Министерство науки и высшего образования Российской Федерации

Рязанский государственный радиотехнический университет  
им. В.Ф. Уткина

Кафедра «Вычислительная и прикладная математика» (ВПМ)

**Пояснительная записка**

**к курсовому проекту**

на тему:

«Проектирование и создание игры «Agario»»

по курсу

**«Конструирование программного обеспечения»**

Выполнил:

Проверил:

доцент кафедры ВПМ

Столчнев В. К.

Рязань, 2023

Содержание

[Введение 4](#_Toc128498138)

[1 Анализ задачи 5](#_Toc128498139)

[1.1 Разработка иерархии классов 5](#_Toc128498140)

[1.1.1 Выделение сущностей 8](#_Toc128498141)

[1.1.2 Зависимости между классами. Диаграмма классов 9](#_Toc128498142)

[1.2 Алгоритмы 10](#_Toc128498143)

[1.2.1 Алгоритм при переходе в «Играть» 10](#_Toc128498144)

[1.2.2 Алгоритм для проверки наложения объектов 11](#_Toc128498145)

[1.3 Разработка интерфейса программы 14](#_Toc128498146)

[1.3.1 Интерфейс главного меню 14](#_Toc128498147)

[1.3.2 Интерфейс игры 15](#_Toc128498148)

[1.3.3 Интерфейс игры после поражения 18](#_Toc128498149)

[1.3.4 Интерфейс «Об игре» 19](#_Toc128498150)

[1.3.5 Интерфейс «Рекорды» 19](#_Toc128498151)

[2 Написание программы 20](#_Toc128498152)

[2.1 Описание разработанных процедур и функций 20](#_Toc128498153)

[2.2 Разработка программы 33](#_Toc128498154)

[2.2.1 Описание классов, перечислений и интерфейсов проекта 33](#_Toc128498155)

[2.3 Описание шаблонов проектирования, которые использовались при написании программы 37](#_Toc128498156)

[2.3.1 Модель-Представление-Контроллер (Model-View-Controller (MVC)) 37](#_Toc128498157)

[2.3.2 Одиночка (Singleton) 37](#_Toc128498158)

[2.3.3 Заместитель (Proxy) 38](#_Toc128498159)

[2.4 Описание методов рефакторинга, которые использовались при оптимизации исходного кода программы 38](#_Toc128498160)

[2.4.1 Пересмотр путей взаимодействия компонентов приложения 38](#_Toc128498161)

[2.4.2 Более явное разделение функионала и обязанностей у частей MVC 39](#_Toc128498162)

[2.4.3 Перенос логически связанной функциональности ближе к компонентам, которые её используют 39](#_Toc128498163)

[2.4.4 Объединение кода из отдельных функций внутрь свойства 39](#_Toc128498164)

[2.4.5 Замена повторяющегося кода наследованием виртуальных функций и полиморфизмом 39](#_Toc128498165)

[2.5 Разработка тестов 40](#_Toc128498166)

[2.5.1 Test Cases 40](#_Toc128498167)

[2.5.2 Модульные тесты 42](#_Toc128498168)

[3 Результат работы программы 43](#_Toc128498169)

[3.1 Консольное приложение 43](#_Toc128498170)

[3.1.1 Главное меню 43](#_Toc128498171)

[3.1.2 Начало игры 44](#_Toc128498172)

[3.1.3 Завершение игры 45](#_Toc128498173)

[3.1.4 Рекорды 45](#_Toc128498174)

[3.1.5 Об игре 46](#_Toc128498175)

[3.2 WPF-версия 46](#_Toc128498176)

[3.2.1 Главное меню 47](#_Toc128498177)

[3.2.2 Начало игры 47](#_Toc128498178)

[3.2.3 Завершение игры 50](#_Toc128498179)

[3.2.4 Рекорды 50](#_Toc128498180)

[3.2.5 Об игре 51](#_Toc128498181)

[Заключение 52](#_Toc128498182)

Введение

В качестве темы для этой курсовой работы была выбрана игра «Agario».

Цель данного проекта состоит в разработке и реализации игры, аналогичной по механике Agar.io, которая позволит игрокам управлять круглыми клетками, увеличивая их массу путем поглощения других клеток и частиц еды. Основной упор делается на динамичный и соревновательный игровой процесс, в котором игроки стремятся достичь наивысшего места в таблице лидеров.

Приложение должно иметь возможность запуска в 2 режимах:

• графическая версия, реализованная средствами платформы WPF

• консольная версия

Общая структура приложения должна соответствовать шаблону MVC.

# Анализ задачи

Необходимо спроектировать игру, похожую на «Agario» и реализовать её в двух разных версиях (консольный и графический режим) так, чтобы алгоритмы и механики игры подключались к ним в виде библиотеки, которая была бы общей для них. Процесс игры проходит так: игрок контролирует клетку, передвигая её по двумерному игровому полю. Для роста клетки необходимо поглощать как еду — маленькие гранулы, разбросанные по игровому полю, так и клетки других игроков, которые должны быть как минимум на 10 % меньше поглощающей клетки.

Клетка может делиться. Эту возможность можно использовать в качестве дальней атаки, чтобы съесть другие, более маленькие клетки, а также как способ побега. Разделённые клетки со временем снова сливаются в одну.

Во время игровой сессии игроки должны видеть таблицу лидеров, демонстрирующую текущее состояние рейтинга игроков.

Скорость клетки должна иметь обратную зависимость от её массы: более маленькие клетки двигаются быстрее крупных.

После того, как клетки игрока съедены, игрок возрождается на поле в случайном месте через некоторое время. Также должна быть реализована возможность установки и снятия паузы, выхода из игровой сессии. Если во время игровой сессии игрок достиг такого рейтинга, который может быть помещён в глобальную таблицу лидеров, то это значение сохраняется туда с текущим ником игрока без подтверждения.

В приложении должно быть реализовано меню, из которого доступны такие возможности:

1. Запуск игры.

2. Просмотр информации об игре.

3. Просмотр таблицы лидеров.

4. Изменение имени игрока.

5. Выход.

## Разработка иерархии классов

Общая структура приложения должна соответствовать шаблону MVC (Model-View-Controller). Пример реализации представлен на рисунке 1.

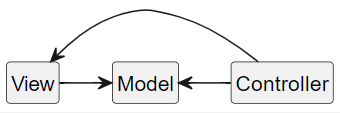


Рисунок – Применение шаблона MVC при проектировании игры «Agario».

Контроллер представляет собой средство для передачи управляющих команд пользователя в приложение (взаимодействие с меню; управление клетками). Представление – это механизм, демонстрирующий внутреннее состояние модели пользователю. Конкретные реализации контроллера и модели могут быть специфичными для каждого режима работы приложения. Моделью является внутренняя игровая логика. Она архитектурно не зависит от используемого режима работы и не «знает» о реализациях, использующих её, а только предоставляет API для изменения и получения своего состояния.

**Контроллеры** представляют собой средство для передачи управляющих команд пользователя в игру. **Представление** – это механизм, демонстрирующий внутреннее состояние модели пользователю. Конкретные реализации контроллера и представления могут быть специфичными для каждого режима работы приложения. **Моделью** является внутренняя игровая логика или механизм взаимодействия с меню. Она архитектурно не зависит от используемого режима работы и не «знает» о реализациях, использующих её, а только предоставляет API для изменения и получения своего состояния.

**AgarioModel** – реализация внутренней игровой логики. Представляет собой набор программных компонентов, необходимых для работы процессов игры и взаимодействия её объектов.

**ControllerGame** – компонент «Контроллер» шаблона MVC, независимый от реализации. Предоставляет API действий пользователя для влияния на состояние игры после её запуска. Может быть реализован как абстрактный полиморфный тип, конкретные реализации которого будут специфичными для каждого режима работы приложения и использовать механизм виртуальных функций.

**ViewGame** – компонент «Представление» шаблона MVC, независимый от реализации. Предоставляет минимальный контракт функциональности отображения состояния модели, которому должны следовать конкретные реализации «Представлений».

**ControllerGame(WPF/Console)** – специфичные для конкретной реализации приложения **ControllerGame’ы** шаблона MVC.

**ViewGame(WPF/Console)** – конкретные реализации **ViewGame** шаблона MVC. Отображают игроку состояние игры.

Аналогичные элементы диаграммы, содержащие **Menu** в своём названии, имеют сходное назначение, но предназначаются для игрового меню, так как оно тоже должно быть сделано с использованием шаблона MVC.

Как видно из диаграммы, компоненты моделей являются общими для обеих реализаций и не зависит от режима, в котором используется. Контроллеры зависят не только от модели, но также и от представлений, поскольку будут создавать также и их экземпляры для связывания с моделью при запуске. Так как из контроллера меню будет запускаться игровая логика, он связан с контроллером игры ассоциативной связью.

На рисунке 2 приведена упрощённая структура компонентов внутренней игровой логики.

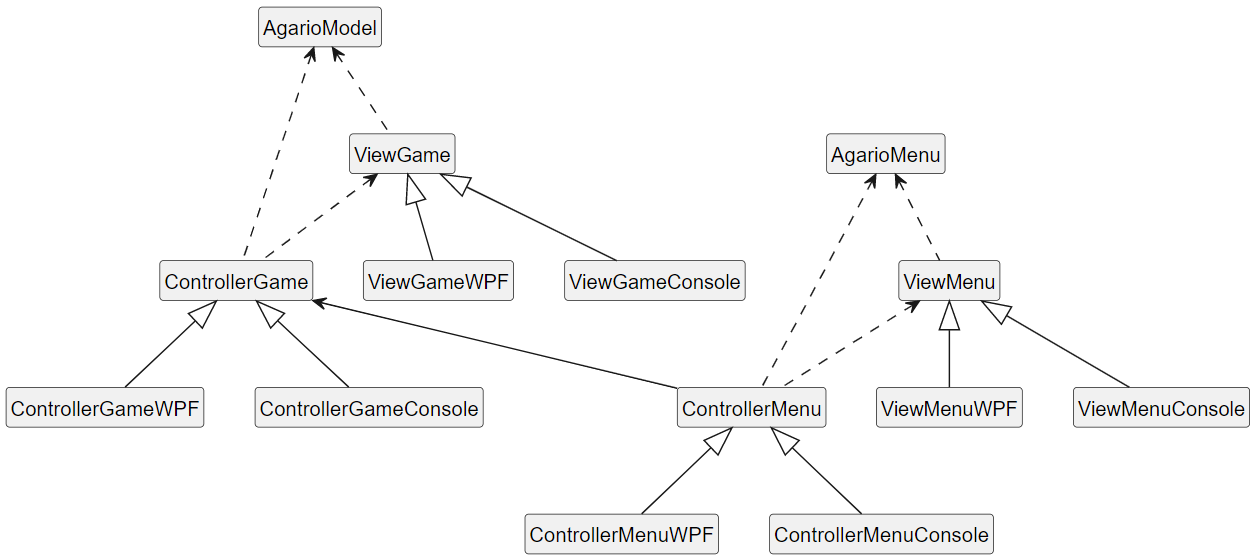


Рисунок – Компоненты внутренней игровой логики.

### Выделение сущностей

Общие игровые сущности:

1. Клетка (еда): окружность на плоскости, имеющая определённое положение, цвет, массу и радиус.

2. Движущаяся клетка: клетка, обладающая свойством скорости и возможностью изменять своё положение.

3. Игрок: сущность, обладающая именем и рейтингом, которая владеет набором движущихся клеток и управляет ими. Клетки одного игрока имеют один и тот же цвет.

Прочие сущности:

1. Меню: список пунктов меню

2. Правила: текст с игровой информацией

3. Рекорды: имя игрока, очки

4. Окно окончания игры: окно, появляющееся при возможности сохранить очки игрока в рекордах

5. Пункт меню: текст, состояние пункта меню

### Зависимости между классами. Диаграмма классов

На следующем рисунке представлена диаграмма классов модели.

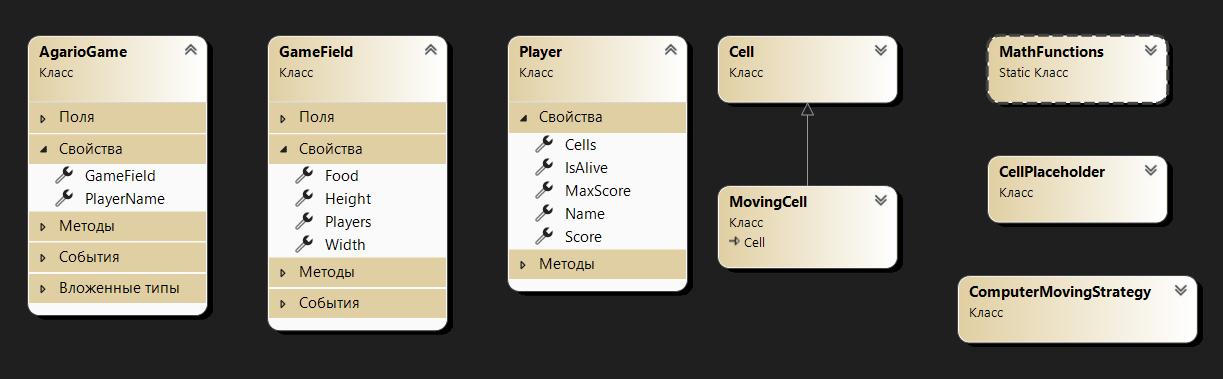


Рисунок – Диаграмма классов игровой логики.

На ней представлены основные компоненты, которые составляют игровую логику и обеспечивают работу механик игры.

## Алгоритмы

### Алгоритм при переходе в «Играть»

Диаграмма последовательности стандартного игрового процесса, представленная на рисунке 5. Данная диаграмма показывает взаимодействие пользователя с игровым приложением.

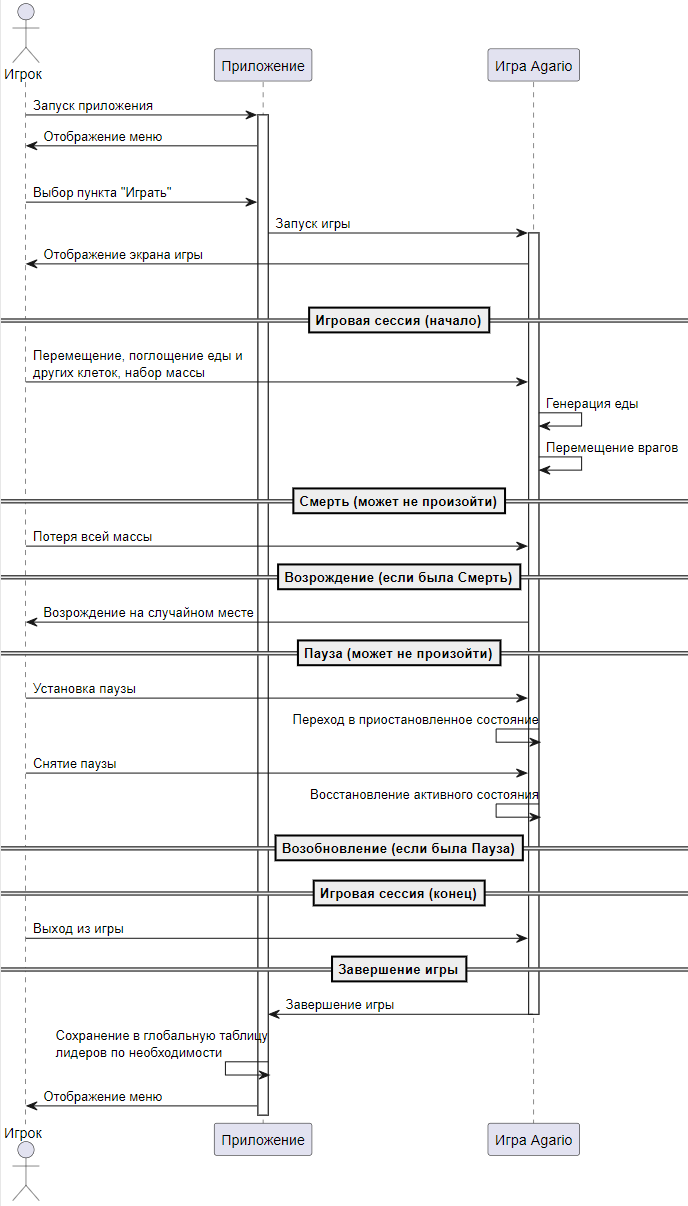


Рисунок 4 – Диаграмма последовательности при взаимодействии с игрой.

На диаграмме представлен процесс запуска игроком приложения, выбор пункта меню «Играть», после которого приложение запускает игровую логику. Пользователь играет в игру, возможно, переводя её в приостановленное состояние. Игрок умирает после потери массы всех своих клеток и возрождается в случайном месте. Пока идёт игровой процесс, приложение генерирует еду и обновляет положение врагов. При выходе в главное меню приостанавливается работа игровой логики, а приложение выполняет обновление глобальной таблицы лидеров и отображает игроку игровое меню.

### Алгоритм поведения врагов, управляемых программой

При отсутствии других врагов вокруг клетка движется в сторону ближайшей еды или меньшей по размеру клетки, поглощая её. Если в процессе движения она замечает другую, большую по размеру клетку, на некотором расстоянии DANGER\_CELL\_DISTANCE в зоне поиска DANGER\_SEARCH\_ANGLE, или приближается к стене WALL\_REVERSE\_DISTANCE, то она меняет направление на противоположное с некоторым отклонением от прямо противоположного направления, выбираемым случайно. Далее она может снова двигаться в сторону ближайшей в направлении движения еды в зоне углового сегмента EAT\_SEARCH\_ANGLE.

Блок-схема на рисунке ниже.

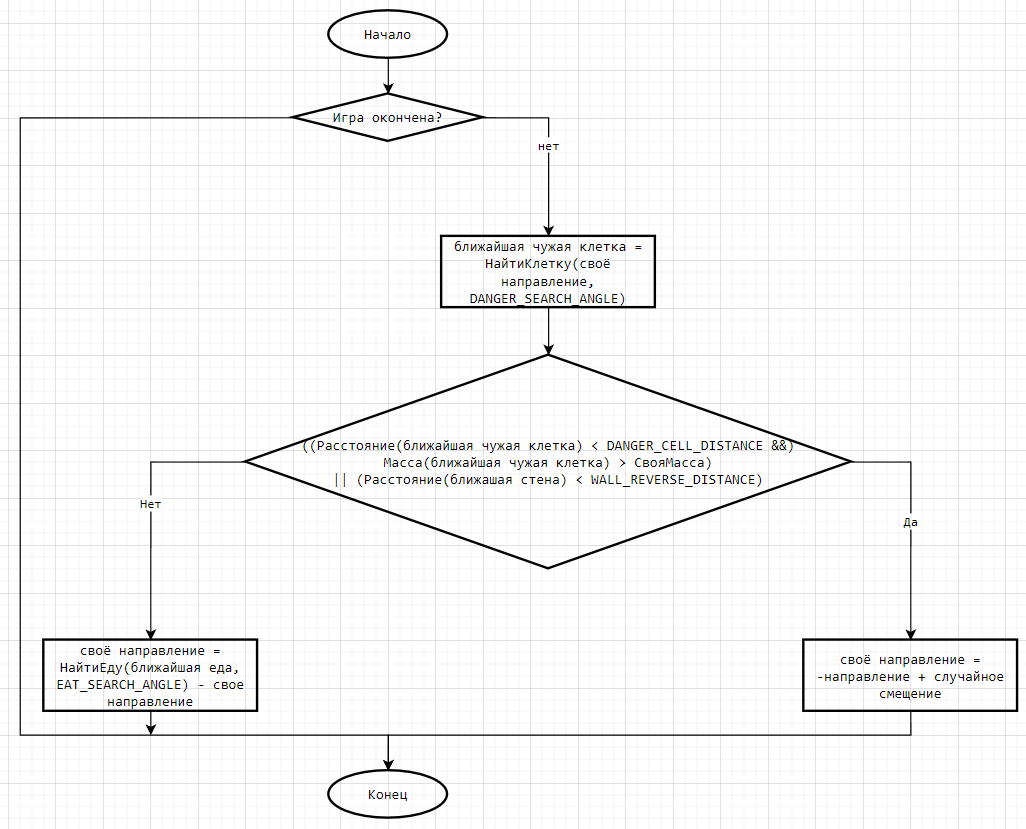


Рисунок – Алгоритм движения управляемых компьютером игроков.

### Алгоритм генерации еды

При запуске игры на поле генерируются первые 1000 клеток еды со случайным положением и случайной массой от 1 до 6. Далее до завершения игры каждые 3 секунды на поле появляются 50 новых клеток еды по такому же принципу. Генерация новой еды не зависит от количества существующей еды на поле. Блок-схема на следующем рисунке.

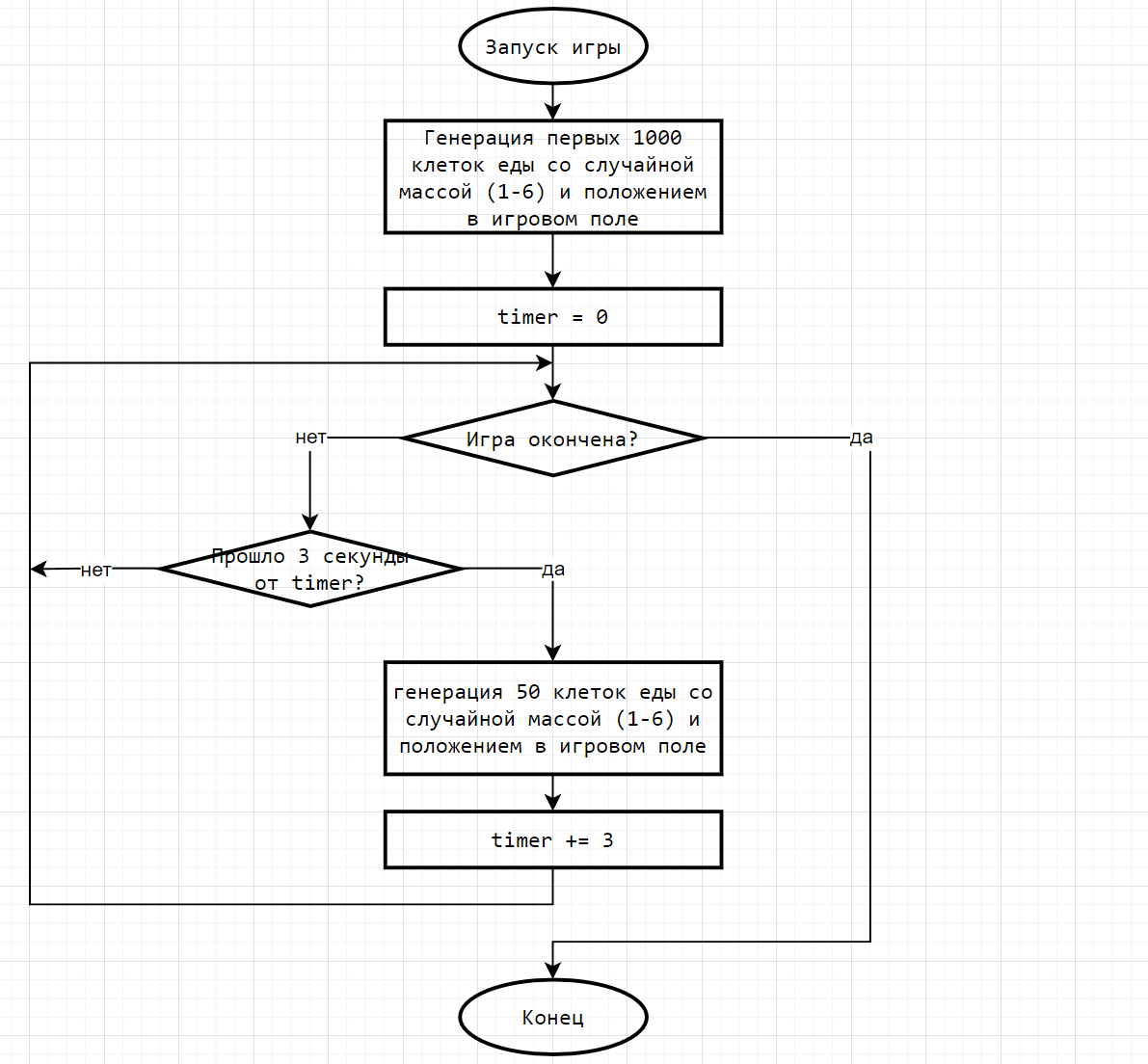


Рисунок – Блок-схема алгоритма генерации еды.

## Разработка интерфейса программы

Интерфейс должен быть реализован в двух реализациях интерфейса: консольное приложение и WPF – полностью графический интерфейс.

### Интерфейс главного меню

Пример дизайна меню в WPF-версии представлен на рисунке ниже.

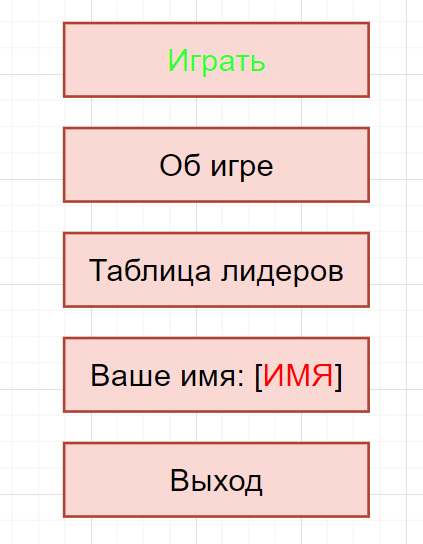


Рисунок – Дизайн меню WPF-версии.

Пример меню в консоли представлен на следующем рисунке.

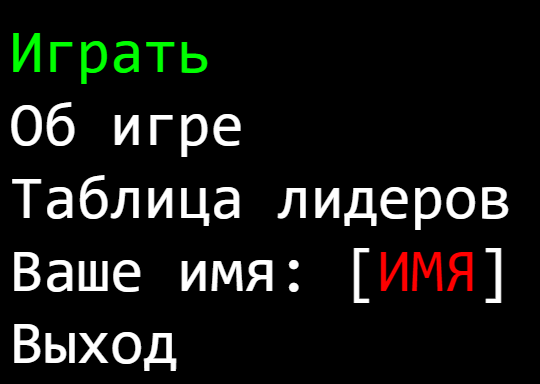


Рисунок – Дизайн меню в консоли

Пункты меню:

* *Играть*;
* *Об* *игре*;
* *Таблица лидеров*;
* *Имя игрока;*
* *Выход*.

Каждый пункт меню могут быть в двух визуальных состояниях:

* в фокусе;
* не выбран.

По умолчанию при открытии меню пункт «*Играть*» всегда находится в состоянии фокуса, а остальные неактивны. Перемещение по пунктам меню (смещение состояния «в фокусе») происходит с помощью клавиш со стрелками, выбор – по клавише «пробел» или «enter». При выборе происходит мгновенное переключение на соответствующий экран (или закрытие игры в случае, если был выбран пункт «*Выход*»), кроме пункта «*Имя игрока*». При нахождении в фокусе пункта «*Имя игрока*» нажатием символьных или цифровых клавиш, а также Backspace можно изменить текущий ник игрока. Клавиша Enter подтверждает сохранение, Escape – отмена редактирования.

### Интерфейс игры

При выборе пункта «Играть» происходит переход приложения в состояние игровой сессии. В состоянии игровой сессии возможны установка и снятие паузы клавишей P (английская). Также должна отображаться таблица лидеров сессии. Игровое поле может иметь большие размеры, чем отображается на экране игрока. Размер участка игрового поля, который виден в кадре, зависит от массы игрока (игрок должен помещаться в кадр целиком, но не быть при этом слишком мелким). Управление клеткой происходит с помощью мышки (клетка движется в направлении вектора между центром и курсором мыши со скоростью, которая зависит от расстояния до курсора), разделение осуществляется по нажатию на пробел.

Примерный вид игрового поля в WPF-режиме представлен на следующем рисунке.

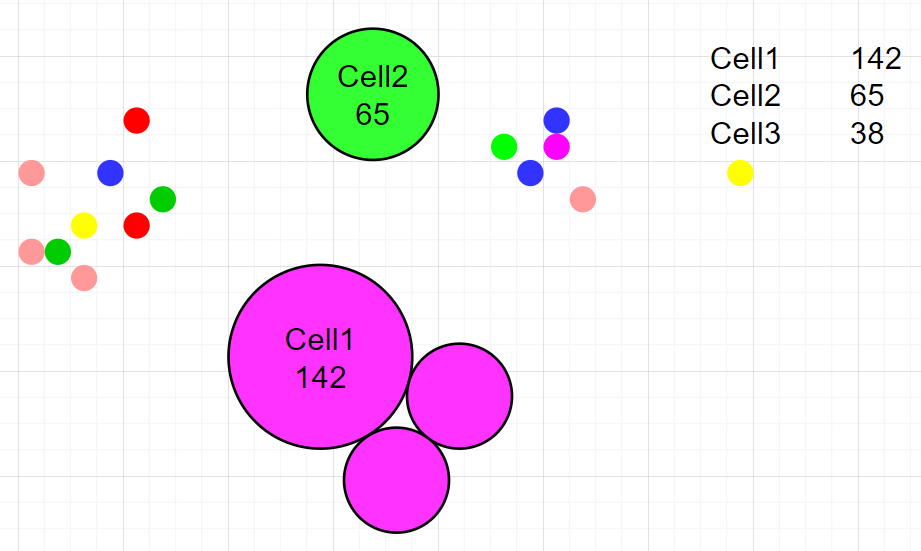


Рисунок 9 – пример вида игрового поля в WPF-версии

На рисунке изображены клетки Cell1 с рейтингом 142, разделённая на 3 клетки, и Cell2 с рейтингом 65. Клетка Cell3 не попадает в кадр. В правом верхнем углу выводится таблица лидеров сессии. Разноцветные мелкие кружки – еда.

Примерный вид игрового поля в консольном режиме представлен на следующем рисунке.

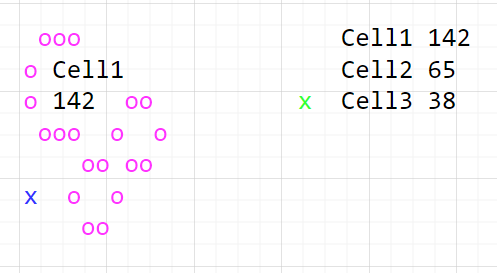


Рисунок 10 – игровое поле в консоли

На этом рисунке видна только клетка Cell1. Разноцветные x – еда. В правом верхнем углу выводится таблица лидеров сессии.

### Интерфейс «Об игре»

Кнопками со стрелками можно перематывать текст, если он не укладывается в экран, а кнопка «Назад» возвращает обратно в главное меню. При отображении в случае, если текст не помещается в экран, отображается номер текущей страницы текста.

Экран с информацией об игре в WPF-версии:

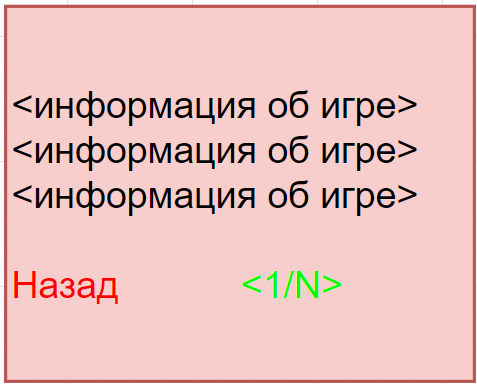


Рисунок 11 – экран информации об игре в WPF-версии

Дизайн экрана с информацией об игре в консоли представлен на следующем рисунке.

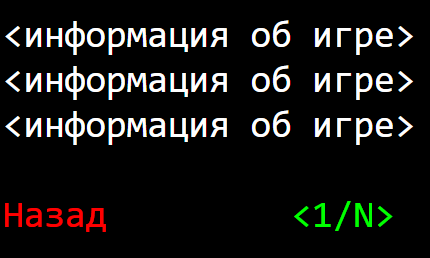


Рисунок 12 – экран с информацией об игре в консоли

Описание относится к экрану с информацией об игре в обеих версиях игры. На этом экране отображается текст информации об игре (вместо <информация об игре>). Если текст настолько большой, что не помещается на один экран, то он разбивается на страницы, перелистывание которых происходит по нажатию клавиш со стрелочками «влево» и «вправо». Буква N в надписи <1/N> заменяется на общее число страниц, цифра 1 в этой надписи заменяется на номер текущей страницы. Доступные действия: возврат в меню по клавише Enter, Space или Escape, прокрутка страниц в большую сторону по клавише «стрелка вправо», прокрутка страниц в меньшую сторону по клавише «стрелка влево».

### Интерфейс «Таблица лидеров»

Глобальная таблица лидеров конечна – отображаются 10 лучших. Её сброс изнутри игры не предусмотрен.

Дизайн экрана глобальной таблицы лидеров в WPF-версии:

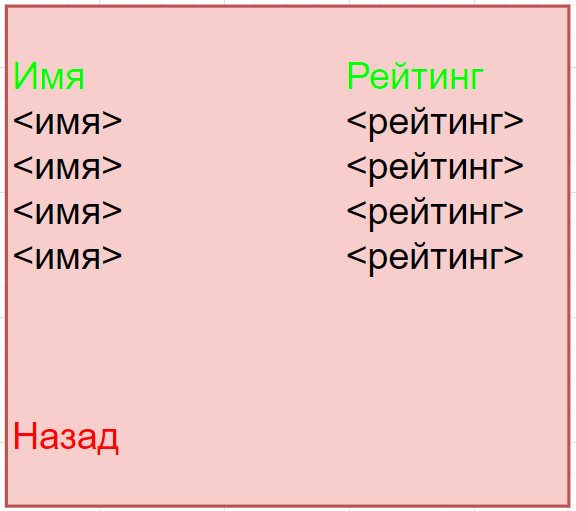


Рисунок 13 – таблица лидеров в WPF-версии

Дизайн глобальной таблицы лидеров в консольном режиме:

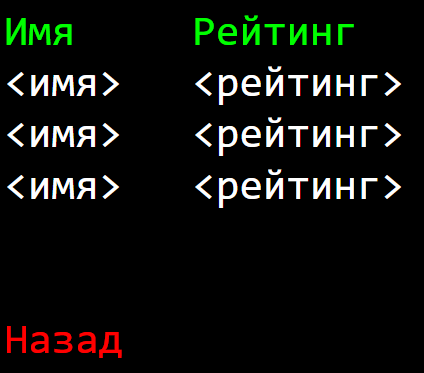


Рисунок 14 – дизайн таблицы лидеров в консоли

Описание относится к экрану таблицы лидеров в обеих версиях игры. На этом экране отображается список имён игроков (вместо <имя>) со значением максимального рейтинга (вместо <рейтинг>), который они когда-либо достигали. Доступные действия: возврат в меню по клавише Enter, Space или Escape.

### Интерфейс «Имя игрока»

При переходе по этому пункту игроку предоставляется возможность задать имя, под которым он будет отображаться на поле и под которым будут сохраняться рекорды. При нажатии символьных или цифровых клавиш происходит изменение текущего имени игрока. По нажатию Enter замена подтверждается, нажатие Escape отменяет изменение и выключает режим редактирования.

# Написание программы

## Описание разработанных процедур и функций

**Таблица 1**

|  |  |  |
| --- | --- | --- |
| **Сигнатура** | **Описание** | **Параметры** |
| AgarioModels.Game.AgarioGame.#ctor | Инициализация экземпляра игры. Модификатор private для реализации шаблона singleton |  |
| AgarioModels.Game.AgarioGame.GetGameInstance | Получение экземпляра игры |  |
| AgarioModels.Game.AgarioGame.AddComputerControlledPlayer(System.Int32) | Добавление игрока, управляемого компьютером | parNumber - Номер |
| AgarioModels.Game.AgarioGame.InitializeGameField | Инициализация игрового поля |  |
| AgarioModels.Game.AgarioGame.StartGame | Запуск игры |  |
| AgarioModels.Game.AgarioGame.Pause | Приостановка игры |  |
| AgarioModels.Game.AgarioGame.Resume | Возобновление игры |  |
| AgarioModels.Game.AgarioGame.Stop | Остановка игры |  |
| AgarioModels.Game.AgarioGame.GetGameTicker | Получение потока, обновляющего внутреннее время и состояние игры |  |
| AgarioModels.Game.Cell.#ctor | Конструктор по умолчанию |  |
| AgarioModels.Game.Cell.#ctor(AgarioModels.Game.Cell) | Конструктор копирования | parCell - Клетка, по данным которой будет инициализирован объект |
| AgarioModels.Game.Cell.IsIntersect(AgarioModels.Game.Cell) | Проверка пересечения с клеткой | parCell - Проверяемая клетка |
| AgarioModels.Game.CellPlaceholder.#ctor(AgarioModels.Game.GameField) | Инициализация | parGameField - Игровое поле |
| AgarioModels.Game.CellPlaceholder.SetRandomPosition(AgarioModels.Game.Cell) | Устанавливает случайное положение на игровом поле для переданной клетки так, чтобы она не перекрывалась существующей на игровом поле едой или другими игроками | parCell - Клетка, которой будет задано случайное положение |
| AgarioModels.Game.ComputerMovingStrategy.#ctor(AgarioModels.Game.Player,AgarioModels.Game.GameField) | Инициализация | parPlayer - Управляемый игрок,parGameField - Игровое поле |
| AgarioModels.Game.ComputerMovingStrategy.CatchPlayer(AgarioModels.Game.Player) | Догнать игрока | parPlayer - Игрок |
| AgarioModels.Game.ComputerMovingStrategy.EscapeFromPlayer(AgarioModels.Game.Player) | Сбежать от игрока | parPlayer - Игрок |
| AgarioModels.Game.ComputerMovingStrategy.GetSpeedByClosestEat | Вычисление вектора скорости, чтобы двигаться к ближайшей еде |  |
| AgarioModels.Game.ComputerMovingStrategy.GetSpeedByOtherPlayersStates(System.Numerics.Vector2) | Оценивая положение и состояние других игроков, по необходимости вычисляет новый вектор скорости или возвращает | parCurrentSpeedVector - Расчитанный вектор скорости, который вернётся, если состояние других игроков не является существенным для изменения скорости |
| AgarioModels.Game.ComputerMovingStrategy.RecalculateSpeedAndDirection | Перерасчёт направления движения |  |
| AgarioModels.Game.ComputerMovingStrategy.Update(System.Single) | Обновление, перерасчёт параметров | parDeltaTime - Изменение внутреннего времени в секундах |
| AgarioModels.Game.GameField.#ctor | Инициализация игрового поля |  |
| AgarioModels.Game.GameField.InitializePlayer(AgarioModels.Game.Player) | Инициализация и размещение игрока на поле | parPlayer - |
| AgarioModels.Game.GameField.AddPlayerOnRandomPosition(AgarioModels.Game.Player,AgarioModels.Game.ComputerMovingStrategy) | Добавление игрока в случайное место на поле | parPlayer - Добавляемый игрок |
| AgarioModels.Game.GameField.UpdatePlayerPosition(AgarioModels.Game.Player,System.Single) | Обновление положения игрока | parPlayer - Обновляемый игрок,parDeltaTime - Временной интервал, на который надо сделать интерполяцию |
| AgarioModels.Game.GameField.UpdateEat(System.Single) | Генерация новой еды через некоторый интервал времени | parDeltaTime - Временной интервал, прошедший с момента последнего обновления |
| AgarioModels.Game.GameField.RebornDeadPlayer(AgarioModels.Game.Player) | Возрождение умершего игрока | parPlayer - |
| AgarioModels.Game.GameField.RebornDeadPlayers(System.Single) | Возрождение умерших игроков через некоторый интервал времени | parDeltaTime - Временной интервал, прошедший с момента последнего обновления |
| AgarioModels.Game.GameField.UpdateComputerControlledPlayers(System.Single) | Обновление состояния игроков, управляемых компьютером | parDeltaTime - Изменение внутреннего игрового времени в секундах |
| AgarioModels.Game.GameField.UpdateMaxScore(System.Single) | Обновление максимального счёта | parDeltaTime - Изменение внутреннего игрового времени в секундах |
| AgarioModels.Game.GameField.Update(System.Single) | Обновление состояния игры | parDeltaTime - Изменение внутреннего игрового времени в секундах |
| AgarioModels.Game.GameField.ShuffleSingleCells | Перемещает все клетки (еду и игроков) на новое случайное место на поле. Вызов функции стоит делать только после того, как будут удалены лишние клетки игроков |  |
| AgarioModels.Game.GameField.ResetPlayerCells(AgarioModels.Game.Player) | Оставляет игроку одну клетку с начальной массой | parPlayer - Игрок |
| AgarioModels.Game.GameField.ResetEat | Возвращает поле к состоянию с начальным количеством еды |  |
| AgarioModels.Game.GameField.Reset | Сброс состояния игрового поля |  |
| AgarioModels.Game.GameField.CalculateRealSpeedVector(System.Numerics.Vector2) | Расчёт реальной устанавливаемой скорости клетки по вектору желаемой скорости | parDesiredSpeedVector - |
| AgarioModels.Game.GameField.FitSpeedByCellWeight(System.Numerics.Vector2,System.Single) | Подстройка скорости в зависимости от массы клетки | parSpeedVector - Вектор скорости,parWeight - Масса клетки |
| AgarioModels.Game.GameField.SetSpeedVectorForCellsMutualAttraction(AgarioModels.Game.Player,System.Numerics.Vector2) | Взаимное притяжение разделённых клеток игрока. Не даёт клеткам разбегаться, мягко подталкивая друг к другу через изменение направления вектора скорости | parPlayer - Игрок,speedVector - Вектор желаемой скорости |
| AgarioModels.Game.GameField.CorrectSpeedByWallCollisions(AgarioModels.Game.MovingCell,System.Numerics.Vector2) | Скорректированный вектор скорости для учёта остановки движения клетки при столкновении со стенами | parCell - Клетка,parSpeed - Скорость до преобразования |
| AgarioModels.Game.GameField.SetSpeedToPlayer(AgarioModels.Game.Player,System.Numerics.Vector2) | Установка скорости для игрока | parPlayer - Игрок,speedVector - Вектор желаемой скорости. При превышении максимального ограничения будет смасштабирован до соответствия ему |
| AgarioModels.Game.GameField.CreateEat(System.Int32) | Создаёт клеток еды в случайных местах поля | parCount - Количество еды |
| AgarioModels.Game.GameField.PlayerEatsFood(AgarioModels.Game.Player,AgarioModels.Game.MovingCell,AgarioModels.Game.Cell) | Обработка съедания клеткой игрока еды | parPlayer - ,parCell - ,parEat - |
| AgarioModels.Game.GameField.PlayerEatsOtherCell(AgarioModels.Game.Player,AgarioModels.Game.MovingCell,AgarioModels.Game.Player,AgarioModels.Game.MovingCell) | Обработка съедания клеткой игрока клетки другого игрока | parPlayer - ,parPlayerCell - ,parOtherPlayer - ,parEatedPlayerCell - |
| AgarioModels.Game.GameField.ProcessEatingOtherPlayer(AgarioModels.Game.Player,AgarioModels.Game.Player) | Обработка съедания игроком клеток другого игрока | parPlayer - ,parOtherEatedPlayer - |
| AgarioModels.Game.GameField.ProcessCellsMerging(AgarioModels.Game.Player) | Обработка слияния разделённых ячеек одного игрока | parPlayer - |
| AgarioModels.Game.GameField.ProcessEatingOtherCellsForPlayer(AgarioModels.Game.Player) | Обработка съедания игроком клеток других игроков | parPlayer - |
| AgarioModels.Game.GameField.ProcessEatingFoodForPlayer(AgarioModels.Game.Player) | Обработка процесса съедания игроком еды | parPlayer - |
| AgarioModels.Game.GameField.ProcessEating | Обработка процесса съедания еды или клеток игроков |  |
| AgarioModels.Game.GameField.CalculateSpeedAndPositionAfterCellDivide(AgarioModels.Game.MovingCell,AgarioModels.Game.MovingCell) | Вычисляет положение и скорость ячеек после разделения | parDividedCell - Разделившаяся ячейка,parNewCell - Новая ячейка |
| AgarioModels.Game.GameField.DivideCell(AgarioModels.Game.MovingCell) | Разделяет клетку на 2. У меняется масса. Возвращает новую появившуюся ячейку | parDividedCell - Разделяемая ячейка |
| AgarioModels.Game.GameField.DividePlayerQuery(AgarioModels.Game.Player) | Запрос на разделение игрока . Фактическое разделение произойдет при обновлении состояния игры вызовом Update() | parPlayer - Игрок, который хочет разделиться |
| AgarioModels.Game.GameField.DividePlayer(AgarioModels.Game.Player) | Разделение игрока на части | parPlayer - Игрок |
| AgarioModels.Game.MathFunctions.Rectangle.IsIntersect(AgarioModels.Game.MathFunctions.Rectangle) | Проверка пересечения с другим прямоугольником | parOther - Другой прямоугольник |
| AgarioModels.Game.MathFunctions.Rectangle.IsIntersect(AgarioModels.Game.Cell) | Проверка пересечения с клеткой | parCell - Клетка |
| AgarioModels.Game.MathFunctions.Rectangle.IsNestedIn(AgarioModels.Game.MathFunctions.Rectangle) | Проверка, вложен этот прямоугольник в прямоугольник или нет | parOther - Прямоугольник, вложенность в который проверяется |
| AgarioModels.Game.MathFunctions.IsIntersect(AgarioModels.Game.MathFunctions.Rectangle,AgarioModels.Game.MathFunctions.Rectangle) | Проверка пересечения прямоугольников | parR1 - ,parR2 - |
| AgarioModels.Game.MathFunctions.IsIntersect(AgarioModels.Game.MathFunctions.Rectangle,AgarioModels.Game.Cell) | Проверка пересечения прямоугольника и клетки | parR - ,parCell - |
| AgarioModels.Game.MathFunctions.IsIntersect(AgarioModels.Game.Cell,AgarioModels.Game.Cell) | Проверка пересечения клеток | parCell1 - ,parCell2 - |
| AgarioModels.Game.MathFunctions.CalculateRadiusByMass(System.Int32) | Вычисление радиуса клетки заданной массы | parMass - |
| AgarioModels.Game.MathFunctions.GetRectangleCenter(AgarioModels.Game.MathFunctions.Rectangle) | Вычисляет центральную точку прямоугольника | parRectangle - |
| AgarioModels.Game.MathFunctions.DistanceBetweenCentersSquared(AgarioModels.Game.MathFunctions.Rectangle,AgarioModels.Game.MathFunctions.Rectangle) | Получение квадрата расстояния между центрами прямоугольников | parRectangle1 - ,parRectangle2 - |
| AgarioModels.Game.MathFunctions.Distance(AgarioModels.Game.Cell,AgarioModels.Game.Cell) | Расстояние между центрами клеток | parCell1 - ,parCell2 - |
| AgarioModels.Game.MathFunctions.Area(AgarioModels.Game.Cell) | Площадь клетки | parCell - |
| AgarioModels.Game.MathFunctions.IsNestedIn(AgarioModels.Game.Cell,AgarioModels.Game.Cell) | Проверка, вложена ли ячейка в | parCell1 - ,parCell2 - |
| AgarioModels.Game.MathFunctions.GetIntersectionArea(AgarioModels.Game.Cell,AgarioModels.Game.Cell) | Вычисление площади пересечения двух окружностей | parCell1 - ,parCell2 - |
| AgarioModels.Game.MathFunctions.CalculateMassCenter``1(System.Collections.Generic.List{``0}) | Поиск центра масс для группы ячеек | parCells - Группа ячеек |
| AgarioModels.Game.Player.GetBoundingRect | Получение минимального ограничивающего прямоугольника, включающего все клетки игрока |  |
| AgarioModels.Menu.AboutGame.DataReader.GetInformationAboutGame | Получает информацию об игре из ресурсов |  |
| AgarioModels.Menu.Menu.AddItem(AgarioModels.Menu.MenuItem) | Добавление элемента меню | parMenuItem - Элемент меню |
| AgarioModels.Menu.Menu.FocusPrevious | Смена фокуса на предыдущий элемент |  |
| AgarioModels.Menu.Menu.FocusNext | Смена фокуса на следующий элемент |  |
| AgarioModels.Menu.Menu.FocusByID(System.Int32) | Установка элемента в состоянии "В фокусе" | parID - ID элемента |
| AgarioModels.Menu.Menu.SelectFocusedElement | Активация элемента, на котором стоит фокус |  |
| AgarioModels.Menu.MenuItem.#ctor(System.Int32,System.String) | Создаёт элемент меню с заданным именем и ID | parID - ID элемента меню,parName - Имя элемента меню |
| AgarioModels.Menu.MenuMain.GetMenu | Получение главного меню |  |
| AgarioModels.Menu.Records.GameRecordsHandler.GetRecords | Получение набора рекордов из \_recordsDataIO |  |
| AgarioModels.Menu.Records.GameRecordsHandler.HandleRecordValueOnEndGame(System.Int32) | Обработка значения рекорда после окончания игры. Сохраняет рекорд по необходимости, запрашивая при этом имя игрока | parRecordValue - Значение рекорда, набранного игроком |
| AgarioModels.Menu.Records.Record.#ctor(System.String,System.Int32) | Создаёт рекорд с заданным именем и значением | parName - Имя,parValue - Значение |
| AgarioModels.Menu.Records.RecordsIO.DescendingRecordComparer.Compare(AgarioModels.Menu.Records.Record,AgarioModels.Menu.Records.Record) | Сравает 2 рекорда по убыванию значения | parX - Рекорд 1,parY - Рекорд 2 |
| AgarioModels.Menu.Records.RecordsIO.GetStoredRecords | Получение сохранённых рекордов из файловой системы |  |
| AgarioModels.Menu.Records.RecordsIO.IsNeedToStore(System.Int32) | Проверка необходимости сохранять рекорд. Возвращает true, если количество сохранённых рекордов меньше максимально разрешённого количества, или если значение рекорда превышает минимальное из уже сохранённых значений (независимо от их количества) | parRecordValue - Значение рекорда, которое проверяется |
| AgarioModels.Menu.Records.RecordsIO.SaveRecord(AgarioModels.Menu.Records.Record) | Сохранение рекорда. Обновляет рекорды в кэше, сортируя по убыванию, и сразу записывет их в файловую систему | parRecord - Сохраняемый рекорд |
| AgarioWPF.App.OnStartup(System.Windows.StartupEventArgs) | Обработка события Startup для запуска контроллера главного меню | e - Аргументы события |
| AgarioWPF.App.Main | Application Entry Point. |  |
| Controllers.Game.GameController.#ctor | Инициализация контроллера игры |  |
| Controllers.Game.GameController.UpdateRecordOnPlayerDead(AgarioModels.Game.Player) | Обновляет значение рекорда после смерти игрока | parPlayer - Игрок |
| Controllers.Game.GameController.UpdateRecordBeforeExit | Обновление максимального рейтинга в записях рекордов перед выходом из игры |  |
| Controllers.Game.GameController.SetHandlerPlayerPaused | Включение обработки установки паузы игроком |  |
| Controllers.Game.GameController.SetHandlerPlayerResumed | Включение обработки возобновления игры игроком |  |
| Controllers.Game.GameController.SetHandlerPlayerExited | Включение обработки выхода игрока |  |
| Controllers.Game.GameController.SetHandlerPlayerSpeedSet | Включание обработки управления скоростью клеток игрока |  |
| Controllers.Game.GameController.SetHandlerPlayerDivided | Включение обработки разделения игрока |  |
| Controllers.Game.GameController.ResetHandlerPlayerPaused | Выключение обработки установки паузы игроком |  |
| Controllers.Game.GameController.ResetHandlerPlayerResumed | Выключение обработки возобновления игры игроком |  |
| Controllers.Game.GameController.ResetHandlerPlayerExited | Выключение обработки выхода игрока |  |
| Controllers.Game.GameController.ResetHandlerPlayerSpeedSet | Выключение обработки управления скоростью клеток игрока |  |
| Controllers.Game.GameController.ResetHandlerPlayerDivided | Выключение обработки разделения игрока |  |
| Controllers.Game.GameController.CreateGameView | Получение представления игры |  |
| Controllers.Game.GameController.Start | Запуск игры |  |
| Controllers.Game.GameController.PauseGame | Приостановка игры |  |
| Controllers.Game.GameController.ResumeGame | Возобновление игры |  |
| Controllers.Game.GameController.StopGame | Остановка игры |  |
| Controllers.Game.GameController.SetPlayerSpeed(System.Numerics.Vector2) | Вызывает установку игроку, за которого отвечает контроллер, значения скорости, задаваемое вектором | parSpeed - Вектор скорости |
| Controllers.Game.GameController.DividePlayer | Разделение игрока |  |
| Controllers.Menu.MenuAboutGameController.ShowNextPage | Показать следующую страницу информации |  |
| Controllers.Menu.MenuAboutGameController.ShowPrevPage | Показать прошлую страницу информации |  |
| Controllers.Menu.MenuController.#ctor(AgarioModels.Menu.Menu) | Конструктор контроллера меню | parMenu - Меню, для которого создаётся контроллер |
| Controllers.Menu.MenuController.Start | Запуск меню |  |
| Controllers.Menu.MenuPlayerNameController.ChangePlayerName | Изменение имени игрока |  |
| Controllers.Menu.MenuScreenController.Start | Открытие раздела меню |  |
| Controllers.Menu.MenuScreenController.GoBackCall | Вызов события GoToBack |  |
| ControllersConsole.AboutGameControllerConsole.#ctor | Инициализация контроллера пункта меню, создание представления |  |
| ControllersConsole.AboutGameControllerConsole.ShowNextPage | Показать следующую страницу информации |  |
| ControllersConsole.AboutGameControllerConsole.ShowPrevPage | Показать прошлую страницу информации |  |
| ControllersConsole.AboutGameControllerConsole.Start | Открытие раздела меню "Об игре" |  |
| ControllersConsole.GameControllerConsole.#ctor | Инициализация контроллера игры, создание представления |  |
| ControllersConsole.GameControllerConsole.Start | Запуск игры |  |
| ControllersConsole.GameControllerConsole.CalculatePlayerScreenPosition | Возвращает положение центра игрока на экране |  |
| ControllersConsole.GameControllerConsole.PlayerSpeedUpdateHandler | Исполнитель действия по обновлению скорости игрока |  |
| ControllersConsole.GameControllerConsole.PlayerPauseCheck(System.ConsoleKeyInfo) | Обработка установки игроком паузы | parKeyInfo - Информация о нажатии |
| ControllersConsole.GameControllerConsole.PlayerResumeCheck(System.ConsoleKeyInfo) | Обработка возобновления игры игроком | parKeyInfo - Информация о нажатии |
| ControllersConsole.GameControllerConsole.PlayerExitCheck(System.ConsoleKeyInfo) | Обработка выхода игрока | parKeyInfo - Информация о нажатии |
| ControllersConsole.GameControllerConsole.PlayerDivideHandler(System.ConsoleKeyInfo) | Обработка разделения игрока | parKeyInfo - Информация о нажатии |
| ControllersConsole.GameControllerConsole.CreateGameView | Получение представления игры |  |
| ControllersConsole.GameControllerConsole.ResetHandlerPlayerDivided | Выключение обработки разделения игрока |  |
| ControllersConsole.GameControllerConsole.ResetHandlerPlayerExited | Выключение обработки выхода игрока |  |
| ControllersConsole.GameControllerConsole.ResetHandlerPlayerPaused | Выключение обработки установки паузы игроком |  |
| ControllersConsole.GameControllerConsole.ResetHandlerPlayerResumed | Выключение обработки возобновления игры игроком |  |
| ControllersConsole.GameControllerConsole.ResetHandlerPlayerSpeedSet | Выключение обработки управления скоростью клеток игрока |  |
| ControllersConsole.GameControllerConsole.SetHandlerPlayerDivided | Включение обработки разделения игрока |  |
| ControllersConsole.GameControllerConsole.SetHandlerPlayerExited | Включение обработки выхода игрока |  |
| ControllersConsole.GameControllerConsole.SetHandlerPlayerPaused | Включение обработки установки паузы игроком |  |
| ControllersConsole.GameControllerConsole.SetHandlerPlayerResumed | Включение обработки возобновления игры игроком |  |
| ControllersConsole.GameControllerConsole.SetHandlerPlayerSpeedSet | Включание обработки управления скоростью клеток игрока |  |
| ControllersConsole.MenuMainControllerConsole.#ctor(AgarioModels.Menu.Menu) | Инициализация контроллера главного меню, создание дочерних компонентов | parMenu - Меню |
| ControllersConsole.MenuMainControllerConsole.Start | Запуск меню |  |
| ControllersConsole.PlayerNameControllerConsole.#ctor | Инициализация |  |
| ControllersConsole.PlayerNameControllerConsole.Start | Открытие раздела меню |  |
| ControllersConsole.RecordsControllerConsole.Start | Запуск перехода на экран рекордов |  |
| ControllersWPF.AboutGameControllerWPF.#ctor(System.Windows.Window) | Создание раздела меню "Об игре" и его представления | parWindow - Окно приложения |
| ControllersWPF.AboutGameControllerWPF.Start | Открытие раздела меню "Об игре" |  |
| ControllersWPF.AboutGameControllerWPF.ConfigureView | Настройка представления |  |
| ControllersWPF.AboutGameControllerWPF.ShowNextPage | Показать следующую страницу информации |  |
| ControllersWPF.AboutGameControllerWPF.ShowPrevPage | Показать прошлую страницу информации |  |
| ControllersWPF.GameControllerWPF.#ctor(System.Windows.Window) | Инициализация игрового контроллера и создание представления | parGameWindow - Окно игры |
| ControllersWPF.GameControllerWPF.CreateGameView | Получение представления игры |  |
| ControllersWPF.GameControllerWPF.PlayerExitCheck(System.Object,System.Windows.Input.KeyEventArgs) | Проверка выхода игрока | parSender - ,parKeyEventArgs - |
| ControllersWPF.GameControllerWPF.PlayerPauseCheck(System.Object,System.Windows.Input.KeyEventArgs) | Проверка установки игроком паузы | parSender - ,parKeyEventArgs - |
| ControllersWPF.GameControllerWPF.PlayerResumeCheck(System.Object,System.Windows.Input.KeyEventArgs) | Обработка возобновления игры игроком | parSender - ,parKeyEventArgs - |
| ControllersWPF.GameControllerWPF.SetHandlerPlayerExited | Обработка выхода игрока |  |
| ControllersWPF.GameControllerWPF.SetHandlerPlayerPaused | Обработка установки паузы игроком |  |
| ControllersWPF.GameControllerWPF.SetHandlerPlayerResumed | Обработка возобновления игры игроком |  |
| ControllersWPF.GameControllerWPF.ResetHandlerPlayerExited | Выключение обработки выхода игрока |  |
| ControllersWPF.GameControllerWPF.ResetHandlerPlayerPaused | Выключение обработки установки паузы игроком |  |
| ControllersWPF.GameControllerWPF.ResetHandlerPlayerResumed | Выключение обработки возобновления игры игроком |  |
| ControllersWPF.GameControllerWPF.CalculatePlayerScreenPosition | Возвращает положение центра игрока на экране |  |
| ControllersWPF.GameControllerWPF.PlayerSpeedUpdateHandler | Исполнитель действия по обновлению скорости игрока |  |
| ControllersWPF.GameControllerWPF.SetHandlerPlayerSpeedSet | Включание обработки управления скоростью клеток игрока |  |
| ControllersWPF.GameControllerWPF.ResetHandlerPlayerSpeedSet | Выключение обработки управления скоростью клеток игрока |  |
| ControllersWPF.GameControllerWPF.PlayerDivideHandler(System.Object,System.Windows.Input.KeyEventArgs) | Обработка разделения игрока | parSender - ,parKeyEventArgs - |
| ControllersWPF.GameControllerWPF.SetHandlerPlayerDivided | Включение обработки разделения игрока |  |
| ControllersWPF.GameControllerWPF.ResetHandlerPlayerDivided | Выключение обработки разделения игрока |  |
| ControllersWPF.MenuMainControllerWPF.#ctor(AgarioModels.Menu.Menu) | Инициализация контроллера главного меню | parMenu - Меню |
| ControllersWPF.MenuMainControllerWPF.Start | Запуск меню |  |
| ControllersWPF.MenuMainControllerWPF.ConfigureMenuView(AgarioModels.Menu.Menu,System.Windows.Window) | Настройка представления меню | parMenu - Меню,parWindow - Окно меню |
| ControllersWPF.MenuMainControllerWPF.DrawMenu | Вывод меню на экран |  |
| ControllersWPF.PlayerNameControllerWPF.#ctor(System.Windows.Window) | Инициализация | parWindow - Окно меню |
| ControllersWPF.PlayerNameControllerWPF.OnNameEditConfirmed(System.String) | Обработчик обновления имени игрока | parNewName - Новое имя |
| ControllersWPF.PlayerNameControllerWPF.Start | Включение редактирования имени |  |
| ControllersWPF.RecordsControllerWPF.#ctor(System.Windows.Window) | Инициализация контроллера рекордов, создание представления | parWindow - Окно меню |
| ControllersWPF.RecordsControllerWPF.Start | Открытие раздела меню "Рекорды" |  |
| Views.Game.Camera.#ctor(AgarioModels.Game.AgarioGame) | Инициализация камеры | parGame - Экземпляр игры |
| Views.Game.Camera.UpdateScreenWidthToHeightRatio | Обновляет соотношение сторон экрана |  |
| Views.Game.Camera.CenterOnTrackedPlayer | Центрирует камеру на игроке, за которым она должна следить |  |
| Views.Game.Camera.Update | Обновление камеры |  |
| Views.Game.Camera.GetCameraRectangle | Получение прямоугольника, соответствующего области просмотра камеры |  |
| Views.Game.Camera.IsGameFieldBordersInViewPort | Определение того, видны ли на момент вызова стенки игрового поля на камере (вышла ли область обзора за границы поля) |  |
| Views.Game.Camera.GetPlayersInViewport | Получение игроков, которые попали в область просмотра камеры |  |
| Views.Game.Camera.GetEatInViewport | Получение еды, которая попала в область просмотра камеры |  |
| Views.Game.Camera.CalculateLineLengthInScreen(System.Single) | Вычисление длины линии, чтобы отобразить её на экране в соответствии с текущими настройками камеры и параметрами экрана | parLineLengthReal - Реальная длина линии |
| Views.Game.Camera.CalculatePointPositionInScreen(System.Numerics.Vector2) | Вычисление положения точки на экране в соответствии с текущими настройками камеры и параметрами экрана | parPoint - Точка в системе координат игрового поля |
| Views.Game.GameView.#ctor | Инициализация представления игры |  |
| Views.Game.GameView.CreateCamera | Создание камеры |  |
| Views.Game.GameView.OnStartGame | Настройка экрана при запуске игры |  |
| Views.Game.GameView.OnPause | Обновление представления при установке паузы |  |
| Views.Game.GameView.OnResume | Обновление представления при возобновлении игры |  |
| Views.Game.GameView.OnStopGame | Обновление представления при остановке игры |  |
| Views.Game.GameView.DrawLeaderboard | Отображение таблицы лидеров |  |
| Views.Game.GameView.DrawGameInfo | Отображение игровой информации |  |
| Views.Game.GameView.DrawGameField | Отображение игрового поля |  |
| Views.Menu.MenuAboutGameView.ShowNextPage | Показать следующую страницу |  |
| Views.Menu.MenuAboutGameView.ShowPrevPage | Показать предыдущую страницу |  |
| Views.Menu.MenuItemView.#ctor(AgarioModels.Menu.MenuItem) | Создание представления для элемента меню | parMenuItem - Элемент меню |
| Views.Menu.MenuView.#ctor(AgarioModels.Menu.Menu) | Создаёт представление для меню | parMenu - Меню |
| Views.Menu.MenuView.CreateMenuItemView(AgarioModels.Menu.MenuItem) | Создание представления для элемента меню | parMenuItem - Элемент меню |
| Views.Menu.MenuViewCommon.Draw | Рисование |  |
| ViewsConsole.ConsoleHelperUtilite.Coord.#ctor(System.Int16,System.Int16) | Инициализация | X - ,Y - |
| ViewsConsole.ConsoleHelperUtilite.ScreenToClient(System.IntPtr,ViewsConsole.ConsoleHelperUtilite.Point@) | Перевод из координат экрана в координаты окна с дескриптором | hWnd - ,lpPoint - |
| ViewsConsole.ConsoleHelperUtilite.GetCursorPos(ViewsConsole.ConsoleHelperUtilite.Point@) | Получение положения курсора | lpPoint - |
| ViewsConsole.ConsoleHelperUtilite.SetConsoleOutputCP(System.UInt32) | Установка кодировки выхода консоли | wCodePageID - |
| ViewsConsole.ConsoleHelperUtilite.SetConsoleCP(System.UInt32) | Установка кодировки консоли | wCodePageID - |
| ViewsConsole.ConsoleHelperUtilite.FindWindow(System.String,System.String) | Поиск дескриптора окна по классу и заголовку | lpClassName - Имя класса окна,lpWindowName - Имя заголовка окна |
| ViewsConsole.ConsoleHelperUtilite.FindWindowByCaption(System.IntPtr,System.String) | Поиск дескриптора окна по заголовку | ZeroOnly - Класс окна в более общей функции. Здесь же - всегда должно быть 0,lpWindowName - Точный заголовок окна |
| ViewsConsole.ConsoleHelperUtilite.SetWindowPos(System.IntPtr,System.IntPtr,System.Int32,System.Int32,System.Int32,System.Int32,System.UInt32) | Установка положения окна | hWnd - Дескриптор окна,hWndInsertAfter - Для управления Z-индексом,X - Новая позиция по X,Y - Новая позиция по Y,cx - Связано с масштабом, не используется,cy - Связано с масштабом, не используется,uFlags - Флаги для управления параметрами окна и размером |
| ViewsConsole.ConsoleHelperUtilite.CreateFile(System.String,System.UInt32,System.UInt32,System.IntPtr,System.IO.FileMode,System.Int32,System.IntPtr) | Создание файла | fileName - ,fileAccess - ,fileShare - ,securityAttributes - ,creationDisposition - ,flags - ,template - |
| ViewsConsole.ConsoleHelperUtilite.WriteConsoleOutput(Microsoft.Win32.SafeHandles.SafeFileHandle,ViewsConsole.ConsoleHelperUtilite.CharInfo[],ViewsConsole.ConsoleHelperUtilite.Coord,ViewsConsole.ConsoleHelperUtilite.Coord,ViewsConsole.ConsoleHelperUtilite.SmallRect@) | Вывод в консоль массива | hConsoleOutput - ,lpBuffer - ,dwBufferSize - ,dwBufferCoord - ,lpWriteRegion - |
| ViewsConsole.ConsoleHelperUtilite.SetLayeredWindowAttributes(System.IntPtr,System.UInt32,System.Byte,System.UInt32) | Установка атрибутов отображения окна | hwnd - Дескриптор окна,crKey - Управление "цветом прозрачности", здесь не используется,bAlpha - Значение прозрачности,dwFlags - Флаги управления параметрами |
| ViewsConsole.ConsoleHelperUtilite.GetWindowRect(System.IntPtr,System.Drawing.Rectangle@) | Получение прямоугольника окна консоли | hwnd - Дескриптор окна консоли,rect - Прямоугольник |
| ViewsConsole.ConsoleHelperUtilite.GetConsoleWindowRectangle(System.IntPtr) | Получение прямоугольника окна консоли | parConsoleWindowHandler - Дескриптор окна консоли |
| ViewsConsole.ConsoleHelperUtilite.WaitSomeEmpiricalTimeInterval | Ожидание достаточно долгого интервала времени, чтобы системные вызовы и инициализация успели сработать. Вызывать при старте, после обновления заголовка (без этого при старте не находится дескриптор окна) |  |
| ViewsConsole.ConsoleHelperUtilite.MoveConsoleWindow(System.String,System.Int32,System.Int32) | Перемещение окна консоли | parWindowName - Заголовок окна консоли,parPixelX - Новая X-координата,parPixelY - Новая Y-координата |
| ViewsConsole.ConsoleHelperUtilite.SetConsoleOpacity(System.String,System.Byte) | Установка прозрачности окна | parWindowName - Заголовок окна,parOpacity - Значение прозрачности [0-255] |
| ViewsConsole.ConsoleHelperUtilite.GetConsoleWindowHandle(System.String) | Получение дескриптора окна консоли | parWindowName - Имя окна консоли |
| ViewsConsole.ConsoleHelperUtilite.GetConsoleOutputHandle | Получение дескриптора вывода консоли |  |
| ViewsConsole.ConsoleHelperUtilite.PrintToConsoleFast(Microsoft.Win32.SafeHandles.SafeFileHandle,ViewsConsole.ConsoleHelperUtilite.CharInfo[],System.Int32,System.Int32) | Быстрый вывод в консоль | parHandle - Дескриптор окна консоли,parBuffer - Буфер с данными,parWidth - Ширина буфера,parHeight - Высота буфера |
| ViewsConsole.ConsoleHelperUtilite.GetCursorPosition(System.IntPtr) | Получение положения курсора мыши | parWindowHandler - Дескриптор окна консоли |
| ViewsConsole.ConsoleHelperUtilite.ChangeConsoleCP | Смена кодировки консоли |  |
| ViewsConsole.Game.CameraConsole.#ctor(AgarioModels.Game.AgarioGame) | Инициализация камеры | parGame - |
| ViewsConsole.Game.CameraConsole.AdjustToPlayerSize | Масштабирование камеры по размерам игрока |  |
| ViewsConsole.Game.CameraConsole.Update | Обновление камеры |  |
| ViewsConsole.Game.GameViewConsole.#ctor | Инициализация |  |
| ViewsConsole.Game.GameViewConsole.CreateCamera | Создание камеры |  |
| ViewsConsole.Game.GameViewConsole.PlaceCharToBuffer(System.Int32,System.Int32,System.ConsoleColor,System.Char) | Вывод символа в буфер консоли | parX - Позиция по горизонтали,parY - Позиция по вертикали,parColor - Цвет,parChar - Символ |
| ViewsConsole.Game.GameViewConsole.DrawCircle(AgarioModels.Game.Cell,System.ConsoleColor,System.Char) | Рисование окружности по алгоритму Брезенхема | parCell - Клетка,parColor - Цвет |
| ViewsConsole.Game.GameViewConsole.DrawEat | Рисование еды |  |
| ViewsConsole.Game.GameViewConsole.DrawPlayer(AgarioModels.Game.Player) | Рисование игрока | parPlayer - Игрок |
| ViewsConsole.Game.GameViewConsole.DrawPlayers | Рисование игроков |  |
| ViewsConsole.Game.GameViewConsole.DrawGameFieldBorders | Рисование границ игрового поля |  |
| ViewsConsole.Game.GameViewConsole.ClearConsoleBuffer | Очистка буфера консоли |  |
| ViewsConsole.Game.GameViewConsole.Render | Функция отрисовки |  |
| ViewsConsole.Game.GameViewConsole.GetRandomForegroundConsoleColor(System.ConsoleColor) | Получение случайного цвета переднего плана, не совпадающего с | parBackgroundColor - Цвет заднего плана |
| ViewsConsole.Game.GameViewConsole.OnPlayerCreated(AgarioModels.Game.Player) | Обработка добавления игрока на поле | parPlayer - Добавленный игрок |
| ViewsConsole.Game.GameViewConsole.OnEatCreated(AgarioModels.Game.Cell) | Обработка появления еды на поле | parEat - Добавленная еда |
| ViewsConsole.Game.GameViewConsole.OnFoodEaten(AgarioModels.Game.Cell,AgarioModels.Game.Player) | Обработка съедания еды | parEat - ,\_ - |
| ViewsConsole.Game.GameViewConsole.DrawGameField | Отображение игрового поля |  |
| ViewsConsole.Game.GameViewConsole.DrawGameInfo | Отображение игровой информации |  |
| ViewsConsole.Game.GameViewConsole.PrintString(System.String,System.ConsoleColor,System.Int32,System.Int32) | Выводит строку с началом в точке (, ) цветом . Без переносов, при нехватке места строка отсечётся | parString - ,parColor - ,parX - ,parY - |
| ViewsConsole.Game.GameViewConsole.DrawLeaderboard | Отображение таблицы лидеров |  |
| ViewsConsole.Game.GameViewConsole.OnPause | Обновление представления при установке паузы |  |
| ViewsConsole.Game.GameViewConsole.OnResume | Обновление представления при возобновлении игры |  |
| ViewsConsole.Game.GameViewConsole.OnStartGame | Настройка экрана при запуске игры |  |
| ViewsConsole.Game.GameViewConsole.OnStopGame | Обновление представления при остановке игры |  |
| ViewsConsole.Menu.AboutGameViewConsole.#ctor | Инициализация представления экрана |  |
| ViewsConsole.Menu.AboutGameViewConsole.Draw | Отрисовка экрана "Об игре" |  |
| ViewsConsole.Menu.AboutGameViewConsole.ShowNextPage | Показать следующую страницу |  |
| ViewsConsole.Menu.AboutGameViewConsole.ShowPrevPage | Показать предыдущую страницу |  |
| ViewsConsole.Menu.MenuItemViewConsole.#ctor(AgarioModels.Menu.MenuItem) | Инициализация представления пункта меню | parMenuItem - |
| ViewsConsole.Menu.MenuItemViewConsole.Draw | Отображение пункта меню |  |
| ViewsConsole.Menu.MenuViewConsole.#ctor(AgarioModels.Menu.Menu) | Инициализация представления меню | parMenu - Меню |
| ViewsConsole.Menu.MenuViewConsole.SetConsoleParametersForMenu | Установка размеров консоли для отображения меню |  |
| ViewsConsole.Menu.MenuViewConsole.InitializeMenuView | Настройка отображения консоли |  |
| ViewsConsole.Menu.MenuViewConsole.CreateMenuItemView(AgarioModels.Menu.MenuItem) | Создание представления элемента меню | parMenuItem - Элемент меню |
| ViewsConsole.Menu.MenuViewConsole.Draw | Отображение меню на экране |  |
| ViewsConsole.Menu.PlayerNameViewConsole.Draw | Рисование |  |
| ViewsConsole.Menu.RecordsViewColsole.Draw | Отображение |  |
| ViewsWPF.Game.CameraWPF.#ctor(AgarioModels.Game.AgarioGame) | Инициализация камеры | parGame - Экземпляр игры |
| ViewsWPF.Game.CameraWPF.Update | Обновление камеры |  |
| ViewsWPF.Game.GameViewWPF.PlayerFigure.#ctor(AgarioModels.Game.Player,System.Windows.Media.Brush,System.Collections.Generic.List{System.Windows.Shapes.Ellipse},System.Windows.Controls.TextBlock) | Инициализация | parPlayer - Игрок,parCellFillingBrush - Кисть для заполнения клеток игрока,parCellFigures - Фигуры для клеток игрока |
| ViewsWPF.Game.GameViewWPF.#ctor(System.Windows.Window) | Инициализация представления игры в WPF | parWindow - Окно игры |
| ViewsWPF.Game.GameViewWPF.CreateGameInfo | Создание блока информации об игре |  |
| ViewsWPF.Game.GameViewWPF.CreateLeaderboard | Создание блока таблицы лидеров |  |
| ViewsWPF.Game.GameViewWPF.CreateFieldBordersShapes | Создание фигур для границ игрового поля |  |
| ViewsWPF.Game.GameViewWPF.GetRandomColor | Возвращает случайный цвет |  |
| ViewsWPF.Game.GameViewWPF.CreatePlayerFigure(AgarioModels.Game.Player) | Создаёт объект фигуры игрока | parPlayer - Игрок |
| ViewsWPF.Game.GameViewWPF.OnPlayerCreated(AgarioModels.Game.Player) | Обработка добавления игрока на поле | parPlayer - Добавленный игрок |
| ViewsWPF.Game.GameViewWPF.OnPlayerDead(AgarioModels.Game.Player) | Обработка смерти игрока | parPlayer - Умерший игрок |
| ViewsWPF.Game.GameViewWPF.OnPlayerReborn(AgarioModels.Game.Player) | Обработка возрождения игрока | parPlayer - Возрождённый игрок |
| ViewsWPF.Game.GameViewWPF.OnEatCreated(AgarioModels.Game.Cell) | Обработка появления еды на поле | parEat - Добавленная еда |
| ViewsWPF.Game.GameViewWPF.OnFoodEaten(AgarioModels.Game.Cell,AgarioModels.Game.Player) | Обработка съедания еды | parEat - ,\_ - |
| ViewsWPF.Game.GameViewWPF.OnStartGame | Настройка экрана при запуске игры |  |
| ViewsWPF.Game.GameViewWPF.OnPause | Обновление представления при установке паузы |  |
| ViewsWPF.Game.GameViewWPF.OnResume | Обновление представления при возобновлении игры |  |
| ViewsWPF.Game.GameViewWPF.OnStopGame | Обновление представления при остановке игры |  |
| ViewsWPF.Game.GameViewWPF.CreateCamera | Создание камеры |  |
| ViewsWPF.Game.GameViewWPF.OnCanRender | Отрисовка по готовности нового кадра |  |
| ViewsWPF.Game.GameViewWPF.Render | Функция отрисовки |  |
| ViewsWPF.Game.GameViewWPF.DrawGameFieldBorders | Рисование границ игрового поля |  |
| ViewsWPF.Game.GameViewWPF.RemoveExcessCellFigures(ViewsWPF.Game.GameViewWPF.PlayerFigure,System.Int32) | Удаление лишних фигур для клеток игрока при рисовании | parPlayerFigure - Данные для рисования игрока,count - Количество лишних клеток |
| ViewsWPF.Game.GameViewWPF.AddMissingCellFigures(ViewsWPF.Game.GameViewWPF.PlayerFigure,System.Int32) | Добавление недостающих фигур для клеток игрока при рисовании | parPlayerFigure - Данные для рисования игрока,count - Количество недостающих клеток |
| ViewsWPF.Game.GameViewWPF.CheckCellFiguresCount(ViewsWPF.Game.GameViewWPF.PlayerFigure) | Удаление лишних клеток на экран или добавление недостающих при отрисовке игрока | parPlayerFigure - Данные для отрисовки игрока |
| ViewsWPF.Game.GameViewWPF.DrawCell(AgarioModels.Game.Cell,System.Windows.Shapes.Ellipse) | Рисование клетки | parCell - Клетка,parCellFigure - Фигура клетки на экране |
| ViewsWPF.Game.GameViewWPF.DrawEat(AgarioModels.Game.Cell) | Рисование еды | parEat - Еда |
| ViewsWPF.Game.GameViewWPF.DrawPlayer(AgarioModels.Game.Player) | Рисование игрока | parPlayer - Игрок |
| ViewsWPF.Game.GameViewWPF.DrawPlayers | Рисование игроков |  |
| ViewsWPF.Game.GameViewWPF.DrawFood | Рисование еды |  |
| ViewsWPF.Game.GameViewWPF.DrawGameField | Отображение игрового поля |  |
| ViewsWPF.Game.GameViewWPF.DrawGameInfo | Отображение игровой информации |  |
| ViewsWPF.Game.GameViewWPF.DrawLeaderboard | Отображение таблицы лидеров |  |
| ViewsWPF.Menu.AboutGameViewWPF.#ctor(System.Windows.Window) | Инциаиализация представления | parWindow - Окно меню |
| ViewsWPF.Menu.AboutGameViewWPF.GetScreenLayout | Получение экрана представления |  |
| ViewsWPF.Menu.AboutGameViewWPF.KeyHandler(System.Object,System.Windows.Input.KeyEventArgs) | Обработчик событий клавиатуры | parSender - Отправитель,parArgs - Аргументы события |
| ViewsWPF.Menu.AboutGameViewWPF.Draw | Рисование пункта меню |  |
| ViewsWPF.Menu.AboutGameViewWPF.ShowNextPage | Выбор следующей страницы |  |
| ViewsWPF.Menu.AboutGameViewWPF.ShowPrevPage | Выбор предыдущей страницы |  |
| ViewsWPF.Menu.MenuItemViewWPF.#ctor(AgarioModels.Menu.MenuItem) | Инициализация представления элемента меню | parMenuItem - Элемент меню |
| ViewsWPF.Menu.MenuItemViewWPF.Draw | Отображение пункта меню |  |
| ViewsWPF.Menu.MenuViewWPF.#ctor(AgarioModels.Menu.Menu,System.Windows.Window) | Инициализация представления меню | parMenu - Меню,parMenuWindow - Окно меню |
| ViewsWPF.Menu.MenuViewWPF.CreateMenuItemView(AgarioModels.Menu.MenuItem) | Создание представления элемента меню | parMenuItem - Элемент меню |
| ViewsWPF.Menu.MenuViewWPF.Draw | Отображение меню |  |
| ViewsWPF.Menu.MenuViewWPF.MenuKeyHandler(System.Object,System.Windows.Input.KeyEventArgs) | Обработчик событий клавиатуры | parSender - Отправитель,parEventArgs - Аргументы события |
| ViewsWPF.Menu.PlayerNameViewWPF.#ctor(System.Windows.Window) | Инициализация представления | parWindow - |
| ViewsWPF.Menu.PlayerNameViewWPF.GetScreenLayout | Получение экрана представления |  |
| ViewsWPF.Menu.PlayerNameViewWPF.KeyHandler(System.Object,System.Windows.Input.KeyEventArgs) | Обработчик событий клавиатуры | parSender - Отправитель,parArgs - Аргументы события |
| ViewsWPF.Menu.PlayerNameViewWPF.Draw | Рисование |  |
| ViewsWPF.Menu.RecordsViewWPF.#ctor(System.Windows.Window) | Инициализация представления пункта меню | parWindow - Окно игры |
| ViewsWPF.Menu.RecordsViewWPF.GetScreenLayout | Получение экрана представления |  |
| ViewsWPF.Menu.RecordsViewWPF.KeyHandler(System.Object,System.Windows.Input.KeyEventArgs) | Обработчик событий клавиатуры | parSender - Отправитель,parArgs - Аргументы события |
| ViewsWPF.Menu.RecordsViewWPF.Draw | Отображение пункта меню |  |
| ViewsWPF.Menu.RecordsViewWPF.FillRecords | Обновление рекордов на экране |  |

## Разработка программы

### Описание классов, перечислений и интерфейсов проекта

**Таблица 2**

|  |  |  |
| --- | --- | --- |
| **Пространство имен** | **Название класса / интерфейса / перечисления** | **Назначение** |
| AgarioModels.Game | AgarioGame | Игра |
| AgarioModels.Game.AgarioGame | State | Состояния игры |
| AgarioModels.Game | Cell | Клетка |
| AgarioModels.Game | CellPlaceholder | Расстановщик клеток еды или игроков |
| AgarioModels.Game | ComputerMovingStrategy | Логика перемещения игроков, управляемых компьютером |
| AgarioModels.Game | GameField | Игровое поле |
| AgarioModels.Game | MathFunctions | Математические функции |
| AgarioModels.Game.MathFunctions | Rectangle | Прямоугольник, задающийся двумя точками: (X1, Y1) - левый верх и (X2, Y2) - правый низ |
| AgarioModels.Game | MovingCell | Движущаяся клетка |
| AgarioModels.Game | Player | Клетка или группа клеток, управляемая как единое целое. Класс представляет собой просто связанную систему клеток с общим управлением и не является специфичным для "реального" игрока за компьютером |
| AgarioModels.Menu.AboutGame | DataReader | Класс, оборачивающий доступ к информации об игре |
| AgarioModels.Menu | Menu | Линейное меню без вложенных подпунктов |
| AgarioModels.Menu | MenuItem | Элемент меню |
| AgarioModels.Menu.MenuItem | MenuItemState | Состояния пункта меню |
| AgarioModels.Menu | MenuMain | Стартовое меню игры |
| AgarioModels.Menu.MenuMain | MenuItemCodes | Коды элементов главного меню |
| AgarioModels.Menu.Records | GameRecordsHandler | Обработчик игровых рекордов. Управляет запросом имени пользователя при сохранении после окончания игры |
| AgarioModels.Menu.Records | RecordsIO | Класс-посредник между файловой системой и игровой моделью. Обеспечивает операции сохранения и получения рекордов, храня их в памяти после первого обращения для кэширования |
| AgarioModels.Menu.Records.RecordsIO | DescendingRecordComparer | Класс для сравнения рекордов по убыванию значения |
| AgarioModels.Properties | Resources | Класс ресурса со строгой типизацией для поиска локализованных строк и т.д. |
| AgarioWPF | App | Interaction logic for App.xaml |
| Controllers.Game | GameController | Контроллер игры |
| Controllers.Menu | MenuAboutGameController | Контроллер раздела меню "Об игре" |
| Controllers.Menu | MenuController | Общее для контроллеров меню |
| Controllers.Menu | MenuPlayerNameController | Контроллер пункта настройки имени игрока |
| Controllers.Menu | MenuRecordsController | Контроллер раздела меню "Рекорды" |
| Controllers.Menu | MenuScreenController | Общее для контроллеров экранов меню |
| ControllersConsole | AboutGameControllerConsole | Контроллер раздела меню "Об игре" |
| ControllersConsole | GameControllerConsole | Контроллер игры в консоли |
| ControllersConsole | MenuMainControllerConsole | Контроллер главного меню в консоли |
| ControllersConsole | PlayerNameControllerConsole | Контроллер пункта настройки имени игрока |
| ControllersConsole | RecordsControllerConsole | Контроллер игровых рекордов в консоли |
| ControllersWPF | AboutGameControllerWPF | Контроллер раздела меню "Об игре" в WPF |
| ControllersWPF | GameControllerWPF | Игровой контроллер в WPF |
| ControllersWPF | MenuMainControllerWPF | Реализация контроллера меню для WPF |
| ControllersWPF | PlayerNameControllerWPF | Контроллер пункта настройки имени игрока в WPF |
| ControllersWPF | RecordsControllerWPF | Контроллер экрана рекордов в WPF |
| Views.Game | Camera | Класс для управления областью игрового поля, которая отображается на экране у игрока. Ширина и высота видимой области игрового поля, захватываемой камерой, всегда будет иметь соотношение сторон, задаваемое выражением ScreenWidth / ScreenHeight |
| Views.Game | GameView | Представление игры |
| Views.Menu | MenuAboutGameView | Представление экрана меню "Об игре" |
| Views.Menu | MenuItemView | Представление элемента меню |
| Views.Menu | MenuPlayerNameView | Представление экрана меню настройки имени игрока |
| Views.Menu | MenuRecordsView | Представление экрана меню "Рекорды" |
| Views.Menu | MenuView | Представление меню |
| Views.Menu | MenuViewCommon | Общее для представлений |
| ViewsConsole | ConsoleHelperUtilite | Вспомогательный класс-прослойка между вызовами WINAPI |
| ViewsConsole.ConsoleHelperUtilite | Coord | Координаты |
| ViewsConsole.ConsoleHelperUtilite | CharUnion | Представление символа |
| ViewsConsole.ConsoleHelperUtilite | CharInfo | Информация о символе |
| ViewsConsole.ConsoleHelperUtilite | SmallRect | Прямоугольник |
| ViewsConsole.ConsoleHelperUtilite | Point | Точка |
| ViewsConsole.Game | CameraConsole | Камера в консоли |
| ViewsConsole.Game | GameViewConsole | Представление игры в консоли |
| ViewsConsole.Menu | AboutGameViewConsole | Представление пункта меню "Об игре" в консоли |
| ViewsConsole.Menu | MenuItemViewConsole | Представление пунктов меню в консоли |
| ViewsConsole.Menu | MenuViewConsole | Представление меню в консоли |
| ViewsConsole.Menu | PlayerNameViewConsole | Представление экрана меню для настройки имени игрока |
| ViewsConsole.Menu | RecordsViewColsole | Представление пункта меню "Рекорды" в консоли |
| ViewsConsole | ViewsProperties | Набор глобальных констант для представлений |
| ViewsWPF.Game | CameraWPF | Камера для WPF-режима |
| ViewsWPF.Game | GameViewWPF | Представление игры в WPF |
| ViewsWPF.Game.GameViewWPF | PlayerFigure | Фигура игрока для отрисовки |
| ViewsWPF.Menu | AboutGameViewWPF | Представление пункта "Об игре" в WPF |
| ViewsWPF.Menu | MenuItemViewWPF | Представление элемента меню в WPF |
| ViewsWPF.Menu | MenuViewWPF | Представление меню в WPF |
| ViewsWPF.Menu | PlayerNameViewWPF | Представление экрана меню настройки имени игрока |
| ViewsWPF.Menu | RecordsViewWPF | Представления пункта меню "Рекорды" в WPF |
| ViewsWPF | ViewProperties | Глобальные константы представлений |

## Описание шаблонов проектирования, которые использовались при написании программы

### Модель-Представление-Контроллер (Model-View-Controller (MVC))

MVC – шаблон проектирования, в котором приложение разделяется на три компонента: модель, представление и контроллер. Основная цель его применения состоит в отделении логики (модели) от её визуализации (представления, вида). За счёт такого разделения повышается возможность повторного использования кода и появляются больше возможностей для внесения независимых изменений в каждый компонент системы. Наиболее полезно применение концепции MVC в тех случаях, когда пользователь должен видеть те же самые данные одновременно в различных контекстах и/или с различных точек зрения. В частности, выполняются следующие задачи:

* К одной модели можно присоединить несколько представлений, при этом не затрагивая реализацию модели;
* Не затрагивая реализацию представлений, можно изменить реакции на действия пользователя — для этого достаточно использовать другой контроллер.

Проекты игры и содержащиеся в них компоненты созданы в соответствии с принципами этого шаблона:

* Models – содержит игровую логику приложения и его меню;
* Controllers – содержит общее для любых контроллеров, предназначенных для использования в этом приложении;
* Views – содержит общее для любых представлений, предназначенных для использования в этом приложении;
* ViewsConsole, ViewsWPF – представления, которые имеют специфичные детали реализации для консольного и графического приложения.
* ControllersConsole, ControllersWPF – контроллеры, реализация которых специфична для консольного и графического приложения.

### Одиночка (Singleton)

Одиночка (Singleton) - порождающий шаблон проектирования, который гарантирует, что для определенного класса будет создан только один объект, а также предоставит к этому объекту точку доступа.

Этот шаблон реализуется для главного класса игровой модели AgarioGame (проект AgarioModels).

### Заместитель (Proxy)

Заместитель (Proxy) — шаблон проектирования, предоставляющий объект, который контролирует доступ к другому объекту, перехватывая все вызовы (выполняет функцию контейнера). У этого шаблона есть несколько разновидностей, поэтому далее будет описание конкретного решения.

Используемый при проектировании игры вариант шаблона Заместитель является комбинацией Виртуального и Кэширующего Заместителя:

* Виртуальный заместитель (англ. virtual proxies): обеспечивает создание реального «Субъекта» только тогда, когда он действительно понадобится. Также может кэшировать часть информации о реальном «Субъекте», чтобы отложить его создание;
* Кэширующий прокси: обеспечивает временное хранение результатов расчёта до отдачи их множественным клиентам, которые могут разделить эти результаты.

В приложении шаблон Proxy применяется для организации доступа к списку игровых рекордов, позволяя обращаться к файловой системе на чтение всего набора рекордов лишь 1 раз, по первому требованию (при проверке сохранения рекордов или при их запросе из пункта меню). До первого обращения набор рекордов не создан, а после программа будет хранить эти данные в памяти программы, не требуя дальнейших обращений к файловой системе для их считывания (обращения будут лишь при записи новых рекордов). Классы, реализующие шаблон в программе: RecordsIO и GameRecordsHandler (находятся в Models/Menu/Records).

## Описание методов рефакторинга, которые использовались при оптимизации исходного кода программы

### Пересмотр путей взаимодействия компонентов приложения

Было устранено нарушение шаблона MVC, которое выражалось в виде прямого влияния контроллера на представление в обход модели. До изменений обновление скорости клетки в методе контроллера сразу сообщало представлению о необходимости перерисовки. После изменения представление получало эту информацию, подписываясь на событие модели.

### Более явное разделение функионала и обязанностей у частей MVC

В WPF версии окно программы появлялось до создания Представления и Контроллера и существовало как бы отдельно от них. После проведения изменений окно программы создавалось Представлением, а Контроллер получал необходимую информацию о нём через запрос к Представлению.

### Выделение логически связанной функциональности для переиспользования в разных компонентах

До проведения этого способа рефакторинга в модели отсутствовал статический вспомогательный класс MathFunctions, в котором содержатся вспомогательные математические функции, предназначенные для работы с игровыми объектами. Весь функционал этого вспомогательного класса существовал в разных компонентах и дублировался. После рефакторинга общие функции были вынесены в отдельный класс.

### Объединение кода из отдельных функций внутрь свойства

До изменений свойство Radius класса клетки надо было пересчитывать отдельно при изменении массы. После изменений его пересчёт был внесён в свойство Weight клетки:

/// <summary>

/// Масса

/// </summary>

public int Weight

{

get => \_weight;

set

{

\_weight = value;

Radius = MathFunctions.CalculateRadiusByMass(value);

}

}

## Разработка тестов

### Test Cases

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Содержание теста | Предусловие | Действия | Ожидаемый результат |
| 1 | Переход из главного меню к рекордам | Открыто главное меню. | Перейти к пункту меню «Рекорды»  Нажать на пункт меню «Рекорды» | Открыто окно рекордов и выведена таблица рекордов |
| 2 | Сохранение состояния меню после возврата из рекордов в главное меню | Открыто окно рекордов. | Нажать клавишу «Назад». | Открыто главное меню, фокус находится на пункте меню «Рекорды» |
| 3 | Переход из главного меню к разделу с информацией об игре | Открыто главное меню. | Перейти к пункту меню «Об игре».  Нажать на пункт меню «Об игре» | Открыто окно правил и выведены правила игры. |
| 4. | Перемотка страниц в разделе с информацией об игре | Открыт раздел с информацией об игре | Нажать стрелку вправо  Нажать стрелку влево | Откроется вторая страница информации, затем снова первая |
| 5. | Проверка циклической прокрутки главного меню снизу вверх | Открыто главное меню, в фокусе находится нижний его пункт | Нажать стрелку вниз | Самый верхний пункт меню станет в фокусе |
| 6. | Проверка циклической прокрутки главного меню сверху вниз | Открыто главное меню, в фокусе находится верхний его пункт | Нажать стрелку вверх | Самый нижний пункт меню станет в фокусе |
| 7. | Проверка возврата из окна рекордов в главное меню 1 | Открыто окно рекордов | Нажать клавишу Enter | Произойдёт возврат в главное меню |
| 8. | Проверка возврата из окна рекордов в главное меню 2 | Открыто окно рекордов | Нажать клавишу Escsape | Произойдёт возврат в главное меню |
| 9. | Проверка возврата из окна рекордов в главное меню 3 | Открыто окно рекордов | Нажать клавишу пробел | Произойдёт возврат в главное меню |
| 10. | Проверка возврата из окна «Об игре» в главное меню 1 | Открыто окно «Об игре» | Нажать клавишу Enter | Произойдёт возврат в главное меню |
| 11. | Проверка возврата из окна «Об игре» в главное меню 2 | Открыто окно «Об игре» | Нажать клавишу Escsape | Произойдёт возврат в главное меню |
| 12. | Проверка возврата из окна «Об игре» в главное меню 3 | Открыто окно «Об игре» | Нажать клавишу пробел | Произойдёт возврат в главное меню |
| 13. | Проверка запуска стартового экрана игры 1 | Открыто главное меню | Нажать клавишу Enter | Произойдёт переход на стартовый экран игры, состояние статично |
| 14. | Проверка запуска стартового экрана игры 2 | Открыто главное меню | Нажать клавишу Escsape | Произойдёт переход на стартовый экран игры, состояние статично |
| 15. | Проверка запуска стартового экрана игры 3 | Открыто главное меню | Нажать клавишу пробел | Произойдёт переход на стартовый экран игры, состояние статично |
| 16. | Проверка движения клетки | Активное состояние игры | Передвинуть мышку | Клетка начнёт движение в направлении мышки |
| 17. | Проверка завершения игры | Активное состояние игры | Нажать escape | Игрок переместится в главное меню |
| 18. | Проверка сохранения рекорда | Набрано достаточное количество очков для попадания в список рекордов | Закончить игру проверить файл с рекордами | В окне рекордов будет запись с введённым именем и набранным значением очков |
| 19. | Проверка выхода | Способ 1: Открыто главное меню  Способ 2: открыт любой экран приложения в случае второго способа) | Способ 1: нажать Enter или пробел  Способ 2: закрыть окно приложения нажатием на крестик в правом верху | Окно приложения будет закрыто |

### Модульные тесты

Для тестирования функциональности игры были созданы модульные тесты для класса MathFunctions с функциями для вычислений. Тесты расположены в проекте UnitTests в папке решения Common. Составлен класс MathFunctionsTests в котором имеются 20 модульных тестов. Пример трех тестов:

[TestMethod]

public void TestCalculateRadiusByMass2()

{

// Arrange

int mass = 25;

// Act

float radius = MathFunctions.CalculateRadiusByMass(mass);

// Assert

Assert.AreEqual(0.5f, radius, 0.001f);

}

[TestMethod]

public void TestGetIntersectionArea\_NoOverlap()

{

// Arrange

Cell cell1 = new() { Position = new Vector2(0, 0), Radius = 2 };

Cell cell2 = new() { Position = new Vector2(5, 5), Radius = 2 };

// Act

float area = MathFunctions.GetIntersectionArea(cell1, cell2);

// Assert

Assert.AreEqual(0, area);

}

[TestMethod]

public void TestGetIntersectionArea\_FullyNested()

{

// Arrange

Cell cell1 = new() { Position = new Vector2(0, 0), Radius = 4 };

Cell cell2 = new() { Position = new Vector2(1, 1), Radius = 2 };

// Act

float area = MathFunctions.GetIntersectionArea(cell1, cell2);

// Assert

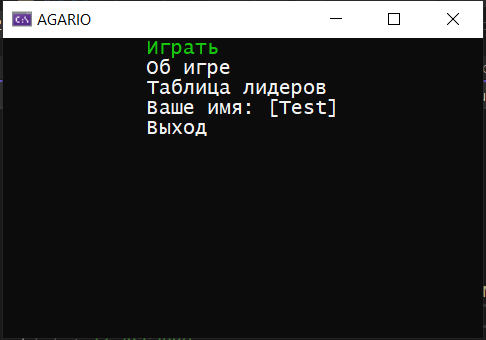
Assert.AreEqual(MathF.PI \* 2 \* 2, area, 0.001f);

}

# Результат работы программы

## Консольное приложение

### Главное меню



**Рисунок 16 – Окно главного меню**

На рисунке 16 приведено окно главного меню. Пользователю доступны 5 действий: начать новую игру («Играть»), просмотреть список рекордов («Рекорды»), просмотреть информацию об игре («Об игре»), войти в режим редактирования имени («Ваше имя»), завершить работу приложения («Выход»).

### Начало игры

При нажатии на пункт главного меню «Играть» пользователь переходит к игре. Экран игры представлен на рисунке 16.

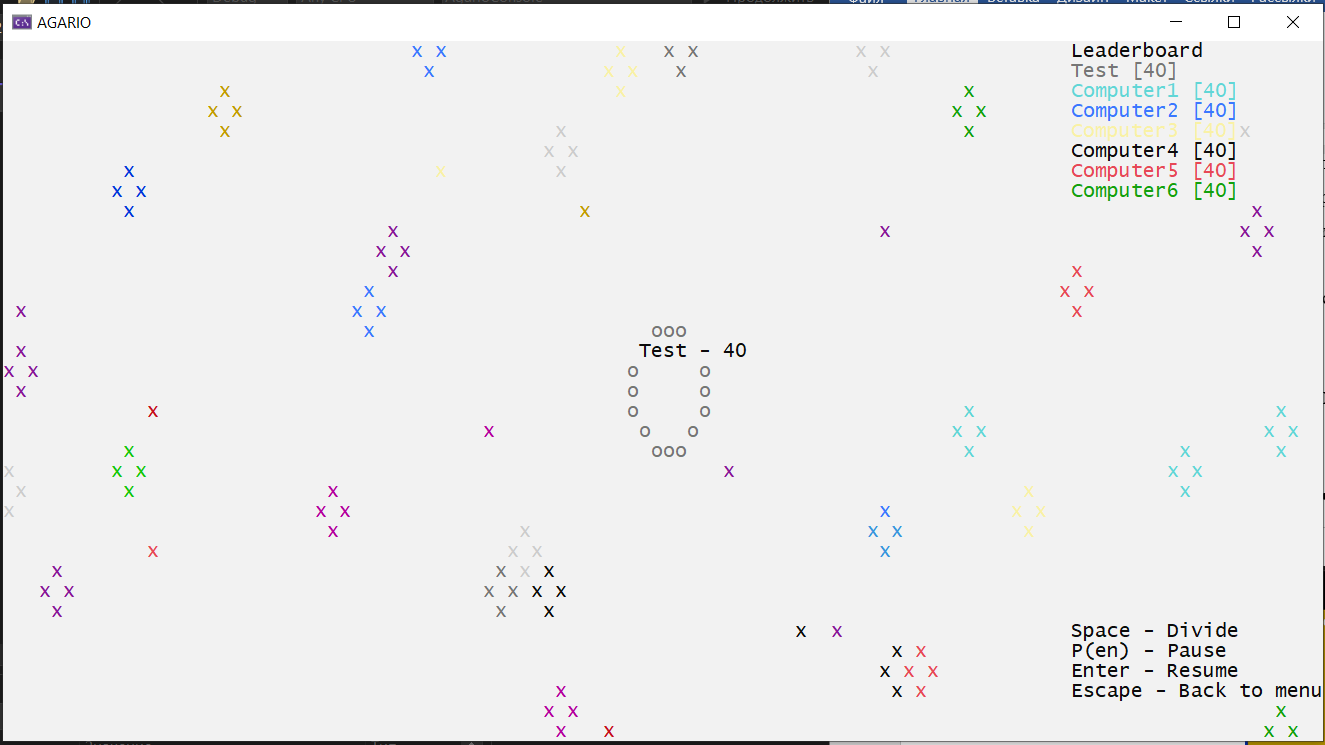


Рисунок 16 - Игра после запуска

### Рекорды

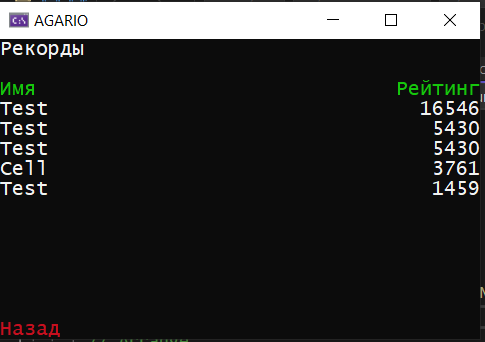


Рисунок – Окно рекордов

При нажатии на пункт главного меню «Рекорды» пользователь переходит в окно рекордов (рисунок 17).

### Об игре

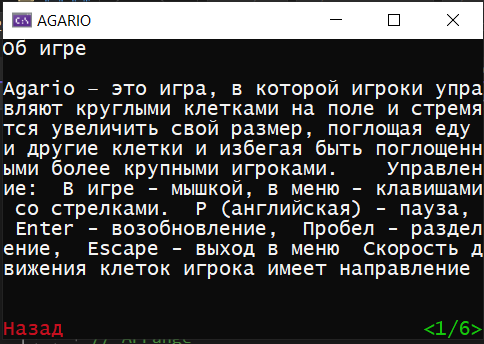


Рисунок – Окно «Об игре»

При нажатии на пункт главного меню «Об ире» пользователь переходит к разделу с информацией об игре (рисунок 18).

## WPF-версия

Игровой процесс и действия с WPF-версией аналогичны консольному приложению, поэтому в данном разделе приведены только снимки игры в WPF-реализации.

### Главное меню

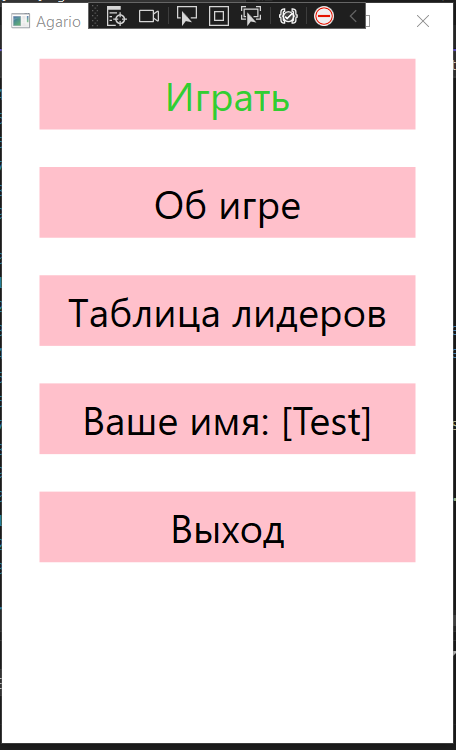


Рисунок – Окно главного меню.

### Начало игры

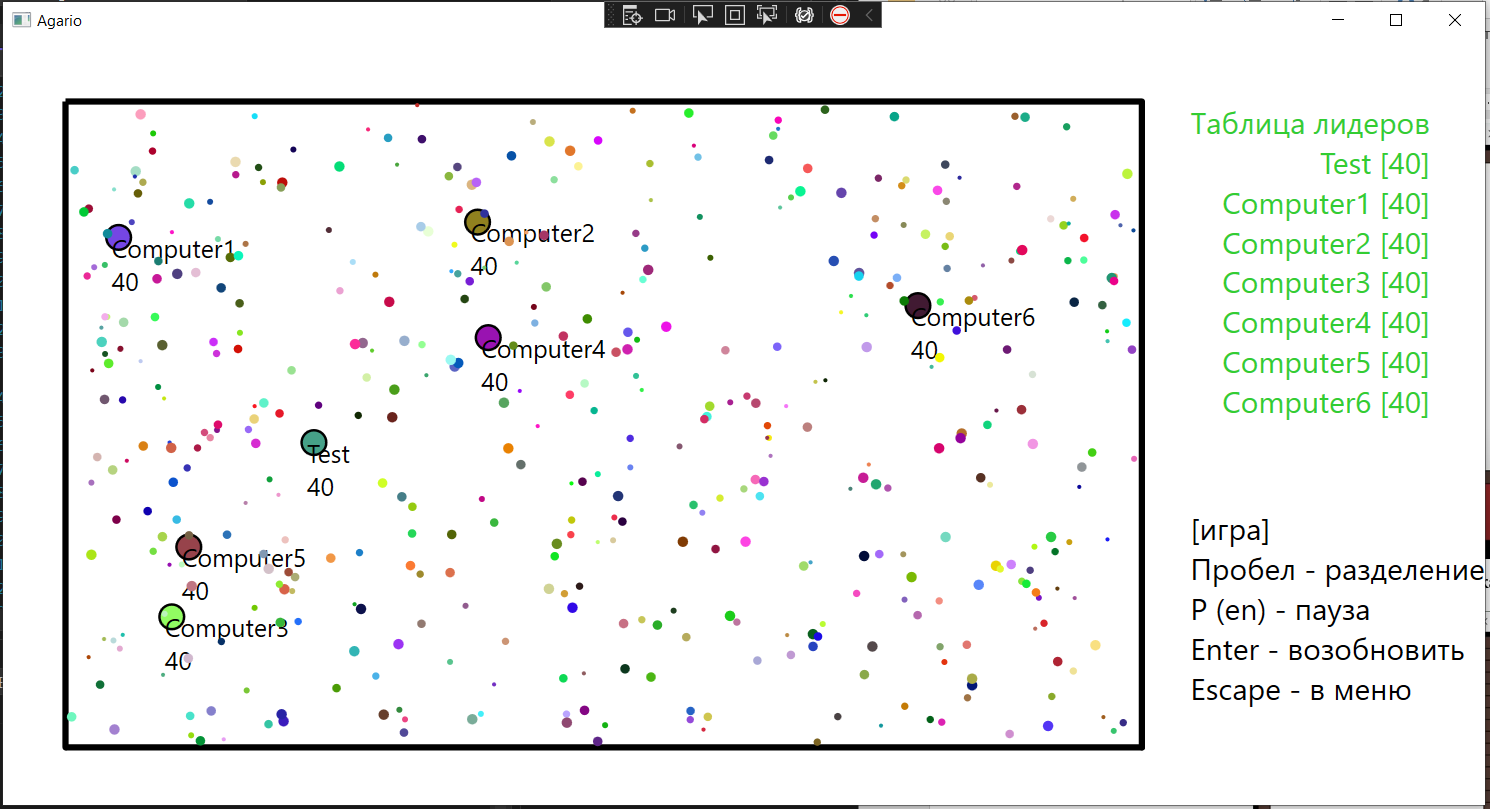


Рисунок – Игра

### Рекорды

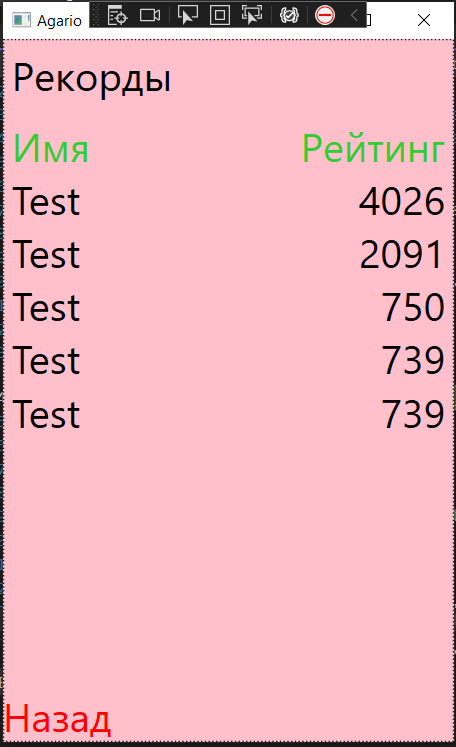


Рисунок – Окно списка рекордов

### Об игре

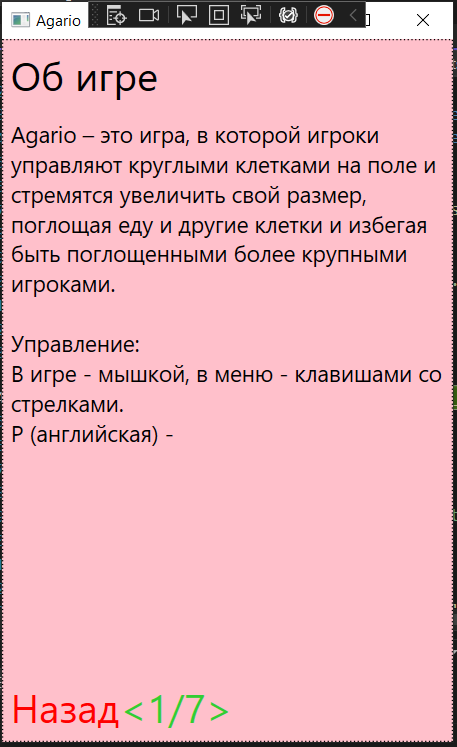


Рисунок – Окно «Об игре»

Заключение

В результате проделанной работы была создана игра «Agario» в двух видах интерфейса: консольном и WPF.