Министерство цифрового развития

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Сибирский государственный университет телекоммуникаций и информатики»

(СибГУТИ)

Кафедра прикладной математики и кибернетики

Отчёт

по лабораторной работе № 2 «Процессы и асинхронное взаимодействие»

Выполнил:

студент группы ИП-216

Русецкий А.С.

Работу проверил: Ассистент

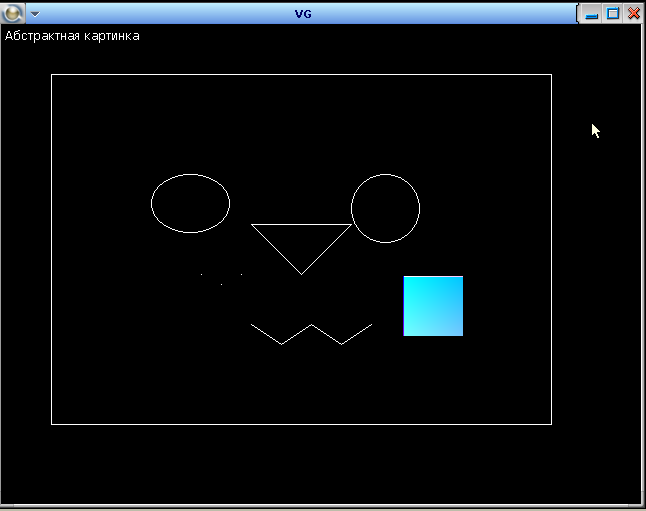
Шевелькова В.Ю.

Новосибирск 2025 г.

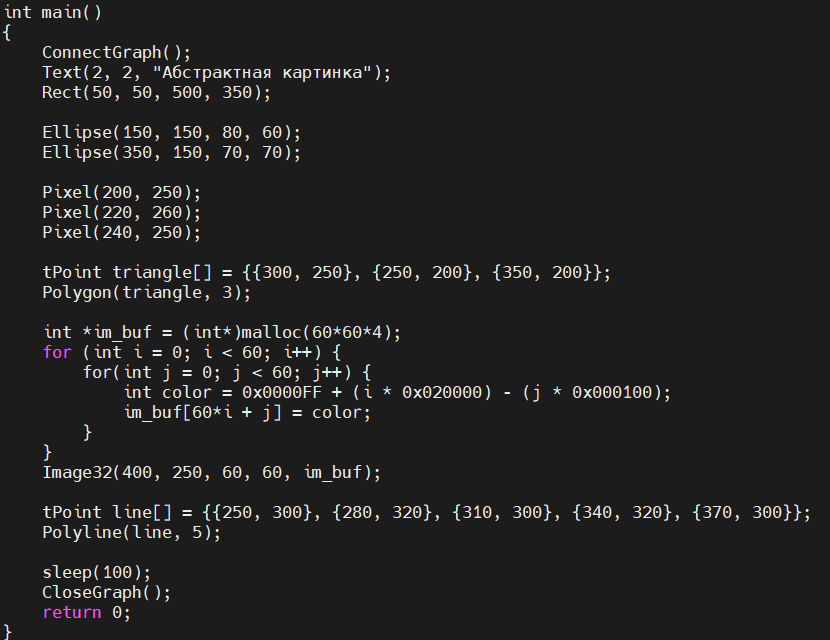
**Задание**

1. Тщательно изучить библиотеку VinGraph.
2. Используя функции библиотеки VinGraph, нарисовать абстрактную картину, которой представлены (почти) все доступные графические элементы.
3. Заставить нарисованные элементы двигаться независимо друг от друга с помощью параллельных процессов (можно изменять во времени положение, цвет, размеры, конфигурацию графических элементов). Предусмотреть завершение программы по нажатию на любую клавишу.
4. Нарисовать нечто, движущееся по замкнутой кривой. Организовать изменение траектории движения по нажатию на клавиши (организуя взаимодействие процессов через общую область памяти (shared memory)). В качестве фона можно использовать (оживленную) картину, созданную на предыдущих этапах работы.
5. Затем последнюю программу сделать с помощью нитей в одном процессе.

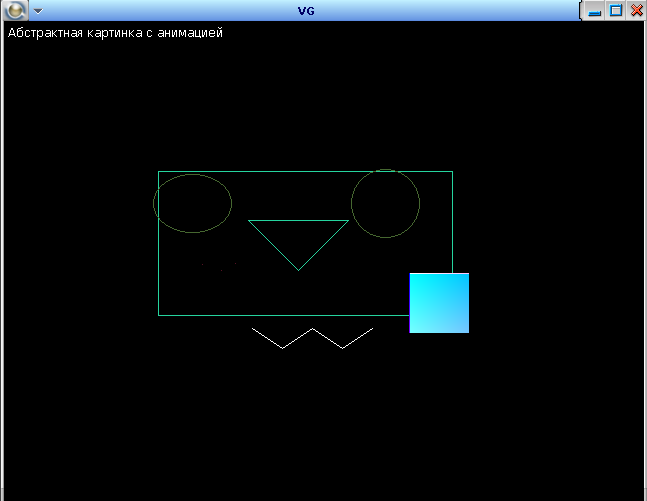
2) Выполнение программы, выводящей в терминал VinGraph почти все доступные графические элементы



Код программы:



3) Для этого задания взяли фигуры из прошлой программы. Все они двигаются независимо друг от друга в параллельных процессах.



Код программы:

#include <stdlib.h>

#include <unistd.h>

#include <vingraph.h>

#include <stdio.h>

#include <sys/types.h>

#include <process.h>

#include <time.h>

int main()

{

    ConnectGraph();

    Text(2, 2, "Абстрактная картинка с анимацией");

    // фигуры

    int rect = Rect(50, 50, 500, 350);

    int elip1 = Ellipse(150, 150, 80, 60);

    int elip2 = Ellipse(350, 150, 70, 70);

    int pix1 = Pixel(200, 250);

    int pix2 = Pixel(220, 260);

    int pix3 = Pixel(240, 250);

    tPoint triangle\_points[] = {{300, 250}, {250, 200}, {350, 200}};

    int polyg = Polygon(triangle\_points, 3);

    int \*im\_buf = (int\*)malloc(60\*60\*4);

    for (int i = 0; i < 60; i++) {

        for(int j = 0; j < 60; j++) {

            int color = 0x0000FF + (i \* 0x020000) - (j \* 0x000100);

            im\_buf[60\*i + j] = color;

        }

    }

    int img = Image32(400, 250, 60, 60, im\_buf);

    tPoint line\_points[] = {{250, 300}, {280, 320}, {310, 300}, {340, 320}, {370, 300}};

    int poly1 = Polyline(line\_points, 5);

    pid\_t proc1, proc2, proc3, proc4;

    int a = getpid();

    printf("\ncurrent process = %d", a);

    if((proc1 = fork()) != 0)

    {

        int b = getpid();

        printf("\ncurrent process 1 = %d", b);

        if((proc2 = fork()) != 0)

        {

            int c = getpid();

            printf("\ncurrent process 2 = %d", c);

            if((proc3 = fork()) != 0)

            {

                int d = getpid();

                printf("\ncurrent process 3 = %d", d);

                if((proc4 = fork()) != 0)

                {

                    int e = getpid();

                    printf("\ncurrent process 4 = %d\n", e);

                    InputChar();

                    CloseGraph();

                }

                else

                {

                    // Процесс 4: двигает изображение и ломаную линию

                    srand(time(0));

                    while(1)

                    {

                        int x = (rand() % 3) - 1;

                        int y = (rand() % 3) - 1;

                        Move(img, x, y);

                        x = (rand() % 3) - 1;

                        y = (rand() % 3) - 1;

                        Move(poly1, x, y);

                        delay(400);

                    }

                }

            }

            else

            {

                // Процесс 3: двигает прямоугольник и полигон

                srand(time(0));

                while(1)

                {

                    int c = RGB(rand() % 255, rand() % 255, rand() % 255);

                    int lr = (rand() % 60) - 30;

                    int x = (rand() % 3) - 1;

                    int y = (rand() % 3) - 1;

                    Move(rect, x, y);

                    SetColor(rect, c);

                    Enlarge(rect, lr, lr);

                    x = (rand() % 3) - 1;

                    y = (rand() % 3) - 1;

                    Move(polyg, x, y);

                    SetColor(polyg, c);

                    delay(300);

                }

            }

        }

        else

        {

            // Процесс 2: двигает эллипсы

            srand(time(0));

            while(1)

            {

                int c = RGB(rand() % 255, rand() % 255, rand() % 255);

                int x = (rand() % 3) - 1;

                int y = (rand() % 3) - 1;

                Move(elip1, x, y);

                SetColor(elip1, c);

                x = (rand() % 3) - 1;

                y = (rand() % 3) - 1;

                Move(elip2, x, y);

                SetColor(elip2, c);

                delay(200);

            }

        }

    }

    else

    {

        // Процесс 1: двигает пиксели

        srand(time(0));

        while(1)

        {

            int c = RGB(rand() % 255, rand() % 255, rand() % 255);

            int x = (rand() % 3) - 1;

            int y = (rand() % 3) - 1;

            Move(pix1, x, y);

            SetColor(pix1, c);

            x = (rand() % 3) - 1;

            y = (rand() % 3) - 1;

            Move(pix2, x, y);

            SetColor(pix2, c);

            x = (rand() % 3) - 1;

            y = (rand() % 3) - 1;

            Move(pix3, x, y);

            SetColor(pix3, c);

            delay(100);

        }

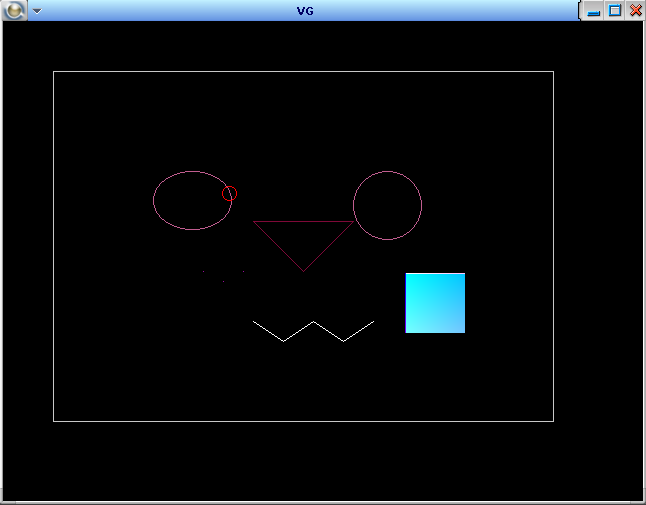
    }

    free(im\_buf);

    return 0;

}

4) Нарисован красный круг, движущийся по замкнутой кривой. На его движение можно влиять с помощью клавиш: W, A, S, D. Взаимодействие процессов реализовано через shared memory.



Код программы:

#include <stdlib.h>

#include <unistd.h>

#include <vingraph.h>

#include <stdio.h>

#include <sys/types.h>

#include <sys/wait.h>

#include <process.h>

#include <time.h>

#include <sys/mman.h>

#include <signal.h>

// Структура для разделяемой памяти

typedef struct {

    float a;      // радиус

    float b;      // форма

    float speed;  // скорость движения

    int trajectory\_type; // тип траектории (0-круг, 1-эллипс, 2-розетка, 3-сердце)

} SharedData;

// тригонометрические функции

float my\_cos(float x) {

    while (x < 0) x += 6.283185307f;

    while (x >= 6.283185307f) x -= 6.283185307f;

    float x2 = x \* x;

    float x4 = x2 \* x2;

    float x6 = x4 \* x2;

    return 1.0f - x2/2.0f + x4/24.0f - x6/720.0f;

}

float my\_sin(float x) {

    while (x < 0) x += 6.283185307f;

    while (x >= 6.283185307f) x -= 6.283185307f;

    float x2 = x \* x;

    float x3 = x2 \* x;

    float x5 = x3 \* x2;

    float x7 = x5 \* x2;

    return x - x3/6.0f + x5/120.0f - x7/5040.0f;

}

int main()

{

    ConnectGraph();

    int key = 0;

    pid\_t proc1, proc2, proc3, proc4;

    float phi = 0.0f;

    // Разделяемая память для параметров траектории

    SharedData \*shared = (SharedData\*)mmap(0, sizeof(SharedData),

                                         PROT\_READ | PROT\_WRITE,

                                         MAP\_SHARED | MAP\_ANON, -1, 0);

    // Инициализация разделяемой памяти

    shared->a = 60.0f;

    shared->b = 40.0f;

    shared->speed = 0.02f;

    shared->trajectory\_type = 0;

    // Фон

    Text(2, 2, "Движение по траектории - Управление: W,A,S,D,1,2,3,4,+,-");

    // отдельные процессы

    if((proc1 = fork()) == 0) {

        // Процесс 1: Анимированные эллипсы

        int el1 = Ellipse(150, 150, 80, 60);

        int el2 = Ellipse(350, 150, 70, 70);

        srand(time(0));

        while(1) {

            int c = RGB(rand() % 255, rand() % 255, rand() % 255);

            SetColor(el1, c);

            SetColor(el2, c);

            delay(200);

        }

    }

    if((proc2 = fork()) == 0) {

        // Процесс 2: Анимированные пиксели

        int pix1 = Pixel(200, 250);

        int pix2 = Pixel(220, 260);

        int pix3 = Pixel(240, 250);

        srand(time(0) + 1);

        while(1) {

            int c = RGB(rand() % 255, rand() % 255, rand() % 255);

            SetColor(pix1, c);

            SetColor(pix2, c);

            SetColor(pix3, c);

            delay(100);

        }

    }

    if((proc3 = fork()) == 0) {

        // Процесс 3: Анимированный треугольник

        tPoint tri\_points[] = {{300, 250}, {250, 200}, {350, 200}};

        int tri = Polygon(tri\_points, 3);

        srand(time(0) + 2);

        while(1) {

            int c = RGB(rand() % 255, rand() % 255, rand() % 255);

            SetColor(tri, c);

            delay(300);

        }

    }

    Rect(50, 50, 500, 350, 0, RGB(200, 200, 200));

    int \*im\_buf = (int\*)malloc(60\*60\*4);

    for (int i = 0; i < 60; i++) {

        for(int j = 0; j < 60; j++) {

            int color\_val = 0x0000FF + (i \* 0x020000) - (j \* 0x000100);

            im\_buf[60\*i + j] = color\_val;

        }

    }

    Image32(400, 250, 60, 60, im\_buf);

    tPoint line\_points[] = {{250, 300}, {280, 320}, {310, 300}, {340, 320}, {370, 300}};

    Polyline(line\_points, 5, RGB(255, 255, 255));

    // движущийся объект

    int moving\_obj = Ellipse(0, 0, 15, 15, RGB(255, 0, 0)); // красный круг

    if((proc4 = fork()) == 0) {

        // Процесс 4: Движение по траектории

        while(1) {

            float x, y, rho;

            // координаты в зависимости от типа траектории

            switch(shared->trajectory\_type) {

                case 0: // Круг

                    x = shared->a \* my\_cos(phi) + 300.0f;

                    y = shared->a \* my\_sin(phi) + 200.0f;

                    break;

                case 1: // Эллипс

                    x = shared->a \* my\_cos(phi) + 300.0f;

                    y = shared->b \* my\_sin(phi) + 200.0f;

                    break;

                case 2: // Розетка

                    rho = shared->a \* my\_cos(3.0f \* phi) + shared->b;

                    x = rho \* my\_cos(phi) + 300.0f;

                    y = rho \* my\_sin(phi) + 200.0f;

                    break;

                case 3: // Сердце

                    x = 16.0f \* my\_sin(phi) \* my\_sin(phi) \* my\_sin(phi);

                    y = 13.0f \* my\_cos(phi) - 5.0f \* my\_cos(2.0f \* phi) -

                        2.0f \* my\_cos(3.0f \* phi) - my\_cos(4.0f \* phi);

                    x = x \* (shared->a / 16.0f) + 300.0f;

                    y = -y \* (shared->b / 13.0f) + 200.0f; // инвертируем Y

                    break;

                default:

                    x = shared->a \* my\_cos(phi) + 300.0f;

                    y = shared->a \* my\_sin(phi) + 200.0f;

            }

            MoveTo((int)x, (int)y, moving\_obj);

            phi += shared->speed;

            if (phi > 6.283185307f) {

                phi = 0.0f;

            }

            delay(15);

        }

    }

    while(key != 27) { // 27 - ESC

        key = InputChar();

        printf("Клавиша: %d\n", key);

        switch(key) {

            case 'w': case 'W':

                shared->a += 5.0f;

                if (shared->a > 100.0f) shared->a = 100.0f;

                break;

            case 's': case 'S':

                shared->a -= 5.0f;

                if (shared->a < 20.0f) shared->a = 20.0f;

                break;

            case 'a': case 'A':

                shared->b += 5.0f;

                if (shared->b > 80.0f) shared->b = 80.0f;

                break;

            case 'd': case 'D':

                shared->b -= 5.0f;

                if (shared->b < 10.0f) shared->b = 10.0f;

                break;

            case '+':

                shared->speed += 0.005f;

                if (shared->speed > 0.1f) shared->speed = 0.1f;

                break;

            case '-':

                shared->speed -= 0.005f;

                if (shared->speed < 0.005f) shared->speed = 0.005f;

                break;

            case '1':

                shared->trajectory\_type = 0; // Круг

                break;

            case '2':

                shared->trajectory\_type = 1; // Эллипс

                break;

            case '3':

                shared->trajectory\_type = 2; // Розетка

                break;

            case '4':

                shared->trajectory\_type = 3; // Сердце

                break;

        }

    }

    // Завершение всех процессов

    kill(proc1, SIGTERM);

    kill(proc2, SIGTERM);

    kill(proc3, SIGTERM);

    kill(proc4, SIGTERM);

    waitpid(proc1, NULL, 0);

    waitpid(proc2, NULL, 0);

    waitpid(proc3, NULL, 0);

    waitpid(proc4, NULL, 0);

    free(im\_buf);

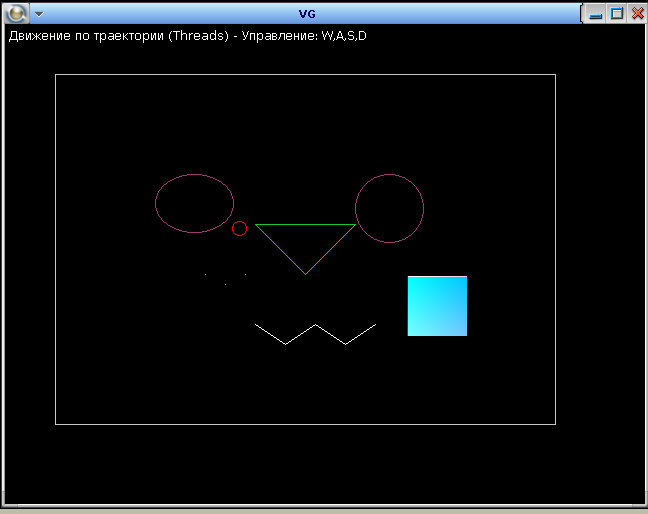
    munmap(shared, sizeof(SharedData));

    CloseGraph();

    return 0;

}

5) Предыдущая программа была изменена для работы с нитями.



Код программы:

#include <stdlib.h>

#include <unistd.h>

#include <vingraph.h>

#include <stdio.h>

#include <time.h>

#include <signal.h>

#include <pthread.h>

// Структура для общих данных

typedef struct {

    float a;      // радиус

    float b;      // форма

    float speed;  // скорость движения

    int trajectory\_type; // тип траектории (0-круг, 1-эллипс, 2-розетка, 3-сердце)

    int running;  // флаг работы программы

} SharedData;

// глобальные переменные для графических объектов

int el1, el2, pix1, pix2, pix3, tri, moving\_obj;

SharedData shared;

// тригонометрические функции

float my\_cos(float x) {

    while (x < 0) x += 6.283185307f;

    while (x >= 6.283185307f) x -= 6.283185307f;

    float x2 = x \* x;

    float x4 = x2 \* x2;

    float x6 = x4 \* x2;

    return 1.0f - x2/2.0f + x4/24.0f - x6/720.0f;

}

float my\_sin(float x) {

    while (x < 0) x += 6.283185307f;

    while (x >= 6.283185307f) x -= 6.283185307f;

    float x2 = x \* x;

    float x3 = x2 \* x;

    float x5 = x3 \* x2;

    float x7 = x5 \* x2;

    return x - x3/6.0f + x5/120.0f - x7/5040.0f;

}

// функция анимации эллипсов

void\* animate\_ellipses(void\* arg) {

    srand(time(0));

    while(shared.running) {

        int c = RGB(rand() % 255, rand() % 255, rand() % 255);

        SetColor(el1, c);

        SetColor(el2, c);

        delay(200);

    }

    return NULL;

}

// функция анимации пикселей

void\* animate\_pixels(void\* arg) {

    srand(time(0) + 1);

    while(shared.running) {

        int c = RGB(rand() % 255, rand() % 255, rand() % 255);

        SetColor(pix1, c);

        SetColor(pix2, c);

        SetColor(pix3, c);

        delay(100);

    }

    return NULL;

}

// функция анимации треугольника

void\* animate\_triangle(void\* arg) {

    srand(time(0) + 2);

    while(shared.running) {

        int c = RGB(rand() % 255, rand() % 255, rand() % 255);

        SetColor(tri, c);

        delay(300);

    }

    return NULL;

}

// функция движения по траектории

void\* move\_trajectory(void\* arg) {

    float phi = 0.0f;

    while(shared.running) {

        float x, y, rho;

        // координаты в зависимости от типа траектории

        switch(shared.trajectory\_type) {

            case 0: // Круг

                x = shared.a \* my\_cos(phi) + 300.0f;

                y = shared.a \* my\_sin(phi) + 200.0f;

                break;

            case 1: // Эллипс

                x = shared.a \* my\_cos(phi) + 300.0f;

                y = shared.b \* my\_sin(phi) + 200.0f;

                break;

            case 2: // Розетка

                rho = shared.a \* my\_cos(3.0f \* phi) + shared.b;

                x = rho \* my\_cos(phi) + 300.0f;

                y = rho \* my\_sin(phi) + 200.0f;

                break;

            case 3: // Сердце

                x = 16.0f \* my\_sin(phi) \* my\_sin(phi) \* my\_sin(phi);

                y = 13.0f \* my\_cos(phi) - 5.0f \* my\_cos(2.0f \* phi) -

                    2.0f \* my\_cos(3.0f \* phi) - my\_cos(4.0f \* phi);

                x = x \* (shared.a / 16.0f) + 300.0f;

                y = -y \* (shared.b / 13.0f) + 200.0f;

                break;

            default:

                x = shared.a \* my\_cos(phi) + 300.0f;

                y = shared.a \* my\_sin(phi) + 200.0f;

        }

        MoveTo((int)x, (int)y, moving\_obj);

        phi += shared.speed;

        if (phi > 6.283185307f) {

            phi = 0.0f;

        }

        delay(15);

    }

    return NULL;

}

int main()

{

    ConnectGraph();

    int key = 0;

    pthread\_t thread1, thread2, thread3, thread4;

    // Инициализация общих данных

    shared.a = 60.0f;

    shared.b = 40.0f;

    shared.speed = 0.02f;

    shared.trajectory\_type = 0;

    shared.running = 1;

    Text(2, 2, "Движение по траектории - Управление: W,A,S,D,ESC");

    Rect(50, 50, 500, 350, 0, RGB(200, 200, 200));

    int \*im\_buf = (int\*)malloc(60\*60\*4);

    for (int i = 0; i < 60; i++) {

        for(int j = 0; j < 60; j++) {

            int color\_val = 0x0000FF + (i \* 0x020000) - (j \* 0x000100);

            im\_buf[60\*i + j] = color\_val;

        }

    }

    Image32(400, 250, 60, 60, im\_buf);

    tPoint line\_points[] = {{250, 300}, {280, 320}, {310, 300}, {340, 320}, {370, 300}};

    Polyline(line\_points, 5, RGB(255, 255, 255));

    el1 = Ellipse(150, 150, 80, 60);

    el2 = Ellipse(350, 150, 70, 70);

    pix1 = Pixel(200, 250);

    pix2 = Pixel(220, 260);

    pix3 = Pixel(240, 250);

    tPoint tri\_points[] = {{300, 250}, {250, 200}, {350, 200}};

    tri = Polygon(tri\_points, 3);

    moving\_obj = Ellipse(0, 0, 15, 15, RGB(255, 0, 0));

    pthread\_create(&thread1, NULL, animate\_ellipses, NULL);

    pthread\_create(&thread2, NULL, animate\_pixels, NULL);

    pthread\_create(&thread3, NULL, animate\_triangle, NULL);

    pthread\_create(&thread4, NULL, move\_trajectory, NULL);

    // основной цикл

    while(key != 27 && shared.running) {

        key = InputChar();

        switch(key) {

            case 'w': case 'W':

                shared.a += 5.0f;

                if (shared.a > 100.0f) shared.a = 100.0f;

                break;

            case 's': case 'S':

                shared.a -= 5.0f;

                if (shared.a < 20.0f) shared.a = 20.0f;

                break;

            case 'a': case 'A':

                shared.b += 5.0f;

                if (shared.b > 80.0f) shared.b = 80.0f;

                break;

            case 'd': case 'D':

                shared.b -= 5.0f;

                if (shared.b < 10.0f) shared.b = 10.0f;

                break;

            case '+':

                shared.speed += 0.005f;

                if (shared.speed > 0.1f) shared.speed = 0.1f;

                break;

            case '-':

                shared.speed -= 0.005f;

                if (shared.speed < 0.005f) shared.speed = 0.005f;

                break;

            case '1':

                shared.trajectory\_type = 0; // Круг

                break;

            case '2':

                shared.trajectory\_type = 1; // Эллипс

                break;

            case '3':

                shared.trajectory\_type = 2; // Розетка

                break;

            case '4':

                shared.trajectory\_type = 3; // Сердце

                break;

        }

    }

    // Завершение работы

    shared.running = 0;

    // ожидание завершения всех нитей

    pthread\_join(thread1, NULL);

    pthread\_join(thread2, NULL);

    pthread\_join(thread3, NULL);

    pthread\_join(thread4, NULL);

    free(im\_buf);

    CloseGraph();

    return 0;

}