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:

«Игровая программа «Electric Cage»»

по курсу

**«Конструирование программного обеспечения»**

Выполнил:

студент группы №943

Бобылева Е. В.

Проверил:

доцент кафедры ВПМ

Столчнев В. К.

Рязань, 2022

Содержание

[Введение 2](#_Toc124697892)

[1 Анализ задачи 2](#_Toc124697893)

[1.1 Разработка иерархии классов 2](#_Toc124697894)

[1.1.1 Выделение сущностей 2](#_Toc124697895)

[1.1.2 Зависимости между классами. Диаграмма классов 2](#_Toc124697896)

[1.2 Алгоритмы 2](#_Toc124697897)

[1.2.1 Алгоритм при переходе в «Играть» 2](#_Toc124697898)

[1.2.2 Алгоритм при переходе в «Правила» 2](#_Toc124697899)

[1.2.3 Алгоритм при переходе в «Рекорды» 2](#_Toc124697900)

[1.2.4 Алгоритм для определения столкновение космического корабля с электрическим шаром 2](#_Toc124697901)

[1.3 Разработка интерфейса программы 2](#_Toc124697902)

[1.3.1 Интерфейс главного меню 2](#_Toc124697903)

[1.3.2 Интерфейс игры 2](#_Toc124697904)

[1.3.3 Интерфейс игры после поражения 2](#_Toc124697905)

[1.3.4 Интерфейс «Правил игры» 2](#_Toc124697906)

[1.3.5 Интерфейс «Рекордов» 2](#_Toc124697907)

[2 Написание программы 2](#_Toc124697908)

[2.1 Описание разработанных процедур и функций 2](#_Toc124697909)

[2.2 Разработка программы 2](#_Toc124697910)

[2.2.1 Описание классов, перечислений и интерфейсов проекта 2](#_Toc124697911)

[2.3 Описание шаблонов проектирования, которые использовались при написании программы 2](#_Toc124697912)

[2.3.1 Модель-Представление-Контроллер (Model-View-Controller (MVC)) 2](#_Toc124697913)

[2.3.2 Одиночка (Singleton) 2](#_Toc124697914)

[2.3.3 Фабричный метод (Factory Method) 2](#_Toc124697915)

[2.4 Описание методов рефакторинга, которые использовались при оптимизации исходного кода программы 2](#_Toc124697916)

[2.4.1 Извлечение метода из другого метода 2](#_Toc124697917)

[2.4.2 Присвоение методу более информативного имени 2](#_Toc124697918)

[2.4.3 Замена магического числа на именованную константу (переменную) 2](#_Toc124697919)

[2.4.4 Введение промежуточной переменной. 2](#_Toc124697920)

[2.4.5 Передача в метод целого объекта вместо отдельных полей 2](#_Toc124697921)

[2.5 Разработка тестов 2](#_Toc124697922)

[2.5.1 Test Cases 2](#_Toc124697923)

[2.5.2 Модульные тесты 2](#_Toc124697924)

[3 Результат работы программы 2](#_Toc124697925)

[3.1 Консольное приложение 2](#_Toc124697926)

[3.1.1 Главное меню 2](#_Toc124697927)

[3.1.2 Начало игры 2](#_Toc124697928)

[3.1.3 Завершение игры 2](#_Toc124697929)

[3.1.4 Рекорды 2](#_Toc124697930)

[3.1.5 Правила 2](#_Toc124697931)

[3.2 Графическое приложение 2](#_Toc124697932)

[3.2.1 Главное меню 2](#_Toc124697933)

[3.2.2 Начало игры 2](#_Toc124697934)

[3.2.3 Завершение игры 2](#_Toc124697935)

[3.2.4 Рекорды 2](#_Toc124697936)

[3.2.5 Правила 2](#_Toc124697937)

[Заключение 2](#_Toc124697938)

Введение

В качестве темы для курсовой работы была выбрана игра «Electric Cage».

«Electric Cage» относится к категории игр-аркад. В данном игровом приложении пользователю, который управляет космическим кораблем, необходимо избегать столкновения со стенками электрической клетки и электрическими шарами. Однако космический корабль поворачивает только в левую сторону. Очки начисляются пользователю в зависимости от того, столкновение со сколькими электрическими шарами он сумел избежать.

В ходе работы было выполнено проектирование и разработка игрового приложения «Electric Cage». Игра реализована в двух интерфейсах: графическом (WPF) и консольном.

# Анализ задачи

Необходимо разработать игровое приложение «Electric Cage». Пользователь управляет космическим кораблем и должен избегать столкновения с стенками электрической клетки и электрическими шарами.

## Разработка иерархии классов

Приложение должно быть построено с использованием шаблона MVC (Model-View-Controller – Модель-Представление-Контроллер).

Модель описывает игровую логику и не связана с другими модулями. Представление отображает состояние модели. Контроллер обрабатывает пользовательские действия и влияет на модель. Для каждого из интерфейсов должны быть созданы свои контроллеры и представления.

### Выделение сущностей

Игровые сущности:

1. Космический корабль: координата х, координата у, направление движения, тип сущности

2. Электрический шар: номер, координата х, координата у, направление движения, тип сущности

Прочие сущности:

1. Меню: список пунктов меню (кнопки), список текстовых полей

2. Правила: список текстовых полей, список пунктов меню (кнопки)

3. Рекорды: имя игрока, счет

4. Окно окончания игры: список текстовых полей, список полей для ввода, список пунктов меню (кнопки)

5. Пункт меню (кнопка): текст, состояние пункта меню

6. Текстовое поле: текст

7. Поле для ввода: текст

### Зависимости между классами. Диаграмма классов

На рисунке 1 представлена диаграмма классов модели.

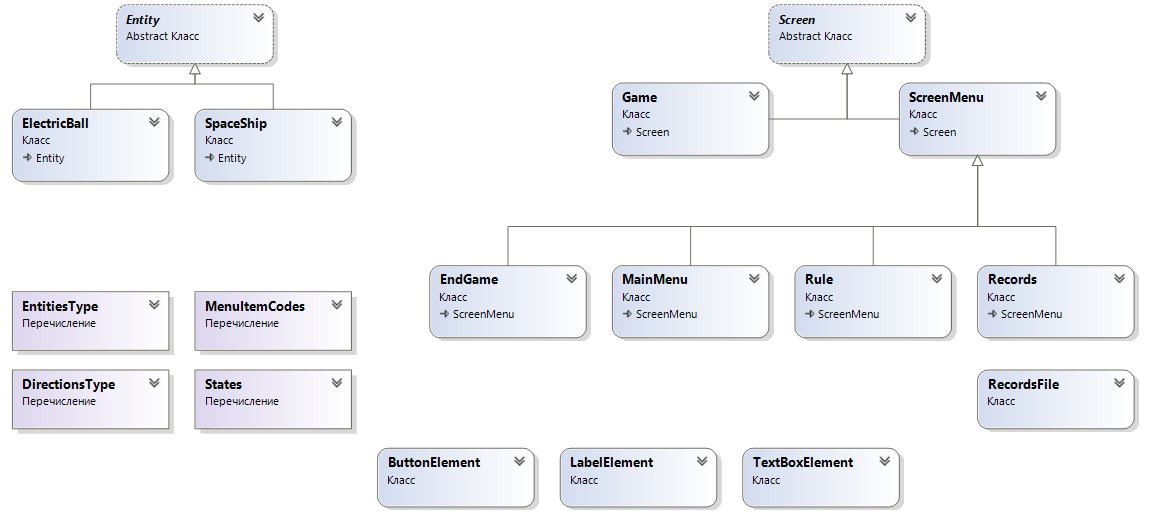


Рисунок 1 - Диаграмма классов модели

На рисунке 2 представлена диаграмма классов контроллера для консольного приложения. Данная диаграмма отображает связь базового контроллера с контроллером консольного приложения.

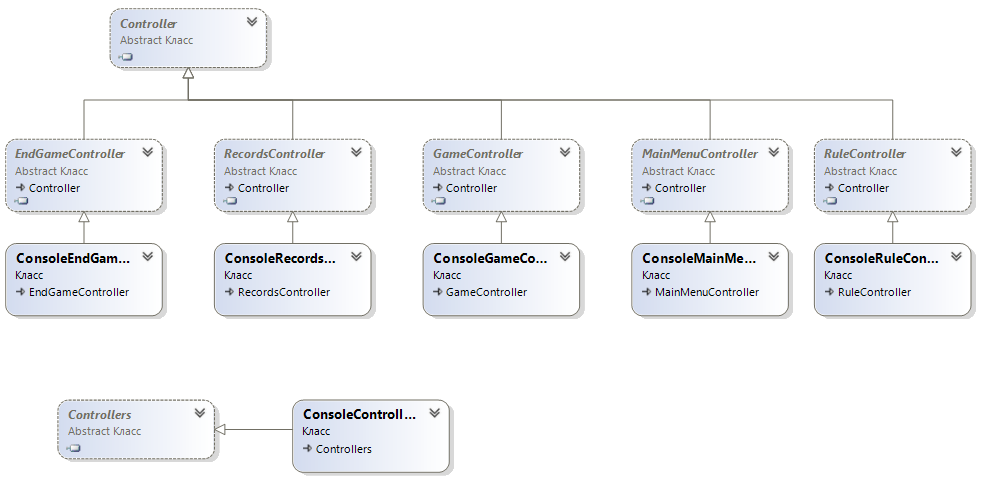


Рисунок 2 - Диаграмма классов контроллеров для консольного приложения

На рисунке 3 представлена диаграмма классов контроллера для графического приложения. Данная диаграмма отображает связь базового контроллера с контроллером графического приложения.

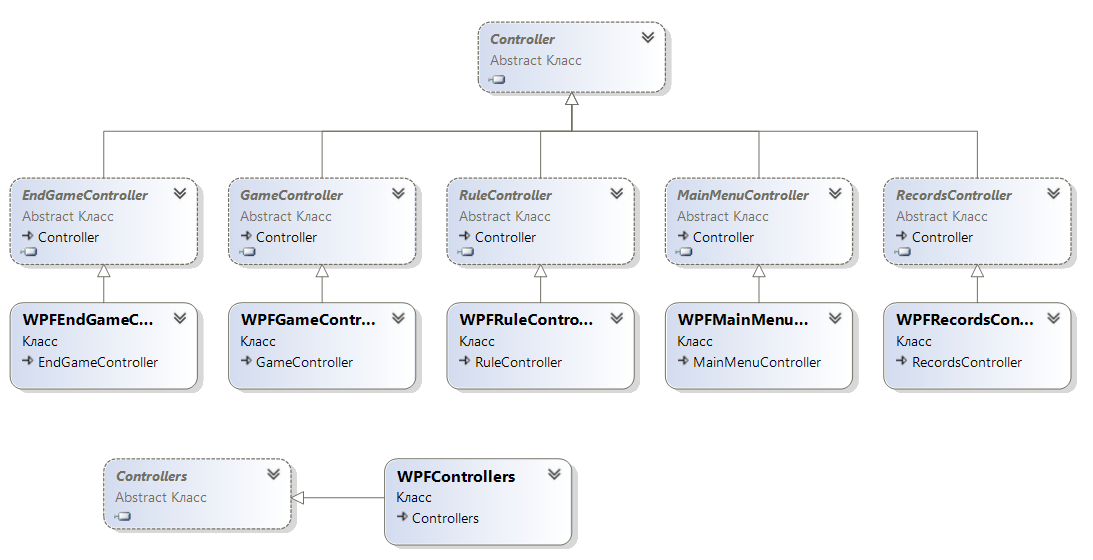


Рисунок 3 - Диаграмма классов контроллеров для графического приложения

На рисунке 4 представлена диаграмма классов представлений для консольного приложения. Данная диаграмма отображает связь базового представления с представлением консольного приложения.

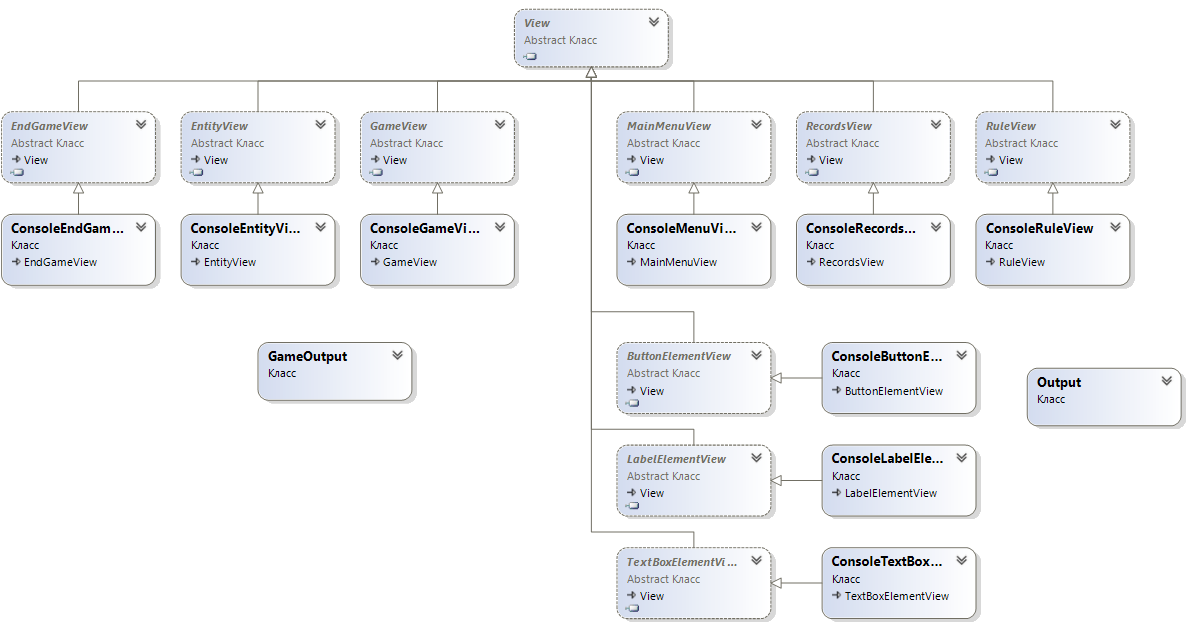


Рисунок 4 - Диаграмма классов представлений для консольного приложения

На рисунке 5 представлена диаграмма классов представлений для графического приложения. Данная диаграмма отображает связь базового представления с представлением графического приложения.

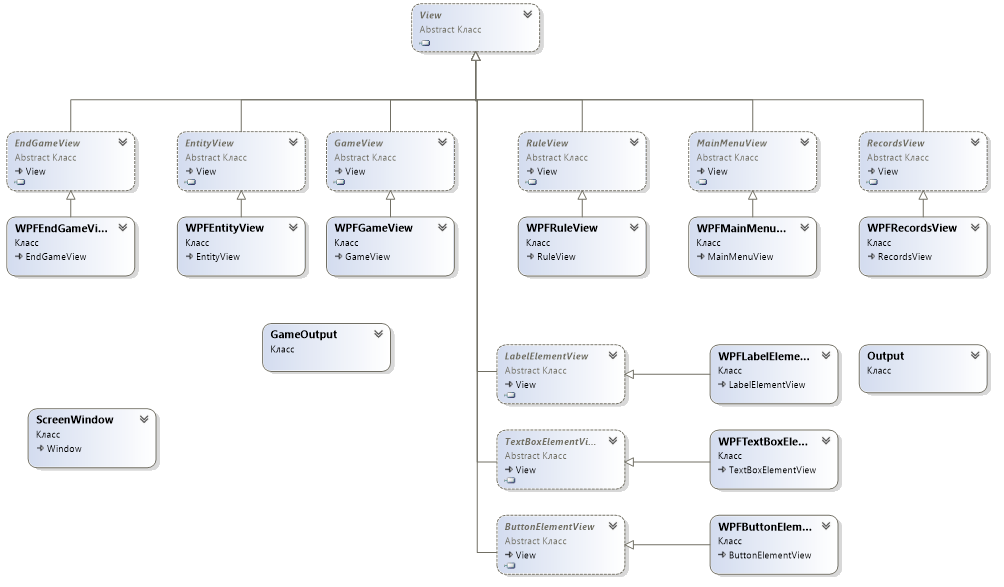


Рисунок 5 - Диаграмма классов представлений для графического приложения

## Алгоритмы

### Алгоритм при переходе в «Играть»

Диаграмма последовательности, представленная на рисунке 8, показывает взаимодействие пользователя с игровым приложением.

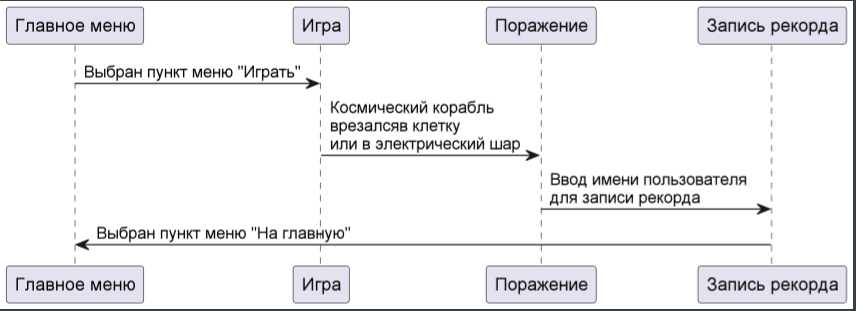


Рисунок 6 – Диаграмма последовательности при взаимодействии с игрой

При выборе пункта меню «Играть» на главном меню пользователь переходит в раздел с игрой. После поражения пользователь переходит в раздел с записью рекордов. В этом разделе пользователю необходимо ввести имя (если имя не введено, то записывается под именем «Неизвестный»). После выбора пункта меню «На главную» происходит запись рекорда и переход в главное меню.

### Алгоритм при переходе в «Правила»

При выборе пункта меню «Правила» пользователь переходит в раздел с правилами игры. Текст правил записан в файле rule.txt. На рисунке 7 показана диаграмма последовательности при взаимодействии с разделом «Правила».

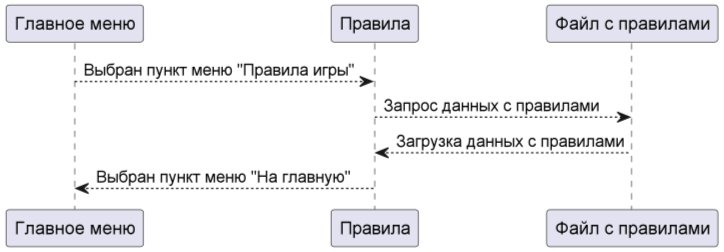


Рисунок 7 – Диаграмма последовательности при взаимодействии с разделом «Правила»

### Алгоритм при переходе в «Рекорды»

При выборе пункта меню «Рекорды» пользователь переходит в раздел с рекордами. Рекорды пользователей записаны в файл records.txt. Рекорды записаны в файл в виде (имя пользователя, тире и счет):

[имя пользователя] [-] [счет]

На рисунке 8 показана диаграмма последовательности при взаимодействии с разделом «Рекорды».

****

Рисунок 8 – Диаграмма последовательности при взаимодействии с разделом «Рекорды»

### Алгоритм для определения столкновение космического корабля с электрическим шаром

Возможные случаи столкновение космического корабля с электрическим шаром показаны на рисунке 9.

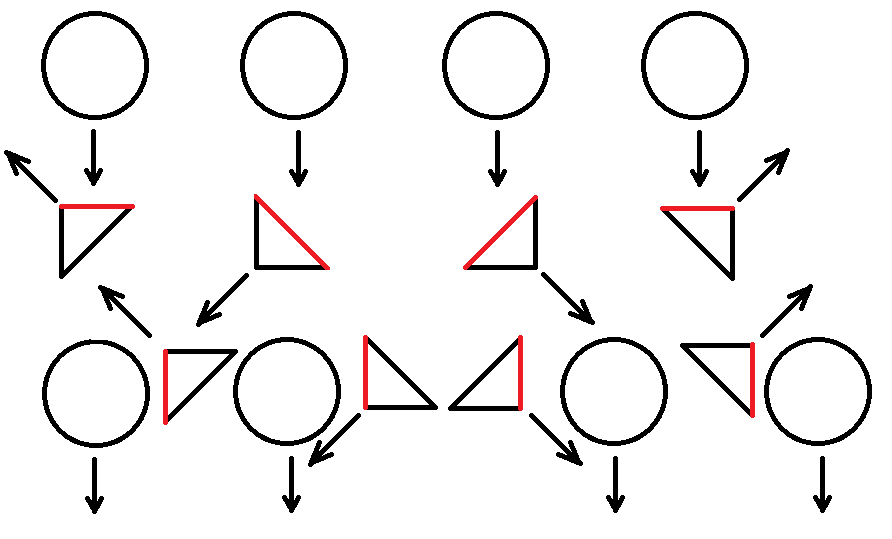


Рисунок 9 – Возможные столкновения электрического шара и космического корабля

Исходя из данного рисунка, необходимо определить случаи коллизии для окружности и отрезка.

Сначала определим, есть ли точки пересечения у окружности и линии. После чего проверим, принадлежит ли точка пересечения отрезку.

Пусть точки A(x1;y1) и B(x2;y2) – координаты начала и конца отрезка, точка O(x0;y0) – координаты центра окружности с радиусом R. Тогда точку пересечения можно вычислить, используя формулы:

Если lec > R, то точек нет.

Если lec = R, то возможна 1 точка пересечения E с координатами (ex;еу). Если ex>=x1 и ex<=x2 и ey>=y1 и ey<=y2, то точка пересечения есть, то есть столкновение электрического шара и космического корабля

Если lec < R, то возможны 2 точка пересечения F и G. Для вычисления координат точек воспользуемся формулами:

По аналогии с точкой Е определяем принадлежат ли точки нашему отрезку. Если хотя бы 1 точка принадлежит, то считаем что электрический шар столкнулся с космическим кораблем.

## Разработка интерфейса программы

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

### Интерфейс главного меню

Интерфейс главного меню представлен на рисунке 10.

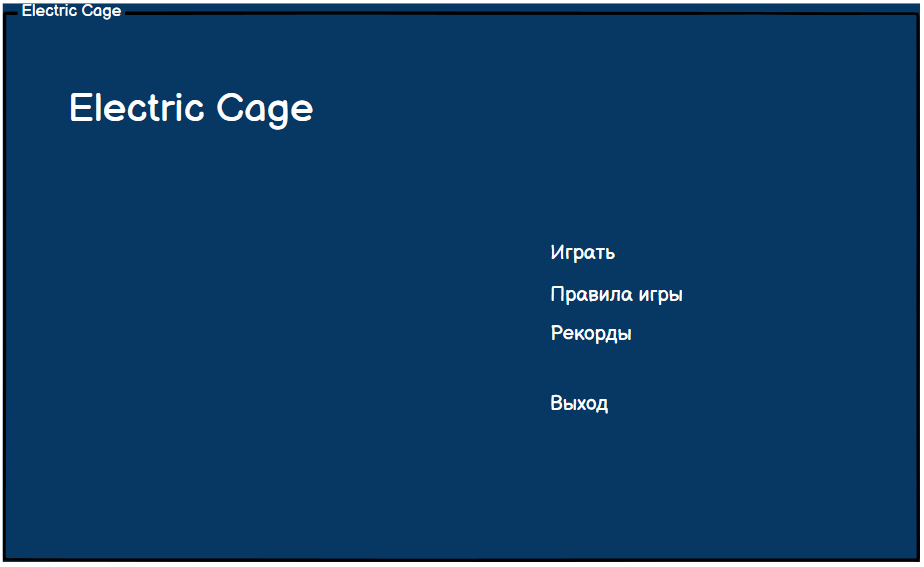


Рисунок 10 – Главное окно игры

На данном окне располагается название игры «Electric Cage» в верхнем левом углу. В правом нижнем углу расположены 4 кнопки – «Играть», «Правила игры», «Рекорды» и «Выход». Перемещение между пунктами меню осуществляется посредством нажатия на клавиши стрелок «↑» и «↓» на клавиатуре. Перемещение зациклено, т.е. при нажатии стрелки «↓», если пользователь находится на пункте «Выход», он перемещается на пункт «Играть», а при нажатии стрелки «↑», если пользователь находится на пункте «Играть», он перемещается на пункт «Выход».

При выборе пункта меню «Играть» пользователь переходит в интерфейс игры. При выборе пункта меню «Правила игры» пользователь переходит в интерфейс правил игры. При выборе пункта меню «Рекорды» пользователь перемещается в интерфейс рекордов. При выборе пункта меню «Выход» пользователь закрывает игровое приложение.

### Интерфейс игры

Интерфейс игры показан на рисунке 11.

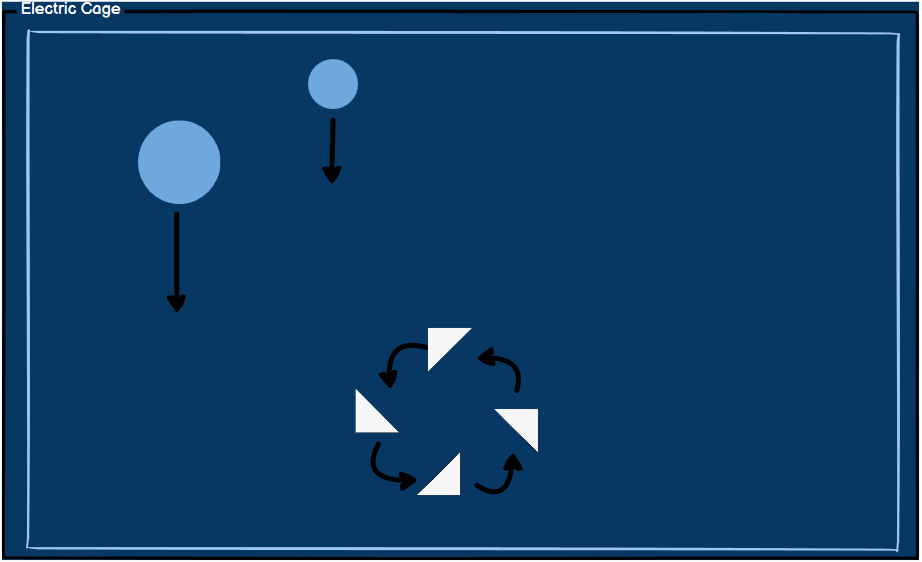


Рисунок 11 – Окно игры

После нажатия на кнопку «Играть» на главном меню, игрок переходит на игровое поле. Игрок управляет космическим кораблем, который движется в пределах игрового поля (ограничено экраном) – электрической клетки. Особенностью движения космического корабля является то, что он движется под углом 45 градусов и может поворачивать только в левую сторону. Игрок должен уворачиваться от электрических шаров, которые движутся сверху вниз. Очки рассчитываются исходя из того от какого количества электрических шаров сможет увернуться космический корабль.

### Интерфейс игры после поражения

Если космический корабль сталкивается с электрическим шаром или врезается в электрическую клетку, то игра считается оконченной. Интерфейс игры после поражения показан на рисунке 12.

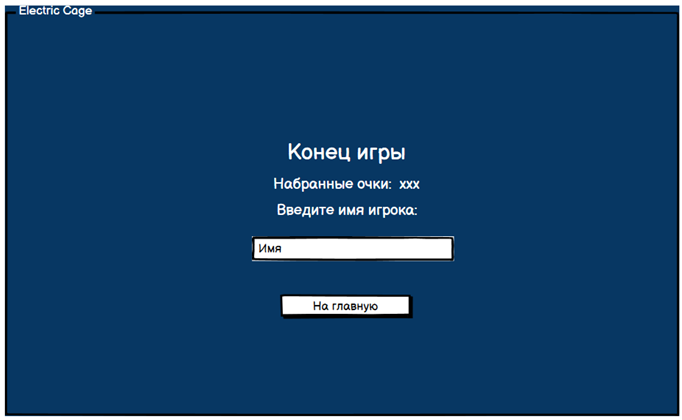


Рисунок 12 – Игровое поле игрока при входе в игру (идентификация пользователя)

Пользователь видит надпись «Конец игры» и количество набранных очков. Также пользователю предлагается ввести имя для записи рекорда. При нажатии на пункт меню «На главную» пользователь переходит в окно с главным меню приложения. Если пользователь сразу нажимает на кнопку «На главную», то в таблицу рекордов будет записан результат с именем «Неизвестный».

### Интерфейс «Правил игры»

Интерфейс приложения, после выбора пункта меню «Правила игры», показан на рисунке 13.

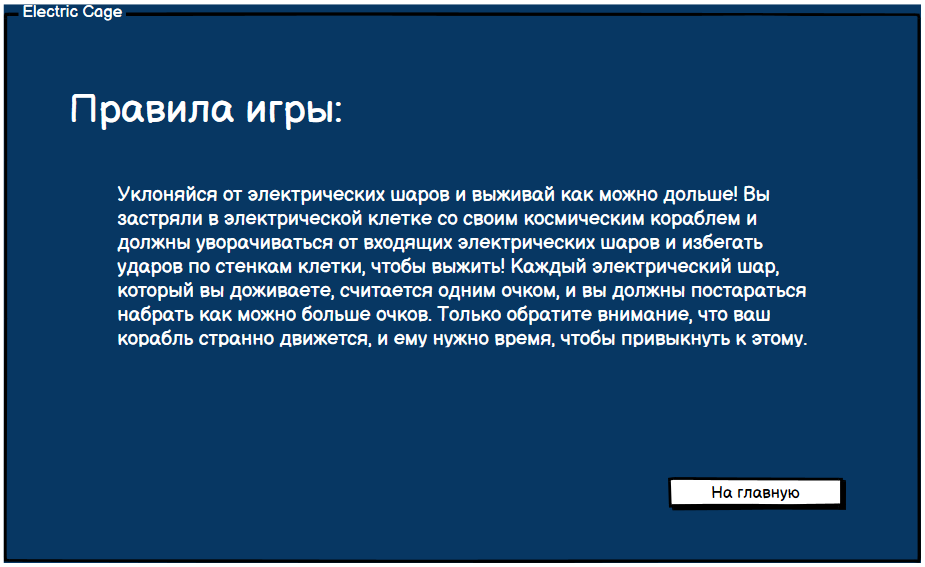


Рисунок 13 – Правила игры

При нажатии на кнопку «Правила игры», расположенную на главном окне, пользователь переходит в окно с правилами игры. В данном окне описаны правила игры, а также в нижнем правом углу окна расположена кнопка «На главную», после нажатия на которую пользователь возвращается в главное окно приложения.

### Интерфейс «Рекордов»

Интерфейс приложения, после выбора пункта меню «Рекорды», показан на рисунке 14.

После нажатия на кнопку «Рекорды», расположенную в главном окне приложения, пользователь попадает в окно с рекордами. В центре данного окна расположена таблица с именами игроков и их очками. В таблице показаны только три лучших результата. В случае если рекорды пока не записаны будет отображена пустая таблица. В нижнем правом углу окна расположена кнопка «На главную», после нажатия на которую пользователь возвращается в главное окно приложения.

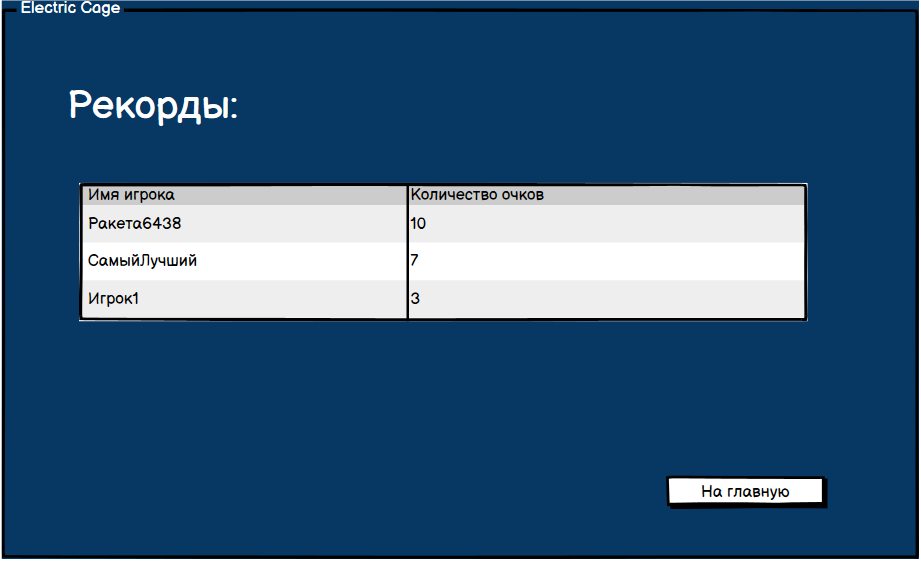


Рисунок 14 – Окно игры с рекордами

# Написание программы

## Описание разработанных процедур и функций

Таблица 1

|  |  |  |  |
| --- | --- | --- | --- |
| Класс/  интерфейс | Метод | Описание | Описание параметров |
| Model.Game.Entities.Entity | public Entity(double parX, double parY, EntitiesType parEntityType, DirectionsType parDirectionsType) | Конструктор | parX – координата х, parY – координата у, parEntityType – тип сущности, parDirectionsType – направление движения |
| public void RedrawEntity() | Вызывает событие перерисовки сущности |  |
| Model.Game.Entities.ElectricBall | public ElectricBall(double parX, double parY, EntitiesType parEntityType, DirectionsType parDirectionType) : base(parX, parY, parEntityType, parDirectionType) | Конструктор | parX – координата х, parY – координата у, parEntityType – тип сущности, parDirectionsType – направление движения |
| public void CreateNewBall() | Увеличивает NumberBall на 1 |  |
| public void Movement(double parSpeed, double parHeightScreen) | Передвигает координату у на parSpeed | parSpeed - скорость, parHeightScreen – высота экрана |
| Model.Game.Entities.SpaceShip | public SpaceShip(double parX, double parY, EntitiesType parEntityType, DirectionsType parDirectionType) : base(parX, parY, parEntityType, parDirectionType) | Конструктор | parX – координата х, parY – координата у, parEntityType – тип сущности, parDirectionsType – направление движения |
| public void Movement(double parSpeed, double parHeightScreen, double parWeightScreen) | Перемещение на parSpeed по х и у в зависимости от направления движения | parSpeed - скорость, parHeightScreen – высота экрана, parWeightScreen – ширина экрана |
| Model.Game.EndGame | public EndGame() : base() | Конструктор (задает элементы на экран поражения) |  |
| public void AddSymbol(int parSymbol) | Добавление символа в TextBoxs[0] | parSymbol – символ (код символа) |
| public void RemoveSymbol() | Удаление символа из TextBoxs[0] |  |
| public void SaveRecord() | Сохранение рекорда в файл |  |
| Model.Game.Game | public Game() : base() | Конструктор |  |
| public void Initialization() | Инициализация сущностей игры |  |
| public void StartGame() | Старт игрового цикла |  |
| public void StopGame() | Остановка игрового цикла |  |
| public void MovementEntities() | Перемещение сущностей |  |
| public void CreateNewBall (ElectricBall parElectricBall, int parMin, int parMax) | Изменение координат электрического шара | parElectricBall – электрический шар, parMin – нижняя граница координаты х, parMax – верхняя граница координаты у |
| public bool CheckBallLocation (ElectricBall parElectricBall) | Проверка положения электрического шара относительно границ | parElectricBall – электрический шар |
| public void CheckBallCollision() | Проверка столкновения космического корабля с электрическими шарами |  |
| public void RedrawEntities() | Вызов события перерисовки сущности |  |
| public void CheckFieldCollision() | Проверка столкновения космического корабля с границами поля |  |
| public bool CheckCoordinate (ElectricBall parElectricBall, int parX1, int parY1, int parX2, int parY2, int parX3, int parY3) | Проверка координат электрического шара с координатами вершин треугольника | parElectricBall – электрический шар, parX1 – первая координата х, parY1 – первая координата у, parX2 – вторая координата х, parY2 – вторая координата у, parX3 – третья координата х, parY3 – третья координата у |
| public bool CheckCoordinateBallShip (ElectricBall parElectricBall, int parX, int parY) | Проверка пересечения координат электрического шара с координатой космического корабля | parElectricBall – электрический шар, parX –координата х, parY1 –координата у космического корабля |
| public void MovementLeftUp() | Задание направления движения влево вверх |  |
| public void MovementRightUp() | Задание направления движения вправо вверх |  |
| public void MovementLeftDown() | Задание направления движения влево вниз |  |
| public void MovementRightDown() | Задание направления движения вправо вниз |  |
| Model.Menu.Elements.ButtonElement | public ButtonElement(int parNumber, string parText) | Конструктор | parNumber - номер, parText – текст кнопки |
| Model.Menu.Elements.LabelElement | public LabelElement(string parText) | Конструктор | parText – текст текстового поля |
| Model.Menu.Elements.TextBoxElement | public TextBoxElement() | Конструктор |  |
| public void ChangeText(string parText) | Изменение текста поля для ввода | parText – текст поля для ввода |
| Model.Menu.MainMenu | public MainMenu() : base() | Конструктор |  |
| Model.Records.Records | public Records() : base() | Конструктор |  |
| public void GetRecords() | Получение таблицы с рекордами |  |
| Model.Records.RecordsFile | public static void WriteRecords (string parFileName, string parText) | Запись рекорда в файл | parFileName – имя файла, parText – текст |
| public static List<Tuple<string, int>> ReadRecords(string parFileName) | Чтение рекордов из файла | parFileName – имя файла |
| Model.Rule.Rule | public Rule() : base() | Конструктор |  |
| private string[] GetRule() | Получение текста правил из файла |  |
| Model.Screen | public Screen() | Конструктор |  |
| Model.ScreenMenu | public ScreenMenu() | Конструктор |  |
| public void FocusNext() | Фокус на следующий пункт меню |  |
| public void FocusPrevious() | Фокус на предыдущий пункт меню |  |
| public void FocusButtonByNumber(int parNumber) | Фокус на пункт меню по его номеру | parNumber – номер кнопки |
| public void SelectFocusButton() | Изменение состояния кнопки на выбрана |  |
| protected void AddButtonElement (ButtonElement parButtonElement) | Добавление кнопки на экран | parButtonElement - кнопка |
| protected void AddLabelElement (LabelElement parLabelElement) | Добавление текстового поля на экран | parLabelElement – текстовое поле |
| protected void AddTextBoxElement (TextBoxElement parTextBoxElement) | Добавление поля для ввода на экран | parTextBoxElement – поле для ввода |
| protected void DeleteLabels() | Удаление всех текстовых полей с экрана |  |
| Controller.Controllers.Controllers | public Controllers() | Конструктор |  |
| public void ChangeAndStartController (MenuItemCodes parMenuItemCode) | Изменение текущего контроллера | parMenuItemCode – пункт меню |
| public void Start() | Запуск работы контроллеров |  |
| protected abstract MainMenuController GetMenuController() | Получение контроллера главного меню |  |
| protected abstract RuleController GetRuleController() | Получение контроллера правил |  |
| protected abstract RecordsController GetRecordsController() | Получение контроллера рекордов |  |
| protected abstract GameController GetGameController() | Получение контроллера игры |  |
| protected abstract EndGameController GetEndGameController() | Получение контроллера конца игры |  |
| Controller.Game.EndGameController | public EndGameController() : base() | Конструктор |  |
| Controller.Game.GameController | public GameController() : base() | Конструктор |  |
| Controller.Menu.MainMenuController | public MainMenuController() : base() | Конструктор |  |
| Controller.Records.RecordsController | public RecordsController() : base() | Конструктор |  |
| Controller.Rule.RuleController | public RuleController() : base() | Конструктор |  |
| Controller.Controller | public Controller() | Конструктор |  |
| public void ChangeCurrentController(MenuItemCodes parItemCode) | Изменение текущего контроллера | parItemCode - пункт меню |
| public abstract void Start(); | Запуск контроллера |  |
| public abstract void Stop(); | Остановка контроллера |  |
| View.Game.Entities. EntityView | public EntityView(Entity parEntity) | Конструктор | parEntity - сущность |
| protected abstract void RedrawEntity(); | Перерисовка сущности |  |
| View.Game.EndGameView | public EndGameView(EndGame parEndGame) | Конструктор | parEndGame - модель окна окончания игры |
| protected abstract void Redraw() | Перерисовка окна |  |
| protected abstract ButtonElementView CreateButtonElement(ButtonElement parButton) | Создание представления кнопки | parButton - кнопка |
| protected abstract LabelElementView CreateLabelElement(LabelElement parLabel); | Создание представления текстового поля | parLabel - текстовое поле |
| protected abstract TextBoxElementView CreateTextBoxElement(TextBoxElement parTextBox) | Создание представления поля для ввода | parTextBox - поле для ввода |
| View.Game.GameView | public GameView(Model.Game.Game parScreenGame) | Конструктор | parScreenGame - модель игры |
| protected abstract void Redraw() | Перерисовка окна |  |
| protected abstract EntityView CreateEntity(Entity parEntity); | Создание представления сущности | parEntity - сущность |
| protected void ClearEntities() | Очищение представлений сущностей |  |
| View.Menu.Elements.ButtonElementView | public ButtonElementView(ButtonElement parButton) | Конструктор | parButton - кнопка |
| protected abstract void Redraw() | Перерисовка кнопки |  |
| View.Menu.Elements.LabelElementView | public LabelElementView(LabelElement parLabel) | Конструктор | parLabel - текстовое поле |
| View.Menu.Elements.TextBoxElementView | public TextBoxElementView(TextBoxElement parTextBox) | Конструктор | parTextBox - поле для ввода |
| protected abstract void Redraw(); | Перерисовка поля для ввода |  |
| View.Menu.MainMenuView | public MainMenuView(ScreenMenu parMenu) | Конструктор | parMenu - главное меню |
| protected abstract void Redraw() | Перерисовка окна |  |
| protected abstract ButtonElementView CreateButtonElement(ButtonElement parButton) | Создание представления кнопки | parButton - кнопка |
| protected abstract LabelElementView CreateLabelElement(LabelElement parLabel); | Создание представления текстового поля | parLabel - текстовое поле |
| View.Records.RecordsView | public RecordsView(ScreenMenu parRecords) | Конструктор | parRecords - рекорды |
| protected abstract void Redraw() | Перерисовка окна |  |
| protected abstract ButtonElementView CreateButtonElement(ButtonElement parButton) | Создание представления кнопки | parButton - кнопка |
| protected abstract LabelElementView CreateLabelElement(LabelElement parLabel); | Создание представления текстового поля | parLabel - текстовое поле |
| View.Rule.RuleView | public RuleView(ScreenMenu parRule) | Конструктор | parRule – модель правил |
| protected abstract void Redraw() | Перерисовка окна |  |
| protected abstract ButtonElementView CreateButtonElement(ButtonElement parButton) | Создание представления кнопки | parButton - кнопка |
| protected abstract LabelElementView CreateLabelElement(LabelElement parLabel); | Создание представления текстового поля | parLabel - текстовое поле |
| View.View | public View() | Конструктор |  |
| public abstract void Draw(); | Отображение окна |  |
| ConsoleController.Game.ConsoleEndGameController | private ConsoleEndGameController() : base() | Коструктор |  |
| public static ConsoleEndGameController GetController() | Получение контроллера окна окончания игры |  |
| public override void Start() | Запуск контроллера |  |
| public override void Stop() | Остановка контроллера |  |
| ConsoleController.Game.ConsoleGameController | private ConsoleGameController() : base() | Коструктор |  |
| public static ConsoleGameController GetController() | Получение контроллера игры |  |
| public override void Start() | Запуск контроллера |  |
| public override void Stop() | Остановка контроллера |  |
| ConsoleController.Menu.ConsoleMainMenuController | private ConsoleMainMenuController() : base() | Коструктор |  |
| public static ConsoleMainMenuController GetController() | Получение контроллера главного меню |  |
| public override void Start() | Запуск контроллера |  |
| public override void Stop() | Остановка контроллера |  |
| ConsoleController.Records.ConsoleRecordsController | private ConsoleRecordsController() : base() | Коструктор |  |
| public static ConsoleRecordsController GetController() | Получение контроллера окна рекордов |  |
| public override void Start() | Запуск контроллера |  |
| public override void Stop() | Остановка контроллера |  |
| ConsoleController.Rule.ConsoleRuleController | private ConsoleRuleController() : base() | Коструктор |  |
| public static ConsoleRuleController GetController() | Получение контроллера правил |  |
| public override void Start() | Запуск контроллера |  |
| public override void Stop() | Остановка контроллера |  |
| ConsoleController.ConsoleControllers | public ConsoleControllers() | Коструктор |  |
| protected override EndGameController GetEndGameController() | Получает консольный контроллер окна окончания игры |  |
| protected override GameController GetGameController() | Получает консольный контроллер окна игры |  |
| protected override RuleController GetRuleController() | Получает консольный контроллер окна правил |  |
| protected override MainMenuController GetMenuController() | Получает консольный контроллер окна главного меню |  |
| protected override RecordsController GetRecordsController() | Получает консольный контроллер окна рекордов |  |
| ConsoleView.Game.Entities.ConsoleEntityView | public ConsoleEntityView(Entity parEntity) : base(parEntity) | Конструктор | parEntity - сущность |
| public override void Draw() | Отображение сущности |  |
| protected override void RedrawEntity() | Перерисовывание сущности |  |
| ConsoleView.Game.ConsoleEndGameView | public ConsoleEndGameView(EndGame parEndGameScreen) : base(parEndGameScreen) | Конструктор | parEndGameScreen - модель окна конца игры |
| public override void Draw() | Отображение окна конца игры |  |
| private void Init() | Инициализация координат объектов окна |  |
| protected override void Redraw() | Перерисовка окна окончания игры |  |
| protected override ButtonElementView CreateButtonElement(ButtonElement parButtonElement) | Создание консольного представления кнопки | parButtonElement - кнопка |
| protected override LabelElementView CreateLabelElement(LabelElement parLabelElement) | Создание консольного представления текстового поля | parLabelElement - текстовое поле |
| protected override TextBoxElementView CreateTextBoxElement(TextBoxElement parTextBox) | Создание консольного представления поля для ввода | parTextBox - поле для ввода |
| ConsoleView.Game.ConsoleGameView | public ConsoleGameView(Model.Game.Game parGameScreen) : base(parGameScreen) | Конструктор | parGameScreen - окно игры |
| public override void Draw() | Отображение игры |  |
| protected override EntityView CreateEntity(Entity parEntity) | Создание представления для сущности | parEntity - сущность |
| protected override void Redraw() | Перерисовка сущностей |  |
| ConsoleView.Game.GameOutput | private GameOutput() |  |  |
| public void DrawEntityView(Entity parEntity, int parX, int parY) | Отображение сущностей | parEntity – сущность,  parX - координата х, parY - координата у |
| public void Redraw(Entity parEntity, int parOldX, int parOldY, int parNewX, int parNewY) | Перерисовывание сущностей | parEntity – сущность, parOldX - предыдущая координата х, parOldY - предыдущая координата у, parNewX - новая координата х, parNewY - новая координата у |
| private void PrintSpaceShip(int parX, int parY, DirectionsType parDirectionType) | Вывод на консоль космического корабля | parX - координата х, parY – координата у, parDirectionType – направление движения |
| public void PrintGameField() | Вывод на консоль границ |  |
| private void PrintElectricBall(int parX, int parY) | Вывод на консоль электрического шара | parX - координата х, parY – координата у |
| private void ClearSpaceShip(int parX, int parY, DirectionsType parDirectionType) | Удаление с консоли космического корабля | parX - координата х, parY – координата у, parDirectionType – направление движения |
| public void ClearElectricBall(int parX, int parY) | Удаление с консоли электрического шара | parX - координата х, parY – координата у |
| ConsoleView.Menu.Elements.ConsoleButtonElementView | public ConsoleButtonElementView(ButtonElement parButtonElement) : base(parButtonElement) | Конструктор | parButtonElement - кнопка |
| public override void Draw() | Отображение кнопки |  |
| protected override void Redraw() | Перерисовывание кнопки |  |
| ConsoleView.Menu.Elements.ConsoleLabelElementView | public ConsoleLabelElementView (LabelElement parLabelElement) : base(parLabelElement) | Конструктор | parLabelElement - текстовое поле |
| public override void Draw() | Отображение текстового поля |  |
| ConsoleView.Menu.Elements.ConsoleTextBoxElementView | public ConsoleTextBoxElementView (TextBoxElement parTextBox) : base(parTextBox) | Коструктор | parTextBox - поле для ввода |
| public override void Draw() | Отображение поля для ввода |  |
| protected override void Redraw() | Перерисовывания поля для ввода |  |
| ConsoleView.Menu.ConsoleMenuView | public ConsoleMenuView(ScreenMenu parMenu) : base(parMenu) | Конструктор | parMenu – главное меню |
| public override void Draw() | Отображение окна главного меню |  |
| protected override ButtonElementView CreateButtonElement(ButtonElement parButtonElement) | Создание консольного представления кнопки | parButtonElement - кнопка |
| protected override LabelElementView CreateLabelElement(LabelElement parLabelElement) | Создание консольного представления текстового поля | parLabelElement - текстовое поле |
| protected override void Redraw() | Перерисовывание окна главного меню |  |
| private void Init() | Инициализация главного меню |  |
| ConsoleView.Menu.Output | public Output() | Конструктор |  |
| public void OutputButton(string parText, int parX, int parY, States parState) | Вывод кнопки на консоль | parText – текст кнопки, parX - координата х, parY - координата у, parState - состояние |
| public void PrintGameTitle() | Вывод заголовка |  |
| public void OutputString(string parString, int parX, int parY) | Вывод строки | parString - строка, parX - координата х, parY - координата у |
| ConsoleView.Records.ConsoleRecordsView | public ConsoleRecordsView (ScreenMenu parRecords) : base(parRecords) | Конструктор | parRecords - рекорды |
| public override void Draw() | Отображение окна рекордов |  |
| protected override ButtonElementView CreateButtonElement(ButtonElement parButtonElement) | Создание консольного представления кнопки | parButtonElement - кнопка |
| protected override LabelElementView CreateLabelElement(LabelElement parLabelElement) | Создание консольного представления текстового поля | parLabelElement - текстовое поле |
| protected override void Redraw() | Перерисовывание окна рекордов |  |
| private void Init() | Инициализация окна рекордов |  |
| ConsoleView.Rule.ConsoleRuleView | public ConsoleRuleView(ScreenMenu parRule) : base(parRule) | Конструктор | parRule - правила |
| public override void Draw() | Отображение окна правил |  |
| protected override ButtonElementView CreateButtonElement(ButtonElement parButtonElement) | Создание консольного представления кнопки | parButtonElement - кнопка |
| protected override LabelElementView CreateLabelElement(LabelElement parLabelElement) | Создание консольного представления текстового поля | parLabelElement - текстовое поле |
| protected override void Redraw() | Перерисовывание окна правил |  |
| private void Init() | Инициализация окна правил |  |
| WPFController.Game.WPFEndGameController | private WPFEndGameController() : base() | Конструктор |  |
| public static WPFEndGameController GetController() | Получение контроллера окна окончания игры |  |
| public override void Start() | Запуск контроллера |  |
| public override void Stop() | Остановка контроллера |  |
| public void OnKeyDown(object sender, KeyEventArgs e) | Нажатие клавиши с клавиатуры |  |
| WPFController.Game.WPFGameController | private WPFGameController() : base() | Конструктор |  |
| public static WPFGameController GetController() | Получение контроллера окна игры |  |
| public override void Start() | Запуск контроллера |  |
| public override void Stop() | Остановка контроллера |  |
| private void EndGame() | Завершение игры |  |
| public void OnKeyDownHandler(object sender, KeyEventArgs e) | Нажатие клавиши с клавиатуры |  |
| WPFController.Menu.WPFMainMenuController | private WPFMainMenuController() : base() | Конструктор |  |
| public static WPFMainMenuController GetController() | Получение контроллера главного меню |  |
| public override void Start() | Запуск контроллера |  |
| public override void Stop() | Остановка контроллера |  |
| public void OnKeyDown(object sender, KeyEventArgs e) | Нажатие клавиши с клавиатуры |  |
| WPFController.Records.WPFRecordsController | private WPFRecordsController() : base() | Конструктор |  |
| public static WPFRecordsController GetController() | Получение контроллера окна рекордов |  |
| public override void Start() | Запуск контроллера |  |
| public override void Stop() | Остановка контроллера |  |
| public void OnKeyDown(object sender, KeyEventArgs e) | Нажатие клавиши с клавиатуры |  |
| WPFController.Rule.WPFRuleController | private WPFRuleController() : base() | Конструктор |  |
| public static WPFRuleController GetController() | Получение контроллера правил |  |
| public override void Start() | Запуск контроллера |  |
| public override void Stop() | Остановка контроллера |  |
| public void OnKeyDown(object sender, KeyEventArgs e) | Нажатие клавиши с клавиатуры |  |
| WPFController.WPFControllers | public WPFControllers() |  |  |
| protected override EndGameController GetEndGameController() | Получает контроллер окна окончания игры |  |
| protected override GameController GetGameController() | Получает контроллер окна игры |  |
| protected override RuleController GetRuleController() | Получает контроллер окна правил |  |
| protected override MainMenuController GetMenuController() | Получает контроллер окна главного меню |  |
| protected override RecordsController GetRecordsController() | Получает контроллер окна рекордов |  |
| WPFView.Game.Entities.WPFEntityView | public WPFEntityView(Entity parEntity) : base(parEntity) | Конструктор | parEntity - сущность |
| public override void Draw() | Отображение сущности |  |
| protected override void RedrawEntity() | Перерисовка сущности |  |
| public void AddChildEntity (FrameworkElement parScreen) | Добавление сущности в окно | parScreen - окно |
| WPFView.Game.GameOutput | public static Shape CreateViewEntity(Entity parEntity) | Создание представления сущности | parEntity - сущность |
| private static Shape CreateRectangle(double parX, double parY, double parWidth, double parHeight) | Создание космического корабля | parX - координата х, parY - координата у, parWidth - ширина, parHeight - высота |
| private static Shape CreateElectricBall(double parX, double parY, double parWidth, double parHeight) | Создание электрического шара | parX - координата х, parY - координата у, parWidth - ширина, parHeight - высота |
| public static Shape CreateField (double parX, double parY, double parWidth, double parHeight) | Создание границ поля | parX - координата х, parY - координата у, parWidth - ширина, parHeight - высота |
| WPFView.Game.WPFEndGameView | public WPFEndGameView(EndGame parEndGameScreen) : base(parEndGameScreen) | Конструктор | parEndGameScreen - модель окна окончания игры |
| public override void Draw() | Отображение окна конца игры |  |
| protected override ButtonElementView CreateButtonElement(ButtonElement parButtonElement) | Создание представления кнопки | parButtonElement - кнопка |
| protected override LabelElementView CreateLabelElement(LabelElement parLabelElement) | Создание представления текстового поля | parLabelElement - текстовое поле |
| protected override TextBoxElementView CreateTextBoxElement(TextBoxElement parTextBox) | Создание представления поля для ввода | parTextBox - поле для ввода |
| protected override void Redraw() | Перерисовка окна окончания игры |  |
| private void Init() | Инициализация окна конца игры |  |
| private void AddScreen(FrameworkElement parScreen) | Добавление элементов на экран | parScreen - экран |
| WPFView.Game.WPFGameView | public WPFGameView(Model.Game.Game parGameScreen) : base(parGameScreen) | Конструктор | parGameScreen - модель игры |
| public override void Draw() | Отображение игры |  |
| protected override EntityView CreateEntity(Entity parEntity) | Создание представления для сущности | parEntity - сущность |
| protected override void Redraw() | Перерисовывание сущностей |  |
| private void AddScreen(FrameworkElement parScreen) | Добавление сущностей на экран | parScreen - экран |
| WPFView.Menu.Elements.WPFButtonElementView | public WPFButtonElementView (ButtonElement parButtonElement) : base(parButtonElement) | Конструктор | parButtonElement - модель кнопки |
| private void Init() | Инициализация кнопки |  |
| public override void Draw() | Отображение кнопки |  |
| protected override void Redraw() | Перерисовывание кнопки |  |
| private void ChangeColourButton() | Изменение цвета текста в зависимости от состояния кнопки |  |
| public void AddChildButton(FrameworkElement parScreen) | Добавление кнопки на экран | parScreen - экран |
| WPFView.Menu.Elements.WPFLabelElementView | public WPFLabelElementView (LabelElement parLabelElement) : base(parLabelElement) | Конструктор | parLabelElement - модель текстового поля |
| public override void Draw() | Отображение текстового поля |  |
| public void AddChildLabel(FrameworkElement parScreen) | Добавление текстового поля на экран | parScreen - экран |
| WPFView.Menu.Elements.WPFTextBoxElementView | public WPFTextBoxElementView(TextBoxElement parTextBox) : base(parTextBox) | Конструктор | parTextBox - модель поля для ввода |
| public override void Draw() | Отображение поля для ввода |  |
| public void AddChildTextBox(FrameworkElement parScreen) | Добавление поля для ввода на экран | parScreen - экран |
| protected override void Redraw() | Перерисовывание поля для ввода |  |
| private void Init() | Инициализация поля для ввода |  |
| WPFView.Menu.Output | public Output() | Конструктор |  |
| public Button CreateButton(int parX, int parY) | Создание представления кнопки | parX - координата х, parY - координата у |
| public TextBlock CreateTextBlock (int parX, int parY, string parText, int parSize) | Создание представления текстового поля | parX - координата х, parY - координата у, parText - текст, parSize - размер |
| public Label CreateLabel(int parX, int parY, int parSize, int parHeight, int parWidth) | Создание представления поля для ввода | parX - координата х, parY - координата у, parSize – размер текста, parHeight -высота, parWidth - ширина |
| WPFView.Menu.WPFMainMenuView | public WPFMainMenuView(ScreenMenu parMenu) : base(parMenu) | Конструктор | parMenu - модель окна главного меню |
| public override void Draw() | Отображение главного меню |  |
| protected override void Redraw() | Перерисовывание окна главного меню |  |
| protected override ButtonElementView CreateButtonElement(ButtonElement parButtonElement) | Создание представления кнопки | parButtonElement - кнопка |
| protected override LabelElementView CreateLabelElement(LabelElement parLabelElement) | Создание представления текстового поля | parLabelElement - текстовое поле |
| private void Init() | Инициализация окна главного меню |  |
| private void AddScreen(FrameworkElement parScreen) | Добавление элементов на экран | parScreen - экран |
| WPFView.Records.WPFRecordsView | public WPFRecordsView(ScreenMenu parRecords) : base(parRecords) | Конструктор | parRecords - модель окна рекордов |
| public override void Draw() | Отображение окна рекордов |  |
| protected override ButtonElementView CreateButtonElement(ButtonElement parButtonElement) | Создание представления кнопки | parButtonElement - кнопка |
| protected override LabelElementView CreateLabelElement(LabelElement parLabelElement) | Создание представления текстового поля | parLabelElement - текстовое поле |
| protected override void Redraw() | Перерисовывание окна рекордов |  |
| private void Init() | Инициализация окна рекордов |  |
| private void AddScreen(FrameworkElement parScreen) | Добавление элементов на экран | parScreen - экран |
| WPFView.Rule.WPFRuleView | public WPFRuleView(ScreenMenu parRule) : base(parRule) | Конструктор | parRule - модель окна правил |
| public override void Draw() | Отображение окна правил |  |
| protected override ButtonElementView CreateButtonElement(ButtonElement parButtonElement) | Создание представления кнопки | parButtonElement - кнопка |
| protected override LabelElementView CreateLabelElement(LabelElement parLabelElement) | Создание представления текстового поля | parLabelElement - текстовое поле |
| protected override void Redraw() | Перерисовывание окна правил |  |
| private void Init() | Инициализация окна правил |  |
| private void AddScreen(FrameworkElement parScreen) | Добавление элементов на экран | parScreen - экран |
| WPFView.ScreenWindow | private ScreenWindow() | Конструктор |  |
| public static ScreenWindow GetWindowScreen() | Получение окна |  |

## Разработка программы

### Описание классов, перечислений и интерфейсов проекта

Таблица 2

|  |  |  |
| --- | --- | --- |
| Пространство имен | Название класса / интерфейса / перечисления | Назначение |
| Model.Enums | DirectionsType | Перечисление направлений движения сущностей |
| Model.Enums | EntitiesType | Типы сущностей |
| Model.Enums | MenuItemCodes | Пункты меню |
| Model.Enums | States | Состояние пункта меню |
| Model.Game.Entities | Entity | Игровые сущности |
| Model.Game.Entities | ElectricBall | Электрический шар |
| Model.Game.Entities | SpaceShip | Космический корабль |
| Model.Game | EndGame | Модель окна окончания игры |
| Model.Game | Game | Модель окна игры |
| Model.Menu.Elements | ButtonElement | Модель кнопки |
| Model.Menu.Elements | LabelElement | Модель текстового поля |
| Model.Menu.Elements | TextBoxElement | Модель поля для ввода |
| Model.Menu | MainMenu | Модель главного меню |
| Model.Records | Records | Модель окна рекордов |
| Model.Records | RecordsFile | Запись и чтение из файла рекордов |
| Model.Rule | Rule | Модель окна правил |
| Model | Screen | Модель окна |
| Model | ScreenMenu | Модель окна меню |
| Controller.Controllers | Controllers | Управление контроллерами |
| Controller.Game | EndGameController | Управление окном окончания игры |
|  | GameController | Управление окном игры |
| Controller.Menu | MainMenuController | Управление окном главного меню |
| Controller.Records | RecordsController | Управление окном рекордов |
| Controller.Rule | RuleController | Управление окном правил |
| Controller | Controller | Базовый класс контроллеров |
| View.Game.Entities | EntityView | Представление сущности |
| View.Game | EndGameView | Представление окна окончания игры |
| View.Game | GameView | Представление окна игры |
| View.Menu.Elements | ButtonElementView | Представление кнопки |
| View.Menu.Elements | LabelElementView | Представление текстового поля |
| View.Menu.Elements | TextBoxElementView | Представление поля для ввода |
| View.Menu | MainMenuView | Представление окна главного меню |
| View.Records | RecordsView | Представление окна рекордов |
| View.Rule | RuleView | Представление окна правил |
| View | View | Базовое представление |
| ConsoleController.Game | ConsoleEndGameController | Управление окном окончания игры с консоли |
| ConsoleController.Game | ConsoleGameController | Управление окном игры с консоли |
| ConsoleController.Menu | ConsoleMainMenuController | Управление окном главного меню с консоли |
| ConsoleController.Records | ConsoleRecordsController | Управление окном рекордов с консоли |
| ConsoleController.Rule | ConsoleRuleController | Управление окном правил с консоли |
| ConsoleController | ConsoleControllers | Управление контроллерами с консоли |
| ConsoleView.Game.Entities | ConsoleEntityView | Представление сущности на консоли |
| ConsoleView.Game | ConsoleEndGameView | Представление окна окончания игры на консоли |
| ConsoleView.Game | ConsoleGameView | Представление окна игры на консоли |
| ConsoleView.Game | GameOutput | Вывод игровых сущностей на консоль |
| ConsoleView.Menu.Elements | ConsoleButtonElementView | Представление кнопки на консоли |
| ConsoleView.Menu.Elements | ConsoleLabelElementView | Представление текстового поля на консоли |
| ConsoleView.Menu.Elements | ConsoleTextBoxElementView | Представление поля для ввода на консоли |
| ConsoleView.Menu | ConsoleMenuView | Представление окна главного меню на консоли |
| ConsoleView.Menu | Output | Вывод элементов на консоль |
| ConsoleView.Records | ConsoleRecordsView | Представление окна рекордов на консоли |
| ConsoleView.Rule | ConsoleRuleView | Представление окна правил на консоли |
| WPFController.Game | WPFEndGameController | Управление окном окончания игры для WPF |
| WPFController.Game | WPFGameController | Управление окном игры для WPF |
| WPFController.Menu | WPFMainMenuController | Управление окном главного меню для WPF |
| WPFController.Records | WPFRecordsController | Управление окном рекордов для WPF |
| WPFController.Rule | WPFRuleController | Управление окном правил для WPF |
| WPFController | WPFControllers | Управление контроллерами для WPF |
| WPFView.Game.Entities | WPFEntityView | Графическое представление сущности |
| WPFView.Game | GameOutput | Создание фигур для сущностей |
| WPFView.Game | WPFEndGameView | Графическое представление окна окончания игры |
| WPFView.Game | WPFGameView | Графическое представление окна игры |
| WPFView.Menu.Elements | WPFButtonElementView | Графическое представление кнопки |
| WPFView.Menu.Elements | WPFLabelElementView | Графическое представление текстового поля |
| WPFView.Menu.Elements | WPFTextBoxElementView | Графическое представление поля для ввода |
| WPFView.Menu | Output | Создание графических представлений для элементов |
| WPFView.Menu | WPFMainMenuView | Графическое представление окна главного меню |
| WPFView.Records | WPFRecordsView | Графическое представление окна рекордов |
| WPFView.Rule | WPFRuleView | Графическое представление окна правил |
| WPFView | ScreenWindow | Создание окна |

## Описание шаблонов проектирования, которые использовались при написании программы

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

MVC - шаблон проектирования, в рамках которого приложение разделяется на три компонента: модель, представление и контроллер.

Этот шаблон реализуется в классах следующих проектов:

* + Model – содержит модель и бизнес-логику приложения.
  + ConsoleView, WPFView – представления, которые предоставляют интерфейсы для консольного и графического приложения соответственно.
  + ConsoleController, WPFController – контроллеры, которые обрабатывают действия пользователя в консольном и графическом приложении соответственно.

### 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 в классах ConsoleEndGameController, ConsoleGameController, ConsoleMainMenuController, ConsoleRecordsController, ConsoleRuleController, WPFEndGameController, WPFGameController, WPFMainMenuController, WPFRecordsController, WPFRuleController.

### Фабричный метод (Factory Method)

Фабричный метод (Factory Method) – порождающий шаблон проектирования, в котором классы всех конечных типов наследуются от одного абстрактного базового класса, предназначенного для полиморфного использования. Таким образом, данный шаблон определяет интерфейс для создания объектов некоторого класса, но непосредственное решение о том, объект какого класса создавать происходит в подклассах. То есть шаблон предполагает, что базовый класс делегирует создание объектов классам-наследникам.

Этот шаблон используется для создания контроллеров Controllers – базовый класс, ConsoleControllers, WPFControllers – наследники базового класса.

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

### Извлечение метода из другого метода

Описание вида рефакторинга: фрагмент метода выносится в отдельный метод.

Пример: столкновение электрических шаров и космического корабля должно рассчитываться для каждого шара и космического корабля. Таким образом, будет повторяться один и тот же код по проверке координат для каждого шара, но с разными координатами шара. Метод проверки координат вынесен в отдельные методы.

Фрагмент кода до рефакторинга:

/// <summary>

/// Проверка столкновения космического корабля с электрическими шарами

/// </summary>

public void CheckBallCollision()

{

int x1 = (int)Math.Round(SpaceShip.X, MidpointRounding.AwayFromZero);

int y1 = (int)Math.Round(SpaceShip.Y, MidpointRounding.AwayFromZero);

int x2 = 0;

int y2 = 0;

int x3 = 0;

int y3 = 0;

switch (SpaceShip.DirectionType)

{

case Enums.DirectionsType.LeftUp:

x2 = (int)Math.Round(SpaceShip.X+1, MidpointRounding.AwayFromZero);

y2 = (int)Math.Round(SpaceShip.Y, MidpointRounding.AwayFromZero);

x3 = (int)Math.Round(SpaceShip.X, MidpointRounding.AwayFromZero);

y3 = (int)Math.Round(SpaceShip.Y+1, MidpointRounding.AwayFromZero);

break;

case Enums.DirectionsType.LeftDown:

x2 = (int)Math.Round(SpaceShip.X+1, MidpointRounding.AwayFromZero);

y2 = (int)Math.Round(SpaceShip.Y, MidpointRounding.AwayFromZero);

x3 = (int)Math.Round(SpaceShip.X, MidpointRounding.AwayFromZero);

y3 = (int)Math.Round(SpaceShip.Y-1, MidpointRounding.AwayFromZero);

break;

case Enums.DirectionsType.RightDown:

x2 = (int)Math.Round(SpaceShip.X-1, MidpointRounding.AwayFromZero);

y2 = (int)Math.Round(SpaceShip.Y, MidpointRounding.AwayFromZero);

x3 = (int)Math.Round(SpaceShip.X, MidpointRounding.AwayFromZero);

y3 = (int)Math.Round(SpaceShip.Y-1, MidpointRounding.AwayFromZero);

break;

case Enums.DirectionsType.RightUp:

x2 = (int)Math.Round(SpaceShip.X-1, MidpointRounding.AwayFromZero);

y2 = (int)Math.Round(SpaceShip.Y, MidpointRounding.AwayFromZero);

x3 = (int)Math.Round(SpaceShip.X, MidpointRounding.AwayFromZero);

y3 = (int)Math.Round(SpaceShip.Y+1, MidpointRounding.AwayFromZero);

break;

}

if (CheckCoordinateBallShip (\_electricBallFirst, x1, y1))

{

Score = \_electricBalls.Count()-4;

StopGame();

End?.Invoke();

}

if (CheckCoordinateBallShip (\_electricBallFirst, x2, y2))

{

Score = \_electricBalls.Count()-4;

StopGame();

End?.Invoke();

}

if (CheckCoordinateBallShip (\_electricBallFirst, x3, y3))

{

Score = \_electricBalls.Count()-4;

StopGame();

End?.Invoke();

}

if (CheckCoordinateBallShip (\_electricBallSecond, x1, y1))

{

Score = \_electricBalls.Count()-4;

StopGame();

End?.Invoke();

}

if (CheckCoordinateBallShip (\_electricBallSecond, x2, y2))

{

Score = \_electricBalls.Count()-4;

StopGame();

End?.Invoke();

}

if (CheckCoordinateBallShip (\_electricBallSecond, x3, y3))

{

Score = \_electricBalls.Count()-4;

StopGame();

End?.Invoke();

}

if (CheckCoordinateBallShip (\_electricBallThird, x1, y1))

{

Score = \_electricBalls.Count()-4;

StopGame();

End?.Invoke();

}

if (CheckCoordinateBallShip (\_electricBallThird, x2, y2))

{

Score = \_electricBalls.Count()-4;

StopGame();

End?.Invoke();

}

if (CheckCoordinateBallShip (\_electricBallThird, x3, y3))

{

Score = \_electricBalls.Count()-4;

StopGame();

End?.Invoke();

}

if (CheckCoordinateBallShip (\_electricBallFourth, x1, y1))

{

Score = \_electricBalls.Count()-4;

StopGame();

End?.Invoke();

}

if (CheckCoordinateBallShip (\_electricBallFourth, x2, y2))

{

Score = \_electricBalls.Count()-4;

StopGame();

End?.Invoke();

}

if (CheckCoordinateBallShip (\_electricBallFourth, x3, y3))

{

Score = \_electricBalls.Count()-4;

StopGame();

End?.Invoke();

}

}

Фрагмент кода после рефакторинга:

/// <summary>

/// Проверка столкновения космического корабля с электрическими шарами

/// </summary>

public void CheckBallCollision()

{

int x1 = (int)Math.Round(SpaceShip.X, MidpointRounding.AwayFromZero);

int y1 = (int)Math.Round(SpaceShip.Y, MidpointRounding.AwayFromZero);

int x2 = 0;

int y2 = 0;

int x3 = 0;

int y3 = 0;

switch (SpaceShip.DirectionType)

{

case Enums.DirectionsType.LeftUp:

x2 = (int)Math.Round(SpaceShip.X+1, MidpointRounding.AwayFromZero);

y2 = (int)Math.Round(SpaceShip.Y, MidpointRounding.AwayFromZero);

x3 = (int)Math.Round(SpaceShip.X, MidpointRounding.AwayFromZero);

y3 = (int)Math.Round(SpaceShip.Y+1, MidpointRounding.AwayFromZero);

break;

case Enums.DirectionsType.LeftDown:

x2 = (int)Math.Round(SpaceShip.X+1, MidpointRounding.AwayFromZero);

y2 = (int)Math.Round(SpaceShip.Y, MidpointRounding.AwayFromZero);

x3 = (int)Math.Round(SpaceShip.X, MidpointRounding.AwayFromZero);

y3 = (int)Math.Round(SpaceShip.Y-1, MidpointRounding.AwayFromZero);

break;

case Enums.DirectionsType.RightDown:

x2 = (int)Math.Round(SpaceShip.X-1, MidpointRounding.AwayFromZero);

y2 = (int)Math.Round(SpaceShip.Y, MidpointRounding.AwayFromZero);

x3 = (int)Math.Round(SpaceShip.X, MidpointRounding.AwayFromZero);

y3 = (int)Math.Round(SpaceShip.Y-1, MidpointRounding.AwayFromZero);

break;

case Enums.DirectionsType.RightUp:

x2 = (int)Math.Round(SpaceShip.X-1, MidpointRounding.AwayFromZero);

y2 = (int)Math.Round(SpaceShip.Y, MidpointRounding.AwayFromZero);

x3 = (int)Math.Round(SpaceShip.X, MidpointRounding.AwayFromZero);

y3 = (int)Math.Round(SpaceShip.Y+1, MidpointRounding.AwayFromZero);

break;

}

if (CheckCoordinate(\_electricBallFirst, x1, y1, x2, y2, x3, y3))

{

Score = \_electricBalls.Count()-4;

StopGame();

End?.Invoke();

}

if (CheckCoordinate(\_electricBallSecond, x1, y1, x2, y2, x3, y3))

{

Score = \_electricBalls.Count()-4;

StopGame();

End?.Invoke();

}

if (CheckCoordinate(\_electricBallThird, x1, y1, x2, y2, x3, y3))

{

Score = \_electricBalls.Count()-4;

StopGame();

End?.Invoke();

}

if (CheckCoordinate(\_electricBallFourth, x1, y1, x2, y2, x3, y3))

{

Score = \_electricBalls.Count()-4;

StopGame();

End?.Invoke();

}

}

/// <summary>

/// Проверка координат электрического шара и космического корабля

/// </summary>

/// <param name="parElectricBall">электрический шар</param>

/// <param name="parX1">первая координата х</param>

/// <param name="parY1">первая координата у</param>

/// <param name="parX2">вторая координата х</param>

/// <param name="parY2">вторая координата у</param>

/// <param name="parX3">третья координата х</param>

/// <param name="parY3">третья координата у</param>

/// <returns>было ли столкновение</returns>

public bool CheckCoordinate(ElectricBall parElectricBall,

int parX1, int parY1, int parX2, int parY2, int parX3, int parY3)

{

if (CheckCoordinateBallShip(parElectricBall, parX1, parY1))

{

return true;

}

if (CheckCoordinateBallShip(parElectricBall, parX2, parY2))

{

return true;

}

if (CheckCoordinateBallShip(parElectricBall, parX3, parY3))

{

return true;

}

return false;

}

### Присвоение методу более информативного имени

Описание вида рефакторинга: присвоение методу имени, которое поясняет его назначение.

Пример: метод проверяющий столкновение между электрическим шаром и космическим кораблем.

Фрагмент кода до рефакторинга:

public bool CheckCoordinate(ElectricBall parElectricBall,

int parX1, int parY1, int parX2, int parY2, int parX3, int parY3)

{

...

}

Фрагмент кода после рефакторинга:

public bool CheckCollisionElectricBallCoordinateShip(ElectricBall parElectricBall,

int parX1, int parY1, int parX2, int parY2, int parX3, int parY3)

{

...

}

### Замена магического числа на именованную константу (переменную)

Описание вида рефакторинга: численный или строковый литерал заменяется его именованной константой / переменной.

Пример: скорость космического корабля задается через переменную \_speedSpaceShip.

Фрагмент кода до рефакторинга:

public void MovementEntities()

{

SpaceShip.Movement(4 \* MoveTime, HeightScreen, WidthScreen);

...

}

Фрагмент кода после рефакторинга:

/// <summary>

/// Скорость космического корабля

/// </summary>

private int \_speedSpaceShip = 4;

public void MovementEntities()

{

SpaceShip.Movement(SpeedSpaceShip \* MoveTime, HeightScreen, WidthScreen);

...

}

### Введение промежуточной переменной.

Описание вида рефакторинга: результат вычисления выражения присваивается промежуточной переменной, имя которой резюмирует суть выражения.

Пример: координату центра электрического шара сделать переменной.

Фрагмент кода до рефакторинга:

public bool CheckCoordinateBallShip(ElectricBall parElectricBall, int parX, int parY)

{

if (((int)Math.Round(parElectricBall.X, MidpointRounding.AwayFromZero) == parX && (int)Math.Round(parElectricBall.Y, MidpointRounding.AwayFromZero)+1 == parY) || ((int)Math.Round(parElectricBall.X, MidpointRounding.AwayFromZero)-1 == parX && (int)Math.Round(parElectricBall.Y, MidpointRounding.AwayFromZero)+1 == parY) || ((int)Math.Round(parElectricBall.X, MidpointRounding.AwayFromZero)+1 == parX && (int)Math.Round(parElectricBall.Y, MidpointRounding.AwayFromZero)+1 == parY) || ((int)Math.Round(parElectricBall.X, MidpointRounding.AwayFromZero)-2 == parX && (int)Math.Round(parElectricBall.Y, MidpointRounding.AwayFromZero) == parY) || ((int)Math.Round(parElectricBall.X, MidpointRounding.AwayFromZero)-1 == parX && (int)Math.Round(parElectricBall.Y, MidpointRounding.AwayFromZero) == parY) || ((int)Math.Round(parElectricBall.X, MidpointRounding.AwayFromZero) == parX && (int)Math.Round(parElectricBall.Y, MidpointRounding.AwayFromZero) == parY) || ((int)Math.Round(parElectricBall.X, MidpointRounding.AwayFromZero)+1 == parX && (int)Math.Round(parElectricBall.Y, MidpointRounding.AwayFromZero) == parY) || ((int)Math.Round(parElectricBall.X, MidpointRounding.AwayFromZero)+2 == parX && (int)Math.Round(parElectricBall.Y, MidpointRounding.AwayFromZero) == parY) || ((int)Math.Round(parElectricBall.X, MidpointRounding.AwayFromZero) -1 == parX && (int)Math.Round(parElectricBall.Y, MidpointRounding.AwayFromZero)-1 == parY) || ((int)Math.Round(parElectricBall.X, MidpointRounding.AwayFromZero) == parX && (int)Math.Round(parElectricBall.Y, MidpointRounding.AwayFromZero)-1 == parY) || ((int)Math.Round(parElectricBall.X, MidpointRounding.AwayFromZero)+1 == parX && (int)Math.Round(parElectricBall.Y, MidpointRounding.AwayFromZero)-1 == parY))

{

return true;

}

return false;

}

Фрагмент кода после рефакторинга:

public bool CheckCoordinateBallShip(ElectricBall parElectricBall, int parX, int parY)

{

int x = (int)Math.Round(parElectricBall.X, MidpointRounding.AwayFromZero);

int y = (int)Math.Round(parElectricBall.Y, MidpointRounding.AwayFromZero);

if ((x == parX && y+1 == parY) || (x-1 == parX && y+1 == parY) || (x+1 == parX && y+1 == parY)

|| (x-2 == parX && y == parY) || (x-1 == parX && y == parY) || (x == parX && y == parY)

|| (x+1 == parX && y == parY) || (x+2 == parX && y == parY) || (x-1 == parX && y-1 == parY)

|| (x == parX && y-1 == parY) || (y+1 == parX && y-1 == parY))

{

return true;

}

return false;

}

### Передача в метод целого объекта вместо отдельных полей

Описание вида рефакторинга: метод, использующий несколько значений одного объекта, принимает сразу весь объект вместо этих значений.

Пример: создание формы представления для сущностей.

Фрагмент кода до рефакторинга:

public static Shape CreateViewEntity(DirectionsType parDirection, double parX, double parY)

{

...

}

Фрагмент кода после рефакторинга:

public static Shape CreateViewEntity(Entity parEntity)

{

...

}

## Разработка тестов

### Test Cases

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Содержание теста | Предусловие | Действия | Ожидаемый результат |
| 1 | Переход из главного меню к рекордам | Открыто главное меню. | 1. Перейти к пункту меню «Рекорды» 2. 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. | 1. Нажать клавишу «Главное меню». | Открыто главное меню, фокус находится на пункте меню «Рекорды». |
| 3 | Переход из главного меню к правилам | Открыто главное меню. | 1. Перейти к пункту меню «Правила». 2. Нажать на пункт меню «Правила». | Открыто окно правил и выведены правила игры. |
| 4 | Сохранение состояния меню после возврата из правил в главное меню | Открыто окно правил. | 1. Нажать клавишу «Главное меню». | Открыто главное меню, фокус находится на пункте меню «Правила». |
| 5 | Переход из главного меню к игре | Открыто главное меню. | 1. Перейти к пункту меню «Играть». 2. Нажать на пункт меню «Играть». | Открыто окно игры и космический корабль движется влево вверх, электрические шары - вниз |
| 6 | Изменение направления движения космического корабля с влево вверх на влево вниз | Открыто окно игры. Космический корабль движется влево вверх | 1. Нажать пробел | Космический корабль сменил направление движения. Движется влево вниз |
| 7 | Изменение направления движения космического корабля с влево вниз на вправо вниз | Открыто окно игры. Космический корабль движется влево вниз | 1. Нажать пробел | Космический корабль сменил направление движения. Движется вправо вниз |
| 8 | Изменение направления движения космического корабля с вправо вниз на вправо вверх | Открыто окно игры. Космический корабль движется вправо вниз | 1. Нажать пробел | Космический корабль сменил направление движения. Движется вправо вверх |
| 9 | Изменение направления движения космического корабля с вправо вверх на влево вверх | Открыто окно игры. Космический корабль движется вправо вверх | 1. Нажать пробел | Космический корабль сменил направление движения. Движется влево вверх |
| 10 | Проигрыш и сохранение результата под введенным именем | Открыто окно окончания игры. Выведен набранный счет | 1. Ввести имя 2. Нажать на «Главное меню». | Сохранение результата в файл с рекордами под введенным именем |
| 11 | Проигрыш и сохранение результата под именем неизвестный | Открыто окно окончания игры. Выведен набранный счет | 1. Нажать на «Главное меню» | Сохранение результата в файл с рекордами под именем неизвестный |
| 12 | Сохранение состояния меню после возврата из рекордов в главное меню | Открыто окно рекордов. | 1. Нажать клавишу Esc. | Открыто главное меню, фокус находится на пункте меню «Рекорды». |
| 13 | Сохранение состояния меню после возврата из правил в главное меню | Открыто окно правил. | 1. Нажать клавишу Esc | Открыто главное меню, фокус находится на пункте меню «Правила». |
| 14 | Столкновение космического корабля с границами поля | Открыто окно игры | 1. Выполнить столкновение космического корабля с границами поля | Открыто окно окончания игры, выведен счет |
| 15 | Столкновение космического корабля с электрическим шаром | Открыто окно игры | 1. Выполнить столкновение электрического шара и космического корабля | Открыто окно окончания игры, выведен счет |
| 16 | Проверка перемещения электрического шара | Открыто окно игры | 1. Дождаться, когда электрический шар приблизится к нижней границе поля | Появление нового электрического шара в верхней части поля |
| 17 | Сохранение состояния меню после возврата из игры в главное меню | Открыто окно игры. | 1. Нажать клавишу Esc | Открыто главное меню, фокус находится на пункте меню «Играть». |
| 18 | Удаление символа из поля для ввода | Открыто окно окончания игры. В поле для ввода введен текст | 1. Нажать на Backspace | Удаление последнего символа из поля для ввода |
| 19 | Перемещение между пунктами меню | Открыто главное меню | 1. Нажать клавишу «↑» («↓») | Верхний (нижний) пункт меню становится в фокусе |
| 20 | Выход из приложения | Открыто главное меню. | 1. Перейти к пункту меню «Выход» 2. Нажать на пункт меню «Выход». | Приложение закрыто. |

### Модульные тесты

Для класса Model.Game.Game были разработаны модульные тесты.

Примеры разработанных модульных тестов:

/// <summary>

/// Тестирование изменения направления движения космического корабля

/// Изменение с LeftDown на LeftUp

[TestMethod]

public void TestChangeDirectionShipLeftUp()

{

Game game = new Game();

game.SpaceShip.DirectionType = DirectionsType.LeftDown;

game.MovementLeftUp();

Assert.AreEqual(DirectionsType.LeftUp, game.SpaceShip.DirectionType);

}

/// <summary>

/// Тестирование движения сущностей

/// сравнение координат космического корабля при движении влево вверх

/// </summary>

[TestMethod]

public void TestMovementShipLeftUp()

{

Game game = new Game();

game.WidthScreen = 90;

game.HeightScreen = 30;

game.MoveTime = 10;

int speedShip = game.SpeedSpaceShip;

game.SpaceShip.Movement(speedShip\*game.MoveTime, game.HeightScreen, game.WidthScreen);

double x = game.SpaceShip.X;

double y = game.SpaceShip.Y;

game.Initialization();

game.WidthScreen = 90;

game.HeightScreen = 30;

game.MoveTime = 10;

game.MovementEntities();

Assert.AreEqual(x, game.SpaceShip.X);

Assert.AreEqual(y, game.SpaceShip.Y);

}

/// <summary>

/// Тестирование проверки на столкновения между электрическим шаром и космическим кораблем

/// если столкновение произошло, то состояние игрового процесса IsStop принимает значение true

/// в противном случае - false

/// </summary>

[TestMethod]

public void TestCheckBallCollision()

{

Game game = new Game();

game.SpaceShip.X = 10;

game.SpaceShip.Y = 10;

game.ElectricBallFirst.X = 10;

game.ElectricBallFirst.Y = 30;

game.CheckBallCollision();

Assert.AreEqual(false, game.IsStop);

game.SpaceShip.X = 10;

game.SpaceShip.Y = 10;

game.ElectricBallFirst.X = 10;

game.ElectricBallFirst.Y = 10;

game.CheckBallCollision();

Assert.AreEqual(true, game.IsStop);

}

# Результат работы программы

## Консольное приложение

### Главное меню



Рисунок 8 – Окно главного меню

На рисунке 15 приведено окно главного меню. Пользователю доступны 4 действия: начать новую игру («Играть»), просмотреть таблицу рекордов («Рекорды»), просмотреть правила («Правила»), завершить работу приложения («Выход»).

### Начало игры

При нажатии на пункт главного меню «Новая игра» пользователь переходит к игре. Данное окно представлено на рисунке 16.

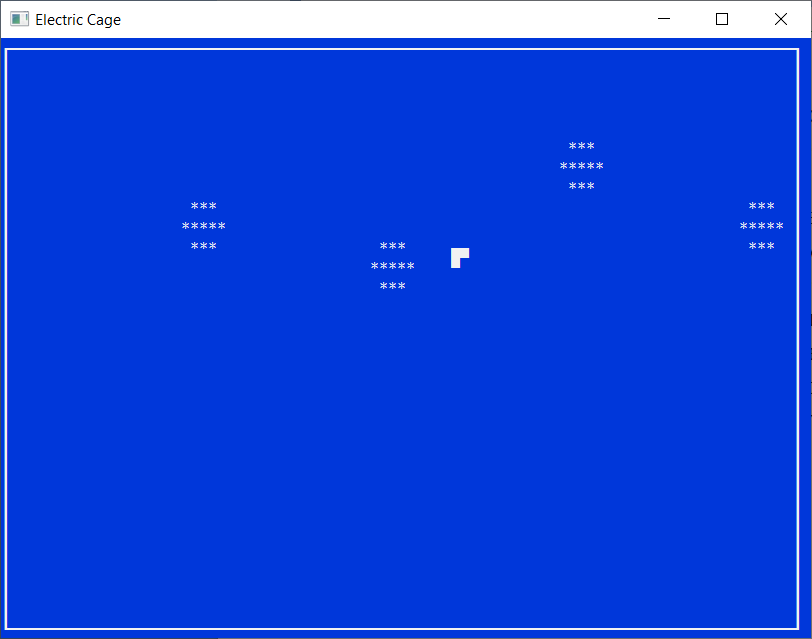


Рисунок 9 - Игра после запуска

Изначально космический корабль двигается влево вверх. Чтобы изменить направление движения космического корабля необходимо нажать на пробел. Чтобы вернуться на главное меню нажмите Esc.

### Завершение игры

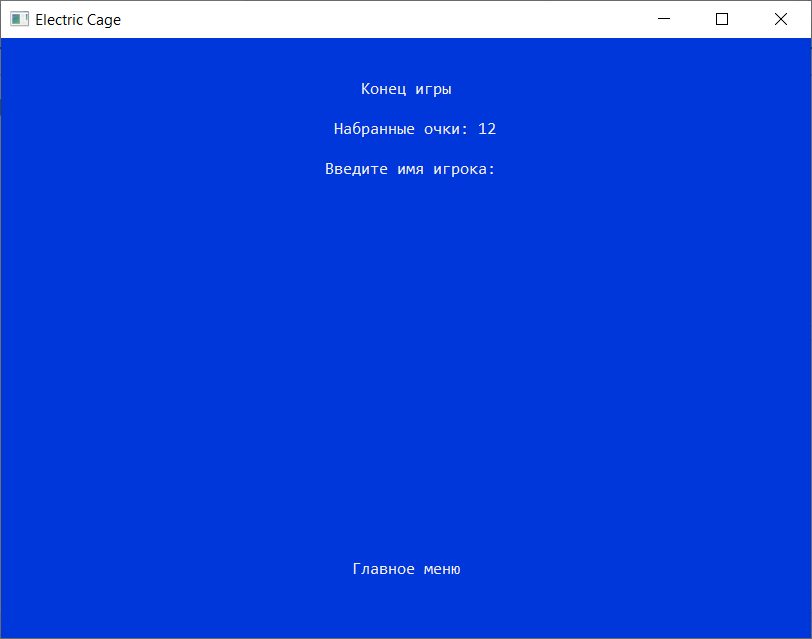


Рисунок 10 – Окно окончания игры

После поражения пользователь переходит в окно окончания игры. Пользователь может ввести свое имя или оставить поле для имени пустым и тогда в таблице рекордов будет записано имя «Неизвестный».

### Рекорды

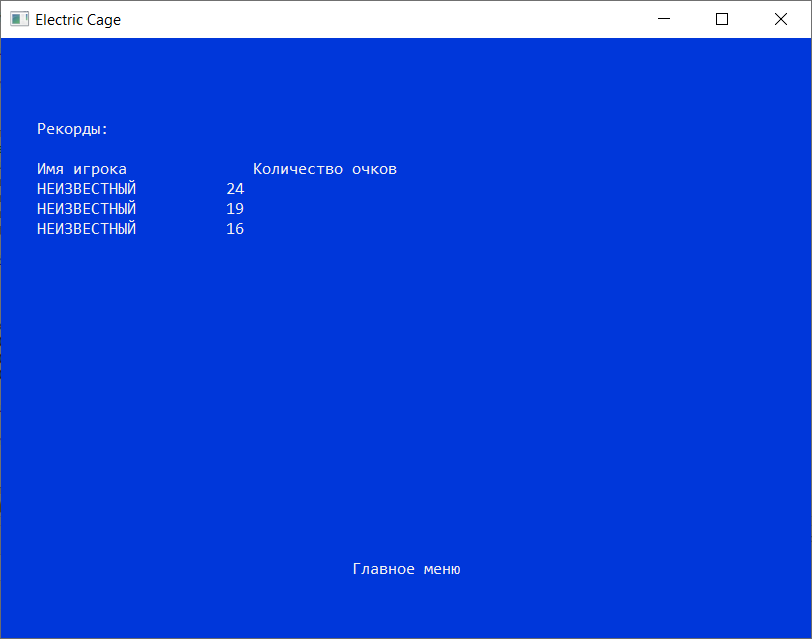


Рисунок 17 – Окно рекордов

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

### Правила

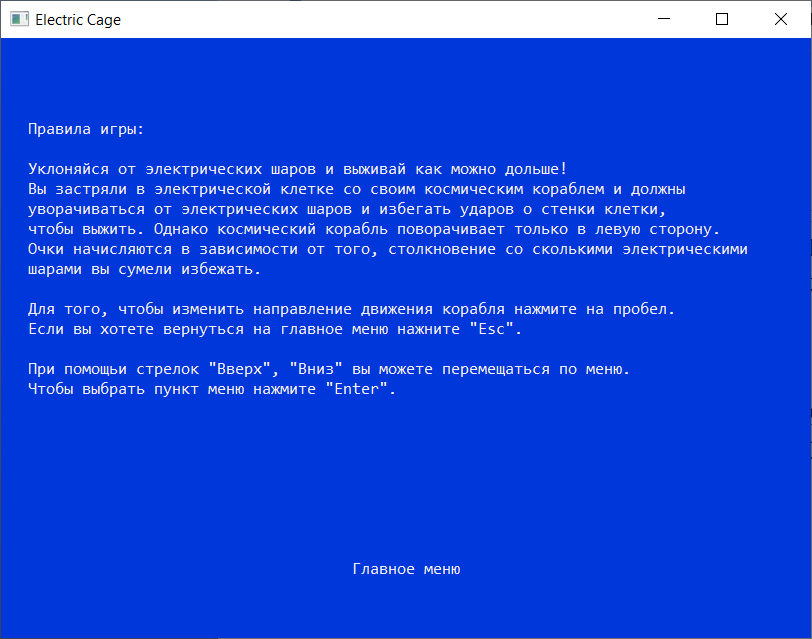


Рисунок 18 - Окно правил

При нажатии на пункт главного меню «Правила» пользователь переходит к тексту правил (рисунки 18).

## Графическое приложение

Игровой процесс и работа с графическим приложением аналогичны консольному приложению, поэтому в этом разделе приведены только скриншоты графического приложения.

### Главное меню

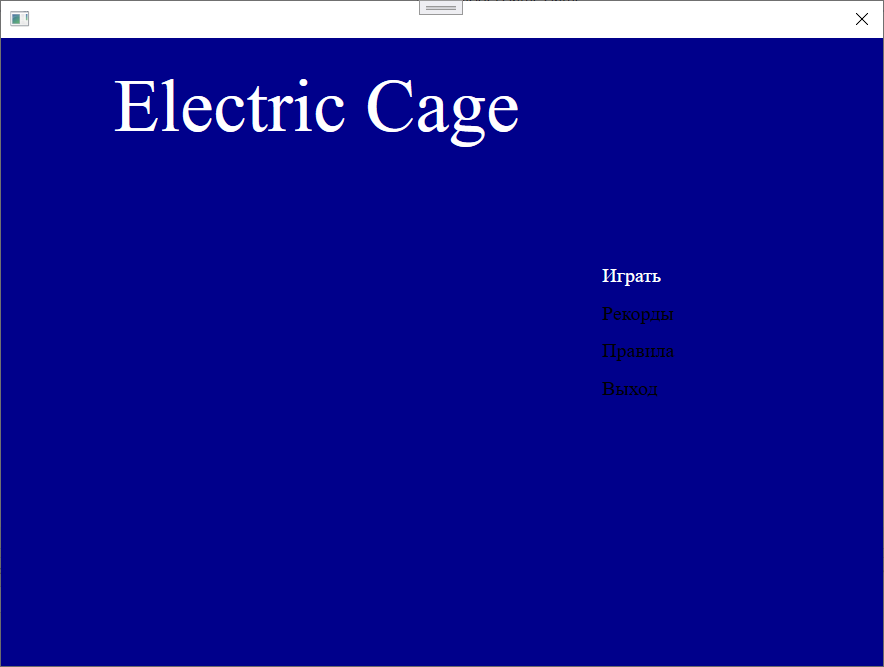


Рисунок 19 - Окно главного меню

### Начало игры

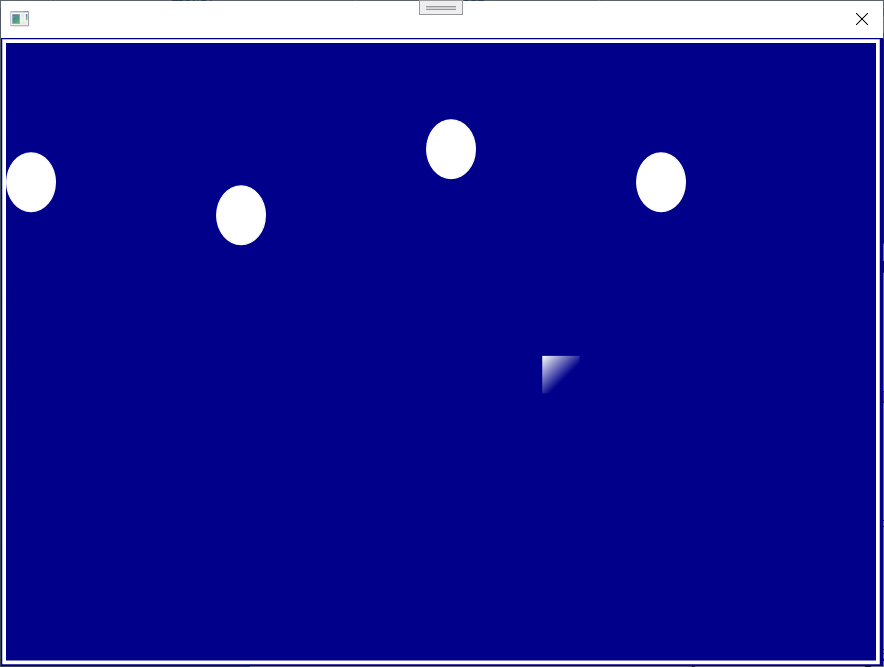


Рисунок 20 - Игра после запуска

### Завершение игры

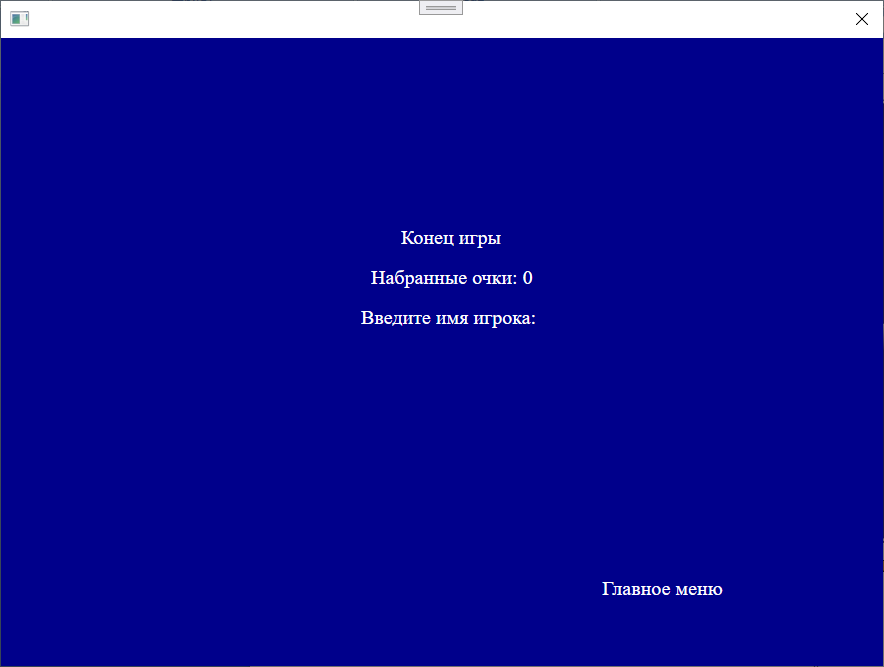


Рисунок 21 - Окно окончания игры

### Рекорды

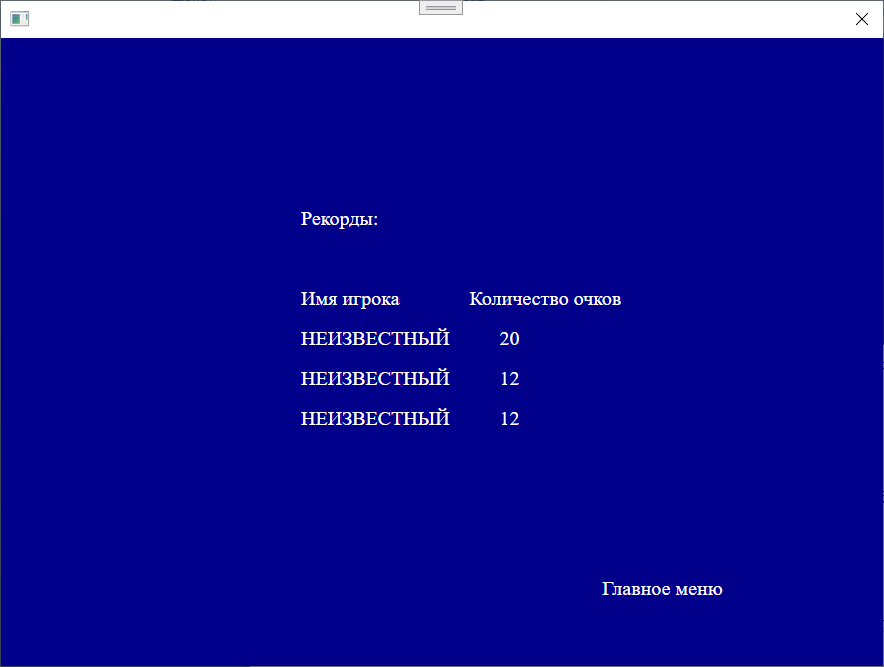


Рисунок 22 - Окно рекордов

### Правила

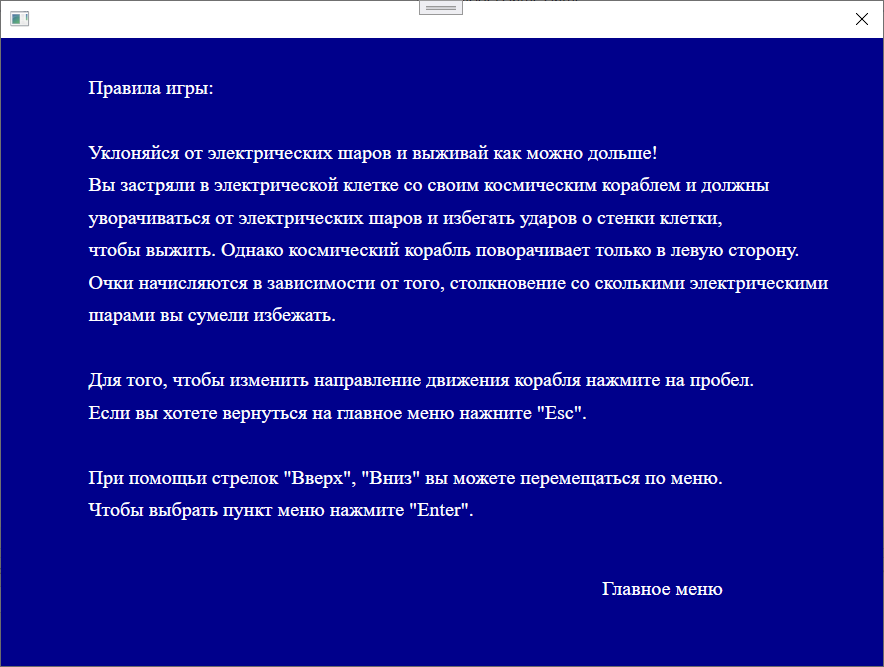


Рисунок 23 - Окно правил

Заключение

В результате работы было спроектирована и разработана игра «Electric Cage» в двух видах интерфейса: консольном и графическом.