Министерство образования Республики Беларусь

Учреждение образования «Белорусский государственный университет информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

Кафедра программного обеспечения информационных технологий

Дисциплина: Основы алгоритмизации и программирования (ОАиП)

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовому проекту

на тему

**ИГРОВОЕ ПРИЛОЖЕНИЕ «BOMBERMAN-2020»**

БГУИР КП 1-40 01 01 022 ПЗ

Студент: гр. 951009 Семенцова К. Д.

Руководитель: Данилова Г.В.

Минск 2020

СОДЕРЖАНИЕ

ВВЕДЕНИE ……………………………………………………………………………...………5

[1 Анализ аналогов программного средства 7](#_Toc41899599)

[1.1 Игра «Bomb Chicken» 7](#_Toc41899600)

[1.2 Игра «Super Bomberman R» 8](#_Toc41899601)

[1.3 Игра «Space Overlords» 9](#_Toc41899602)

[2 Цели и задачи 10](#_Toc41899603)

[2.1 Постановка цели и задачи курсовой работы 10](#_Toc41899604)

[2.2 Необходимые ресурсы для курсовой работы 10](#_Toc41899605)

[3 Разработка программного средства 12](#_Toc41899606)

[3.1 Работа с изображениями 12](#_Toc41899607)

[3.2 Работа со звуком 12](#_Toc41899608)

[3.3 Создание игрового поля 12](#_Toc41899609)

[3.4 Разработка блоков 14](#_Toc41899613)

[3.5 Разработка модели персонажей 14](#_Toc41899614)

[3.6 Реализация движения персонажа 15](#_Toc41899615)

[3.8 Работа над столкновением 16](#_Toc41899616)

[3.9 Работа с бомбой 16](#_Toc41899617)

[4 Тестирование программного средства 18](#_Toc41899618)

[5 Руководство пользователя 19](#_Toc41899619)

[5.1 Системные требования 19](#_Toc41899620)

[5.2 Инструкция по использованию 19](#_Toc41899621)

[ЗАКЛЮЧЕНИЕ 25](#_Toc41899622)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 26](#_Toc41899623)

[Приложение А**.**](#_Toc41899624) [Исходный код программы 27](#_Toc41899625)

[Приложение Б. Схема метода initLevelController 67](#_Toc8769804)



ВВЕДЕНИЕ

История компьютерных игр начинается с 1940 года, тогда появилась самая первая компьютерная игра и самый первый игровая автомат «Nimatron». В 1947 году свет увидел «Ракетный симулятор», однако он базировался как развлекательное средство, а не как игра. В последующие годы были произведены попытки по переносу классических игр на компьютеры, так в 1948-1950 Алан Тьюринг и Дэйвид Чампернаун разработали «Алгоритм шахматной игры», однако в то время не нашлось достаточно мощного компьютера, чтобы реализовать данный алгоритм. И лишь в марте 1950 года Клодом Шенноном была разработана шахматная программа, которая появилась в статье «Программирование шахматных игр для компьютера», данную статью можно по праву считать первой связанной с проблемой компьютерных шахмат.

После выхода игры «ОХО», компьютерной реализации «крестиков-ноликов», разработанной А.С. Дугласом, компьютерные игры начинают стремительно набирать популярность и делиться на жанры. Так 1960-е годы можно именовать десятилетием зарождения спортивных симуляторов. Так в 1967 году вышло сразу две культовые игры в данном жанре: симулятор баскетбола от Чарли Бачелле и симулятор бейсбола «1967 World Series» от Джэйкоба Бергмана – обе эти игры было написаны на языке BASIC и для компьютера PDP-10. 24 мая 1972 компанией Meganavox была представлена первая игровая приставка Odyssey, с этого момента игры стали доступны обычному пользователю.

С выхода первой игровой приставки прошло около 47 лет, и в связи с бурным ростом производительности домашних ПК и появлением новых технологий произошёл огромный скачок и разделение игровой отрасли, появились новые жанры и игровые платформы. Так на сегодняшний день игры есть на всех известных платформах, начиная с Android, iOS и заканчивая игровыми консолями. Из-за различий в вычислительной мощности игры под каждую платформу начали различаться. На сегодняшний день игровые консоли используются для «Triple A projects», в то время как под мобильные платформы в основном выходят игры жанра «Time killer» (простые в реализации и понимании игры, без особого сюжета, целью которых часто является зарабатывание очков и попадание в таблицу рекордов).

В 1983 году японский разработчик Y. Tanaka разработал игру «Bakudan Otoko» в качестве демонстрации работы компилятора языка программирования Бейсик. В 1983 году Bakudan Otoko была выпущена малым тиражом. В последствие, она стала первой игрой в одноимённой серии.

С момента своего создания игра Бомбермен стала неимоверно популярной. В настоящее время популярность этой игры нисколько не уменьшилась. Игра «Bombermen» имеет захватывающий сюжет.

На данный момент серия состоит из 66 игр. Во всех версиях игрок управляет персонажем, который может класть бомбы, взрывающиеся через несколько секунд. Цель игры – уничтожить всех врагов на уровне и пройти на следующий этап за определённое время.

Целью данного курсовой работы является разработка игрового программного средства «BOMBERMAN-2020». В данной пояснительной записке описаны этапы разработки программы, используемые для этого средства и руководство по использованию.

# АНАЛИЗ АНАЛОГОВ ПРОГРАММНОГО СРЕДСТВА

Оригинальная версия игры «Bomberman» была представлена в 1983 году. В настоящее время существует огромное количество приложений-последователей классического «Bomberman». В течение почти четырех десятилетий появилось множество различных версий данной игры.

Самой популярной версией на данный момент является «Super Bomberman R» 2018 года .

Недавно разработанные персонажи более привлекательные и интересные. Игровой интерфейс игры «Bomberman» полностью эволюционировал, стал ярче, динамичней, но концепция почти не претерпела изменений. В данном разделе будут рассмотрены аналоги создаваемого приложения.

## Игра «Bomb Chicken»

«Bomb Chicken» – пиксельная аркадная пазл-платформерменная игра с видом сбоку, где пользователю предстоит примерить роль курицы, несущей вместо яиц бомбы, разгадывать различные задачи на уровне, обходить препятствия и уничтожать врагов. Внешний вид данного приложения представлен на рисунке 1.1.



Рисунок 1.1 – Интерфейс приложения «Bomb Chicken»

В данной версии был замечен недостаток: движение противников достаточно предсказуемо, что не совпадает с первоначальной концепцией игры и значительно умоляет удовлетворение от игрового процесса.

## 1.2 Игра «Super Bomberman R»

«Super Bomberman R» – реинкарнация серии игр «Bomberman» в современном стиле. В игре действуют все те же простые правила: управляйте персонажем по имени Bomberman, размещайте бомбы и сражайтесь с врагами и соперниками. Но в этот раз вас ждут трехмерные уровни с фотореалистичной графикой. Кроме того, вы можете наслаждаться битвами с противниками со всего мира.

Игра содержит в себе 50 уровней в классическом стиле, основной целью которых является выполнение определённых требований для того, чтобы активировать переход на следующий этап. Как правило, вам придётся зачищать уровень от врагов, взрывая их, или же искать ключи для того, чтобы активировать нужный вам портал. При этом сами уровни разительно отличаются от того, к чему мы привыкли в самой первой игре серии. Игрок видит 3D-локацию в изометрической проекции, кроме того, она теперь полна всевозможных ловушек, механизмов, переключателей, движущихся элементов и крайне непредсказуемых врагов. Уровни отныне не ограничены одной плоскостью. Они могут быть многоярусными, что, наряду с другими нововведениями, привносит немало разнообразия в игровой процесс. Внешний вид основного окна данного приложения представлен на рисунке 1.2.

У данной версии имеется один недостаток: пусть игра является самой новой из всех версий и в ней даже реализованы 3D-эффект, но дизайн игры и игровой интерфейс реализован плохо. Игровые текстуры локации слабо сочетаются с игровыми текстурами персонажей. Создается впечатление незавершенности.



Рисунок 1.2 – Интерфейс приложения «Super Bomberman R»

## 1.3 Игра «Space Overlords»

«Space Overlords» – футуристический экшен с видом сверху. Игроку предстоит руководить древней сущностью, которая веками пыталась выбраться на свободу. И вот, когда она выбралась из заточения, вам предстоит утолить её жажду к разрушениям. События игры происходят на поверхностях различных планет, где предстоит решать какой город будет уничтожен раньше, а какой позже. Пример локаций представлен на рисунке 1.3.



Рисунок 1.3 – Интерфейс приложения «Space Overlords»

Игра имеет достаточно низкий рейтинг. Приложение оставляет плохое впечатление от игрового процесса.

# ЦЕЛИ И ЗАДАЧИ

## Постановка цели и задачи курсовой работы

Целью данной курсовой работы является создание игрового приложения «Bomberman», направленного на разнообразие досуга пользователя.

В результате сравнения аналогов программного средства и анализа предметной области в данном курсовом проекте поставлены следующие задачи:

1. разработать:
   1. алгоритм генерации локации уровня;
   2. систему управления персонажем;
   3. алгоритм передвижения врагов;
   4. алгоритм взрыва бомб;
   5. звуковое сопровождение.
2. создать:
   1. эстетичный и понятный пользователю интерфейс;
   2. гармоничные игровые текстуры.

## 2.2 Необходимые ресурсы для курсовой работы

Для разработки программного средства будет использоваться язык программирования Java. Среда разработки – IntelliJ IDEA. IntelliJ IDEA представляет собой высокотехнологичный комплекс тесно интегрированных инструментов программирования, включающий интеллектуальный редактор исходных текстов с развитыми средствами автоматизации, мощные инструменты рефакторинга кода, встроенную поддержку технологий J2EE, механизмы интеграции со средой тестирования Ant/JUnit и системами управления версиями, уникальный инструмент оптимизации и проверки кода Code Inspection, а также визуальный конструктор графических интерфейсов.

Использование данной среды разработки дает множество возможностей в создании оконных приложений Windows. С помощью данной среды можно разработать качественное оконное приложение, обеспечить удобные средства для ввода, редактирования, сохранения и отображения данных. Также IntelliJ IDEA предлагает широкие возможности при работе с анимацией, что позволяет создать привлекательное игровое приложение.

Проанализировав возможности программного языка Java и среды разработки IntelliJ IDEA, предпочтение было отдано платформе JavaFX. JavaFX позволяет создавать приложения с богатой насыщенной графикой благодаря использованию аппаратного ускорения графики и возможностей GPU.

С помощью JavaFX можно создавать программы для различных операционных систем: Windows, MacOS, Linux и для самых различных устройств: десктопы, смартфоны, планшеты, встроенные устройства, ТВ. Приложение на JavaFX будет работать везде, где установлена исполняемая среда Java (JRE).

JavaFX предоставляет большие возможности по сравнению с рядом других подобных платформ, в частности, по сравнению со Swing. Это и большой набор элементов управления, и возможности по работе с мультимедиа, двухмерной и трехмерной графикой, декларативный способ описания интерфейса с помощью языка разметки FXML, возможность стилизации интерфейса с помощью CSS, интеграция со Swing и многое другое.

Для создания игровых текстур будет использоваться приложение для трехмерного моделирования и печати Pain4 3D и многофункциональный графический редактор Adobe Photoshop.

# РАЗРАБОТКА ПРОГРАММНОГО СРЕДСТВА

## Работа с изображениями

Все большинство объектов, используемые в данном программном средстве, используют те или иные изображения. Все изображения располагаются в директории src/images. Доступ к изображениям происходит по средствам относительных путей к файлам, что позволяет без лишних усилий запускать программу на других устройствах. Для удобства все относительные пути к файлам, содержащим изображения, были записаны константами в классе Constants. Пример загрузки изображения в компонент ImageView представлен ниже.

// Загрузка изображения

Image navigationImg = new Image(Constants.BACK\_TO\_LEVEL\_MENU);

// Помещаем изображение в ImageView

ImageView navigationImageView = new ImageView(navigationImg);

## Работа со звуком

В данном программном средстве присутствует музыкальное сопровождение. Все используемые аудиофайлы располагаются в директории src/music. На протяжении всей игровой сессии воспроизводится 8-и битный аналог современного музыкального произведения. Аналогично доступу к изображениям, доступ к аудиофайлам происходит по средствам относительных путей. Относительные пути к аудиофайлам были вынесены записаны константами в классе Constants. Для воспроизведения аудиофайла в приложении был использован компонент MediaPlayer.

## Создание игрового поля

## Получение информации о выбранном уровне

Создание локации уровня происходит после нажатия на один из уровней в окне levelMenu. На рисунке 3.2 представлено меню уровней.



Рисунок 3.2 – Демонстрация меню уровней

При выборе уровня вызывается метод initLevelController, которому в качестве параметра передается номер необходимого уровня. Метод initeLevelController ответственен за сознание окна уровня. Контекст уровней находится в файлах, располагаемых в директории levels. Каждому уровню соответствует определенный файл.

Файлы, содержащие контекст уровня, имеют свою структуру. В первых двух строках хранится информации о размерах игрового поля. В первой строке указывается количество блоков по горизонтали. Во второй – количество блоков по вертикал. После следует матрица соответствующей размерности. Матрица состоит из цифр от нуля до трех. Каждому значению соответствует свой тип блока.

## Получение содержимого файла

Класс LevelData отвечает за игровой процесс. Конструктор класса LevelData принимает в качестве параметра двумерный массив целочисленных значений.

Метод initLevelController создает экземпляр класса LevelGenerator, передавая в качестве параметра номер выбранного уровня. Метод getBlockCodesFromFile из класса LevelGenerator, генерирует из содержимого выбранного файла матрицу целочисленных значений.

Метод genarateLevelData из класса LevelGenerator, вызывается в методе initLevelController сразу после создания экземпляра класса LevelGenerator. Метод genarateLevelData создает экземпляр класса LevelData, передавая в качестве параметра матрицу, получаемую в методе getBlockCodesFromFile.

Класс LevelGenerator следует принципу single responsibility и занимается только получением содержимого из файла уровня.

## Генерация уровня

В конструкторе класса LevelData генерируется двумерный массив, состоящий из экземпляров класса Block, такой же размерности, как и у полученного в качестве параметра двумерного массива целочисленных значений. Для создания содержимого уровня используется экземпляр класса BlockFactory.

Класс BlockFactory соблюдает паттерн single responsibility и реализует патерн «фабрика», суть которого заключается в инкапсуляции создания экземпляров классов, наследуемых от абстрактного класса Block. Ответственность класса BlockFactory заключается в создании экземпляра класса, наследуемого от класса Block. Метод createBlock из класса BlockFactory в качестве параметра принимает целочисленное значение. В зависимости от значения переданного параметра создаются разные блоки:

– 0 – создается экземпляр класса BackGroundBlock;

– 1 – создается экземпляр класса BreakableBlock;

– 2 – создается экземпляр класса BorderBlock;

– 3 – создается BackGroundBlock.

После, по месту расположения цифры три будут созданы экземпляры класса Enemy.

В случае, если переданный параметр не совпал ни с одним из вышеперечисленных случаев, в консоль выводится сообщение об ошибке.

## Разработка блоков

Абстрактный класс Block наследуется от Pane, что позволяет использовать данный класс как компонент интерфейса в иерархии графических элементов JavaFX. Данный класс содержит в своей структуре следующие абстрактные методы: метод isDestroyable, определяющий разрушим блок или нет; метод isPermeable, определяет проходимость блока; метод getImagePath возвращающий адрес изображения блока. При создании блока вызывается конструктор класса, который создает ImageView блока. В ImageView загружается изображение, полученное из метода getImagePath.

Классы BackGroundBlock, BreakableBlock, BorderBlock и ExplosionBlock являются потомками класса Block . Абстрактные методы родительского класса, предварительно указав аннотацию @Overrid, переопределяются в дочерних классах, формируя необходимые свойства блоков.

## Разработка модели персонажей

Класс Character является абстрактным классом. Данный класс не предполагает создания экземпляров класса, но допускает наследование. Класс Character наследуется от абстрактного класса Block, следовательно, может вызывать методы родителя. От класса Character наследуются все персонажи, задействованные в игровом процессе.

В данном классе находятся методы, которые реализуют основную логику передвижения и поведения персонажей на игровом поле. Все методы класса можно разбить на три группы: методы, ответственные за передвижение; методы, ответственные за взаимодействие персонажей между собой; методы, ответственные, за воздействие взрыва на персонажа.

## 3.6 Реализация движения персонажа

Класс Character имеет методы передвижение персонажа в следующих направлениях: вверх, вниз, вправо, влево. Ниже представлен метод, реализующий передвижения персонажа влево.

public void moveLeft() {

// Получаем максимально допустимое смещений влево

final int availableSteps = getAvailableLeftSteps();

// Смещаем в заданном направлении

setLayoutX(getLayoutX() - availableSteps);

}

Метод getAvailableLeftSteps вызывается для получения доступного смещения в заданном направлении. Данный метод реализован с целью не допускать выход за границы игрового поля. Метод getAvailableLeftSteps представлен ниже.

private int getAvailableLeftSteps() {

for (int i = 1; i <= moveSpeed; i++) {

// Проверяем верхний левый блок на проходимость

final boolean firstBlockIsPermeable =

getBlockPermeableByOffset(-i, 0);

// Проверяем нижний левый угол на проходимость

final boolean secondBlockIsPermeable =

getBlockPermeableByOffset(-i,

Constants.CHARACTER\_SIZE - 1);

// Итерация продолжается пока блоки проходимые

// или пока не завершится цикл

if (!(firstBlockIsPermeable &&

secondBlockIsPermeable)) {

return i - 1;

}

}

// Метод возвращает целочисленное значение от 0 до moveSpeed

return moveSpeed;

}

Передвижение персонажа в другие направления имеют схожую структуру.

## 3.8 Работа над столкновением

Методы, отвечающие за столкновение персонажей, реализованы в классе Character. Метод getCharacterFrame дает возможность, используя текущие угловые координаты, создать фрейм для любого из экземпляров класса.

Для каждой угловой точки создается экземпляр класса Point, который является POJO классом. Класс Point содержит в себе два поля: координата по оси OX; координата по оси OY. Данный класс позволяет добиться определенного уровня абстракции.

Фрейм определяет границы объекта и служит для определения пересечения с другими фреймами. Метод isOverlay проверяет наложение одного фрейма на другой. Если одна из точек любого другого фрейма лежит на границах текущего фрейма, то фиксируется наложение.

Метод isOverlayCharacter выполняет логику столкновения героя и противника. Если фрейм героя пересекается с фреймом врага, приватному полю isAlive героя передается значение false.

## 3.9 Работа с бомбой

Методы, отвечающие за установку бомбы, реализованы в классе Hero, наследуемого от класса Character. Установка бомбы доступна только герою игры, поэтому размещать данный пласт методов в родительском классе не имеет смысла. Количество активных бомб, которые могут располагаться в игровом поле одновременно, ограничено числом три. Для контроля данного фактора в классе Hero имеется приватное поле availableBombCount, выступающее в роли счетчика. При установке бомбы значения переменной декрементируется. После взрыва бомбы, availableBombCount инкрементируется. Перед непосредственной установкой бомбы на игровое поле, проверяются следующие условия:

– место установки является проницаемым блоком;

– количество доступных бомб больше нуля.

Если все условия соблюдены, экземпляр класса Bomb создается на месте блока, по отношению к которому герой находится ближе в текущий момент.

Методы, исполняющие взрыв бомбы, расположены в классе Bomb. Было принято решении о создании класса ExsplosiveTimer, наследуемого от Thread, в качестве таймера для бомбы. Переопределенный метод run потока ExsplosiveTimer выполняет следующие функции:

– вызывает метод sleep у потока на определенное количество миллисекунд;

– по истечению времени, взрывает бомбу.

Реализация класса ExsplosiveTimer представлен ниже.

private class ExplosiveTimer extends Thread {

@Override

public void run() {

try {

// Поток приостанавливается на определенное время

Thread.sleep(Constants.EXPLOSION\_TIME);

// Происходит взрыв бомбы

explosion.execute();

} catch (InterruptedException e) {

// Вывод

e.printStackTrace();

}

}

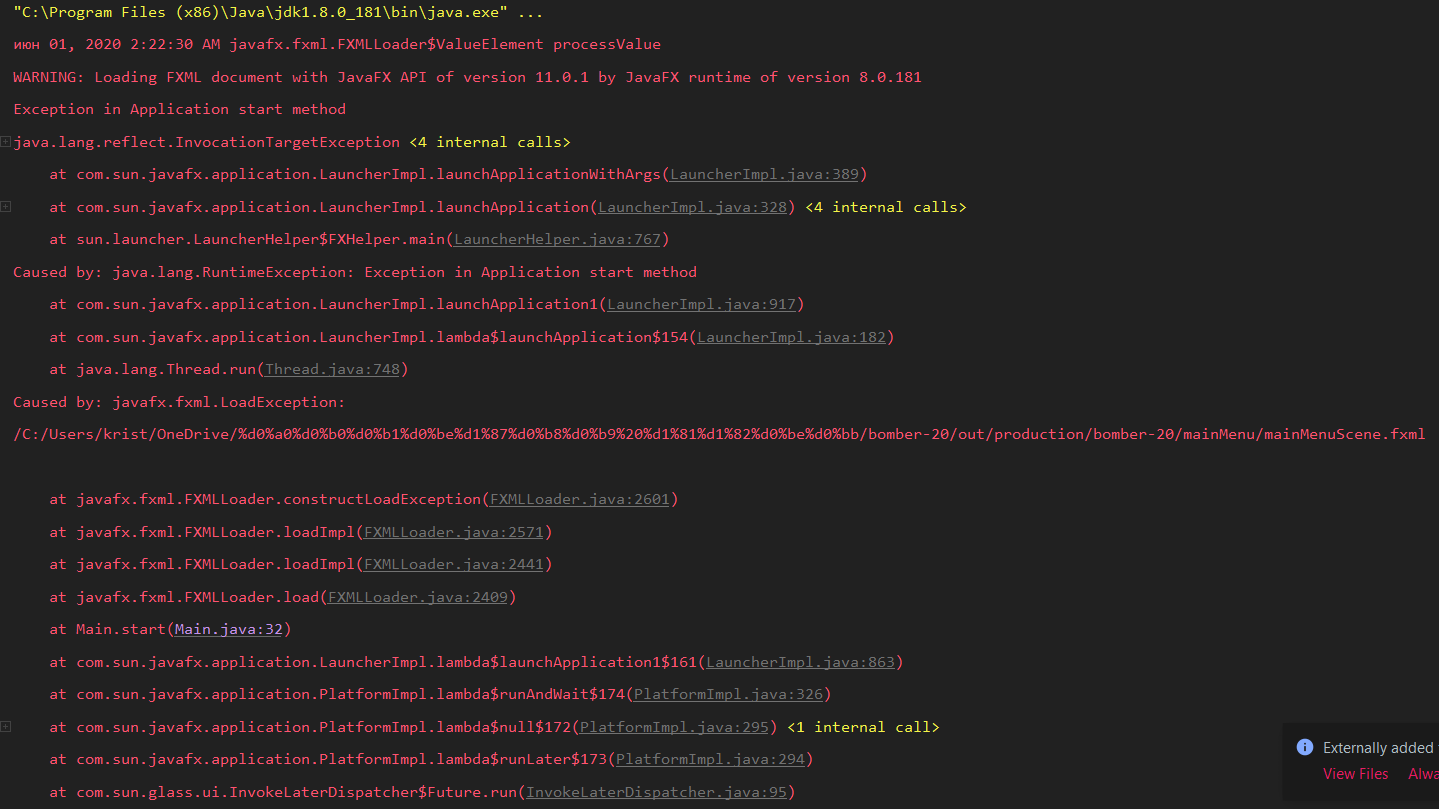
}

# 4 ТЕСТИРОВАНИЕ ПРОГРАММНОГО СРЕДСТВА

В процессе разработки программного средства был выявлен ряд случаев, приводящих к ошибке.

Одной из распространенных проблем стала работа с изображениями. Причиной ошибки в большинстве случаев был ошибочный относительный путь к файлу. Проблема была решена в процессе разработки программы. Сообщение о данной ошибке продемонстрированно на рисунке 4.1.

Во время тестирования первоначальной версии игры была выявлена некорректная работа программного средства. Последствием стало отсутствие взрыва бомбы. Фактором, вызвавшем такой результат была ошибка в логике. Решением данной проблемы заключалось в изменение значения метода, отвечающего за проходимость блока бомбы.

  
Рисунок 4.1 – Ошибка, появляющаяся при работе с ресурсами

Остальные случаи возможного некорректного поведения программы были обнаружены на стадии проектирования и исправлены на стадии разработки.

# 5 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ

В данном разделе приводится вся необходимая информация, которая поможет пользователю в работе с приложением «BOMBERMAN-2020».

## 5.1 Системные требования

Для корректной работы приложения необходимо иметь одну из следующих операционных систем: Windows, MacOS, Linux.

## 5.2 Инструкция по использованию

**5.2.1** Запуск программы

Для запуска программы необходимо открыть исполняемый файл «BOMBERMAN-2020.exe».

**5.2.2** Начало работы

При запуске программы можно увидеть следующий интерфейс, продемонстрированный на рисунке 5.1. Перемещение указателя для выбора режима выполняется курсором мыши. Пункт, на который наведен курсор мыши, подсвечивается желтым цветом, с левой стороны появляется указатель. Для выбора пункта меню необходимо нажать левую кнопку мыши. Приведённые ниже пункты меню выполняют следующие функции:

– NEW GAME – открывает окно с выбором уровня сложности игры;

– INFORMATION – запускает раздел основной информации о игровом приложения;

– EXIT – закрывает игру.



Рисунок 5.1 – Главное меню приложения

По нажатию на пункт меню «INFORMATION», происходит переход в окно, содержащие название игры, авторство, инструкциям к эксплуатации данного программного средства. Окно основной информации о программе представлено на рисунке 5.2.

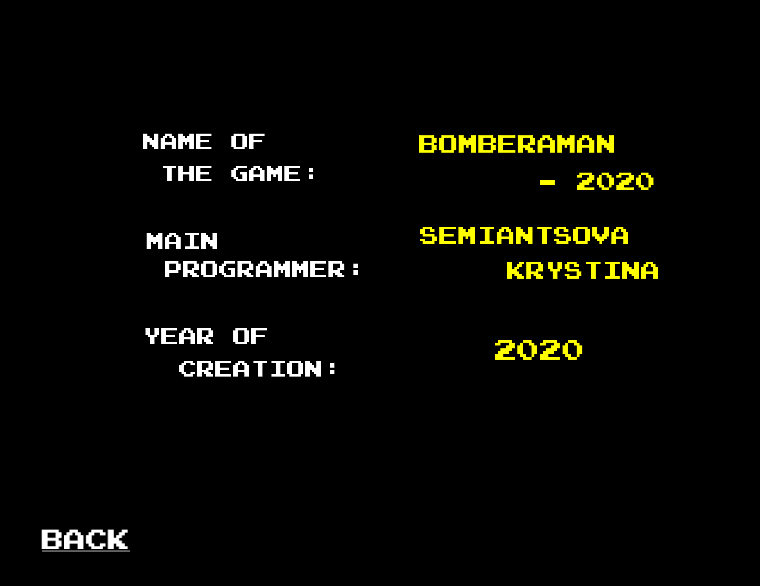


Рисунок 5.2 – Окно основных инструкций

Вернуть в предыдущие окно можно, нажав на кнопку, изображенную на рисунке 5.3.



Рисунок 5.3 – Кнопка возврата в предыдущие окно

Переход к основным структурам игрового процесса осуществляется нажатием левой кнопки мыши по пункту меню «NEW GAME». В результате открывается окно с различными уровнями сложности. Окно выбора уровня сложности представлено на рисунке 5.4.

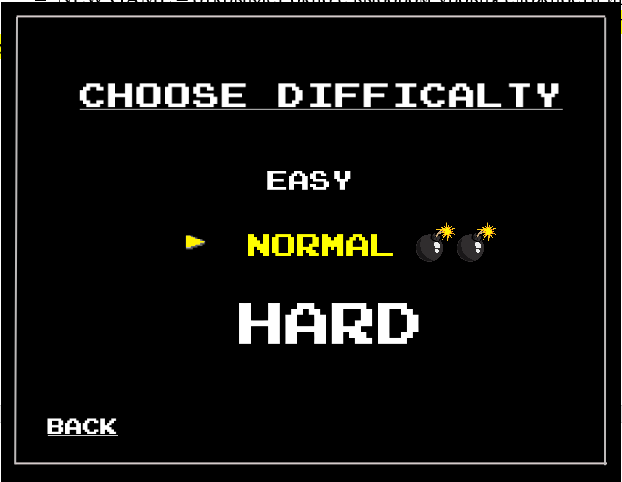


Рисунок 5.4 – Окно «Выбор уровня сложности»

Приведённые выше пункты меню сложности выполняют следующие функции:

– EASY – открывает окно с выбором уровня сложности игры;

– NORMAL – запускает раздел описания игрового приложения, базовый инструктаж по игре;

– HARD – закрывает игру.

Выбрав и перейдя по пункту меню «EASY», происходит переход к меню уровней. На рисунки 5.5 показана визуальное представление меню уровней.



Рисунок 5.5 – Окно меню уровней

Переход по пунктам меню «NORMAL» и «HARD» открывает окно «В разработке». Данное окно представлено на рисунке 5.6.



Рисунок 5.6 – Окно «В разработке»

Выбор любого из пунктов в меню уровней открывает окно уровня, которое генерируется в зависимости от содержимого уровня, и осуществляет переход к игровому процессу как таковому. Окно уровня продемонстрированно на рисунке 5.7.



Рисунок 5.7 – Окно уровня

В зависимости от того, заканчивается игровой процесс победой или поражением открываются соответствующие окна. Если все враги повержены, то открывается панель «YOU WIN2», представленное на рисунке 5.8.

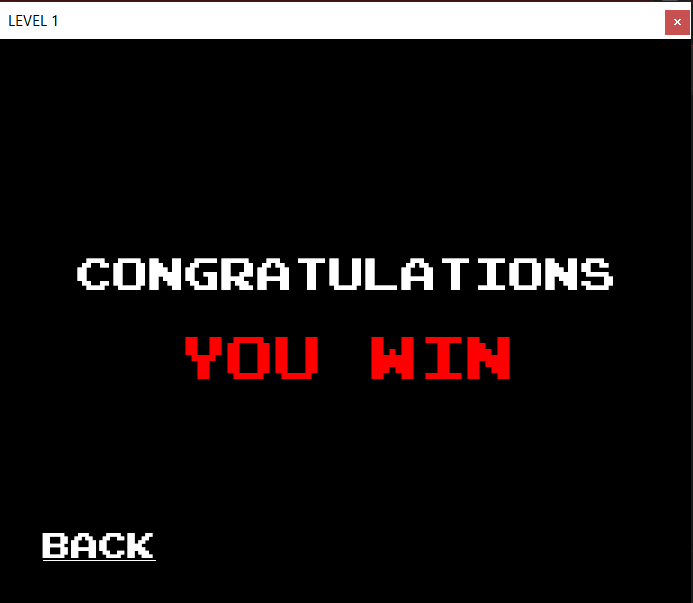


Рисунок 5.8 – Панель «YOU WIN»

Если игровой сеанс заканчивается поражением, открывается панель «YOU DIED». Данное окно представлено на рисунке 5.9.

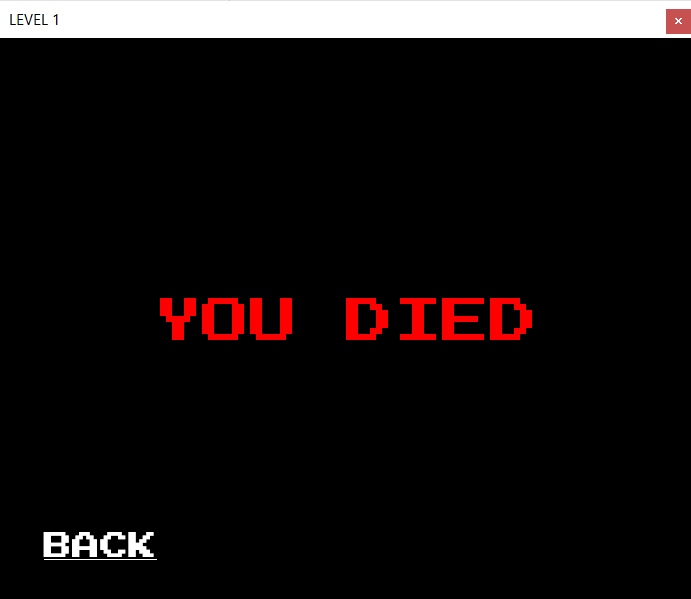


Рисунок 5.9 – Панель «YOU DIED»

Переход по кнопке, изображенной на рисунке 5.2, вернет игровое приложение в меню уровней.

**ЗАКЛЮЧЕНИЕ**

В рамках данного курсового проекта было разработано игровое программное средство «BOMBERMAN-2020», которое обеспечит увлекательное времяпрепровождение. В данном приложении были реализованы следующие функции:

– выбор уровня сложности;

– управление персонажем;

– произвольное движение противников;

– эффект взрыва;

– звукового сопровождения.

Для успешного выполнения всех поставленных целей потребовалось изучить объектно-ориентированные возможности языка Java. Подробно познакомиться с многими компонентами и их возможностями.

Существует множество возможностей для дальнейшего улучшения приложения. Режимы сложности «NORMAL» и «HARD» будут реализованы в последующих версиях игры. Одним из вероятных путей улучшения игры может послужить добавление новых локаций и противников. Следующем вариантом развития является преобразование данного приложения в многопользовательское, добавление режима турнира, используя возможности сервера.

«BOMBERMAN-2020» позволит уйти от рутины и провести время с удовольствием, погрузившись в атмосферу пиксельной игры 90-х.

**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

[1] Java. Библиотека профессионала, том 1. Основы. 9-е изд. : Пер. с англ. – М.: ООО «И.Д.Вильямс», 2014. – 864 с. :ил. – Парад, тит. англ.;

[2] Java 8. Полное руководство; 9-е изд.: Пер. с англ. – М. : ООО «И.Д. Вильяме», 2015. – 1376 с. : ил. – Парал. тит. англ.;

[3] Глухова, Л. А. Основы алгоритмизации и программирования: Лаб. практикум для студ. спец. I-40 01 01 «Программное обеспечение информационных технологий» дневной формы обуч. В 4 ч. / Л. А. Глухова, Е. Е. Фадеева, Е. П. Фадеева. – Минск: БГУИР, 2007. – Ч. 3. – 51 с.

[4] Философия Java. 4-е полное изд. – СПб.: Питер, 2015. – 1168 c.: ил. – (Серия «Классика computer science»).

[5] Структуры данных и алгоритмы в Java. Классика Computers Science. 2-е изд. – СПб.: Питер, 2013. – 704 с.: ил. – (Серия «Классика computer science»);

[6] Vertex Academy [электронный ресурс]. – Электронные данные. – Режим доступа: <https://vertex-academy.com/tutorials/ru/>

[7] GeekBrains [электронный ресурс]. – Электронные данные. – Режим доступа: <https://geekbrains.ru/>

**ПРИЛОЖЕНИЕ А.**

**Исходный код программы**

complexityMenu;

import game.Constants;

import javafx.fxml.FXML;

import javafx.fxml.FXMLLoader;

import javafx.scene.Parent;

import javafx.scene.Scene;

import javafx.scene.image.Image;

import javafx.scene.image.ImageView;

import javafx.stage.Stage;

import javafx.stage.StageStyle;

import utilities.StageUtils;

import java.io.IOException;

public class ComplexityMenuController {

@FXML

private ImageView pointer1;

@FXML

ImageView pointer2;

@FXML

ImageView pointer3;

@FXML

ImageView easyGame;

@FXML

ImageView normalGame;

@FXML

ImageView hardGame;

@FXML

ImageView bomb1;

@FXML

ImageView bomb2;

@FXML

ImageView bomb3;

@FXML

ImageView bomb4;

@FXML

ImageView bomb5;

@FXML

ImageView bomb6;

@FXML

ImageView backImg;

private Image easy;

private Image selEasy;

private Image normal;

private Image selNormal;

private Image hard;

private Image selHard;

private Image back;

private Image selBack;

@FXML

public void initialize() {

easy = new Image(Constants.EASY\_IMAGE);

selEasy = new Image(Constants.SELECTED\_EASY\_IMAGE);

normal = new Image(Constants.NORMAL\_IMAGE);

selNormal = new

Image(Constants.SELECTED\_NORMAL\_IMAGE);

hard = new Image(Constants.HARD\_IMAGE);

selHard = new Image(Constants.SELECTED\_HARD\_IMAGE);

back = new Image(Constants.BACK\_TO\_LEVEL\_MENU);

selBack = new

Image(Constants.SEL\_BACK\_TO\_LEVEL\_MENU);

}

@FXML

public void onEasyGamePressed() throws IOException {

FXMLLoader loader = new FXMLLoader(getClass().

getResourc

e("../levelMenu/levelMenuScene.fxml"));

Parent root = (Parent) loader.load();

Stage stage = new Stage();

stage.setTitle("BOMBERMEN");

stage.setScene(new Scene(root, 640, 480));

stage.initStyle(StageStyle.TRANSPARENT);

stage.show();

StageUtils.closeStage(pointer2);

}

@FXML

public void onNormalGamePressed() throws IOException {

loadUnderConstructionScene();

}

private void loadUnderConstructionScene() throws

IOException

{

FXMLLoader loader = new FXMLLoader(getClass().

getResource

("../underConstruction/

underConstructionScene.fxml"));

Parent root = (Parent) loader.load();

Stage stage= new Stage();

stage.setTitle("BOMBERMEN");

stage.initStyle(StageStyle.TRANSPARENT);

stage.setScene(new Scene(root, 620, 480));

stage.show();

StageUtils.closeStage(pointer2);

}

@FXML

public void onBackToLevelMenu() throws IOException {

FXMLLoader loader = new FXMLLoader(getClass().

getResource

("../mainMenu/mainMenuScene.fxml"));

Parent root = (Parent) loader.load();

Stage stage= new Stage();

stage.setTitle("BOMBERMEN");

stage.initStyle(StageStyle.TRANSPARENT);

stage.setScene(new Scene(root, 620, 480));

stage.show();

StageUtils.closeStage(pointer2);

}

@FXML

public void onHardGamePressed() throws IOException {

loadUnderConstructionScene();

}

private void setInvisible(){

bomb1.setVisible(false);

bomb2.setVisible(false);

bomb3.setVisible(false);

bomb4.setVisible(false);

bomb5.setVisible(false);

bomb6.setVisible(false);

}

private void setUnselected() {

setInvisible();

pointer1.setVisible(false);

easyGame.setImage(easy);

pointer2.setVisible(false);

normalGame.setImage(normal);

pointer3.setVisible(false);

hardGame.setImage(hard);

backImg.setImage(back);

}

@FXML

public void onEasyEntered() {

setUnselected();

pointer1.setVisible(true);

easyGame.setImage(selEasy);

bomb1.setVisible(true);

}

@FXML

public void onNormalEntered() {

setUnselected();

pointer2.setVisible(true);

normalGame.setImage(selNormal);

bomb2.setVisible(true);

bomb3.setVisible(true);

}

@FXML

public void onHardEntered() {

setUnselected();

pointer3.setVisible(true);

hardGame.setImage(selHard);

bomb4.setVisible(true);

bomb5.setVisible(true);

bomb6.setVisible(true);

}

@FXML

public void onBackEntered(){

setUnselected();

backImg.setImage(selBack);

}

}

package game.blocks;

import game.Constants;

public class BackgroundBlock extends Block {

public BackgroundBlock() {

super();

}

@Override

public boolean isDestroyable() {

return true;

}

@Override

public boolean isPermeable() {

return true;

}

@Override

protected String getImagePath() {

return Constants.BACKGROUND\_BLOCK\_IMAGE;

}

}

package game.blocks;

import javafx.scene.image.Image;

import javafx.scene.image.ImageView;

import javafx.scene.layout.Pane;

public abstract class Block extends Pane {

protected Block() {

final Image image = new Image(getClass().

getResourceAsStream(getImagePath()));

final ImageView imageView = new ImageView(image);

this.getChildren().add(imageView);

}

public abstract boolean isDestroyable();

public abstract boolean isPermeable();

protected abstract String getImagePath();

}

package game.blocks;

import com.sun.istack.internal.NotNull;

public class BlockFactory {

private static final int BACKGROUND\_CODE = 0;

private static final int BREAKABLE\_CODE = 1;

private static final int BORDER\_CODE = 2;

private static final int ENEMY\_CODE = 3;

@NotNull

public Block createBlock(final int blockCode) {

switch (blockCode) {

case BACKGROUND\_CODE:

case ENEMY\_CODE:

return new BackgroundBlock();

case BREAKABLE\_CODE:

return new BreakableBlock();

case BORDER\_CODE:

return new BorderBlock();

default:

throw new IllegalArgumentException("Illegal

block code.");

}

}

public boolean isEnemy(final int code){

return code == ENEMY\_CODE;

}

}

Класс

package game.blocks;

import game.Constants;

public class BreakableBlock extends Block {

@Override

public boolean isDestroyable() {

return true;

}

@Override

public boolean isPermeable() {

return false;

}

@Override

protected String getImagePath() {

return Constants.BREAKABLE\_BOCK\_IMAGE;

}

}

package game.blocks;

import game.Constants;

import utilities.Executable;

public class ExplosionBlock extends Block {

Executable fireTask;

public ExplosionBlock(Executable fireTask) {

this.fireTask = fireTask;

new FireThread().start();

}

@Override

public boolean isDestroyable() {

return true;

}

@Override

public boolean isPermeable() {

return false;

}

@Override

protected String getImagePath() {

return Constants.EXPLOSION\_BLOCK\_IMAGE;

}

private class FireThread extends Thread{

@Override

public void run() {

try {

sleep(180);

} catch (InterruptedException e) {

e.printStackTrace();

}

fireTask.execute();

}

}

}

package game.characters;

import game.Constants;

import game.LevelData;

import game.Point;

import game.blocks.Block;

public abstract class Character extends Block {

private boolean isAlive = true;

private int moveSpeed = 3;

final LevelData levelData;

Character(LevelData levelData) {

this.levelData = levelData;

}

public boolean isAlive() {

return isAlive;

}

private int getAvailableLeftSteps() {

for (int i = 1; i <= moveSpeed; i++) {

final boolean firstBlockIsPermeable =

getBlockPermeableByOffset(-i, 0);

final boolean secondBlockIsPermeable =

getBlockPermeableByOffset(-i,

Constants.CHARACTER\_SIZE - 1);

if (!(firstBlockIsPermeable &&

secondBlockIsPermeable)) {

return i - 1;

}

}

return moveSpeed;

}

private int getAvailableRightSteps() {

for (int i = 1; i <= moveSpeed; i++) {

final boolean firstBlockIsPermeable =

getBlockPermeableByOffset

(Constants.CHARACTER\_SIZE + i, 0);

final boolean secondBlockIsPermeable =

getBlockPermeableByOffset

(Constants.CHARACTER\_SIZE + i,

Constants.CHARACTER\_SIZE - 1);

if (!(firstBlockIsPermeable &&

secondBlockIsPermeable)) {

return i - 1;

}

}

return moveSpeed;

}

private int getAvailableUpSteps() {

for (int i = 1; i <= moveSpeed; i++) {

final boolean firstBlockIsPermeable =

getBlockPermeableByOffset(0, -1);

final boolean secondBlockIsPermeable =

getBlockPermeableByOffset

(Constants.CHARACTER\_SIZE - 1, -1);

if (!(firstBlockIsPermeable &&

secondBlockIsPermeable)) {

return i - 1;

}

}

return moveSpeed;

}

private int getAvailableDownSteps() {

for (int i = 1; i <= moveSpeed; i++) {

final boolean firstBlockIsPermeable =

getBlockPermeableByOffset(0,

Constants.CHARACTER\_SIZE + i);

final boolean secondBlockIsPermeable =

getBlockPermeableByOffset

(Constants.CHARACTER\_SIZE - 1,

Constants.CHARACTER\_SIZE + i);

if (!(firstBlockIsPermeable &&

secondBlockIsPermeable)) {

return i - 1;

}

}

return moveSpeed;

}

public void moveLeft() {

final int availableSteps = getAvailableLeftSteps();

setLayoutX(getLayoutX() - availableSteps);

}

public void moveRight() {

final int availableSteps = getAvailableRightSteps();

setLayoutX(getLayoutX() + availableSteps);

}

public void moveUp() {

final int availableSteps = getAvailableUpSteps();

setLayoutY(getLayoutY() - availableSteps);

}

public void moveDown() {

final int availableSteps = getAvailableDownSteps();

setLayoutY(getLayoutY() + availableSteps);

}

public void explosive(final int posX, final int posY) {

final double cordX = getLayoutX();

final double cordY = getLayoutY();

final int characterPosX = LevelData.getPositionByCoordinate(cordX);

final int characterPosY = LevelData.getPositionByCoordinate(cordY);

if (characterPosX == posX && characterPosY == posY) {

isAlive = false;

}

}

private boolean getBlockPermeableByOffset(final int offsetX, final int offsetY) {

final int posX = (int) (getLayoutX() + offsetX);

final int posY = (int) (getLayoutY() + offsetY);

final Block block = levelData.getBlockByCoordinates(posX, posY);

return block.isPermeable();

}

private CharacterFrame getCharacterFrame() {

final Point topLeftPoint = new Point(getLayoutX(), getLayoutY());

final Point topRightPoint = new Point(getLayoutX() + Constants.CHARACTER\_SIZE, getLayoutY());

final Point downLeftPoint = new Point(getLayoutX(), getLayoutY() + Constants.CHARACTER\_SIZE);

final Point downRightPoint = new Point(getLayoutX() + Constants.CHARACTER\_SIZE,

getLayoutY() + Constants.CHARACTER\_SIZE);

return new CharacterFrame(topLeftPoint, topRightPoint, downLeftPoint, downRightPoint);

}

public boolean isOverlayCharacter(final Character other){

final CharacterFrame otherFrame = other.getCharacterFrame();

final CharacterFrame thisFrame = getCharacterFrame();

if (thisFrame.isOverLay(otherFrame)){

isAlive = false;

return true;

}

return false;

}

}

package game.characters;

import game.Point;

class CharacterFrame {

private final Point topLeftPoint;

private final Point topRightPoint;

private final Point downLeftPoint;

private final Point downRightPoint;

CharacterFrame(final Point topLeftPoint,

final Point topRightPoint,

final Point downLeftPoint,

final Point downRightPoint) {

this.topLeftPoint = topLeftPoint;

this.topRightPoint = topRightPoint;

this.downLeftPoint = downLeftPoint;

this.downRightPoint = downRightPoint;

}

boolean isOverLay(final CharacterFrame other) {

if (isPointInFrame(other.getTopLeftPoint())) {

return true;

}

if (isPointInFrame(other.getTopRightPoint())) {

return true;

}

if (isPointInFrame(other.getDownRightPoint())) {

return true;

}

if (isPointInFrame(other.getDownLeftPoint())) {

return true;

}

return false;

}

private boolean isPointInFrame(final Point otherPoint) {

final int otherX = otherPoint.getCordX();

final int otherY = otherPoint.getCordY();

final int tlX = topLeftPoint.getCordX();

final int tlY = topLeftPoint.getCordY();

final int drX = downRightPoint.getCordX();

final int drY = downRightPoint.getCordY();

return otherX > tlX && otherX < drX && otherY > tlY && otherY < drY;

}

private Point getTopLeftPoint() {

return topLeftPoint;

}

private Point getTopRightPoint() {

return topRightPoint;

}

private Point getDownLeftPoint() {

return downLeftPoint;

}

private Point getDownRightPoint() {

return downRightPoint;

}

}

Класс

package game.characters;

import game.Constants;

import game.LevelData;

public class Hero extends Character {

private final Object LOCK = new Object();

private int availableBombCount = 3;

public Hero(LevelData levelData) {

super(levelData);

}

@Override

public boolean isDestroyable() {

return false;

}

@Override

public boolean isPermeable() {

return false;

}

@Override

protected String getImagePath() {

return Constants.CHARACTER\_IMAGE;

}

public void setBomb() {

final int bombPosX = LevelData.getPositionByCoordinate(getLayoutX());

final int bombPosY = LevelData.getPositionByCoordinate(getLayoutY());

levelData.plantBomb(bombPosX, bombPosY);

availableBombCount--;

}

public void releaseBomb() {

synchronized (LOCK) {

availableBombCount++;

}

}

private boolean hasSpaceForBomb() {

final int posX = LevelData.getPositionByCoordinate(getLayoutX());

final int posY = LevelData.getPositionByCoordinate(getLayoutY());

return levelData.getBlockByPosition(posX, posY).isPermeable();

}

public boolean canPlantBomb() {

return ((availableBombCount > 0) && hasSpaceForBomb() && isAlive());

}

}

Класс

package game;

import game.blocks.Block;

import utilities.Executable;

public class Bomb extends Block {

private final Executable explosion;

Bomb(Executable explosion) {

this.explosion = explosion;

new ExplosiveTimer().start();

}

@Override

public boolean isDestroyable() {

return true;

}

@Override

public boolean isPermeable() {

return true;

}

@Override

protected String getImagePath() {

return Constants.BOMB\_IMAGE;

}

private class ExplosiveTimer extends Thread {

@Override

public void run() {

try {

Thread.sleep(Constants.EXPLOSION\_TIME);

explosion.execute();

} catch (InterruptedException e) {

e.printStackTrace();

}

}

}

}

package game;

public class Constants {

static final int BLOCK\_SIZE = 64;

public static final int CHARACTER\_SIZE = 48;

static int EXPLOSION\_TIME = 3 \* 1000;

static int EXPLOSIVE\_POWER = 2;

public static final String CHARACTER\_IMAGE = "../../images/bomberman.png";

public static final String ENEMY\_IMAGE = "../../images/enemy.png";

static final String BOMB\_IMAGE = "../images/bomb1.png";

public static final String UNBREAKABLE\_BLOCK\_IMAGE = "../../images/unB.png";

public static final String BREAKABLE\_BOCK\_IMAGE = "../../images/br.png";

public static final String BACKGROUND\_BLOCK\_IMAGE = "../../images/bg.png";

public static final String EXPLOSION\_BLOCK\_IMAGE = "../../images/fireHigh.png";

public static final String START\_IMAGE = "images/NG.png";

public static final String SELECTED\_START\_IMAGE = "images/selNG.png";

public static final String INF\_IMAGE = "images/inf.png";

public static final String SELECTED\_INF\_IMAGE = "images/selInf.png";

public static final String EXIT\_IMAGE = "images/exit.png";

public static final String SELECTED\_EXIT\_IMAGE = "images/selExit.png";

public static final String EASY\_IMAGE = "images/easy.png";

public static final String SELECTED\_EASY\_IMAGE = "images/selEasy.png";

public static final String NORMAL\_IMAGE = "images/normal.png";

public static final String SELECTED\_NORMAL\_IMAGE = "images/selNormal.png";

public static final String HARD\_IMAGE = "images/hard.png";

public static final String SELECTED\_HARD\_IMAGE = "images/selHard.png";

public static final String LEV1\_IMAGE = "images/level1.png";

public static final String SEL\_LEV1\_IMAGE = "images/sel1.png";

public static final String LEV2\_IMAGE = "images/level2.png";

public static final String SEL\_LEV2\_IMAGE = "images/sel2.png";

public static final String LEV3\_IMAGE = "images/level3.png";

public static final String SEL\_LEV3\_IMAGE = "images/sel3.png";

public static final String LEV4\_IMAGE = "images/level4.png";

public static final String SEL\_LEV4\_IMAGE = "images/sel4.png";

public static final String LEV5\_IMAGE = "images/level5.png";

public static final String SEL\_LEV5\_IMAGE = "images/sel5.png";

public static final String LEV6\_IMAGE = "images/level6.png";

public static final String SEL\_LEV6\_IMAGE = "images/sel6.png";

public static final String LEV7\_IMAGE = "images/level7.png";

public static final String SEL\_LEV7\_IMAGE = "images/sel7.png";

public static final String LEV8\_IMAGE = "images/level8.png";

public static final String SEL\_LEV8\_IMAGE = "images/sel8.png";

public static final String LEV9\_IMAGE = "images/level9.png";

public static final String SEL\_LEV9\_IMAGE = "images/sel9.png";

public static final String LEV10\_IMAGE = "images/level10.png";

public static final String SEL\_LEV10\_IMAGE = "images/sel10.png";

public static final String YOU\_DIED\_IMAGE = "images/youDied.png";

public static final String WIN\_IMAGE = "images/youWin.png";

public static final String BACK\_TO\_LEVEL\_MENU = "images/backToLevelMenu.png";

public static final String SEL\_BACK\_TO\_LEVEL\_MENU = "images/selBackToLevMenu.png";

public static final String MEDIA\_URL\_1 = "src/music/mp1.mp3";

}

package game;

import com.sun.istack.internal.Nullable;

import game.blocks.BackgroundBlock;

import game.blocks.Block;

import game.blocks.BlockFactory;

import game.blocks.ExplosionBlock;

import game.characters.Character;

import game.characters.Enemy;

import game.characters.Hero;

import utilities.Executable;

import java.util.ArrayList;

import java.util.List;

public class LevelData {

private boolean gameInProcess = true;

private boolean needRebuilding = true;

private Hero bomberman;

public Hero getBomberman() {

return bomberman;

}

public List<Enemy> getEnemies() {

return enemies;

}

public Boolean getGameInProcess() {

return gameInProcess;

}

public Boolean getNeedRebuilding() {

return needRebuilding;

}

public void setNeedRebuilding() {

this.needRebuilding = false;

}

private Block[][] levelContent;

private ArrayList<Enemy> enemies;

private final BlockFactory blockFactory;

LevelData(final int[][] blockCodes) {

blockFactory = new BlockFactory();

levelContent = new Block[blockCodes.length][blockCodes[0].length];

for (int i = 0; i < levelContent.length; i++) {

for (int j = 0; j < levelContent[0].length; j++) {

levelContent[i][j] = blockFactory.createBlock(blockCodes[i][j]);

}

}

creteHero();

createEnemy(blockCodes);

new OverlayThread().start();

}

private boolean isActive() {

return bomberman.isAlive();

}

private void creteHero() {

bomberman = new Hero(this);

bomberman.setLayoutX(getBlockCordX(1));

bomberman.setLayoutY(getBlockCordY(1));

}

private void createEnemy(final int[][] blockCodes) {

enemies = new ArrayList<>();

for (int i = 0; i < blockCodes.length; i++) {

for (int j = 0; j < blockCodes[0].length; j++) {

final int code = blockCodes[i][j];

if (blockFactory.isEnemy(code)) {

final Enemy enemy = new Enemy(this);

enemy.setLayoutX(getBlockCordX(i));

enemy.setLayoutY(getBlockCordY(j));

enemies.add(enemy);

}

}

}

}

public int getWidth() {

return levelContent.length;

}

public int getHeight() {

return levelContent[0].length;

}

@Nullable

public Block getBlockByPosition(final int posX, final int posY) {

if (posX < 0 || posY < 0) {

return null;

}

if (posX >= levelContent.length || posY >= levelContent[0].length) {

return null;

}

return levelContent[posX][posY];

}

public Block getBlockByCoordinates(final int cordX, final int cordY) {

final int posX = cordX / Constants.BLOCK\_SIZE;

final int posY = cordY / Constants.BLOCK\_SIZE;

return getBlockByPosition(posX, posY);

}

public static int getPositionByCoordinate(final double cord) {

return ((int) cord / Constants.BLOCK\_SIZE);

}

public void plantBomb(final int posX, final int posY) {

if (bomberman.canPlantBomb()) {

final Executable explosiveTask = new ExplosiveTask(posX, posY);

final Block bomb = new Bomb(explosiveTask);

levelContent[posX][posY] = bomb;

needRebuilding = true;

}

}

public double getPaneHeight() {

return getHeight() \* Constants.BLOCK\_SIZE;

}

public double getPaneWidth() {

return getWidth() \* Constants.BLOCK\_SIZE;

}

public double getBlockCordX(final int posX) {

return posX \* Constants.BLOCK\_SIZE;

}

public double getBlockCordY(final int posY) {

return posY \* Constants.BLOCK\_SIZE;

}

private void explosion(final int posX, final int posY) {

levelContent[posX][posY] = new ExplosionBlock(new FireTask(posX, posY));

//todo анимация

needRebuilding = true;

}

private class FireTask implements Executable {

private final int posX;

private final int posY;

private FireTask(int posX, int posY) {

this.posX = posX;

this.posY = posY;

}

@Override

public void execute() {

levelContent[posX][posY] = new BackgroundBlock();

needRebuilding = true;

}

}

private class ExplosiveTask implements Executable {

private final int posX;

private final int posY;

ExplosiveTask(final int posX, final int posY) {

this.posX = posX;

this.posY = posY;

}

@Override

public void execute() {

bomberman.releaseBomb();

explosiveTop();

explosiveBottom();

explosiveLeft();

explosiveRight();

;

explosion(posX, posY);

}

private void explosiveTop() {

boolean active = true;

for (int i = 0; i < Constants.EXPLOSIVE\_POWER && active; i++) {

final int posY = this.posY - i;

final Block block = getBlockByPosition(posX, posY);

if (block != null) {

active = block.isDestroyable();

if (active) {

explosion(posX, posY);

}

}

notifyExplosive(posX, posY);

}

}

private void explosiveBottom() {

boolean active = true;

for (int i = 0; i < Constants.EXPLOSIVE\_POWER && active; i++) {

final int posY = this.posY + i;

final Block block = getBlockByPosition(posX, posY);

if (block != null) {

active = block.isDestroyable();

if (active) {

explosion(posX, posY);

}

}

notifyExplosive(posX, posY);

}

}

private void explosiveLeft() {

boolean active = true;

for (int i = 0; i < Constants.EXPLOSIVE\_POWER && active; i++) {

final int posX = this.posX - i;

final Block block = getBlockByPosition(posX, posY);

if (block != null) {

active = block.isDestroyable();

if (active) {

explosion(posX, posY);

}

}

notifyExplosive(posX, posY);

}

}

private void explosiveRight() {

boolean active = true;

for (int i = 0; i < Constants.EXPLOSIVE\_POWER && active; i++) {

final int posX = this.posX + i;

final Block block = getBlockByPosition(posX, posY);

if (block != null) {

active = block.isDestroyable();

if (active) {

explosion(posX, posY);

}

}

notifyExplosive(posX, posY);

}

}

private void notifyExplosive(final int posX, final int posY) {

bomberman.explosive(posX, posY);

for (final Enemy enemy : enemies) {

enemy.explosive(posX, posY);

}

}

}

private class OverlayThread extends Thread {

@Override

public void run() {

while (LevelData.this.isActive()) {

for (final Character enemy : enemies) {

if (bomberman.isOverlayCharacter(enemy)) {

needRebuilding = true;

}

}

}

}

}

}

package game;

import java.io.File;

import java.io.FileNotFoundException;

import java.util.Scanner;

public class LevelGenerator {

private static final String BASE\_FILE\_PATH = "./src/levels/level";

private final int levelNumber;

public LevelGenerator(final int levelNumber) {

this.levelNumber = levelNumber;

}

private String getFilePath() {

return BASE\_FILE\_PATH + levelNumber;

}

private int[][] getBlockCodesFromFile() {

final String filePath = getFilePath();

Scanner scan = null;

try {

scan = new Scanner(new File(filePath));

} catch (FileNotFoundException e) {

e.printStackTrace();

}

assert scan != null;

final int matrixWidth = scan.nextInt();

int matrixHeight = scan.nextInt();

final int[][] matrix = new int[matrixWidth][matrixHeight];

for (int i = 0; i < matrixHeight; i++) {

for (int j = 0; j < matrixWidth; j++) {

matrix[j][i] = scan.nextInt();

}

}

return matrix;

}

public LevelData generateLevelData() {

return new LevelData(getBlockCodesFromFile());

}

}

Класс

package game;

public class Point {

private final int cordX;

private final int cordY;

public Point(final double cordX, final double cordY){

this.cordX = (int) cordX;

this.cordY = (int) cordY;

}

public int getCordX(){

return cordX;

}

public int getCordY() {

return cordY;

}

}

Класс

package infScene;

import game.Constants;

import javafx.fxml.FXML;

import javafx.fxml.FXMLLoader;

import javafx.scene.Parent;

import javafx.scene.Scene;

import javafx.scene.image.Image;

import javafx.scene.image.ImageView;

import javafx.stage.Stage;

import javafx.stage.StageStyle;

import utilities.StageUtils;

import java.io.IOException;

public class InformationSceneController {

@FXML

ImageView backImg;

private Image back;

private Image selBack;

@FXML

public void initialize() {

back = new Image(Constants.BACK\_TO\_LEVEL\_MENU);

selBack = new Image(Constants.SEL\_BACK\_TO\_LEVEL\_MENU);

}

@FXML

private void onBackToLevelMenu() throws IOException {

FXMLLoader loader = new FXMLLoader(getClass().getResource("../mainMenu/mainMenuScene.fxml"));

Parent root = (Parent) loader.load();

Stage stage = new Stage();

stage.setTitle("BOMBERMEN");

stage.setScene(new Scene(root, 620, 480));

stage.initStyle(StageStyle.TRANSPARENT);

stage.show();

StageUtils.closeStage(backImg);

}

@FXML

public void onBackEntered() {

backImg.setImage(selBack);

}

@FXML

public void onBackExited() {

backImg.setImage(back);

}

}

Файл

package level;

import game.Constants;

import game.characters.Enemy;

import game.characters.Hero;

import game.LevelData;

import game.blocks.Block;

import javafx.application.Platform;

import javafx.fxml.FXML;

import javafx.fxml.FXMLLoader;

import javafx.scene.Parent;

import javafx.scene.Scene;

import javafx.scene.image.Image;

import javafx.scene.image.ImageView;

import javafx.scene.input.KeyCode;

import javafx.scene.layout.Pane;

import javafx.stage.Stage;

import javafx.stage.StageStyle;

import utilities.StageUtils;

import java.io.IOException;

import java.util.List;

public class LevelController{

private LevelData levelData;

@FXML

private Pane root;

private void setListeners() {

final Scene scene = root.getScene();

scene.setOnKeyPressed(event -> {

if ((event.getCode() == KeyCode.UP) || event.getCode() == KeyCode.W) {

levelData.getBomberman().moveUp();

}

if ((event.getCode() == KeyCode.DOWN) || event.getCode() == KeyCode.S) {

levelData.getBomberman().moveDown();

}

if ((event.getCode() == KeyCode.LEFT) || event.getCode() == KeyCode.A) {

levelData.getBomberman().moveLeft();

}

if ((event.getCode() == KeyCode.RIGHT) || event.getCode() == KeyCode.D) {

levelData.getBomberman().moveRight();

}

if (event.getCode() == KeyCode.SPACE) {

levelData.getBomberman().setBomb();

}

});

}

public void setLevelData(final LevelData levelData) {

this.levelData = levelData;

initPane();

setListeners();

}

private void initPane() {

final double paneHeight = levelData.getPaneHeight();

final double paneWidth = levelData.getPaneWidth();

root.setPrefHeight(paneHeight);

root.setPrefWidth(paneWidth);

renderBackground();

new BackgroundRenderingRunnable().start();

}

private void renderBackground() {

clearRoot();

for (int posX = 0; posX < levelData.getWidth(); posX++) {

for (int posY = 0; posY < levelData.getHeight(); posY++) {

final Block block = levelData.getBlockByPosition(posX, posY);

block.setLayoutX(levelData.getBlockCordX(posX));

block.setLayoutY(levelData.getBlockCordY(posY));

root.getChildren().add(block);

}

}

List<Enemy> enemies = levelData.getEnemies();

boolean haveEnemies = false;

for (final Enemy enemy : enemies) {

if (enemy.isAlive()) {

root.getChildren().add(enemy);

haveEnemies = true;

}

}

final Hero bomberman = levelData.getBomberman();

if (bomberman.isAlive()) {

root.getChildren().

add(levelData.getBomberman());

if(!haveEnemies) {

endOfTheGame(Constants.WIN\_IMAGE);

}

} else {

endOfTheGame(Constants.YOU\_DIED\_IMAGE);

}

}

private void endOfTheGame(final String imagePath) {

Pane gameOverPane = new Pane();

gameOverPane.setPrefWidth(root.getWidth());

gameOverPane.setPrefHeight(root.getHeight());

gameOverPane.setStyle("-fx-background-color:

#000000");

root.getChildren().add(gameOverPane);

Image img = new Image(imagePath);

ImageView imageView = new ImageView(img);

int temp = (int) ((root.getWidth() / 2) - (int)

(img.getWidth() / 2));

imageView.setLayoutX(temp);

temp = (int) ((root.getHeight() / 2) - (int)

(img.getHeight() /2));

imageView.setLayoutY(temp);

root.getChildren().add(imageView);

Image navigationImg = new Image

(Constants.BACK\_TO\_LEVEL\_MENU);

Image selNavigationImg = new Image

(Constants.SEL\_BACK\_TO\_LEVEL\_MENU);

ImageView navigationImageView = new ImageView

(navigationImg);

navigationImageView.setLayoutX(30);

temp = (int) (root.getHeight()) –

(int)(navigationImg.getHeight()) - 30;

navigationImageView.setLayoutY(temp);

navigationImageView.setOnMouseEntered(e ->

navigationImageView.setImage(selNavigationImg));

navigationImageView.setOnMouseExited(e ->

navigationImageView.setImage(navigationImg));

navigationImageView.setOnMousePressed(e -> {

FXMLLoader loader = new FXMLLoader(getClass().

getResource("../levelMenu/

levelMenuScene.fxml"));

Parent root = null;

try {

root = (Parent) loader.load();

} catch (IOException ex) {

ex.printStackTrace();

}

Stage stage = new Stage();

stage.setTitle("BOMBERMEN");

stage.setScene(new Scene(root, 620, 480));

stage.initStyle(StageStyle.TRANSPARENT);

stage.show();

StageUtils.closeStage(gameOverPane);

});

root.getChildren().add(navigationImageView);

}

private void clearRoot() {

root.getChildren().clear();

}

private class BackgroundRenderingRunnable extends Thread

{

@Override

public void run() {

while (levelData.getGameInProcess()) {

if (levelData.getNeedRebuilding()) {

Platform.runLater

(LevelController.this:

:renderBackground);

levelData.setNeedRebuilding();

}

}

try {

Thread.sleep(20);

} catch (InterruptedException e) {

e.printStackTrace();

}

}

}

}

package levelMenu;

import game.Constants;

import game.LevelData;

import game.LevelGenerator;

import javafx.fxml.FXML;

import javafx.fxml.FXMLLoader;

import javafx.scene.Parent;

import javafx.scene.Scene;

import javafx.scene.image.Image;

import javafx.scene.image.ImageView;

import javafx.stage.Modality;

import javafx.stage.Stage;

import javafx.stage.StageStyle;

import level.LevelController;

import utilities.StageUtils;

import java.io.IOException;

import java.net.URL;

public class LevelMenuController {

@FXML

ImageView level1;

@FXML

ImageView level2;

@FXML

ImageView level3;

@FXML

ImageView level4;

@FXML

ImageView level5;

@FXML

ImageView level6;

@FXML

ImageView level7;

@FXML

ImageView level8;

@FXML

ImageView level9;

@FXML

ImageView level10;

@FXML

ImageView pointer1;

@FXML

ImageView pointer2;

@FXML

ImageView pointer3;

@FXML

ImageView pointer4;

@FXML

ImageView pointer5;

@FXML

ImageView pointer6;

@FXML

ImageView pointer7;

@FXML

ImageView pointer8;

@FXML

ImageView pointer9;

@FXML

ImageView pointer10;

@FXML

ImageView backImg;

private Image lev1;

private Image lev2;

private Image lev3;

private Image lev4;

private Image lev5;

private Image lev6;

private Image lev7;

private Image lev8;

private Image lev9;

private Image lev10;

private Image back;

private Image selLev1;

private Image selLev2;

private Image selLev3;

private Image selLev4;

private Image selLev5;

private Image selLev6;

private Image selLev7;

private Image selLev8;

private Image selLev9;

private Image selLev10;

private Image selBack;

@FXML

public void level1Pressed() {

initLevelController(1);

}

@FXML

public void level2Pressed() {

initLevelController(2);

}

@FXML

public void level3Pressed() {

initLevelController(3);

}

@FXML

public void level4Pressed() {

initLevelController(4);

}

@FXML

public void level5Pressed() {

initLevelController(5);

}

@FXML

public void level6Pressed() {

initLevelController(6);

}

@FXML

public void level7Pressed() {

initLevelController(7);

}

@FXML

public void level8Pressed() {

initLevelController(8);

}

@FXML

public void level9Pressed() {

initLevelController(9);

}

@FXML

public void level10Pressed() {

initLevelController(10);

}

private void initLevelController(final int levelNumber) {

try {

final Class<?> currentClass = getClass();

final URL fxmlUrl = currentClass.getResource

("../level/level.fxml");

FXMLLoader fxmlLoader = new FXMLLoader(fxmlUrl);

fxmlLoader.setController(new LevelController());

Parent root = (Parent) fxmlLoader.load();

LevelController controller = fxmlLoader.

getController();

final LevelGenerator generator = new

LevelGenerator(levelNumber);

final LevelData levelData =

generator.generateLevelData();

controller.setLevelData(levelData);

Stage stage = new Stage();

stage.setTitle("LEVEL " + levelNumber);

final double paneWidth = levelData.

getPaneWidth();

final double paneHeight =

levelData.getPaneHeight();

stage.setScene(new Scene(root, paneWidth,

paneHeight));

stage.initStyle(StageStyle.UTILITY);

stage.initModality(Modality.APPLICATION\_MODAL);

stage.showAndWait();

StageUtils.closeStage(pointer1);

} catch (IOException e) {

e.printStackTrace();

}

}

@FXML

public void initialize() {

lev1 = new Image(Constants.LEV1\_IMAGE);

lev2 = new Image(Constants.LEV2\_IMAGE);

lev3 = new Image(Constants.LEV3\_IMAGE);

lev4 = new Image(Constants.LEV4\_IMAGE);

lev5 = new Image(Constants.LEV5\_IMAGE);

lev6 = new Image(Constants.LEV6\_IMAGE);

lev7 = new Image(Constants.LEV7\_IMAGE);

lev8 = new Image(Constants.LEV8\_IMAGE);

lev9 = new Image(Constants.LEV9\_IMAGE);

lev10 = new Image(Constants.LEV10\_IMAGE);

back = new Image(Constants.BACK\_TO\_LEVEL\_MENU);

selLev1 = new Image(Constants.SEL\_LEV1\_IMAGE);

selLev2 = new Image(Constants.SEL\_LEV2\_IMAGE);

selLev3 = new Image(Constants.SEL\_LEV3\_IMAGE);

selLev4 = new Image(Constants.SEL\_LEV4\_IMAGE);

selLev5 = new Image(Constants.SEL\_LEV5\_IMAGE);

selLev6 = new Image(Constants.SEL\_LEV6\_IMAGE);

selLev7 = new Image(Constants.SEL\_LEV7\_IMAGE);

selLev8 = new Image(Constants.SEL\_LEV8\_IMAGE);

selLev9 = new Image(Constants.SEL\_LEV9\_IMAGE);

selLev10 = new Image(Constants.SEL\_LEV10\_IMAGE);

selBack = new Image

(Constants.SEL\_BACK\_TO\_LEVEL\_MENU);

}

@FXML

private void onBackToLevelMenu() throws IOException {

FXMLLoader loader = new FXMLLoader(getClass().

getResource("../complexityMenu/

complexityMenuScene.fxml"));

Parent root = (Parent) loader.load();

Stage stage = new Stage();

stage.setTitle("BOMBERMEN");

stage.setScene(new Scene(root, 620, 480));

stage.initStyle(StageStyle.TRANSPARENT);

stage.show();

StageUtils.closeStage(pointer1);

}

private void setUnselected() {

level1.setImage(lev1);

level2.setImage(lev2);

level3.setImage(lev3);

level4.setImage(lev4);

level5.setImage(lev5);

level6.setImage(lev6);

level7.setImage(lev7);

level8.setImage(lev8);

level9.setImage(lev9);

level10.setImage(lev10);

backImg.setImage(back);

pointer1.setVisible(false);

pointer2.setVisible(false);

pointer3.setVisible(false);

pointer4.setVisible(false);

pointer5.setVisible(false);

pointer6.setVisible(false);

pointer7.setVisible(false);

pointer8.setVisible(false);

pointer9.setVisible(false);

pointer10.setVisible(false);

}

@FXML

public void onLevel1() {

setUnselected();

level1.setImage(selLev1);

pointer1.setVisible(true);

}

@FXML

public void onLevel2() {

setUnselected();

level2.setImage(selLev2);

pointer2.setVisible(true);

}

@FXML

public void onLevel3() {

setUnselected();

level3.setImage(selLev3);

pointer3.setVisible(true);

}

@FXML

public void onLevel4() {

setUnselected();

level4.setImage(selLev4);

pointer4.setVisible(true);

}

@FXML

public void onLevel5() {

setUnselected();

level5.setImage(selLev5);

pointer5.setVisible(true);

}

@FXML

public void onLevel6() {

setUnselected();

level6.setImage(selLev6);

pointer6.setVisible(true);

}

@FXML

public void onLevel7() {

setUnselected();

level7.setImage(selLev7);

pointer7.setVisible(true);

}

@FXML

public void onLevel8() {

setUnselected();

level8.setImage(selLev8);

pointer8.setVisible(true);

}

@FXML

public void onLevel9() {

setUnselected();

level9.setImage(selLev9);

pointer9.setVisible(true);

}

@FXML

public void onLevel10() {

setUnselected();

level10.setImage(selLev10);

pointer10.setVisible(true);

}

@FXML

public void onBackEntered(){

setUnselected();

backImg.setImage(selBack);

}

}

package mainMenu;

import game.Constants;

import javafx.fxml.FXML;

import javafx.fxml.FXMLLoader;

import javafx.scene.Parent;

import javafx.scene.Scene;

import javafx.scene.image.Image;

import javafx.scene.image.ImageView;

import javafx.stage.Stage;

import javafx.stage.StageStyle;

import utilities.StageUtils;

import java.io.IOException;

public class MainMenuController {

@FXML

ImageView pointer1;

@FXML

ImageView pointer2;

@FXML

ImageView pointer3;

@FXML

ImageView newGame;

@FXML

ImageView information;

@FXML

ImageView exit;

private Image start;

private Image selStart;

private Image infImg;

private Image selInfImg;

private Image exitImg;

private Image selExitImg;

@FXML

public void initialize() {

start = new Image(Constants.START\_IMAGE);

selStart = new Image

(Constants.SELECTED\_START\_IMAGE);

infImg = new Image(Constants.INF\_IMAGE);

selInfImg = new Image(Constants.SELECTED\_INF\_IMAGE);

exitImg = new Image(Constants.EXIT\_IMAGE);

selExitImg = new Image

(Constants.SELECTED\_EXIT\_IMAGE);

}

@FXML

public void onNewGamePressed() throws IOException {

FXMLLoader loader = new FXMLLoader(getClass().

getResource("../complexityMenu/

complexityMenuScene.fxml"));

Parent root = (Parent) loader.load();

Stage stage = new Stage();

stage.setTitle("BOMBERMEN");

stage.setScene(new Scene(root, 620, 480));

stage.initStyle(StageStyle.TRANSPARENT);

stage.show();

StageUtils.closeStage(pointer1);

}

@FXML

public void onInformationPressed() throws IOException {

FXMLLoader loader = new

FXMLLoader(getClass().

getResource

("../infScene/informationScene.fxml"));

Parent root = (Parent) loader.load();

Stage stage = new Stage();

stage.setTitle("BOMBERMEN");

stage.setScene(new Scene(root, 620, 480));

stage.initStyle(StageStyle.TRANSPARENT);

stage.show();

StageUtils.closeStage(pointer1);

}

@FXML

public void onExitPressed() {

StageUtils.closeStage(newGame);

}

@FXML

public void onStartEntered() {

pointer2.setVisible(false);

information.setImage(infImg);

pointer3.setVisible(false);

exit.setImage(exitImg);

pointer1.setVisible(true);

newGame.setImage(selStart);

}

@FXML

public void onInformationEntered() {

pointer1.setVisible(false);

newGame.setImage(start);

pointer3.setVisible(false);

exit.setImage(exitImg);

pointer2.setVisible(true);

information.setImage(selInfImg);

}

@FXML

public void onExitEntered() {

pointer1.setVisible(false);

newGame.setImage(start);

pointer2.setVisible(false);

information.setImage(infImg);

pointer3.setVisible(true);

exit.setImage(selExitImg);

}

}

package underConstruction;

import game.Constants;

import javafx.fxml.FXML;

import javafx.fxml.FXMLLoader;

import javafx.scene.Parent;

import javafx.scene.Scene;

import javafx.scene.image.Image;

import javafx.scene.image.ImageView;

import javafx.stage.Stage;

import javafx.stage.StageStyle;

import utilities.StageUtils;

import java.io.IOException;

public class underConstructionController {

@FXML

ImageView backImg;

private Image back;

private Image selBack;

@FXML

public void initialize() {

back = new Image(Constants.BACK\_TO\_LEVEL\_MENU);

selBack = new Image

(Constants.SEL\_BACK\_TO\_LEVEL\_MENU);

}

@FXML

private void onBackToLevelMenu() throws IOException {

FXMLLoader loader = new FXMLLoader(getClass().

getResource("../complexityMenu/

complexityMenuScene.fxml"));

Parent root = (Parent) loader.load();

Stage stage = new Stage();

stage.setTitle("BOMBERMEN");

stage.setScene(new Scene(root, 620, 480));

stage.initStyle(StageStyle.TRANSPARENT);

stage.show();

StageUtils.closeStage(backImg);

}

@FXML

public void onBackEntered(){

backImg.setImage(selBack);

}

@FXML

public void onBackExited(){

backImg.setImage(back);

}

}

package utilities;

public interface Executable {

void execute();

}

Класс

package utilities;

import javafx.scene.Node;

import javafx.stage.Stage;

public class StageUtils {

private StageUtils() {

}

public static void closeStage(final Node view){

final Stage stage = (Stage)view.getScene().

getWindow();

stage.close();

}

}