**Лабораторная работа 1. Геометрические фракталы.**

**Студент:** Петров Артем Евгеньевич

**Группа:** НКНбд-01-21

**Студенческий билет:** 1032219251

**Цель:** написать компьютерную программу для построения геометрического фрактала «Дерево Пифагора».

**Условие:** За построение фрактала должна отвечать функция

void pifagor(int n, int x0,int y0,int a,double fi,double alpha)

Функция pifagor должна рекурсивно вызывать себя два раза с фактическими параметрами, соответствующими задаче построения двух «домиков» на скатах крыши. В программе должна быть установлена максимальная глубина рекурсии;

В функции pifagor должны вычисляться растровые координаты других, помимо точки привязки, вершин «домика»;

При построении фрактала должна использоваться функция line графической библиотеки.

**Дано:** n – глубина рекурсии построения фрактала,

x0, y0 – растровые координаты точки привязки «домика»,

a – длина основания «домика»,

fi – угол наклона основания «домика»,

alpha – угол наклона крыши «домика»;

**Ход работы:**

1. Подключим все необходимые бибилиотеки и воспользуемся именем окружения:

#include <D:\documents\c++\Computer graphics\graphics.h>

#include <iostream>

#include "graphics.h"

#include <stdlib.h>

#include <time.h>

#include <cmath>

using namespace std;

1. Основной функцией, выполняющей всю геометрическую логику, будет функция поворота прямой по координатам начала, длинне отрезка и углу поворта относительно горизонтального положения. Формула следующая(рисунок 1):

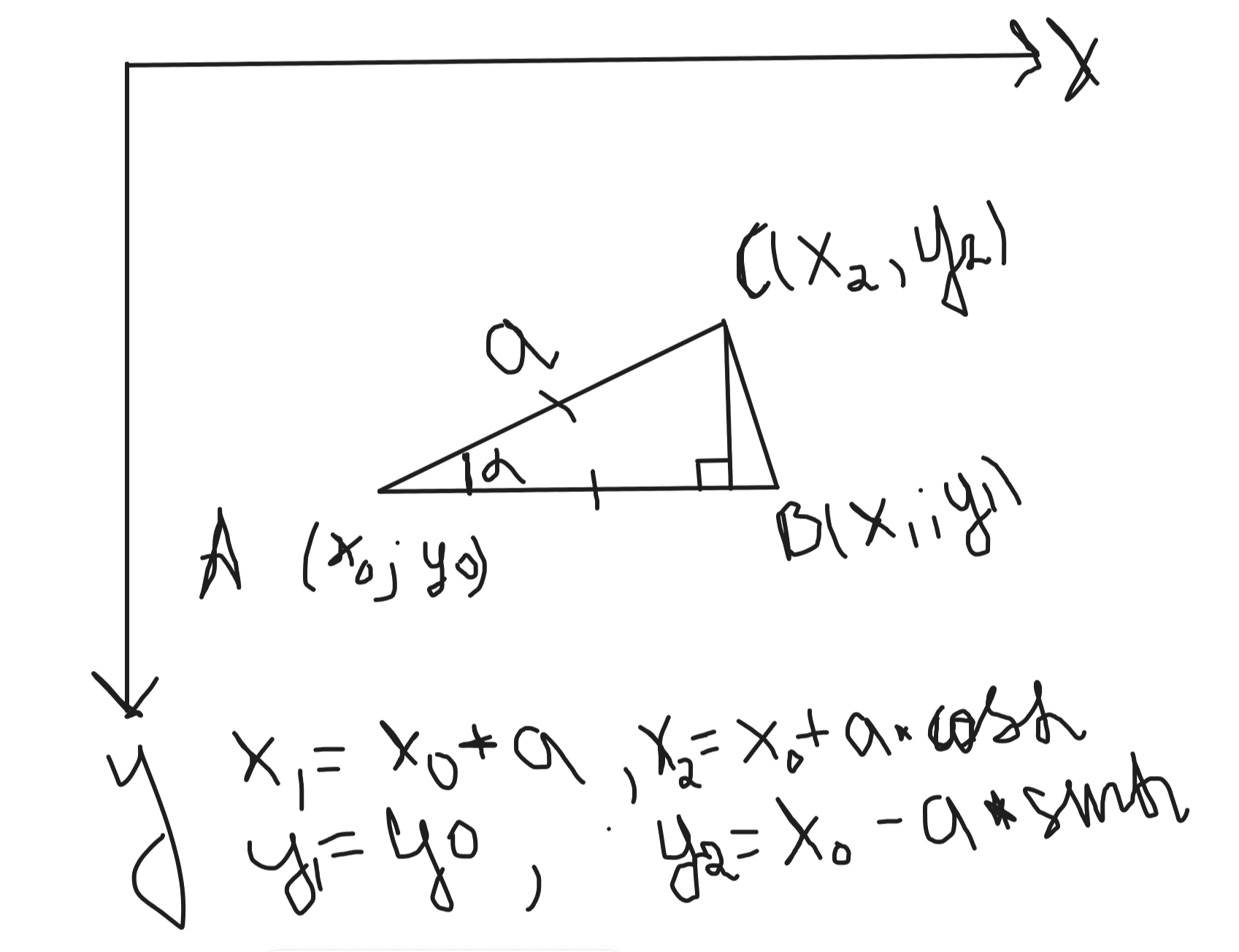


Рисунок 1. Вычисление вершины повернутой прямой.

Код же будет выглядеть так:

int\* rotate(int x0, int y0, int a, double alpha){

    /\*

    M = | cos(a) -sin(a) |

        | sin(a)  cos(a) |

    \*/

//    line(x0, y0, x0+cos(M\_PI\*alpha/180)\*(x0 + a) - sin(M\_PI\*alpha/180)\*)

    int x1;

    int y1;

    x1 = x0 + a \* cos(alpha \* M\_PI/180);

    y1 = y0 - a \* sin(alpha \* M\_PI/180);

    int \*dot = new int[2];

    dot[0] = x1; dot[1] = y1;

    return dot;

}

Возвращать же будем координаты нового конца перевернутой прямой.

1. Теперь опишем функцию, задающую правила построения квадрата:
2. int\* drawSquare(int x0, int y0, double alpha, int a, double fi){
3. // circle(x0, y0, 5);
4. int \*dotD = rotate(x0, y0, a, alpha);
5. line(x0, y0, dotD[0], dotD[1]);
6. int \*dotB = rotate(x0, y0, a, 90+alpha);
7. line(x0, y0, dotB[0], dotB[1]);
8. // circle(dotB[0], dotB[1], 5);
10. int\* dotC = rotate(dotB[0], dotB[1], a, alpha);
11. line(dotB[0], dotB[1], dotC[0], dotC[1]);
13. line(dotD[0], dotD[1], dotC[0], dotC[1]);
14. int\* roof = new int[2];
15. roof = rotate(dotB[0], dotB[1], a\*cos((fi)\*M\_PI/180), alpha+fi);
16. // circle(roof[0], roof[1], 5);
17. int \*ptr = new int[4];
18. ptr[0] = dotB[0];
19. ptr[1] = dotB[1];
20. ptr[2] = roof[0];
21. ptr[3] = roof[1];
22. return ptr;
23. }

Поочередно будет вычислять вершины нашего квадрата. Однако, самыми важными точками у нас будут В и точка будущего пересечения квадратов пифагора(см. Фотографию 2).

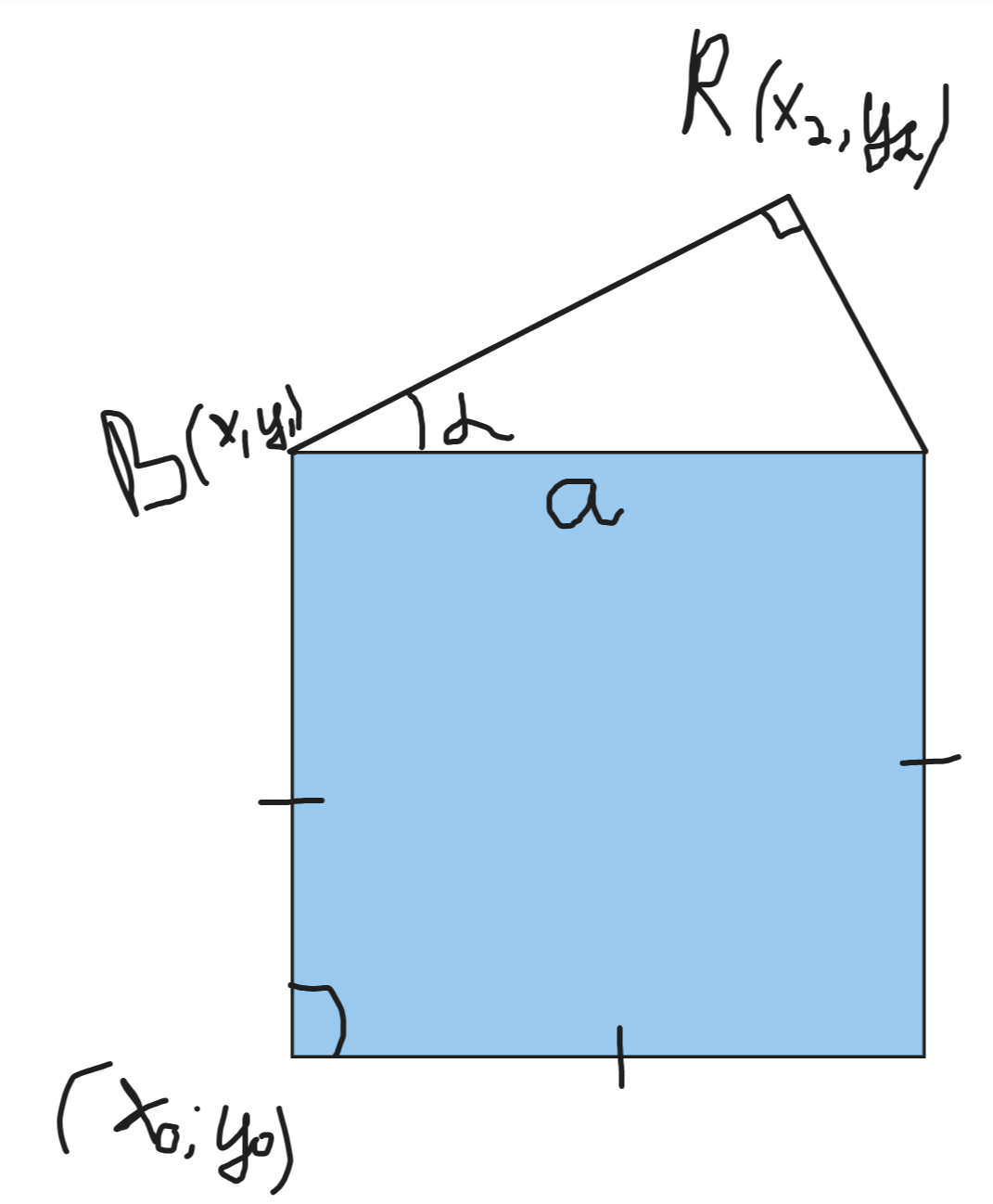


Рисунок 2. Расчет координат точек B и R для фрактального процесса.

5. Содержание самого рекурсивного алгоритма построения дерева Пифагора.

int pifagor(int n, int x0, int y0, int a, double fi, double alpha){

    if(n < 1 || a\*a <= 4){

        return 0;

    }

    else{

        int\* ptr = drawSquare(x0, y0, fi, a, alpha);

        // circle(ptr[2], ptr[3], 5);

        pifagor(n-1, ptr[0], ptr[1], a\*cos(alpha\*M\_PI/180), fi+alpha, alpha);

        pifagor(n-1, ptr[2], ptr[3], a\*sin(alpha\*M\_PI/180), -90+fi+alpha, alpha);

    }

}

Каждый рекурсивный вызов мы :

1. Уменьшаем количество вызовов на 1
2. Уменьшаем сторону на косинус угла alpha или синус alpha(для левого и правого квадратов соответственно). Кроме того, нужно обязательно каждый раз увеличивать или уменьшать угол поворота для новой пары квадратов.
3. Поворачиваем наши новые квадраты на fi+alpha для левого и -90+fi+alpha(так как квадраты постоянно сильнее наклоняютсяя)

6. Функция main:

int main(){

    // printf("idk");

    int gddriver = DETECT, gmode, errorcode;

    initgraph(&gddriver, &gmode, "");

    srand(time(NULL));

    // int\* ptr = drawSquare(1280, 720, 45, 30);

    pifagor(10, 700, 700, 150, 30, 30);

    getch();

    closegraph();

    return 0;

}

7. Результат построения(рисунок 3):

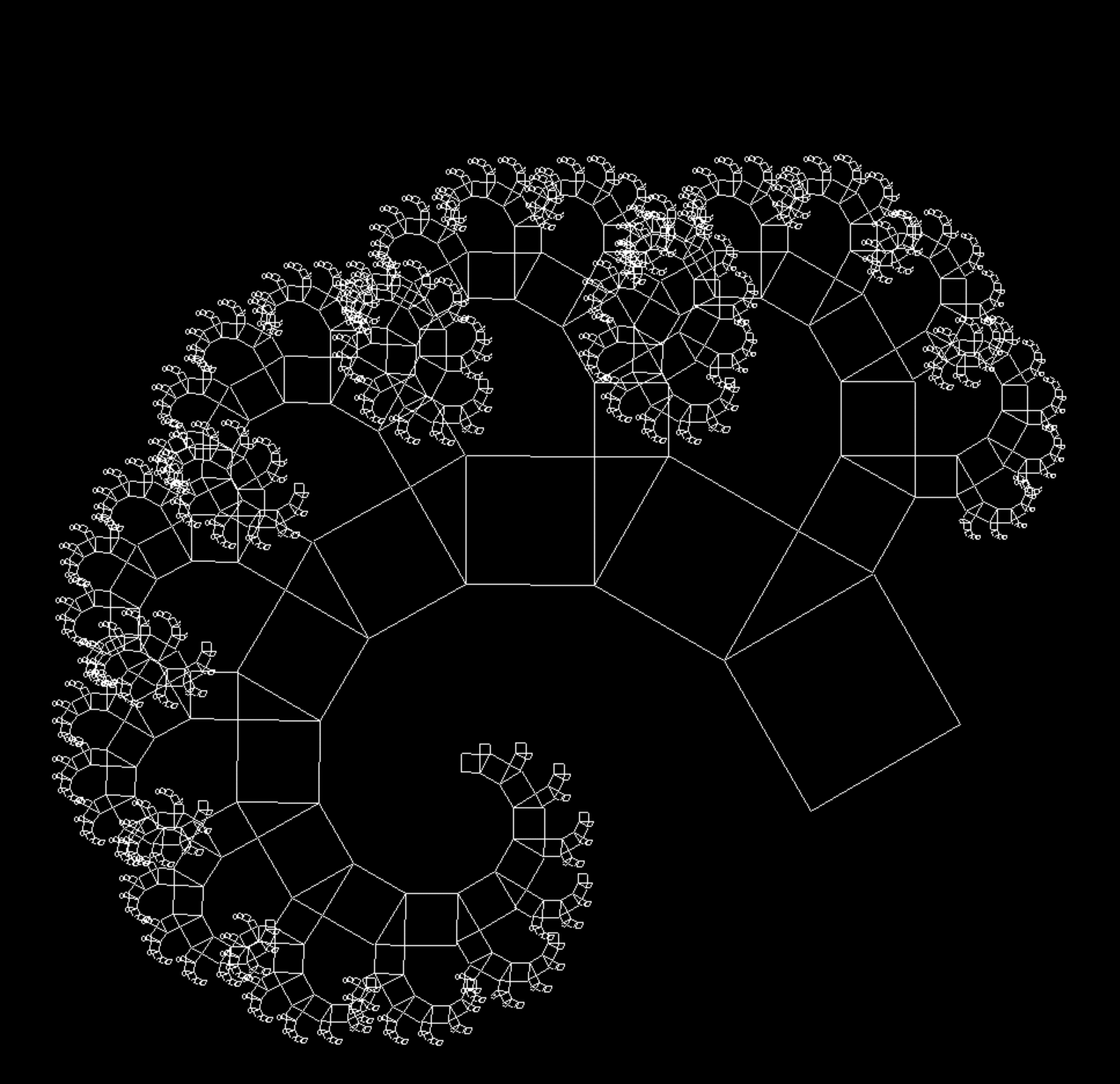


Рисунок 3. Дерева пифагора при заданных параметрах.

Вывод: Сейчас вы увидели программное построение дерева Пифагора с помощью рекурсивного алгоритма.