Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение  
высшего образования  
«Рязанский государственный радиотехнический университет»  
(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](#_Toc27533777)

[1. Анализ задачи 4](#_Toc27533778)

[1.1. Разработка иерархии классов 4](#_Toc27533779)

[1.1.1. Выделение сущностей 4](#_Toc27533780)

[1.1.2. Зависимости между классами. Диаграмма классов 4](#_Toc27533781)

[1.2. Алгоритмы 4](#_Toc27533782)

[1.3. Разработка интерфейса программы 4](#_Toc27533783)

[2. Написание программы 7](#_Toc27533784)

[2.1.1. Описание разработанных процедур и функций 7](#_Toc27533785)

[2.2. Разработка программы 8](#_Toc27533786)

[2.2.1. Описание классов, перечислений и интерфейсов проекта 8](#_Toc27533787)

[2.3. Описание шаблонов проектирования, которые использовались при написании программы 8](#_Toc27533788)

[2.3.1. Шаблон «Одиночка» (Singletone) 8](#_Toc27533789)

[2.3.2. Шаблон «Абстрактная фабрика» (Abstract Factory) 8](#_Toc27533790)

[2.4. Описание методов рефакторинга, которые использовались при оптимизации исходного кода программы 8](#_Toc27533791)

[2.4.1. Замена магического числа на именованную константу 8](#_Toc27533792)

[2.4.2. Присвоение переменной более ясного или информативного имени 8](#_Toc27533793)

[2.5. Разработка тестов 9](#_Toc27533794)

[3. Результат работы программы 9](#_Toc27533795)

[Заключение 9](#_Toc27533796)

[Приложения 9](#_Toc27533797)

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.

# 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.
* Игровое поле (Game field) – представляет собой совокупность ячеек и реализует логику захвата ячеек игроками.

### Зависимости между классами. Диаграмма классов

[Описание зависимостей между классами, разработка диаграммы классов]

## Алгоритмы

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

Рис. – Блок-схема алгоритма захвата ячейки

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

Рис. – Блок-схема 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. – Главное меню



Рис. – Ввод имени игроков



Рис. – Игровое поле в состоянии захвата ячейки



Рис. – Игровое поле в состоянии раздачи очков



Рис. – Просмотр рекордов



Рис. – Справка

# Написание программы

### Описание разработанных процедур и функций

Описание разработанных процедур и функций представлено в таблице ниже.

| **Название процедуры/функции** | **Описание функциональности** | **Описание параметров** |
| --- | --- | --- |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

## 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 представления (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 следующей таблице.

|  |  |  |
| --- | --- | --- |
| **Пространство имен** | **Название класса / интерфейса / перечисления** | **Назначение** |
|  |  |  |

## 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.

## 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:

Пример кода после рефакторинга:

## Разработка тестов

[Описание последовательности разработки]

# Результат работы программы

[Результаты работы программы с пояснениями и скриншотами]

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» и «Абстрактная фабрика».

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

Приложения