ПРОЕКТ финальной работы

(подготовил Ветров Д.Д.)

Основное задание:

1. Цель работы: Разработать игру и собрать этот проект под платформы Windows.

2. Цель игрока: Дойти до конца уровня.

3. Сцены: Проект должен состоять из четырёх сцен

3.1. Сцена главного меню: Из этой сцены пользователь может перейти к прохождению уровня, воспользоваться меню настроек звука или выйти из игры, закрыв приложение.

3.2. Сцена прохождения уровня: Игра состоит из одного уровня. Персонаж игрока всегда появляется в одном и том же месте. Задача игрока — пройти до конца уровня (выхода). За уничтожение врагов игроку начисляются очки. В случае смерти персонажа происходит переход на «Сцену поражения». В случае достижения конца уровня — переход на «Сцену победы».

3.2.1. Сцена победы: Отображается сообщение о победе, информация о набранных очках, и появляется кнопка «Далее», возвращающая в главное меню.

3.2.2. Сцена поражения: Отображается сообщение о поражении, информация о набранных очках, и появляется кнопка «Далее», возвращающая в главное меню.

4. Интерфейс главного меню: Вы можете оформить и организовать элементы интерфейса сцены главного меню на своё усмотрение, исходя из сеттинга игры.

5. Интерфейс игры: Элементы пользовательского интерфейса в режиме игры вы можете оформить и разместить на своему усмотрению.

6. Персонаж игрока

6.1 Передвигается в плоскости при нажатии WASD.

6.2 Стрельба — нажатие левой клавиши мыши.

6.3 За каждого убитого врага игроку начисляются очки.

6.4 При значении количества боеприпасов 0 стрельба невозможна, несколько секунд происходит перезарядка.

7. Враги с дистанционной атакой: Враг, имеющий стрелковое оружие, начинает стрелять в сторону персонажа игрока когда персонаж находится в зоне действия врага. Вы также можете реализовать для этих врагов режим патрулирования.

8. На уровне можно разместить «аптечки», пополняющие здоровья игрока. После того как персонаж коснулся «аптечки», его здоровье восстанавливается на несколько единиц, а сама «аптечка» исчезает.

Старт проекта:

Описание:

Игра представляет из себя примитивный танковый стимулятор, как опытная площадка для тестирования идей и решений (с подготовленными моделями и пространством).

Модели созданы в Blender, с примитивной анимацией. Применен единый материал(альбом) по заранее подготовленным UV развертке.

Аудио ресурсы получены с доступным интерент-ресурсов.

Основной целью при реализации проекта получить практику в самостоятельном решении новых подходах к поставленной задаче. Идеология предполагала большую часть функционала реализовать именно посредством скриптов, а так же максимально исключить применение «тяжелых» встроенных методов, тэгов и аналогичное, так же минимизировать «ручную» настройку через инспектор единичных объектов и исключить связи.

В итоге основной игровой процесс заключается в управление объектом-танк (Player), с возможностью управления WASD (применено присвоение скорости Rigibody), при значении скорости более 0 срабатывает триггер анимации, управление курсором посредством «мыши», автоматическое слежение объекта-башня или объекта-пулемет за курсором, в случае совпадения позиции курсора с позицией активного объекта-Enemy меняется спраит(для индикации). Выстрел производится нажатием левый клавиши «мыши», кнопкой Tab меняется тип стрельбы. Кнопки и «мышь» возможно переназначить в MapInput. Определены два типа врагов: объект-танк и объект-авто, игрок через Tab определяет режим стрельбы «снаряд-Bull», который наносит рандомный урон объекту-танк и объекту-авто, в режиме «пуля-Rif» рандомный урон наносится только объекту-авто. В объектах Enemy реализовано перемещение посредством NavMeshAgent, с автоматическим назначением позиций. Позиции назначаются в массив через инспектор, чем определен маршрут движения объекта, в случае отсутствие назначенных позиций назначается дефолтная позиция (получение дефолтной позиции при активации объекта), при попадании в зону Physics.OverlapSphere коллайдера объекта-Player, назначается позиция для объекта Enemy, в случае выхода из зоны Physics.OverlapSphere, объекту Enemy назначается дефолтная позиция или предустановленная позиция из массива. Аналогично реализована слежение объектов-орудия Enemy. В случае деактивации объекта Enemy, активирует объект-лут (Healt) из класса пула-объектов. В функционально основных скриптах(классов) допустимо изменение параметров (или назначение ресурсов) через назначенный ScriptableObject. Объекты-пули(снаряды) активируются через пул-объектов, цикл активности определен попаданием в коллайдер объекта-цель или таймером. При попадании в коллайдер объекта-цель, объект-пуля(снаряд) получает хэш объекта-цели, так же передает значение в скрипт(класс) Healt объекта-цели, и возвращает данные об объекте-цели, что позволяет производить статистику.

Ознакомился информацией из разных источников (книги: ГауравА.,Джеффри Ч. ”Паттерны проектирования для С#”, Ларкович С., Евдокимов П. “C# для Unity” и интернет-ресурсы: [METANIT.COM](https://metanit.com/), [habr.com](https://habr.com/ru/articles/313694/) и аналогичные), было принято решение построить решение с уклоном на независимость объектов, то есть классы описывают определённый функционал объекта, а обмен данными реализован в виде класса событий (по проекту класс EventBus). Для оперативного доступа к активному объекту, в каждый объект внедрен класс-регистратор, с функцией получения всех требуемых компонентов и хэшкода объекта, с записью в экземпляр структуры (по проекту Construction), с последующим сохранением в коллекцию. Остальные классы представляют из себя реализацию определённого функционала.

Сложности: Для понимания общей структуры и планирования иерархии классов (наследование), потребовалось составить блок-схемы реализации какой-либо функции (с абстракцией до уровня метода), данная практика не предполагала в учебном материале, было потрачено время на переделывания работы. Полученный опыт, определил, что проект начинается с блок-схем с понятным техзаданием. Менее сложные вопросы решались посредством книг, интернет-ресурсов (вопрос только в качестве контента). Основная сложность: мало практики.

Удовольствие: в принципе реализация проекта в удовольствие, получен ряд практик такие как использование пул-объектов и mapInput, так же принято решение, данный проект использовать как опытную площадку для разных идей, на перспективный проект с использованием Zenject, DOTween, RigAnimation.

В остальном проект отражен в виде блок-схем.

Блок схемы:

Схема сцены главного меню:

Сцена главного меню, представляет из себя канвас с группой панелей (панели определены определённым функциональным применением). Доступный вызов сцены уровня.

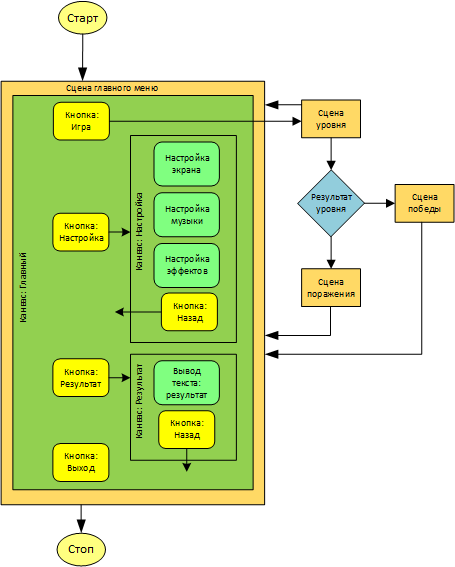


Схема сцены уровня:

Сцена игрового уровня, представляет из себя пространство с независимыми игровыми объектами, доступно менять количество объектов, место положение без корректировки кода, за исключением объекта типа Player. Основной канвас (интерфейс) отражает очки, здоровье, тип зарядов, кнопки вызова панели меню. Вызов сцен победа/поражение по результату уровня здоровья объекта Player или отсутствие активных объектов Enemy.

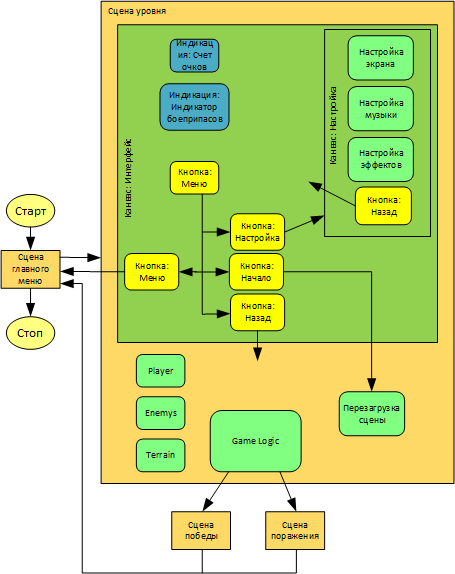
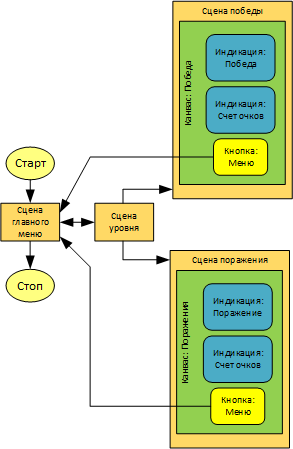


Схема сцен победы и поражения:

Сцены победа/поражение, представляет из себя канвас с текстовым полями, отражающие общую информацию и очки (полученные из сохранённого в классе PlayerPrefs). Кнопка вызова сцены главного меню.



Сцена Главного меню:

Схема классов(скрипты) главного меню:



Сцены победы и поражения:

Схема классов(скрипты) победы и поражения:



Сцена уровня:

Схема классов(скрипты) объекта Camera, Cursor:



Схема классов(скрипты) объекта Canvas:





Схема классов(скрипты) объекта Player:











Схема классов(скрипты) объекта Enemy:





Схема классов(скрипты) генерируемых объектов и учет хешкодов активных объектов:



