**EnemyScripts**

**Описание класса : EnemyTargetMouse**

Этот скрипт реализует изменение цвета объекта при наведении и отведении мыши. Когда мышь наводится на объект, его цвет меняется на красный, а когда мышь уходит с объекта, его цвет возвращается к белому.

**Описание методов класса:**

Start(): Вызывается при старте скрипта. Получает компонент Renderer объекта.

OnMouseEnter(): Вызывается при наведении мыши на объект. Изменяет цвет объекта на красный.

OnMouseExit(): Вызывается при отведении мыши с объекта. Изменяет цвет объекта на белый.

**Описание класса : EnemyAI**

Отвечает за управление искусственным интеллектом вражеских персонажей в игре. Он определяет цель атаки для вражеских персонажей и управляет их движением к этой цели.

**Поля:**

GameObject enemyTarget: Цель атаки вражеского персонажа.

**Описание методов класса:**

Awake(): Вызывается при инициализации объекта. Получает ссылку на компонент NavMeshAgent и находит цель атаки вражеского персонажа.

Start(): Вызывается перед первым обновлением кадра. Устанавливает начальные настройки вражеского персонажа, перемещает врага к enemyTarget.

TargetOfEnemyAttack(): Увеличивает скорость передвижения вражеского персонажа и устанавливает цель движения вражеского персонажа к enemyTarget.

**Описание класса : EnemyAssignmentComponent**

отвечает за назначение **EnemyTargetMouse**  для вражеских персонажей в сцене.

**Описание методов класса:**

Start(): Вызывает метод OverlayEnemyTarget() для назначения **EnemyTargetMouse** и определения количества вражеских персонажей в сцене.

OverlayEnemyTarget(): Находит всех вражеских персонажей в сцене и назначает им компонент **EnemyTargetMouse**, который выделяет врага при наведении мыши.

**Описание класса : EnemyCount**

Отображает количество врагов на экране, используя текстовый компонент UI, и обновляет его в реальном времени.

**Описание методов класса:**

Awake(): Вызывается при инициализации объекта. Получает ссылки на компонент текста UI и компонент **EnemyAssignmentComponent**.

Update(): Вызывается каждый кадр. Обновляет отображаемое количество врагов.

EnemyCounter(): Обновляет отображаемое количество врагов на основе количества врагов, подсчитанных в компоненте **EnemyAssignmentComponent**.

**Описание класса : EnemySpawnPoint**

Отвечает за спавн врагов в случайных точках спауна с заданной задержкой.

**Описание методов класса:**

Awake(): Вызывается при инициализации объекта. Получает ссылку на компонент **EnemyAssignmentComponent** и находит все точки спауна в сцене.

Start(): Вызывается перед первым обновлением кадра. Начинает процесс спауна врагов.

Update(): Вызывается каждый кадр. Обновляет отображаемое количество врагов.

EnemSpawnPointRandom(): Спаунит врагов в случайных точках спауна с заданной задержкой.

**LevelScripts**

**Описание класса : AltarTakeDamage**

Отвечает за управление прочностью защиты алтаря в игре. Когда враг входит в зону триггера алтаря, прочностью защиты уменьшается на определенное количество.

**Описание методов класса:**

OnTriggerEnter(Collider other): Вызывается, когда другой коллайдер входит в триггер объекта Altar. Проверяет, является ли объект, вошедший в триггер, врагом (с компонентом **EnemyAI**). Если да, прочность защиты алтаря уменьшается и обновляется отображение прочности. Если прочность защиты достигает нуля, при последующем столкновении с врагом отображается анимация уничтожения(смерти) алтаря.

**Описание класса : CollectionScripts**

Представляет собой компонент, который хранит коллекцию скриптов, связанных с объектом в Unity.

**Поля:**

List<MonoBehaviour> collectionScripts: Список моноскриптов, связанных объектом.

**Описание класса : GameQuit**

Представляет собой компонент, который отвечает за завершение выполнения приложения.

**Описание методов класса:**

Exit(): Завершает выполнение приложения. Когда этот метод вызывается, приложение закрывается и выходит из режима выполнения.

**Описание класса : GameWinDefeat**

Отвечает за определение условий победы и поражения в игре, а также за отображение соответствующего интерфейса в случае победы или поражения.

**Описание методов класса:**

Awake(): Вызывается при инициализации объекта. Инициализирует ссылки на компоненты **AltarTakeDamage** и **EnemyAssignmentComponent**.

Update(): Выполняет проверку состояния алтаря и количества врагов каждый кадр.

CheckAltarHealth(): Проверяет здоровье алтаря и количество врагов. Если алтарь разрушен или врагов больше нет, запускает процесс завершения игры.

EndGameRoutine(): Процесс завершения игры, который включает отображение победного или пораженческого интерфейса и остановку времени.

**Описание класса : LevelLoading**

Отвечает за загрузку указанной сцены в Unity.

**Описание методов класса:**

SceneLoad(): Вызывается для загрузки указанной сцены с помощью SceneManager.LoadScene(index).

**Описание класса : SoundManager**

Управляет настройками звука в игре, позволяя изменять громкость звука с помощью слайдера и сохранять это состояние в формате JSON.

**Описание методов класса:**

Start(): Вызывается при запуске игры. Устанавливает начальное значение слайдера и загружает сохраненные настройки звука.

Update(): Вызывается каждый кадр. Обновляет громкость звука при изменении слайдера.

SetVolume(float value): Устанавливает громкость звука в соответствии с заданным значением и сохраняет настройки звука.

SaveSoundSettings(): Сохраняет текущие настройки звука в формате JSON.

LoadSoundSettings(): Загружает сохраненные настройки звука из файла JSON при запуске игры.

SoundSettingsData(): Представляет данные о настройках звука для сохранения в формате JSON.

**Конструктор:**

SoundSettingsData(float volume): Создает новый экземпляр класса SoundSettingsData с указанной громкостью звука.

**Описание класса : TimeControl**

Отвечает за управление временем в игре, позволяя останавливать и возобновлять его воспроизведение.

**Описание методов класса:**

Start(): Вызывается при запуске сцены. Устанавливает время в игре на паузу.

TimeStopPlay(): Переключает между паузой и воспроизведением времени в игре. Если время воспроизведения равно 1 (включено), останавливает его; если равно 0 (остановлено), возобновляет.

**CharacterScripts : AbilitiesScripts**

**Описание класса : AttackDistant**

**О**твечает за реализацию дальнего атакующего действия.

**Описание методов класса:**

Start(): Вызывается при запуске сцены. Инициализирует ссылку на компонент **AttackDistantTarget.**

Update(): Вызывается каждый кадр. Проверяет возможность атаки в дальнем бою и обновляет состояние атаки.

AttackCheck(): Проверяет возможность атаки и обновляет состояние атаки.

**Описание класса : AttackMelee**

Управляет атакой в ближнем бою, включая отображение и обновление состояния атаки.

**Описание методов класса:**

Awake(): Вызывается при загрузке объекта. Инициализирует компоненты анимации и коллайдера.

Update(): Вызывается каждый кадр. Проверяет возможность атаки в ближнем бою.

AttackMeleeCheck(): Проверяет возможность атаки в ближнем бою дальнем бою и обновляет состояние атаки.

**CharacterScripts : AuxiliaryScriptsAttack**

**Описание класса : AttackDistantTarget**

Отслеживает позицию цели для дальней атаки и обновляет положение объекта к ней.

**Описание методов класса:**

Update(): Вызывается каждый кадр. Проверяет позицию цели для дальней атаки.

PositionCheck(): Проверяет позицию цели для дальней атаки и обновляет положение объекта.

**Описание класса : AttackMeleeTarget**

Отслеживает столкновения коллайдера с объектами и изменяет цвет целей.

**Описание методов класса:**

Awake(): Вызывается при старте объекта. Получает компонент Renderer текущего объекта.

OnTriggerEnter(Collider other): Вызывается при входе коллайдера в триггерную область. Изменяет цвет цели на красный.

OnTriggerExit(Collider other): Вызывается при выходе коллайдера из триггерной области. Возвращает цвет цели к исходному состоянию.

**Описание класса : DamageTakeEnemy**

Оотслеживает столкновения коллайдера с объектами и обрабатывает получение урона объектом.

**Описание методов класса:**

Awake(): Вызывается при старте объекта. Получает ссылки на компоненты **AttackDistantTarget** и **EnemyAssignmentComponent**.

OnTriggerEnter(Collider other): Вызывается при входе коллайдера в триггерную область. Обрабатывает получение урона объектом.

**CharacterScripts : CommonScripts**

**Описание класса : BarrierCollision**

Отслеживает столкновения коллайдера персонажа с объектами барьеров и управляет флагом изменения скорости.

**Описание методов класса:**

OnTriggerEnter(Collider other): Вызывается при входе коллайдера в триггерную область. Устанавливает флаг изменения скорости, если столкнулись с объектом барьера.

OnTriggerExit(Collider other): Вызывается при выходе коллайдера из триггерной области. Сбрасывает флаг остановки движения, если объект барьера был покинут.

**Описание класса : CheckCollisionBarrier**

Отслеживает столкновения игрока с барьером и управляет скоростью его движения.

**Описание методов класса:**

Awake(): Вызывается при старте объекта. Получает ссылки на компоненты BarrierCollision и PlayerMove.

Start(): Вызывается при старте объекта. Устанавливает скорость движения при замедлении.

Update(): вызывается каждый кадр. Проверяет и устанавливает скорость движения игрока в зависимости от наличия столкновения с барьером и нажатия клавиши LeftShift.

CheckSpeed(): Проверяет наличие столкновения с барьером и устанавливает скорость движения игрока.

**Описание класса : ConditionStorage**

Отслеживает состояние нажатия клавиши LeftShift и управляет активностью игровых объектов scythe и spirit.

**Описание методов класса:**

Update(): Вызывается каждый кадр. Проверяет условие и управляет активностью игровых объектов.

ConditionCheck(): Проверяет условие и управляет активностью игровых объектов в зависимости от нажатия клавиши LeftShift.

**CharacterScripts : MoveScripts**

**Описание класса : PlayerInput**

Отслеживает ввод от игрока по вертикали и горизонтали, а также определяет направление движения игрока.

**Поля:**

public float vInput: Ввод от игрока по вертикали.

public float hInput: Ввод от игрока по горизонтали.

public Vector3 movementDirection: Направление движения игрока.

**Описание методов класса:**

Update(): Вызывается каждый кадр. Проверяет ввод от игрока.

InputCheck(): Получает ввод от клавиатуры и обновляет значения vInput, hInput и movementDirection в соответствии с вводом.

**Описание класса : PlayerMove**

Отвечает за перемещение игрока на основе ввода от игрока, управляемого классом **PlayerInput**.

**Описание методов класса:**

Awake(): Вызывается при создании объекта. Получает ссылки на компоненты инициализирует поля.

FixedUpdate(): Вызывается на каждом фиксированном кадре. Обрабатывает логику перемещения игрока.

MoveLogic(): Осуществляет перемещение игрока на основе ввода от игрока.