Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение  
высшего образования  
«Рязанский государственный радиотехнический университет им. В. Ф. Уткина»  
(ФГБОУ ВО «РГРТУ», РГРТУ)

Кафедра «Вычислительная и прикладная математика» (ВПМ)

**Пояснительная записка**

**к курсовому проекту**

на тему: «Разработка игровой программы «Influence»

по курсу

**«Конструирование программного обеспечения»**

Выполнил:

студент группы № 6413

Лащилин Максим Александрович

Проверил:

доцент кафедры ВПМ

Столчнев В. К.

Рязань, 2019

Оглавление

[Введение 3](#_Toc29424325)

[1. Анализ задачи 4](#_Toc29424326)

[1.1. Разработка иерархии классов 4](#_Toc29424327)

[1.1.1. Выделение сущностей 4](#_Toc29424328)

[1.1.2. Зависимости между классами. Диаграмма классов 4](#_Toc29424329)

[1.2. Алгоритмы 6](#_Toc29424330)

[1.3. Разработка интерфейса программы 6](#_Toc29424331)

[2. Написание программы 9](#_Toc29424332)

[2.1.1. Описание разработанных процедур и функций 9](#_Toc29424333)

[2.2. Разработка программы 11](#_Toc29424334)

[2.2.1. Описание классов, перечислений и интерфейсов проекта 11](#_Toc29424335)

[2.3. Описание шаблонов проектирования, которые использовались при написании программы 12](#_Toc29424336)

[2.3.1. Шаблон «Одиночка» (Singleton) 12](#_Toc29424337)

[2.3.2. Шаблон «Абстрактная фабрика» (Abstract Factory) 12](#_Toc29424338)

[2.3.3. Шаблон «Адаптер» (Adapter) 12](#_Toc29424339)

[2.4. Описание методов рефакторинга, которые использовались при оптимизации исходного кода программы 12](#_Toc29424340)

[2.4.1. Замена магического числа на именованную константу 12](#_Toc29424341)

[2.4.2. Присвоение переменной более ясного или информативного имени 13](#_Toc29424342)

[2.4.3. Присвоение перечислению более ясного или информативного имени 13](#_Toc29424343)

[2.5. Разработка тестов 13](#_Toc29424344)

[2.5.1. Тест-кейсы 13](#_Toc29424345)

[2.5.2. Unit-тесты 15](#_Toc29424346)

[3. Результат работы программы 15](#_Toc29424347)

[3.1. Результаты работы консольной реализации 15](#_Toc29424348)

[3.2. Результаты работы Windows Forms реализации 15](#_Toc29424349)

[Заключение 15](#_Toc29424350)

[Приложения 15](#_Toc29424351)

Введение

На сегодняшний день компьютерные игры, как феномен культуры, активно развиваются и занимают прочную позицию в жизни людей, и не только молодежи. Из всех видов электронных развлечений компьютерные игры представляют особый интерес для науки, так как обладают максимальной информативностью и системной сложностью по отношению к другим электронным играм. Разработкой компьютерных игр может заниматься как один человек, так и большая команда специалистов. Обычно крупномасштабными коммерческими играми занимаются компании, специализирующиеся на играх для персонального компьютера или консолей. Однако разработкой несложной игры вполне может заниматься и один человек.

Рынок компьютерных игр на сегодняшний день, как в мире, так и в России в частности занимает довольно большую долю от всей разработки программного обеспечения. Объясняется это высокой доходностью данных проектов.

В рамках курсового проекта требуется разработать игру «Influence». Для достижения поставленной цели требуется выполнить следующие задачи:

* разработать иерархию классов;
* разработать алгоритмы работы программы;
* разработать макеты пользовательского интерфейса;
* программно реализовать игру;
* произвести тестирование разработанной программы;
* произвести анализ полученных результатов.

# Анализ задачи

## Разработка иерархии классов

Основной задачей данного курсового проекта является разработка игровой программы «Influence» с использованием архитектурного шаблона Model-View-Controller (Модель – Представление – Контроллер). MVC предполагает разделение ответственности классами приложения на три основных блока. Так, каждый из компонентов приложения ответственен за свою задачу:

* модель ответственна за бизнес-логику приложения (обработка данных, изменение состояния контроллером);
* контроллер отвечает за обработку действий пользователя и изменение состояния модели, вызовом её методов;
* представление ответственно за отображение данных модели пользователю.

Использование данного архитектурного паттерна имеет следующие достоинства:

* возможность независимой модификации одного из компонентов относительно других;
* на основе одной модели можно реализовать несколько представлений;
* не затрагивая реализацию представлений можно изменить реакции на действия пользователя путём использования другого контроллера.

Для определённых классов, объекты которых требуются в единственном экземпляре, необходимо применить шаблон «Одиночка». Например, для описания работы курсора.

Для создания контроллеров состояний приложения возможно использование шаблона «Абстрактная фабрика».

### Выделение сущностей

Основными сущностями в игре «Influence» являются следующие:

* Ячейка (Cell) – представляет собой единицу игрового поля, которая может принадлежать одному из игроков или являться свободной. В случае если ячейка принадлежит одному из игроков, она обязательно содержит определённое количество очков, но не больше максимально возможного.
* Игрок (Player) – представляет собой описание сущностей, которые принимают участие в игре.
* Игровое поле (Game field) – представляет собой совокупность ячеек и реализует логику захвата ячеек игроками.

### Зависимости между классами. Диаграмма классов

В данном разделе описаны связи между классами для основных компонентов программы: модели, представления и контроллера.

На рис. 1 представлена диаграмма классов модели.

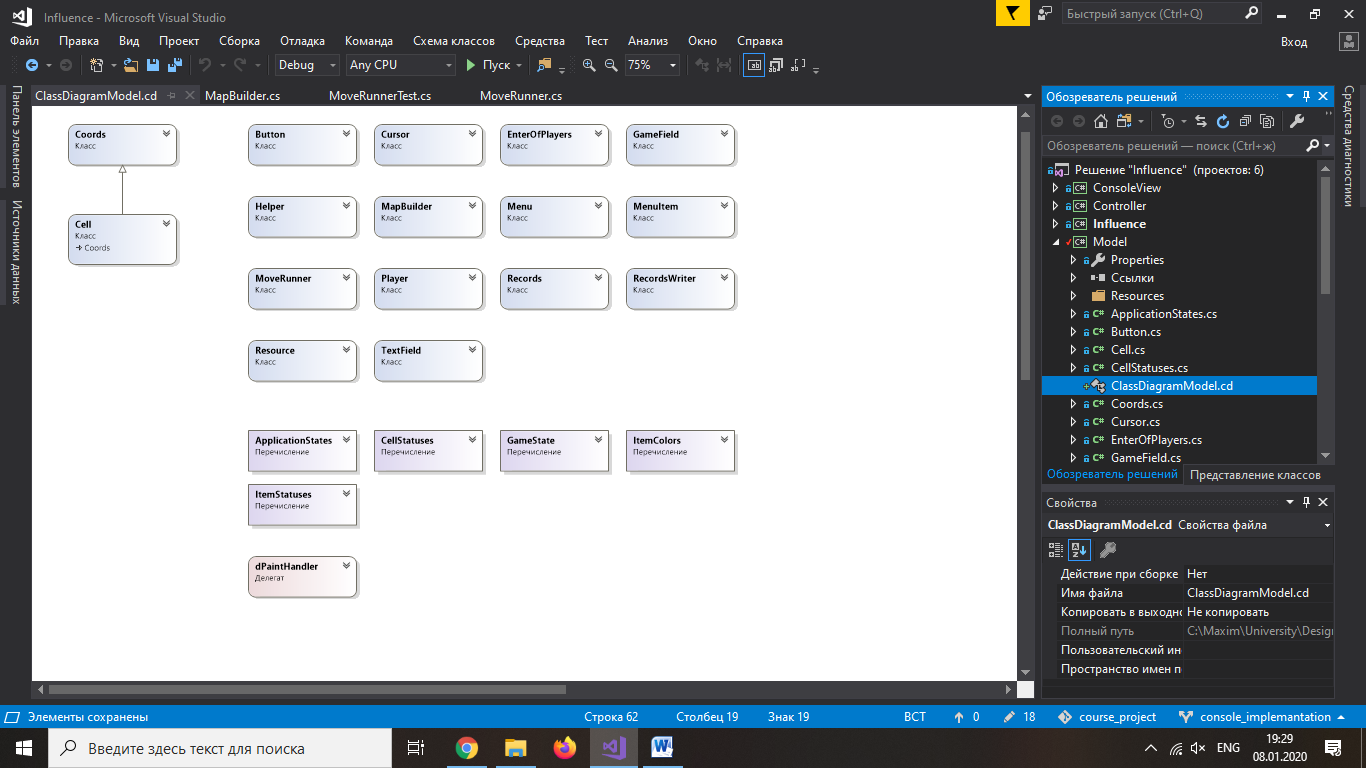


Рис. 1 – Диаграмма классов модели

На рис. 2 представлена диаграмма классов контроллера.

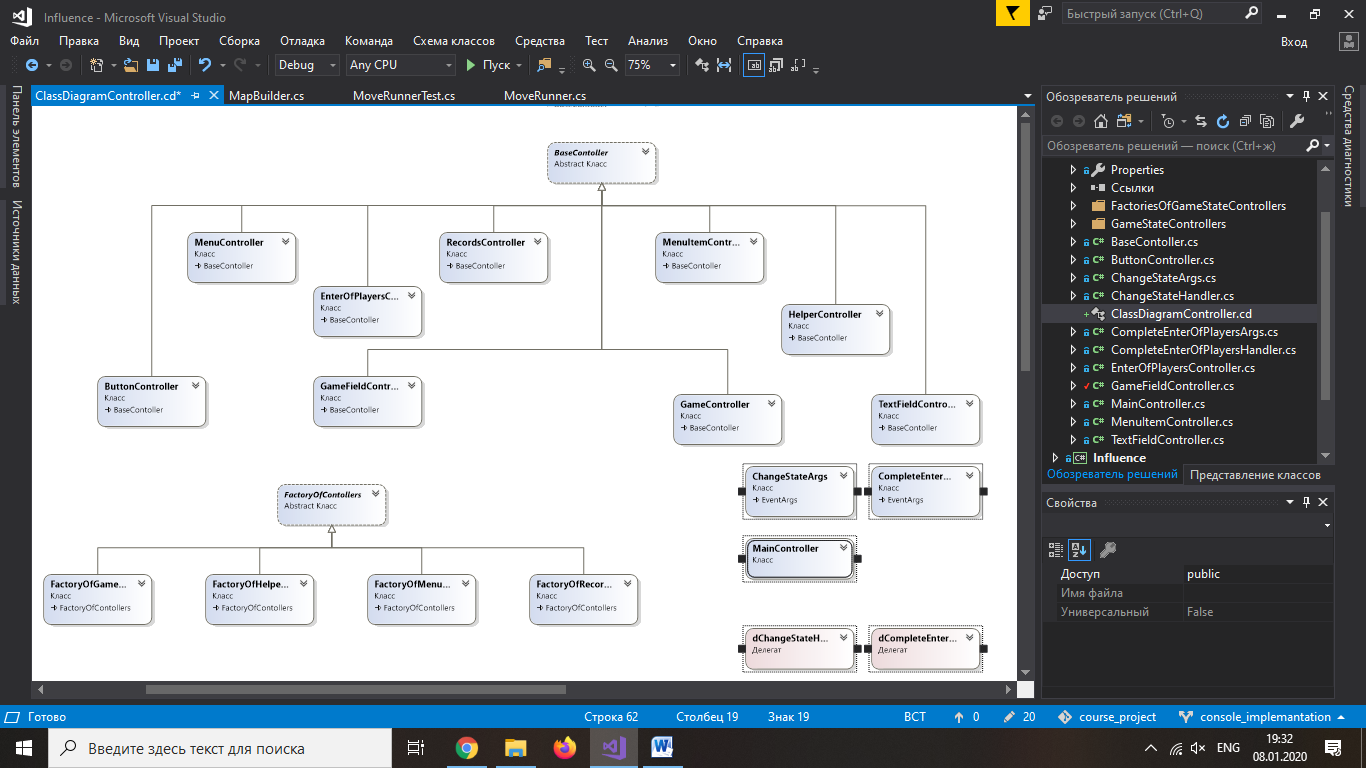


Рис. 2 – Диаграмма классов контроллера

На рис. 3 представлена диаграмма классов представления.

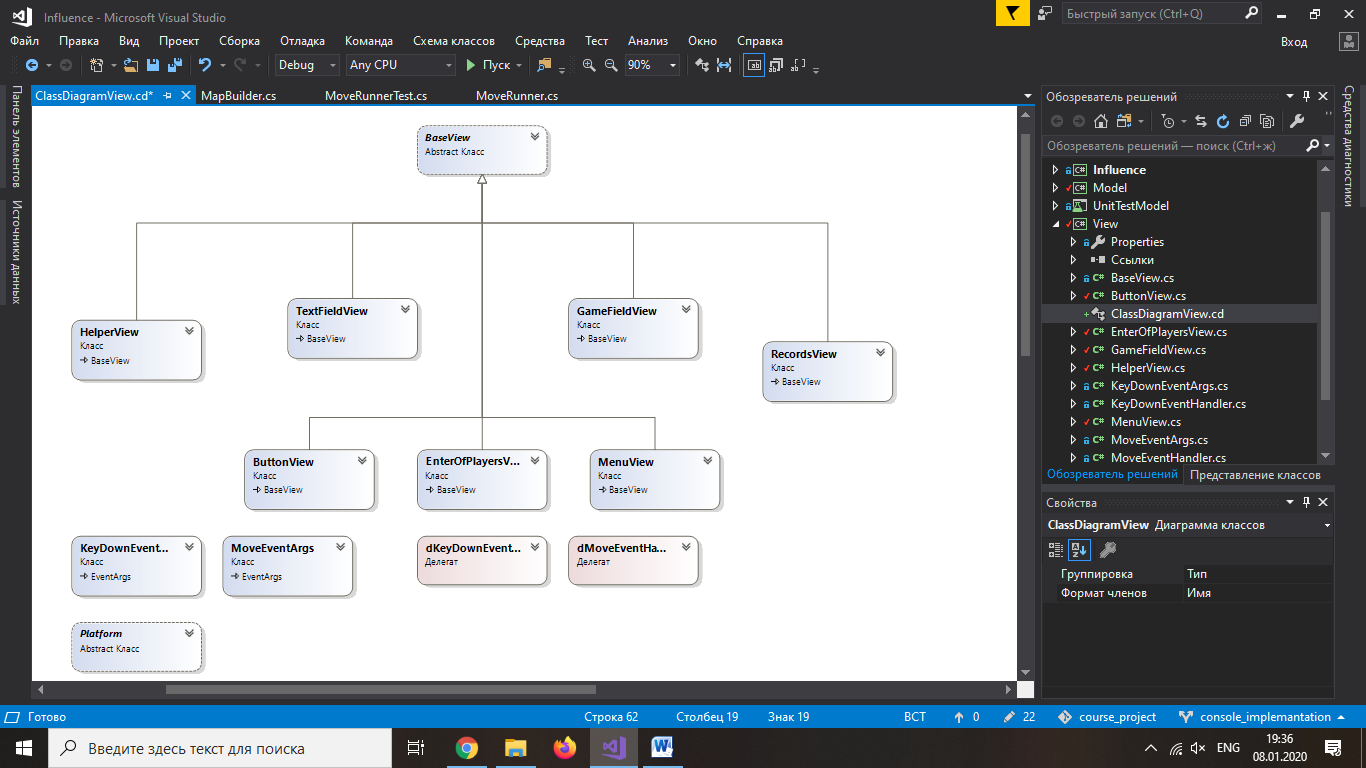


Рис. 3 – Диаграмма классов представления

## Алгоритмы

Одним из важных алгоритмов в игре является алгоритм захвата ячейки игроком. Его блок-схема представлена на рис. 4.

Рис. 4 – Блок-схема алгоритма захвата ячейки

Также важен алгоритм проверки возможности совершения хода из одной ячейки в другую. Его блок-схема представлена на рис. 5.

Рис. 5 – Блок-схема алгоритма проверки возможности совершения хода

## Разработка интерфейса программы

На этапе проектирования программы требуется разработать её графический пользовательский интерфейс. Интерфейс должен включать в себя следующие элементы:

* главное меню игры;
* ввод имён игроков;
* игровое поле;
* просмотр рекордов;
* справку.

Макеты перечисленных элементов представлены на рисунках ниже.



Рис. 6 – Главное меню



Рис. 7 – Ввод имени игроков



Рис. 8 – Игровое поле в состоянии захвата ячейки



Рис. 9 – Игровое поле в состоянии раздачи очков



Рис. 10 – Просмотр рекордов



Рис. 11 – Справка

# Написание программы

### Описание разработанных процедур и функций

Описание разработанных процедур и функций представлено в таблице 1.

Таблица 1 – Описание разработанных процедур и функций

| **Название процедуры/функции** | **Описание функциональности** | **Описание параметров** |
| --- | --- | --- |
| Button. CallClick() | Вызывает событие нажатия на кнопку | - |
| Button. CallPaintEvent() | Вызывает событие перерисовки | - |
| Cell.IncreaseScore() | Увеличивает счёт на единицу | - |
| Cell. DecreaseScore() | Уменьшает счёт на единицу | - |
| Cell.ActiveCell() | Делает ячейку активной | - |
| Cell.DisactiveCell() | Делает ячейку неактивной | - |
| Cell.IsCellFree() | Проверяет, имеет ли ячейка владельца | - |
| Cursor.GetInstance() | Получает экземпляр курсора | - |
| Cursor.Move(float parX, float parY) | Изменяет координаты курсора | parX – Координата X  parY - Координата Y |
| EnterOfPlayers.Initialize() | Инициализирует ввод имен игроков | - |
| EnterOfPlayers.Next() | Переводит фокус на следующее текстовое поле | - |
| EnterOfPlayers.Previous() | Переводит фокус на предыдущее текстовое поле | - |
| GameField.OnFinishedEvent(object parSender, EventArgs parE) | Обрабатывает событие окончания игры | parSender – Источник события  parE - Параметры события |
| GameField.Initialize() | Инициализирует игровое поле | - |
| GameField.SelectCell() | Выбирает ячейку | - |
| GameField.AtackCell() | Производит атаку на ячейку | - |
| GameField.DistributeScore() | Раздает очки | - |
| GameField.UnselectAllCells() | Снимает выделение со всех ячеек | - |
| GameField.PassMove() | Передает ход следующему игроку | - |
| GameField.GetFocusedCell() | Получает ячейку, на которую наведен курсор | - |
| GameField.GetSelectedCell() | Получает выбранную ячейку | - |
| GameField.GetPlayerNumOfCells(Player parPlayer) | Получает число ячеек игрока | parPlayer - Объект игрока |
| GameField.CompleteAtackOrPassMove() | Завершает атаку или передает ход | - |
| GameField.PerformGameAction() | Выполняет одно из игровых действий | - |
| GameField.CalculateScorePlayer() | Вычисляет счет текущего игрока | - |
| GameField.GetActivePlayer() | Возвращает текущего игрока | - |
| GameField.IsFinishedGame() | Проверяет, окончена ли игра | - |
| MapBuilder.BuildMap(int parVerticalSize, int parHorizontalSize, List<Player> parPlayers) | Строит карту | parVerticalSize – Вертикальный размер  parHorizontalSize – Горизонтальный размер  parPlayers – Список игроков |
| MapBuilder.CreateGameCells(int parVerticalSize, int parHorizontalSize) | Создает массив игровых ячеек | parVerticalSize – Вертикальный размер  parHorizontalSize – Горизонтальный размер |
| MapBuilder.SetPlayers(Cell[,] parCells, List<Player> parPlayers) | Устанавливает игроков для первоначальных ячеек | parCells – Массив ячеек  parPlayers – Список игроков |
| Menu.Initialize() | Инициализирует меню | - |
| Menu.AddItem(int parId, string parName) | Добавляет элемент в меню | parId – Идентификатор  parName – Название пункта |
| Menu.Next() | Переводит фокус на следующиий элемент меню | - |
| Menu.Previous() | Переводит фокус на предыдущий элемент меню | - |
| MoveRunner.Move(Cell parSourceCell, Cell parDestinationCell, Player parCurrentPlayer) | Совершает ход | parSourceCell – Исходная ячейка  parDestinationCell – Ячейка назначения  parCurrentPlayer - Текущий игрок |
| MoveRunner.IsMove(int parSourceVerticalCoord, int parSourceHorizontalCoord, int parDestinationVerticalCoord, int parDestinationHorizontalCoord) | Проверяет возможность совершения хода | parSourceVerticalCoord – Индекс ряда исходной ячейки  parSourceHorizontalCoord – Индекс колонки исходной ячейки  parDestinationVerticalCoord – Индекс ряда ячейки назначения  parDestinationHorizontalCoord – Индекс колонки ячейки назначения |
| MoveRunner.IsCellOccupied(Cell parCell, Player parCurrentPlayer) | Проверяет, занята ли ячейка другим игроком | parCell – Объект ячейки  parCurrentPlayer – Текущий игрок |
| Records.Initialize() | Инициализирует просмотр рекордов | - |
| Records.ReadTextFromFile(string parFileName) | Читает текст из файла | parFileName - Имя файла |
| Records.FilterBestResults(List<string> parRecords) | Фильтрует результаты | parRecords - Список результатов |
| RecordsWriter.RecordResult(Player parPlayer) | Записывает результат игрока | parPlayer - Объект игрока |
| TextField.Initialize() | Инициализирует текстовое поле | - |
| TextField.AddChar(char parChar) | Добавляет символ в текстовое поле | parChar - Добавляемый символ |
| TextField.DeleteLastChar() | Удаляет последний символ из текстового поля | - |
|  |  |  |

## Разработка программы

В разработке программы можно выделить следующие этапы:

1. Разработка общей структуры проекта. В процессе этого этапа были выделены основные архитектурные слои (модель, представление, контроллер), разработана общая модель взаимодействия классов при помощи событий.
2. Разработка бизнес-логики игры.
3. Разработка конкретных реализаций для представлений (консольного и Windows Forms).
4. Тестирование.
5. Оптимизация алгоритмов и рефакторинг программного кода.

### Описание классов, перечислений и интерфейсов проекта

Описание классов, перечислений и интерфейсов проекта представлено в таблице 2.

Таблица 2 – Описание классов, перечислений и интерфейсов проекта

|  |  |  |
| --- | --- | --- |
| **Пространство имен** | **Название класса / интерфейса / перечисления** | **Назначение** |
| Model | ApplicationStates | Состояния приложения |
| Model | Button | Кнопка |
| Model | Cell | Игровая ячейка |
| Model | CellStatuses | Статусы ячейки |
| Model | Coords | Координаты ячейки в массиве |
| Model | Cursor | Курсор |
| Model | EnterOfPlayers | Отвечает за ввод игроков |
| Model | GameField | Игровое поле |
| Model | GameStates | Состояния игры |
| Model | Helper | Справка |
| Model | ItemColors | Цвета, отличающие игроков |
| Model | ItemStatuses | Статусы элементов управления |
| Model | MapBuilder | Строитель карты |
| Model | Menu | Меню |
| Model | MenuItem | Элемент меню |
| Model | MoveRunner | Отвечает за логику совершения ходов |
| Model | Player | Класс игрока |
| Model | Records | Отвечает за просмотр рекордов |
| Model | RecordsWriter | Отвечает за запись результата игры в файл |
| Model | TextField | Текстовое поле |

## Описание шаблонов проектирования, которые использовались при написании программы

### Шаблон «Одиночка» (Singleton)

Одиночка – это порождающий шаблон проектирования, который гарантирует, что у класса есть только один экземпляр, и предоставляет к нему глобальную точку доступа.

Данный шаблон используется в классах «Cursor» и «EventListener».

### Шаблон «Абстрактная фабрика» (Abstract Factory)

Абстрактная фабрика – это порождающий шаблон проектирования, который позволяет создавать семейства связанных объектов, не привязываясь к конкретным классам создаваемых объектов.

Данный шаблон применяется для создания контроллеров, которые отвечают за обработку действий от пользователя в следующих состояниях: работе в главном меню, игровом процессе, просмотре справочной информации и рекордов.

### Шаблон «Адаптер» (Adapter)

Адаптер – это структурный шаблон проектирования, предназначенный для организации использования функций объекта, недоступного для модификации, через специально созданный интерфейс.

Данный шаблон применяется в классе EventListener, который использует функциональность класса NativeMethodsProvider.

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

### Замена магического числа на именованную константу

Данный метод подразумевает замену численного или строкового литерала именованной константой.

Пример кода до рефакторинга:

switch (parE.VirtualKeyCode)

{

case 13:

CallEnterDown();

break;

case 38:

CallArrowUpDown();

break;

case 40:

CallArrowDown();

break;

}

Пример кода после рефакторинга:

const int ENTER\_CODE = 13;

const int ARROW\_UP\_CODE = 38;

const int ARROW\_DOWN\_CODE = 40;

if (parE.KeyDown)

{

switch (parE.VirtualKeyCode)

{

case ENTER\_CODE:

CallEnterDown();

break;

case ARROW\_UP\_CODE:

CallArrowUpDown();

break;

case ARROW\_DOWN\_CODE:

CallArrowDown();

break;

}

### Присвоение переменной более ясного или информативного имени

Присвоение переменной более ясного или информативного имени позволяет улучшить читаемость и/или информативность кода.

Пример кода до рефакторинга:

private int \_virtualKeyCode;

Пример кода после рефакторинга:

private int \_keyCode;

### Присвоение перечислению более ясного или информативного имени

Присвоение переменной более ясного или информативного имени позволяет улучшить читаемость и/или информативность кода.

Пример кода до рефакторинга:

public enum ItemStatus

Пример кода после рефакторинга:

public enum ItemStatuses

## Разработка тестов

### Тест-кейсы

Для проверки работоспособности игры были разработаны тест-кейсы, представленные в таблице 3.

Таблица 3 – Тест-кейсы

| **Название теста** | **Предусловия** | **Действия** | **Ожидаемый результат** | **Результат выполнения** |
| --- | --- | --- | --- | --- |
| Запуск игры | - | 1. Запустить игру 2. В главном меню выбрать пункт «New game» 3. В появившемся окне ввода игроков ввести имена игроков 4. Нажать Enter | Появляется игровое поле с ячейками, кнопкой «Complete atack» и подписью текущего игрока. | Выполнен |
| Занятие свободной ячейки | Игра запущена | 1. Выбрать ячейку текущего игрока, число очков которой больше единицы 2. Выбрать соседнюю свободную ячейку | Счет соседней ячейки стал равен единице. Счет выбранной соседней ячейки стал равен предыдущему значению исходной ячейки минус один. Свободная ячейка окрашена в цвет ячеек текущего игрока. | Выполнен |
| Занятие ячейки соперника, счёт которой больше счета атакующей ячейки | Игра запущена, ячейки игроков соседствуют | 1. Выбрать ячейку текущего игрока, число очков которой больше единицы 2. Выбрать соседнюю ячейку, занятую другим игроком, счет которой больше счета атакующей ячейки | Счет соседней ячейки стал равен разнице между счетом атакуемой и атакующей ячейки. Счет атакующей ячейки стал равен единице. Цвета обоих ячеек остались прежними. | Выполнен |
| Занятие ячейки соперника, счёт которой равен счету атакующей ячейки | Игра запущена, ячейки игроков соседствуют | 1. Выбрать ячейку текущего игрока, число очков которой больше единицы 2. Выбрать соседнюю ячейку, занятую другим игроком, счет которой равен счету атакующей ячейки | Счет атакуемой ячейки стал равен единице. Счет атакующей ячейки стал равен единице. Цвета обоих ячеек остались прежними. | Выполнен |
| Занятие ячейки соперника, счёт которой меньше счета атакующей ячейки | Игра запущена, ячейки игроков соседствуют | 1. Выбрать ячейку текущего игрока, число очков которой больше единицы 2. Выбрать соседнюю ячейку, занятую другим игроком, , счет которой меньше счета атакующей ячейки | Счет атакуемой ячейки стал равен разнице между счетом атакующей ячейки и счетом атакуемой ячейки. Счет атакующей ячейки стал равен единице. Атакующая и атакуемая ячейки приняли цвет текущего игрока. | Выполнен |
| Завершение атаки | Игра запущена | 1. Нажать кнопку «Complete atack» | Надпись на кнопке изменилась на «Pass move». | Выполнен |
| Раздача очков | Игра запущена, завершена атака | 1. Выбрать ячейку текущего игрока | Счет выбранной ячейки увеличился на единицу (при условии, что выбранная ячейка не имела максимальное значение счета, равное восьми). | Выполнен |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

### Unit-тесты

Для тестирования класса MoveRunner, отвечающего за логику совершения ходов, были созданы модульные тесты в классе MoveRunnerTest.

В классе MoveRunner были протестированы следующие методы:

* Move;
* IsMove;
* IsCellOccupied.

# Результат работы программы

## Результаты работы консольной реализации

## Результаты работы Windows Forms реализации

Заключение

В результате работы над курсовым проектом была разработана игровая программа «Influence». При разработке были использованы программа Diagram Designer для разработки прототипов пользовательского интерфейса и интегрированная среда разработки Microsoft Visual Studio 2017 Community.

В процессе разработки был использован архитектурный паттерн Модель – Представление – Контроллер, что позволило сделать игру легко модифицируемой. Также это позволило облегчить переносимость игры, что было доказано реализацией игры в консольном и Windows Forms режимах без модификации основных слоёв программы.

В работе были использованы шаблоны проектирования «Одиночка», «Абстрактная фабрика» и «Адаптер».

Кроме того, была использована многопоточность для работы вспомогательных модулей программы, а также для работы управляющего контроллера.

Приложения