Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

Рязанский государственный радиотехнический университет имени В.Ф. Уткина

Кафедра САПР ВС

К защите

Руководитель работы:

_____

дата, подпись

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

К КУРСОВОЙ РАБОТЕ

по дисциплине

«Компьютерная графика»

Тема:

«Изучение фреймворка создания игр libGDX на примере игры Star Battle»

Выполнил студент группы 045

Вашкулатов Н.А.

_____

дата сдачи на проверку,
подпись

Руководитель работы

доцент кафедры САПР ВС
Митрошин А.А.

_____ _____

оценка дата защиты, подпись

Рязань 2022

ЗАДАНИЕ

на курсовую работу по дисциплине

«Компьютерная графика»

Тема: «Изучение библиотеки создания игр libGDX на примере игры Морской Бой».

Задание: Изучить принцип работы и главные особенности фреймворка для создания игр libGDX на примере создания игры Star Battle.

Содержание

Введение

С развитием цифровых технологий компьютеры все больше вливаются в жизнь человека. Если раньше ЭВМ использовались исключительно для сложных математических вычислений, то сегодня сфера их применения существенно расширилась. Компьютерные игры - одно из наиболее массовых применений электронных вычислительных машин.

Развитие игровой индустрии шло стремительным темпом, и особенно пользовалось популярностью у подростков. Первые игры отличались простотой интерфейса и логики, но со временем они становились все сложнее и сложнее, над их созданием работал уже не один человек, а целая команда разработчиков.

Современные игры требуют достаточно большой производительности от компьютера, и не каждая офисная машина в силах воспроизводить их. Однако для отдыха от монотонной работы зачастую достаточно простой, не требовательной к технике, игры. Именно такой разработке посвящен данный курсовой проект - игра «*Star Battle*».

Star Battle – аналог Морского боя со всеми его принципами и правилами, но в тематике вселенной Звездных Войн.

Для упрощения разработки игры мною был выбран LibGDX — фреймворк для создания игр и приложений, написанный на Java с использованием C и C++ и позволяющий писать кроссплатформенные игры и приложения, используя один код.

1 Теоретическая часть

1.1 Фреймворк libGDX

LibGDX — фреймворк для создания игр и приложений, написанный на Java с использованием C и C++ (для более быстрой работы) и позволяющий писать кроссплатформенные игры и приложения, используя один код.

Основной особенностью фреймворка является кроссплатформенность. Любое приложение, написанное на этом фреймворке можно запустить на Windows, Linux, Mac, Android, WebGL. Это достигается за счет отделения логики приложения, от процедуры отрисовки при помощи прокси (в libGDX называются backends), которые оборачивают библиотеку для определенной платформы (lgjwl, Android API).

LibGDX позволяет перейти на такой низкий уровень, какой требуется в той или иной ситуации, давая примой доступ к файловой системе, устройствам ввода, аудио устройствам и OpenGL через единый OpenGL ES 2.0 и 3.0 интерфейс.

Наверху таких низкоуровневых возможностей создан мощный набор API, который позволяет решать общие в разработке игр задачи, такие как визуализация спрайтов, теста, построение пользовательских интерфейсов, проигрывание звуковых эффектов и музыки, линейная алгебра и тригонометрические вычисления, разбор JSON и XML, и так далее.

По своей сути libGDX состоит из шести модулей в виде интерфейсов, которые предоставляют средства для взаимодействия с операционной системой. Каждый прокси реализует эти интерфейсы:

Application: запускает приложение и информирует клиента API о событиях уровня приложения, таких как изменение размера окна. Предоставляет средства ведения журнала и методы запроса, например, использование памяти.

Files: предоставляет базовую файловую систему платформы. Обеспечивает абстракцию над различными типами расположения файлов

поверх пользовательской системы обработки файлов (которая не взаимодействует с классом файлов Java).

Input: информирует клиента API о вводимых пользователем данных, таких как события мыши, клавиатуры, касания или акселерометра. Поддерживаются как опрос, так и обработка, управляемая событиями.

Net: предоставляет средства для доступа к ресурсам через HTTP/HTTPS, а также для создания TCP-серверных и клиентских сокетов.

Audio: предоставляет средства для воспроизведения звуковых эффектов и потоковой передачи музыки, а также прямого доступа к аудиоустройствам для ввода / вывода звука PCM.

Graphics: предоставляет OpenGL ES 2.0 (там, где это доступно) и позволяет запрашивать / устанавливать видеорежимы и подобные вещи.

Приложение libGDX имеет четко определенный жизненный цикл, управляющий состояниями приложения, такими как создание, приостановка и возобновление, рендеринг и завершение работы приложения.

Разработчик приложения подключается к этим событиям жизненного цикла, реализуя интерфейс ApplicationListener и передавая экземпляр этой реализации прикладной реализации конкретного серверного модуля (прокси). С этого момента приложение будет вызывать ApplicationListener каждый раз, когда происходит событие уровня приложения.

При разработке игры были использованы классы пакета scene2D

1.1.1 Архитектура игры на основе scene2D

Пакет **scene2D** в libGDX представляет собой классы для реализации двухмерной сцены, которые могут быть полезны для управления группой иерархически связанных актеров (сущностей). Ниже будут рассмотрены преимущества данного пакета.

Во-первых, с помощью scene2D осуществляется вращение и масштабирование группы объектов. Дочерние объекты работают в своей системе координат, так что родительские преобразования для них прозрачны.

Во-вторых, данный пакет упрощает 2D-рендеринг изображения. Каждый объект существует в своей не вращаемой и не скалируемой системе координат, где (0,0) - нижний левый угол объекта.

В-третьих, scene2D имеет очень гибкую систему событий, которая позволяет обрабатывать события родителя перед или после обработки событий дочерних объектов.

Но у данного пакета также есть и недостаток, который заключается в том, что объекты совмещают и модель, и представление. Такое сцепление логики не позволяет полноценно использовать MVC (Model-View-Controller). Но в некоторых случаях это может быть полезно и удобно.

Главной особенностью scene2D является то, что у него есть три центровых класса: Stage, Group и Actor.

Класс **Stage** (сцена) имеет «корневую» группу (т.е. класс Group), куда приложение может добавлять своих актеров. У класса Stage есть своя камера и упаковщик спрайтов (SpriteBatch). Stage отсылает события дочерним элементам и реализует интерфейс InputProcessor, который предназначен для считывания событий ввода.

Класс **Group** является актером, который может содержать в себе других актеров. Вращение и масштабирование группы соответствующим образом отражается на его дочерних актерах. Класс Group делегирует рисование и события ввода соответствующим дочерним актерам. Он превращает координаты событий ввода так, что дочерние элементы получают эти координаты в своей собственной системе координат.

Класс **Actor** представляет собой некоторая сущность, которая может быть нарисована и которая может обрабатывать события ввода. Актер имеет позицию, размер, информацию о масштабе, вращение и родительский

компонент (origin). Также актер может иметь имя, что позволяет найти его по этому имени - это может быть полезно при отладке.

Чтобы использовать абстрактные классы Stage, Group, Actor нужно их расширить и переопределить методы для рисования, определение нажатий и т.д.

1.1.2 Работа с графическими компонентами

При написании игр больше всего времени тратится именно на работу с графикой. В данном параграфе будет рассмотрено, как правильно работать с графикой (текстурами, регионами, атласами и т.д.) в libGDX.

Текстура – двумерное изображение, хранящееся в памяти компьютера или графического акселератора в одном из пиксельных форматов. В случае хранения в сжатом виде на дисках компьютера текстура может представлять собой битмап (от англ. binary map – двоичная карта), который можно встретить в форматах bmp, jpg, gif и т.д.

Спрайт – растровое изображение, свободно перемещающееся по экрану. По своей сути, спрайт – это проекция объекта на плоскости. В 2D-графике это реализуется без проблем, но при работе с 3D-графикой при изменении угла обзора происходит разрушение иллюзии. Если представить себе стену в игре, то, когда камера смотрит на неё прямо – всё в порядке, но, если посмотреть на неё сбоку – камера увидит лишь линию.

Текстурный атлас – большое изображение, которое содержит много изображений меньшего размера, каждое из которых является текстурой для какой-то части объекта. Такие «подтекстуры» можно накладывать путём указания текстурных координат на атласе, то есть, выбирая только определённую часть изображения. В приложениях, в которых часто используются мелкие текстуры, более эффективно хранить текстуры в текстурном атласе, так как это позволяет сократить количество смен состояний до одного для всего атласа, уменьшает количество занятых текстурных слотов, а также минимизируется фрагментация видеопамяти.

Чаще всего изображения, которые содержатся в атласе, имеют одинаковый размер, поэтому рассчитать их координаты не составит труда.

С помощью класса **SpriteBatch** в libGDX можно легко работать с графическими компонентами. Также данный класс помогает оптимизировать работу с графикой, поскольку привязка текстуры является довольно дорогостоящей операцией, он хранит много маленьких изображений в атласе, а затем рисует регионы из атласа для предотвращения частой смены текстур.

2 Создание проекта с подключенным libGDX

Для создания проекта с подключенными зависимостями libGDX в официальной документации предлагается утилита gdx-setup при помощи которой можно создать рабочий проект для выбранных платформ и с выбранными расширениями. Получим проект с сборщиком Gradle такой структуры (рисунок 1)

```
settings.gradle           <- definition of sub-modules. By default core, desktop, android, html, ios
build.gradle              <- main Gradle build file, defines dependencies and plugins
gradlew                   <- local Gradle wrapper
gradlew.bat               <- script that will run Gradle on Windows
gradle                    <- script that will run Gradle on Unix systems
local.properties          <- IntelliJ only file, defines Android SDK location

assets/                   <- contains your graphics, audio, etc.

core/
    build.gradle          <- Gradle build file for core project
    src/                  <- Source folder for all your game's code

desktop/
    build.gradle          <- Gradle build file for desktop project
    src/                  <- Source folder for your desktop project, contains LWJGL launcher class

android/
    build.gradle          <- Gradle build file for android project
    AndroidManifest.xml   <- Android specific config
    res/                  <- contains icons for your app and other resources
    src/                  <- Source folder for your Android project, contains android launcher class

html/
    build.gradle          <- Gradle build file for the html project
    src/                  <- Source folder for your html project, contains launcher and html definition
    webapp/               <- War template, on generation the contents are copied to war. Contains startup url index page and web.xml

ios/
    build.gradle          <- Gradle build file for the iOS project
    src/                  <- Source folder for your iOS project, contains launcher
```

Рисунок 1 – Стандартная структура проекта

При разработке проект был настроен вручную с сборщиком Maven, который загрузил необходимые зависимости. Мною был выбран Maven поскольку его структура проще, сборка проекта в jar выполняется быстрее и все работает стабильнее.

Были подключены и загружены следующие зависимости:

com.badlogicgames.gdx – классы для разработки, одинаковые для обоих платформ.

com.badlogic.gdx.backends.lwjgl3 – прокси для Windows, Linux, Mac.

com.badlogicgames.gdx.gdx-freetype – классы для генерации BitMapFont из otf.

Все зависимости были загружены с репозитория Maven Central.

3 Разработка ПО

3.1 Основные классы

3.1.1 Структура проекта и описание классов

Конечная структура пакетов проекта показана на рисунке 2.

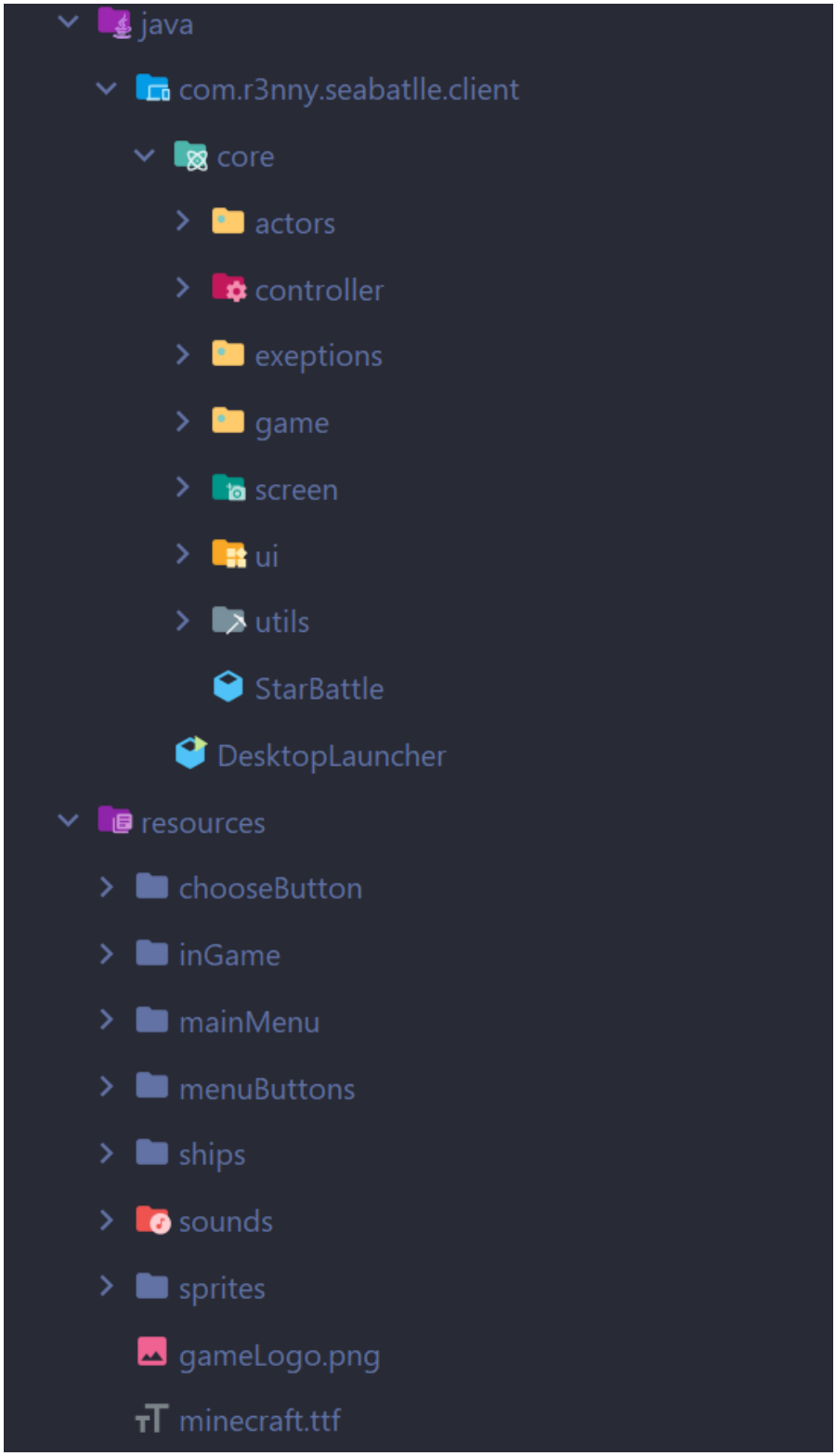

Рисунок 2 – Конечная структура пакетов проекта

1. Класс ***com.r3nny.seabattle.client.DesktopLauncher.java***

Этот класс является стартовым классом. В методе main создается конфигурация приложения и само приложение на основе прокси *com.badlogic.gdx.backends.lwjgl3.Lwjgl3Application.* Конструктор приложения принимает в себя объект, реализующий интерфейс *ApplicationListener* и конфигурацию. Все события жизненного цикла будут обрабатываться этим объектом.

2. Класс ***com.r3nny.seabattle.client.core.StarBattle.java***

Этот класс является основным. Он расширяет абстрактный класс com.badlogic.gdx.*Game.java*, который реализует интерфейс *ApplicationListener*.

В методе create (вызываемый при создании приложения) сразу меняется экран на SplashScreen.

В этом классе содержатся статические поля менеджера звука, анимации и ресурсов для удобного доступа к ним.

Метод setUpStage() возвращает настроенную сцену для использования на экранах.

3. Пакет m***com.r3nny.seabattle.client.core.actors*** содержит классы, расширяющие классы com.badlogic.gdx.scenes.scene2d.Actor или com.badlogic.gdx.scenes.scene2d.Group. Экземпляры этих классов отображаются на сцене Stage.

3.1. Класс ***com.r3nny.seabattle.client.core.actors.Cell.java***

Данный класс является клеткой поля. Содержит информацию о своем положении (row, column), ссылку на корабль в этой клетке, а также статус клетки (море, промах, целый корабль, ранен, убит).

3.2. Класс ***com.r3nny.seabattle.client.core.actors.Ship.java*** содержит информацию о своем типе (количестве палуб), ссылку на клетки, в которых он расположен, ориентацию, расположение.

3.3. Класс ***com.r3nny.seabattle.client.core.actors.Ship.ShipInputListener.java*** является внутренним классом класса Ship и обрабатывает

взаимодействие с кораблем. При помощи мыши можно перетащить корабль на игровое поле, повернуть его, обвести при наведении.

3.4. Класс ***com.r3nny.seabattle.client.core.actors.ShipCountDTO.java*** является вспомогательным и отражает количество кораблей определенного типа. Содержит тип корабля и количество.

3.5. Класс ***com.r3nny.seabattle.client.core.actors.ShipManager.java*** отображает количество оставшихся кораблей у определенного поля. Содержит список из *ShipCountDTO.*

3.6. Класс ***com.r3nny.seabattle.client.core.actors.ShipCreatingArea.java*** отображает подсказки по созданию кораблей и поле, на котором изначально расположены корабли.

3.7. Класс ***com.r3nny.seabattle.client.core.actors.GameField.java*** абстрактный класс игрового поля. Содержит массив клеток, отражающих поле, список кораблей на поле, ShipManager. При создании поля создаются клетки игрового поля и корабли на начальных позициях. Через объект поля можно автоматически заполнить его кораблями, убить корабль и отобразить это в ShipManager.

3.8. Класс ***com.r3nny.seabattle.client.core.actors.EnemyGameField.java*** расширяет класс *GameField.java*. При создании экземпляра сразу автоматически расставляются корабли.

3.9. Класс ***com.r3nny.seabattle.client.core.actors.PlayerGameField.java*** расширяет класс *GameField.java.* При создании экземпляра создается ShipsCreatingArea и на основе ее расставляются корабли на начальных позициях готовые для ручной расстановки.

4. Пакет ***com.r3nny.seabattle.client.core.controller*** содержат вспомогательные для логики игры классы.

4.1 Класс ***com.r3nny.seabattle.client.core.controller.ShipsCreator.java*** содержит массив с типами кораблей, необходимых для расстановки и два открытых статических метода: autoCreateShips, который автоматически

расставляет корабли на необходимом поле и addShipToGameField, который добавляет на поле корабль, начиная с определенной клетки. Для этого в методе пытаемся получить клетки, которые должен занимать корабль. Если корабль вылезет за пределы карты или клетки уже заняты, то выбрасывается исключение CantCreateShipException. Если получается получить все клетки, то вокруг корабля помечаем клетки как занятые и связываем корабль и полученные клетки. Код прилагается в приложении А.

4.2 Класс ***com.r3nny.seabattle.client.core.controller.ShootController.java*** содержит открытый метод shoot, который изменяет состояние клетки на поле в зависимости от того, куда был сделан выстрел и кем он был сделан.

4.3 Класс ***com.r3nny.seabattle.client.core.controller.Bot.java*** объект данного класса производит стрельбу по полю игрока через определенно заданный промежуток. Если бот промахивается, то он продолжает работу в обычном режиме. Если же бот попадает, он запоминает клетку, в которой находится поврежденный корабль и выбирает случайное направление стрельбы. Через промежуток времени бот снова начинает свою работу. Если направление установлено, проверяется может ли бот стрелять в том направлении (проверка на границу карты и на то не стреляли ли в эту клетку раннее). Если выстрел разрешен, то он производится. Если бот промазал он меняет свое направление по принципу, который будет описан чуть позже. В случае попадания бот записывает эту клетку в список клеток корабля, а если этот выстрел убил корабль, то бот сбрасывает свои настройки и работает в обычном режиме. Если же выстрел в направлении запрещен, то бот меняет направление следующему принципу. Если направление уже менялось, то бот меняет направление на 90 градусов и сбрасывает флаг, указывающий на то менялось ли оно. Если же мы первый раз меняем направление, то меняем его на противоположное и указываем, что направление было изменено.

Код прилагается в приложении Б.

5. Класс ***com.r3nny.seabattle.client.core.exceptions .CantCreateShipException.java*** – это исключение, которое выбрасывается при попытке создать корабль вне поля или на занятых клетках.

6. Пакет ***com.r3nny.seabattle.client.core.game*** содержит классы, связанные с логикой проведения игры.

   6.1 Класс ***com.r3nny.seabattle.client.core.game.Game.java*** **–** абстрактный класс, который содержит два игровых поля и статус игры (Меню, Создание Корабля, Расстановка кораблей, Ход игрока, Ход противника, Конец).

   6.2 Класс ***com.r3nny.seabattle.client.core.game.SingleGame.java*** **–** класс, расширяющий Game.java, содержит в себе ссылку на Bot.java, который вызывается в процессе игры. В конструкторе инициализируются игровые поля.

7. Пакет ***com.r3nny.seabattle.client.core.screen*** содержат классы, реализующие интерфейс *com.badlogic.gdx.Screen.* Они являются как бы заготовкой отображения. Содержат методы show(), render(), hide(), resize(), dispose(), hide().

   7.1 Класс ***com.r3nny.seabattle.client.core.screen.Screen.java*** **-** абстрактный класс, реализующий интерфейс *com.badlogic.gdx.Screen.* Содержит ссылку на ApplicationListener, Stage и protected методы для доступа к менеджерам Assets, SoundManager и AnimationManager.

   7.2 Класс ***com.r3nny.seabattle.client.core.screen.InGameScreen.java*** **–** абстрактный класс Screen.java. Содержит настроенную кнопку возвращения в меню, фон экрана и лого игры.

   7.3 Класс ***com.r3nny.seabattle.client.core.screen.SplashScreen.java*** – загрузочный экран, содержащий лого игры. Параллельно отображению лого игры загружаются все ресурсы из класса Asset.java.

7.4 Класс ***com.r3nny.seabattle.client.core.screen.MenuScreen.java*** – содержит кнопку для смены экрана на ChooseScreen.java и кнопку выхода из игры.

7.5 Класс ***com.r3nny.seabattle.client.core.screen.ChooseScreen.java*** – содержит кнопки выбора режима игры (одиночный и сетевой). Активна только кнопка одиночного режима.

7.6 Класс ***com.r3nny.seabattle.client.core.screen.ShipCreatingScreen.java*** содержит ShipsCreatingArea, поле игрока с кораблями. На данном поле создается экземпляр SingleGame.java и происходит расстановка кораблей игрока.

7.7 Класс ***com.r3nny.seabattle.client.core.screen.SingleGameScreen.java*** содержит поля игрока и противника, два ShipManager, SingleGame. В методе render() этого класса вызывается логика игры бота.

7.8 Класс ***com.r3nny.seabattle.client.core.screen.EndScreen.java*** в конструктор принимает флаг выиграл ли игрок и в зависимости от этого воспроизводит соответствующую музыку и выводит соответствующий текст.

8. Класс ***com.r3nny.seabattle.client.core.ui.ChangeScreenButton.java*** является оберткой класса com.badlogic.gdx.scenes.scene2d.ui.TextButton. Принимает в себя текст кнопки и два Runnable. Один вызывается сразу после нажатия кнопки и вызывается анимация пропадания кнопки. Второй вызывается после исчезновения кнопки.

9. Пакет ***com.r3nny.seabattle.client.core.utils*** содержит вспомогательные классы не связанные с логикой.

9.1 Класс ***com.r3nny.seabattle.client.core.utils.Assets.java*** **–** обертка над com.badlogic.gdx.assets.AssetsManager. При создании в загрузочном экране Asset загружает все ресурсы и предоставляет методы доступа к ним.

9.2 Класс ***com.r3nny.seabattle.client.core.utils.SoundManager.java*** – загружает все необходимые аудиофайлы, настраивает их громкость и предоставляет методы для запуска определенных звуков и их остановки.
9.3 Класс ***com.r3nny.seabattle.client.core.utils.AnimationManager.java*** – загружает все спрайты, преобразует их в удобный класс Animation для отрисовки в Actor, так же содержит готовые Actions для появления, исчезновения и передвижения.

3.1.2 Диаграмма классов

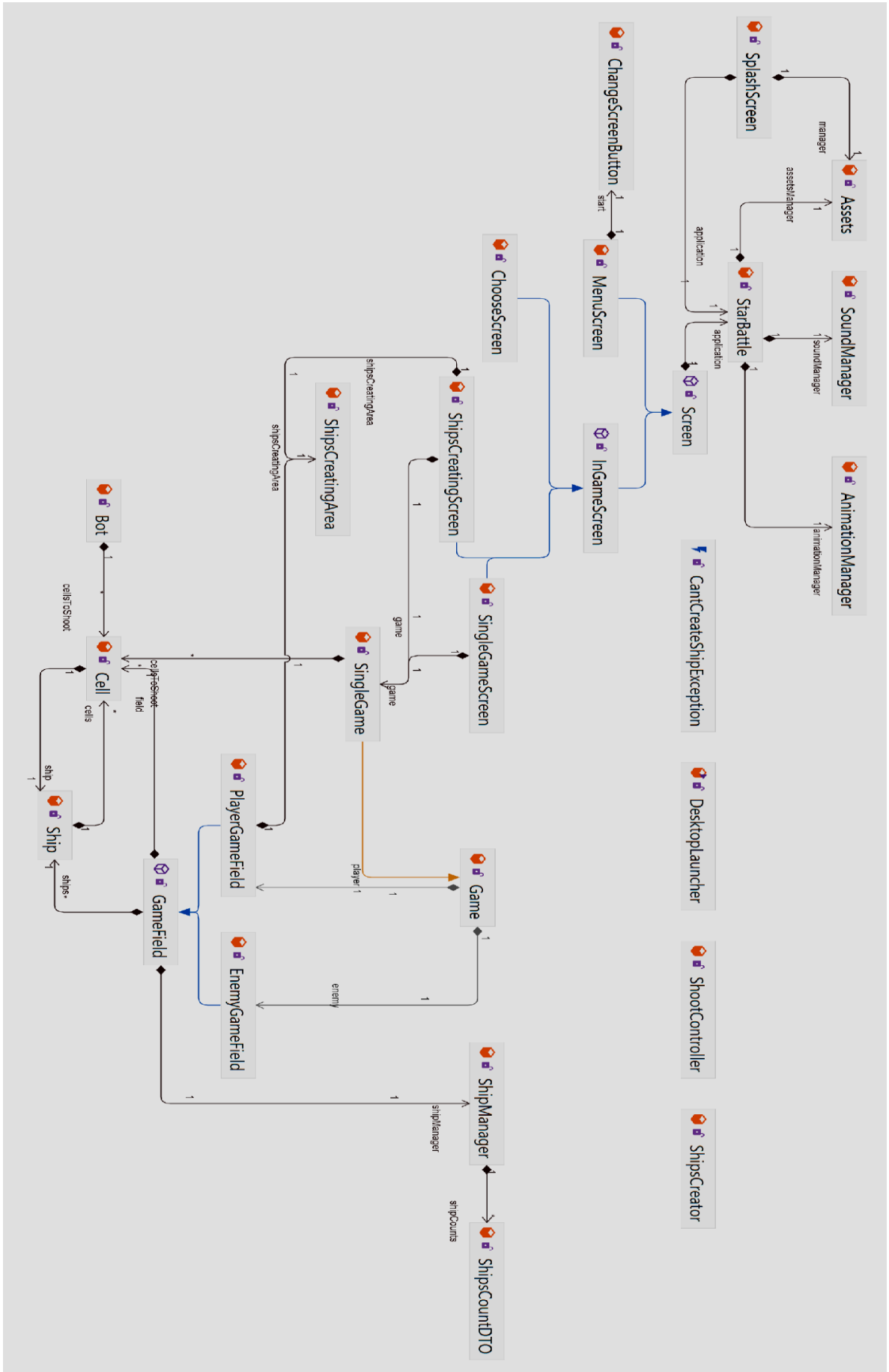

Рисунок 3 – Диаграмма классов

3.2 Таблица основных методов и переменных

Таблица 1 – Основные методы и переменные

Класс	Метод или переменная	Назначение
Cell	int column, int row	Позиция клетки на поле
	cellStatus status	Статус клетки
	Ship ship	Ссылка на корабль
	float animationTime	Время с начала анимации
Ship	ShipType type	Тип корабля (размер)
	List<Cell> cells	Список клеток корабля
	boolean isVertical	Ориентация корабля
	boolean isSelected	Наведена ли мышка на корабль
	updateBounds()	Обновляет границы объекта. Нужно при передвижении
	makeCellsKilled()	Убивает все клетки
	rotate()	Переворачивает текстуру корабля
	canMoveShip()	Можно ли взаимодействовать мышкой с кораблем (корабль не расположен и не прибывает)
GameField	ShipManager shipManager	Учет количества кораблей и отображение этого параметра
	boolean isShipsReady	Расставлены ли корабли
	Cell[][] field	Поле игры
	List<Ship> ships	Корабли поля
	killShip(Ship ship)	Убивает корабль, вызывая у него метод makeCellsKilled, уменьшает количество кораблей этого типа у ShipManager, убирает корабль из группы
	clearAllMissed()	Заменяет все промахи на море
ShipInputListener	exit()	Вызывает unSelect() у Ship
	enter()	Вызывает select() у Ship
	touchdown()	При нажатии ПКМ вызывает rotate()
	touchDragged()	Передвигает корабль по экрану
	touchUp()	Если отпускается, когда корабль на клетке, выполняется попытка создать корабль и анимация появления

Продолжение таблицы 1

Класс	Метод или переменная	Назначение
ShipInputListener	Optional<Cell> getCellFrom(Event e)	Возвращает клетку, над которой был отпущен корабль
	animateShip()	Запускает анимацию появления корабля
PlayerGameField	ShipsCreatingArea area	Панель на которой расположены корабли в начале
	createShipsAutomaticaly	Автоматически заполняет поле кораблями и добавляет их в группу
ShipsCountDTO	ShipType type	Тип корабля
	int count	Оставшееся количество кораблей данного типа
ShipManager	Set<ShipsCountDTO> shipCounts	Сколько осталось кораблей каждого типа
	decByType(ShipType type)	Уменьшает количество кораблей определенного типа
	boolean isAllShipsKilled()	Уничтожены ли все корабли
ShipsCreator	ShipTypes[] shipTypes	Список типов кораблей, которые нужно создать
	int createdPlayerShips	Учет количества успешно созданных в ручную кораблей игрока
	Boolean isAnyShipsLanding	Приземляется ли какой-нибудь корабль в данный момент
	autocreateShips(GameField gf)	Заполняет поле кораблями случайны образом
	addShipToGameField(Cell cell, Ship ship, GameField gf)	Добавляет корабль на поле, начиная с заданной клетки, на заданном поле. Выбрасывает исключение CantCreateShipException
ShootController	shoot(int row, int column)	Изменяет состояние клетки на определенном поле в зависимости от того, чья очередь ходить и меняет ход, если нужно
Game	GameField player, enemy	Поля игрока и противника
	GameStatus status	Состояние игры
SingleGame	Bot bot	Объект, который будет автоматически стрелять по полю игрока.

Продолжение таблицы 1

Класс	Метод или переменная	Назначение
Screen	Stage stage	Сцена, на которой все отображается
	StarBattle application	Ссылка на ApplicationListener. Получаем доступ к нашим менеджерам и меняем сцену
InGameScreen	Image gameLogo	Лого игры
	Image bgImage	Фон сцены
StarBattle	Stage setUpStage()	Возвращает сцену с настроенными Viewport и SpriteBatch для отображения в нужной нам системе координат
Bot	boolean isDirectionChanged	Менялось ли направление на противоположное
	List<Cell> cellsToShoot	Список клеток, в которые может стрелять бот. По умолчанию содержит все клетки поля игрока. С каждым выстрелом их количество уменьшается
	Optional<Direction> direction	Текущее направление
	doRandomShot()	Производит случайный выстрел
	keepDestroingShip()	Продолжает стрелять опираясь на текущее направление и список клеток, в которые попали
	List<Cell> hittedCells	Клетки в которые попали
	changeDirection()	Изменяет направление исходя из текущего состояния бота

4 Результат работы

1) Меню игры (рисунок 4)

Рисунок 4 - Меню игры

2) Экран выбора режима игры (рисунок 5)

Рисунок 5 – Выбор режима

3) Экран расстановки кораблей (рисунок 6)

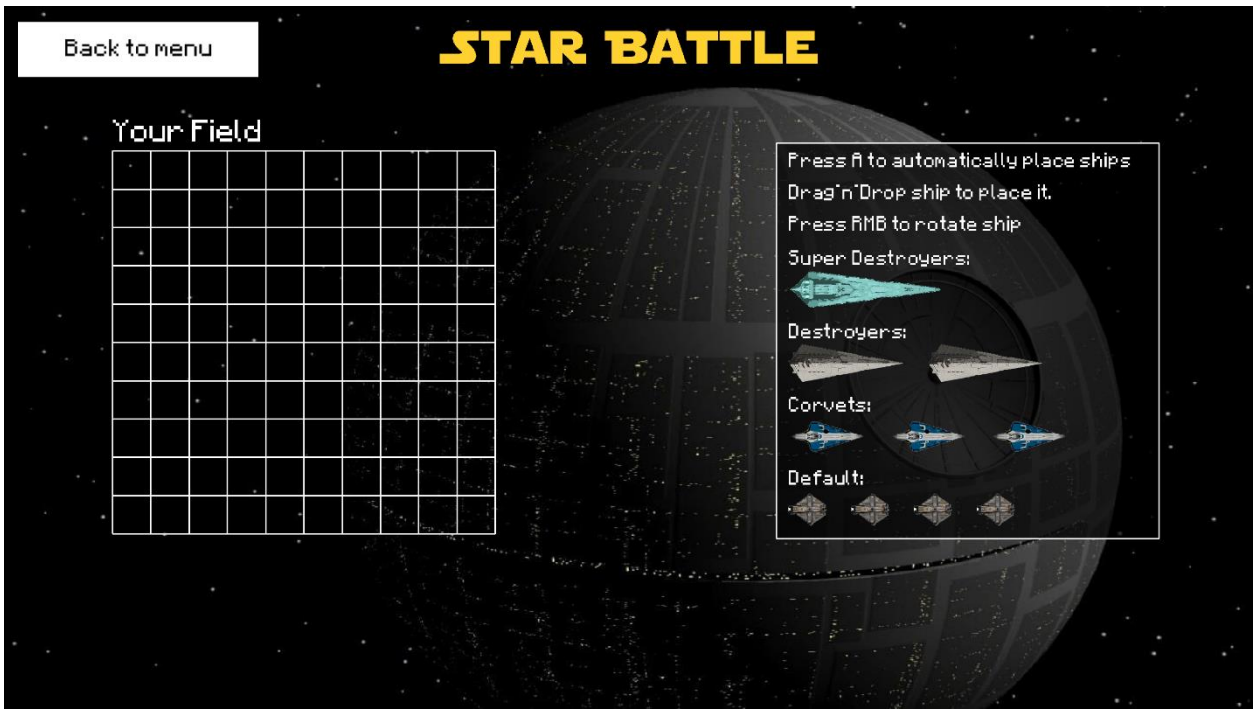

Рисунок 6 – Расстановка кораблей

4) Экран игры (рисунок 7)

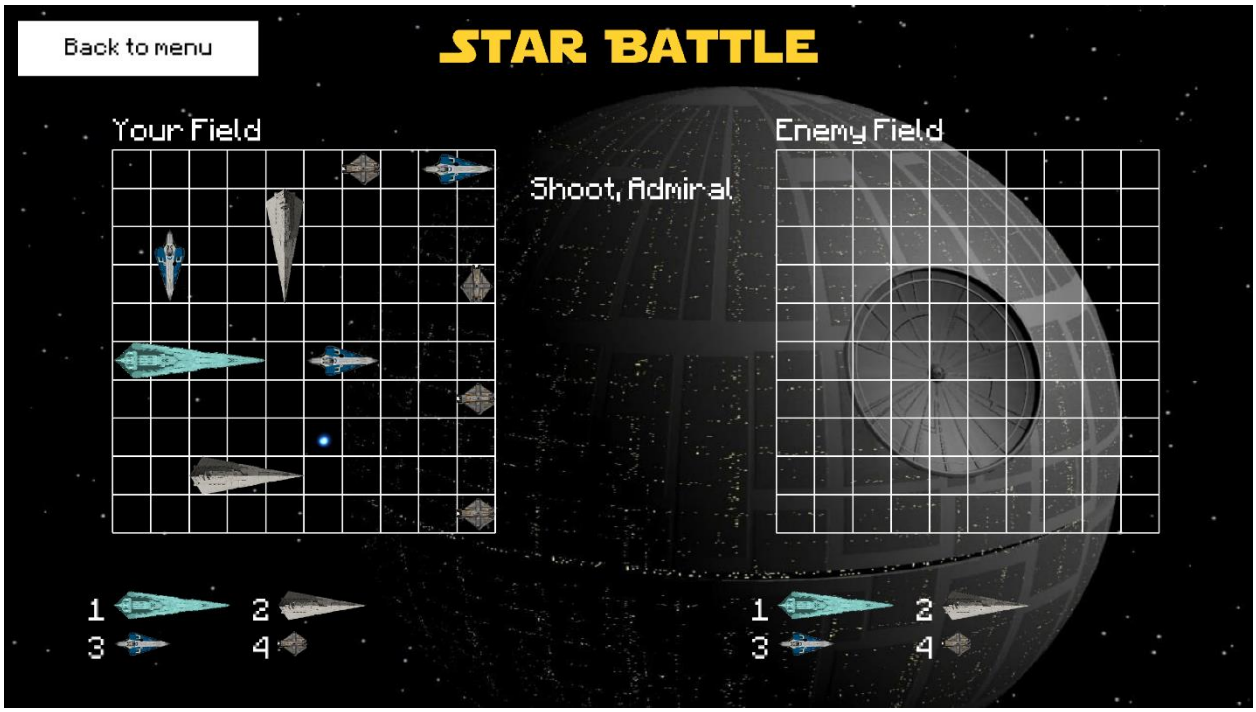

Рисунок 7 – Экран игры

5) Экран поражения (рисунок 8)

Рисунок 8 – Экран при поражении

6) Экран поражения (рисунок 9)

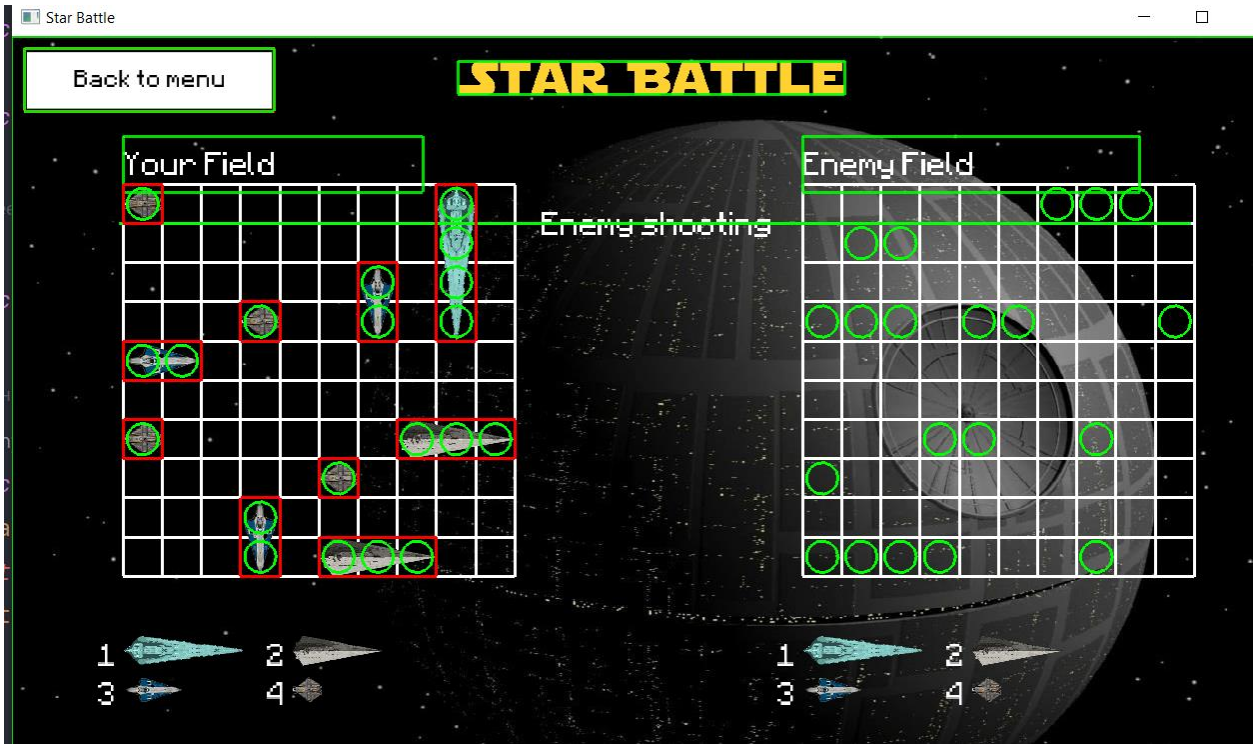

Рисунок 9 – Игра в режиме отладки

Заключение

В ходе курсовой работы были изучены основы фреймворка libGDX. С использованием этих знаний была создана игра «Морской бой» в стилистике Звездных Войн.

Список использованных источников:

1) libGDX Wiki: [Электронный ресурс]. URL: https://libgdx.com/wiki/ (Дата обращения: 20.11.2022).
2) Game Develompent Tutorials: [Электронный ресурс]. URL: https://gamefromscratch.com (Дата обращения: 15.12.2022).
3) Stackoverfow libGDX questions [Электронный ресурс]. URL: https://stackoverflow.com/tags/libgdx (Дата обращения: 27.11.2022).

Приложение А. Код класса com.r3nny.seabattle.client.core.controller.ShipsCreator.java

```
package com.r3nny.seabatlle.client.core.controller;

import com.badlogic.gdx.Gdx;
import com.r3nny.seabatlle.client.core.actors.Cell;
import com.r3nny.seabatlle.client.core.actors.GameField;
import com.r3nny.seabatlle.client.core.actors.Ship;
import com.r3nny.seabatlle.client.core.actors.ShipType;
import
com.r3nny.seabatlle.client.core.exeptions.CantCreateShipException;

import java.util.LinkedList;
import java.util.List;
import java.util.Random;

/**
 * Статические методы для создания кораблей
 */
public class ShipsCreator {
    private static Random rd;

    /**
     * Учет успешно созданных кораблей игрока
     */
    public static int createdPlayerShips = 0;

    public static boolean isAnyShipLanding = false;

    public static ShipType[] shipTypes = {
            ShipType.FOUR_DECK,
            ShipType.THREE_DECK,
            ShipType.THREE_DECK,
            ShipType.TWO_DECK,
            ShipType.TWO_DECK,
            ShipType.TWO_DECK,
            ShipType.ONE_DECK,
            ShipType.ONE_DECK,
            ShipType.ONE_DECK,
            ShipType.ONE_DECK
    };

    /**
     * Автоматически создает корабли на поле
     *
     * @param gf Игровое поле
     */
    public static void autoCreateShips(GameField gf) {
        rd = new Random();
        Cell[][] field = gf.getField();
        List<Ship> ships = gf.getShips();
        int createdShips = 0;
        while (createdShips < shipTypes.length) {
            Cell cell = getRandomCellFromField(field);
            Ship ship = ships.get(createdShips);
            if (rd.nextBoolean())
```

```
                ship.rotate();
            try {
                addShipToGameField(cell, ships.get(createdShips),
gf);
                createdShips++;
            } catch (CantCreateShipException ex) {
                System.err.println("Cant create ship on Cell" +
cell);
            }
        }
        Gdx.app.log("ShipsCreator", "All ships automatically
created");
    }

    private static Cell getRandomCellFromField(Cell[][] field) {
        int row = rd.nextInt(GameField.FIELD_SIZE);
        int column = rd.nextInt(GameField.FIELD_SIZE);
        return field[row][column];
    }


    /**
     * Помещает корабль на поле с началом на определенной клетке
     *
     * @param cell начало корабля
     * @param ship корабль
     * @param gf   поле
     * @throws CantCreateShipException если нельзя поместить корабль
     */
    public static void addShipToGameField(Cell cell, Ship ship,
GameField gf) throws CantCreateShipException {
        List<Cell> shipCells = getShipCells(cell, ship,
gf.getField());
        linkShipAndCells(ship, shipCells);
        setCellsAroundShipMissed(ship, gf.getField());
    }

    /**
     * @return Список клеток в которые добавляется корабль
     * @throws CantCreateShipException если нельзя поместить корабль
     */
    private static List<Cell> getShipCells(Cell startCell, Ship ship,
Cell[][] field)
            throws CantCreateShipException {
        List<Cell> cells = new LinkedList<>();
        for (int i = 0; i < ship.getType().getSize(); i++) {
            tryAddCellToList(startCell, ship, field, i, cells);
        }
        return cells;
    }

    private static void tryAddCellToList(
            Cell startCell, Ship ship, Cell[][] field, int i,
List<Cell> cells)
            throws CantCreateShipException {
        try {
            Cell currentCell = getCurrentCell(startCell, ship, field,
i);
            if (currentCell.isSea())
```

```
                cells.add(currentCell);
            else
                throw new CantCreateShipException("Cells not clear");
        } catch (IndexOutOfBoundsException e) {
            throw new CantCreateShipException("Ship cant fit on
map");
        }
    }

    private static Cell getCurrentCell(Cell startCell, Ship ship,
Cell[][] field, int i)
            throws IndexOutOfBoundsException {
        if (ship.isVertical())
            return field[startCell.getRow() -
i][startCell.getColumn()];
        else
            return field[startCell.getRow()][startCell.getColumn() +
i];
    }

    /**
     * Связывает корабль и клетки в двухстороннем порядке
     */
    private static void linkShipAndCells(Ship ship, List<Cell>
shipCells) {
        Cell startCell = shipCells.get(0);
        ship.setPosition(startCell.getX(), startCell.getY());
        ship.setCells(shipCells);
        setShipToCells(ship);
    }

    private static void setShipToCells(Ship ship) {
        for (Cell cell : ship.getCells()) {
            cell.setShip(ship);
            cell.setHealthy();
        }
    }

    public static void setCellsAroundShipMissed(Ship ship, Cell[][]
field) {
        setMissBeforeAndAfterShip(ship, field);
        setMissOnSidesOfShip(ship, field);
    }

    private static void setMissBeforeAndAfterShip(Ship ship, Cell[][]
field) {
        setMissBefore(ship, field);
        setMissAfter(ship, field);
    }

    private static void setMissBefore(Ship ship, Cell[][] field) {
        List<Cell> cells = ship.getCells();
        Cell firstCell = cells.get(0);
        try {
            if (ship.isVertical())
                field[firstCell.getRow() +
1][firstCell.getColumn()].setMiss();
            else
                field[firstCell.getRow()][firstCell.getColumn() -
```

```
1].setMiss();
        } catch (IndexOutOfBoundsException ignored) {
        }

    }

    private static void setMissAfter(Ship ship, Cell[][] field) {
        List<Cell> cells = ship.getCells();
        Cell lastCell = cells.get(cells.size() - 1);
        try {
            if (ship.isVertical())
                field[lastCell.getRow() -
1][lastCell.getColumn()].setMiss();
            else
                field[lastCell.getRow()][lastCell.getColumn() +
1].setMiss();
        } catch (IndexOutOfBoundsException ignored) {
        }


    }

    private static void setMissOnSidesOfShip(Ship ship, Cell[][]
field) {
        Cell startCell = ship.getCells().get(0);
        int x = startCell.getColumn();
        int y = startCell.getRow();
        for (int i = 0; i < ship.getType().getSize() + 2; i++) {
            try {
                if (ship.isVertical()) {
                    if ((y - i + 1) >= 0) {
                        Cell cell = field[y - i + 1][x];
                        setMissLeftFromShip(field, cell);
                        setMissRightFromShip(field, cell);
                    }
                } else {
                    if ((x + i - 1) < GameField.FIELD_SIZE) {
                        Cell cell = field[y][x + i - 1];
                        setMissUnderShip(field, cell);
                        setMissTopOfShip(field, cell);
                    }
                }
            } catch (IndexOutOfBoundsException ignored) {
            }
        }
    }

    private static void setMissLeftFromShip(Cell[][] field, Cell
cell) {
        try {
            field[cell.getRow()][cell.getColumn() - 1].setMiss();
        } catch (IndexOutOfBoundsException ignored) {
        }
    }

    private static void setMissRightFromShip(Cell[][] field, Cell
cell) {
        try {
            field[cell.getRow()][cell.getColumn() + 1].setMiss();
```

```
            } catch (IndexOutOfBoundsException ignored) {
            }
        }


        private static void setMissUnderShip(Cell[][] field, Cell cell) {
            try {
                field[cell.getRow() + 1][cell.getColumn()].setMiss();
            } catch (IndexOutOfBoundsException ignored) {
            }
        }

        private static void setMissTopOfShip(Cell[][] field, Cell cell) {
            try {
                field[cell.getRow() - 1][cell.getColumn()].setMiss();
            } catch (IndexOutOfBoundsException ignored) {
            }
        }
    }
```

Приложение В. Код класса

com.r3nny.seabattle.client.core.controller.Bot.java

```
package com.r3nny.seabatlle.client.core.controller;

import com.badlogic.gdx.Gdx;
import com.r3nny.seabatlle.client.core.actors.Cell;
import com.r3nny.seabatlle.client.core.game.Game;
import com.r3nny.seabatlle.client.core.game.GameStatus;

import java.util.*;

public class Bot {
    private boolean isDirectionChanged;
    enum Direction {
        LEFT, RIGHT, TOP, BOTTOM
    }
    private final Random rd = new Random();
    private float timeFromLastShoot = 0F;
    private List<Cell> hittedCells;
    private Optional<Direction> direction = Optional.empty();
    private final List<Cell> cellsToShoot;


    public Bot() {
        this.hittedCells = new LinkedList<>();
        this.cellsToShoot = initCellsToShoot();

    }

    private List<Cell> initCellsToShoot() {
        List<Cell> cellsToShoot = new ArrayList<>();
        Cell[][] fields = Game.player.getField();
        for (Cell[] cell : fields) {
            cellsToShoot.addAll(Arrays.asList(cell));
        }
        return cellsToShoot;
    }

    public void update() {
        if (isEnemyTurn()) {
            timeFromLastShoot += Gdx.graphics.getDeltaTime();
        }
        if (isItTimeToShoot()) {
            Gdx.app.log("Bot", "Time to
shoot======================");
            if (direction.isPresent())
                keepDestroyingShip();
             else
                doRandomShot();
            timeFromLastShoot = 0;
        }

    }

    private boolean isEnemyTurn() {
        return GameStatus.ENEMY_TURN == Game.status;
    }
```

```
    private boolean isItTimeToShoot() {
        float TIME_BETWEEN_SHOOTS = 1F;
        return timeFromLastShoot > TIME_BETWEEN_SHOOTS &&
isEnemyTurn();
    }
    private void keepDestroyingShip() {
        Gdx.app.log("Bot", "Have direction: " + direction.get() + "
hitted: " + hittedCells);
        var optCell = getNextCellInCurrentDirection(direction.get());
        if (optCell.isPresent()) {
            shootOnSameDirection(optCell.get());
        } else {
            Gdx.app.log("Bot", "Cant shoot");
            changeDirection(direction.get());
        }
    }

    private Optional<Cell> getNextCellInCurrentDirection(Direction
direction) {
        Cell[][] field = Game.player.getField();
        Cell cell = getLastHittedCell();
        int row = cell.getRow();
        int column = cell.getColumn();
        try {
            Cell nextCell;
            switch (direction) {
                case TOP -> nextCell = field[row - 1][column];
                case BOTTOM -> nextCell = field[row + 1][column];
                case LEFT -> nextCell = field[row][column - 1];
                default -> nextCell = field[row][column + 1];
            }
            return nextCell.isSea() || nextCell.isHealthy() ?
Optional.of(nextCell) : Optional.empty();
        } catch (IndexOutOfBoundsException ignored) {
            return Optional.empty();
        }
    }

    private Cell getLastHittedCell() {
        return hittedCells.get(hittedCells.size() - 1);
    }

    private void shootOnSameDirection(Cell cell) {
        Gdx.app.log("Bot", "Can shoot on this");
        ShootController.shoot(cell.getRow(), cell.getColumn());
        cellsToShoot.remove(cell);
        if (cell.isMissed()) {
            Gdx.app.log("Bot", "Missed");
            changeDirection(direction.get());
        } else {
            if (cell.isInjured()){
                Gdx.app.log("Bot", "Injured. Adding cell to hitted");
                hittedCells.add(cell);
            }
            else{
                Gdx.app.log("Bot", "Killed");
                resetBot();
            }
```

```
        }
    }

    private void changeDirection(Direction current) {
        removeAllExtraCells();
        Gdx.app.log("Bot", "Changing direction. isChanged: " +
isDirectionChanged);
        if (isDirectionChanged) {
            changeDirectionToCorner(current);
            Gdx.app.log("Bot", "Changed corner to " +
direction.get());
        } else {
            changeDirectionToOpposite(current);
            Gdx.app.log("Bot", "Changed oposite to " +
direction.get());
        }
        isDirectionChanged = !isDirectionChanged;
    }
    private void removeAllExtraCells() {
        Cell firstCell = hittedCells.get(0);
        hittedCells = new ArrayList<>();
        hittedCells.add(firstCell);
    }


    private void changeDirectionToOpposite(Direction current) {
        switch (current) {
            case TOP -> direction = Optional.of(Direction.BOTTOM);
            case BOTTOM -> direction = Optional.of(Direction.TOP);
            case LEFT -> direction = Optional.of(Direction.RIGHT);
            default -> direction = Optional.of(Direction.LEFT);
        }
    }

    private void changeDirectionToCorner(Direction current) {
        switch (current) {
            case TOP -> direction = Optional.of(Direction.RIGHT);
            case BOTTOM -> direction = Optional.of(Direction.LEFT);
            case LEFT -> direction = Optional.of(Direction.TOP);
            default -> direction = Optional.of(Direction.BOTTOM);
        }
    }
    private void resetBot() {
        Gdx.app.log("Bot", "ResetBot");
        direction = Optional.empty();
        hittedCells = new ArrayList<>();
        isDirectionChanged = false;
    }
    private void doRandomShot() {
        Cell cell = getRandomCellToShoot();
        Gdx.app.log("Bot", "Random shoot to cell " + cell);
        ShootController.shoot(cell.getRow(), cell.getColumn());
        if (cell.isInjured()) {
            Gdx.app.log("Bot", "Injured");
            hittedCells.add(cell);
            cellsToShoot.remove(cell);
            setRandomDirection();
        } else {
            Gdx.app.log("Bot", "Miss or killed");
```

```
            resetBot();
        }
    }
    private Cell getRandomCellToShoot() {
        Cell cell = getRandomCell();
        while (!(cell.isSea() || cell.isHealthy())) {
            cell = getRandomCell();
        }
        return cell;
    }

    private Cell getRandomCell() {
        return cellsToShoot.get(rd.nextInt(cellsToShoot.size()));
    }
    private void setRandomDirection() {
        Direction[] directions = Direction.values();
        direction =
Optional.of(directions[rd.nextInt(directions.length)]);
    }
}
```

Приложение С. Весь исходный код

```
package com.r3nny.seabatlle.client;

import com.badlogic.gdx.backends.lwjgl3.Lwjgl3Application;
import com.badlogic.gdx.backends.lwjgl3.Lwjgl3ApplicationConfiguration;
import com.r3nny.seabatlle.client.core.StarBattle;

public class DesktopLauncher {
    public static void main(String[] arg) {
        Lwjgl3ApplicationConfiguration config = new
Lwjgl3ApplicationConfiguration();
        if (StarBattle.DEBUG) {
            config.setWindowedMode(1280, 720);
        } else {

config.setFullscreenMode(Lwjgl3ApplicationConfiguration.getDisplayMode());
        }
        config.useVsync(true);
        config.setForegroundFPS(60);
        config.setTitle("Star Battle");
        new Lwjgl3Application(new StarBattle(), config);
    }
}


package com.r3nny.seabatlle.client.core;

import com.badlogic.gdx.Gdx;
import com.badlogic.gdx.graphics.Color;
import com.badlogic.gdx.scenes.scene2d.Stage;
import com.badlogic.gdx.utils.ScreenUtils;
import com.badlogic.gdx.utils.viewport.FitViewport;
import com.r3nny.seabatlle.client.core.screen.SplashScreen;
import com.r3nny.seabatlle.client.core.utils.AnimationManager;
import com.r3nny.seabatlle.client.core.utils.Assets;
import com.r3nny.seabatlle.client.core.utils.SoundManager;

/**
 * Основной класс с которым рабоет LGJWL App*/
public class StarBattle extends com.badlogic.gdx.Game {

    public static Assets assetsManager;
    public static SoundManager soundManager;
    public static AnimationManager animationManager;
    public static final float WORLD_WIDTH = 1024;
    public static final float WORLD_HEIGHT = 576;

    /**Поле, отвечающее за режим отладки*/
    public static boolean DEBUG = false;

    /**@return Настроенная сцена для всех экранов*/
    public static Stage setUpStage() {
        Stage stage = new Stage(new FitViewport(StarBattle.WORLD_WIDTH,
StarBattle.WORLD_HEIGHT)); //Устанавливаю Viewport для одинаковой системы
координат при людом разрешении
        Gdx.input.setInputProcessor(stage);
        return stage;
    }

    @Override
    public void create() {
        this.setScreen(new SplashScreen());
    }
```

```
    public void resize(int width, int height) {
        super.resize(width, height);
    }

    @Override
    public void render() {
        super.render();
    }

    @Override
    public void dispose() {}
}
```

```
package com.r3nny.seabatlle.client.core.utils;

import com.badlogic.gdx.audio.Music;
import com.badlogic.gdx.audio.Sound;
import com.r3nny.seabatlle.client.core.StarBattle;

import java.util.Random;

/**Включение звуков и музыки*/
public class SoundManager {
    private final Random random = new Random();

    private final Music mainMusic;
    private final Music battleMusic;
    private final Music wonMusic;
    private final Music loseMusic;

    private final Sound newGameClick;
    private final Sound focusSound;
    private final Sound shipEnterSound;
    private final Sound clickSound;
    private final Sound missSound_1;
    private final Sound missSound_2;
    private final Sound injuredSound_1;
    private final Sound injuredSound_2;
    private final Sound killedSound;
    private final float musicVolume = 0.1F;
    private final float soundVolume = 0.8F;

    public SoundManager() {
        this.wonMusic = StarBattle.assetsManager.getWonMusic();
        this.loseMusic = StarBattle.assetsManager.getLoseMusic();
        this.mainMusic = StarBattle.assetsManager.getMainMusic();
        this.battleMusic = StarBattle.assetsManager.getBattleMusic();
        this.newGameClick = StarBattle.assetsManager.getNewGameClickSound();
        this.focusSound = StarBattle.assetsManager.getFocusSound();
        this.shipEnterSound = StarBattle.assetsManager.getShipEnterSound();
        this.clickSound = StarBattle.assetsManager.getClickSound();
        this.missSound_1 = StarBattle.assetsManager.getMissSound_1();
        this.missSound_2 = StarBattle.assetsManager.getMissSound_2();
        this.injuredSound_1 = StarBattle.assetsManager.getInjuredSound_1();
        this.injuredSound_2 = StarBattle.assetsManager.getInjuredSound_2();
        this.killedSound = StarBattle.assetsManager.getKilledSound();
        mainMusic.setLooping(true);
        battleMusic.setLooping(true);
    }

    public void playMainMusic() {
        mainMusic.setVolume(musicVolume);
        mainMusic.play();
    }
```

```
public void stopMainMusic() {
    mainMusic.stop();
}

public void playBattleMusic() {
    battleMusic.setVolume(musicVolume);
    battleMusic.play();
}
public void playWonMusic() {
    wonMusic.setVolume(musicVolume);
    wonMusic.play();
}

public void stopWonMusic() {
    wonMusic.stop();
}
public void playLoseMusic() {
    loseMusic.setVolume(musicVolume);
    loseMusic.play();
}

public void stopLoseMusic() {
    loseMusic.stop();
}

public void stopBattleMusic() {
    battleMusic.stop();
}

public void playNewGameSound() {
    newGameClick.play(soundVolume);
}

public void playShipEnterSound() {
    shipEnterSound.play(soundVolume);
}


public void playFocusButton() {
    focusSound.play(soundVolume);
}

public void playClickSound() {
    clickSound.play(soundVolume);
}

public void playMissSound() {

    if (random.nextBoolean()) {
        missSound_1.play(soundVolume);
    } else {
        missSound_2.play(soundVolume);
    }
}

public void playInjuredSound() {
    if (random.nextBoolean()) {
        injuredSound_1.play(soundVolume);
    } else {
        injuredSound_2.play(soundVolume);
    }
}
```

```
    public void playKilledSound() {
        killedSound.play(soundVolume);
    }
}
```

```
package com.r3nny.seabatlle.client.core.utils;

import com.badlogic.gdx.Gdx;
import com.badlogic.gdx.assets.AssetManager;
import com.badlogic.gdx.audio.Music;
import com.badlogic.gdx.audio.Sound;
import com.badlogic.gdx.graphics.Texture;
import com.badlogic.gdx.graphics.g2d.BitmapFont;
import com.badlogic.gdx.graphics.g2d.TextureAtlas;
import com.badlogic.gdx.graphics.g2d.freetype.FreeTypeFontGenerator;
import com.badlogic.gdx.scenes.scene2d.ui.Skin;

import static
com.badlogic.gdx.graphics.g2d.freetype.FreeTypeFontGenerator.DEFAULT_CHARS;

/**
 * Прокси над com.badlogic.gdx.assets.AssetManager
 */
public class Assets {

    private final String MAIN_MENU_BG_PATH = "mainMenu/bg.png";
    private final String ONE_DECK_SHIP_PATH = "ships/1xShip.png";
    private final String TWO_DECK_SHIP_PATH = "ships/2xShip.png";
    private final String THREE_DECK_SHIP_PATH = "ships/3xShip.png";
    private final String FOUR_DECK_SHIP_PATH = "ships/4xShip.png";
    private final String IN_GAME_BACKGROUND = "inGame/bg.jpg";

    private final String GAME_LOGO = "gameLogo.png";

    private final String MENU_BUTTON_SKIN = "menuButtons/skin.json";
    private final String CHOOSE_BUTTON_SKIN = "chooseButton/button.json";
    private final String MENU_BUTTON_ATLAS = "menuButtons/skin.atlas";
    private final String CHOOSE_BUTTON_ATLAS = "chooseButton/button.atlas";

    private final String MAIN_MUSIC = "sounds/mainMusic.mp3";
    private final String BATTLE_MUSIC = "sounds/battleMusic.mp3";

    private final String NEW_GAME_CLICK = "sounds/newGameClick.mp3";
    private final String FOCUS_SOUND = "sounds/focus.WAV";
    private final String SHIP_ENTER_SOUND = "sounds/shipEnter.WAV";
    private final String CLICK_SOUND = "sounds/click.WAV";
    private final String MISS_SOUND_1 = "sounds/miss_1.mp3";
    private final String MISS_SOUND_2 = "sounds/miss_2.mp3";

    private final String SHIP_INJURED_SOUND_1 = "sounds/injured_1.wav";
    private final String SHIP_INJURED_SOUND_2 = "sounds/injured_2.wav";
    private final String KILLED_SOUND = "sounds/killed.WAV";

    private final String INJURED_ANIMATION = "sprites/injuredSprite.png";
    private final String BURNING_ANIMATION = "sprites/burningSprite.png";
    private final String MISS_ANIMATION = "sprites/missSprite.png";

    private final String EXPLOSION_ANIMATION =
"sprites/explosionSprite.png";
    private final String WON_IMAGE = "won.png";
    private final String LOSE_IMAGE = "lose.png";
    private final String WON_SOUND = "sounds/winSound.mp3";
    private final String LOST_SOUND = "sounds/lostSound.mp3";
```

```
private final AssetManager manager;

public Assets() {
    manager = new AssetManager();
}

public void loadAllAssets() {
    manager.load(MAIN_MENU_BG_PATH, Texture.class);
    manager.load(ONE_DECK_SHIP_PATH, Texture.class);
    manager.load(TWO_DECK_SHIP_PATH, Texture.class);
    manager.load(THREE_DECK_SHIP_PATH, Texture.class);

    manager.load(FOUR_DECK_SHIP_PATH, Texture.class);
    manager.load(IN_GAME_BACKGROUND, Texture.class);
    manager.load(GAME_LOGO, Texture.class);
    manager.load(INJURED_ANIMATION, Texture.class);
    manager.load(BURNING_ANIMATION, Texture.class);
    manager.load(MISS_ANIMATION, Texture.class);
    manager.load(EXPLOSION_ANIMATION, Texture.class);
    manager.load(WON_IMAGE, Texture.class);
    manager.load(LOSE_IMAGE, Texture.class);


    manager.load(MENU_BUTTON_SKIN, Skin.class);
    manager.load(CHOOSE_BUTTON_SKIN, Skin.class);

    manager.load(MENU_BUTTON_ATLAS, TextureAtlas.class);
    manager.load(CHOOSE_BUTTON_ATLAS, TextureAtlas.class);
    String EXPLOSION_ATLAS = "sprites/explosionAtlas.atlas";
    manager.load(EXPLOSION_ATLAS, TextureAtlas.class);

    manager.load(MAIN_MUSIC, Music.class);
    manager.load(BATTLE_MUSIC, Music.class);
    manager.load(WON_SOUND, Music.class);
    manager.load(LOST_SOUND, Music.class);

    manager.load(NEW_GAME_CLICK, Sound.class);
    manager.load(SHIP_ENTER_SOUND, Sound.class);
    manager.load(FOCUS_SOUND, Sound.class);
    manager.load(CLICK_SOUND, Sound.class);
    manager.load(MISS_SOUND_1, Sound.class);
    manager.load(MISS_SOUND_2, Sound.class);
    manager.load(SHIP_INJURED_SOUND_1, Sound.class);
    manager.load(SHIP_INJURED_SOUND_2, Sound.class);
    manager.load(KILLED_SOUND, Sound.class);
}

public Music getWonMusic() {
    return manager.get(WON_SOUND);
}

public Music getLoseMusic() {
    return manager.get(LOST_SOUND);
}

public Texture getWonImage() {
    return manager.get(WON_IMAGE);
}

public Texture getLoseImage() {
    return manager.get(LOSE_IMAGE);
}

public Texture getMenuBackground() {
    return manager.get(MAIN_MENU_BG_PATH);
```

```
}

public Texture getOneDeckShip() {
    return manager.get(ONE_DECK_SHIP_PATH);
}

public Texture getTwoDeckShip() {
    return manager.get(TWO_DECK_SHIP_PATH);
}

public Texture getThreeDeckShip() {
    return manager.get(THREE_DECK_SHIP_PATH);
}

public Texture getFourDeckShip() {
    return manager.get(FOUR_DECK_SHIP_PATH);
}

public Texture getInGameBackground() {
    return manager.get(IN_GAME_BACKGROUND);
}

public Texture getMenuLogo() {
    return manager.get(GAME_LOGO);
}

public Skin getMenuButtonSkin() {
    Skin skin = manager.get(MENU_BUTTON_SKIN);
    skin.addRegions(manager.get(MENU_BUTTON_ATLAS));
    return skin;
}

public Skin getChooseButtonSkin() {
    Skin skin = manager.get(CHOOSE_BUTTON_SKIN);
    skin.addRegions(manager.get(CHOOSE_BUTTON_ATLAS));
    return skin;
}

public Music getMainMusic() {
    return manager.get(MAIN_MUSIC);
}

public Music getBattleMusic() {
    return manager.get(BATTLE_MUSIC);
}

public Sound getNewGameClickSound() {
    return manager.get(NEW_GAME_CLICK);
}

public Sound getFocusSound() {
    return manager.get(FOCUS_SOUND);
}

public Sound getShipEnterSound() {
    return manager.get(SHIP_ENTER_SOUND);
}

public Sound getClickSound() {
    return manager.get(CLICK_SOUND);
}

public Sound getMissSound_1() {
    return manager.get(MISS_SOUND_1);
}
```

```
    public Sound getMissSound_2() {
        return manager.get(MISS_SOUND_2);
    }

    public Sound getInjuredSound_1() {
        return manager.get(SHIP_INJURED_SOUND_1);
    }

    public Sound getInjuredSound_2() {
        return manager.get(SHIP_INJURED_SOUND_2);
    }

    public Texture getInjuredAnimation() {
        return manager.get(INJURED_ANIMATION);
    }

    public boolean update() {
        return manager.update();
    }

    public Sound getKilledSound() {
        return manager.get(KILLED_SOUND);
    }

    public Texture getBurningAnimation() {
        return manager.get(BURNING_ANIMATION);
    }

    public Texture getMissAnimation() {
        return manager.get(MISS_ANIMATION);
    }

    public Texture getExplosionAnimation() {
        return manager.get(EXPLOSION_ANIMATION);
    }

    public BitmapFont getFont(int size) {
        FreeTypeFontGenerator generator =
                new
FreeTypeFontGenerator(Gdx.files.internal("minecraft.ttf"));
        FreeTypeFontGenerator.FreeTypeFontParameter parameter =
                new FreeTypeFontGenerator.FreeTypeFontParameter();
        parameter.size = size;
        parameter.characters = DEFAULT_CHARS;
        return generator.generateFont(parameter);
    }
}

/* Nikita Vashkulatov(C) 2022 */
package com.r3nny.seabatlle.client.core.utils;

import com.badlogic.gdx.graphics.Texture;
import com.badlogic.gdx.graphics.g2d.Animation;
import com.badlogic.gdx.graphics.g2d.TextureRegion;
import com.badlogic.gdx.scenes.scene2d.Action;
import com.badlogic.gdx.scenes.scene2d.actions.Actions;
import com.badlogic.gdx.scenes.scene2d.actions.AlphaAction;
import com.badlogic.gdx.scenes.scene2d.actions.SequenceAction;
import com.r3nny.seabatlle.client.core.StarBattle;
import com.r3nny.seabatlle.client.core.actors.Ship;

/**Разбиение спрайтов на анимации, заготовление actions*/
public class AnimationManager {

    private final Animation<TextureRegion> injuredAnimation;
    private final Animation<TextureRegion> burningAnimation;
```

```
    private final Animation<TextureRegion> missAnimation;

    private final Animation<TextureRegion> explosionAnimation;

    public AnimationManager() {
        this.injuredAnimation =

loadAnimation(StarBattle.assetsManager.getInjuredAnimation(), 2, 5, 0.07F);
        this.burningAnimation =

loadAnimation(StarBattle.assetsManager.getBurningAnimation(), 2, 5, 0.07F);
        this.missAnimation =
                loadAnimation(StarBattle.assetsManager.getMissAnimation(),
4, 5, 0.05F);
        this.explosionAnimation =

loadAnimation(StarBattle.assetsManager.getExplosionAnimation(), 13, 1,
0.07f);
    }

    private Animation<TextureRegion> loadAnimation(
            Texture sheet, int rows, int cols, float duration) {
        TextureRegion[][] tmp =
                TextureRegion.split(sheet, sheet.getWidth() / cols,
sheet.getHeight() / rows);
        TextureRegion[] frames = new TextureRegion[cols * rows];
        int index = 0;
        for (int i = 0; i < rows; i++) {
            for (int j = 0; j < cols; j++) {
                frames[index++] = tmp[i][j];
            }
        }
        return new Animation<>(duration, frames);
    }

    public Animation<TextureRegion> getInjuredAnimation() {
        return injuredAnimation;
    }

    public Animation<TextureRegion> getBurningAnimation() {
        return burningAnimation;
    }

    public Animation<TextureRegion> getMissAnimation() {
        return missAnimation;
    }

    public Animation<TextureRegion> getExplosionAnimation() {
        return explosionAnimation;
    }


    public SequenceAction getFadeInAnimation() {
       return  Actions.sequence(Actions.alpha(0F), Actions.fadeIn(0.8F));
    }
    public AlphaAction getFadeOutAnimation() {
        return  Actions.fadeOut(0.5F);
    }

    /**Иммирует появление корабля со стороны
     * @param ship корабль
     * @param run код, который запускается после анимации*/
    public Action getShipEnterAction(Ship ship, Runnable run) {
        SequenceAction sequence;
        if (!ship.isVertical()) {
            sequence =
```

```
                    Actions.sequence(
                            Actions.moveTo(ship.getX() - 40, ship.getY()),
                            Actions.moveTo(ship.getX(), ship.getY(), 1F));
        } else {
            sequence =
                    Actions.sequence(
                            Actions.moveTo(ship.getX(), ship.getY() + 40),
                            Actions.moveTo(ship.getX(), ship.getY(), 1F));
        }
        return Actions.sequence(
                Actions.parallel(sequence,
Actions.sequence(Actions.alpha(0F), Actions.fadeIn(1F))),
                Actions.run(run));
    }
}
```

```
package com.r3nny.seabatlle.client.core.ui;

import com.badlogic.gdx.scenes.scene2d.Actor;
import com.badlogic.gdx.scenes.scene2d.InputEvent;
import com.badlogic.gdx.scenes.scene2d.actions.Actions;
import com.badlogic.gdx.scenes.scene2d.ui.TextButton;
import com.badlogic.gdx.scenes.scene2d.utils.ClickListener;
import com.r3nny.seabatlle.client.core.StarBattle;

/**Прокси над com.badlogic.gdx.scenes.scene2d.ui.TextButton; Исчезает после
нажатия*/
public class ChangeScreenButton extends TextButton {

    /**@param text Текст на кнопке
     * @param withFadeOut Действие, выполняемое во время анимации
исчезновения
     * @param afterFadeOut  Действие, выполняемое после исчезновения */
    public ChangeScreenButton(String text, Runnable withFadeOut, Runnable
afterFadeOut) {
        super(text, StarBattle.assetsManager.getMenuButtonSkin());
        super.addAction(StarBattle.animationManager.getFadeInAnimation());
        super.setSize(300, 50);
        super.addListener(new ClickListener() {
            @Override
            public void enter(
                    InputEvent event, float x, float y, int pointer, Actor
fromActor) {
                StarBattle.soundManager.playFocusButton();
            }

            @Override
            public void clicked(InputEvent event, float x, float y) {
                StarBattle.soundManager.playClickSound();
                StarBattle.soundManager.stopBattleMusic();
                StarBattle.soundManager.stopLoseMusic();
                StarBattle.soundManager.stopWonMusic();
                withFadeOut.run();
                ChangeScreenButton.super.addAction(
                        Actions.sequence(
                                Actions.fadeOut(0.5F),
                        Actions.run(afterFadeOut)));
            }
        });
    }

    public void removeWithFade() {
        super.addAction(StarBattle.animationManager.getFadeOutAnimation());
    }
```

```
}

/* Nikita Vashkulatov(C) 2022 */
package com.r3nny.seabatlle.client.core.screen;

import com.badlogic.gdx.Gdx;
import com.badlogic.gdx.Screen;
import com.badlogic.gdx.graphics.Color;
import com.badlogic.gdx.graphics.Texture;
import com.badlogic.gdx.scenes.scene2d.Stage;
import com.badlogic.gdx.scenes.scene2d.actions.Actions;
import com.badlogic.gdx.scenes.scene2d.ui.Image;
import com.badlogic.gdx.utils.ScreenUtils;
import com.r3nny.seabatlle.client.core.StarBattle;
import com.r3nny.seabatlle.client.core.utils.AnimationManager;
import com.r3nny.seabatlle.client.core.utils.Assets;
import com.r3nny.seabatlle.client.core.utils.SoundManager;

/**
 * Экран загрузки
 * */
public class SplashScreen implements Screen {
    private Image splashImage;
    private Stage stage;
    private StarBattle application;
    private Assets manager;
    private boolean startLoading = false;

    @Override
    public void show() {
        stage = StarBattle.setUpStage();
        StarBattle.assetsManager = new Assets();
        this.manager = StarBattle.assetsManager;
        application = ((StarBattle) Gdx.app.getApplicationListener());
        splashImage = new Image(new
Texture(Gdx.files.internal("gameLogo.png")));

        setUpSplashImage();
    }

    private void setUpSplashImage() {
        splashImage.setSize(620, 55);
        splashImage.setX(StarBattle.WORLD_WIDTH / 2 - splashImage.getWidth()
/ 2);
        splashImage.setY(StarBattle.WORLD_HEIGHT / 2);

        splashImage.addAction(
                Actions.sequence(
                        Actions.alpha(0.0F),
                        Actions.fadeIn(1.25F),
                        Actions.delay(1F),
                        Actions.run(
                                () -> {
                                    manager.loadAllAssets();
                                    startLoading = true;
                                })));

        stage.addActor(splashImage);

    }

    @Override
    public void render(float v) {
        ScreenUtils.clear(Color.BLACK);
        stage.act();
```

```
        stage.draw();
        if (manager.update() && startLoading) {
            StarBattle.soundManager = new SoundManager();
            StarBattle.animationManager = new AnimationManager();
            stage.addAction(
                    Actions.sequence(
                            Actions.fadeOut(0.5F),
                            Actions.run(
                                    () -> {
                                        stage.clear();
                                        if (StarBattle.DEBUG) {
                                            application.setScreen(new
ShipsCreatingScreen());
                                        } else {
                                            application.setScreen(new
MenuScreen());
                                        }
                                    })));
        }
    }

    @Override
    public void resize(int width, int height) {
        stage.getViewport().update(width, height, true);
    }

    @Override
    public void pause() {}

    @Override
    public void resume() {}

    @Override
    public void hide() {}

    @Override
    public void dispose() {}
}

/* Nikita Vashkulatov(C) 2022 */
package com.r3nny.seabatlle.client.core.screen;


import com.badlogic.gdx.graphics.Color;
import com.badlogic.gdx.scenes.scene2d.EventListener;
import com.badlogic.gdx.scenes.scene2d.actions.Actions;
import com.badlogic.gdx.scenes.scene2d.ui.Image;
import com.badlogic.gdx.scenes.scene2d.ui.Label;
import com.badlogic.gdx.utils.Align;
import com.badlogic.gdx.utils.ScreenUtils;
import com.r3nny.seabatlle.client.core.StarBattle;
import com.r3nny.seabatlle.client.core.actors.Cell;
import com.r3nny.seabatlle.client.core.game.Game;
import com.r3nny.seabatlle.client.core.game.GameStatus;
import com.r3nny.seabatlle.client.core.game.SingleGame;

import java.util.Random;

/**
 * Экран при одиночной игре
 */
public class SingleGameScreen extends InGameScreen {
    private final SingleGame game;

    private Label turnLabel;
```

```
    public SingleGameScreen(SingleGame game) {
        super(new Image(StarBattle.assetsManager.getInGameBackground()));
        this.game = game;

        Game.player.createShipsManager();
        Game.enemy.createShipsManager();
        Game.status = randomGameStatus();

        disableClicksOnPlayerField();
        setUpLabels();
        stage.addActor(Game.player);

        stage.addActor(Game.enemy);
        stage.setDebugAll(StarBattle.DEBUG);
    }

    private void setUpLabels() {
        Label.LabelStyle skin = new Label.LabelStyle();
        skin.font = assetManager().getFont(40);

        Label playerFieldLabel = new Label("Your Field", skin);
        playerFieldLabel.setFontScale(0.5F);
        playerFieldLabel.setPosition(
                Game.PLAYER_FIELD_X, Game.FIELD_Y +
playerFieldLabel.getHeight() - 20);
        Label enemyFieldLabel = new Label("Enemy Field", skin);
        enemyFieldLabel.setFontScale(0.5F);
        enemyFieldLabel.setPosition(
                Game.ENEMY_FIELD_X, Game.FIELD_Y +
enemyFieldLabel.getHeight() - 20);

        turnLabel = new Label("", skin);
        turnLabel.setFontScale(0.5F);
        turnLabel.setWidth(Game.ENEMY_FIELD_X - Game.PLAYER_FIELD_X +
Cell.SIZE * 10);
        turnLabel.setPosition(
                StarBattle.WORLD_WIDTH / 2 + 3 - turnLabel.getWidth() / 2,
Game.FIELD_Y);
        turnLabel.setAlignment(Align.center);
        stage.addActor(playerFieldLabel);
        stage.addActor(enemyFieldLabel);
        stage.addActor(turnLabel);

    }

    private void disableClicksOnPlayerField() {
        Cell[][] cells = Game.player.getField();
        for (Cell[] cell : cells) {
            for (int j = 0; j < cells.length; j++) {
                var listeners = cell[j].getListeners();
                for (EventListener listener : listeners) {
                    cell[j].removeListener(listener);
                }
            }
        }
    }


    private GameStatus randomGameStatus() {
        Random rd = new Random();
        return rd.nextBoolean() ? GameStatus.PLAYER_TURN :
GameStatus.ENEMY_TURN;
    }

    @Override
    public void show() {
```

```
    }


    private boolean isEnemyDead() {
        return Game.enemy.isAllShipsKilled();
    }

    private boolean isPlayerDead() {
        return Game.player.isAllShipsKilled();
    }

    private boolean isGameOver() {
        return isEnemyDead() || isPlayerDead();
    }

    @Override
    public void render(float v) {
        updateTurnLabel();
        game.update();
        if (isGameOver()) {
            soundManager().stopBattleMusic();
            Game.status = GameStatus.END;
            stage.addAction(
                    Actions.sequence(
                            animationManager().getFadeOutAnimation(),
                            Actions.run(() -> {
                                stage.clear();
                                application.setScreen(new
EndScreen(isEnemyDead()));

                            })));
        }
        stage.act();
        stage.draw();
    }

    private void updateTurnLabel() {
        turnLabel.setText(
                (Game.status == GameStatus.PLAYER_TURN) ? "Shoot, Admiral" :
"Enemy shooting");
        ScreenUtils.clear(new Color(Color.BLACK));
    }

    @Override
    public void resize(int width, int height) {
        stage.getViewport().update(width, height, true);
    }

    @Override
    public void pause() {
    }

    @Override
    public void resume() {
    }

    @Override
    public void hide() {
    }

    @Override
    public void dispose() {
    }
}
```

```java
/* Nikita Vashkulatov(C) 2022 */
package com.r3nny.seabatlle.client.core.screen;

import com.badlogic.gdx.Gdx;
import com.badlogic.gdx.Input;
import com.badlogic.gdx.scenes.scene2d.InputEvent;
import com.badlogic.gdx.scenes.scene2d.ui.Image;
import com.badlogic.gdx.scenes.scene2d.ui.Label;
import com.badlogic.gdx.scenes.scene2d.ui.TextButton;
import com.badlogic.gdx.scenes.scene2d.utils.ClickListener;
import com.r3nny.seabatlle.client.core.StarBattle;
import com.r3nny.seabatlle.client.core.actors.ShipsCreatingArea;
import com.r3nny.seabatlle.client.core.controller.ShipsCreator;
import com.r3nny.seabatlle.client.core.game.Game;
import com.r3nny.seabatlle.client.core.game.GameStatus;
import com.r3nny.seabatlle.client.core.game.SingleGame;

import static com.r3nny.seabatlle.client.core.game.Game.player;

/**
 * Экран с созданием кораблей игрока
 */
public class ShipsCreatingScreen extends InGameScreen {

    private final SingleGame game;

    private TextButton acceptButton;

    private ShipsCreatingArea shipsCreatingArea;

    public ShipsCreatingScreen() {
        super(new Image(StarBattle.assetsManager.getInGameBackground()));
        Game.status = GameStatus.SHIPS_STAGE;
        setUpAcceptButton();
        setUpShipsCreatingArea();
        setUpLabels();
        game = new SingleGame();
        stage.addActor(Game.player);
        stage.setDebugAll(StarBattle.DEBUG);
    }


    private void setUpAcceptButton() {
        acceptButton = new TextButton("Accept",
assetManager().getMenuButtonSkin());
        acceptButton.setSize(200, 50);
        acceptButton.setX(StarBattle.WORLD_WIDTH - acceptButton.getWidth() -
10);
        acceptButton.setY(10);
        acceptButton.setVisible(false);
        acceptButton.addListener(
                new ClickListener() {
                    @Override
                    public void clicked(InputEvent event, float x, float y)
{
                        player.setShipsReady(true);
                    }
                });

        stage.addActor(acceptButton);
    }


    private void setUpShipsCreatingArea() {
        shipsCreatingArea = new ShipsCreatingArea(Game.ENEMY_FIELD_X,
Game.FIELD_Y);
```

```
        stage.addActor(shipsCreatingArea);
    }

    private void setUpLabels() {

        Label.LabelStyle skin = new Label.LabelStyle();
        skin.font = StarBattle.assetsManager.getFont(40);

        Label playerFieldLabel = new Label("Your Field", skin);
        playerFieldLabel.setFontScale(0.5F);
        playerFieldLabel.setPosition(
                Game.PLAYER_FIELD_X, Game.FIELD_Y +
playerFieldLabel.getHeight() - 20);
        stage.addActor(playerFieldLabel);

    }


    @Override
    public void show() {
        soundManager().playBattleMusic();
    }

    @Override
    public void render(float delta) {
        super.render(delta);
        if (Gdx.input.isKeyPressed(Input.Keys.A)) {
            if (!ShipsCreator.isAnyShipLanding) {
                acceptButton.setVisible(true);
                Game.player.createShipsAutomaticaly();
            }
        }
        if (game.isShipsReady()) {
            Game.player.clearAllMissed();
            shipsCreatingArea.remove();
            application.setScreen(new SingleGameScreen(game));
        }
    }

    @Override
    public void resize(int width, int height) {
        stage.getViewport().update(width, height, true);
    }

    @Override
    public void pause() {
    }

    @Override
    public void resume() {
    }

    @Override
    public void hide() {
        stage.clear();
    }

    @Override
    public void dispose() {
        stage.dispose();
    }
}

package com.r3nny.seabatlle.client.core.screen;

import com.badlogic.gdx.Gdx;
```

```
import com.badlogic.gdx.graphics.Color;
import com.badlogic.gdx.scenes.scene2d.Stage;
import com.badlogic.gdx.utils.ScreenUtils;
import com.r3nny.seabatlle.client.core.StarBattle;
import com.r3nny.seabatlle.client.core.utils.AnimationManager;
import com.r3nny.seabatlle.client.core.utils.Assets;
import com.r3nny.seabatlle.client.core.utils.SoundManager;

public abstract class Screen implements com.badlogic.gdx.Screen {
    protected final StarBattle application;
    protected final Stage stage;



    protected Screen() {
        this.stage = StarBattle.setUpStage();
        this.application = (StarBattle) Gdx.app.getApplicationListener();


    }

    protected AnimationManager animationManager() {
        return StarBattle.animationManager;
    }
    protected Assets assetManager() {
        return StarBattle.assetsManager;
    }
    protected SoundManager soundManager() {
        return StarBattle.soundManager;
    }

    @Override
    public void render(float v) {
        ScreenUtils.clear(Color.BLACK);
        stage.act();
        stage.draw();
    }
}

/* Nikita Vashkulatov(C) 2022 */
package com.r3nny.seabatlle.client.core.screen;

import com.badlogic.gdx.Gdx;

import com.badlogic.gdx.graphics.Color;
import com.badlogic.gdx.scenes.scene2d.Stage;
import com.badlogic.gdx.scenes.scene2d.actions.Actions;
import com.badlogic.gdx.scenes.scene2d.ui.Image;
import com.badlogic.gdx.scenes.scene2d.ui.Label;
import com.badlogic.gdx.utils.ScreenUtils;
import com.r3nny.seabatlle.client.core.StarBattle;
import com.r3nny.seabatlle.client.core.game.Game;
import com.r3nny.seabatlle.client.core.game.GameStatus;
import com.r3nny.seabatlle.client.core.ui.ChangeScreenButton;
import com.r3nny.seabatlle.client.core.utils.Assets;

/**
 * Экран Меню
 */
public class MenuScreen extends Screen {
    private Image menuLogo;
    private ChangeScreenButton start = null;
    private ChangeScreenButton end;


    public MenuScreen() {
```

```
        super();
        Game.status = GameStatus.MENU;
        stage.addAction(Actions.sequence(Actions.alpha(0F),
Actions.fadeIn(1F)));
        setUpBgImage();
        setUpMenuLogo();
        setUpLabel();
        setUpButtons();
    }

    private void setUpBgImage() {
        Image bgImage = new Image(assetManager().getMenuBackground());
        bgImage.setSize(StarBattle.WORLD_WIDTH, StarBattle.WORLD_HEIGHT);

        stage.addActor(bgImage);
    }

    private void setUpMenuLogo() {
        menuLogo = new Image(assetManager().getMenuLogo());
        menuLogo.setSize(620, 55);
        menuLogo.setX(StarBattle.WORLD_WIDTH / 2 - menuLogo.getWidth() / 2);
        menuLogo.setY(StarBattle.WORLD_HEIGHT - menuLogo.getHeight() - 20);

        stage.addActor(menuLogo);
    }

    private void setUpLabel() {
        Label.LabelStyle labelStyle = new Label.LabelStyle();
        labelStyle.fontColor = new Color(Color.WHITE);
        labelStyle.font = assetManager().getFont(16);
        Label label = new Label("by Nikita Vashkulatov, design by Danil
Ionov", labelStyle);
        label.setPosition(10, 0);
        stage.addActor(label);
    }

    private void setUpButtons() {
        end = new ChangeScreenButton("Exit", smoothFadeOut(), () ->
Gdx.app.exit());
        start = new ChangeScreenButton("New game", smoothFadeOut(), () ->
MenuScreen.this.application.setScreen(new ChooseScreen()));

        start.setX(StarBattle.WORLD_WIDTH - 350);
        start.setSize(300, 50);
        end.setX(start.getX());
        start.setY(100);
        end.setY(start.getY() - 60);

        stage.addActor(start);
        stage.addActor(end);
    }


    public Runnable smoothFadeOut() {
        return () -> {
            start.removeWithFade();
            end.removeWithFade();
            menuLogo.addAction(animationManager().getFadeOutAnimation());
        };
    }

    @Override
    public void show() {
        soundManager().playMainMusic();
        Gdx.app.log("Menu Screen", "Showing menu screen");
    }
```

```
    @Override
    public void resize(int i, int i1) {

    }

    @Override
    public void pause() {

    }

    @Override
    public void resume() {

    }

    @Override
    public void hide() {

    }

    @Override
    public void dispose() {

    }


}
```

```
package com.r3nny.seabatlle.client.core.screen;

import com.badlogic.gdx.scenes.scene2d.ui.Image;
import com.r3nny.seabatlle.client.core.StarBattle;
import com.r3nny.seabatlle.client.core.ui.ChangeScreenButton;

public abstract class InGameScreen extends Screen{
    protected Image gameLogo;
    protected Image bgImage;
    protected InGameScreen(Image bgImage) {
        this.bgImage = bgImage;
        setUpBgImage();
        setUpGameLogo();
        setUpBackButton();
    }

    private void setUpBgImage() {
        bgImage.setSize(StarBattle.WORLD_WIDTH, StarBattle.WORLD_HEIGHT);

        stage.addActor(bgImage);
    }

    private void setUpGameLogo() {
        Image  gameLogo = new Image(assetManager().getMenuLogo());
        gameLogo.setSize(310, 27);
        gameLogo.setX(StarBattle.WORLD_WIDTH / 2 - gameLogo.getWidth() / 2);
        gameLogo.setY(StarBattle.WORLD_HEIGHT - gameLogo.getHeight() - 20);

        stage.addActor(gameLogo);
    }

    private void setUpBackButton() {
        ChangeScreenButton backButton =
                new ChangeScreenButton("Back to menu",
```

```
runOnWhenChangingScreen(), () -> {
                    stage.clear();
                    application.setScreen(new MenuScreen());
                });
        backButton.setX(10);
        backButton.setSize(200, 50);
        backButton.setY(StarBattle.WORLD_HEIGHT - 10 -
backButton.getHeight());

        stage.addActor(backButton);
    }

    private Runnable runOnWhenChangingScreen(){
        return () ->
stage.addAction(animationManager().getFadeOutAnimation());
    }
}

package com.r3nny.seabatlle.client.core.screen;

import com.badlogic.gdx.graphics.Texture;
import com.badlogic.gdx.scenes.scene2d.ui.Image;
import com.r3nny.seabatlle.client.core.StarBattle;

import static com.r3nny.seabatlle.client.core.StarBattle.soundManager;

public class EndScreen extends InGameScreen {


    public EndScreen(boolean isPlayerWin) {
        super(new Image(StarBattle.assetsManager.getMenuBackground()));

        if (isPlayerWin)
            soundManager().playWonMusic();
        else
            soundManager.playLoseMusic();

        Texture texture = isPlayerWin ? assetManager().getWonImage() :
assetManager().getLoseImage();
        //    private final ShipManager shipsLeft;
        Image result = new Image(texture);
        result.setSize((float)texture.getWidth()/2,
(float)texture.getHeight()/2);
        result.setPosition(StarBattle.WORLD_WIDTH/2 - result.getWidth()/2
,StarBattle.WORLD_HEIGHT/2 - result.getHeight()/2);

        stage.addActor(result);
    }



    @Override
    public void show() {

    }


    @Override
    public void resize(int i, int i1) {

    }

    @Override
    public void pause() {

    }
```

```
    @Override
    public void resume() {

    }

    @Override
    public void hide() {

    }

    @Override
    public void dispose() {

    }
}

/* Nikita Vashkulatov(C) 2022 */
package com.r3nny.seabatlle.client.core.screen;

import com.badlogic.gdx.graphics.Color;
import com.badlogic.gdx.scenes.scene2d.Actor;
import com.badlogic.gdx.scenes.scene2d.InputEvent;
import com.badlogic.gdx.scenes.scene2d.ui.Image;
import com.badlogic.gdx.scenes.scene2d.ui.ImageButton;
import com.badlogic.gdx.scenes.scene2d.utils.ClickListener;
import com.badlogic.gdx.utils.ScreenUtils;
import com.r3nny.seabatlle.client.core.StarBattle;

/**Экран с выбором режима игры*/
public class ChooseScreen extends InGameScreen {
    private ImageButton singleGame;


    public ChooseScreen( ) {
        super(new Image (StarBattle.assetsManager.getMenuBackground()));
        setUpSingleGameButton();
        setUpMultiGameButton();

    }



    private void setUpSingleGameButton() {
        singleGame = new ImageButton(assetManager().getChooseButtonSkin());
        singleGame.setSize(261, 257);
        singleGame.setX(StarBattle.WORLD_WIDTH / 2 - 25 -
singleGame.getWidth());
        singleGame.setY(StarBattle.WORLD_HEIGHT / 2 - singleGame.getHeight()
/ 2);
        singleGame.addListener(
                new ClickListener() {
                    @Override
                    public void enter(
                            InputEvent event, float x, float y, int pointer,
Actor fromActor) {
                        soundManager().playFocusButton();
                    }

                    @Override
                    public void clicked(InputEvent event, float x, float y)
{
                        soundManager().stopMainMusic();
                        soundManager().playNewGameSound();
                        application.setScreen(new ShipsCreatingScreen());
                    }
```

```
                });
        singleGame.addAction(animationManager().getFadeInAnimation());

        stage.addActor(singleGame);
    }

    private void setUpMultiGameButton() {
        ImageButton multiGame = new
ImageButton(assetManager().getChooseButtonSkin(), "multiPlayer");

        multiGame.setSize(singleGame.getWidth(), singleGame.getHeight());
        multiGame.setX(StarBattle.WORLD_WIDTH / 2 + 25);
        multiGame.setY(StarBattle.WORLD_HEIGHT / 2 - singleGame.getHeight()
/ 2);

        multiGame.addAction(animationManager().getFadeInAnimation());

        stage.addActor(multiGame);
    }



    @Override
    public void show() {
    }



    @Override
    public void resize(int width, int height) {
        stage.getViewport().update(width, height, true);
    }

    @Override
    public void pause() {
    }

    @Override
    public void resume() {
    }

    @Override
    public void hide() {
    }

    @Override
    public void dispose() {
    }
}

/* Nikita Vashkulatov(C) 2022 */
package com.r3nny.seabatlle.client.core.game;

import com.r3nny.seabatlle.client.core.actors.EnemyGameField;
import com.r3nny.seabatlle.client.core.actors.PlayerGameField;
import com.r3nny.seabatlle.client.core.controller.Bot;

public class SingleGame extends Game {
    private final Bot bot;

    public SingleGame() {
        player = new PlayerGameField(PLAYER_FIELD_X, FIELD_Y);
        enemy = new EnemyGameField(ENEMY_FIELD_X, FIELD_Y);
        bot = new Bot();
```

```
    }

    public boolean isShipsReady() {
        return player.isShipsReady() && enemy.isShipsReady();
    }

    public  void  update(){
        bot.update();
    }

}

/* Nikita Vashkulatov(C) 2022 */
package com.r3nny.seabatlle.client.core.game;

public enum GameStatus {
    MENU,
    SHIPS_STAGE,
    PLAYER_TURN,
    ENEMY_TURN,
    END
}

/* Nikita Vashkulatov(C) 2022 */
package com.r3nny.seabatlle.client.core.game;

import com.r3nny.seabatlle.client.core.actors.EnemyGameField;
import com.r3nny.seabatlle.client.core.actors.PlayerGameField;

/**Контейнер с статусом и полями*/
public class Game {
    public static final int FIELD_Y = 426;
    public static final int PLAYER_FIELD_X = 89;
    public static final int ENEMY_FIELD_X = 633;
    public static GameStatus status = GameStatus.SHIPS_STAGE;
    public static PlayerGameField player;
    public static EnemyGameField enemy;
}

/* Nikita Vashkulatov(C) 2022 */
package com.r3nny.seabatlle.client.core.exeptions;

public class CantCreateShipException extends Exception {
    public CantCreateShipException(String message) {
        super(message);
    }
}

/* Nikita Vashkulatov(C) 2022 */
package com.r3nny.seabatlle.client.core.controller;

import static com.r3nny.seabatlle.client.core.game.Game.enemy;
import static com.r3nny.seabatlle.client.core.game.Game.player;

import com.r3nny.seabatlle.client.core.game.Game;
import com.r3nny.seabatlle.client.core.game.GameStatus;
import com.r3nny.seabatlle.client.core.StarBattle;
import com.r3nny.seabatlle.client.core.actors.Cell;
import com.r3nny.seabatlle.client.core.actors.Ship;
import java.util.List;


/**Содержит единственный метод для обработки выстрела*/
public class ShootController {

    /**Меняет состояния клетки на поле в зависимости от результата и хода*/
    public static void shoot(int row, int column) {
```

```
        Cell cell = getHittedCell(row, column);
        if (isShotMissed(cell)) {
            changeGameTurn();
            makeMiss(cell);
        } else {
            if (isShotHitted(cell)) {
                makeInured(cell);
                if (isKilling(cell)) makeKilled(cell);
            }
        }
    }

    private static Cell getHittedCell(int row, int column) {
        return isPlayerTurn()
                ? Game.enemy.getField()[row][column]
                : Game.player.getField()[row][column];
    }

    private static boolean isPlayerTurn() {
        return Game.status == GameStatus.PLAYER_TURN;
    }

    private static boolean isShotMissed(Cell cell) {
        return cell.isSea();
    }

    private static void changeGameTurn() {
        if (isPlayerTurn()) setEnemyTurn();
        else setPlayerTurn();
    }

    private static void setEnemyTurn() {
        Game.status = GameStatus.ENEMY_TURN;
    }

    private static void setPlayerTurn() {
        Game.status = GameStatus.PLAYER_TURN;
    }

    private static void makeMiss(Cell cell) {
        StarBattle.soundManager.playMissSound();
        cell.setMiss();
    }

    private static boolean isShotHitted(Cell cell) {
        return cell.isHealthy();
    }

    private static void makeInured(Cell cell) {
        cell.setInjured();
        StarBattle.soundManager.playInjuredSound();
    }

    private static boolean isKilling(Cell cell) {
        Ship ship = cell.getShip();
        List<Cell> cells = ship.getCells();
        return cells.stream().filter(Cell::isHealthy).toList().isEmpty();
    }

    private static void makeKilled(Cell cell) {
        Ship ship = cell.getShip();
        if (isPlayerTurn())
            enemy.killShip(ship);
        else
            player.killShip(ship);
        StarBattle.soundManager.playKilledSound();
```

```
    }
}

/* Nikita Vashkulatov(C) 2022 */
package com.r3nny.seabatlle.client.core.controller;

import com.badlogic.gdx.Gdx;
import com.r3nny.seabatlle.client.core.actors.Cell;
import com.r3nny.seabatlle.client.core.actors.GameField;
import com.r3nny.seabatlle.client.core.actors.Ship;
import com.r3nny.seabatlle.client.core.actors.ShipType;
import com.r3nny.seabatlle.client.core.exeptions.CantCreateShipException;

import java.util.LinkedList;
import java.util.List;
import java.util.Random;

/**
 * Статические методы для создания кораблей
 */
public class ShipsCreator {
    private static Random rd;

    /**
     * Учет успешно созданных кораблей игрока
     */
    public static int createdPlayerShips = 0;

    public static boolean isAnyShipLanding = false;

    public static ShipType[] shipTypes = {
            ShipType.FOUR_DECK,
            ShipType.THREE_DECK,
            ShipType.THREE_DECK,
            ShipType.TWO_DECK,
            ShipType.TWO_DECK,
            ShipType.TWO_DECK,
            ShipType.ONE_DECK,
            ShipType.ONE_DECK,
            ShipType.ONE_DECK,
            ShipType.ONE_DECK
    };

    /**
     * Автоматически создает корабли на поле
     *
     * @param gf Игровое поле
     */
    public static void autoCreateShips(GameField gf) {
        rd = new Random();
        Cell[][] field = gf.getField();
        List<Ship> ships = gf.getShips();
        int createdShips = 0;
        while (createdShips < shipTypes.length) {
            Cell cell = getRandomCellFromField(field);
            Ship ship = ships.get(createdShips);
            if (rd.nextBoolean())
                ship.rotate();
            try {
                addShipToGameField(cell, ships.get(createdShips), gf);
                createdShips++;
            } catch (CantCreateShipException ex) {
                System.err.println("Cant create ship on Cell" + cell);
            }
        }
        Gdx.app.log("ShipsCreator", "All ships automatically created");
```

```
    }

    private static Cell getRandomCellFromField(Cell[][] field) {
        int row = rd.nextInt(GameField.FIELD_SIZE);
        int column = rd.nextInt(GameField.FIELD_SIZE);
        return field[row][column];
    }


    /**
     * Помещает корабль на поле с началом на определенной клетке
     *
     * @param cell начало корабля
     * @param ship корабль
     * @param gf   поле
     * @throws CantCreateShipException если нельзя поместить корабль
     */
    public static void addShipToGameField(Cell cell, Ship ship, GameField
gf) throws CantCreateShipException {
        List<Cell> shipCells = getShipCells(cell, ship, gf.getField());
        linkShipAndCells(ship, shipCells);
        setCellsAroundShipMissed(ship, gf.getField());
    }

    /**
     * @return Список клеток в которые добавляется корабль
     * @throws CantCreateShipException если нельзя поместить корабль
     */
    private static List<Cell> getShipCells(Cell startCell, Ship ship,
Cell[][] field)
            throws CantCreateShipException {
        List<Cell> cells = new LinkedList<>();
        for (int i = 0; i < ship.getType().getSize(); i++) {
            tryAddCellToList(startCell, ship, field, i, cells);
        }
        return cells;
    }

    private static void tryAddCellToList(
            Cell startCell, Ship ship, Cell[][] field, int i, List<Cell>
cells)
            throws CantCreateShipException {
        try {
            Cell currentCell = getCurrentCell(startCell, ship, field, i);
            if (currentCell.isSea())
                cells.add(currentCell);
            else
                throw new CantCreateShipException("Cells not clear");
        } catch (IndexOutOfBoundsException e) {
            throw new CantCreateShipException("Ship cant fit on map");
        }
    }

    private static Cell getCurrentCell(Cell startCell, Ship ship, Cell[][]
field, int i)
            throws IndexOutOfBoundsException {
        if (ship.isVertical())
            return field[startCell.getRow() - i][startCell.getColumn()];
        else
            return field[startCell.getRow()][startCell.getColumn() + i];
    }

    /**
     * Связывает корабль и клетки в двухстороннем порядке
     */
    private static void linkShipAndCells(Ship ship, List<Cell> shipCells) {
```

```
        Cell startCell = shipCells.get(0);
        ship.setPosition(startCell.getX(), startCell.getY());
        ship.setCells(shipCells);
        setShipToCells(ship);
    }

    private static void setShipToCells(Ship ship) {
        for (Cell cell : ship.getCells()) {
            cell.setShip(ship);
            cell.setHealthy();
        }
    }

    public static void setCellsAroundShipMissed(Ship ship, Cell[][] field) {
        setMissBeforeAndAfterShip(ship, field);
        setMissOnSidesOfShip(ship, field);
    }

    private static void setMissBeforeAndAfterShip(Ship ship, Cell[][] field)
{
        setMissBefore(ship, field);
        setMissAfter(ship, field);
    }

    private static void setMissBefore(Ship ship, Cell[][] field) {
        List<Cell> cells = ship.getCells();
        Cell firstCell = cells.get(0);
        try {
            if (ship.isVertical())
                field[firstCell.getRow() +
1][firstCell.getColumn()].setMiss();
            else
                field[firstCell.getRow()][firstCell.getColumn() -
1].setMiss();
        } catch (IndexOutOfBoundsException ignored) {
        }

    }

    private static void setMissAfter(Ship ship, Cell[][] field) {
        List<Cell> cells = ship.getCells();
        Cell lastCell = cells.get(cells.size() - 1);
        try {
            if (ship.isVertical())
                field[lastCell.getRow() -
1][lastCell.getColumn()].setMiss();
            else
                field[lastCell.getRow()][lastCell.getColumn() +
1].setMiss();
        } catch (IndexOutOfBoundsException ignored) {
        }


    }

    private static void setMissOnSidesOfShip(Ship ship, Cell[][] field) {
        Cell startCell = ship.getCells().get(0);
        int x = startCell.getColumn();
        int y = startCell.getRow();
        for (int i = 0; i < ship.getType().getSize() + 2; i++) {
            try {
                if (ship.isVertical()) {
                    if ((y - i + 1) >= 0) {
                        Cell cell = field[y - i + 1][x];
                        setMissLeftFromShip(field, cell);
                        setMissRightFromShip(field, cell);
```

```
                    }
                } else {
                    if ((x + i - 1) < GameField.FIELD_SIZE) {
                        Cell cell = field[y][x + i - 1];
                        setMissUnderShip(field, cell);
                        setMissTopOfShip(field, cell);
                    }
                }
            } catch (IndexOutOfBoundsException ignored) {
            }
        }
    }


    private static void setMissLeftFromShip(Cell[][] field, Cell cell) {
        try {
            field[cell.getRow()][cell.getColumn() - 1].setMiss();
        } catch (IndexOutOfBoundsException ignored) {
        }
    }

    private static void setMissRightFromShip(Cell[][] field, Cell cell) {
        try {
            field[cell.getRow()][cell.getColumn() + 1].setMiss();
        } catch (IndexOutOfBoundsException ignored) {
        }
    }


    private static void setMissUnderShip(Cell[][] field, Cell cell) {
        try {
            field[cell.getRow() + 1][cell.getColumn()].setMiss();
        } catch (IndexOutOfBoundsException ignored) {
        }
    }

    private static void setMissTopOfShip(Cell[][] field, Cell cell) {
        try {
            field[cell.getRow() - 1][cell.getColumn()].setMiss();
        } catch (IndexOutOfBoundsException ignored) {
        }
    }

}

package com.r3nny.seabatlle.client.core.controller;

import com.badlogic.gdx.Gdx;
import com.r3nny.seabatlle.client.core.actors.Cell;
import com.r3nny.seabatlle.client.core.game.Game;
import com.r3nny.seabatlle.client.core.game.GameStatus;

import java.util.*;

public class Bot {
    private boolean isDirectionChanged;
    enum Direction {
        LEFT, RIGHT, TOP, BOTTOM
    }
    private final Random rd = new Random();
    private float timeFromLastShoot = 0F;
    private List<Cell> hittedCells;
    private Optional<Direction> direction = Optional.empty();
    private final List<Cell> cellsToShoot;
```

```
    public Bot() {
        this.hittedCells = new LinkedList<>();
        this.cellsToShoot = initCellsToShoot();

    }

    private List<Cell> initCellsToShoot() {
        List<Cell> cellsToShoot = new ArrayList<>();
        Cell[][] fields = Game.player.getField();
        for (Cell[] cell : fields) {
            cellsToShoot.addAll(Arrays.asList(cell));
        }
        return cellsToShoot;
    }

    public void update() {
        if (isEnemyTurn()) {
            timeFromLastShoot += Gdx.graphics.getDeltaTime();
        }
        if (isItTimeToShoot()) {
            Gdx.app.log("Bot", "Time to shoot======================");
            if (direction.isPresent())
                keepDestroyingShip();
             else
                doRandomShot();
            timeFromLastShoot = 0;
        }

    }

    private boolean isEnemyTurn() {
        return GameStatus.ENEMY_TURN == Game.status;
    }

    private boolean isItTimeToShoot() {
        float TIME_BETWEEN_SHOOTS = 1F;
        return timeFromLastShoot > TIME_BETWEEN_SHOOTS && isEnemyTurn();
    }
    private void keepDestroyingShip() {
        Gdx.app.log("Bot", "Have direction: " + direction.get() + " hitted:
" + hittedCells);
        var optCell = getNextCellInCurrentDirection(direction.get());
        if (optCell.isPresent()) {
            shootOnSameDirection(optCell.get());
        } else {
            Gdx.app.log("Bot", "Cant shoot");
            changeDirection(direction.get());
        }
    }

    private Optional<Cell> getNextCellInCurrentDirection(Direction
direction) {
        Cell[][] field = Game.player.getField();
        Cell cell = getLastHittedCell();
        int row = cell.getRow();
        int column = cell.getColumn();
        try {
            Cell nextCell;
            switch (direction) {
                case TOP -> nextCell = field[row - 1][column];
                case BOTTOM -> nextCell = field[row + 1][column];
                case LEFT -> nextCell = field[row][column - 1];
                default -> nextCell = field[row][column + 1];
            }
            return nextCell.isSea() || nextCell.isHealthy() ?
Optional.of(nextCell) : Optional.empty();
```

```
            } catch (IndexOutOfBoundsException ignored) {
                return Optional.empty();
            }
        }

        private Cell getLastHittedCell() {
            return hittedCells.get(hittedCells.size() - 1);
        }

        private void shootOnSameDirection(Cell cell) {
            Gdx.app.log("Bot", "Can shoot on this");
            ShootController.shoot(cell.getRow(), cell.getColumn());
            cellsToShoot.remove(cell);
            if (cell.isMissed()) {
                Gdx.app.log("Bot", "Missed");
                changeDirection(direction.get());
            } else {
                if (cell.isInjured()){
                    Gdx.app.log("Bot", "Injured. Adding cell to hitted");
                    hittedCells.add(cell);
                }
                else{
                    Gdx.app.log("Bot", "Killed");
                    resetBot();
                }
            }
        }

        private void changeDirection(Direction current) {
            removeAllExtraCells();
            Gdx.app.log("Bot", "Changing direction. isChanged: " +
    isDirectionChanged);
            if (isDirectionChanged) {
                changeDirectionToCorner(current);
                Gdx.app.log("Bot", "Changed corner to " + direction.get());
            } else {
                changeDirectionToOpposite(current);
                Gdx.app.log("Bot", "Changed oposite to " + direction.get());
            }
            isDirectionChanged = !isDirectionChanged;
        }
        private void removeAllExtraCells() {
            Cell firstCell = hittedCells.get(0);
            hittedCells = new ArrayList<>();
            hittedCells.add(firstCell);
        }


        private void changeDirectionToOpposite(Direction current) {
            switch (current) {
                case TOP -> direction = Optional.of(Direction.BOTTOM);
                case BOTTOM -> direction = Optional.of(Direction.TOP);
                case LEFT -> direction = Optional.of(Direction.RIGHT);
                default -> direction = Optional.of(Direction.LEFT);
            }
        }

        private void changeDirectionToCorner(Direction current) {
            switch (current) {
                case TOP -> direction = Optional.of(Direction.RIGHT);
                case BOTTOM -> direction = Optional.of(Direction.LEFT);
                case LEFT -> direction = Optional.of(Direction.TOP);
                default -> direction = Optional.of(Direction.BOTTOM);
            }
        }
        private void resetBot() {
```

```
        Gdx.app.log("Bot", "ResetBot");
        direction = Optional.empty();
        hittedCells = new ArrayList<>();
        isDirectionChanged = false;
    }
    private void doRandomShot() {
        Cell cell = getRandomCellToShoot();
        Gdx.app.log("Bot", "Random shoot to cell " + cell);
        ShootController.shoot(cell.getRow(), cell.getColumn());
        if (cell.isInjured()) {
            Gdx.app.log("Bot", "Injured");
            hittedCells.add(cell);
            cellsToShoot.remove(cell);
            setRandomDirection();
        } else {
            Gdx.app.log("Bot", "Miss or killed");
            resetBot();
        }
    }
    private Cell getRandomCellToShoot() {
        Cell cell = getRandomCell();
        while (!(cell.isSea() || cell.isHealthy())) {
            cell = getRandomCell();
        }
        return cell;
    }

    private Cell getRandomCell() {
        return cellsToShoot.get(rd.nextInt(cellsToShoot.size()));
    }
    private void setRandomDirection() {
        Direction[] directions = Direction.values();
        direction = Optional.of(directions[rd.nextInt(directions.length)]);
    }




}


/* Nikita Vashkulatov(C) 2022 */
package com.r3nny.seabatlle.client.core.actors;

public enum ShipType {
    ONE_DECK(1),
    TWO_DECK(2),
    THREE_DECK(3),
    FOUR_DECK(4);
    private final int size;

    ShipType(int size) {
        this.size = size;
    }

    public int getSize() {
        return size;
    }
}

/* Nikita Vashkulatov(C) 2022 */
package com.r3nny.seabatlle.client.core.actors;
```

```
import com.badlogic.gdx.Gdx;
import com.badlogic.gdx.graphics.Color;
import com.badlogic.gdx.graphics.GL30;
import com.badlogic.gdx.graphics.g2d.Batch;
import com.badlogic.gdx.graphics.glutils.ShapeRenderer;
import com.badlogic.gdx.scenes.scene2d.Group;
import com.badlogic.gdx.scenes.scene2d.ui.Label;
import com.r3nny.seabatlle.client.core.StarBattle;

public class ShipsCreatingArea extends Group {

    private final ShapeRenderer shape;
    Label controlsHelp;
    Label fourDeskLabel;
    Label treeDeskLabel;
    Label twoDeskLabel;
    Label oneDeskLabel;

    public ShipsCreatingArea(float x, float y) {
        super.setPosition(x, StarBattle.WORLD_HEIGHT - y + 10);
        Label.LabelStyle skin = new Label.LabelStyle();

        skin.font = StarBattle.assetsManager.getFont(52);
        skin.fontColor = new Color(Color.WHITE);
        controlsHelp =
                new Label(
                        "Press A to automatically place ships\n\nDrag`n`Drop
ship to place it. \n\nPress RMB to rotate ship",
                        skin);
        controlsHelp.setFontScale(0.25F);
        controlsHelp.setPosition(x + 10, getY() + 130);
        fourDeskLabel = new Label("Super Destroyers: ", skin);
        fourDeskLabel.setFontScale(0.25F);
        fourDeskLabel.setPosition(controlsHelp.getX(), controlsHelp.getY() +
50);
        treeDeskLabel = new Label("Destroyers: ", skin);
        treeDeskLabel.setFontScale(0.25F);
        treeDeskLabel.setPosition(fourDeskLabel.getX(), fourDeskLabel.getY()
- 60);
        twoDeskLabel = new Label("Corvets: ", skin);
        twoDeskLabel.setFontScale(0.25F);
        twoDeskLabel.setPosition(treeDeskLabel.getX(), treeDeskLabel.getY()
- 60);
        oneDeskLabel = new Label("Default: ", skin);
        oneDeskLabel.setFontScale(0.25F);
        oneDeskLabel.setPosition(twoDeskLabel.getX(), twoDeskLabel.getY() -
60);

        this.shape = new ShapeRenderer();
    }

    @Override
    public void draw(Batch batch, float parentAlpha) {
        super.draw(batch, parentAlpha);
        batch.end();
        Gdx.gl.glEnable(GL30.GL_BLEND);
        Gdx.gl.glBlendFunc(GL30.GL_SRC_ALPHA, GL30.GL_ONE_MINUS_SRC_ALPHA);
        shape.setProjectionMatrix(batch.getProjectionMatrix());
        shape.begin(ShapeRenderer.ShapeType.Filled);
        shape.setColor(0, 0, 0, 0.7F);
        shape.rect(getX(), getY() - 20, Cell.SIZE * 10, Cell.SIZE * 10 +
10);
        shape.end();
        shape.begin(ShapeRenderer.ShapeType.Line);
        shape.setColor(Color.WHITE);
        shape.rect(getX(), getY() - 20, Cell.SIZE * 10, Cell.SIZE * 10 +
```

```
10);
        shape.end();
        Gdx.gl.glDisable(GL30.GL_BLEND);
        batch.begin();
        controlsHelp.draw(batch, parentAlpha);
        fourDeskLabel.draw(batch, parentAlpha);
        treeDeskLabel.draw(batch, parentAlpha);
        twoDeskLabel.draw(batch, parentAlpha);
        oneDeskLabel.draw(batch, parentAlpha);
    }

    @Override
    public void act(float delta) {
        super.act(delta);
    }
}
```

```
/* Nikita Vashkulatov(C) 2022 */
package com.r3nny.seabatlle.client.core.actors;

import com.badlogic.gdx.graphics.g2d.Batch;
import com.badlogic.gdx.graphics.g2d.BitmapFont;
import com.badlogic.gdx.graphics.g2d.Sprite;
import com.badlogic.gdx.scenes.scene2d.Actor;
import com.r3nny.seabatlle.client.core.StarBattle;

/**Описывает тип корабля и количество оставшихся*/
public class ShipsCountDTO extends Actor {

    private final ShipType type;
    private final Sprite texture;
    private int count;
    BitmapFont font;

    public ShipsCountDTO(float x, float y, ShipType type, int count) {
        super.setX(x);
        super.setY(y);
        this.type = type;
        this.count = count;
        switch (type) {
            case ONE_DECK -> texture = new
Sprite(StarBattle.assetsManager.getOneDeckShip());
            case TWO_DECK -> texture = new
Sprite(StarBattle.assetsManager.getTwoDeckShip());
            case THREE_DECK -> texture = new
Sprite(StarBattle.assetsManager.getThreeDeckShip());
            default -> texture = new
Sprite(StarBattle.assetsManager.getFourDeckShip());
        }
        texture.setSize(Cell.SIZE / 1.3F * type.getSize(), Cell.SIZE /
1.3F);
        texture.setX(getX());
        texture.setY(getY());
        font = StarBattle.assetsManager.getFont(20);
    }

    @Override
    public void draw(Batch batch, float parentAlpha) {
        super.draw(batch, parentAlpha);
        texture.draw(batch);
        font.draw(
                batch, String.valueOf(count), getX() - 20, getY() +
(texture.getHeight() * 0.75F));
    }

    public ShipType getType() {
```

```
        return type;
    }

    public void dec() {
        count--;
    }

    public int getCount() {
        return count;
    }
}

/* Nikita Vashkulatov(C) 2022 */
package com.r3nny.seabatlle.client.core.actors;

import com.badlogic.gdx.scenes.scene2d.Group;

import java.util.HashSet;
import java.util.Set;

/**Учет количества кораблей и отображение количества*/
public class ShipManager extends Group {

    private final Set<ShipsCountDTO> shipCounts;

    public ShipManager(float x, float y) {
        shipCounts = new HashSet<>();
        shipCounts.add(new ShipsCountDTO(x, y, ShipType.FOUR_DECK, 1));
        shipCounts.add(
                new ShipsCountDTO(
                        x + ShipType.FOUR_DECK.getSize() * Cell.SIZE + 10,
                        y,
                        ShipType.THREE_DECK,
                        2));
        shipCounts.add(new ShipsCountDTO(x, y - Cell.SIZE,
ShipType.TWO_DECK, 3));
        shipCounts.add(
                new ShipsCountDTO(
                        x + ShipType.FOUR_DECK.getSize() * Cell.SIZE + 10,
                        y - Cell.SIZE,
                        ShipType.ONE_DECK,
                        4));

        for (ShipsCountDTO shipType : shipCounts) {
            super.addActor(shipType);
        }
    }

    public void decByType(ShipType type) {
        shipCounts.stream().filter(s -> s.getType() ==
type).findFirst().ifPresent(ShipsCountDTO::dec);

    }

    public boolean isAllShipsKilled(){
        return shipCounts.stream().filter(c->
c.getCount()!=0).toList().isEmpty();
    }
}

/* Nikita Vashkulatov(C) 2022 */
package com.r3nny.seabatlle.client.core.actors;

import static
com.r3nny.seabatlle.client.core.controller.ShipsCreator.isAnyShipLanding;
```

```
import com.badlogic.gdx.Gdx;
import com.badlogic.gdx.Input;
import com.badlogic.gdx.graphics.Color;
import com.badlogic.gdx.graphics.g2d.Batch;
import com.badlogic.gdx.graphics.g2d.Sprite;
import com.badlogic.gdx.graphics.glutils.ShapeRenderer;
import com.badlogic.gdx.scenes.scene2d.Actor;
import com.badlogic.gdx.scenes.scene2d.InputEvent;
import com.badlogic.gdx.scenes.scene2d.InputListener;
import com.r3nny.seabatlle.client.core.exeptions.CantCreateShipException;
import com.r3nny.seabatlle.client.core.game.Game;
import com.r3nny.seabatlle.client.core.StarBattle;
import com.r3nny.seabatlle.client.core.controller.ShipsCreator;
import java.util.Collections;
import java.util.List;
import java.util.Optional;

/**Корабль*/
public class Ship extends Actor {
    private final ShipType type;
    private List<Cell> cells;

    /**Координата для окна создания корабля*/
    private float startX;
    /**Координата для окна создания корабля*/
    private float startY;
    private boolean isVertical;
    private boolean isSelected;
    private final ShapeRenderer shape;
    private final Sprite sprite;

    /**Обработчик нажатия на корабль*/
    private class ShipInputListener extends InputListener {

        @Override
        public void exit(InputEvent event, float x, float y, int pointer,
Actor toActor) {
            unSelect();
        }

        @Override
        public void enter(InputEvent event, float x, float y, int pointer,
Actor fromActor) {
            if (canMoveShip()) {
                select();
            }
        }

        @Override
        public boolean touchDown(InputEvent event, float x, float y, int
pointer, int button) {
            if (canMoveShip()) {
                if (Gdx.input.isButtonJustPressed(Input.Buttons.RIGHT)) {
                    rotate();
                    StarBattle.soundManager.playClickSound();
                }
            }
            return true;
        }

        @Override
        public void touchDragged(InputEvent event, float x, float y, int
pointer) {
            if (Gdx.input.isButtonPressed(Input.Buttons.LEFT)) {
                if (canMoveShip()) {
                    updatePosition(event.getStageX() - 10, event.getStageY()
```

```
- 10);
                }
            }
        }

        @Override
        public void touchUp(InputEvent event, float x, float y, int pointer,
int button) {
            if (canMoveShip()) {
                Optional<Cell> optionalCell = getCellFromEvent(event);
                if (optionalCell.isPresent()) {
                    Cell cell = optionalCell.get();
                    try {
                        ShipsCreator.addShipToGameField(cell, Ship.this,
Game.player);
                        updatePosition(cell.getX(), cell.getY());
                        animateShip();
                        ShipsCreator.createdPlayerShips++;
                        Game.player.setShipsReady(
                                ShipsCreator.createdPlayerShips
                                        == ShipsCreator.shipTypes.length);
                    }
                    catch (CantCreateShipException ex){
                        resetCoordinates();
                    }
                } else {
                    resetCoordinates();
                }
            }
        }

        private void animateShip() {
            isAnyShipLanding = true;
            Ship.this.addAction(
                    StarBattle.animationManager.getShipEnterAction(
                            Ship.this,
                            () -> isAnyShipLanding = false));
            StarBattle.soundManager.playShipEnterSound();
            updateBounds();
        }



        /**@return Клетка на которую был брошен корабль*/
        private Optional<Cell> getCellFromEvent(InputEvent event) {
            float x = event.getStageX() - 5;
            float y = event.getStageY() - 5;
            Cell[][] field = Game.player.getField();
            for (Cell[] cells : field) {
                for (int j = 0; j < field.length; j++) {
                    Cell currentCell = cells[j];
                    float startX = cells[j].getX();
                    float startY = cells[j].getY();
                    float endX = startX + cells[j].getWidth();
                    float endY = startY + cells[j].getHeight();
                    // TODO: Contains написать
                    if ((x > startX) && (x < endX) && (y > startY) && (y <
endY)) {
                        return Optional.of(currentCell);
                    }
                }
            }
            return Optional.empty();
        }
    }
```

```
    public Ship(float x, float y, ShipType type) {
        shape = new ShapeRenderer();
        this.cells = Collections.emptyList();
        this.type = type;
        super.setX(x);
        super.setY(y);
        this.sprite = initSprite();
        updateBounds();
        this.addListener(new ShipInputListener());
    }

    public boolean canMoveShip() {
        return isShipNotPlaced() && !isAnyShipLanding;
    }
    private boolean isShipNotPlaced() {
        return this.cells.isEmpty();
    }

    private void updatePosition(float v, float v1) {
        setPosition(v, v1);
        updateBounds();
    }

    private void resetCoordinates() {
        setPosition(getStartX(), getStartY());
    }



    private void select() {
        Ship.this.isSelected = true;
        StarBattle.soundManager.playFocusButton();
    }

    private void unSelect() {
        Ship.this.isSelected = false;
    }

    private Sprite initSprite() {
        Sprite texture;
        switch (type) {
            case ONE_DECK -> texture = new
Sprite(StarBattle.assetsManager.getOneDeckShip());
            case TWO_DECK -> texture = new
Sprite(StarBattle.assetsManager.getTwoDeckShip());
            case THREE_DECK -> texture = new
Sprite(StarBattle.assetsManager.getThreeDeckShip());
            default -> texture = new
Sprite(StarBattle.assetsManager.getFourDeckShip());
        }
        texture.setSize(Cell.SIZE * type.getSize(), Cell.SIZE);
        return texture;
    }

    @Override
    public void act(float delta) {
        super.act(delta);
    }

    private void updateBounds() {
        if (isVertical) {
            this.setBounds(getX(), getY(), Cell.SIZE, type.getSize() *
Cell.SIZE);
        } else {
            this.setBounds(getX(), getY(), type.getSize() * Cell.SIZE,
Cell.SIZE);
```

```
        }
    }

    public void makeCellsKilled() {
        for (Cell c : cells) {
            c.setKilled();
        }
    }

    @Override
    public void draw(Batch batch, float parentAlpha) {
        super.draw(batch, parentAlpha);
        if (isSelected)
            drawBounds(batch);
        drawShip(batch, parentAlpha);

    }



    private void drawBounds(Batch batch) {
        batch.end();
        shape.setProjectionMatrix(batch.getProjectionMatrix());
        shape.begin(ShapeRenderer.ShapeType.Line);
        shape.setColor(Color.WHITE);
        if (isVertical)
            shape.rect(getX(), getY(), Cell.SIZE, type.getSize() *
Cell.SIZE);
        else
            shape.rect(getX(), getY(), type.getSize() * Cell.SIZE,
Cell.SIZE);
        shape.end();
        batch.begin();
    }

    private void drawShip(Batch batch, float parentAlpha) {
        Color color = getColor();
        sprite.setColor(color.r, color.g, color.b, color.a * parentAlpha);
        sprite.setPosition(getX(), getY());
        sprite.draw(batch);
        sprite.setColor(color.r, color.g, color.b, 1f);
    }

    @Override
    public void drawDebug(ShapeRenderer shapes) {
        shapes.setColor(Color.RED);
        if (isVertical) {
            shapes.rect(getX(), getY(), Cell.SIZE, type.getSize() *
Cell.SIZE);

        } else {
            shapes.rect(getX(), getY(), type.getSize() * Cell.SIZE,
Cell.SIZE);
        }
    }

    public float getStartX() {
        return startX;
    }

    public void setStartX(float startX) {
        setX(startX);
        this.startX = startX;
    }

    public boolean isVertical() {
```

```
        return isVertical;
    }


    /**Поворачивает корабль на 90 градусов*/
    public void rotate() {
        boolean rotation = !isVertical();
        sprite.rotate90(rotation);
        isVertical = rotation;
        updateBounds();
        sprite.setSize(super.getWidth(), super.getHeight());
    }

    public float getStartY() {
        return startY;
    }

    public void setStartY(float startY) {
        setY(startY);
        this.startY = startY;
    }

    public ShipType getType() {
        return type;
    }

    public List<Cell> getCells() {
        return cells;
    }

    public void setCells(List<Cell> cells) {
        this.cells = cells;
    }

}

/* Nikita Vashkulatov(C) 2022 */
package com.r3nny.seabatlle.client.core.actors;

import com.r3nny.seabatlle.client.core.game.Game;
import com.r3nny.seabatlle.client.core.StarBattle;
import com.r3nny.seabatlle.client.core.controller.ShipsCreator;
import java.util.List;

/**Игровое поле игрока*/
public class PlayerGameField extends GameField {

    private final ShipsCreatingArea shipsCreatingArea;

    public PlayerGameField(float x, float y) {
        super(x, y);
        this.shipsCreatingArea = new ShipsCreatingArea(Game.ENEMY_FIELD_X,
Game.FIELD_Y);
        placeShipsOnCreatingArea();
        addShipsToGroup();
    }

    private void placeShipsOnCreatingArea() {
        placeByShipType(ShipType.FOUR_DECK,
shipsCreatingArea.fourDeskLabel.getY());
        placeByShipType(ShipType.THREE_DECK,
shipsCreatingArea.treeDeskLabel.getY());
        placeByShipType(ShipType.TWO_DECK,
shipsCreatingArea.twoDeskLabel.getY());
        placeByShipType(ShipType.ONE_DECK,
shipsCreatingArea.oneDeskLabel.getY());
```

```
    }

    private void placeByShipType(ShipType type, float startY) {
        float startX = shipsCreatingArea.controlsHelp.getX();
        List<Ship> shipsByType = ships.stream().filter(s -> s.getType() ==
type).toList();
        for (int i = 0; i < shipsByType.size(); i++) {
            shipsByType.get(i).setStartX(startX + (type.getSize() *
Cell.SIZE + 20) * i);
            shipsByType.get(i).setStartY(startY - 12);
        }
    }

    private void addShipsToGroup() {
        for (Ship ship : this.ships) {
            super.addActor(ship);
        }
    }

    @Override
    public void createShipsAutomaticaly() {
        super.createShipsAutomaticaly();
        animateShipsCreating();
    }

    private void animateShipsCreating() {
        ShipsCreator.isAnyShipLanding = true;
        for (Ship ship : ships) {
            ship.addAction(
                    StarBattle.animationManager.getShipEnterAction(
                            ship,
                            () -> ShipsCreator.isAnyShipLanding = false));
        }
    }


}


/* Nikita Vashkulatov(C) 2022 */
package com.r3nny.seabatlle.client.core.actors;

import com.badlogic.gdx.scenes.scene2d.Group;
import com.r3nny.seabatlle.client.core.StarBattle;
import com.r3nny.seabatlle.client.core.controller.ShipsCreator;

import java.util.LinkedList;
import java.util.List;

/**Игровое поле*/
public abstract class GameField extends Group {
    public static final int FIELD_SIZE = 10;
    private final float x;
    private final float y;
    private ShipManager shipManager;
    private boolean isShipsReady;
    private final Cell[][] field;
    protected final List<Ship> ships;

    public GameField(float x, float y) {
        this.x = x;
        this.y = y;
        field = initCells();
        ships = initShips();
    }
```

```
    private Cell[][] initCells() {
        Cell[][] field = new Cell[FIELD_SIZE][FIELD_SIZE];
        for (int i = 0; i < field.length; i++) {
            for (int j = 0; j < field.length; j++) {
                field[i][j] = new Cell(x + Cell.SIZE * j, y - Cell.SIZE * i,
j, i);
                super.addActor(field[i][j]);
            }
        }
        return field;
    }

    private List<Ship> initShips() {
        return createShipsAllTypes(ShipsCreator.shipTypes);
    }

    private List<Ship> createShipsAllTypes(ShipType[] shipTypes) {
        List<Ship> ships = new LinkedList<>();
        for (int i = 0; i < shipTypes.length; i++) {
            Ship ship = new Ship(0, 0, ShipsCreator.shipTypes[i]);
            ships.add(ship);
        }
        return ships;
    }

    public void createShipsAutomaticaly() {
        StarBattle.soundManager.playShipEnterSound();
        clearField();
        ShipsCreator.autoCreateShips(this);
        clearAllMissed();
    }

    private void clearField() {
        for (Cell[] cells : this.field) {
            for (int j = 0; j < field.length; j++) {
                cells[j].setSea();
            }
        }
    }

    public void killShip(Ship ship) {
        ship.makeCellsKilled();
        decByShipType(ship.getType());
        ShipsCreator.setCellsAroundShipMissed(ship, this.getField());
        removeActor(ship);
    }

    public void clearAllMissed() {
        for (Cell[] cells : field) {
            for (int j = 0; j < field.length; j++) {
                if (cells[j].isMissed()) {
                    cells[j].setSea();
                }
            }
        }
    }

    public void createShipsManager() {
        shipManager = new ShipManager(x, y - (Cell.SIZE * field.length) -
40);
        super.addActor(shipManager);
    }

    private void decByShipType(ShipType type) {
        shipManager.decByType(type);
```

```
    }

    public boolean isShipsReady() {
        return isShipsReady;
    }

    public void setShipsReady(boolean shipsReady) {
        isShipsReady = shipsReady;
    }

    public Cell[][] getField() {
        return field;
    }

    public List<Ship> getShips() {
        return ships;
    }

    public boolean isAllShipsKilled(){
        return shipManager.isAllShipsKilled();
    }

    public ShipManager getShipManager() {
        return shipManager;
    }
}

/* Nikita Vashkulatov(C) 2022 */
package com.r3nny.seabatlle.client.core.actors;

/**Игровое поле противника*/
public class EnemyGameField extends GameField {
    public EnemyGameField(float x, float y) {
        super(x, y);
        createShipsAutomaticaly();
        super.setShipsReady(true);
    }
}

/* Nikita Vashkulatov(C) 2022 */
package com.r3nny.seabatlle.client.core.actors;

import com.badlogic.gdx.Gdx;
import com.badlogic.gdx.graphics.Color;
import com.badlogic.gdx.graphics.g2d.Animation;
import com.badlogic.gdx.graphics.g2d.Batch;
import com.badlogic.gdx.graphics.g2d.TextureRegion;
import com.badlogic.gdx.graphics.glutils.ShapeRenderer;
import com.badlogic.gdx.scenes.scene2d.Actor;
import com.badlogic.gdx.scenes.scene2d.InputEvent;
import com.badlogic.gdx.scenes.scene2d.InputListener;
import com.r3nny.seabatlle.client.core.game.Game;
import com.r3nny.seabatlle.client.core.game.GameStatus;
import com.r3nny.seabatlle.client.core.StarBattle;
import com.r3nny.seabatlle.client.core.controller.ShootController;

enum CellStatus {
    SEA,
    MISS,
    HEALTHY,
    INJURED,
    KILLED
}

/**Клетка*/
public class Cell extends Actor {
```

```
    public static final float SIZE = 31.37f;

    private final int column;
    private final int row;
    private Ship ship;
    private CellStatus status;
    private final ShapeRenderer shape;
    private float animationTime = 0f;

    private final Animation<TextureRegion> injuredAnimation;
    private final Animation<TextureRegion> burningAnimation;
    private final Animation<TextureRegion> missAnimation;
    private final Animation<TextureRegion> explosionAnimation;

    public Cell(float x, float y, int column, int row) {
        this.column = column;
        this.row = row;
        this.ship = null;
        super.setX(x);
        super.setY(y);
        this.injuredAnimation =
StarBattle.animationManager.getInjuredAnimation();
        this.burningAnimation =
StarBattle.animationManager.getBurningAnimation();
        this.missAnimation = StarBattle.animationManager.getMissAnimation();
        this.explosionAnimation =
StarBattle.animationManager.getExplosionAnimation();
        this.status = CellStatus.SEA;
        shape = new ShapeRenderer();
        shape.setAutoShapeType(true);
        this.setBounds(x, y, SIZE, SIZE);

        this.addListener(
                new InputListener() {
                    public boolean touchDown(
                            InputEvent event, float x, float y, int pointer,
int button) {
                        if (Game.status == GameStatus.PLAYER_TURN) {
                            ShootController.shoot(Cell.this.row,
Cell.this.column);
                        }
                        return true;
                    }
                });
    }

    @Override
    public void draw(Batch batch, float parentAlpha) {
        animationTime += Gdx.graphics.getDeltaTime();
        drawBorders(batch);
        setUpBatch(batch, parentAlpha);
        drawInjured(batch);
        drawKilled(batch);
        drawMissed(batch);
        tearDownBatch(batch);
    }

    private void drawBorders(Batch batch) {
        setUpShapeRenderer(batch);
        shape.rect(getX(), getY(), SIZE, SIZE);
        tearDownShapeRenderer();
    }

    private void setUpShapeRenderer(Batch batch) {
        batch.end();
```

```
        shape.setProjectionMatrix(batch.getProjectionMatrix());
        shape.begin();
        Gdx.gl20.glLineWidth(3);
        shape.setColor(Color.WHITE);
    }

    private void tearDownShapeRenderer() {
        shape.end();
    }

    private void setUpBatch(Batch batch, float parentAlpha) {
        batch.begin();
        Color color = getColor();
        batch.setColor(color.r, color.g, color.b, color.a * parentAlpha);
    }

    private void drawInjured(Batch batch) {
        if (isInjured()) {
            if (!injuredAnimation.isAnimationFinished(animationTime))
                batch.draw(getCurrentFrame(injuredAnimation), getX(),
getY(), Cell.SIZE, Cell.SIZE);
            else
                batch.draw(
                        getCurrentFrame(burningAnimation, animationTime,
true),
                        getX(),
                        getY(),
                        Cell.SIZE,
                        Cell.SIZE);
        }
    }

    private TextureRegion getCurrentFrame(Animation<TextureRegion>
animation) {
        return getCurrentFrame(animation, animationTime, false);
    }



    private TextureRegion getCurrentFrame(Animation<TextureRegion>
animation, float time, boolean looping) {
        return animation.getKeyFrame(time, looping);
    }

    private void drawKilled(Batch batch) {
        if (isKilled())
            batch.draw(getCurrentFrame(explosionAnimation), getX(), getY(),
Cell.SIZE, Cell.SIZE);
    }

    private void drawMissed(Batch batch) {
        if (isMissed()) {
            batch.draw(
                    getCurrentFrame(missAnimation,
missAnimation.getFrameDuration() * 3, false),
                    getX(),
                    getY(),
                    Cell.SIZE,
                    Cell.SIZE);
            if (!missAnimation.isAnimationFinished(animationTime)) {
                batch.draw(getCurrentFrame(missAnimation), getX(), getY(),
Cell.SIZE, Cell.SIZE);
            }
        }
    }
```

```
    private void tearDownBatch(Batch batch) {
        Color color = getColor();
        batch.setColor(color.r, color.g, color.b, 1f);
    }

    @Override
    public void drawDebug(ShapeRenderer shapes) {
        if (isHealthy()) {
            shapes.setColor(Color.GREEN);
            shapes.circle(getX() + SIZE / 2, getY() + SIZE / 2, SIZE / 2 -
3);
        }
    }

    public boolean isMissed() {
        return status == CellStatus.MISS;
    }

    public boolean isHealthy() {
        return status == CellStatus.HEALTHY;
    }

    public boolean isInjured() {
        return status == CellStatus.INJURED;
    }

    public boolean isSea() {
        return status == CellStatus.SEA;
    }

    public boolean isKilled() {
        return status == CellStatus.KILLED;
    }

    public void setSea() {
        this.status = CellStatus.SEA;
    }

    public void setMiss() {
        animationTime = 0;
        this.status = CellStatus.MISS;
    }

    public void setHealthy() {
        this.status = CellStatus.HEALTHY;
    }

    public void setInjured() {
        animationTime = 0;
        this.status = CellStatus.INJURED;
    }

    public void setKilled() {
        animationTime = 0;
        this.status = CellStatus.KILLED;
    }

    public Ship getShip() {
        return ship;
    }

    public int getColumn() {
        return column;
    }

    public int getRow() {
```

```
        return row;
    }

    public void setShip(Ship ship) {
        this.ship = ship;
    }

    @Override
    public String toString() {
        return "Cell{" +
                "column=" + column +
                ", row=" + row +
                '}';
    }
}
```