**Введение**

Моими целями в проекте Самосбор было написание кода на C# для создания разных механик и получение новых знаний в среде разработки Unity. Из тех задач что были выполнены мной это передвижение персонажа (ходьба, плавный бег, прыжок, присед, блокирование камеры на персонаже по углам наклона, стамина персонажа для бега), добавление фонарика персонажу как источник света (эта задача была не только моей, но финальная версия была выбрана моя), инвентарь персонажа для добавления вещей в него и соответственно выкладывание из него предметов, главное меню с кнопками играть, настройки, выход, так же добавлено было меню паузы во время игры с кнопками продолжить, настройки, выход, добавлен в игру был звук ходьбы и прыжка.

Первый код под названием [PlayerController](https://github.com/Zolotushka1/MorLovecraft/blob/armagedontkd/Assets/Scripts/armagedontkd/PlayerController.cs)

Класс PlayerController имеет несколько переменных, определяющих скорость перемещения и силу прыжка, а также состояние поверхности, обновляемое, когда персонаж касается земли. Метод Update вызывается каждый раз, когда игровой движок проходит цикл обновления и здесь вызывается метод GetInput(). Метод GetInput() обрабатывает входные данные, для управления персонажем. Если нажата клавиша W, тогда персонаж передвигается вперед и если нажата клавиша S - то назад. Если нажата клавиша A, то персонаж поворачивается влево, а при нажатии клавиши D - вправо. Также, если персонаж касается земли и нажата клавиша пробела, то персонаж прыгает. Методы OnCollisionEnter и OnCollisionExit вызываются, когда происходит столкновение персонажа с объектами, на которые он может наступать. Если персонаж касается земли, тогда состояние поверхности устанавливается в true, если же нет, то в false.

Второй код под названием [Player\_Move](https://github.com/Zolotushka1/MorLovecraft/blob/armagedontkd/Assets/Scripts/armagedontkd/Player_Move.cs)

Этот код представляет собой скрипт для движения игрока в Unity. В нём определены переменные для управления скоростью бега, стамины и прыжка. Также определены слайдеры для отображения оставшейся стамины игрока и текст на нём. Более подробно, данный скрипт отвечает за управление скоростью бега в зависимости от уровня стамины, прыжком, гравитацией и другими параметрами. Метод Update() вызывается каждый кадр и вызывает два других метода - Move() и Stamina(), которые реализуют логику перемещения персонажа и управления его выносливостью соответственно. Метод Move() получает ввод от игрока с помощью методов класса Input и устанавливает переменные x\_Move и z\_Move в зависимости от того, какие клавиши были нажаты. Затем метод формирует новый вектор движения и преобразует его в локальные координаты. Если игрок находится на земле, то метод реагирует на нажатие клавиши пробела и изменяет y-компоненту вектора движения на величину jump. Метод также реализует функциональность приседания. Когда игрок нажимает на клавишу LeftControl, его высота соответственно уменьшается, что позволяет пройти сквозь узкие проемы. Игрок также может зажать клавишу LeftControl, чтобы присесть, при этом его высота уменьшается до 0.7f, и он перемещается со скоростью, которая зависит от значения speed\_Run. Когда игрок отпускает клавишу LeftControl, высота игрока вновь увеличивается до исходного значения, и он снова перемещается со скоростью speed\_Run2.

Кроме того, в методе реализуется воспроизведение различных звуков движения персонажа в зависимости от исходных установок и действий.

Третий код [Player\_MouseMove](https://github.com/Zolotushka1/MorLovecraft/blob/armagedontkd/Assets/Scripts/armagedontkd/Player_MouseMove.cs)

Данный код используется для управления камерой и поворота персонажа в Unity. Класс Player\_MouseMove содержит методы Start() и Update(). В функции Start() устанавливается начальная позиция курсора внутри окна приложения и задаются начальные значения переменных. В функции Update() обрабатываются движения мыши (Input.GetAxis) и осуществляется изменение местоположения камеры и персонажа в соответствии с этими движениями. Также, поскольку мышь может двигаться свободно по осям X и Y, обрабатываются соответствующие ограничения, чтобы предотвратить переход камеры за пределы допустимых значений по осям позиции. Для плавного перехода используется метод Mathf.SmoothDamp для достижения более плавного движения.

Четвертый код [CameraController](https://github.com/Zolotushka1/MorLovecraft/blob/armagedontkd/Assets/Scripts/armagedontkd/CameraController.cs)

Данный код написан на языке C# и используется для управления камерой и поворота персонажа в Unity. Класс CameraController содержит методы Start() и Update(). В функции Start() устанавливается начальная позиция курсора внутри окна приложения. В функции Update() обрабатываются движения мыши (Input.GetAxis) и осуществляется изменение местоположения камеры и персонажа в соответствии с этими движениями. В данном случае поворот персонажа осуществляется относительно его оси Y достигается при помощи метода Rotate класса Transform для объекта Player. Поворот камеры осуществляется относительно оси X также при помощи метода Rotate, но уже для самого объекта CameraController.

Пятый код [MouseSensitivity](https://github.com/Zolotushka1/MorLovecraft/blob/armagedontkd/Assets/Scripts/armagedontkd/MousSensitivity.cs)

Данный код используется для установки и получения чувствительности мыши в Unity. Класс MousSensitivity содержит 3 метода: GetValue(), Start(), SetValue(). Метод GetValue() является статическим и возвращает значение чувствительности мыши, которое было сохранено в PlayerPrefs (интерфейс хранения данных для Unity). В методе Start() устанавливается значение на объект слайдера, который содержит значение чувствительности, с помощью метода GetComponent(). Метод SetValue() используется для сохранения значения чувствительности мыши в PlayerPrefs.

В данном коде используется переменная PREFS\_KEY для более удобной работы с PlayerPrefs и получения сохраненных настроек.

Шестой код [MenuManager](https://github.com/Zolotushka1/MorLovecraft/blob/armagedontkd/Assets/Scripts/armagedontkd/MenuManager.cs)

Данный код используется для управления меню в Unity.

Класс MenuManager содержит четыре метода: Play(), Quit(), Settings(), и Start(). Метод Play() используется для загрузки определенной сцены игры при нажатии на кнопку "Play" в меню. В данном случае, с помощью метода LoadScene класса SceneManager загружается сцена с индексом 1.

Метод Quit() используется для выхода из игры при нажатии на кнопку "Quit" в меню. Для выхода из приложения используется метод Quit класса Application, а также выводится сообщение в консоль, которое содержит информацию о происходящем действии.

Метод Settings() отображает панель настроек в меню и скрывает главное меню, а также производит переход от основной камеры к камере настроек.

Метод Start() не отображается в данном фрагменте кода, но он вызывается при запуске сцены и используется для установки начальных значений, если это необходимо.

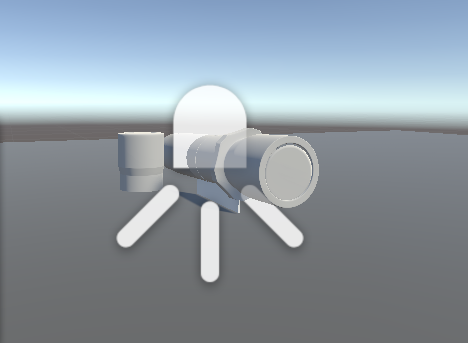
Седьмой код [MenuSettings](https://github.com/Zolotushka1/MorLovecraft/blob/armagedontkd/Assets/Scripts/armagedontkd/MenuSettings.cs)

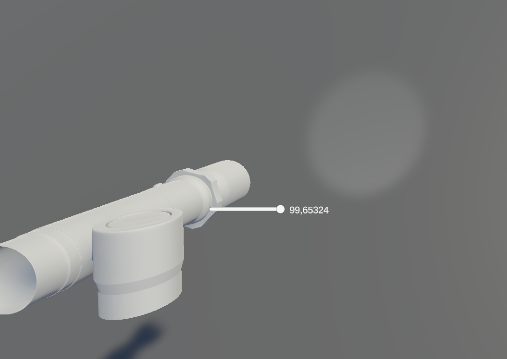
Данный код используется для управления настройками в меню в Unity. Класс MenuSettings содержит методы Start(), SetFullscreen(), SetResolution(), SetQuality() и ExitSettings(), а также несколько public полей, включая поля типа TMP\_Dropdown и Slider и ссылки на разные элементы интерфейса. Метод Start() используется для создания списка резолюций и заполнения поля resolutionDropdown в меню этим списком в виде строковых объектов. Также вызывается метод LoadSettings(int index), который загружает сохраненные настройки. Метод SetFullscreen() используется для установки параметра на полноэкранный режим игры. В параметр isFullscreen передается значение true или false в зависимости от того, выбран ли полноэкранный режим. Метод SetResolution() используется для изменения текущего разрешения экрана. Новое разрешение берется из массива resolutions на основе переданного в метод индекса resolutionIndex. В конце метода вызывается метод SetResolution() класса Screen, который изменяет текущее разрешение экрана на заданное. Метод SetQuality() используется для изменения уровня качества графики в настройках. Новое значение уровня качества определяется на основе переданного в метод индекса qualityIndex (от 0 до 5). В конце метода вызывается метод SetQualityLevel() класса QualitySettings, который изменяет текущий уровень качества графики на заданный. Метод ExitSettings() используется для скрытия панели настроек и отображения главного меню, а также производит переход от камеры настроек к главной камере. Метод SaveSettings() используется для сохранения текущих настроек игры. Он сохраняет значение поля mouseSensitivity в качестве параметра чувствительности мыши, выбранный уровень качества графики в виде числа, выбранное разрешение экрана в виде числа и параметр полноэкранного режима в виде булевого значения.

Метод LoadSettings() используется для загрузки сохраненных настроек. Если настройки уже сохранены в PlayerPrefs, то они загружаются и устанавливаются как текущие значения в соответствующих полях интерфейса. Если нет, то устанавливаются стандартные значения по умолчанию. По сути, данная часть кода управляет сохранением и загрузкой настроек игры, которые в дальнейшем будут использоваться при установке разрешения экрана, уровня качества графики и других параметров в меню игры.

Восьмой код [Flashlight](https://github.com/Zolotushka1/MorLovecraft/blob/armagedontkd/Assets/Scripts/armagedontkd/Flashlight.cs)

Данный код реализует функциональность фонарика для игры в Unity. Переменные flashlightObject и flashlight — это GameObject-ы, которые представляют собой фонарик в игровом мире и его источник света соответственно. В методе Start() получается ссылка на компонент света у объекта фонарика и выключается объект фонарика. В методе Update() происходит проверка на нажатие клавиши G. Если клавиша была нажата, то вызывается метод TurnOnFlashlight() через 1 секунду с помощью метода Invoke(). Если фонарик включен (переменная flashlightOn равна true), то фонарик поворачивается в сторону, куда смотрит камера. Метод TurnOnFlashlight() включает или выключает фонарик и его источник света, а также отображает или скрывает объект фонарика в зависимости от значения переменной flashlightOn. Таким образом, данный код реализует работу фонарика.





Девятый код [PausedMenuSettings](https://github.com/Zolotushka1/MorLovecraft/blob/armagedontkd/Assets/Scripts/armagedontkd/PausedMenuSettings.cs)

Данный код на реализует функциональность меню настроек паузы для игры в Unity. Переменные resolutionDropdown, qualityDropdown и mouseSensitivity представляют собой UI-элементы интерфейса пользователя в Unity, которые используются для регулирования качества графики, установки разрешения экрана, полноэкранного режима и чувствительности мыши. Метод Start() заполняет\_Dropdown с разрешением экрана и устанавливает разрешение в соответствии с текущим разрешением экрана. Также метод Start() загружает настройки графики. Методы SetFullscreen(), SetResolution() и SetQuality() используются для установки соответствующих параметров качества графики. Метод ExitSettings() используется для выхода из меню настроек и возврата к основному меню. Таким образом, данный код реализует меню настроек паузы для игрового процесса.

Десятый код [SoundsManager](https://github.com/Zolotushka1/MorLovecraft/blob/armagedontkd/Assets/Scripts/armagedontkd/SoundsManager.cs)

Данный код представляет собой определение класса SoundsManager в Unity, в котором задаются звуковые эффекты для разных событий. В данном классе есть публичные поля footstepSound и oneshotSounds, которые соответствуют источникам звуков в игре. Также есть методы Jump(), StepsActive(), StepsDeactive() и Fall(), которые проигрывают звуковые эффекты при выполнении соответствующих действий в игре. Метод Jump() проигрывает звук при прыжке, StepsActive() активирует звук шагов персонажа, StepsDeactive() деактивирует звук шагов персонажа, Fall() проигрывает звук падения.

Одиннадцатый код [Settings](https://github.com/Zolotushka1/MorLovecraft/blob/armagedontkd/Assets/Scripts/armagedontkd/Settings.cs)

Данный код представляет собой определение класса Settings в Unity, который отвечает за настройки игры. В классе есть несколько публичных полей, таких как resolutionDropdown, qualityDropdown, которые являются выпадающими списками со списками настроек разрешения и качества графики. С помощью метода Start() задаются различные опции разрешения экрана, которые очищают список значений в resolutionDropdown, устанавливают список опций, устанавливают значение текущего разрешения и обновляют значение в resolutionDropdown, а также загружают настройки в индекс текущего разрешения. Метод SetFullScreen() устанавливает полноэкранный режим в зависимости от аргумента isFullscreen, метод SetResolution() устанавливает разрешение экрана из списка по выбранному индексу, метод SetQuality() устанавливает качество графики по указанному индексу. Метод ExitSettings() используется для выхода из окна настроек и загрузки главного меню игры. Метод SaveSettings() сохраняет выбранные значения настроек - качества графики, разрешения экрана и полноэкранного режима, используя метод PlayerPrefs.SetInt(), который сохраняет значение в реестре (для Windows) или в хранилище на устройстве (для мобильных устройств). Метод LoadSettings() загружает предыдущие настройки, если они сохранены в PlayerPrefs, используя методы PlayerPrefs.HasKey() и PlayerPrefs.GetInt(), и устанавливает значения элементов управления в соответствии с загруженными настройками. currentResolutionIndex - это индекс текущего разрешения экрана, который используется, если значение разрешения не сохранено в PlayerPrefs.

Двенадцатый код [Item](https://github.com/Zolotushka1/MorLovecraft/blob/armagedontkd/Assets/Scripts/armagedontkd/Inventory/Item.cs)

Этот код определяет класс Item в Unity, который отвечает за предметы в игре. Класс содержит два публичных поля item и amount. item является ссылкой на объект скриптового объекта, который хранит основные свойства предмета (например, название, описание, изображения), а amount представляет количество данного предмета в инвентаре игрока.

Тринадцатый код [ItemScriptableObject](https://github.com/Zolotushka1/MorLovecraft/blob/armagedontkd/Assets/Scripts/armagedontkd/Inventory/ItemScriptableObject.cs)

Этот код определяет скриптовый объект ItemScriptableObject, который представляет собой шаблон для создания конкретных игровых предметов в Unity. ItemScriptableObject содержит публичные поля, такие как itemName, maximumAmount, itemPrefab, icon, itemType и itemDescription, которые задают основные свойства создаваемых предметов: itemName - название предмета maximumAmount - максимальное количество предмета, которое может быть взято в инвентарь itemPrefab - префаб игрового объекта, который будет создаваться при добавлении этого предмета в игру, icon - изображение предмета, которое будет отображаться в меню инвентаря и в других местах, где необходимо показывать этот предмет, itemType - тип предмета (Default, Food, Weapon, Instrument) itemDescription - описание предмета ItemType является перечислением, которое содержит типы предметов, которые могут быть созданы для использования в игре.

Четырнадцатый код [ItemDetails](https://github.com/Zolotushka1/MorLovecraft/blob/armagedontkd/Assets/Scripts/armagedontkd/Inventory/ItemDetails.cs)

Этот код определяет класс ItemDetails в Unity, который отвечает за отображение подробной информации об игровом предмете в дополнении к основному инвентарю. Класс содержит статический метод ActivateWith(), который используется для отображения дополнительных сведений о предмете. В методе вызывается метод Use(), который отображает информацию о данном предмете. Класс также имеет реализацию синглтона, то есть только один экземпляр класса может быть создан и управляться во время выполнения программы Unity. ItemDetails определяет различные компоненты интерфейса, такие как изображение предмета, описание предмета и панель отображения, которые используются для отображения более подробной информации о предметах. Метод ReturnToInventory() используется для возвращения к основному меню инвентаря после отображения дополнительной информации о предмете.

Пятнадцатый код [InventoryManager](https://github.com/Zolotushka1/MorLovecraft/blob/armagedontkd/Assets/Scripts/armagedontkd/Inventory/InventoryManager.cs)

Этот код определяет класс InventoryManager в Unity, который отвечает за управление инвентарём игрока. Класс InventoryManager содержит реализацию синглтона, что означает, что только один экземпляр класса может быть создан и управляться во время выполнения программы Unity. Класс отвечает за отображение интерфейса инвентаря, управление элементами инвентаря и активацию/деактивацию панели инвентаря при нажатии определенной клавиши. Класс InventoryManager содержит поля, такие как inventoryPanel, slots, reachDistance, mouseMove, UIPanel и DestroyOnOpen, которые используются для управления компонентами интерфейса инвентаря и решениями игровых механик. Метод Update() проверяет, была ли нажата клавиша "E". Если это так, то создается луч, который проходит через центр камеры и попадает в границы, определенные параметром reachDistance. Если луч сталкивается с объектом, который содержит компонент Item, то этому компоненту передаются параметры item и amount, которые извлекаются из компонента. Затем метод AddItem() вызывается с этими параметрами в качестве входных данных. Метод AddItem() проверяет, существуют ли слоты, занятые элементами инвентаря, которые соответствуют переданному параметру в качестве аргумента функции \_item. Если элементы обнаружены, то соответствующее количество переданного параметра \_amount добавляется к количеству элементов в слоте. В противном случае, в инвентаре создается новый элемент с параметрами \_item и \_amount, и его иконка и количество отображаются в соответствующем слоте.

Шестнадцатый код [InventorySlot](https://github.com/Zolotushka1/MorLovecraft/blob/armagedontkd/Assets/Scripts/armagedontkd/Inventory/InventorySlot.cs)

Этот код определяет класс InventorySlot, который представляет слот для хранения предметов в инвентаре игрока. У этого класса есть несколько свойств: item, amount и isEmpty, которые представляют собой информацию о предмете, количестве и том, является ли слот пустым. Также есть ссылки на объект-иконку слота и текстовый объект, который отображает количество предметов в слоте. Метод SetIcon() устанавливает изображение, отображаемое для иконки слота, и делает его видимым, установив цвет объекта изображения на непрозрачный (new Color(1, 1, 1, 1)). Параметр icon этого метода - это спрайт, используемый в качестве иконки предмета.

Семнадцатый код [FoodItem](https://github.com/Zolotushka1/MorLovecraft/blob/armagedontkd/Assets/Scripts/armagedontkd/Inventory/FoodItem.cs)

Этот код определяет класс FoodItem, который является наследником класса ItemScriptableObject и представляет собой конкретный тип предмета - еду. У этого класса есть свойство healAmount, указывающее количество очков здоровья, которое будет восстановлено при использовании предмета. Также есть метод Start(), который определяет тип предмета как ItemType.Food. Этот класс наследует от ItemScriptableObject, который определяет общую функциональность для всех типов предметов в игре, например, название, описание и изображение. Этот код также использует [CreateAssetMenu], который позволяет создавать новые экземпляры объекта через контекстное меню в редакторе Unity.