Design Document

Авторы проекта: Георгий Ангени, Даниил Любаев, Денис Коротченко

Общие сведения о системе

Проект представляет из себя игру в жанре Roguelike. Игра доступна в однопользовательском режиме, однако помимо персонажа, управляемым игроком, в игре также представлены иные персонажи. Персонаж, управляемый игроком представляет из себя героя с заданными начальными характеристиками (например, здоровье). В процессе игры персонаж может взаимодействовать с иными персонажами или же предметами, тем самыми изменяя свои характеристики. Снижение характеристики здоровья до нуля в большинстве случаев приводит к поражению игрока.
Управление персонажем происходит в пошаговом режиме путём нажатия кнопок на клавиатуре. При взаимодействии с предметами, персонаж имеет возможность добавить их в свой инвентарь для дальнейшего использования.
Основной особенностью игр в жанре Roguelike является невозможность восстановления после потери всех единиц здоровья и необходимость проходить уровень сначала в таком случае.

Architectural drivers

Переносимость	Игра должна быть доступна для запуска под различным ОС.
Стабильность	В 99% запусков игры, она не должна приводить к падению в процессе работы.
Производительность	Время реакции на действие пользователя должно составлять не более 100 мс в 90-м перцентиле.
Удобство использования	Система управления должна быть интуитивна понятной и не требовать большого времени на изучение от пользователя.

Расширяемость	Игра должна предоставлять простую возможность к расширению путём добавления или генерации новых уровней, что приведёт к более высокой реиграбельности.

Функциональные требования

1. Возможность взаимодействия с предметами и иными персонажами.
2. Возможность добавления предмета в инвентарь.
3. Возможность управления (и просмотра) предметов, находящихся в инвентаре.
4. Наличие нескольких уровней в игре.
5. Наличие инструкции к игре.

Ограничения

1. Пошаговое управление персонажем игрока.
2. Наличие только однопользовательского режима в игре.
3. Невозможность восстановления после потери всех единиц здоровья.

Роли и случаи использования

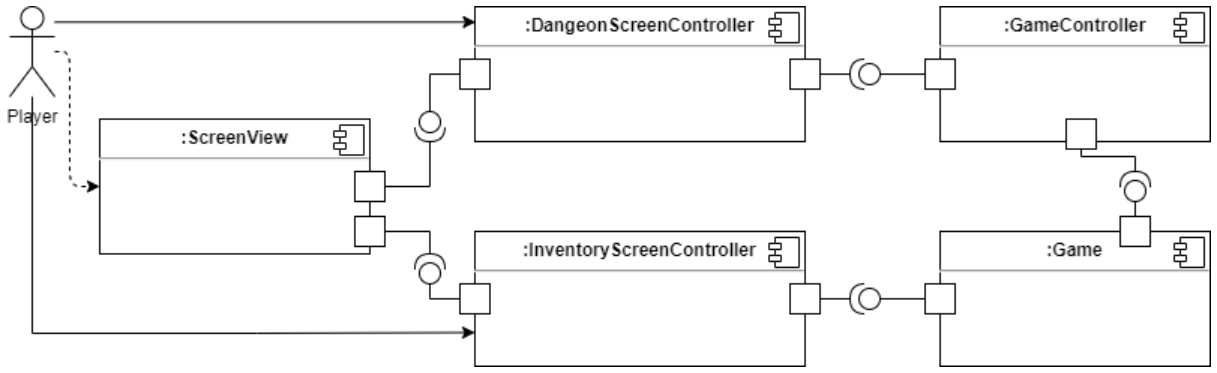
Основные заинтересованные лица в проекте:

- Пользователь (игрок) – при помощи клавиатуры управляет персонажем и играет в игру, проходя различные уровни.
- Разработчики игры – реализуют и поддерживают проект.

Описание типичного пользователя

Пользователем является любой человек старше 16 лет, знающий или имеющий возможность осуществить перевод базовых слов английского языка, использующихся в интерфейсе игры. Опыт игры в иные игры жанры roguelike не важен для освоения игры. Пользователь ожидает получения удовольствия от прохождения игры.

Композиция

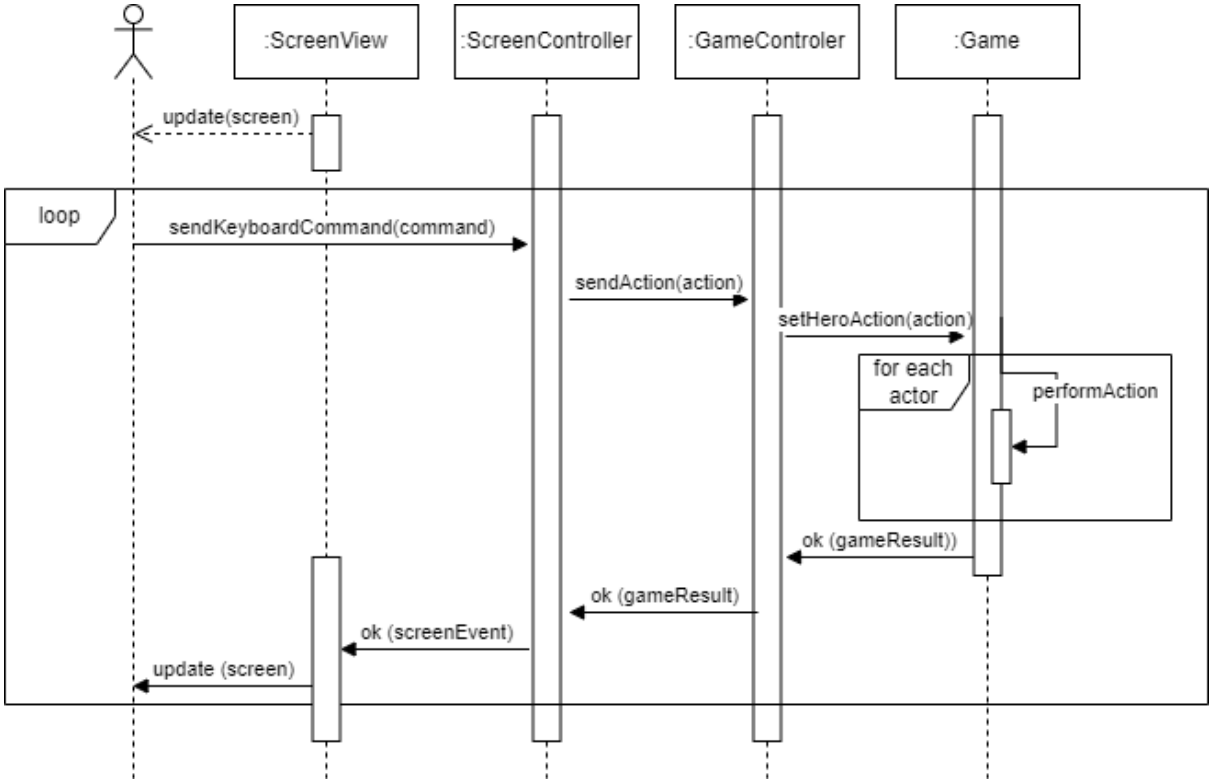


Пользовательские команды обрабатываются одним из контроллеров в зависимости от того, какой из экранов (уровня или инвентаря) сейчас на переднем плане, после чего компонента Game реализует основные механики и модели игры, моделируя взаимодействие персонажей друг с другом и с предметами. В результаты моделирования взаимодействий компонента генерирует суммарные изменения, которые необходимо отобразить пользователю, после чего View отображает новое состояние экрана.

Логическая структура

Основной принцип, используемый в архитектуре проекта – Model-View-Controller.
Диаграмма классов доступна в Miro по ссылке (раздел Roguelike):
https://miro.com/app/board/uXjVMoXRn9w=/?share_link_id=912828473504
Модуль View отвечает за отрисовку игровых экранов. Модуль Controller отвечает за управление игрой. Так, основная функция, обрабатывающая пользовательские команды – функция HandleKey(Key). В зависимости от того, какой экран сейчас находится на переднем плане (с каким экраном взаимодействует пользовать), пользовательская команда обрабатывается тем или иным наследником класса ScreenController. В случае, если действие подразумевает изменение внутреннего состояния игры, вызывается метод ParseInput(Input) у GameController, который в дальнейшем передаёт команду классу Game в модуле Model, которые осуществляет изменения внутреннего состояния всех персонажей и иных объектов игры в зависимости от переданной команды. Результатом изменений в модели является набор событий, которые обрабатываются модулем View и вносят изменения в отрисовывающийся экран.

Взаимодействия и состояния



ScreenController обрабатывает команду пользователя, после чего передаёт Action (в зависимости от команды) GameController, который устанавливает Action персонажу игрока. После этого Game устанавливает Action всем остальным персонажам, после чего выполняет последовательно для каждого из персонажей action, что приводит к изменениям во внутреннем состоянии игры. После обработки все action, Game возвращает GameResult – обобщённый результат обработки действия пользователя, который ScreenController превращает в набор событий для изменения текущего экрана, ScreenView вносит соответствующие изменения и отображает пользователю новое состояние.