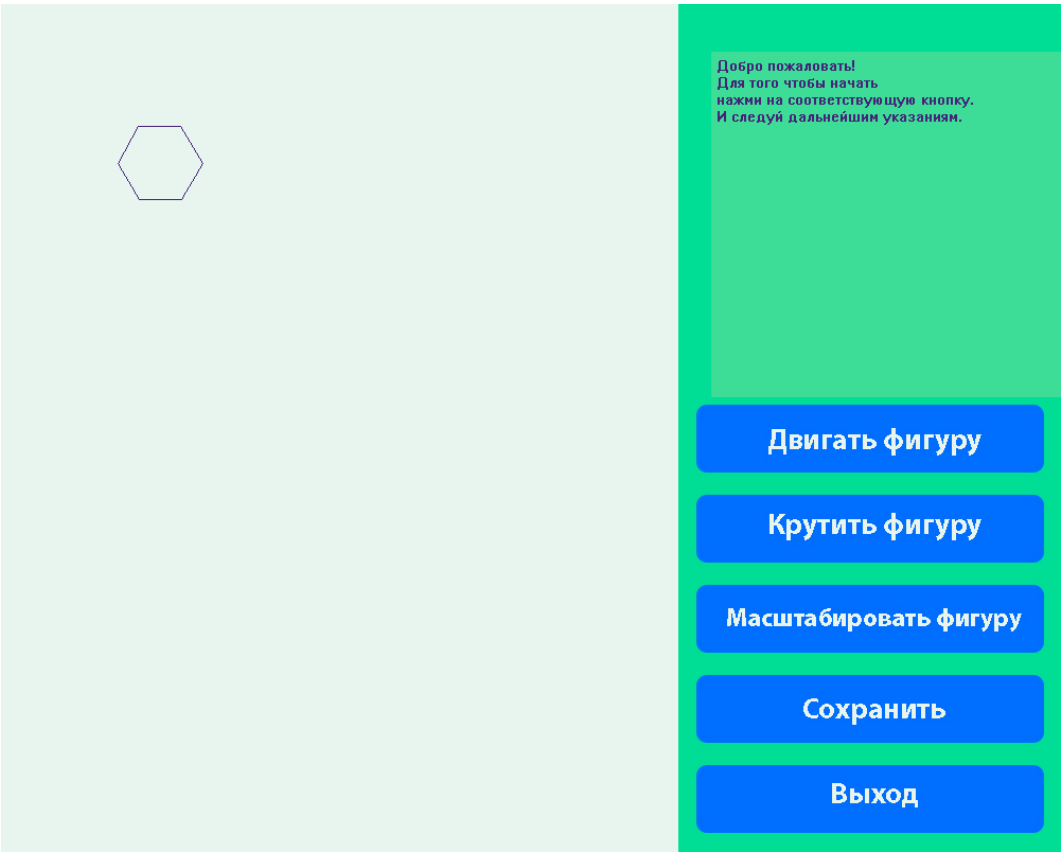

Рисунок 4.1 – Результат выполнения программы, после включения программы

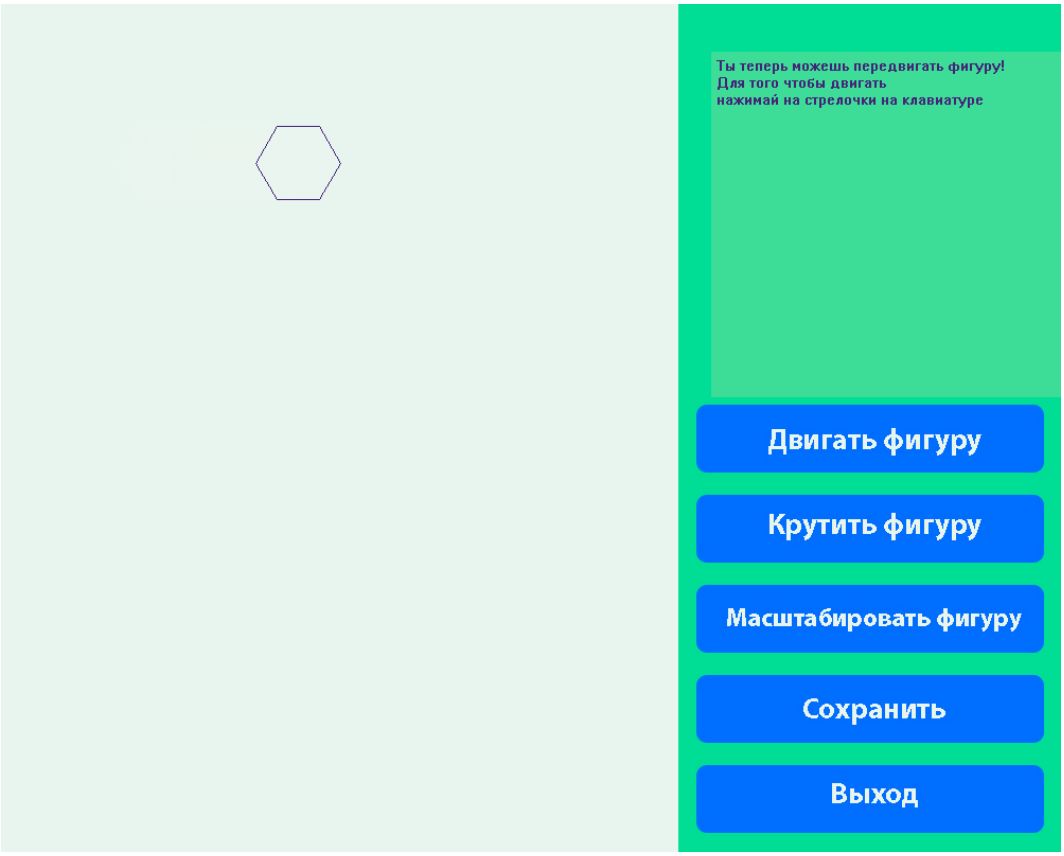

Рисунок 4.2 – Результат выполнения программы, после нажатия кнопки двигать фигуру

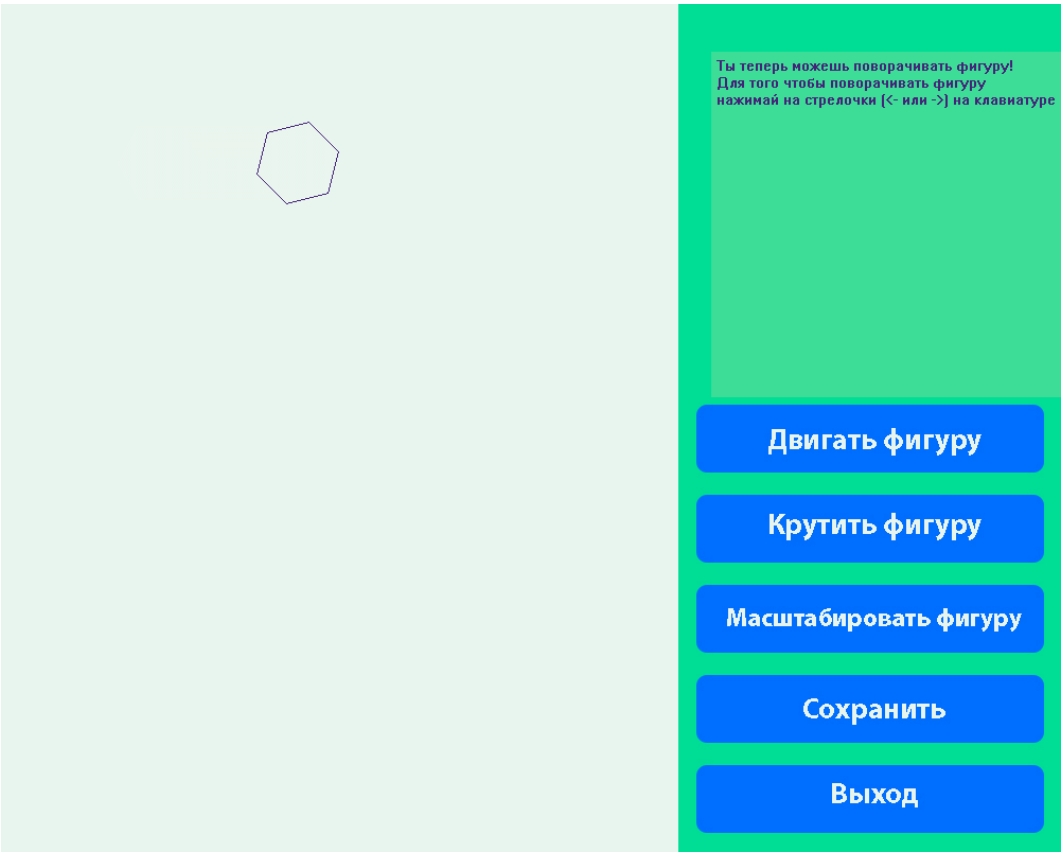

Рисунок 4.3 – Результат выполнения программы, после нажатия кнопки крутить фигуру

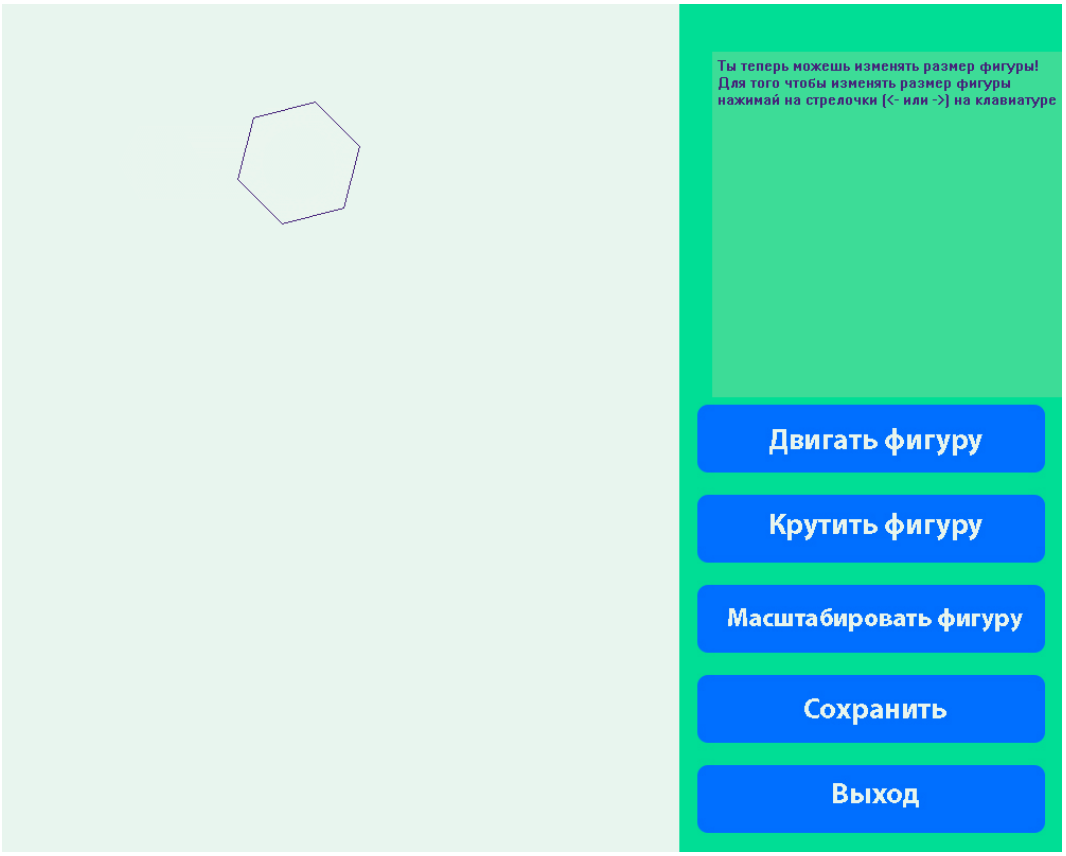

Рисунок 4.4 – Результат выполнения программы, после нажатия кнопки масштабировать фигуру

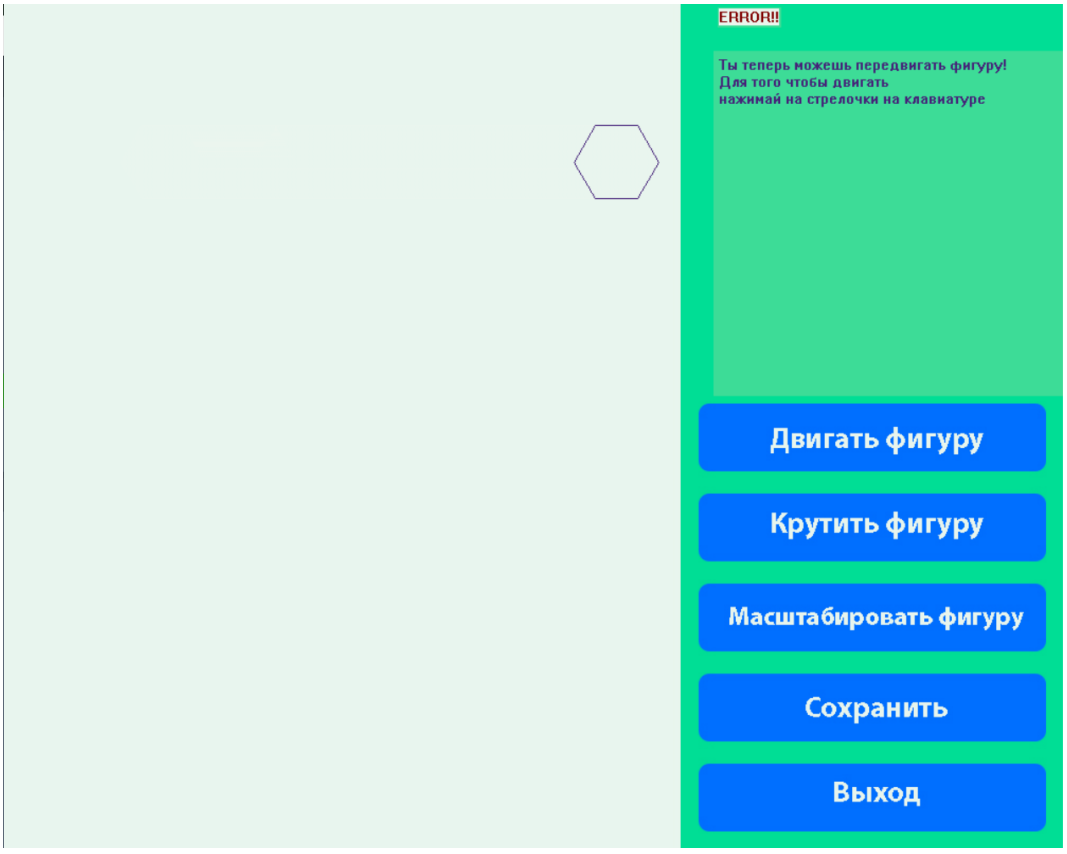

Рисунок 4.5 – Результат выполнения программы, случай попытки выхода за рабочий экран