**Содержание**

Ход работы 2

V0.0.011 2

V0.0.012 3

V0.0.013 6

V0.0.014 9

V0.0.015 10

V0.0.016 11

Дальнейшие действия в реализации проекте 16

Ход работы

V0.0.011

В этой версии было решено добавить начальный экран и меню паузы с возможностью возобновить, перезапустить или выключить игру.

Для контролирования состояния игры необходимо создать GameManager. Это класс-синглтон. Этот скрипт хранит текущее состояние игры и только он может изменять его, поэтому в нем описаны методы для установки/снятия паузы, загрузки/выгрузки сцен, перезапуска и выхода из игры. При изменении состояния игры вызывается событие OnGameStateChanged. Таким образом игровые объекты могут корректно реагировать на изменение состояния игры. Пока в игре будет 3 состояния: Pregame, Running, Paused. Также этот скрипт размещает другие системные префабы (например, UIManager, который будет описан чуть позже).

При запуске игры GameManager размещает на сцене объект со скриптом UIManager. Этот скрипт будет связывать скрипты различных окон со скриптом GameManager. Также этот скрипт обрабатывает нажатия пользователя.



Рисунок 1. Иерархия объекта UIManager.

После размещения объекта UIManager на сцене пользователь увидит начальный экран.



Рисунок 2. Начальный экран.

Если коснуться экрана в любом месте, UIManager вызовет метод StartGame класса GameManager. Этот метод начнет загрузку сцены первого уровня. После загрузки сцены состояние игры изменится на Running и вызовется соответствующее событие. Скрипт UIManager отреагирует на это событие, сделав видимым игровой интерфейс (джойстики, кнопка смены оружия и т.д.). Также отреагирует и скрипт MainMenu, ответственный за работу с анимациями и открытие/закрытие начального экрана. Этот скрипт начнет проигрывать анимацию затухания начального экрана. После её окончания объект начального меню деактивируется и вызовется событие, сообщающее, что затухание было завершено. Это событие через UIManager отправляется в GameManager, где начинается выгрузка сцены меню. На этом переход от состояния Pregame в состояние Running завершается. Переход от состояния Running в состояние Pregame происходит подобным образом.

В игре при нажатии клавиши “назад” UIManager вызовет метод TogglePause класса GameManager. Состояние сменится с Running на Paused. После цепочки методов откроется меню паузы.



Рисунок 3. Меню паузы.

В меню паузы слева расположен инвентарь игрока. Здесь будут располагаться усиления-предметы игрока. При нажатии на них справа-вверху будет отображаться информация о них. Справа внизу расположены 3 кнопки для возобновления игры, рестарта и закрытия. Кнопка Resume переведет состояние игры из Paused в Running, а кнопка Restart из Paused в Pregame.

Также в этой версии был добавлен отдельный ввод для мыши, теперь с помощью неё можно воспроизводить касания.

V0.0.012

В этой версии было решено добавить нового врага и первые предметы.

Для добавления нового врага был нарисован новый спрайт. Затем создана копия префаба слизня, где был заменен спрайт и изменены характеристики.



Рисунок 4. Спрайт скелета.

Также возникла необходимость добавить отдельный коллайдер для обработки столкновения с пулями. Т.к. столкновение с препятствиями будет происходить лишь небольшой областью около ног, а пули должны попадать в любое место врага. Коллайдер для столкновения с пулями помечен как Trigger, для столкновения с объектами – нет.



Рисунок 5. Коллайдеры скелета.

Для добавления первых предметов был создан скрипт ресурса (наследуемый от ScriptableObject) - Item\_SO. В таких ресурсах хранятся название предмета, тип (предмет или оружие), иконка и информация о том, был ли этот предмет уже получен.

Добавлена платформа для предмета с прикрепленным скриптом ItemPlatform. В методе Start этого скрипта выбирается случайный предмет и размещается на сцене (становится дочерним к платформе). Платформа имеет анимацию, перемещающую дочерние объекты вверх-вниз. Таким образом создается вид, что предмет “летает” над платформой.

Все собираемые предметы будут иметь скрипт ItemPickUp, который хранит ссылку на ресурс с предметом Item\_SO. Этот скрипт имеет метод StoreItemInInventory, который вызывается при подборе предмета и сохраняет его в инвентаре, обращаясь к скрипту CharacterInventory.

Скрипт-синглтон CharacterInventory сохраняет ссылки на полученные предметы.

К меню предметов прикреплен скрипт, который в методе OnEnable (метод срабатывает каждый раз, когда объект появляется на сцене) получает из класса CharacterInventory ссылки на полученные предметы и размещает их в ячейках меню.

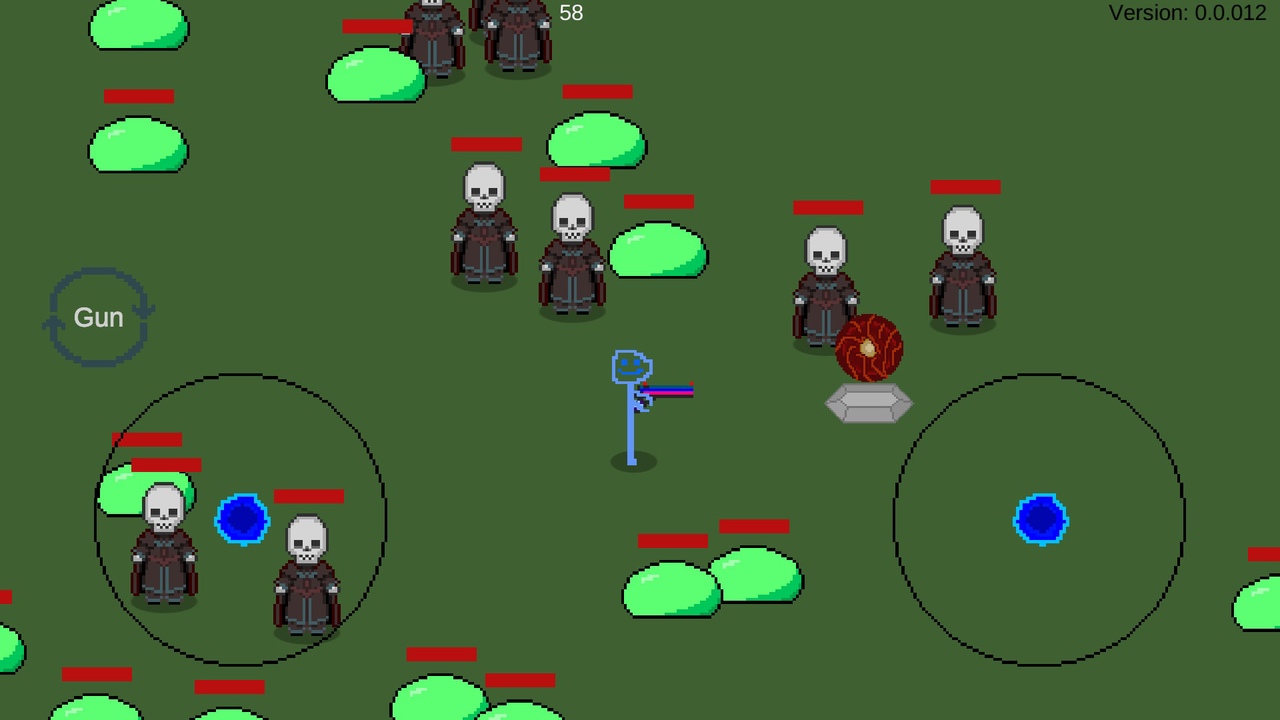


Рисунок 6. Новые монстры и платформа с предметом.

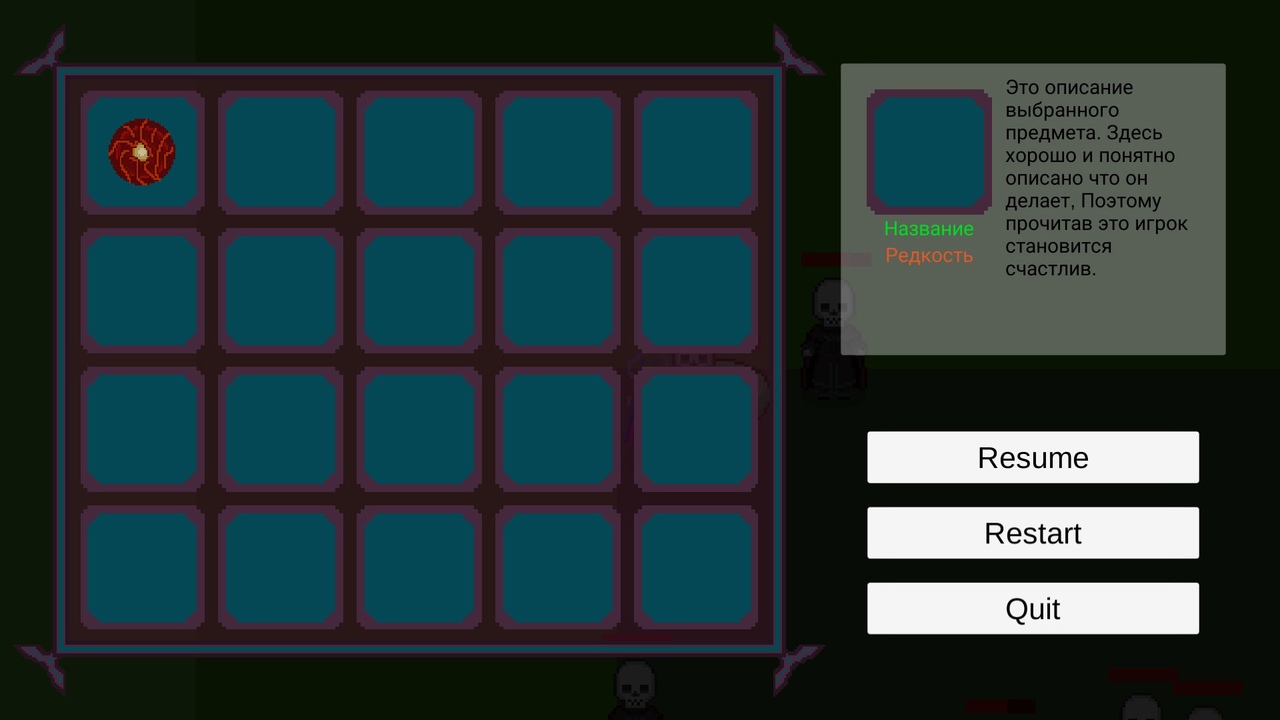


Рисунок 7. Подобранный предмет в меню предметов.

V0.0.013

В этой версии будет множество нововведений:

* Всплывающие цифры урона, цифры здоровья и менеджер, для работы с всплывающим текстом.
* Характеристики всех персонажей будут храниться одинаковым способом.
* Система стрельбы переделана для работы с врагами.
* Врагам добавлена атака – стрельба.
* У предметов будет общий интерфейс.
* Игрок начнет получать урон. При смерти игра будет перезапускаться.

Текст, располагаемый в игровом мире отличается от текста интерфейса. Весь интерфейс располагается на канвасе (холсте). Относительно его размеров задаются размеры и расположения всех элементов пользовательского интерфейса. В игровом же мире подобного холста нет, поэтому текст должен быть объектом. Подобную логику дает компонент TextMeshPro. С помощью этого компонента и будут реализованы все надписи внутри игры.

Для работы с всплывающим текстом был создан класс Pop\_Ups. Он создает пустые объекты, добавляет к ним компонент TextMeshPro, задает им текст, цвет, положение, отвечает за их передвижения. Здесь, как и для пуль, реализован пул объектов.

К полоскам здоровья врагов также был добавлен текст. Изменяет этот текст скрипт EnemyHealthbar.

Для хранения данных о персонажах скрипт PlayerStats\_SO был переименован в CharacterStats\_SO, также теперь в нем больше различных полей: уровень, максимальное здоровье, текущее здоровье, урон, шанс и множитель критического урона, скорость, трение, время перезарядки, скорость и время жизни пуль, номер оружия, радиус атаки (дальность полета снаряда определяется временем его жизни, радиус атаки влияет на расстояние до игрока, про котором враги его заметят), текущее оружие, спрайт и контроллер анимаций.

Был добавлен скрипт CharacterStats, который имеет те же поля, что и CharacterStats\_SO, и имеет ссылку на ресурсный объект CharacterStats\_SO. В методе Awake все данные из ресурсного объекта копируются в поля скрипта CharacterStats. К полям этого класса будут обращаться другие скрипты, которым потребуются данные о персонаже.

Изменение стрельбы было начато с класса Bullet. Здесь были созданы несколько структур, чтобы проще передавать информацию о выстреле между объектами. Структура DamageInfo хранит только ту информацию, которая влияет на урон и имеет 2 поля: damage, isCritical (bool). Структура BulletInfo хранит остальную информацию о пуле: тег персонажа, выпустившего снаряд, урон, скорость, крит шанс и множитель, направление, стартовую позицию, время жизни и обладает ли пуля “сквозным” эффектом. FireManager в методе Fire получает на вход структуру BulletInfo, эти данные передаются объекту Bullet для инициализации значений. Когда пуля сталкивается с объектом обладающем компонентом TakeDamage она обновляет поле isCritical структуры DamageInfo на основе поля CritChance структуры BulletInfo, после чего передает объект структуры DamageInfo в метод Damage класса TakeDamage.

Класс WeaponManager также имеет объект структуры BulletInfo. Его значения задаются в методе Start из объекта CharacterStats. При выстреле обновляются поля стартовой позиции и направления, после чего объект BulletInfo отправляется в метод Fire класса FireManager.

Для добавления атаки врагам был создан скрипт EnemyAttack. По аналогии со скриптом EnemyMove в нем будут описаны разные типы атак, после чего можно будет выбрать один из них. Первой атакой стала стрельба. В игрока выпускается луч (частота отправки зависит от перезарядки врага). Если длина луча меньше, чем радиус атаки, то выполняется выстрел. Для этого в метод Fire класса FireManager отправляется объект структуры BulletInfo со всей информацией о выстреле.

Скрипт TakeDamage был изменен для работы как с врагами, так и с игроком. Так как полоса здоровья игрока будет находиться на канвасе, для изменения её значений добавлен отдельный скрипт PlayerHealthbar, связанный с компонентом TakeDamage, прикрепленным к игроку. Когда здоровье игрока опускается до 0, игра перезапускается.

Врагам добавлены анимации. Для этого используется тот же контроллер анимаций, что и для игрока. Управление анимациями осуществляется в скрипте EnemyMove.

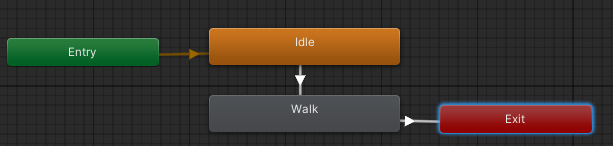


Рисунок 8. Контроллер анимаций для персонажей.

Для более удобной работы с предметами был создан интерфейс ICollectableItem описывающий метод ItemPickUpped. Для каждого предмета будет писаться скрипт, наследуемый от этого интерфейса и описывающий действия при подборе предмета в методе ItemPickUpped. При подборе предмета скрипт ItemPlatform вызывает метод StoreItemInInventory скрипта ItemPickUp, в котором и вызывается интерфейсный метод ItemPickUpped, связанный с предметом. Для первого предмета был написан скрипт BulletPenetration, который изменяет значение поля penetration в объекте структуры BulletInfo в классе WeaponManager на true. Таким образом после подбора предмета игрок начинает стрелять сквозь врагов.

При тестировании игры стало понятно, что из-за маленького обзора, высокой скорости пуль врагов и их большой дальности атаки от вражеских снарядов невозможно увернуться. Было необходимо либо замедлить скорость полета пуль и уменьшить всем персонажам дальность атаки, либо увеличить видимую область. Уменьшение скорости полета пуль и уменьшение радиуса атаки сделало бы игру на много менее динамичной, из-за чего было решено отдалить камеру.

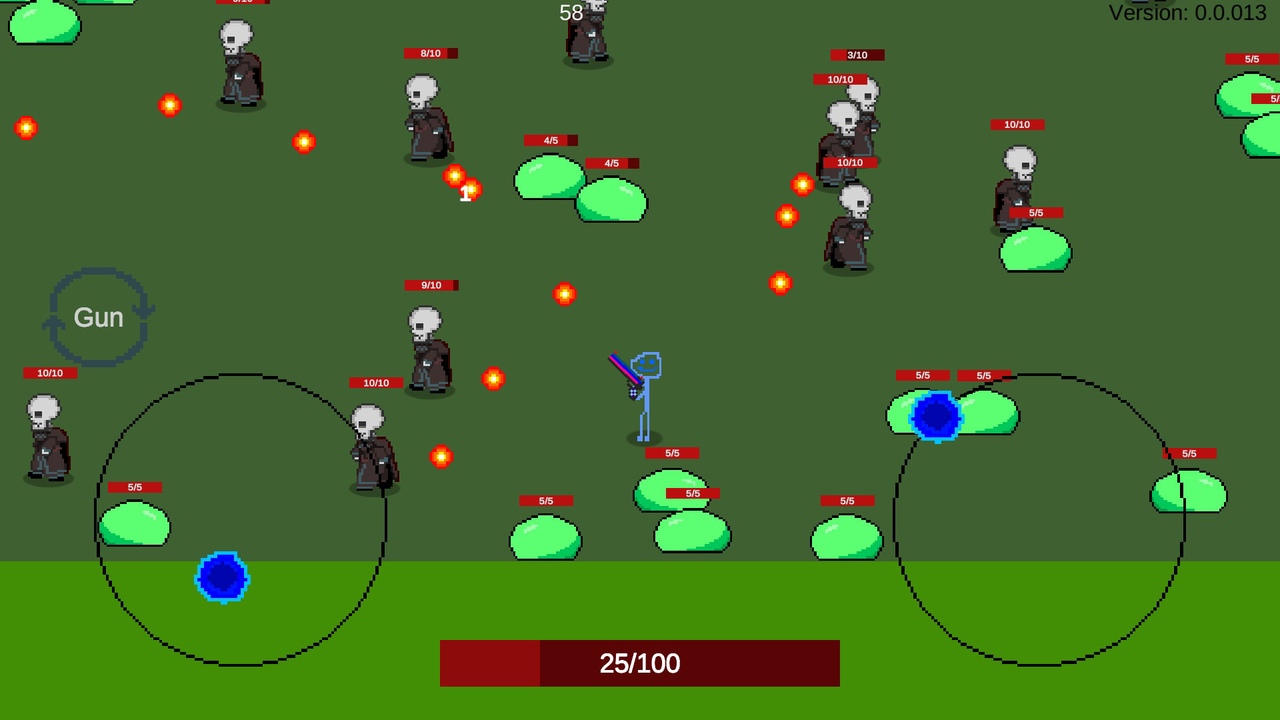




Рисунок 9. Скриншоты из игры v0.0.013.

Также был исправлен баг, связанный с тем, что в меню паузы можно было войди до того, как начальный экран полностью потухнет. Для устранения этой проблемы было добавлено новое состояние игры Pregame\_Running\_Switch, которое включается на время загрузки.

Найден новый баг, спрайт пуль меняется у всех врагов, когда игрок меняет оружие.

V0.0.014

В этой версии было добавлено:

* Управление с помощью клавиш WASD и мыши, как альтернативный способ управления.
* Система баффов и дебаффов (положительных и негативных эффектов). И первый эффект – эффект горения.
* Новый предмет, делающий пули поджигающими.

Изменено:

* Характеристики пуль теперь в трех структурах.

Для добавления управления с помощью клавиатуры и мыши были изменены некоторые настройки системы ввода.

Buff\_Debuff\_System – класс, отвечающий за баффы и дебаффы. Для его работы он должен быть прикреплен к объекту, на который будут действовать эффекты. После вызова метода Burn вызовется корутина Burn, в которой на протяжении указанного времени, с определенной периодичностью будет вызываться метод Damage компонента (скрипта) TakeDamage, прикрепленного к тому же объекту, что и система эффектов. При уничтожении или отключении системы частиц все корутины прекращают свою работу.

**Coroutine (корутина, сопрограмма)** – это выполняемая процедура. Она создается, как и обычный метод, но имеет некоторые полезные функции, которые отличают ее от стандартных методов. В Unity — простой и удобный способ запускать функции, которые должны работать параллельно в течение некоторого времени

Для добавления новых предметов необходимо:

* Нарисовать спрайт:
* Создать ресурс Item\_SO
* Написать скрипт, наследуемый от ICollectableItem
* Скопировать любой другой префаб предмета
* Заменить спрайт и предыдущий скрипт, наследуемый от ICollectableItem на новый.

Так для нового предмета был создан скрипт EnemyBurner, который после подбора предмета подписывается на новое событие класса Bullet – OnDamageDone. Событие вызывается при нанесении урона врагу и, если тег выпустившего снаряд персонажа – Player, то у системы эффектов цели вызовется метод Burn.

Для описания информации о пуле была добавлена еще одна структура – DamageType, где хранятся свойства урона. На данный момент в нем указывается является ли урон критическим, огненным или периодическим. В структуре DamageInfo вместо поля isCritical, которое отвечало за то, является ли урон критическим, теперь поле типа DamageType. В структуре BulletInfo поле Damage также было заменено на поле DamageInfo.

V0.0.015

Добавлено:

* Шейдер исчезновения
* Эффект смерти врагов

Исправлено:

* Снаряды врагов не попадают в других врагов

Было замечено, что когда вражеский снаряд задевает другого врага, то последний получает урон. Исправлено с помощью определения типа цели по тегу.

Начата работа с шейдерами для создания эффектов в игре. Т.к. Игра рассчитана для мобильных устройств шейдеры должны быть как можно проще. Для создания шейдеров в Unity есть удобный графический интерфейс, поэтому во время создания шейдера можно сразу видеть результат.

Во-первых, для добавления шейдеров, пришлось изменить способ рендеринга на Universal Render Pipeline. После его изменения все объекты все спрайты будут отрисовываться с новым материалом. Есть два вида материалов Lit и Unlit. Первый работает с освещением. Т.е. без источников света спрайты с этим материалом будут полностью черными, второй отрисовывает спрайты без изменений. Эти материалы могут изменяться с помощью шейдеров. Для экономии ресурсов устройства мы будем использовать неосвещаемые шейдеры и материалы.

Для создания эффекта исчезновения был создан новый шейдер и материал, использующий его. На вход шейдеру подаются текстура, степень исчезновения (дробное число от 0 до 1), ширина контура, цвет. При создании шейдера использовались блоки умножения, сложения, вычитания, шага, создания шума и позиции объекта. Благодаря шуму эффекты будут всегда разными.

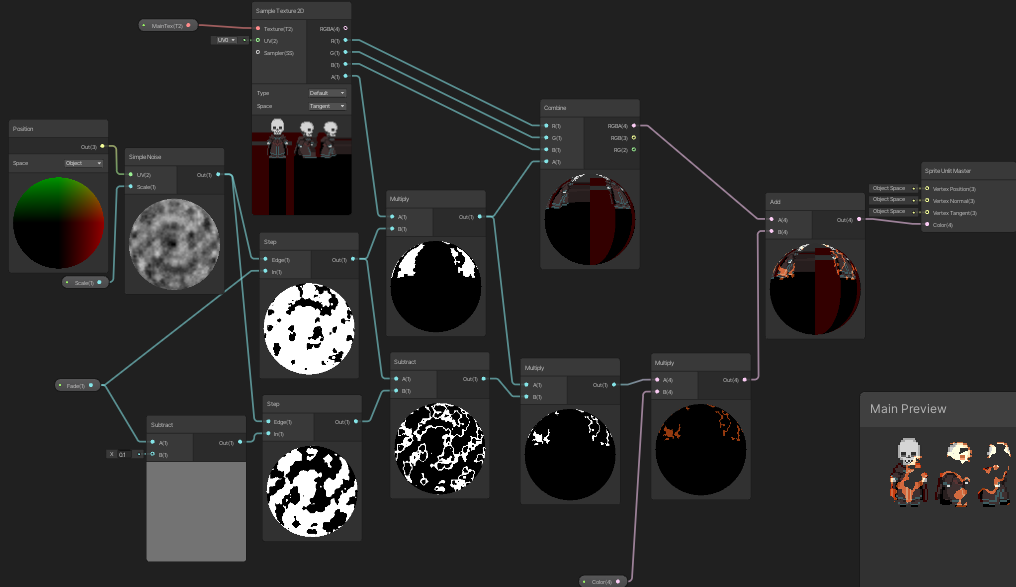


Рисунок 10. Шейдер-граф эффекта исчезновения.

Для добавления шейдеру эффекта свечения используется HDR цвет с интенсивностью 2, а также для рендеринга и камеры включена пост-обработка и на сцену добавлен объект Global Volume с компонентом Volume. В этом компоненте настраиваются различные элементы пост-обработки. Для свечения добавлен элемент Bloom.

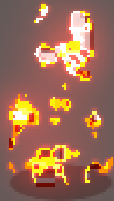


Рисунок 11. Эффект исчезновения.

Контролируется этот эффект скриптом Enemy. При смерти запускается корутина, которая плавно понижает параметр Fade от 1 до 0. Также в методе Awake устанавливается случайный цвет.

V0.0.016

Добавлено:

* Добавлен новый предмет – ледяной браслет. После получения 70 урона выпускают в 8 сторон шипы, наносящие урон и замораживающие врагов.
* Собранные предметы теперь можно выбирать в меню и читать их описание.
* Новый эффект – замерзание.
* Добавлены шейдеры для горения и замерзания.
* Цвет исчезновения врагов зависит от эффектов, наложенных на них.
* Добавлена поддержка нескольких языков.
* Для всех предметов добавлен абстрактный класс, реализующий их общую логику.

Изменено:

* Теперь предметы на платформах берутся из общего пула предметов.
* Изменен способ поворота персонажа.
* Камера теперь не является дочерней к игроку, а преследует его через отдельный скрипт.
* Изменено количество кадров в анимациях.
* Теперь игрок имеет класс-синглтон Player.

Для добавления шейдеров эффектов был создан новый шейдер – BuffEffects. Для создания эффекта горения используется два шума. Один из них всегда движется на верх. От него отсекаются все значения ниже 0.8, остальные устанавливаются в значение 1. Таким образом получаются частицы. Для добавления эффекта “мерцания” второй шум. Ему задано медленной движение. При перемножении двух шумов и добавлении цвета получился эффект, который может имитировать горение.

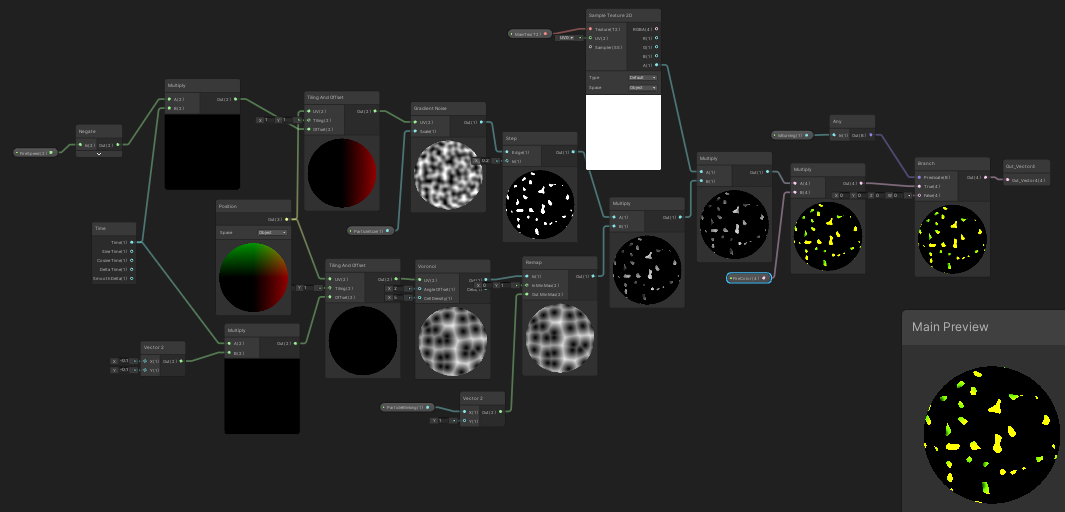


Рисунок 12. Шейдер-граф эффекта горения.

Создание эффекта заморозки можно разбить на два этапа: создание контура и создание заполняющей части. Для создания контура у текстуры материала берется альфа-канал (отвечающий за прозрачность), смещается в четырех направления, после чего все суммируется, значения выше нуля приводятся к нулю, вычитается начальное значение альфа-канала.

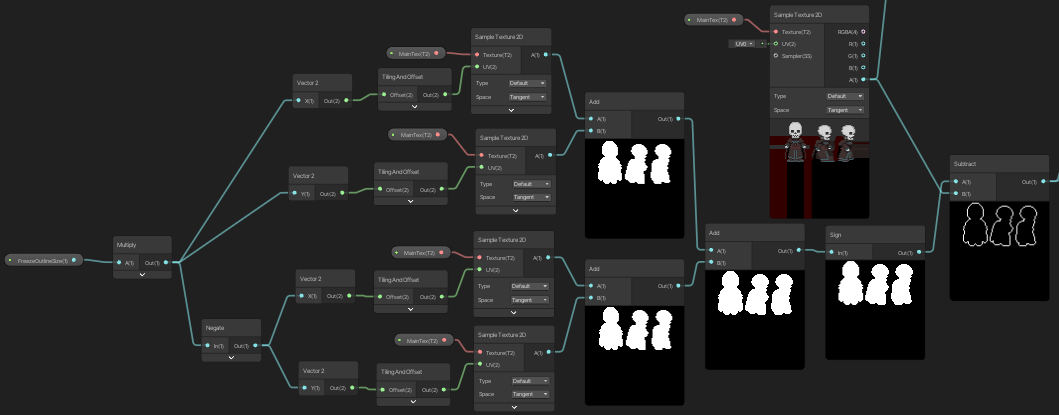


Рисунок 13.Шейдер-граф контура спрайта.

Далее контур окрашивается, значение альфа канала спрайта уменьшается и также окрашивается. После сложения элементов получается эффект заморозки.

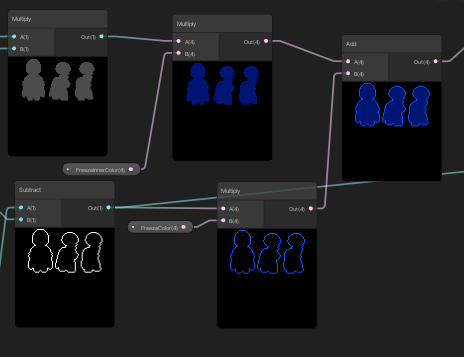


Рисунок 14. Конечный этап эффекта заморозки.

Далее эффекты горения и заморозки накладываются на изначальный спрайт.

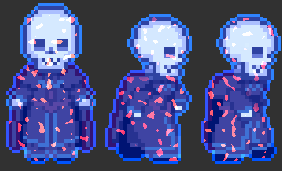
 

Рисунок 15. Эффект заморозки, горения и их сочетание.

Для добавления предмета был нарисован спрайт и написан скрипт. Скрипт подписывается на событие получения игроком урона. Когда получено 70 единиц урона вызывается метод Fire класса FireManager, для произведения восьми выстрелов. Данные для снарядов берутся из класса WeaponManager, поэтому пули, выпущенные этим предметом, будут получать эффекты от других предметов, действующие на пули. Например, после подбора поджигающего предмета, пули будут не только замораживать и наносить урон, но и поджигать.

В скрипте Buff\_Debuff\_System добавлены метод и корутина Freeze. При вызове метода в нем вызывается корутина, которая на некоторое время уменьшает скорость передвижения объекта. Также при старте и завершении корутин Burn и Freeze включаются соответствующие эффекты шейдера.

При смерти класс Enemy обращается к классу Buff\_Debuff\_System, для получения информации о текущих эффектах. В зависимости от наложенных эффектов устанавливается цвет эффекта исчезновения. Цвет получается путем определенного смешивания цветов, назначенных для каждого шейдера. Если враг погибает без эффектов, он растворяется белым цветом, если горит – оранжевым, если замерз – синим, если замерз и горит – фиолетовым.

Для добавления общей логики всем предметам, был создан интерфейс IItem, содержащий один метод Initialize. Описывается этот метод в абстрактном классе Item. Уже от класса Item наследуется класс CollectableItem.

Также для работы с предметами был создан скрипт ItemPool. Он хранит ссылки на предметы (любой объект/префаб, обладающий компонентом, наследуемым от абстрактного класса Item) и разделяет их на полученные/не полученные игроком и размещенные/не размещенные на платформах. Этот скрипт имеет методы для получения ранее не установленного предмета (ItemPlatform теперь использует этот метод) и метод для получения предмета (вызывается скриптом CharacterInventory).

При добавлении текста в игру было решено сразу добавить возможность добавлять новые языки. Если не реализовать эту возможность сейчас, то позже такое решение займет очень много времени. Для хранения текста было решено использовать файл в xml формате.

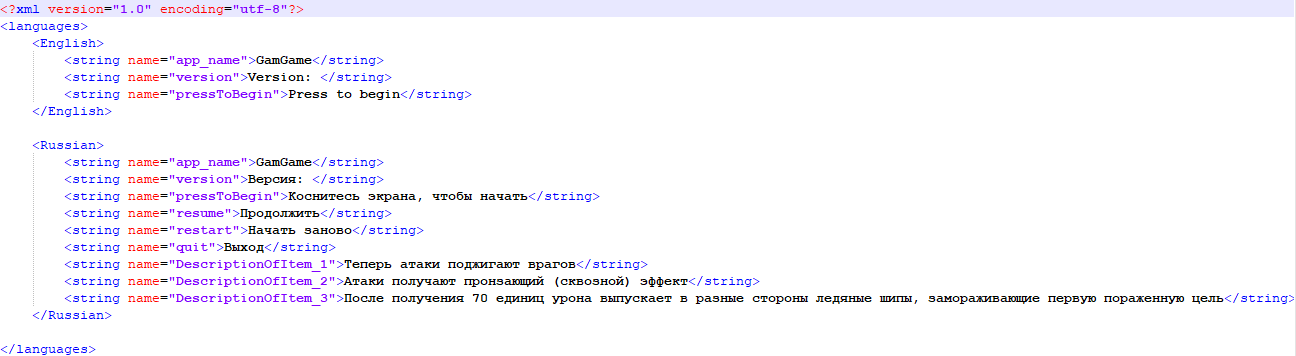


Рисунок 16. Структура текстового файла.

В файле внутри тега Languages находится несколько тегов с языками (English, Russian), внутри которых теги string с определенными именами и содержанием. Файл с данными располагается в отдельной папке StreamingAssets. Данные в этой папке сериализуются, а во время работы программы их можно получить через поток. Класс LanguageController считывает строки указанного языка и записывает их в хеш-таблицу. Класс-синглтон GameText в методе Awake сообщает объекту класса LanguageController данные какого класса необходимо считать. После завершения считывания вызывается событие EventLanguageChanged, на которое могут отреагировать объекты, имеющие текст. После смены языка в метод GetText класса GameText можно передавать имя, чтобы получить соответствующий ему текст.

В меню предметов был добавлен метод, срабатывающий при нажатии на предмет в инвентаре. Этот метод отображает спрайт выбранного предмета в окне выбора и соответствующий ему текст. Теперь для добавления предмета также необходимо добавлять описание.

Ранее персонаж поворачивался при помощи изменения его размера по оси y с 1 на -1, теперь для этого используется метод Rotate. Так как камера является дочерним элементом игроку элементом, то поворачивается вместе с ним, из-за чего было решено вынести её на сцену как самостоятельный объект и добавить скрипт для преследования игрока.

Так как множество скриптов обращаются к характеристикам игрока (скрипты стрельбы, для получения значений урона, перезарядки и т.д, скрипты управления персонажем, скрипты атаки врагов, текущие и будущие скрипты предметов), то было решено создать класс-синглтон Player, который будет хранить ссылку на ресурс CharacterStats игрока и предоставлять доступ к его значениям, а также здесь будут храниться ссылки на компоненты WeaponManager, CharacterInventory, TakeDamage, Rigidbody2D, SpriteRenderer и Animator. Таким образом доступ к любой информации об игроке можно получить через класс Player.

Последним изменением в этой версии стали анимации персонажей. Ранее они были записаны как 60-ти кадровые, но изображение изменялось лишь раз в 10-20 кадров. Теперь длина анимаций уменьшена до количества спрайтов в анимации. Такое изменение значительно влияет на производительность при большом количестве врагов.

Дальнейшие действия в реализации проекте

Далее игра будет доведена до логического завершения. Будет добавлено множество новых предметов, врагов, локаций и боссов. В игре появятся сохранения, а для более красивого внешнего вида будет проведена работа со спрайтами, светом и частицами. Также в игре появится конец.