Пермский филиал федерального государственного автономного образовательного учреждения высшего образования

«Национальный исследовательский университет

«Высшая школа экономики»

Факультет экономики, менеджмента и бизнес-информатики

Образовательная программа бакалавриата «Бизнес-информатика»

**ОТЧЕТ**

**по учебной практике**

Выполнил студент БИ-17-2

Хатипова Симона Ирековна

(подпись)

Проверил:

Руководитель практики

Преподаватель кафедры   
информационных технологий в бизнесе

Гордеева О.И.

\_\_\_\_\_\_\_\_\_\_\_

(оценка) (подпись)

(дата)

Пермь, 2018

Оглавление

[Введение 3](#_Toc516756930)

[Глава 1. Анализ существующих аналогов игры «Волчий остров» 5](#_Toc516756931)

[1.1. Анализ игры «Siberian Forest» 5](#_Toc516756932)

[1.2. Анализ игры «Жизнь» 8](#_Toc516756933)

[1.3. Анализ игры имитационной модели «хищник-жертва» 11](#_Toc516756934)

[1.4. Сравнительная таблица по всем проанализированным играм 12](#_Toc516756935)

[1.5. Вывод по главе 13](#_Toc516756936)

[Глава 2. Проектирование алгоритма 14](#_Toc516756937)

[2.1. Описание Windows Forms 14](#_Toc516756938)

[2.1.1. Main.cs 14](#_Toc516756939)

[2.1.2. Rules.cs 16](#_Toc516756940)

[2.1.3. Instruction.cs 17](#_Toc516756941)

[2.1.4. End.cs 17](#_Toc516756942)

[2.2. Классы, используемые при разработке программы 18](#_Toc516756943)

[2.3. Блок-схемы и краткое описание основных алгоритмов 19](#_Toc516756944)

[2.3.1. Добавление персонажа на поле 19](#_Toc516756945)

[2.3.2. Действия персонажей 20](#_Toc516756946)

[2.3.3. Передвижение 22](#_Toc516756947)

[2.3.4. Размножение (рассмотрим у зайцев) 23](#_Toc516756948)

[2.3.4. Охота 24](#_Toc516756949)

[2.4. Вывод по главе 25](#_Toc516756950)

[Глава 3. Реализация спроектированного алгоритма 26](#_Toc516756951)

[3.1. Описание глобальных переменных 26](#_Toc516756952)

[В программе используются следующие глобальные переменные: 26](#_Toc516756953)

[3.2. Тестирование разработанного алгоритма 27](#_Toc516756954)

[3.3. Руководство пользователя 29](#_Toc516756955)

[3.4. Вывод по главе 30](#_Toc516756956)

[Заключение 31](#_Toc516756957)

[Библиографический список 33](#_Toc516756958)

[Приложение 1. Тесты 34](#_Toc516756959)

[Приложение 2. Листинг программы (с комментариями) 38](#_Toc516756960)

# Введение

В современном мире достаточно много высших учебных заведений, где учат правильно управлять разными вещами и отраслями жизни человека: цепями поставок, коллективом, системами воздушного транспорта, но, наверное, не каждый хотел бы попробовать себя в роли управленца жизнью на Земле, возложить на свои плечи тяжелый груз ответственности за последствия своих действий. Это, безусловно, тяжело, потому что, скорее всего, влечет необратимые последствия. В игре-симуляторе под названием «Волчий остров» или «Жизнь» абсолютно каждый может попробовать свои силы в управлении целой экосистемой, ведь если не получится настроить идиллию и верный баланс сил с первой попытки, вполне можно начать «заселять мир» с чистого листа.

Игра, рассматривающая развитие собственной экосистемы, построенной на модели хищник-жертва, не реализуется вручную (т.к. достаточно трудно проследить за передвижениями всех героев одновременно), но отдалено напоминает шахматы. Возможность понаблюдать за последствиями своей расстановки персонажей и мирным или не очень протеканием жизни в игре – это именно то, что может привлечь как детей, так и взрослых. Универсальность игры заключается в том, что она не требует никаких специальных познаний от пользователя – достаточно лишь расположить животных на поле по своему усмотрению и наблюдать.

Подобная игра не имеет множество вариаций и не сильно распространена, наверное потому, что программисты считают алгоритмы простыми и неинтересными, но можно предположить, если расширить список характеристик животных и добавить в симулятор обстоятельства из реальный жизни, такие как природные условия, голод, засуха и т.п., можно создать очень увлекательную и «живую» игру, похожую на реальный мир.

*Целью данной учебной практики* является создание игры «Волчий остров».

*Задачи учебной практики:*

* изучение предметной области: анализ уже реализованных игр о развитии жизни, выбранных из предложений в интернете;
* разработка своего алгоритма для игры «Волчий остров»;
* реализация разработанного алгоритма в виде приложения на языке С#, используя навыки объектно-ориентированного программирования;
* тестирование реализованного алгоритма с использованием различных критериев тестирования.

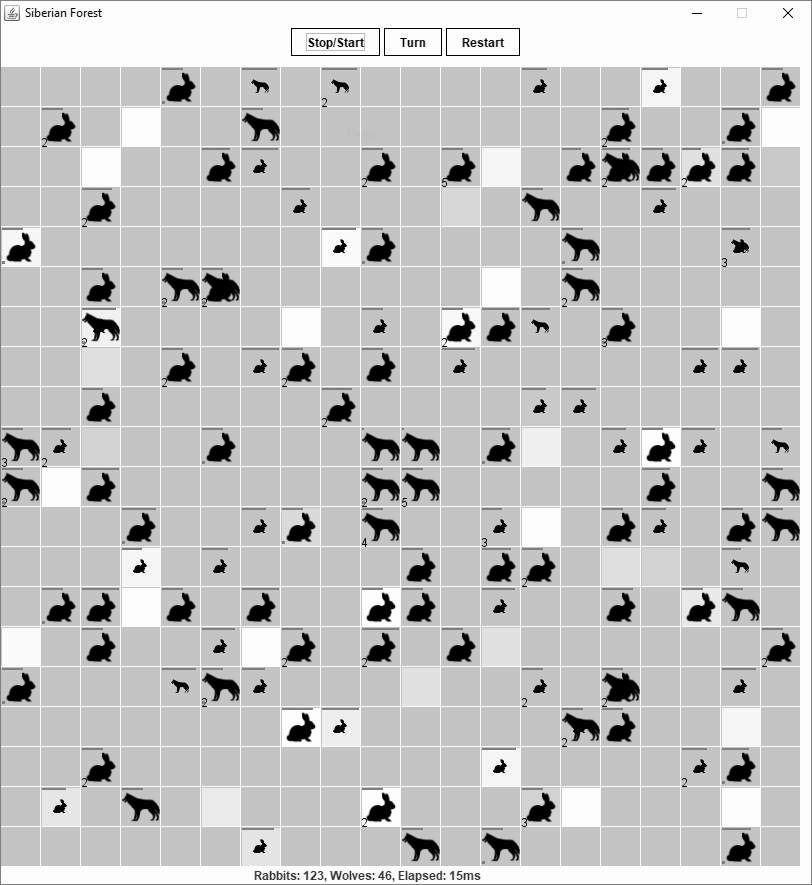
# Глава 1. Анализ существующих аналогов игры «Волчий остров»

Для реализации первой задачи данной учебной практики необходимо проанализировать уже созданные интерпретации алгоритма игры «Волчий остров». Игры были выбраны среди доступных для пользователя бесплатно в интернете.

## 1.1. Анализ игры «Siberian Forest»

Основные сведения об игре: создатель – Роман Прохоров, последнее обновление вышло 10 месяцев назад, игра не загружена в App Store или Play Market.[[1]](#footnote-1)

Интерфейс игры представлен на рисунке 1.1. Нет необходимости создавать какой-либо виртуальный счет игрока с некоторым количеством виртуальных монет, потому что для роли создателя и управленца ресурсы, придуманные людьми, не имеют никакого значения.



***Рисунок 1.1. Игровое поле***

Легенда поля:

Stop/Start — Запустить мир.

Turn — Остановить мир.

Restart — Пересоздать мир.  
Зеленые клетки — Клетки с травой. Чем зеленее, тем больше травы.  
Маленькие зайцы и волки — щенки.  
Большие зайцы и волки — взрослые особи.  
Красные и синие полоски на пиктограммой зверей — текущая сытость. Красные — самцы, синие — самки.  
Число в левом нижнем углу каждой клетки — количество существ на данной клетке.  
Внизу общее количество зайцев и волков, а также время, занявшее обработку последнего ход.

Волчий остров размером N \* N заселен дикими зайцами и волками. Имеется по несколько представителей каждого вида. Зайцы в каждый момент времени с одинаковой вероятностью 1/9 передвигаются в один из восьми соседних квадратов (за исключением участков, ограниченных береговой линией) или сидят неподвижно. Каждый заяц с вероятностью p(br) превращается в двух зайцев. Каждый волк с вероятностью p(bw) превращается в двух волков. Каждый волк передвигается случайным образом (как и заяц), пока в одном из соседних восьми квадратов не окажется заяц, за которым он охотится. Если волк и заяц оказываются в одном квадрате, волк съедает зайца и получает h «очков». В противном случае он теряет d «очков». Волки с нулевым количеством очков умирают. В начальный момент времени все волки имеют H0 очков. Параметры задачи могли незначительно изменяться, часто имели место модификации: например, «честное» половое размножение хотя бы волков, но суть оставалась все той же.

Других интерфейсов в игре не предоставлено, уровней в игре тоже нет. Изначально пользователю доступно пустое поле, на котором он сам вправе расположить персонажей. Пользователю необходимо расставить по своему усмотрению волков и кроликов, используя для этого левую и правую кнопки мыши.

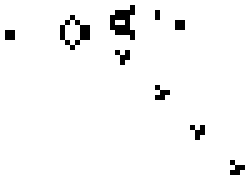
Достоинства и недостатки представим в виде таблицы (таблица 1.1):

***Таблица 1.1. Анализ «Siberian Forest»***

|  |  |
| --- | --- |
| **Плюсы** | **Минусы (неудобства)** |
| * Возможность пользователю добавлять участников игры. * Цветной интерфейс пользовательский. * Наглядное отображение сытости животных. * Показывает численость. * Образуется ареал обитания. * Возможность выполнения операции 1 ход. * Наличие на игровом поле детенышей обоих видов животных. * Понятное изображение персонажей. * Показывает время, прошедшее с завершения предыдущего хода (в милисекундах!). | * Невозможность пользователю настраивать масштаб поля. * Непонятно когда заканчивается игра (возможно, это большое достижение, потому,что найден баланс сил). * Непонятно образовывается ареал обитания. * Не полусается проследить за персонажем в отдельности. * Непонятно развитие событий в игре(где кто родился или кого съели). * Непонятно, где копятся очки каждого персонажа. * Невозможность настройки интерфейса. |

## 1.2. Анализ игры «Жизнь»

Игра «Жизнь» или же «Конуэ» ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) Conway's Game of Life) — [клеточный автомат](https://ru.wikipedia.org/wiki/%D0%9A%D0%BB%D0%B5%D1%82%D0%BE%D1%87%D0%BD%D1%8B%D0%B9_%D0%B0%D0%B2%D1%82%D0%BE%D0%BC%D0%B0%D1%82), придуманный [английским](https://ru.wikipedia.org/wiki/%D0%92%D0%B5%D0%BB%D0%B8%D0%BA%D0%BE%D0%B1%D1%80%D0%B8%D1%82%D0%B0%D0%BD%D0%B8%D1%8F) [математиком](https://ru.wikipedia.org/wiki/%D0%9C%D0%B0%D1%82%D0%B5%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0) [Джоном Конвеем](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BD%D0%B2%D0%B5%D0%B9,_%D0%94%D0%B6%D0%BE%D0%BD_%D0%A5%D0%BE%D1%80%D1%82%D0%BE%D0%BD) в [1970 году](https://ru.wikipedia.org/wiki/1970_%D0%B3%D0%BE%D0%B4).[[2]](#footnote-2)

[](https://commons.wikimedia.org/wiki/File:Gospers_glider_gun.gif?uselang=ru)

***Рисунок 1.2. Игровое поле***

Правила:

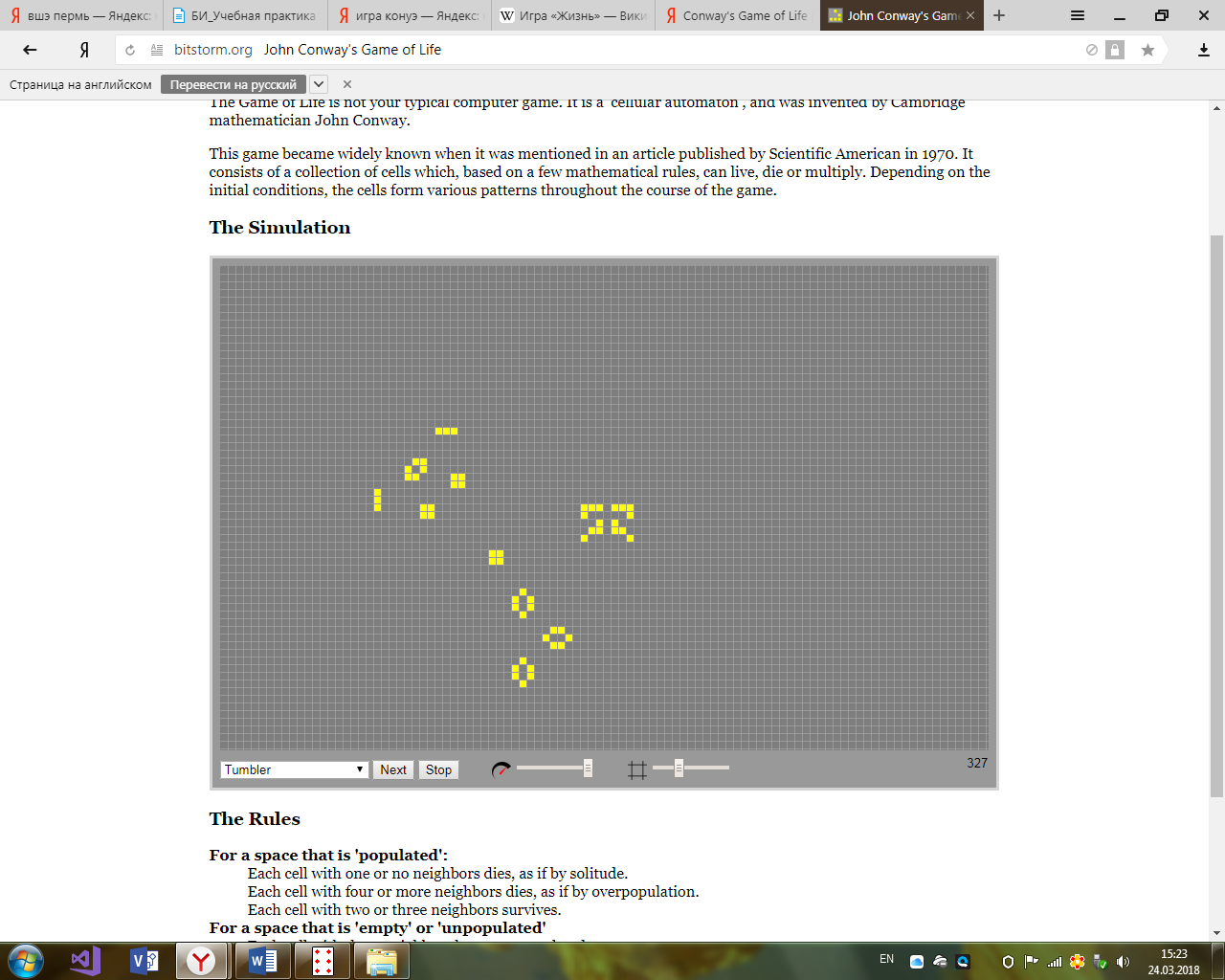
* Место действия этой игры — «вселенная» — это размеченная на клетки поверхность или плоскость — безграничная, ограниченная, или замкнутая (в пределе — бесконечная плоскость).
* Каждая клетка на этой поверхности может находиться в двух состояниях: быть «живой» (заполненной) или быть «мёртвой» (пустой). Клетка [имеет восемь соседей](https://ru.wikipedia.org/wiki/%D0%9E%D0%BA%D1%80%D0%B5%D1%81%D1%82%D0%BD%D0%BE%D1%81%D1%82%D1%8C_%D0%9C%D1%83%D1%80%D0%B0), окружающих её.

Достоинства и недостатки предоставим в виде таблицы (таблица 1.2):

***Таблица 1.2. Анализ «Жизни»***

|  |  |
| --- | --- |
| **Плюсы** | **Минусы (неудобства)** |
| * Возможность пользователю добавлять участников игры. * Цветной интерфейс пользовательский. * Показывает численость. * Возможность выполнения операции 1 ход. * Показывает время, прошедшее с завершения предыдущего хода (в милисекундах!). * Понятно, когда игра закончится. * Легкие правила, добавление клеток. * Можно проследить за развитием тел. | * Невозможность пользователю настраивать масштаб поля. * Быстро исчезают клетки. * Не получается проследить за персонажем в отдельности. * Непонятно развитие событий в игре (где кто родился или кого съели). * Невозможность настройки интерфейса. * Нет выбора персонажа. * Не показывает численность клеток на поле. * мало встроенных фигур из клеток. * происходит зацикливание в развитие и игра просто будто зависает(на самом деле просто кончаются пути развития для клетки и она становится будто статичной). * нет разных видов клеток, все лишь «клетки». |

* Распределение живых клеток в начале игры называется первым поколением. Каждое следующее поколение рассчитывается на основе предыдущего по таким правилам:
* в пустой (мёртвой) клетке, рядом с которой ровно три живые клетки, зарождается жизнь;
* если у живой клетки есть две или три живые соседки, то эта клетка продолжает жить; в противном случае, если соседей меньше двух или больше трёх, клетка умирает («от одиночества» или «от перенаселённости»).
* Игра прекращается, если на поле не останется ни одной «живой» клетки;
* конфигурация на очередном шаге в точности (без сдвигов и поворотов) повторит себя же на одном из более ранних шагов (складывается периодическая конфигурация);
* при очередном шаге ни одна из клеток не меняет своего состояния (складывается стабильная конфигурация; предыдущее правило, вырожденное до одного шага назад).



***Рисунок 1.3. Игровое поле Конуэ***

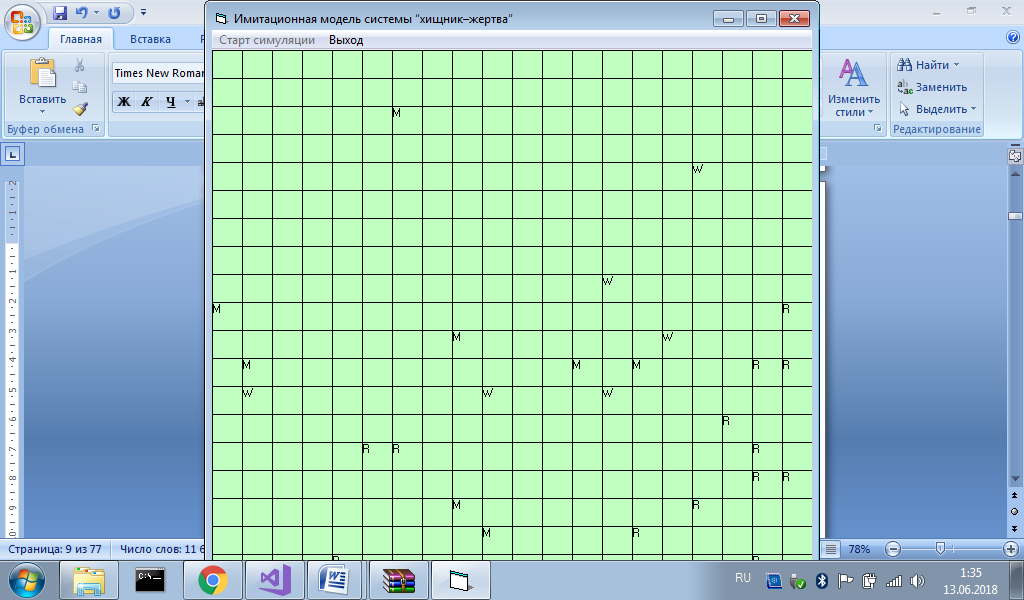
Эти простые правила приводят к огромному разнообразию форм, которые могут возникнуть в игре.

Игрок не принимает прямого участия в игре, а лишь расставляет или генерирует начальную конфигурацию «живых» клеток, которые затем взаимодействуют согласно правилам уже без его участия (он является наблюдателем).

Главное игровое поле игры представлено на рисунке 1.3. Изначально пользователю доступны фигуры из клеток, а впоследствии мы можем наблюдать за их развитием в нечто другое или же зацикливание.

## 1.3. Анализ игры имитационной модели «хищник-жертва»

Главное (единственное) поле представлено на рисунке 1.4. Пользователю не доступен выбор расстановки и изменение персонажей.[[3]](#footnote-3)



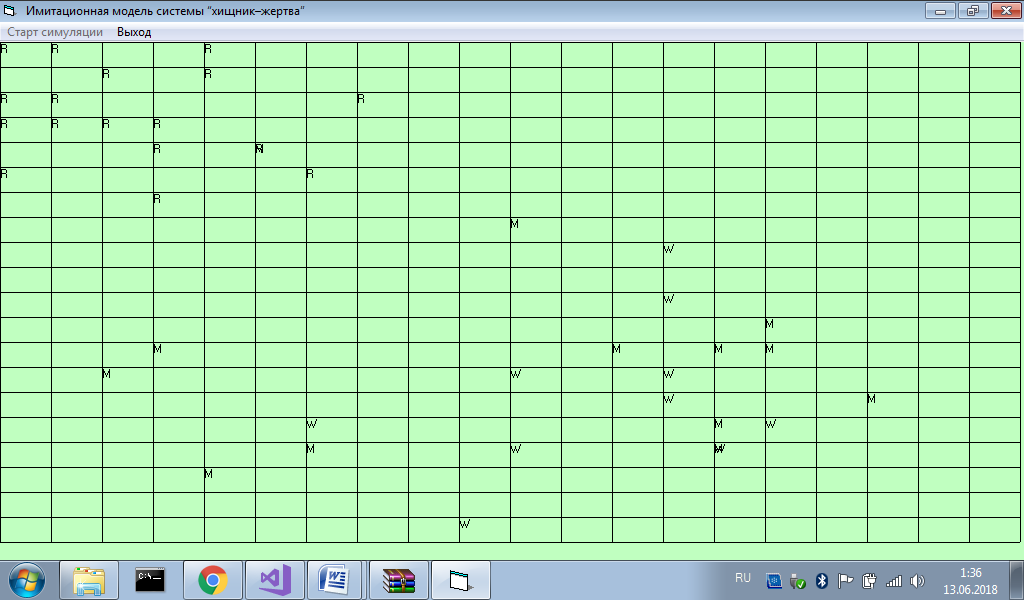
***Рисунок 1.4. Главное поле (исходный размер)***

Достоинства и недостатки предоставим в виде таблицы (таблица 1.3)

Пользователь может запустить процесс симуляции кнопкой на панели меню «Старт симуляции» и выйти из игры, нажав на кнопку «Выход». Больше пользователь не может никак повлиять на систему. Поле можно расширить на весь экран (рисунок 1.5)

***Таблица 1.3. Анализ игры-симуляции***

|  |  |
| --- | --- |
| **Плюсы** | **Минусы (неудобства)** |
| * Возможность пользователю добавлять участников игры * Цветной интерфейс пользовательский * Показывает численость * Возможность выполнения операции 1 ход * Показывает время, прошедшее с завершения предыдущего хода (в милисекундах!) * Понятно, когда игра закончится * Легкие правила, добавление клеток * Можно проследить за развитием тел | * Невозможность пользователю настраивать масштаб поля * Быстро исчезают клетки * Не получается проследить за персонажем в отдельности * Непонятно развитие событий в игре (где кто родился или кого съели) * Невозможность настройки интерфейса * Нет выбора персонажа * Не показывает численность клеток на поле * мало встроенных фигур из клеток * происходит зацикливание в развитие и игра просто будто зависает(на самом деле просто кончаются пути развития дл яклетки и она становится будто статичной) * нет разных видов клеток, все лишь «клетки». |

***Рисунок 1.5. Главное поле (расширенное на весь экран)***

.

## 1.4. Сравнительная таблица по всем проанализированным играм

Чтобы лучше понять, какие преимущества и недостатки проанализированных игр можно выделить, можно воспользоваться сравнительной таблицей. В таблице 1.4 представлены наиболее важные для пользователя критерии.

***Таблица 1.1. Сравнение проанализированных игр***

| **Игра** | Siberian Forest | Жизнь | Игра Конуэ (симулятор модели хищник-жертва) |
| --- | --- | --- | --- |
| **Визуальное представление (поле)** | **+** | **+** | **+** |
| **Самостоятельная расстановка персонажей** | + | + | - |
| **Отображение уровня сытости (здоровья)** | + | - | - |
| **Наличие молодых особей (родившихся во время игры)** | + | - | - |
| **Пользователю видно количество персонажей на поле** | - | + | - |
| **Встроенные шаблоны расстановки** | + | + | + |
| **Время** | + | **+** | **-** |
| **Пошаговая и ускоренная реализации** | + | **+** | **-** |

## 1.5. Вывод по главе

Подводя итог, можно выделить следующие характеристики, которыми должен обладать проектируемый для учебной практики алгоритм:

1. Игра должна иметь игровое поле, на котором пользователь будет создавать свою экосистему.
2. Должны быть представлены несколько видов «клеток»/организмов для дальнейшего взаимодействия.
3. Необходимо вести счет времени в игре.
4. В начале игры пользователю должна предоставляться возможность собственной расстановки или же ранее заготовленной.
5. Необходима пошаговая реализация для наглядности.
6. Также нужна и ускоренная версия происходящих в игре действий.
7. У пользователя должна быть возможность добавлять персонажей во время игры.

Назначение разработанного по выделенным характеристикам алгоритма – развивающие и развлекательные цели. Игра, реализованная на основе разработанного алгоритма, повышает навык прагматичного мышления, тренирует память и внимательность.

# Глава 2. Проектирование алгоритма

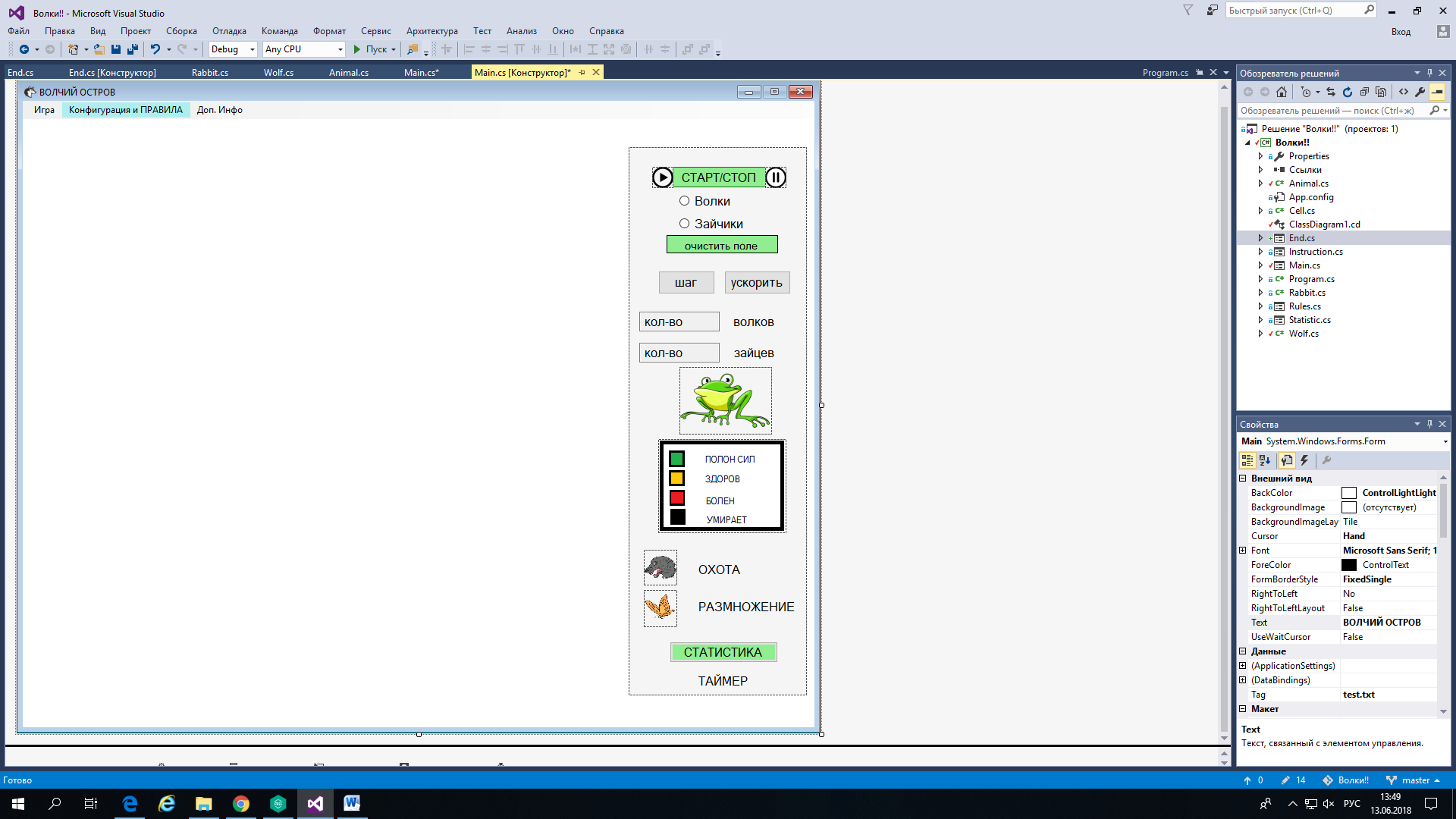
В разработке алгоритма игры можно выделить несколько этапов: создание Windows Forms, разработка структуры файлов, описание необходимых для создания глобальных переменных, проектирование блок-схем, а также краткое словесное описание работы основных алгоритмов и функций.

## 2.1. Описание Windows Forms

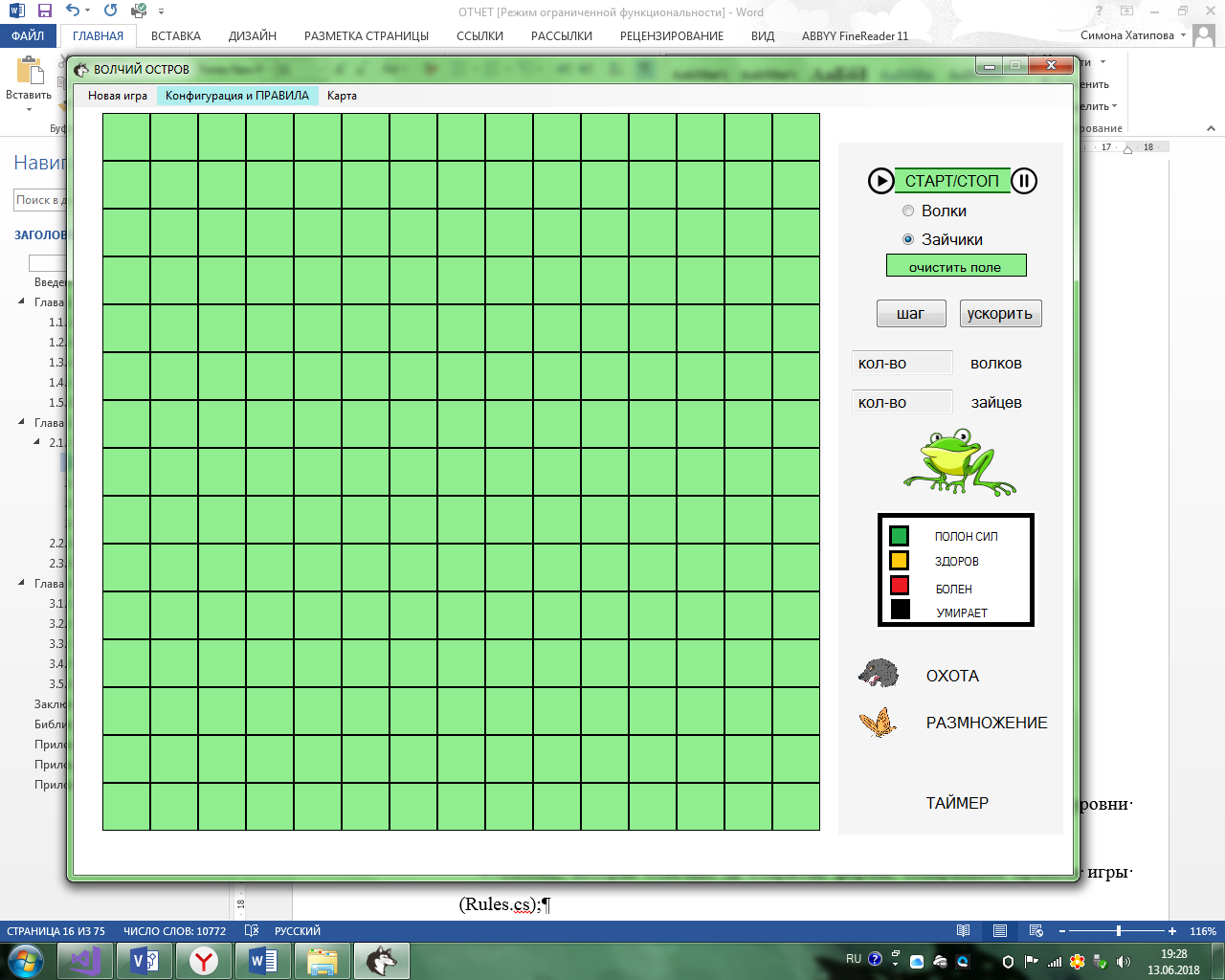
Первый этап в проектировании алгоритма – создание Windows. Формы более наглядны и удобны, чем, например, консольное приложение, поэтому они были выбраны для взаимодействия с пользователем.

### **2.1.1. Main.cs**

(рис.2.1 и рис 2.2.)



***Рисунок 2.1 Форма Main.cs в режиме конструктора***



***Рисунок 2.2. Форма Main.cs во время исполнения проекта***

Зеленое поле не видно в режиме конструктора, так как оно будет нарисовано только после запуска программы.

Форма содержит (основные элементы):

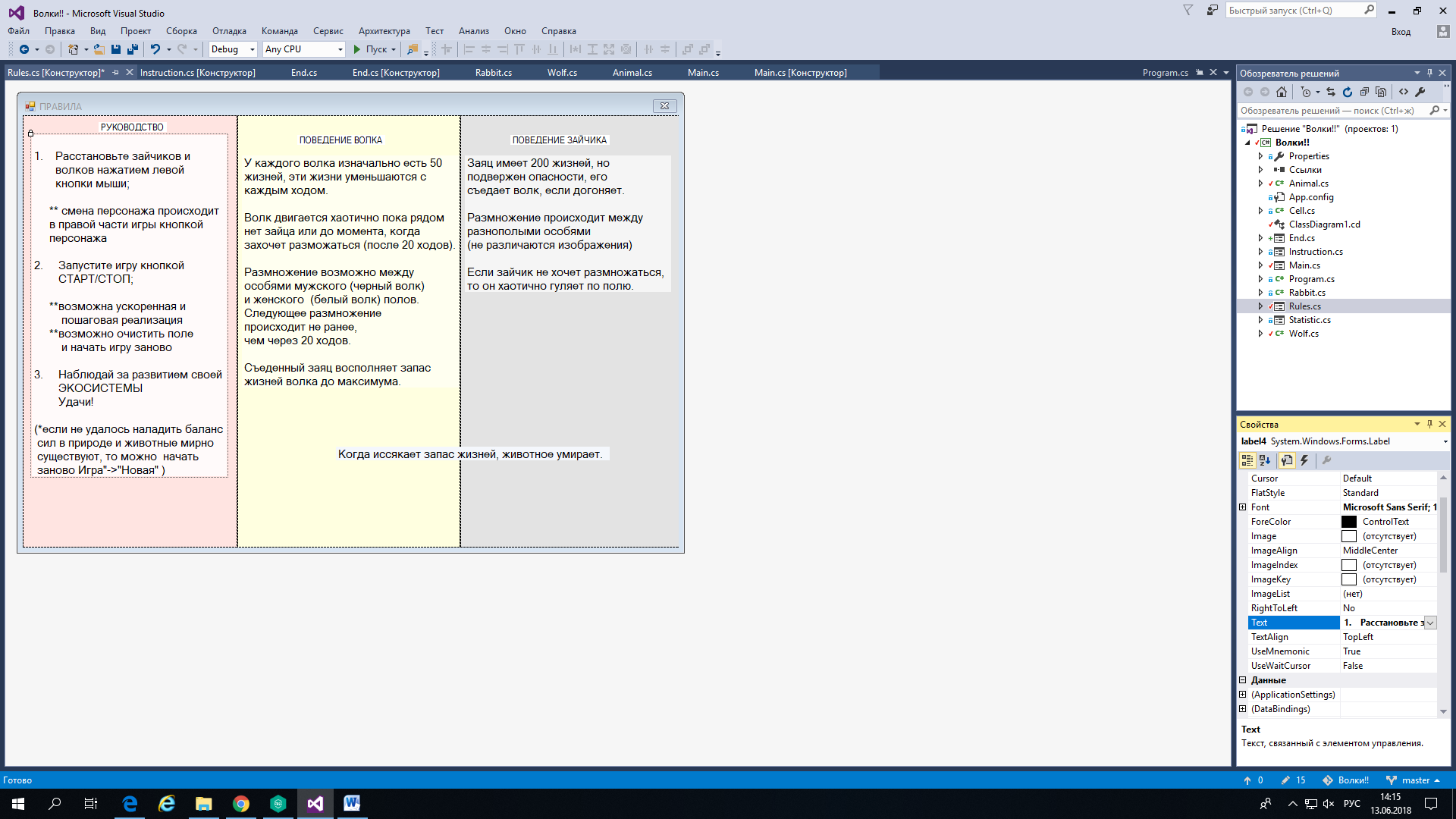
* btnClear, которая отвечает за очистку поля игры, count\_rabbits, count\_wolves, timer\_label;
* btnPlayPause, которая отвечает за запуск или же остановку реализации алгоритмов игры;
* btnQuick, которая отвечает за ускоренную реализацию алгоритмов игры, чтобы пользователь имел возможность за меньшее количество времени увидеть развитие экосистемы(для пользователя время идет быстрее, но в игре и таймер начинает двигаться быстрее), может быть задействована только после того, когда будет включена btnPlayPause;
* btnStep, которая отвечает за пошаговую реализацию алгоритмов игры, чтобы пользователь имел возможность проследить за персонажем на всем его жизненном пути;
* count\_rabbits, который отражает текущее количество зайцев в игре;
* count\_wolves, который отражает текущее количество волков в игре;
* label 1-4, которые выводят текст, необходимый для понимания программы;
* timer\_label, которая отражает текущее время работы запущенной программы;
* pictureBox 1-6, которые показывают пользователю изображения персонажей, процессов в игре и значение цветов фона;
* ИграToolStripMenuItem, которая отвечает за очистку поля и таймера (как и btnClear);
* ДопИнфоToolStripMenuItem, которая отвечает за переход к форме, содержащей карту (Instruction.cs);
* КонфигурацияToolStripMenuItem, которая отвечает за переход к форме, содержащей правила (Rules.cs);

### **2.1.2. Rules.cs**

(рис. 2.3)

Форма содержит:

* label 1-7, которые отражают руководства пользования программой, объяснение поведения волка, объяснение поведения зайца.



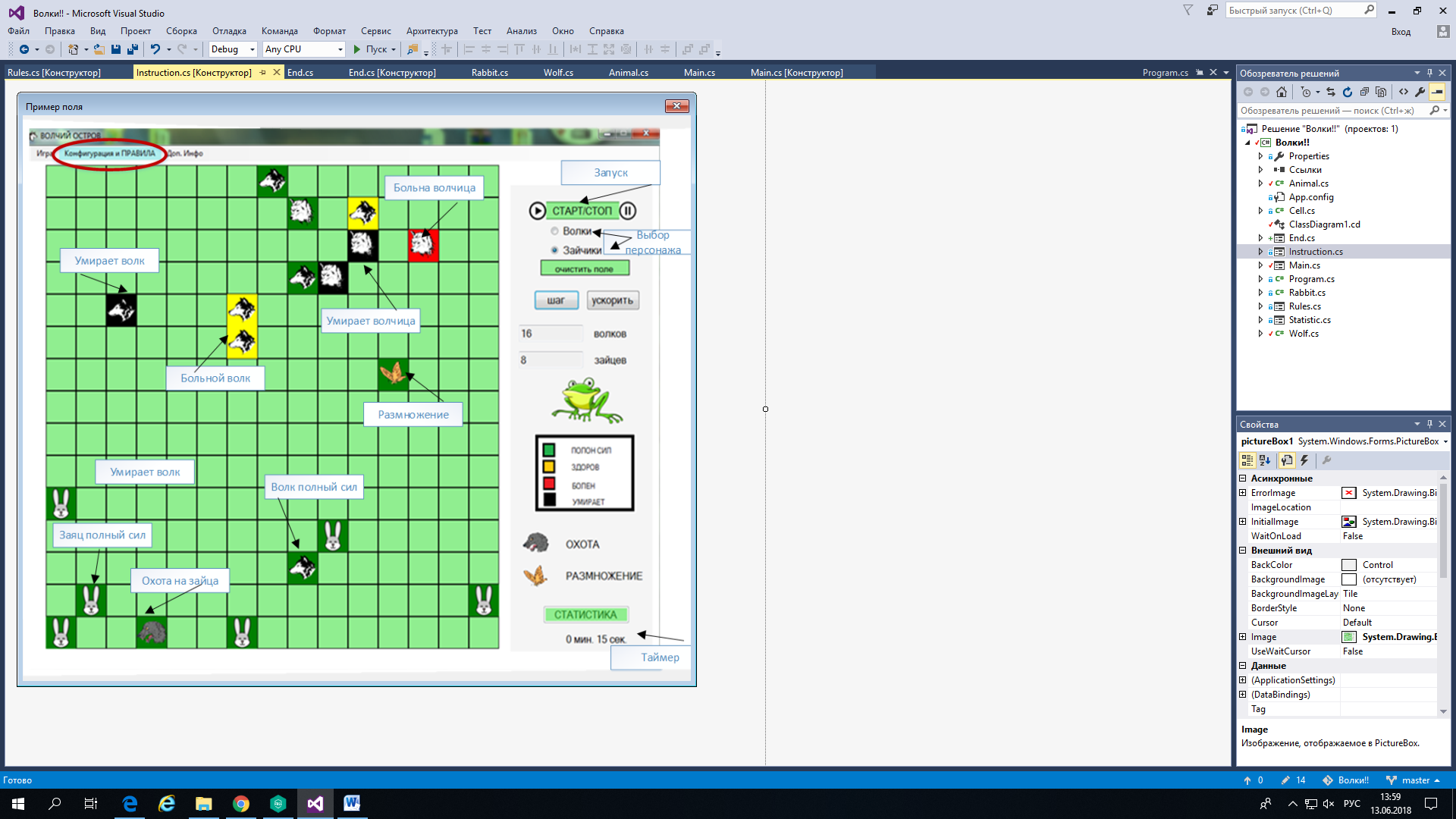
***Рисунок 2.3. Форма Rules.cs***

### **2.1.3. Instruction.cs**

(рис. 2.4)

Форма содержит:

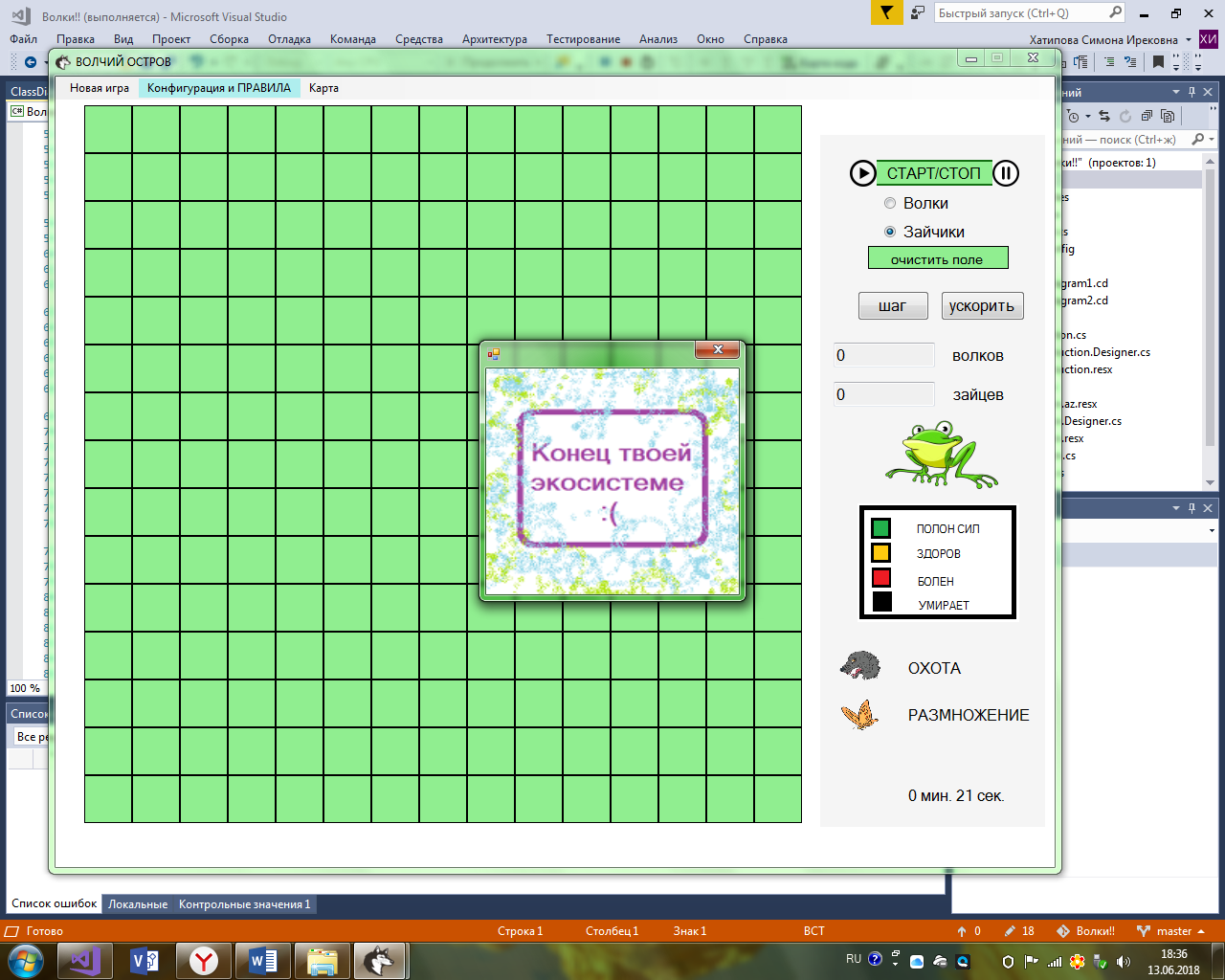
* PictureBox1, который содержит карту с разъяснением элементов находящихся на поле вовремя игры.



***Рисунок 2.4. Форма Instruction.сs***

### **2.1.4. End.cs**

**(**рис. 2.4.1)



***Рисунок 2.4.1. Форма End.cs***

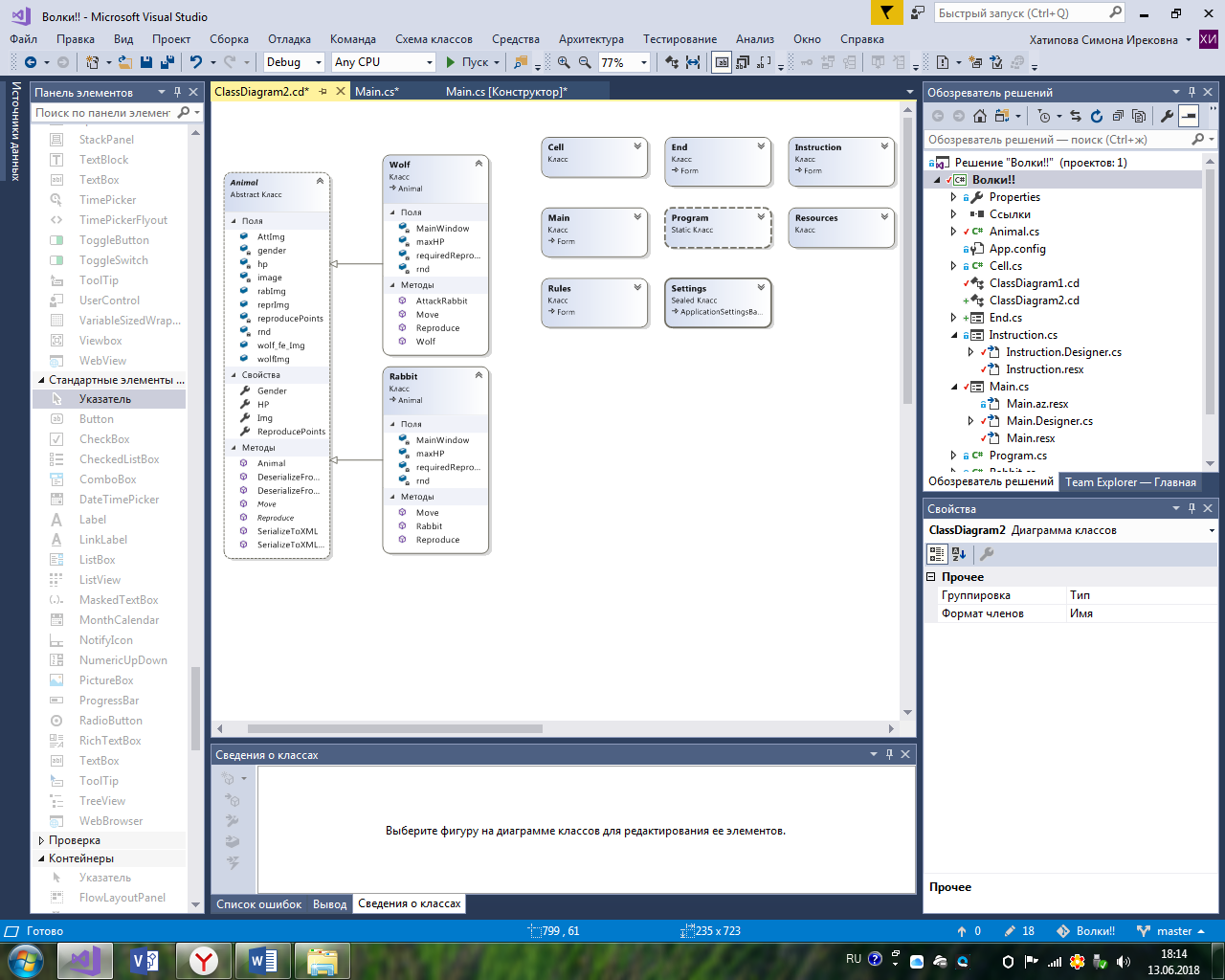
Форма содержит:

* PictureBox1, который содержит информацию о том, что симуляция прекратилась из-за отсутствия на поле хотя бы 1 персонажа.

## 2.2. Классы, используемые при разработке программы

На рисунке 2.6 представлена диаграмма классов.

Класс Animal является базовым классом для производных классов Wolf и Rabbit. Наследники получают шаблон, причем каждый из них заполнит этот его собственной информацией. Abstract class Animal определяет методы, которые производные классы должны реализовать, но сам при этом не обеспечивает реализации одного или нескольких методов.



***Рисунок 2.6. Диаграмма классов проекта***

## 2.3. Блок-схемы и краткое описание основных алгоритмов

Одним из способов записи алгоритмов, наиболее наглядным из всех, является блок-схема. Третий этап проектирования алгоритма – это создание блок-схем для основных функций, а также их описание.

### **2.3.1. Добавление персонажа на поле**

(см. рис. 2.7)

public void AddAnimal(object sender, EventArgs e). Расположен в Main.cs.

В зависимости от желания пользователя добавить на поле зайца (выбрано по умолчанию), выполняется алгоритм вычисления места, указанного пользователем, затем выводится изображение зайца, автоматически у него определяется пол, но изображения не отличаются. В противном случае, если пользователь при выборе персонажа отметит волка, то на поле появятся изображения волка или же волчицы. Автоматически будет вычислено количество уже добавленных на поле персонажей, которое обновляется с каждым добавлением.



***Рисунок 2.7. Добавление персонажа***

### **2.3.2. Действия персонажей**

(см. рис. 2.8)

public void Act(). Расположен в Main.cs.

Запускается алгоритм, управляющий поведением животных, после прорисовки поля. Сначала проверяется наличие животных на поле. Лишь после расстановки хотя бы одного животного на поле, пользователь сможет начать игру, потому что если на поле не остается (или не имеется) ни одного героя, то выведется форма End.cs, в которой говорится, что экосистема нарушена, процесс симуляции проходить не будет (если работал таймер, он остановится). Но если же пользователь разместил на поле героев, то начинает работать алгоритм приоритетности действий зайца: размножиться, а затем погулять по полю, покушать травку, которая растет на всем ареале их обитания. Каждый заяц имеет в запасе 200 жизней, с каждым ходом они уменьшаются на единицу. Здесь происходит проверка, если заяц обладает нулевым запасом энергии, он умирает.

У волка приоритет действий отличается от заячьего, потому что он кушает зайца первым делом, а потом уже заняться остальными делами. Он решает покушать, размножиться, а потом погулять. И так же как и заяц погибнет из-за нехватки здоровья, однако он имеет возможность пополнять его, если настигнет зайца и съест.



***Рисунок 2.8. Модель поведения***

### **2.3.3. Передвижение**

(см. рис. 2.9)

public abstract void Move(); Расположен в Animal.cs.

public override void Move(). Расположен в Wolf.cs.

public override void Move().Расположен в Rabbit.cs.

Алгоритм движения волков схож с алгоритмом движения зайцев (виды: за противоположным полом, просто прогулка), но усложняется желанием покушать. Персонах может двигаться равновероятно в 1 из 8 соседних клеток или посидеть на месте. Равная вероятность соблюдается при условии невозможности выполнения более приоритетных действий. В алгоритме происходит поиск нахождение желаемого сдвига по вертикали и горизонтали одновременно, затем проверяется, не выходит ли эта новая клетка за границы поля. Если выбранная соседняя клетка находится за границами поля, то персонаж остается на месте. Далее у героя отнимается одна жизнь за совершенный ход, и, в зависимости от текущего количества жизней, выбирается задний фон изображения. В то же время, на каждом ходе растет желание размножаться.



***Рисунок 2.9. Передвижение***

### **2.3.4. Размножение (рассмотрим у зайцев)**

(см. рис. 2.12)

public abstract bool Reproduce(); Расположен в Animal.cs.

public override bool Reproduce();. Расположен в Wolf.cs.

public override bool Reproduce();.Расположен в Rabbit.cs.

Заяц оглядывается вокруг и ищет партнера для размножения, в случае, если он уже набрал достаточно опыта, прошел минимум 20 шагов, он перемещается на клетку партнера. В этой же клетке появляется новый персонаж игры, детеныш. И сразу после обе взрослые особи теряют желание размножаться.



***Рисунок 2.10. Размножение зайцев***

### **2.3.4. Охота**

(см. рис. 2.11)

public bool AttackRabbit(). Размещен в Wolf.cs.

Охотится волк на зайца только при условии, что он хочет кушать, здоровье упало. Съеденный зайчик восполняет запас здоровья волку до максимума. Для трапезы животные должны находиться в 1 клетке. Пол не важен. После охоты у валка увеличивается желание размножаться, так как он уже сыт и полностью здоров. Погибший заяц исчезает с поля игры.



***Рисунок 2.11. Охота***

## 2.4. Вывод по главе

Результатом проектирования алгоритма игры в слова является создание всех необходимых Windows Forms, распределение между элементами созданных форм функций (то есть определение того, за что отвечает каждый элемент), описание работы распределённых по элементам форм функций и определение структуры файлов, которые будут созданы при реализации спроектированного алгоритма.

# Глава 3. Реализация спроектированного алгоритма

Для реализации спроектированного алгоритма, необходимо определить, какие глобальные переменные должны использоваться в программе, а также более подробно описать работу основных функций.

## 3.1. Описание глобальных переменных

Удобно использовать глобальные переменные для настройки поля или исходных констант для использования в функциях. Можно менять значение только в одном месте и вся программа будет работать уже с новым значением.

В программе используются следующие глобальные переменные:

* public static int marginLeft = 30, marginTop = 30; – отступы таблицы от левого и верхнего краёв
* public static int rows = 15, cols = 15; – размер таблицы
* public static int cellSize = 50; – размер ячейки
* public static bool isPlaying = false; – продолжить/остановить
* public static bool isQuick = false; – ускорить/нет
* public static List<Rabbit> rabbits = new List<Rabbit>(); – писок зайчиков
* public static List<Wolf> wolves = new List<Wolf>(); – список волков
* public static List<Animal> animals= new List<Animal>(); –список всех животных
* public static bool addRabbit = true; – если true, то добавляются зайцы. если false, то волки
* public static int tim = 0; – начальное значение таймера
* public static int speed = 200; – скорость передвижения
* public static int oneSecond=1000; – в миллисекундах
* public static Timer timer = new Timer(); – переменная таймера общего
* public static Timer timer\_act = new Timer(); – переменная таймера в действии

## 3.2. Тестирование разработанного алгоритма

Для того, чтобы убедиться в корректности работы разработанного алгоритма, необходимо провести ряд тестов, направленных на поиск ошибок в работе программы. Результаты тестирования алгоритма игры в слова, разработанного в рамках данной учебной практики, представлены в таблице 1 (см. Приложение 1). Таблица 3.1 построена на основе тестов из таблицы 1 (см. Приложение 1).

***Таблица 3.1. Тестирование методом черного ящика (на основе входных и выходных данных)***

| № | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| На поле нет ни одного персонажа (изначально) | + |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| На поле нет ни одного персонажа (умерли в игре) | + |  | + | + | + | + | + | + | + | + | + | + | + | + | + |
| На поле 1 заяц |  |  |  |  |  |  |  |  |  |  |  |  | + |  |  |
| На поле несколько зайцев |  | + |  | + | + | + | + | + | + | + | + | + |  |  |  |
| На поле 1 волк |  |  |  |  |  |  |  |  |  |  |  |  |  | + |  |
| На поле несколько волков |  |  | + |  | + | + | + | + | + | + | + | + |  |  |  |
| На поле 1 заяц и 1 волк |  |  |  |  |  |  |  |  |  |  |  |  |  |  | + |
| На поле равное количество зайцев и волков |  |  |  |  | + |  |  | + | + | + | + | + |  |  |  |
| На поле волков больше, чем зайцев |  |  | + |  |  |  | + |  |  |  |  |  |  | + |  |
| На поле зайцев больше, чем волков |  | + |  |  |  | + |  |  |  |  |  |  | + |  |  |
| На поле работающей программы всегда есть персонаж (бесконечна) |  | + |  |  |  |  |  |  |  |  |  |  |  |  |  |

Еще проведем тестирование на нахождение сбалансированной экосистемы (табл. 3.2).

***Таблица 3.2. Пропорции расстановки персонажей***

|  |  |  |
| --- | --- | --- |
| Кол-во зайцев | Кол-во волков | Общее время существования системы |
| 1 | 1 | 0:42 |
| 2 | 2 | 0:42 |
| 10 | 0 | Бесконечность |
| 10 | 1 | Бесконечность (устойчивая, умеренный рост зайцев) |
| 10 | 2 | 1:07 |
| 2 | 10 | 0:16 |
| 0 | 10 | 0:13 |
| 30 | 3 | Бесконечная развивающаяся  В начале сохраняется соотношение (3:1) |

Тестирование записи текущего результата уровня в файл можно представить в виде скриншота созданного файла (рис. 3.1 и рис. 3.2).



***Рисунок 3.1. Бесконечная развивающаяся система из расстановки 30 к 3***



***Рисунок 3.2. Начальная расстановка 10 к 1***

Можно сделать вывод, что пропорция 10 зайцев к 1 волку дает успешный результат выживания экосистемы, но если добавить волков больше одного, сохранить пропорции, тот и волки будут развиваться.

## 3.3. Руководство пользователя

Руководство, поведение волков и зайцев представлено в ФормеRules.cs и выделено на панели меню цветом.

1. Расставьте зайчиков и волков нажатием левой кнопки мыши;

Смена персонажа происходит правой части игры кнопкой персонажа.

1. Запустите игру кнопкой СТАРТ/СТОП;

Возможна ускоренная и пошаговая реализация.

Возможно очистить поле и начать игру заново.

1. Наблюдай за развитием своей ЭКОСИСТЕМЫ. Анализируй, улучшай результат. Создай сбалансированную систему существования модели «хищник-жертва» в природе

Удачи! Если не удалось наладить баланс сил в природе и животные мирно существуют, то можно начать заново, щелкнув по пункту меню «Новая Игра»

Пользователь может посмотреть обозначения в меню игры «Карта».

*Поведение волка*

У каждого волка изначально есть 50 жизней, эти жизни уменьшаются с каждым ходом. Волк двигается хаотично пока рядом нет зайца или до момента, когда захочет размножаться (после 20 ходов). Размножение возможно между особями мужского (черный волк) и женского (белый волк) полов. Следующее размножение происходит не ранее, чем через 20 ходов. Съеденный заяц восполняет запас жизней волка до максимума. Когда иссякнет запас жизней, животное умрет.

*Поведение зайца*

Заяц имеет 200 жизней, но подвержен опасности, его съедает волк, если догоняет. Размножение происходит между разнополыми особями (не различаются изображения) Если зайчик не хочет размножаться, то он хаотично гуляет по полю.

Когда иссякнет запас жизней, животное умрет.

## 3.4. Вывод по главе

Результатом реализации спроектированного алгоритма является написанная на языке C# игра в слова. Программа была написана в соответствии с описанием основных функций, а также протестирована в соответствии с тестами, описанными в пункте 3.2. Было предоставлено руководство пользователя, сведения о поведении персонажей в каждой возможной ситуации.

# Заключение

При прохождении учебной практики были реализованы все задачи, поставленные изначально, благодаря чему была достигнута главная цель – создана игра «Волчий остров».

1. Была подробно изучена предметная область: были проанализированы несколько игр на данную тематику, находящихся в Интернете. На основе выделенных преимуществ и недостатков игр были сформированы требования, которые необходимо было выполнить в проектируемом алгоритме.
2. Были разработаны Windows Forms для взаимодействия с пользователем. Был продуман интерфейс игры, а также определены функции каждого элемента, расположенного на формах. Основные функции были кратко описаны, а также представлены в виде блок-схем.
3. Был реализован алгоритм игры на языке C#, чему предшествовало подробное описание работы основных функций, описание используемых в программе глобальных переменных.

Преимуществами (для пользователя) созданной игры по сравнению с её аналогами, проанализированными в Главе 1 являются:

* приятная для пользователя реализация интерфейса;
* наличие таймера, действующего на всем протяжении игры;
* реализация разнополых особей обоих персонажей и разнополого размножения, как следствие;
* понятный интерфейс и наличие правил, руководства к действию, объяснения поведения персонажей на поле;
* пошаговая и ускоренная реализации;
* яркое отображение уровня здоровья и присутствие легенды, разное отображение размножения, охоты, разнополых волков и зайцев.

1. Реализованных алгоритм был протестирован, ошибок в работе выявлено не было. Также было разработано руководство пользователя, которое может быть использовано по назначению, если программа будет представлена пользователям.

Разработанная в результате прохождения учебной практики игра может быть применена для развития внимательности, тренировки навыков управления и моделирования процессов, а также развития логического мышления как у детей, так и у взрослых. В соответствии с конкретными требованиями заказчика, данная игра может быть доработана и предоставлена широкому кругу пользователей.

# Библиографический список

1. Siberian Forest [Электронный ресурс]//Habr.com [Сайт] URL:  https://habr.com/post/336330/ (дата обращения: 04.04.2018).
2. Конуэ [Электронный ресурс]//ConwayLife [Сайт] URL:  http://www.conwaylife.com/ (дата обращения: 04.04.2018).
3. Имитационная модели «хищник-жертва» [Электронный ресурс]//App Store [Магазин приложений] URL:  https://appsto.re/ru/bcFpab.i (дата обращения: 04.04.2018).
4. Словарь [Электронный ресурс]//Wiktinary [Сайт] URL: https://ru.wiktionary.org/ (дата последнего обращения: 14.05.2018).
5. Троелсен Э. Язык программирования C# 2010 и платформа .NET 4: пер. с англ. М.: Издательский дом "Вильямс", 2012, 1392 с.
6. Microsoft Developer Network [Сайт] URL: https://msdn.microsoft.com/ (дата обращения 30.04.2018).

# Приложение 1. Тесты

***Таблица 1. Тестирование***

| № | Начальное состояние | Ожидаемый результат | Полученный результат | Комментарий |
| --- | --- | --- | --- | --- |
| 1 | Пользователь не добавил на поле ни 1 персонажа. Нажатие на кнопку «СТАРТ/СТОП». | Открытие формы End.cs, в которой говорится о нарушении работы системы, об отсутствии возможности управлять персонажами. | Открытие формы End.cs, в которой говорится о нарушении работы системы, об отсутствии возможности управлять персонажами. | Достигнут верный результат. |
| 2 | Пользователь добавил несколько зайцев (5). Нажатие на кнопку «СТАРТ/СТОП». | В течение первой минуты игры популяция увеличилась примерно в пропорции от 5 до 12 раз, а к концу второй минуты примерно в 35-45 раз.  Экосистема процветает, она бесконечна. | Зайцы начали размножаться уже через 6 секунд после запуска. В течение первой минуты игры популяция увеличилась примерно в 11 раз, а к концу второй минуты примерно в 43 раз. | Достигнут верный приемлемый результат. |
| 3 | Пользователь добавил на поле несколько волков (5). Нажатие на кнопку «СТАРТ/СТОП». | Система должна просуществовать недолгий промежуток времени. Возможно размножение волков. | Через 12 секунд погибли все волки, размножения не произошло, вывелась форма End.cs. | Достигнут верный приемлемый результат. |
| 4 | Пользователь добавил несколько зайцев (5). Нажатие на кнопку «СТАРТ/СТОП».А затем нажал кнопку «очистить поле» | Должны очиститься поле, таймер, количество персонажей на поле в текущий момент. | Очистились поле, таймер, количество персонажей на поле в текущий момент. | Достигнут верный результат. |
| 5 | Пользователь добавил на поле несколько волков (10) и несколько зайцев (10).(Равное количество) Нажатие на кнопку «СТАРТ/СТОП». | Должно произойти размножение и тех, и тех особей, но значительного увеличения популяции не предусмотрено. Система может быть бесконечна только в случае, если останется 2 разнополые особи зайца, и они найдут друг друга на поле. | Через 6 секунд начали погибать зайцы, через 15 – волки. На 55 секунде умер последний волк. На 1м 10 сек умер от старости последний заяц. | Достигнут верный приемлемый результат. |
| 6 | Пользователь добавил на поле несколько волков (5) и несколько зайцев (10).(волков меньше) Нажатие на кнопку «СТАРТ/СТОП». | Возможно несколько вариантов развития событий: экосистема может стать бесконечной при быстром размножении зайцев, но если волки найдут их быстрее, то экосистема прекратит существовать через примерно 1 минуту. | Через 5 секунд зайцы начали размножаться, но на 7 секунде оба вида стали медленно вымирать. Начиная с 46 секунды, почти мирно существовали 2 зайца и 1 волк, еще через 10 секунд умер волк, а на 1 минуте 7 секундах погиб последний заяц. | Достигнут верный приемлемый результат. |
| 7 | Пользователь добавил на поле несколько волков (10) и несколько зайцев (5).(волков больше) Нажатие на кнопку «СТАРТ/СТОП». | Естественная смерть волков и зайцев, почти одновременное сокращение численности видов. Максимальное существование около 40 секунд. | Резкое сокращение количества зайцев уже через 3 секунды, небольшая репродукция волков до 12 особей, потом стремительное сокращение популяции. В 19 секунд погибла система. | Достигнут верный результат. |
| 8 | Пользователь добавил на поле несколько волков (10) и несколько зайцев (10).(Равное количество) Нажатие на кнопку «шаг». | Необходимо многократное нажатие кнопки. Почти одновременное сокращение популяции, существование больше 200 ходов по максимуму жизни зайца. Таймер остановлен на 1 секунде. | Мирное существование до 12 секунд, потом началась охота, таймер остановлен на 1 секунде. Существование 200 ходов. | Достигнут верный результат. |
| 9 | Пользователь добавил на поле несколько волков (10) и несколько зайцев (10).(Равное количество) Нажатие на кнопку «СТАРТ/СТОП», а затем «ускорить» | Должно произойти ускорение жизни на поле, размножение и тех, и тех особей, но значительного увеличения популяции не предусмотрено. Система может быть бесконечна только в случае, если останется 2 разнополые особи зайца, и они найдут друг друга на поле. | Ускорилось протекание биологических процессов на поле, на 1 минуте 52 секундах экосистема погибла. | Достигнут верный приемлемый результат. |
| 10 | Пользователь добавил на поле несколько волков (10) и несколько зайцев (10).(Равное количество) Нажатие на кнопку «СТАРТ/СТОП», а затем «ускорить», ещё раз «ускорить» (отключение) | Должно произойти ускорение жизни на поле после первого нажатия на кнопку, размножение и тех, и тех особей, но значительного увеличения популяции не предусмотрено. Когда кнопка ускорить будет нажата 2 раз, время в игре станет стандартным. Система может быть бесконечна только в случае, если останется 2 разнополые особи зайца, и они найдут друг друга на поле. | Ускорилось протекание биологических процессов на поле с помощью нажатия кнопки ускорения, посредством повторного нажатия, вернулось умеренное движение героев, на 1 минуте 13 секундах экосистема погибла. | Достигнут верный приемлемый результат.. |
| 11 | Пользователь добавил на поле несколько волков (10) и несколько зайцев (10).(Равное количество) Нажатие на кнопку «СТАРТ/СТОП», а затем еще раз «СТАРТ/СТОП». | Должна начаться реализация алгоритмов игры, а потом по повторному нажатию на кнопку должны остановиться все протекаемые процессы. | Запуск системы и остановка работы. | Достигнут верный результат. |
| 12 | Пользователь добавил на поле несколько волков (10) и несколько зайцев (10).(Равное количество) Нажатие на кнопку «СТАРТ/СТОП», а затем «шаг» | Должна начаться автоматическая работа системы, но с нажатием шага остановится таймер и начнется работа после нажатия шага. | Экосистема просуществовала 12 секунд автоматически. Затем была переключена в ручное управление пошагово, остановился таймер. | Достигнут верный результат. |
| 13 | Пользователь добавил на поле 1 зайца. После нажал кнопку «СТАРТ/СТОП». | Некоторое время существования (около 42 секунд). | к2 секунды существования зайца. Затем вывелась форма End.cs. | Достигнут верный результат. |
| 14 | Пользователь добавил на поле 1 зайца. После нажал кнопку «СТАРТ/СТОП». | Некоторое время существования (около 12 секунд). | 11 существования волка на поле. Затем вывелась форма End.cs. | Достигнут верный результат. |
| 15 | Пользователь добавил на поле 1 зайца и 1 волка. После нажал кнопку «СТАРТ/СТОП». | Возможно, герои не встретятся на поле, и тогда система просуществует по длительности жизни зайца. Или же волк съест зайца, восполнит жизни, и тогда система просуществует по длительности жизни волка. | Волк не нашел зайца на поле, заяц мирно бродил по полю и умер от старости спустя 42 секунды. | Достигнут верный результат. |

# Приложение 2. Листинг программы (с комментариями)

*Форма Main.cs*

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace ВолкииЁлки

{

public partial class Main : Form

{

// ЗДЕСЬ ВЫНЕСЕНЫ ВСЕ ЗНАЧЕНИЯ РАЗМЕРОВ И ОТСТУПОВ. ЕСЛИ МЕНЯЕШЬ ИХ ЗДЕСЬ, ТО ОНИ МЕНЯЮТСЯ ВЕЗДЕ

// удобно же! :)

//отступы таблицы от левого и верхнего краёв

public static int marginLeft = 30, marginTop = 30;

//размер таблицы

public static int rows = 15, cols = 15;

//размер ячейки

public static int cellSize = 50;

// продолжить/остановить

public static bool isPlaying = false;

public static bool isQuick = false;

//список зайчиков

public static List<Rabbit> rabbits = new List<Rabbit>();

// список волков

public static List<Wolf> wolves = new List<Wolf>();

public static List<Animal> animals= new List<Animal>();

// если true, то добавляются зайцы. если false, то волки

// для изменения животного надо добавить какой-нибудь переключатель на форме

public static bool addRabbit = true;

//количество на поле животных

public static int tim = 0;

public static int speed = 200;

public static int oneSecond=1000;

public static Timer timer = new Timer();

public static Timer timer\_act = new Timer();

private void toolStripMenuItem2\_Click(object sender, EventArgs e)

{

}

private void label3\_Click(object sender, EventArgs e)

{

}

private void pictureBox3\_Click(object sender, EventArgs e)

{

}

private void конфигурацияToolStripMenuItem\_Click(object sender, EventArgs e)

{

Волки\_\_.Rules rules = new Волки\_\_.Rules();

rules.Show();

}

//Обработчик нажатия на кнопку "Старт"

private void Main\_Load(object sender, EventArgs e)

{

// рисуем таблицу при загрузки формы

DrawTable();

Act();

Time();

timer.Stop();

timer\_act.Stop();

// по умолчанию сначала создаются зайчики

makeRabbits.Checked = true;

}

public Main()

{

InitializeComponent();

}

//выводим на экран таблицу

public void DrawTable()

{

for (int row = 0; row < rows; row++)

for (int col = 0; col < cols; col++)

{

Cell newCell = new Cell(col, row);

// добавляем обработчик клика для всех клеток

newCell.cell.Click += AddAnimal;

// добавим кнопку на форму

this.Controls.Add(newCell.cell);

}

}

public void AddAnimal(object sender, EventArgs e)

{

if (addRabbit)

{

// вычисляем столбец и строку

int x = (((Button)sender).Location.X - marginLeft) / cellSize;

int y = (((Button)sender).Location.Y - marginTop) / cellSize;

Rabbit rabbit = new Rabbit(x, y, this);

this.Controls.Add(rabbit.Img);

rabbit.Img.BringToFront();

rabbits.Add(rabbit);

}

if (!addRabbit)

{

// вычисляем столбец и строку

int x = (((Button)sender).Location.X - marginLeft) / cellSize;

int y = (((Button)sender).Location.Y - marginTop) / cellSize;

Wolf wolf = new Wolf(x, y,this);

this.Controls.Add(wolf.Img);

wolf.Img.BringToFront();

wolves.Add(wolf);

}

count\_wolves.Text = wolves.Count.ToString();

count\_rabbits.Text = rabbits.Count.ToString();

}

public void Act()

{

// задержка перед повторением движения

// увеличь, если хочешь замедлить и наоборот

timer\_act.Interval = speed;

timer\_act.Tick += new EventHandler((o, ev) =>

{

try

{

if (wolves.Count.ToString() == rabbits.Count.ToString())

if (wolves.Count.ToString() == "0")

{

End form = new ВолкииЁлки.End();

form.Show();

isPlaying = false;

}

foreach (Rabbit r in rabbits)

{

//приоритет действий: размножиться -> погулять

if (!r.Reproduce())

r.Move();

if (r.HP <= 0)

{

// удаляем мёртвого зайца

Controls.Remove(r.Img);

Main.rabbits.Remove(r);

}

}

foreach (Wolf w in wolves)

{

//приоритет действий: поесть -> размножиться -> погулять

if (!w.AttackRabbit())

if (!w.Reproduce()) w.Move();

if (w.HP <= 0)

{

// удаляем мёртвого волка

Controls.Remove(w.Img);

Main.wolves.Remove(w);

}

}

}

catch { }

count\_wolves.Text = wolves.Count.ToString();

count\_rabbits.Text = rabbits.Count.ToString();

// прервать анимацию

if (!isPlaying)

{

Timer t = o as Timer;

t.Stop();

}

});

timer\_act.Start();

}

// ОБРАБОТЧИКИ НАЖАТИЙ И ДРИГИЕ ВТОРОСТЕПЕННЫЕ ВЕЩИ

private void btnPlayPause\_Click(object sender, EventArgs e)

{

if (isPlaying) isPlaying = false;

else

{

isPlaying = true;

//возобновить анимацию

//Act();

//Time();

timer\_act.Start();

timer.Start();

}

}

public void Time()

{

// задержка перед повторением движения

// увеличь, если хочешь замедлить и наоборот

timer.Interval = oneSecond;

timer.Tick += new EventHandler((o, ev) =>

{

tim++;

//var time = TimeSpan.FromSeconds(sec).ToString("HH:mm:ss");

timer\_label.Text = (tim/60).ToString()+" мин. "+ (tim%60).ToString() +" сек.";

//"Текущее время: T: {0,T}"+ tim; //Convert.ToDateTime(tim.ToString() );//Format(DateTime.tim.ToString("HH:mm:ss");

// прервать анимацию

if (!isPlaying)

{

Timer t = o as Timer;

t.Stop();

}

});

timer.Start();

}

private void btnQuick\_Click(object sender, EventArgs e)

{

if (!isQuick)

{

timer\_act.Interval = speed / 50;

timer.Interval /= 50;

isQuick = true;

}

else

{

timer\_act.Interval = speed ;

timer.Interval = oneSecond ;

isQuick = false ;

}

// timer1.Enabled = !timer1.enabled;

}

private void timer\_label\_Click(object sender, EventArgs e)

{

}

private void Timer1\_Tick(object sender, EventArgs e)

{

}

private void размерToolStripMenuItem\_Click(object sender, EventArgs e)

{

}

private void btnClear\_Click(object sender, EventArgs e)

{

foreach (Wolf w in wolves)

Controls.Remove(w.Img);

foreach (Rabbit r in rabbits)

Controls.Remove(r.Img);

wolves.Clear();

rabbits.Clear();

isPlaying = false;

count\_rabbits.Text = "0";

count\_wolves.Text = "0";

tim = 0;

timer\_label.Text= "0 сек";

}

private void btnStep\_Click(object sender, EventArgs e)

{

try

{

foreach (Rabbit r in rabbits)

{

//приоритет действий: размножиться -> погулять

if (!r.Reproduce())

r.Move();

if (r.HP <= 0)

{

// удаляем мёртвого зайца

Controls.Remove(r.Img);

Main.rabbits.Remove(r);

}

}

foreach (Wolf w in wolves)

{

//приоритет действий: поесть -> размножиться -> погулять

if (!w.AttackRabbit())

if (!w.Reproduce()) w.Move();

if (w.HP <= 0)

{

// удаляем мёртвого волка

Controls.Remove(w.Img);

Main.wolves.Remove(w);

}

}

}

catch { }

count\_wolves.Text = wolves.Count.ToString();

count\_rabbits.Text = rabbits.Count.ToString();

isPlaying = false;

}

private void makeWolves\_CheckedChanged(object sender, EventArgs e)

{

addRabbit = false;

}

private void makeRabbits\_CheckedChanged(object sender, EventArgs e)

{

addRabbit = true;

}

private void допИнфоToolStripMenuItem\_Click(object sender, EventArgs e)

{

Волки\_\_.Instruction form = new Волки\_\_.Instruction();

form.Show();

}

private void count\_wolves\_TextChanged(object sender, EventArgs e)

{

}

}

}

*Класс Animal.cs*

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace ВолкииЁлки

{

// это общий класс для зайцев и волков

public abstract class Animal

{

static Random rnd = new Random();

// пусть жизни и пол будут у всех (если потом понадобятся)

// зайчики могут есть травку

int hp;

// если true, то самец

bool gender;

// определяет возможность размножения

// пока не будет больше определённого значения, размножение не получится

int reproducePoints;

// само изображение

PictureBox image = new PictureBox();

public static Image rabImg = Image.FromFile("зайка.png"),

wolfImg = Image.FromFile("волк.png"),

wolf\_fe\_Img = Image.FromFile("волчица.png"),

reprImg = Image.FromFile("бабочка для разм.png"),

AttImg = Image.FromFile("для охоты.png");

//конструктор

public Animal(int x, int y)

{

image.Width = Main.cellSize;

image.Height = Main.cellSize;

// надо сделать задний фон прозрачным, поэтому цвет совпадает с цветом ячейки

image.BackColor = Color.Green;

// положение изображения

image.Location = new Point(x \* Main.cellSize + Main.marginLeft, y \* Main.cellSize + Main.marginTop);

// картинка будет уменьшаться по размеру клетки. Лучше подбирать близкие к квадратным

image.SizeMode = PictureBoxSizeMode.Zoom;

// изначально все особи способны размножаться

ReproducePoints = 0;// будет увеличиваться с количеством ходов, когда достигнет требуемого, животное сможет размножаться

}

// свойства //всего 30 здоровья мало!! надо 200

public int HP

{

get { return hp; }

set

{

hp = value;//200 и 50

}

}

public int ReproducePoints

{

get { return reproducePoints; }

set { reproducePoints = value; }

}

[XmlElement("Gender")]

public bool Gender

{

get { return gender; }

set { gender = value; }

}

[XmlElement("Img")]

public PictureBox Img

{

get { return image; }

set { image = value; }

}

// АБСТРАКТНЫй, т.к.зайцы свободно передвига.ncz

public abstract void Move();

// метод размножения немного отличается

// если был создан ребёнок, то вернёт true

public abstract bool Reproduce();

//удаление мертвых особей в ACT() при просмотре массива животных

}

}

*Класс Wolf.cs*

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace ВолкииЁлки

{

public class Wolf : Animal

{

Form MainWindow;

int requiredReproducePoints = 15;

// максимальное количество очков здоровья

int maxHP = 50;

// сколько здоровья добавляет заяц

static Random rnd = new Random();

public Wolf(int x, int y,Form MainWindow) : base(x, y)

{

if (rnd.Next(0, 2) == 0)

{

Gender = true;

Img.Image = Image.FromFile("волк.png");

}

else

{

Gender = false;

Img.Image = Image.FromFile("волчица.png");

}

HP = maxHP;

this.MainWindow = MainWindow;

}

public override bool Reproduce()

{

// аналогично зайцам, но с учётом пола

if ((ReproducePoints > requiredReproducePoints)&&(HP>maxHP\*0.7))

{

foreach (Wolf w in Main.wolves)

{

//не считая самого себя

if (w != this)

{

if ((Math.Abs(Img.Location.X - w.Img.Location.X) <= Main.cellSize && Math.Abs(Img.Location.Y - w.Img.Location.Y) <= Main.cellSize) && w.ReproducePoints > requiredReproducePoints && this.Gender != w.Gender) // разнополые особи

{

// перемещаем волка на клетку партнёра для начала процесса

this.Img.Location = new Point(w.Img.Location.X, w.Img.Location.Y);

// создаём нового волка в той же клетке

Wolf child = new Wolf((w.Img.Location.X - Main.marginLeft) / Main.cellSize, (w.Img.Location.Y - Main.marginTop) / Main.cellSize, MainWindow);

Main.wolves.Add(child);

MainWindow.Controls.Add(child.Img);

child.Img.BringToFront();

child.Img.Image = reprImg;

// оба временно теряют желание размножаться

this.ReproducePoints = 0;

w.ReproducePoints = 0;

return true;

}

}

}

}

return false;

}

public override void Move()

{

// 0: - ; 1: стой на месте; 2: +

int verticalShift = rnd.Next(0, 3);

int horizontalShift = rnd.Next(0, 3);

// смещение по вертикали

//чтобы не вышел за пределы таблицы надо проверить положение относительно её границ

if (verticalShift == 0 && Img.Location.Y > Main.marginTop) Img.Location = new Point(Img.Location.X, Img.Location.Y - Main.cellSize);

else if (verticalShift == 2 && Img.Location.Y < Main.marginTop + (Main.rows - 1) \* Main.cellSize) Img.Location = new Point(Img.Location.X, Img.Location.Y + Main.cellSize);

// смещение по горизонтали

if (horizontalShift == 0 && Img.Location.X > Main.marginLeft) Img.Location = new Point(Img.Location.X - Main.cellSize, Img.Location.Y);

else if (horizontalShift == 2 && Img.Location.X < Main.marginLeft + (Main.cols - 1) \* Main.cellSize) Img.Location = new Point(Img.Location.X + Main.cellSize, Img.Location.Y);

// волк очень сильно страдает из-за движения, и его жизнь уменьшается

HP--;

if (HP > maxHP \* 0.6) Img.BackColor = Color.Green;

else if (HP > maxHP \* 0.3) Img.BackColor = Color.Yellow;//khaki

else if (HP > 10) Img.BackColor = Color.Red;

else Img.BackColor = Color.Black;

//возвращаем картинку после охоты

if (Gender)

Img.Image = wolfImg;

else

Img.Image = wolf\_fe\_Img;

// при движении желание спариваться возрастает

ReproducePoints++;

}

public bool AttackRabbit()

{

//ReproducePoints++;

// если уровень здоровья меньше 70% от максимального, то волк начинает охотиться

if (HP < maxHP \* 0.8)

{

// ищем зайчика

foreach (Rabbit r in Main.rabbits)

{

if (Math.Abs(Img.Location.X - r.Img.Location.X) <= Main.cellSize && Math.Abs(Img.Location.Y - r.Img.Location.Y) <= Main.cellSize)

{

// перемещаем волка на клетку зайца для начала трапезы

this.Img.Location = new Point(r.Img.Location.X, r.Img.Location.Y);

// волк убивает зайца

MainWindow.Controls.Remove(r.Img);

Main.rabbits.Remove(r);

ReproducePoints++;

HP = maxHP;

this.Img.Image = AttImg;

Img.BringToFront();

return true;

}

}

}

return false;

}

}

}

*Класс Rabbit.cs*

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace ВолкииЁлки

{

// класс для кролика

public class Rabbit : Animal

{

Form MainWindow;

static Random rnd = new Random();

// сколько очков размножения нужно для начала размножения

int requiredReproducePoints = 20;

// максимальное количество очков здоровья

int maxHP = 200;

public Rabbit(int x, int y, Form MainWindow) : base(x, y)

{

if (rnd.Next(0, 2) == 0)

{

Gender = true;

}

else

{

Gender = false;

}

Img.Image = Image.FromFile("зайка.png");

HP = maxHP;

this.MainWindow = MainWindow;

}

public override bool Reproduce()

{

// оглядываемся вокруг и ищем партнёра для размножения

// возможность размножаться появляется только после 4 шагов

// аналогично зайцам, но с учётом пола

if (ReproducePoints > requiredReproducePoints)

{

foreach (Rabbit r in Main.rabbits)

{

//не считая самого себя

if (r != this)

{

if ((Math.Abs(Img.Location.X - r.Img.Location.X) <= Main.cellSize && Math.Abs(Img.Location.Y - r.Img.Location.Y) <= Main.cellSize) && r.ReproducePoints > requiredReproducePoints && this.Gender != r.Gender) // разнополые особи

{

// перемещаем волка на клетку партнёра для начала процесса

this.Img.Location = new Point(r.Img.Location.X, r.Img.Location.Y);

// создаём нового зайца в той же клетке

Rabbit child = new Rabbit((r.Img.Location.X - Main.marginLeft) / Main.cellSize, (r.Img.Location.Y - Main.marginTop) / Main.cellSize, MainWindow);

Main.rabbits.Add(child);

MainWindow.Controls.Add(child.Img);

child.Img.BringToFront();

child.Img.Image = reprImg;

// оба временно теряют желание размножаться

this.ReproducePoints = 0;

r.ReproducePoints = 0;

return true;

}

}

}

}

return false;

}

//от ходов тоже стареет,

//зайчику неважно где волк, лишь бы не съел, но если видит зайчика другого пола идет

public override void Move()

{

// 0: - ; 1: стой на месте; 2: +

int verticalShift = rnd.Next(0, 3);

int horizontalShift = rnd.Next(0, 3);

// смещение по вертикали

//чтобы не вышел за пределы таблицы надо проверить положение относительно её границ

if (verticalShift == 0 && Img.Location.Y > Main.marginTop) Img.Location = new Point(Img.Location.X, Img.Location.Y - Main.cellSize);

else if (verticalShift == 2 && Img.Location.Y < Main.marginTop + (Main.rows - 1) \* Main.cellSize) Img.Location = new Point(Img.Location.X, Img.Location.Y + Main.cellSize);

// смещение по горизонтали

if (horizontalShift == 0 && Img.Location.X > Main.marginLeft) Img.Location = new Point(Img.Location.X - Main.cellSize, Img.Location.Y);

else if (horizontalShift == 2 && Img.Location.X < Main.marginLeft + (Main.cols - 1) \* Main.cellSize) Img.Location = new Point(Img.Location.X + Main.cellSize, Img.Location.Y);

// зайчик стареет, очень сильно страдает из-за движения, и его жизнь уменьшается

HP--;

if (HP > maxHP \* 0.6) Img.BackColor = Color.Green;

else if (HP > maxHP \* 0.3) Img.BackColor = Color.Yellow;//khaki

else if (HP>10)Img.BackColor = Color.Tomato;

else Img.BackColor = Color.Black;

if (Img.Image != rabImg)

{

Img.Image = rabImg;

}

// при движении желание спариваться возрастает

ReproducePoints++;

}

}

}

1. Siberian Forest [Электронный ресурс]//Habr.com [Сайт] URL:  https://habr.com/post/336330/ (дата обращения: 04.04.2018). [↑](#footnote-ref-1)
2. Конуэ [Электронный ресурс]//ConwayLife [Сайт] URL:  http://www.conwaylife.com/ (дата обращения: 04.04.2018). [↑](#footnote-ref-2)
3. Имитационная модели «хищник-жертва» [Электронный ресурс]//App Store [Магазин приложений] URL:  https://appsto.re/ru/bcFpab.i (дата обращения: 04.04.2018). [↑](#footnote-ref-3)