Министерство науки и высшего образования РФ

ФГАОУ ВО ЮЖНО-УРАЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ (НИУ)

Институт естественных и точных наук

Факультет математики, механики и компьютерных технологий

Кафедра прикладной математики и программирования

«Планетарная система. Универсальная демонстрационная астрономическая модель»

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА К ПРОЕКТУ   
по дисциплине «Учебная практика»

ЮУрГУ–01.03.02.2023.11.ПЗ

|  |  |  |
| --- | --- | --- |
|  |  | *Руководитель,*  *\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Демидов А.К.*  *«\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2023г.* |
| *Авторы работы:*  *Студент группы: ЕТ – 112*  *\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Бакалова Я.О.*  *Студент группы: ЕТ – 112*  *\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Бурматова Д.М.*  *Студент группы: ЕТ – 112*  *\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Щипков Т.В.* |  | *Студент группы: ЕТ – 112*  *\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Шаталов В.Д.*  *Студент группы: ЕТ – 112*  *\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Саломатина П.Д.*  *Студент группы: ЕТ – 112*  *\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Хрусталёв Д.А.*  *«\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2023г.* |
|  |  | *Работа защищена с оценкой*  *\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*  *«\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2023г.* |

Челябинск – 2023

Оглавление

[ВВЕДЕНИЕ 3](#_Toc140787628)

[1 ПОСТАНОВКА ЗАДАЧИ 4](#_Toc140787629)

[2 РАЗРАБОТКА АЛГОРИТМА 4](#_Toc140787630)

[2.1 Основной алгоритм 5](#_Toc140787631)

[3 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ 6](#_Toc140787632)

[ЗАКЛЮЧЕНИЕ 8](#_Toc140787633)

[БИБЛИОГРАФИЧЕСКИЙ СПИСОК 9](#_Toc140787634)

[ПРИЛОЖЕНИЕ А 10](#_Toc140787635)

# ВВЕДЕНИЕ

**Актуальность темы**. Основными вопросами коллективной разработки программного обеспечения является разделение труда и организация взаимодействия между участниками.

**Цель работы** – разработать программу «Планетарная система. Универсальная демонстрационная астрономическая модель».

**Задачи работы**:

– научиться ставить цели, формулировать задачи индивидуальной и совместной деятельности, решать поставленные задачи в кооперации с коллегами;

– научиться проводить поиск и анализ научной литературы и алгоритмов с последующим исследованием и разработкой математических моделей и программ для предложенных задач;

– ознакомиться с основными видами и задачами будущей профессиональной деятельности.

**Объект работы** – планетарная система. Универсальная демонстрационная астрономическая модель.

**Предмет работы** – применение технологий коллективной разработки программного обеспечения для разработки программы.

**Результаты работы** можно использовать в процессе последующего обучения в соответствии с учебным планом подготовки бакалавров по направлению «Прикладная математика и информатика».

# 1 ПОСТАНОВКА ЗАДАЧИ

Необходимо разработать программу для приложения «Планетарная система. Универсальная демонстрационная астрономическая модель». Для разработки необходимо использовать язык программирования C++ и графическую библиотеку winBGIm.

Программа должна считывать входной файл и вывести анимированную схему планетарной системы. Неподвижная центральная планета помещается в центр экрана.

Программа должна иметь следующие функции:

– масштабирование с помощью колёсика мыши;

– считывание с файла идентификатора планеты, идентификатора базовой планеты, радиуса планеты, радиуса орбиты, угловой скорости планеты;

– отображение комет (рисунок 1);

– отображение звёзд (см. рисунок 1);

– отображение планет, их спутников и орбит (см. рисунок 1).

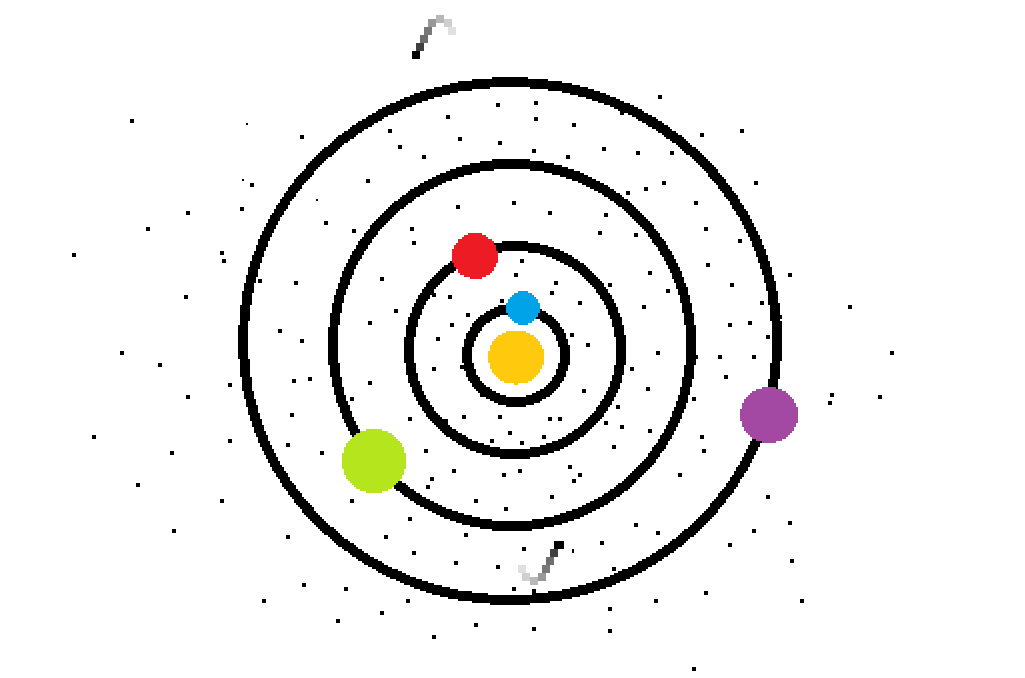
Примерный интерфейс программы показан на рисунке 1.

Рисунок 1. – Примерный интерфейс программы «Планетарная система. Универсальная демонстрационная астрономическая модель»

# 2 РАЗРАБОТКА АЛГОРИТМА

Входные данные программы записаны в текстовом файле. Каждая строка описывает одну планету с помощью пяти значений: ID, BASEID, R\_PAD, ORB\_RAD, SPEED.

Основные сущности в программе:

– одномерный массив звёзд stars\_rem, каждая звезда является структурой, содержащей координаты звезды и её цвет;

– вектор планет all\_planets, каждая планета является структурой, содержащей идентификатор планеты, идентификатор базовой планеты, радиус планеты, радиус орбиты планеты, угловая скорость планеты.

– уникальный идентификатор планеты ID – целое число;

– идентификатор базовой планеты BASEID, спутником которой является текущая (описываемая) планета – целое число;

– радиус планеты (в пикселях) R\_PAD – целое число;

– радиус орбиты планеты (в пикселях) ORB\_RAD – целое число;

– угловая скорость планеты SPEED – вещественное число.

Значения BASEID, ORB\_RAD и SPEED для центральной планеты системы (планеты с ID равным 0) игнорируются.

## 2.1 Основной алгоритм

При запуске программы сначала выполняется инициализация приложения. Далее запускается функция, которая считывает движения колёсика мыши и за счёт них масштабирует выводимое на экран изображение. После запускается считывание данных планет из текстового файла, отображаются звёзды и выводятся планеты, основываясь на входных данных из файла. Далее запускается основной цикл, в котором отображаются изменения положения планет и некоторых звёзд.

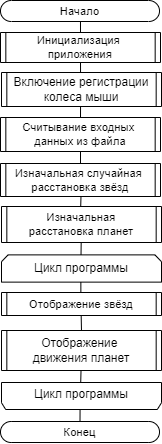
Схема основного алгоритма программы показана на рисунке 2.

Рисунок 2. – Схема основного алгоритма

# 3 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ

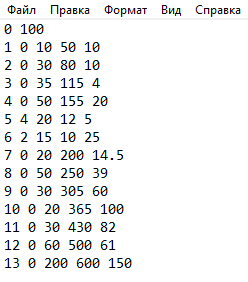
Пользователь вписывает данные о планетах в текстовый файл (рисунок 3).

Рисунок 3. – Входные данные в текстовом файле

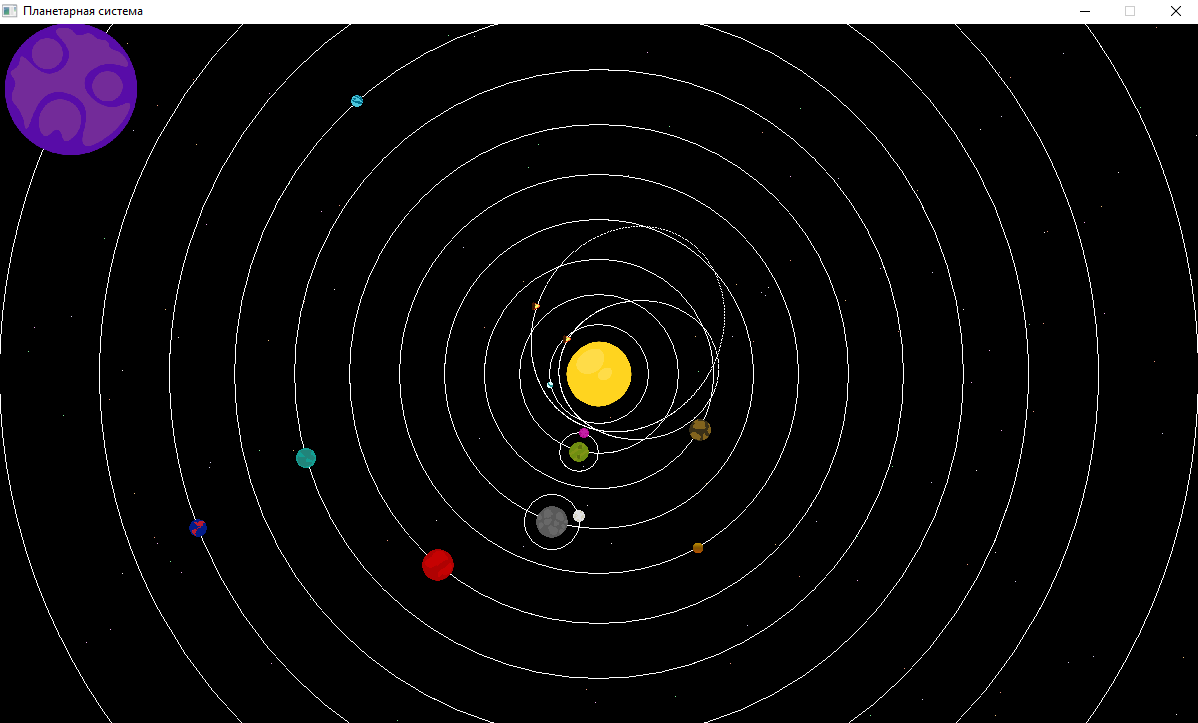
При запуске программы открывается окно с планетарной системой, планеты которой отображаются в соответствии с входными данными из файла (рисунок 4).

Рисунок 4. – Планетарная система

Пользователь может масштабировать изображения планет с помощью колёсика мыши (рисунок 5).

При нажатии на крестик в правом верхнем углу окна приложение закрывается.

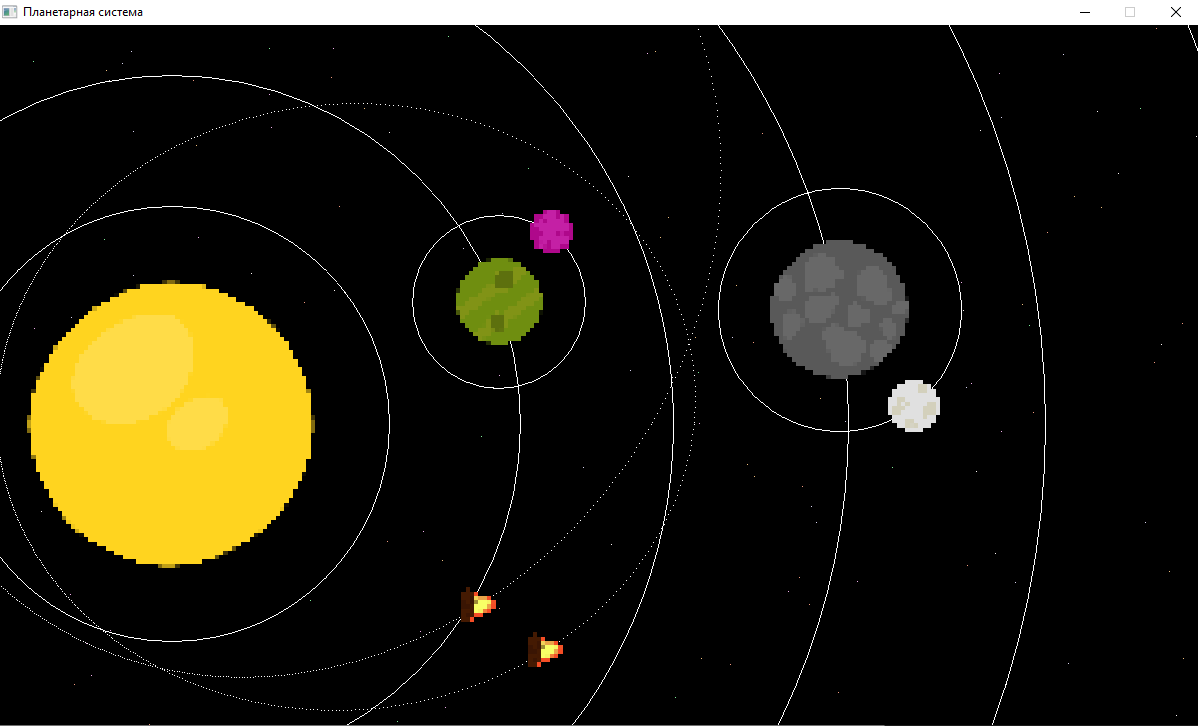


Рисунок 5. – Масштабирование планетарной системы

# ЗАКЛЮЧЕНИЕ

В ходе коллективной работы над проектом были поставлены точные требования к программе, затем были выявлены элементы интерфейса пользователя, разработаны необходимые математические модели, определены и детализированы структуры данных и алгоритмы. После завершения проектирования алгоритмы были реализованы на языке С++. Разработанный код был проверен на контрольных тестах и в код были внесены необходимые исправления. Для программы было разработано руководство пользователя. Таким образом, цель работы была достигнута, задачи – решены.

Результаты работы можно использовать в процессе последующего обучения в форме практических навыков коллективной разработки и организации взаимодействия на всех этапах разработки сложных программных систем.

# БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Лауферман, О. В. Разработка программного продукта: профессиональные стандарты, жизненный цикл, командная работа: учебное пособие / О. В. Лауферман, Н. И. Лыгина. — Новосибирск: НГТУ, 2019. — 75 с. — URL: https://e.lanbook.com/book/152251 (дата обращения: 18.07.2023). .

2. Солдатенко, И. С. Практическое введение в язык программирования Си: учебное пособие / И. С. Солдатенко, И. В. Попов. — Санкт-Петербург: Лань, 2018. — 132 с. — URL: https://e.lanbook.com/book/109619 (дата обращения: 18.07.2023).

3. Графическая библиотека WinBGIm. – URL: https://ipc.susu.ru/20786.html (дата обращения: 18.07.2023).

# ПРИЛОЖЕНИЕ а

А.1 Файл main.cpp

#include "stars.h"

#include "movement.h"

#include "graphics.h"

double scale = 1.0, scalen = 1.0;

double offx = 0, offy = 0, offxn = 0, offyn = 0;

extern remember stars\_rem[col\_stars];

void wheelhandler(int x, int y)

{

double grmin = 0.4, grmax = 15;

double xa,ya;

xa=mousex()/scale+offx;

ya=mousey()/scale+offy;

if(x<0 && scale \* 0.9 > grmin) scale\*=0.9; // вверх

else if (x > 0 && scale / 0.9 < grmax) scale/=0.9; // вниз

offx=-mousex()/scale+xa;

offy=-mousey()/scale+ya;

}

int main()

{

srand(time(NULL));

initwindow(WX, WY, "Планетарная система", 200, 200, true);

registermousehandler(WM\_MOUSEWHEEL,wheelhandler);

read\_file();

setbkcolor(BLACK);

clearviewport();

stars();

put\_planets();

while(true){

stars\_for\_main();

move\_planet();

scalen = scale; offxn = offx; offyn = offy;

delay(10);

}

return 0;

}

А.2 Файл movement.h

#ifndef \_const\_

#define \_const\_

#include <iostream>

#include <fstream>

#include <vector>

#include <string>

#include <cmath>

#include <ctime>

#include <algorithm>

#include <random>

#include "graphics.h"

using namespace std;

#define WX 1200

#define WY 700

#define col\_stars 150

#define M\_PI 3.141592653589793

typedef struct planet{

int ID, BASEID, X, Y;

int P\_RAD, ORB\_RAD;

double SPEED, t;

IMAGE \* bmp;

}Planet;

void read\_file();

void put\_planets();

void move\_planet();

void change\_dir(int index);

void comet(double p, double e, double phi);

#endif

А.3 Файл movement.cpp

#include "movement.h"

#include "graphics.h"

vector<Planet> all\_planets;

planet cometa;

extern double scale, scalen, offx, offy, offxn, offyn;

void move\_planet()

{

putimage((all\_planets[0].X - offxn)\*scalen, (all\_planets[0].Y - offyn)\*scalen, all\_planets[0].bmp, TRANSPARENT\_PUT, 2 \* all\_planets[0].P\_RAD \* scalen);

for (int i = all\_planets.size() - 1; i >= 1 ; i--) {change\_dir(i);}

comet(80, 0.5, 1);

comet(60, 0.5, 0.1);

swapbuffers();

clearviewport();

}

void change\_dir(int index)

{

int k, x, y;

double dt = 0.01;

double v = all\_planets[index].SPEED;

if (all\_planets[index].BASEID != 0){

k = all\_planets[index].BASEID;

int r = all\_planets[index].ORB\_RAD + all\_planets[k].P\_RAD;

int xe = all\_planets[k].X + all\_planets[k].P\_RAD;

int ye = all\_planets[k].Y + all\_planets[k].P\_RAD;

circle((xe - offxn) \* scalen, (ye - offyn) \* scalen, r \* scalen);

x = xe - all\_planets[index].P\_RAD + round(r \* cos(all\_planets[index].t));

y = ye - all\_planets[index].P\_RAD + round(r \* sin(all\_planets[index].t));

}

else{

int r = all\_planets[index].ORB\_RAD;

circle((WX / 2 - offxn) \* scalen, (WY / 2 - offyn) \* scalen, r \* scalen);

x = WX / 2 - all\_planets[index].P\_RAD + round(r \* cos(all\_planets[index].t));

y = WY / 2 - all\_planets[index].P\_RAD + round(r \* sin(all\_planets[index].t));

}

all\_planets[index].X = x; all\_planets[index].Y = y;

all\_planets[index].t += dt \* (v / all\_planets[index].ORB\_RAD);

if (all\_planets[index].t > 2 \* M\_PI)

all\_planets[index].t -= 2 \* M\_PI;

putimage((all\_planets[index].X - offxn)\*scalen, (all\_planets[index].Y - offyn)\*scalen, all\_planets[index].bmp, TRANSPARENT\_PUT, 2 \* all\_planets[index].P\_RAD \* scalen);

}

void read\_file()

{

string s;

ifstream F("pl.txt");

if (F.is\_open()){

while(getline(F, s))

{

vector<string> out;

stringstream ss(s);

string word;

while (ss >> word) {

out.push\_back(word);

}

planet temp;

if(out.size() == 2){

temp.ID = stoi(out[0]); temp.P\_RAD = stoi(out[1]) / 3 ;

}

else{

temp.ID = stoi(out[0]);

temp.BASEID = stoi(out[1]);

temp.P\_RAD = stoi(out[2]) / 3;

temp.ORB\_RAD = stoi(out[3]);

temp.SPEED = stod(out[4]);

}

temp.t = 0;

all\_planets.push\_back(temp);

}

F.close();

}

}

void comet(double p, double e, double phi){

double angle = cometa.t;

double r=p/(1-e\*cos(angle+phi));

for(double angle = 0;angle<acos(-1)\*2;angle+=0.01)

{

double r=p/(1-e\*cos(angle+phi));

putpixel((r\*cos(angle)+WX/2 - offxn) \* scalen , (r\*sin(angle)+WY/2 - offyn) \* scalen, WHITE);

}

putimage((r\*cos(angle)+WX/2 - offxn - 5) \* scalen, (r\*sin(angle)+WY/2 - offyn - 5) \* scalen, cometa.bmp, TRANSPARENT\_PUT, 10 \* scalen);

cometa.t += 0.01;

}

void put\_planets()

{

vector<int> pics;

for (int i = 1; i <= 30; i++) pics.push\_back(i);

random\_shuffle(pics.begin(), pics.end());

char s[30];

int n, k;

IMAGE \* bmp;

bmp = loadBMP("./Pic\_Plan/comet.bmp");

cometa.bmp = imageresize(bmp, 10, 10, COLORONCOLOR\_RESIZE);

for (int i = 0; i < all\_planets.size(); i++){

if (all\_planets[i].ID != 0){

n = pics[i];

sprintf(s, "./Pic\_Plan/Planet%d.bmp", n);

bmp = loadBMP(s);

}

else {bmp = loadBMP("./Pic\_Plan/Planet0.bmp");}

bmp = imageresize(bmp, all\_planets[i].P\_RAD \* 2, all\_planets[i].P\_RAD \* 2, COLORONCOLOR\_RESIZE);

all\_planets[i].bmp = bmp;

if (all\_planets[i].ID == 0 ) {

all\_planets[i].X = WX / 2 - all\_planets[i].P\_RAD; all\_planets[i].Y = WY / 2 - all\_planets[i].P\_RAD;

}

else{

if (all\_planets[i].BASEID == 0){

all\_planets[i].X = WX / 2 - all\_planets[i].P\_RAD + all\_planets[i].ORB\_RAD; all\_planets[i].Y = WY / 2 - all\_planets[i].P\_RAD;

}

else{

k = all\_planets[i].BASEID;

all\_planets[i].X = all\_planets[k].X + all\_planets[k].P\_RAD + all\_planets[i].ORB\_RAD; all\_planets[i].Y = all\_planets[k].Y + all\_planets[k].P\_RAD + all\_planets[i].ORB\_RAD;

}

}

putimage(all\_planets[i].X, all\_planets[i].Y, all\_planets[i].bmp, TRANSPARENT\_PUT);

}

swapbuffers();

}

А.4 stars.h

#ifndef \_STARS\_

#define \_STARS\_

typedef struct remember { // структура для хранения координат и цветов звёзд

int x, y, col;

} remember;

void rast\_stars(int i);

void stars();

void stars\_for\_main();

#endif

А.5 stars.cpp

#include <ctime>

#define WX 1200

#define WY 700

#define col\_stars 150

#include "stars.h"

#include "graphics.h"

int count\_st = 0;

int colors[10] = { COLOR(252, 245, 245), COLOR(127, 239, 153), COLOR(239, 153, 127), COLOR(214, 210, 239), COLOR(239, 191, 127), COLOR(242, 176, 237) }; // массив цветов звёзд

remember stars\_rem[col\_stars];

void rast\_stars(int i) { // случайный выбор расположений и цветов звёзд

stars\_rem[i].x = rand() % WX;

stars\_rem[i].y = rand() % WY;

stars\_rem[i].col = rand() % 7;

putpixel(stars\_rem[i].x, stars\_rem[i].y, colors[stars\_rem[i].col]);

}

void stars() { // изначальная случайная расстановка звёзд

srand(static\_cast<unsigned>(time(NULL)));

for (int i = 0; i < col\_stars; i++)

rast\_stars(i);

}

void stars\_for\_main() { // закрашивание случайных звёзд и отрисовка новых на смену закрашенным

if (count\_st == 40) {

int col\_help = rand() % col\_stars;

rast\_stars(col\_help);

count\_st = 0;

}

count\_st++;

clearviewport();

for (int j = 0; j < col\_stars; j++)

putpixel(stars\_rem[j].x, stars\_rem[j].y, colors[stars\_rem[j].col]);

}