МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«МИРЭА – Российский технологический университет»

РТУ МИРЭА

Институт информационных технологий (ИТ)
Кафедра инструментального и прикладного программного обеспечения (ИиППО)

КУРСОВОЙ ПРОЕКТ (РАБОТА)

по дисциплине: Объектно-ориентированное программирование
по профилю: Анализ данных
направления профессиональной подготовки: Прикладная математика (01.03.04), бакалавриат

Тема: «Разработка компьютерной экологической модели “Волчий остров”»

Студент: Блинков Денис Евгеньевич
Группа: ИМБО-01-18
Работа представлена к защите_____(дата)_____/_____/
(подпись и Ф. И. О. студента)

Руководитель: Строганкова Наталья Владимировна, ассистент кафедры ИиППО

Работа допущена к защите_____(дата)_____/_____/
(подпись и Ф. И. О. руководителя)

Оценка по итогам защиты: _____
_____ / _____ /
_____ / _____ /
(подписи, дата, Ф. И. О., должность, звание, уч. степень двух преподавателей, принявших защиту)

М. МИРЭА. 2019 г.

УДК 004.4

Блинков Д.Е. Разработка компьютерной экологической модели “Волчий остров” / **Курсовая работа** по дисциплине «Объектно-ориентированное программирование» профиля «Анализ данных» направления профессиональной подготовки бакалавриата 01.03.04. «Прикладная математика» (2ой семестр) / руководитель асс. Н.В. Строганкова / кафедра ИиППО Института ИТ РТУ МИРЭА.

Целью работы является разработка компьютерной экологической модели “Волчий остров”.

В рамках работы осуществлены анализ предметной области, проектирование и разработка приложения. Была разработана логика игры и построена модель.

Blinkov D.E. Development of the computer ecological model “Wolf Island” / Coursework in the discipline “Object-Oriented Programming” of the “Data Analysis” profile in the direction of undergraduate training 01.03.04. "Applied Mathematics" (2nd semester) / Head Ass. N.V. Strogankov / Department of the Institute of IT and RPO ITU RTU MIREA.

The aim of the work is to develop a computer ecological model “Wolf Island”.

As part of the work, the analysis of the subject area, design and development of the application were carried out The logic of the game was developed and a model was built.

М. МИРЭА. Ин-т ИТ. Каф. ИиППО. 2019 г. Блинков Д.Е.

МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«МИРЭА – Российский технологический университет»

РТУ МИРЭА

Институт информационных технологий (ИТ)
Кафедра инструментального и прикладного программного обеспечения (ИнППО)

ЗАДАНИЕ НА ВЫПОЛНЕНИЕ КУРСОВОЙ РАБОТЫ

по дисциплине: Объектно-ориентированное программирование
по профилю: «Анализ данных»
направления профессиональной подготовки: 01.03.04 Прикладная математика (академический/прикладной бакалавриат)

Студент: Блинков Денис Евгеньевич
Группа: ИМБО-01-18
Срок представления к защите 20.05.2019 г.

Руководитель: Строганкова Н.В.
Тема: «Разработка компьютерной экологической модели “Волчий остров”»

Исходные данные:
1) Индивидуальное задание на курсовую работу.
2) Среда разработки: C++ Clion.
3) Комплект разработчика приложений на языке C++, руководство по языку программирования C++, стандартные библиотеки классов C++.
4) Интернет-ресурсы и источники по теме курсовой работы.

Перечень вопросов к разработке (основная задача), графические материалы (слайды):
1) Разработать техническое задание на программный продукт.
2) Изучить и научиться применять принципы поэтапной разработки и отладки программ средней сложности.

3) Проанализировать предметную область согласно заданию и разработать собственные классы, разработать реализацию классов, а также разработать алгоритмы, необходимые для функционирования программы.
4) Разработать интерфейс для программы.
5) Разработать тесты для проверки работоспособности программного продукта.
6) Разработать руководство по использованию разработанного программного продукта.
7) Перечень обязательных графических материалов: UML-диаграмма(ы), Листинги программы.

Руководителем произведён инструктаж по технике безопасности, противопожарной технике и правилам внутреннего распорядка.

Заведующий кафедрой ИиППО: _____ /В.А. Мордвинов/, дата: 18.02.19

Задание на КР выдал _____ (ассистент кафедры ИиППО Строганкова Н.В.)

«18» 02 2019 г.

Задание на КР получил Блинков (Блинков Д.Е)

«18» 02 2019 г.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Наверняка многие слышали про экологическую модель под названием “Волчий остров”. Эта экологическая модель представляет из себя Волчий остров размером 20 × 20 заселен дикими кроликами, волками и волчицами. Имеется по несколько представителей каждого вида. Кролики довольно глупы: в каждый момент времени они с одинаковой вероятностью 1/9 передвигаются в один из восьми соседних квадратов (за исключением участков, ограниченных береговой линией) или просто сидят неподвижно. Каждый кролик с вероятностью 0.2 превращается в двух кроликов. Каждая волчица передвигается случайным образом, пока в одном из соседних восьми квадратов не окажется кролик, за которым она охотится. Если волчица и кролик оказываются в одном квадрате, волчица съедает кролика и получает одно "очко". В противном случае она теряет 0,1 "очка". Волки и волчицы с нулевым количеством очков умирают.
В начальный момент времени все волки и волчицы имеют 1 очко. Волк ведет себя подобно волчице до тех пор, пока в соседних квадратах не исчезнут все кролики; тогда если волчица находится в одном из восьми близлежащих квадратов, волк гонится за ней. Если волк и волчица окажутся в одном квадрате и там нет кролика, которого нужно съесть, они производят потомство случайного пола.

Эта задача не является игрой в классическом понимании этого термина. У этой задачи другое применение. Экологическая модель “Волчий остров” помогает приблизиться к моделированию реальной ситуации, которая обстоит в природе. Мы не можем знать наверняка, что произойдет, если мы реальных животных поставим в условие экологической модели “Волчий Остров”. Поэтому я создал эту программу, опираясь на математическое моделирование ситуации с помощью программирование, чтобы посмотреть исход ситуации при различных параметрах: геометрические размеры поля, начальное количество волков, волчиц, кроликов, их начальное

местоположение. В создании такой программы есть определенный практический смысл (польза):

1. Можно смоделировать реальную ситуацию, используя только вычислительные мощности компьютера.
2. Нет необходимости проводить данные эксперименты в реальной жизни.
3. Можно улучшить данную экологическую модель, чтобы она была максимальна была похожа на ситуацию в природе за счёт увеличение сложности и увеличении количества параметров, от которых зависит исход эксперимента (потому что в реальной жизни эта модель имеет не одну сотню параметров, от которых зависит исход).
4. Можно менять различные параметры и смотреть, как меняется исход эксперимента (Например, поменять начальное количество кроликов, их расположение, количество волков и т.д.)

Разработку экологической системы “Волчий остров” я выбрал в качестве темы для своей курсовой работы.

1. АНАЛИЗ ПРОГРАММНЫХ СРЕДСТВ

Изначально идея разработки такой экологической системы возникла из практических соображений – нужно было провести множество экспериментов и быстро. В условиях реальной жизни это сделать очень непросто, с помощью вычислительных мощностей компьютера – намного легче. Затем не только исследователей подстегнул интерес в создании такой экологической модели, но и программистов. Поэтому данная модель реализована на многих платформах, с различными реализациями, эвристиками. Рассмотрим некоторые их них.

1.1.«Волчий остров» на Android.

Приложение предназначено для смартфонов. Я рассмотрел конкретное приложение, которое я нашел в Google play. Среди особенностей данной реализации можно выделить:

1. Понятный графический интерфейс. Возможность быстро поменять начальные параметры модели.
2. Под игровым окном пишется текущее количество всех видов животных.
3. Усложненная экологическая модель - включает много параметров, такие как сытость волков, волчиц и кроликов, беременность волчиц, добавлена гибель от недостатка пищи, созревание травы и некоторые другие.

Интерфейс приложения (рис. 1.1.) довольно интересен и даёт достаточное полное понимание текущего развития ситуации, даёт возможность анализа ситуации.

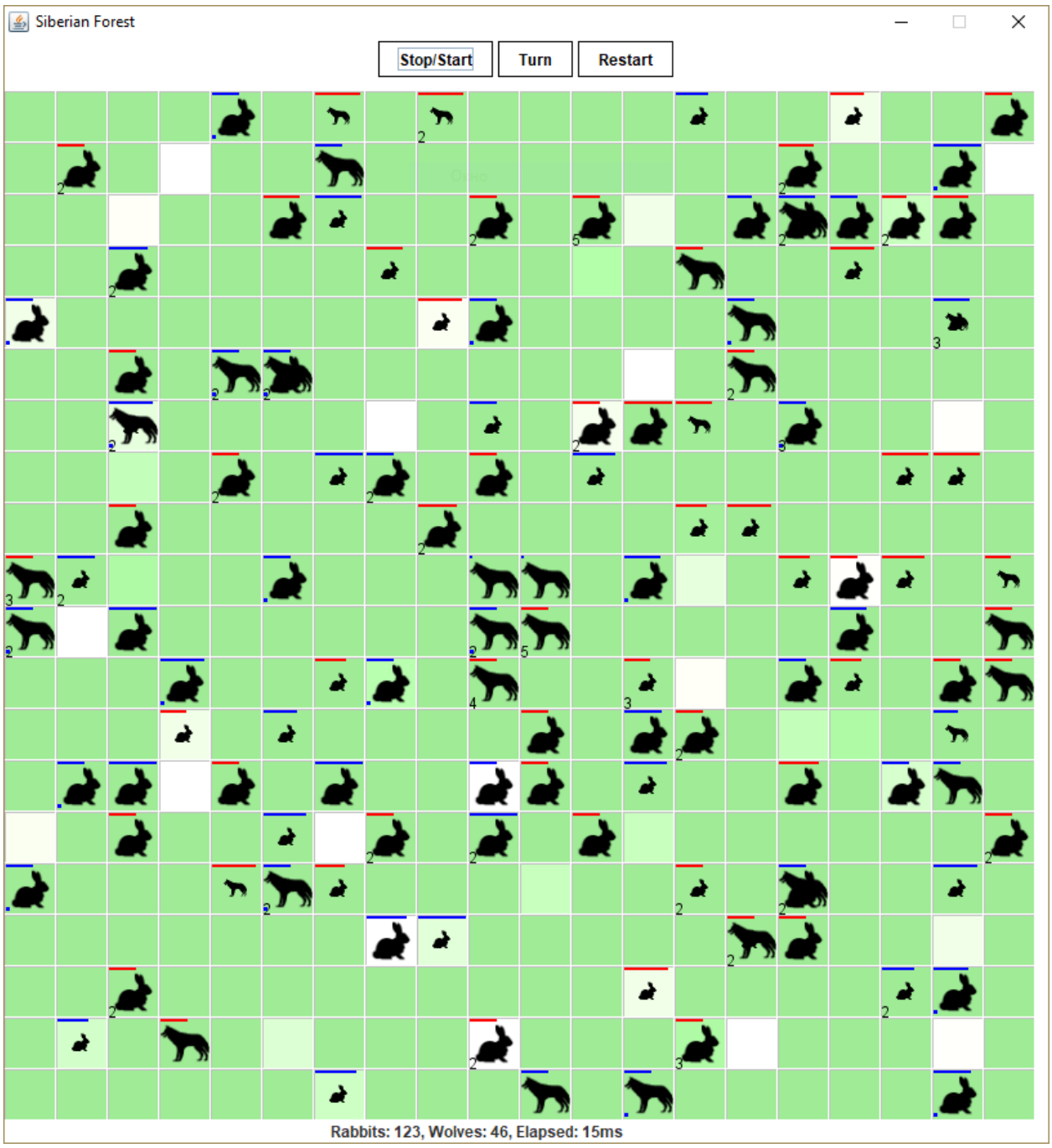


Рис. 1.1. Интерфейс приложения «Волчий остров» на Android

1.2.Выводы к главе 1

Рассмотрев реализацию экологической модели выше, можем поставить следующие задачи для собственной имплементации данной модели:

1. Спроектировать само приложение – какие классы будут созданы, какие параметры будут присутствовать, как примерно должен развиваться экологический процесс.
2. Разработать алгоритмы, которые будут управлять животными, моделируя тем самым условия реальной жизни.
3. Протестировать разработанный продукт на различных параметрах, выявить некоторые закономерности, выдвинуть гипотезы.

2. МЕТОДОЛОГИЯ РАЗРАБОТКИ ПРИЛОЖЕНИЯ

2.1.Выбор языка программирования

Для разработки приложения был выбран язык программирования C++, принадлежащий семейству объектно-ориентированных языков программирования. Именно на этом языке ведется обучение основам объектно-ориентированного программирования в этом семестре.

2.2.Выбор подхода к созданию продукта

Подход к написанию продукта я выбрал не совсем стандартный. Изначально я написал прототип данной модели на языке программирования Python, отладил процесс, устранил ошибки. Затем я перенес написанную ранее программу на язык C++. Это так называемым подход “Прототипирование -> продакшен”.

2.3.Выбор среды для разработки приложения

Для разработки приложения будет использоваться среда Clion от компании JetBrains. Она включает в себя удобный отладчик и редактор кода. Приложение будет разрабатываться как консольное, поэтому разработка интерфейса не будет зависеть от выбора программной среды разработки.

2.4.Разработка структуры приложения

Приложение будет обладать следующими функциями:

- Консольный интерфейс.
- Поясняющее руководство по использованию приложения.
- Мониторинг текущего состояния симуляции.
- Настройка параметров модели симуляции.
- Вывод основной информации после завершения симуляции.

2.5.Интерфейс приложения

2.5.1. Начальное окно

Начальное окно (рис. 2.1.) содержит информацию об основных сведениях приложения (поясняющее руководство), которые нужно знать для работы.

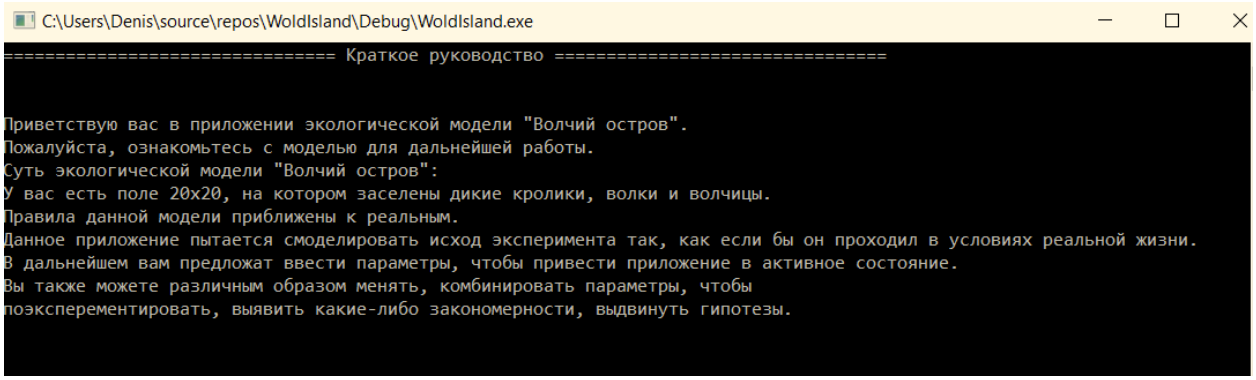


Рис. 2.1. Инструкция при запуске приложения.

2.5.2. Окно с настройкой модели

Далее предлагается ввести желаемое количество зверей и выбрать координаты их расположения (рис. 2.2.).

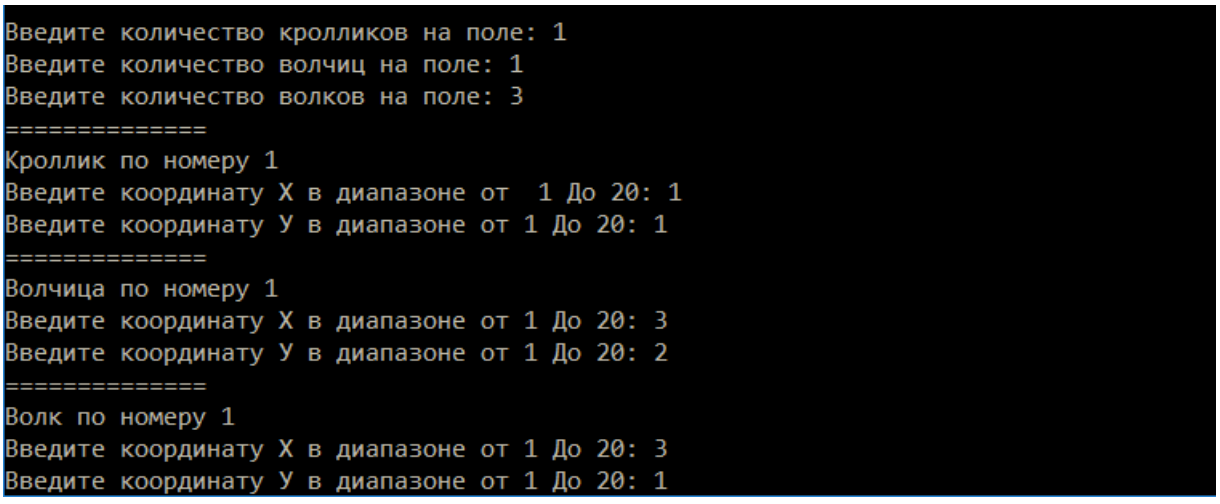


Рис. 2.2. Выбор количества всех зверей на карте.

2.5.3. Окно игры

После настройки параметров модель переходит в активное состояние. Происходит симуляция реального мира (рис. 2.3.).

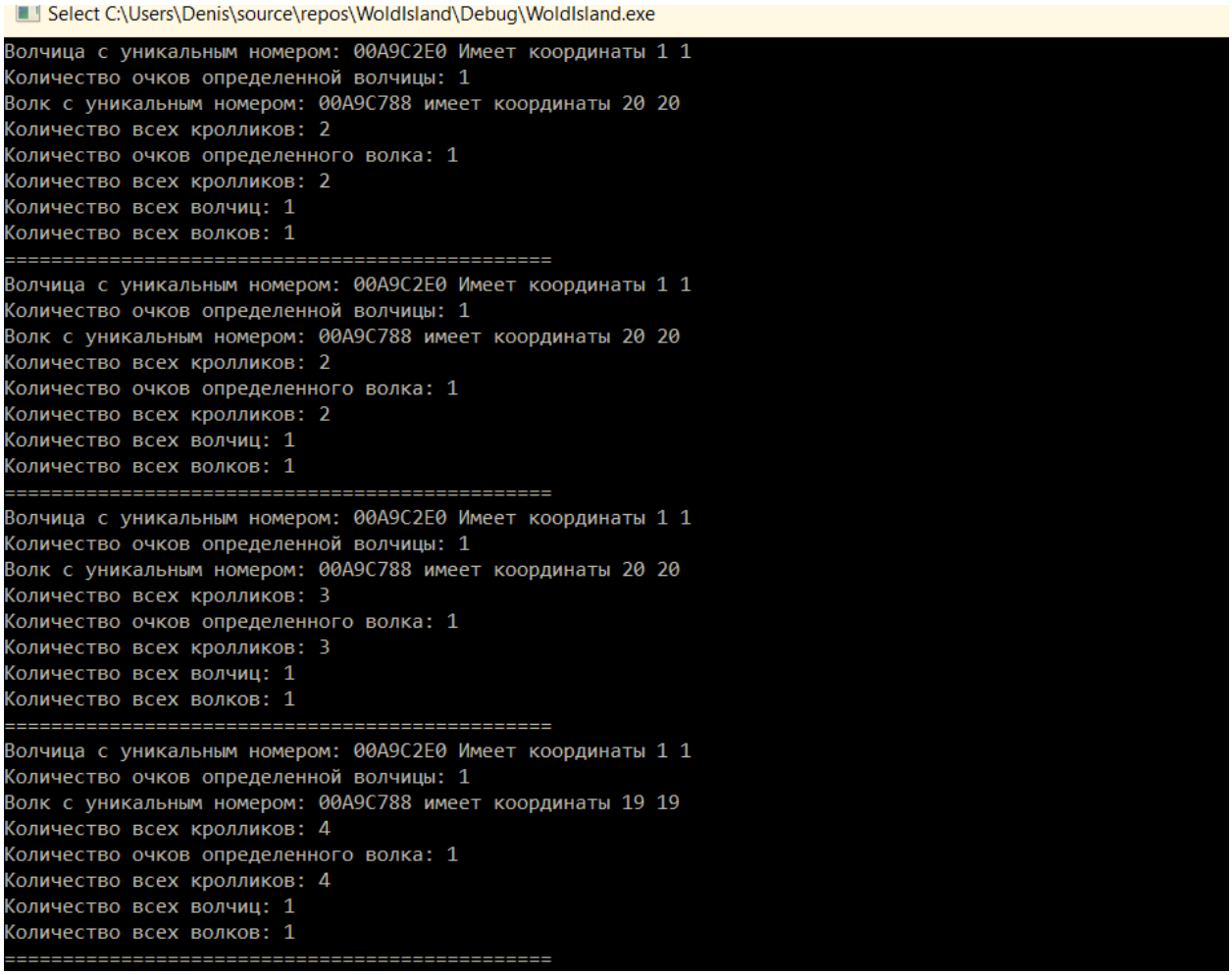


Рис. 2.3. Активная стадия симуляции

2.5.4. Окно по завершении игры

Игра завершается, когда все кролики будут съедены. По завершении игры выводится основная информация – количество волков и волчиц, которые остались живы.

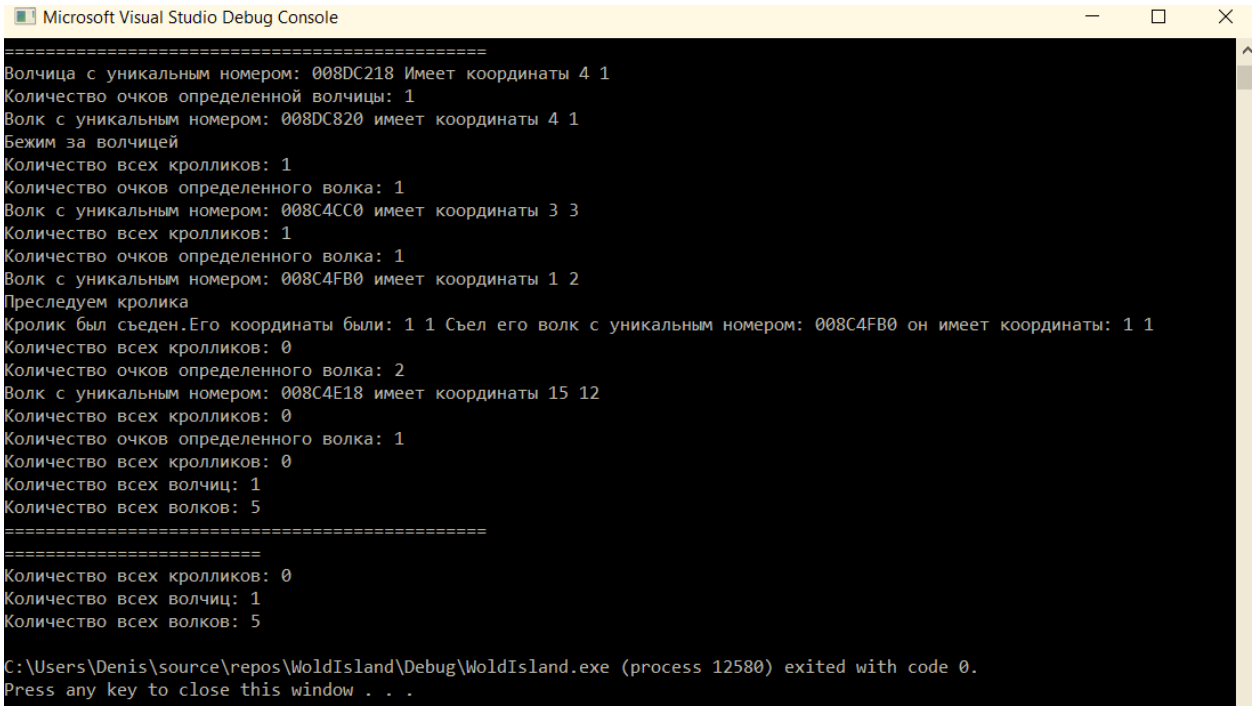


Рис. 2.4. Стадия завершения симуляции

2.6.Выводы к главе 2

Во второй главе:

1. Был рассмотрен интерфейс приложения.
2. Был рассмотрен полный цикл работы приложения от настройки модели до её завершения.
3. Посмотрели на работу приложения в действии.

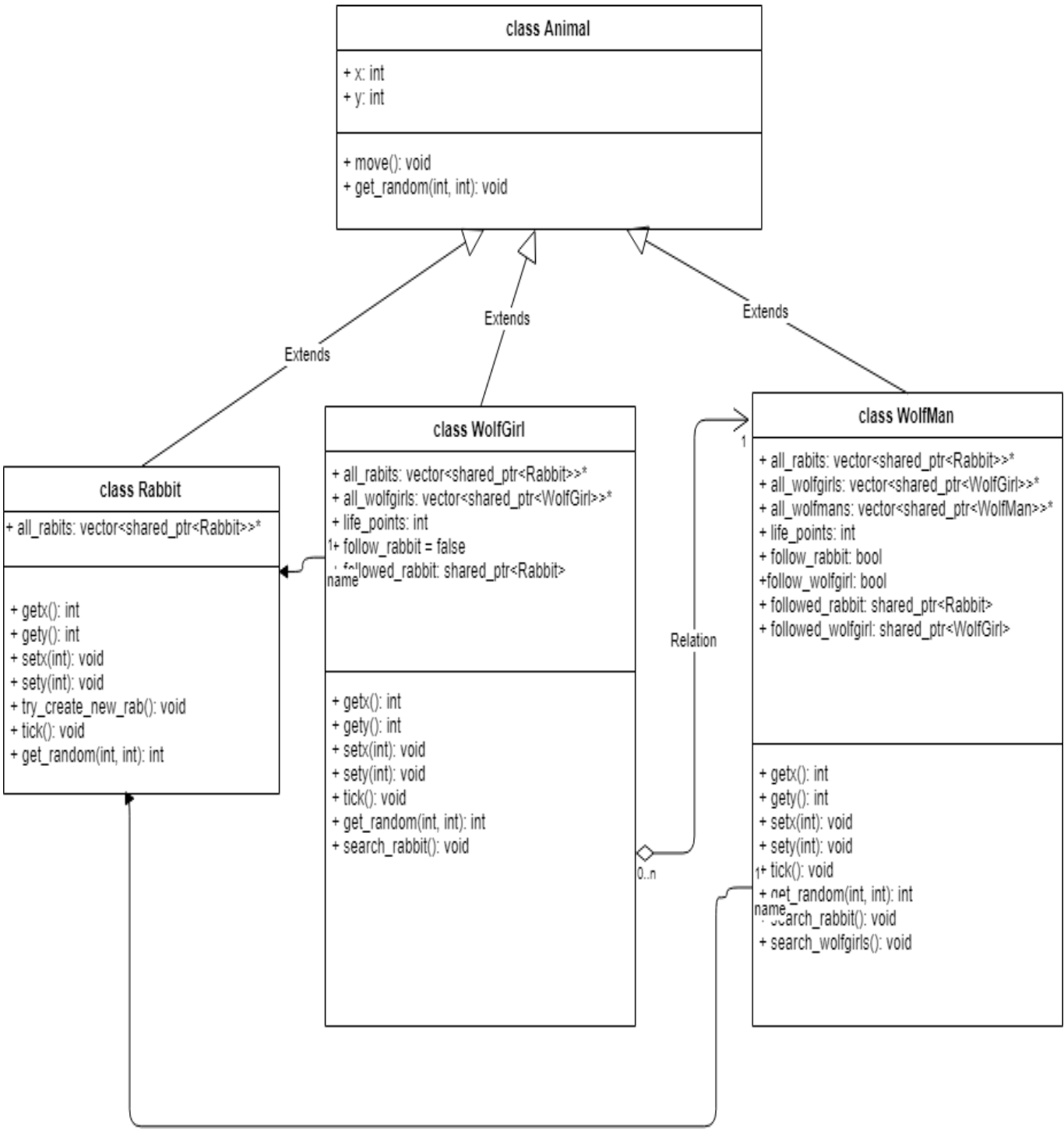
3. РЕАЛИЗАЦИЯ ПРИЛОЖЕНИЯ

3.1.Стадии и этапы разработки

Таблица 1 Содержание этапов работ

№ этапа	Содержание работ	Срок выполнения
1.	Разработка структуры входных и выходных данных	5-6 неделя
2.	Разработка алгоритма решения задачи	6-7 неделя
3.	Разработка структуры программы	8 неделя
4.	Разработка алгоритмов	9 неделя
5.	Написание текста программы	10-11 неделя
6.	Отладка программы	12 неделя
7.	Написание и оформление отчета	13-14 неделя

3.2.UML-диаграммы классов



3.3.Выводы к главе 3

В третьей главе были продемонстрированы UML-диаграммы классов, показаны основные этапы разработки приложения.

4. РУКОВОДСТВО ПО ИСПОЛЬЗОВАНИЮ ПРОГРАММЫ

4.1.Основные правила игры “Волчий Остров”. Описание модели.

Экологическая модель представляет из себя Волчий остров размером 20 × 20 заселен дикими кроликами, волками и волчицами. Имеется по несколько представителей каждого вида. Кролики довольно глупы: в каждый момент времени они с одинаковой вероятностью Y(параметр) передвигаются в один из восьми соседних квадратов (за исключением участков, ограниченных береговой линией) или просто сидят неподвижно. Каждый кролик с вероятностью X(параметр) превращается в двух кроликов. Каждая волчица передвигается случайным образом, пока в одном из соседних восьми квадратов не окажется кролик, за которым она охотится. Если волчица и кролик оказываются в одном квадрате, волчица съедает кролика и получает V(параметр) "очков". В противном случае она теряет W(параметр) "очков". Волки и волчицы с нулевым количеством очков умирают.
В начальный момент времени все волки и волчицы имеют P(параметр) очков. Волк ведет себя подобно волчице до тех пор, пока в соседних квадратах не исчезнут все кролики; тогда если волчица находится в одном из восьми близлежащих квадратов, волк гонится за ней. Если волк и волчица окажутся в одном квадрате и там нет кролика, которого нужно съесть, они производят потомство случайного пола.

4.2.Окно приложения

После запуска приложения вам будет предложено выбрать количество всех зверей, которые будут находиться в симуляции. Затем для каждого из них можно будет выбрать координаты, на которые вы хотите расставить каждого зверя. После этого начинается активная стадия симуляции. После завершения игры выведется соответствующее сообщение (“Симуляция завершена”), и выведется основная информация.

4.3.Выводы к главе 4

Были описаны правила модели и руководство использования приложения.

5. ТЕСТИРОВАНИЕ ПРИЛОЖЕНИЕ

В ходе разработке приложения были проведены различные тесты. Находились и исправлялись некоторые ошибки приложения. Вносились корректировки, улучшающие интерфейс.

Также в ходе разработки данной модели были выявлены некоторые закономерности. Данная экологическая модель не является устойчивой – происходят взрывы популяции, исход эксперимента может поменяться из-за малейших изменений системы, часто модель может идти бесконечно долго, неприходя к её завершению. Чтобы стабилизировать экологическую систему, как в реальной жизни, нам потребуется не одна сотня параметров, так как системы в реальной жизни крайне сложны.

5.1.Выводы к главе 5

Проведение итоговых тестовых запусков показало, что приложение:

- Работоспособно.
- Интуитивно понятно и удобно.
- Функционально.

В ходе выполнения курсовой работы было разработано соответствующее техническому заданию приложение «Волчий остров».

ЗАКЛЮЧЕНИЕ

По итогам разработки были выполнены следующие задачи:

- Был спроектирован интуитивно понятный интерфейс и логика приложения.
- Были реализованы необходимые для работы приложения алгоритмы и функции.
- Были проведены тесты для оценки работоспособности и корректности работы приложения.
- Исследована стабильность экологической системы.
- Проведены исследования зависимости исхода модели в зависимости от начальных параметров.
- Продемонстрированы навыки владения объектно-ориентированным программированием (использование принципов полиморфизма, наследования, инкапсуляции, абстрагирования)

Поставленная задача выполнена.

СПИСОК ИСТОЧНИКОВ

1. UML-Диаграммы классов

https://www.draw.io/

2. Майкл Доусон «Изучаем C++ через программирование игр»

[Книжное издание 2016 год]

3. Бьерн Страуструп «Язык программирования C++, краткий курс»

[Книжное издание 2017 год]

4. https://habr.com – Рассмотрел текущие имплементации на андройд
5. https://stackoverflow.com/ - Сайт, где программисты отвечают на вопросы друг другу.

ПРИЛОЖЕНИЕ

ИСХОДНЫЙ КОД ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

https://github.com/Nwke/Wolf-Island - Исходный код также можно посмотреть здесь.

WolfIsland.cpp

```
#include "pch.h"
#include <iostream>
#include <vector>
#include <memory>

#include <time.h>
#include <algorithm>
#include <cmath>

#include "Rabbit.h"
#include "WolfGirl.h"
#include "WolfMan.h"
#include "Constants.h"



using namespace std;

int main()
{
    setlocale(LC_ALL, "Russian");
    cout << "================================ Краткое руководство
================================\n\n" << endl;
    cout << "Приветствую вас в приложении экологической модели “Волчий
остров”.\nПожалуйста, ознакомьтесь с моделью для дальнейшей работы.\nСуть
экологической модели “Волчий остров”:\nУ вас есть поле 20x20, на котором
заселены дикие кролики, волки и волчицы.\nПравила данной модели приближены
к реальным.\nДанное приложение пытается смоделировать исход эксперимента
так, как если бы он проходил в условиях реальной жизни.\nВ дальнейшем вам
предложат ввести параметры, чтобы привести приложение в активное
состояние.\nВы также можете различным образом менять, комбинировать
параметры, чтобы \nпоэксперементировать, выявить какие-либо закономерности,
выдвинуть гипотезы." << endl;

    vector<shared_ptr<Rabbit>> all_rabbits;
    vector<shared_ptr<WolfGirl>> all_wolfgirl;
    vector<shared_ptr<WolfMan>> all_wolfman;


    int count_rabbits;
    int count_wolfgirl;
    int count_wolfman;

    cout << "\n\n" << endl;
    cout << "Введите количество кролликов на поле: ";
    cin >> count_rabbits;

    cout << "Введите количество волчиц на поле: " ;
    cin >> count_wolfgirl;

    cout << "Введите количество волков на поле: " ;
    cin >> count_wolfman;
```

```
    for (size_t i = 0; i < count_rabbits; i++) {
        int x = 0;
        int y = 0;
        cout << "==============" << endl;
        cout << "Кроллик по номеру " << i + 1  << endl;
        cout << "Введите координату X в диапазоне от  " << FIELD_X_D << "
До " << FIELD_X_U <<": ";

        cin >> x;

        cout << "Введите координату У в диапазоне от " << FIELD_Y_D << " До
" << FIELD_Y_U << ": ";
        cin >> y;

        shared_ptr<Rabbit> new_rabbit(new Rabbit(x, y, &all_rabbits));
        all_rabbits.push_back(new_rabbit);
    }

    for (size_t i = 0; i < count_wolfgirl; i++) {
        int x = 0;
        int y = 0;
        cout << "==============" << endl;
        cout << "Волчица по номеру " << i + 1  << endl;
        cout << "Введите координату X в диапазоне от " << FIELD_X_D << " До
" << FIELD_X_U << ": ";

        cin >> x;

        cout << "Введите координату У в диапазоне от " << FIELD_Y_D << " До
" << FIELD_Y_U << ": ";
        cin >> y;

        shared_ptr<WolfGirl> new_wolfgirl(new WolfGirl(x, y, &all_rabbits,
&all_wolfgirl));

        all_wolfgirl.push_back(new_wolfgirl);
    }


    for (size_t i = 0; i < count_wolfman; i++) {
        int x = 0;
        int y = 0;
        cout << "==============" << endl;
        cout << "Волк по номеру " << i + 1 << endl;
        cout << "Введите координату X в диапазоне от " << FIELD_X_D << " До
" << FIELD_X_U << ": ";

        cin >> x;

        cout << "Введите координату У в диапазоне от " << FIELD_Y_D << " До
" << FIELD_Y_U << ": ";
        cin >> y;

        shared_ptr<WolfMan> new_wolfman(new WolfMan(x, y, &all_rabbits,
&all_wolfman, &all_wolfgirl));

        all_wolfman.push_back(new_wolfman);
    }

    cout << "\n\n";
    cout << "=========== НАЧАЛО СИМУЛЯЦИИ ===========" << endl;
```

```
    while (all_rabbits.size() != 0) {
        vector<shared_ptr<Rabbit>> rabbits_copy(all_rabbits);
        vector<shared_ptr<WolfGirl>> wolfgirl_copy(all_wolfgirl);
        vector<shared_ptr<WolfMan>> wolfman_copy(all_wolfman);



        for (auto rabb : rabbits_copy) {
            (*rabb).tick();
        }



        for (auto wolf_girl : wolfgirl_copy) {
            (*wolf_girl).tick();
            cout << "Количество очков определенной волчицы: " <<
(*wolf_girl).life_points << endl;

        }

        for (auto wolf_man : wolfman_copy) {
            (*wolf_man).tick();
            cout << "Количество всех кролликов: " << all_rabbits.size() <<
endl;
            cout << "Количество очков определенного волка: " <<
(*wolf_man).life_points << endl;
        }

        cout << "Количество всех кролликов: " << all_rabbits.size() <<
endl;
        cout << "Количество всех волчиц: " << all_wolfgirl.size() << endl;
        cout << "Количество всех волков: " << all_wolfman.size() << endl;

        cout << "=============================================" << endl;


    }

    cout << "=========== КОНЕЦ СИМУЛЯЦИИ ===========" << endl;
    cout << endl;
    cout << endl;
    cout << "=========== ИТОГИ ===========" << endl;

    cout << "Количество всех кролликов: " << all_rabbits.size() << endl;
    cout << "Количество всех волчиц: " << all_wolfgirl.size() << endl;
    cout << "Количество всех волков: " << all_wolfman.size() << endl;




    all_rabbits.clear();
    cout << "Игра окончена" << endl;
    system("pause");
}
```

Animal.h

```
#pragma once
class Animal
{
public:
```

```
    Animal();
    ~Animal();
    void move();
    int get_random(int, int);
    int x;
    int y;
};
```

Animal.cpp

```
#include "pch.h"
#include "Animal.h"
#include <iostream>
#include "Constants.h"
#include <stdlib.h>
#include <time.h>
#include <algorithm>


Animal::Animal()
{
}


Animal::~Animal()
{
}

void Animal::move()
{
    int code_movement = get_random(1, 9);

    if ((code_movement == 1) && (y != FIELD_Y_D)) {
        y -= 1;
    }
    if ((code_movement == 2) && (x != FIELD_X_D)) {
        x -= 1;
    }
    if ((code_movement == 3) && (y != FIELD_Y_U)) {
        y += 1;
    }
    if ((code_movement == 4) && (y != FIELD_X_U)) {
        x += 1;
    }
    if ((code_movement == 5) && (x != FIELD_X_D) && (y != FIELD_Y_U)) {
        x -= 1;
        y += 1;
    }
    if ((code_movement == 6) && (x != FIELD_X_D) && (y != FIELD_Y_D)) {
        x -= 1;
        y -= 1;
    }
    if ((code_movement == 7) && (x != FIELD_X_U) && (y != FIELD_Y_U)) {
        x += 1;
        y += 1;
    }
    if ((code_movement == 8) && (x != FIELD_X_U) && (y != FIELD_Y_D)) {
        x += 1;
        y -= 1;
    }
    if (code_movement == 9) {
```

```
    }
}

int Animal::get_random(int from, int to)
{
    int random;
    random = rand() % to + from;
    return random;
}
```

Constants.h

```
#pragma once
const int FIELD_X_D = 1;
const int FIELD_X_U = 20;

const int FIELD_Y_D = 1;
const int FIELD_Y_U = 20;
```

Rabbit.h

```
#pragma once
#include "pch.h"
#include "Animal.h"


using namespace std;

class Rabbit: public Animal
{
public:
    Rabbit(int, int, vector<shared_ptr<Rabbit>>*);
    ~Rabbit();

    int getx();

    int gety();

    void setx(int);

    void sety(int);

    int get_random(int, int);

    void try_create_new_rab();
    void tick();


    vector<shared_ptr<Rabbit>> *all_rabits;


};
```

Rabbit.cpp

```
#include "pch.h"
#include <iostream>
#include <vector>
#include <memory>
#include <stdlib.h>
#include <time.h>
#include <algorithm>
#include <cmath>

#include "Rabbit.h"
#include "Constants.h"



void Rabbit::try_create_new_rab() {
    int code = get_random(1, 100);

    if (code == 2) {
        shared_ptr<Rabbit> p1(new Rabbit(this->x, this->y, this-
>all_rabits));
        (*all_rabits).push_back(p1);
    }


}

void Rabbit::tick() {
    try_create_new_rab();
    move();
}


int Rabbit::getx() {
    return x;
}

int Rabbit::gety() {
    return y;
}

void Rabbit::setx(int value) {
    x = value;
}

void Rabbit::sety(int value) {
    y = value;
}

int Rabbit::get_random(int from, int to) {
    int random;
    random = rand() % to + from;
    return random;
}

Rabbit::Rabbit(int x, int y, vector<shared_ptr<Rabbit>> *all_rabits) {
    this->x = x;
    this->y = y;
    this->all_rabits = all_rabits;
}
Rabbit::~Rabbit() {
}
```

WolfGirl.h

```
#pragma once
#include "pch.h"
#include "Animal.h"


using namespace std;



class WolfGirl: public Animal
{
public:
    WolfGirl(int, int, vector<shared_ptr<Rabbit>> *,
vector<shared_ptr<WolfGirl>> *);
    ~WolfGirl();

    vector<shared_ptr<Rabbit>> *all_rabbits;
    vector<shared_ptr<WolfGirl>> *all_wolfgirls;


    double life_points = 1;

    bool follow_rabbit = false;
    shared_ptr<Rabbit> followed_rabbit;


    int getx();

    int gety();

    void setx(int);

    void sety(int);

    int get_random(int, int);


    void tick();

    void search_rabbit();

};
```

WolfGirl.cpp

```
#include "pch.h"
#include <iostream>
#include <vector>
#include <memory>
#include <stdlib.h>
#include <time.h>
#include <algorithm>
#include <cmath>

#include "Rabbit.h"
#include "WolfGirl.h"
#include "Animal.h"

#include "Constants.h"
```

```
WolfGirl::WolfGirl(int x, int y, vector<shared_ptr<Rabbit>> *all_rabbits,
vector<shared_ptr<WolfGirl>> *all_wolfgirls) {
    this->x = x;
    this->y = y;
    this->all_rabbits = all_rabbits;
    this->all_wolfgirls = all_wolfgirls;


}


WolfGirl::~WolfGirl() {

}


int WolfGirl::getx() {
    return x;
}

int WolfGirl::gety() {
    return y;
}

void WolfGirl::setx(int value) {
    this->x = value;
}

void WolfGirl::sety(int value) {
    y = value;
}

int WolfGirl::get_random(int from, int to) {
    int random;
    random = rand() % to + from;
    return random;
}

void WolfGirl::search_rabbit() {
    if (!follow_rabbit) {
        for (auto rab : *this->all_rabbits) {


            if ((!follow_rabbit) && (sqrt(pow(x - (*rab).getx(), 2) + pow(y -
(*rab).gety(), 2) < 2))) {


                follow_rabbit = true;
                followed_rabbit = rab;
                break;



                cout << "Преследуем кролика" << endl;
            }
        }
    }
}

void WolfGirl::tick() {
```

```
    cout << "Волчица с уникальным номером: " << this << " Имеет координаты "
<< this->x << " " << this->y << endl;

    if (follow_rabbit) {

        if (find((*all_rabbits).begin(), (*all_rabbits).end(),
followed_rabbit) != (*all_rabbits).end()) {
            follow_rabbit = false;
            followed_rabbit = 0;
        }
    }

    search_rabbit();

    if (follow_rabbit) {

        if (x < (*followed_rabbit).getx()) {
            x += 1;
        }
        else if (x > (*followed_rabbit).getx()) {
            x -= 1;
        }


        if (y < (*followed_rabbit).gety()) {
            y += 1;
        }
        else if (y > (*followed_rabbit).gety()) {
            y -= 1;
        }

        if ((x == (*followed_rabbit).getx()) && (y ==
(*followed_rabbit).gety())) {
            cout << "Кроллик был съеден. Его координаты были: " << x << " "
<< y << " Уникальный номер волчицы,съевшая кролика: " << this << " Она имеет
координаты " << x << " " << y << endl;

            if (find((*all_rabbits).begin(), (*all_rabbits).end(),
followed_rabbit) != (*all_rabbits).end()) {
                (*all_rabbits).erase(remove((*all_rabbits).begin(),
(*all_rabbits).end(), followed_rabbit));
            }

            life_points += 1;
        }
        else {
            life_points -= 0.1;

            if (life_points == 0) {
                cout << "Волчица умерла. Её уникальный номер: " << this << "
Она находилась в клетке " << x << " " << y << endl;
                shared_ptr<WolfGirl> tmp(this);
                (*all_wolfgirls).erase(remove((*all_wolfgirls).begin(),
(*all_wolfgirls).end(), tmp));

            }
        }
    }
    else {
        move();
    }
}
```

WolfMan.h

```
#pragma once
#include "pch.h"
#include "Animal.h"


class WolfMan : public Animal
{
public:
    WolfMan(int x, int y, vector<shared_ptr<Rabbit>> *all_rabbits,
vector<shared_ptr<WolfMan>> *all_wolfman, vector<shared_ptr<WolfGirl>>
*all_wolfgirls);
    ~WolfMan();

    vector<shared_ptr<Rabbit>> *all_rabbits;
    vector<shared_ptr<WolfMan>> *all_wolfmans;
    vector<shared_ptr<WolfGirl>> *all_wolfgirls;


    double life_points = 1;

    bool follow_rabbit = false;
    bool follow_wolfgirl = false;


    shared_ptr<Rabbit> followed_rabbit;
    shared_ptr<WolfGirl> followed_wolfgirl;


    int getx();

    int gety();

    void setx(int);

    void sety(int);

    int get_random(int, int);

    void tick();

    void search_rabbit();
    void search_wolfgirls();

};
```

WolfMan.cpp

```
#include "pch.h"
#include <iostream>
#include <vector>
#include <memory>
#include <stdlib.h>
#include <time.h>
#include <algorithm>
#include <cmath>

#include "Rabbit.h"
#include "WolfGirl.h"
#include "WolfMan.h"
```

```
#include "Constants.h"



WolfMan::WolfMan(int x, int y, vector<shared_ptr<Rabbit>> *all_rabbits,
vector<shared_ptr<WolfMan>> *all_wolfmans, vector<shared_ptr<WolfGirl>>
*all_wolfgirls) {
    this->x = x;
    this->y = y;
    this->all_rabbits = all_rabbits;
    this->all_wolfmans = all_wolfmans;
    this->all_wolfgirls = all_wolfgirls;


}
WolfMan::~WolfMan() {

}

int WolfMan::getx() {
    return x;
}

int WolfMan::gety() {
    return y;
}

void WolfMan::setx(int value) {
    x = value;
}

void WolfMan::sety(int value) {
    y = value;
}

int WolfMan::get_random(int from, int to) {
    int random;
    random = rand() % to + from;
    return random;
}




void WolfMan::tick() {
    cout << "Волк с уникальным номером: " << this << " имеет координаты " <<
x << " " << y << endl;

    if (this->follow_rabbit) {

        if (find((*all_rabbits).begin(), (*all_rabbits).end(),
followed_rabbit) != (*all_rabbits).end()) {
            follow_rabbit = false;
            followed_rabbit = 0;
        }
    }

    search_rabbit();

    if (follow_rabbit) {
```

```
        if (x < (*followed_rabbit).getx()) {
            x += 1;
        }
        else if (x > (*followed_rabbit).getx()) {
            x -= 1;
        }


        if (y < (*followed_rabbit).gety()) {
            y += 1;
        }
        else if (y > (*followed_rabbit).gety()) {
            y -= 1;
        }

        if ((x == (*followed_rabbit).getx()) && (y ==
(*followed_rabbit).gety())) {
            cout << "Кролик был съеден.Его координаты были: " << x << " " <<
y << " Съел его волк с уникальным номером: " << this << " он имеет
координаты: " << x << " " << y << endl;


            if (find((*all_rabbits).begin(), (*all_rabbits).end(),
followed_rabbit) != (*all_rabbits).end()) {
                (*all_rabbits).erase(remove((*all_rabbits).begin(),
(*all_rabbits).end(), followed_rabbit));
            }

            life_points += 1;
        }
        else {
            life_points -= 0.1;

            if (life_points == 0) {
                cout << "Волк умер. Его униклаьный номер: " << this << " Он
имел координаты " << x << " " << y << endl;
                shared_ptr<WolfMan> tmp(this);
                (*all_wolfmans).erase(remove((*all_wolfmans).begin(),
(*all_wolfmans).end(), tmp));

            }
        }
    }
    else if (!follow_rabbit) {
        search_wolfgirls();
    }


    if (follow_wolfgirl) {
        cout << "Бежим за волчицей" << endl;

        if (x < (*followed_wolfgirl).getx()) {
            x += 1;
        }
        else if (x > (*followed_wolfgirl).getx()) {
            x -= 1;
        }


        if (y < (*followed_wolfgirl).gety()) {
            y += 1;
        }
```

```
        else if (y > (*followed_wolfgirl).gety()) {
            y -= 1;
        }



        if ((x == (*followed_wolfgirl).getx()) && (y ==
(*followed_wolfgirl).gety())) {
            int sex = get_random(0, 1);

            // coords of next childs are a little random

            if (sex == 0) {
                shared_ptr<WolfMan> tmp(new WolfMan(((x + 10) % 20) + 1, ((y
+ 10) % 20) + 1, all_rabbits, all_wolfmans, all_wolfgirls));

                (*all_wolfmans).push_back(tmp);
            }
            else {
                shared_ptr<WolfGirl> tmp(new WolfGirl(((x + 10) % 20) + 1,
((y + 10) % 20) + 1, all_rabbits, all_wolfgirls));
                (*all_wolfgirls).push_back(tmp);
            }

        }
    }
    else {
        this->move();
    }
}

void WolfMan::search_wolfgirls() {

    if (!follow_wolfgirl) {
        for (auto wolfgirl : *all_wolfgirls) {


            if ((!follow_wolfgirl) && (sqrt(pow(x - (*wolfgirl).getx(), 2) +
pow(y - (*wolfgirl).gety(), 2) < 2))) {
                cout << "Волк с уникальным номером: " << this << " "<< "начал
преследовать волчицу" << endl;
                follow_wolfgirl = true;
                followed_wolfgirl = wolfgirl;
                break;
            }
        }
    }

}

void WolfMan::search_rabbit() {
    if (!follow_rabbit) {
        for (auto rab : *all_rabbits) {
            if ((!follow_rabbit) && (sqrt(pow(x - (*rab).getx(), 2) + pow(y -
(*rab).gety(), 2) < 2))) {
                follow_rabbit = true;
                followed_rabbit = rab;

                follow_wolfgirl = false;
                followed_wolfgirl = 0;

                cout << "Преследуем кролика" << endl;
            }
```

```
            }
        }

    }
```