Bound by Shadows

Wygenerowano za pomocą Doxygen 1.14.0

Rozdział 1

Indeks przestrzeni nazw

1.1 Lista przestrzeni nazw

Tutaj znajdują się wszystkie przestrzenie nazw wraz z ich krótkimi opisami:

EthanTheHero . . . . . ??

Rozdział 2

Indeks hierarchiczny

2.1 Hierarchia klas

Ta lista dziedziczenia posortowana jest z grubsza, choć nie całkowicie, alfabetycznie:

IPointerEnterHandler
- ButtonHoverSound . . . ??
- ButtonHoverUnderline . . . ??

IPointerExitHandler
- ButtonHoverUnderline . . . ??

ISaveable . . . ??
- BarrelSaveData . . . ??
- PlayerSaveData . . . ??

ISerializationCallbackReceiver
- SerializationWrapper . . . ??

LetterData . . . ??

MonoBehaviour
- AttackController . . . ??
- AttackHitbox . . . ??
- AxeTrap . . . ??
- Barrel . . . ??
- BarrelSaveData . . . ??
- Bootstrapper . . . ??
- ButtonHoverSound . . . ??
- ButtonHoverUnderline . . . ??
- CameraController . . . ??
- ChestController . . . ??
- ChestPanelManager . . . ??
- DoorTrigger . . . ??
- EncounteredGhostDialog . . . ??
- EndGameTrigger . . . ??
- EthanTheHero.PlayerAnimation . . . ??
- EthanTheHero.PlayerAttackMethod . . . ??
- EthanTheHero.PlayerMovement . . . ??
- EventSystemController . . . ??
- GhostCameraFollow . . . ??
- GhostFloating . . . ??
- GhostFollow . . . ??
- HUDVisibility . . . ??
- Health . . . ??

PlayerHealth . . . . . ??
HealthCollectible . . . . . ??
HintArea . . . . . ??
HintController . . . . . ??
IntroController . . . . . ??
InventoryManager . . . . . ??
InventorySlotSpawner . . . . . ??
InventoryUI . . . . . ??
LavaDamage . . . . . ??
LeafRevealer . . . . . ??
Level1FadeIn . . . . . ??
LeverRiddle . . . . . ??
LeverTrigger . . . . . ??
MainMenu . . . . . ??
MeleeEnemy . . . . . ??
MusicManager . . . . . ??
PatrolEnemy . . . . . ??
PauseMenu . . . . . ??
PersistentAudioListener . . . . . ??
PlayerSaveData . . . . . ??
SaveableObject . . . . . ??
SetOrderInLayerForChildren . . . . . ??
SoundManager . . . . . ??
SpikesTrap . . . . . ??
TextUI . . . . . ??
TriggerThoughtZone . . . . . ??
UIManager . . . . . ??
PlayerData . . . . . ??
SaveSystem.SaveData . . . . . ??
SaveSystem.SaveEntry . . . . . ??
ScriptableObject
AttackData . . . . . ??
EthanTheHero.PlayerMovementData . . . . . ??

Rozdział 3

Indeks klas

3.1 Lista klas

Tutaj znajdują się klasy, struktury, unie i interfejsy wraz z ich krótkimi opisami:

AttackController
: Klasa odpowiedzialna za zarządzanie atakiem postaci (tworzeniem hitboxów, cooldownem) . . ??

AttackData
: ScriptableObject przechowujące dane dotyczące ataku . . . . . ??

AttackHitbox
: Odpowiada za detekcję kolizji i zadawanie obrażeń . . . . . ??

AxeTrap
: Obsługuje ruchome pułapki typu wahadłowy topór . . . . . ??

Barrel
: Obsługuje niszczenie beczki po uderzeniu atakiem gracza . . . . . ??

BarrelSaveData
: Odpowiada za trwałe przechowywanie informacji o zniszczonych beczkach między sesjami gry ??

Bootstrapper
: Ładuje sceny startowe gry przy uruchomieniu aplikacji . . . . . ??

ButtonHoverSound
: Odtwarza dźwięk najechania na przycisk w interfejsie użytkownika . . . . . ??

ButtonHoverUnderline
: Dodaje i usuwa podkreślenie tekstu przycisku podczas najechania kursorem myszy . . . . . ??

CameraController
: Klasa odpowiadająca za płynne podążanie kamery za graczem . . . . . ??

ChestController
: Otwiera skrzynię po interakcji gracza i dodaje przedmiot (list) do ekwipunku . . . . . ??

ChestPanelManager
: Globalny menedżer odpowiedzialny za stan panelu skrzyni . . . . . ??

DoorTrigger
: Skrypt obsługujący teleportację gracza po naciśnięciu klawisza, gdy znajduje się przy drzwiach ??

EncounteredGhostDialog
: Obsługuje dialog pomiędzy graczem a napotkanym duszkiem . . . . . ??

EndGameTrigger
: Obsługuje zakończenie gry po dotarciu gracza do punktu końcowego . . . . . ??

EventSystemController
: Kontroluje istnienie tylko jednego aktywnego EventSystemu po zmianie sceny . . . . . ??

GhostCameraFollow
: Prosty skrypt do podążania obiektu (np. kamery) za wskazanym celem . . . . . ??

GhostFloating
: Odpowiada za animację 'unoszenia się' duszka w górę i w dół . . . . . ??

GhostFollow
Skrypt do płynnego śledzenia gracza przez duszka . . . . . ??
Health
Bazowa klasa obsługująca zdrowie dla wszelkich istot w grze . . . . . ??
HealthCollectible
Skrypt odpowiedzialny za przedmiot przywracający zdrowie graczowi . . . . . ??
HintArea
Wyświetla wiadomość podpowiedzi, gdy gracz znajdzie się w określonym obszarze . . . . . ??
HintController
Zarządza wyświetlaniem tekstowych podpowiedzi w interfejsie użytkownika . . . . . ??
HUDVisibility
Ukrywa interfejs HUD w scenie menu głównego . . . . . ??
IntroController
Odpowiada za odtwarzanie sekwencji wprowadzenia do gry (intro) . . . . . ??
InventoryManager
Zarządza systemem ekwipunku gracza . . . . . ??
InventorySlotSpawner
Generuje dynamiczne sloty listów w UI ekwipunku . . . . . ??
InventoryUI
Zarządza interfejsem ekwipunku oraz wyświetlaniem treści listów . . . . . ??
ISaveable
Interfejs do obsługi zapisu i odczytu stanu obiektów w grze . . . . . ??
LavaDamage
Skrypt zadający obrażenia graczowi przy kontakcie z lawą . . . . . ??
LeafRevealer
Odsłania ukrytą lokalizację, gdy gracz wejdzie w obszar kolizji z liśćmi . . . . . ??
LetterData
Dane pojedynczego listu kolekcjonerskiego . . . . . ??
Level1FadeIn
Realizuje efekt płynnego zanikania czarnego ekranu po uruchomieniu poziomu . . . . . ??
LeverRiddle
Sprawdza poprawność ułożenia dźwigni i aktywuje ukrytą platformę . . . . . ??
LeverTrigger
Obsługuje interakcję gracza z dźwignią i zmienia jej stan . . . . . ??
MainMenu
Obsługuje przyciski menu głównego: rozpoczęcie gry, wczytanie stanu i wyjście . . . . . ??
MeleeEnemy
Skrypt przeciwnika atakującego wręcz, wykrywającego gracza za pomocą BoxCast . . . . . ??
MusicManager
Globalny menedżer muzyki i narracji działający między scenami . . . . . ??
PatrolEnemy
Skrypt odpowiedzialny za patrolowanie przeciwnika między dwoma punktami . . . . . ??
PauseMenu
Zarządza stanem pauzy i końca gry w trakcie rozgrywki . . . . . ??
PersistentAudioListener
Zapewnia, że w scenie znajduje się tylko jeden aktywny `AudioListener` . . . . . ??
EthanTheHero.PlayerAnimation
Steruje animacjami gracza na podstawie jego stanu ruchu i fizyki . . . . . ??
EthanTheHero.PlayerAttackMethod
Obsługuje podstawowy system ataku postaci gracza (combo 3-atakowe) . . . . . ??
PlayerData
Struktura danych zawierająca informacje do zapisu o graczu . . . . . ??
PlayerHealth
Klasa zarządzająca zdrowiem i wytrzymałością gracza . . . . . ??
EthanTheHero.PlayerMovement
Odpowiada za ruch postaci gracza (bieganie, skok, dash, wall slide, wall jump) . . . . . ??
EthanTheHero.PlayerMovementData
ScriptableObject przechowujący dane konfiguracyjne ruchu gracza . . . . . ??

PlayerSaveData
Klasa odpowiedzialna za zapis i odczyt stanu gracza . . . . . ??
SaveableObject
Przypisuje unikalny identyfikator (`UniqueId`) do obiektu gry, aby umożliwić jego zapis i odtworzenie . . . . . ??
SaveSystem.SaveData
(Nieużywana) Lista wpisów `SaveEntry` — używana w alternatywnym podejściu . . . . . ??
SaveSystem.SaveEntry
(Nieużywana) Struktura zapisu jednego obiektu z danymi jako JSON . . . . . ??
SerializationWrapper
Pomocnicza klasa do serializacji słownika `Dictionary<string, object>` w Unity . . ??
SetOrderInLayerForChildren
Ustawia warstwę rysowania (`sortingOrder`) dla wszystkich dzieci posiadających komponent SpriteRenderer . . . . . ??
SoundManager
Centralny menedżer efektów dźwiękowych w grze . . . . . ??
SpikesTrap
Pułapka kolców zadająca obrażenia i odpychająca gracza po wejściu w trigger . . . . . ??
TextUI
Wyświetla tymczasowe wiadomości dialogowe na ekranie w komponentach TextMeshProUGUI ??
TriggerThoughtZone
Wyświetla myśl bohatera po wejściu do specjalnej strefy w grze . . . . . ??
UIManager
Singleton odpowiedzialny za zarządzanie elementami interfejsu użytkownika w całej grze . . . ??

Rozdział 4

Indeks plików

4.1 Lista plików

Tutaj znajduje się lista wszystkich plików wraz z ich krótkimi opisami:

Assets/Scripts/Attack/AttackController.cs . . . ??
Assets/Scripts/Attack/AttackData.cs . . . ??
Assets/Scripts/Attack/AttackHitbox.cs . . . ??
Assets/Scripts/Core/CameraController.cs . . . ??
Assets/Scripts/Core/EventSystemController.cs . . . ??
Assets/Scripts/Core/GhostCameraFollow.cs . . . ??
Assets/Scripts/Doors/DoorTrigger.cs . . . ??
Assets/Scripts/Enemy/MeleeEnemy.cs . . . ??
Assets/Scripts/Enemy/PatrolEnemy.cs . . . ??
Assets/Scripts/Ghost/EncounteredGhostDialog.cs . . . ??
Assets/Scripts/Ghost/GhostFloating.cs . . . ??
Assets/Scripts/Ghost/GhostFollow.cs . . . ??
Assets/Scripts/Health/Health.cs . . . ??
Assets/Scripts/Health/HealthCollectible.cs . . . ??
Assets/Scripts/Health/LavaDamage.cs . . . ??
Assets/Scripts/Health/PlayerHealth.cs . . . ??
Assets/Scripts/Interactive/Barrel.cs . . . ??
Assets/Scripts/Interactive/LeverRiddle.cs . . . ??
Assets/Scripts/Interactive/LeverTrigger.cs . . . ??
Assets/Scripts/Inventory/ChestController.cs . . . ??
Assets/Scripts/Inventory/InventoryManager.cs . . . ??
Assets/Scripts/Inventory/InventorySlotSpawner.cs . . . ??
Assets/Scripts/Inventory/InventoryUI.cs . . . ??
Assets/Scripts/Inventory/LetterData.cs . . . ??
Assets/Scripts/Map and locations/LeafRevealer.cs . . . ??
Assets/Scripts/Map and locations/SetOrderInLayerForChildren.cs . . . ??
Assets/Scripts/Menu HUDs/Bootstrapper.cs . . . ??
Assets/Scripts/Menu HUDs/ButtonHoverUnderline.cs . . . ??
Assets/Scripts/Menu HUDs/ChestPanelManager.cs . . . ??
Assets/Scripts/Menu HUDs/EndGameTrigger.cs . . . ??
Assets/Scripts/Menu HUDs/HintArea.cs . . . ??
Assets/Scripts/Menu HUDs/HintController.cs . . . ??
Assets/Scripts/Menu HUDs/HUDVisibility.cs . . . ??
Assets/Scripts/Menu HUDs/Level1FadeIn.cs . . . ??
Assets/Scripts/Menu HUDs/MainMenu.cs . . . ??

Assets/Scripts/Menu HUDs/PauseMenu.cs . . . ??
Assets/Scripts/Menu HUDs/TextUI.cs . . . ??
Assets/Scripts/Menu HUDs/TriggerThoughtZone.cs . . . ??
Assets/Scripts/Menu HUDs/UIManager.cs . . . ??
Assets/Scripts/Menu HUDs/UIStateManager.cs . . . ??
Assets/Scripts/Music&Sound/ButtonHoverSound.cs . . . ??
Assets/Scripts/Music&Sound/IntroController.cs . . . ??
Assets/Scripts/Music&Sound/MusicManager.cs . . . ??
Assets/Scripts/Music&Sound/PersistentAudioListener.cs . . . ??
Assets/Scripts/Music&Sound/SoundManager.cs . . . ??
Assets/Scripts/Player/PlayerAnimation.cs . . . ??
Assets/Scripts/Player/PlayerAttackMethod.cs . . . ??
Assets/Scripts/Player/PlayerMovement/PlayerMovement.cs . . . ??
Assets/Scripts/Player/PlayerMovement/PlayerMovementData.cs . . . ??
Assets/Scripts/Save/BarrelSavaData.cs . . . ??
Assets/Scripts/Save/ISaveable.cs . . . ??
Assets/Scripts/Save/PlayerSaveData.cs . . . ??
Assets/Scripts/Save/SaveableObject.cs . . . ??
Assets/Scripts/Save/SaveSystem.cs . . . ??
Assets/Scripts/Save/SerializationWrapper.cs . . . ??
Assets/Scripts/Traps/AxeTrap.cs . . . ??
Assets/Scripts/Traps/SpikesTrap.cs . . . ??

Rozdział 5

Dokumentacja przestrzeni nazw

5.1 Dokumentacja przestrzeni nazw EthanTheHero

Komponenty

- class PlayerAnimation

  Steruje animacjami gracza na podstawie jego stanu ruchu i fizyki.
- class PlayerAttackMethod

  Obsługuje podstawowy system ataku postaci gracza (combo 3-atakowe).
- class PlayerMovement

  Odpowiada za ruch postaci gracza (bieganie, skok, dash, wall slide, wall jump).
- class PlayerMovementData

  ScriptableObject przechowujący dane konfiguracyjne ruchu gracza.

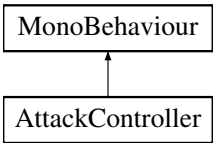
Rozdział 6

Dokumentacja klas

6.1 Dokumentacja klasy AttackController

Klasa odpowiedzialna za zarządzanie atakiem postaci (tworzeniem hitboxów, cooldownem).

Diagram dziedziczenia dla AttackController



Metody publiczne

- void PerformAttack (AttackData data)

  Wywołuje atak na podstawie danych z AttackData.

6.1.1 Opis szczegółowy

Klasa odpowiedzialna za zarządzanie atakiem postaci (tworzeniem hitboxów, cooldownem).

Wykorzystywana razem z obiektami typu AttackData, zawiera logikę instancjowania prefabów ataku oraz obsługę czasu odnowienia między atakami.

Autor

Filip Kudła

6.1.2 Dokumentacja funkcji składowych

6.1.2.1 PerformAttack()

```
void AttackController.PerformAttack (
            AttackData data)  [inline]
```

Wywołuje atak na podstawie danych z AttackData.

Parametry

	data	Obiekt przechowujący informacje o ataku (prefab, obrażenia, knockback, cooldown).

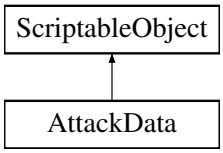
Dokumentacja dla tej klasy została wygenerowana z pliku:

- Assets/Scripts/Attack/AttackController.cs

6.2 Dokumentacja klasy AttackData

ScriptableObject przechowujące dane dotyczące ataku.

Diagram dziedziczenia dla AttackData



Atrybuty publiczne

- string attackName

  Nazwa ataku.
- float damage

  Ilość obrażeń zadawanych przez atak.
- float knockback

  Siła odrzutu zadawana przeciwnikowi.
- float cooldown

  Czas odnowienia ataku w sekundach.
- GameObject hitboxPrefab

  Prefab obiektu hitboxa generowanego w momencie ataku.
- float duration

  Czas trwania hitboxa w sekundach.

6.2.1 Opis szczegółowy

ScriptableObject przechowujące dane dotyczące ataku.

Ten obiekt zawiera informacje o nazwie ataku, jego obrażeniach, sile odrzutu, czasie odnowienia, oraz prefabie hitboxa i czasie jego trwania. Używany w systemie walki do konfiguracji poszczególnych ataków.

Autor

Filip Kudła

6.2.2 Dokumentacja atrybutów składowych

6.2.2.1 attackName

```
string AttackData.attackName
```

Nazwa ataku.

6.2.2.2 cooldown

```
float AttackData.cooldown
```

Czas odnowienia ataku w sekundach.

6.2.2.3 damage

```
float AttackData.damage
```

Ilość obrażeń zadawanych przez atak.

6.2.2.4 duration

```
float AttackData.duration
```

Czas trwania hitboxa w sekundach.

6.2.2.5 hitboxPrefab

```
GameObject AttackData.hitboxPrefab
```

Prefab obiektu hitboxa generowanego w momencie ataku.

6.2.2.6 knockback

```
float AttackData.knockback
```

Siła odrzutu zadawana przeciwnikowi.

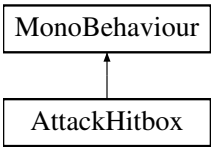
Dokumentacja dla tej klasy została wygenerowana z pliku:

- Assets/Scripts/Attack/AttackData.cs

6.3 Dokumentacja klasy AttackHitbox

Odpowiada za detekcję kolizji i zadawanie obrażeń.

Diagram dziedziczenia dla AttackHitbox



Metody publiczne

- void Init (float dmg, float kb, GameObject source)

  Inicjalizuje hitbox danymi ataku.

6.3.1 Opis szczegółowy

Odpowiada za detekcję kolizji i zadawanie obrażeń.

Tworzony dynamicznie prefab hitboxa, który sprawdza kolizję z przeciwnikiem. Gdy wykryje obiekt z komponentem Health, zadaje mu obrażenia.

Autor

Filip Kudła

6.3.2 Dokumentacja funkcji składowych

6.3.2.1 Init()

```
void AttackHitbox.Init (
            float dmg,
            float kb,
            GameObject source)  [inline]
```

Inicjalizuje hitbox danymi ataku.

Parametry

	dmg	Obrażenia do zadania.
	kb	Siła odrzutu.
	source	Obiekt będący właścicielem ataku.

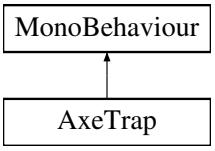
Dokumentacja dla tej klasy została wygenerowana z pliku:

- Assets/Scripts/Attack/AttackHitbox.cs

6.4 Dokumentacja klasy AxeTrap

Obsługuje ruchome pułapki typu wahadłowy topór.

Diagram dziedziczenia dla AxeTrap



6.4.1 Opis szczegółowy

Obsługuje ruchome pułapki typu wahadłowy topór.

Topór porusza się jak fizyczne wahadło (z tłumieniem) pomiędzy dwoma wychyleniami. Po kolizji z graczem zadaje obrażenia.

Autor

Filip Kudla

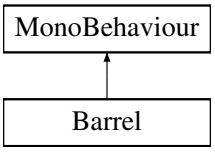
Dokumentacja dla tej klasy została wygenerowana z pliku:

- Assets/Scripts/Traps/AxeTrap.cs

6.5 Dokumentacja klasy Barrel

Obsługuje niszczenie beczki po uderzeniu atakiem gracza.

Diagram dziedziczenia dla Barrel



6.5.1 Opis szczegółowy

Obsługuje niszczenie beczki po uderzeniu atakiem gracza.

Po kolizji z atakiem, uruchamia animację zniszczenia, odtwarza dźwięk i zapisuje stan w systemie zapisu. Beczka jest niszczona po 0.9s, co pozwala dokończyć animację.

Autor

Julia Bigaj

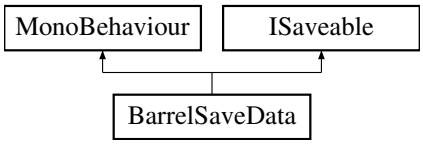
Dokumentacja dla tej klasy została wygenerowana z pliku:

- Assets/Scripts/Interactive/Barrel.cs

6.6 Dokumentacja klasy BarrelSaveData

Odpowiada za trwałe przechowywanie informacji o zniszczonych beczkach między sesjami gry.

Diagram dziedziczenia dla BarrelSaveData



Metody publiczne

- object CaptureState ()

  Zapisuje listę zniszczonych beczek.
- void RestoreState (object state)

  Przywraca stan listy zniszczonych beczek.

Statyczne metody publiczne

- static void RegisterDestroyedBarrel (string uniqueId)

  Rejestruje beczkę jako zniszczoną w pamięci podręcznej.

6.6.1 Opis szczegółowy

Odpowiada za trwałe przechowywanie informacji o zniszczonych beczkach między sesjami gry.

Implementuje interfejs `ISaveable` do integracji z globalnym systemem zapisu stanu gry. Przechowuje unikalne identyfikatory (`UniqueId`) zniszczonych beczek i usuwa je z sceny podczas wczytywania.

Beczki powinny mieć komponent `SaveableObject` z przypisanym `UniqueId`.

Autor

Filip Kudła

6.6.2 Dokumentacja funkcji składowych

6.6.2.1 CaptureState()

```
object BarrelSaveData.CaptureState ()  [inline]
```

Zapisuje listę zniszczonych beczek.

Zwraca

Obiekt zawierający listę identyfikatorów zniszczonych beczek.

Implementuje ISaveable.

6.6.2.2 RegisterDestroyedBarrel()

```
void BarrelSaveData.RegisterDestroyedBarrel (
            string uniqueId)  [inline], [static]
```

Rejestruje beczkę jako zniszczoną w pamięci podręcznej.

Parametry

	uniqueId	Unikalny identyfikator obiektu beczki.

6.6.2.3 RestoreState()

```
void BarrelSaveData.RestoreState (
            object state)  [inline]
```

Przywraca stan listy zniszczonych beczek.

Parametry

	state	Obiekt zawierający listę identyfikatorów z poprzedniego zapisu.

Implementuje ISaveable.

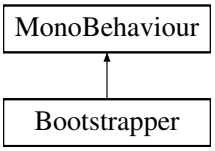
Dokumentacja dla tej klasy została wygenerowana z pliku:

- Assets/Scripts/Save/BarrelSavaData.cs

6.7 Dokumentacja klasy Bootstrapper

Ładuje sceny startowe gry przy uruchomieniu aplikacji.

Diagram dziedziczenia dla Bootstrapper



6.7.1 Opis szczegółowy

Ładuje sceny startowe gry przy uruchomieniu aplikacji.

Skrypt sprawdza, czy scena inicjalizacyjna (`InitScene`) jest załadowana. Jeśli nie — ładuje ją jako scenę dodatkową (Additive), a następnie ładuje scenę główną (`MainMenu`) jako podstawową (Single).

Używany jako punkt wejściowy aplikacji.

Autor

Julia Bigaj

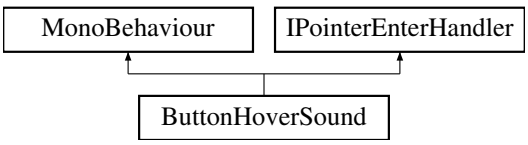
Dokumentacja dla tej klasy została wygenerowana z pliku:

- Assets/Scripts/Menu HUDs/Bootstrapper.cs

6.8 Dokumentacja klasy ButtonHoverSound

Odtwarza dźwięk najechania na przycisk w interfejsie użytkownika.

Diagram dziedziczenia dla ButtonHoverSound



Metody publiczne

- void OnPointerEnter (PointerEventData eventData)

  Reaguje na najechanie kursorem na komponent UI — odtwarza dźwięk.

Atrybuty publiczne

- AudioClip hoverSound

  Dźwięk odtwarzany przy najechaniu na przycisk.

6.8.1 Opis szczegółowy

Odtwarza dźwięk najechania na przycisk w interfejsie użytkownika.

Implementuje interfejs `IPointerEnterHandler`, aby reagować na zdarzenie najechania kursorem na element UI. Przy wejściu kursora na przycisk odtwarzany jest przypisany dźwięk typu "hover". Źródło dźwięku (`Audio↩ Source`) wyszukiwane jest dynamicznie w scenie przy starcie.

Autor

Julia Bigaj

6.8.2 Dokumentacja funkcji składowych

6.8.2.1 OnPointerEnter()

```
void ButtonHoverSound.OnPointerEnter (
            PointerEventData eventData)  [inline]
```

Reaguje na najechanie kursorem na komponent UI — odtwarza dźwięk.

Parametry

	eventData	Dane dotyczące zdarzenia wskaźnika (myszy).

6.8.3 Dokumentacja atrybutów składowych

6.8.3.1 hoverSound

```
AudioClip ButtonHoverSound.hoverSound
```

Dźwięk odtwarzany przy najechaniu na przycisk.

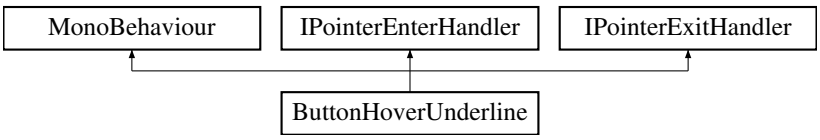
Dokumentacja dla tej klasy została wygenerowana z pliku:

- Assets/Scripts/Music&Sound/ButtonHoverSound.cs

6.9 Dokumentacja klasy ButtonHoverUnderline

Dodaje i usuwa podkreślenie tekstu przycisku podczas najechania kursorem myszy.

Diagram dziedziczenia dla ButtonHoverUnderline



Metody publiczne

- void OnPointerEnter (PointerEventData eventData)

  Dodaje podkreślenie tekstu po najechaniu kursorem.
- void OnPointerExit (PointerEventData eventData)

  Usuwa podkreślenie tekstu po opuszczeniu obszaru przycisku przez kursor.

Atrybuty publiczne

- TextMeshProUGUI label

  Referencja do komponentu tekstowego (TextMeshProUGUI), który będzie podkreślany.

6.9.1 Opis szczegółowy

Dodaje i usuwa podkreślenie tekstu przycisku podczas najechania kursorem myszy.

Skrypt wykorzystuje interfejsy `IPointerEnterHandler` i `IPointerExitHandler`, aby reagować na zdarzenia interfejsu użytkownika i umożliwiać wizualne wyróżnienie aktywnego elementu.

Autor: Julia Bigaj

6.9.2 Dokumentacja funkcji składowych

6.9.2.1 OnPointerEnter()

```
void ButtonHoverUnderline.OnPointerEnter (
            PointerEventData eventData)  [inline]
```

Dodaje podkreślenie tekstu po najechaniu kursorem.

Parametry

	eventData	Dane zdarzenia wskaźnika (myszy).

6.9.2.2 OnPointerExit()

```
void ButtonHoverUnderline.OnPointerExit (
            PointerEventData eventData)  [inline]
```

Usuwa podkreślenie tekstu po opuszczeniu obszaru przycisku przez kursor.

Parametry

	eventData	Dane zdarzenia wskaźnika (myszy).

6.9.3 Dokumentacja atrybutów składowych

6.9.3.1 label

```
TextMeshProUGUI ButtonHoverUnderline.label
```

Referencja do komponentu tekstowego (TextMeshProUGUI), który będzie podkreślany.

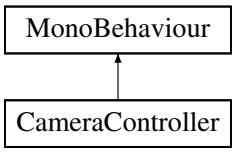
Dokumentacja dla tej klasy została wygenerowana z pliku:

- Assets/Scripts/Menu HUDs/ButtonHoverUnderline.cs

6.10 Dokumentacja klasy CameraController

Klasa odpowiadająca za płynne podążanie kamery za graczem.

Diagram dziedziczenia dla CameraController



6.10.1 Opis szczegółowy

Klasa odpowiadająca za płynne podążanie kamery za graczem.

Kamera przewiduje ruch gracza w poziomie (look ahead) oraz dynamicznie reaguje na pionowe przemieszczenia (skoki).

Autor

Filip Kudła

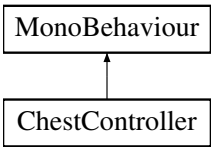
Dokumentacja dla tej klasy została wygenerowana z pliku:

- Assets/Scripts/Core/CameraController.cs

6.11 Dokumentacja klasy ChestController

Otwiera skrzynię po interakcji gracza i dodaje przedmiot (list) do ekwipunku.

Diagram dziedziczenia dla ChestController



Metody publiczne

- void CloseChestFromOutside ()

  Zamyka skrzynię z zewnętrznych skryptów.

Atrybuty publiczne

- GameObject chestPanel

  Panel UI wyświetlany po otwarciu skrzyni.
- LetterData letterData

  Dane przechowywanego listu.
- GameObject promptUI

  UI z podpowiedzią do interakcji (np. naciśnij "F").
- Image letterIcon

  Ikona reprezentująca list w UI skrzyni.
- bool isLetterTaken = false

  Czy list został już zabrany przez gracza.

6.11.1 Opis szczegółowy

Otwiera skrzynię po interakcji gracza i dodaje przedmiot (list) do ekwipunku.

Odpowiada za animację otwierania skrzyni, interakcję z graczem, wyświetlanie UI oraz dodanie listu do systemu ekwipunku.

Autor

Julia Bigaj

6.11.2 Dokumentacja funkcji składowych

6.11.2.1 CloseChestFromOutside()

```
void ChestController.CloseChestFromOutside ()  [inline]
```

Zamyka skrzynię z zewnętrznych skryptów.

6.11.3 Dokumentacja atrybutów składowych

6.11.3.1 chestPanel

```
GameObject ChestController.chestPanel
```

Panel UI wyświetlany po otwarciu skrzyni.

6.11.3.2 isLetterTaken

```
bool ChestController.isLetterTaken = false
```

Czy list został już zabrany przez gracza.

6.11.3.3 letterData

```
LetterData ChestController.letterData
```

Dane przechowywanego listu.

6.11.3.4 letterIcon

```
Image ChestController.letterIcon
```

Ikona reprezentująca list w UI skrzyni.

6.11.3.5 promptUI

```
GameObject ChestController.promptUI
```

UI z podpowiedzią do interakcji (np. naciśnij "F").

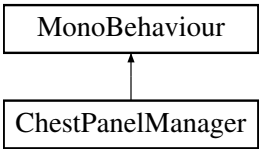
Dokumentacja dla tej klasy została wygenerowana z pliku:

- Assets/Scripts/Inventory/ChestController.cs

6.12 Dokumentacja klasy ChestPanelManager

Globalny menedżer odpowiedzialny za stan panelu skrzyni.

Diagram dziedziczenia dla ChestPanelManager



Metody publiczne

- bool IsChestOpen ()

  Sprawdza, czy skrzynia jest aktualnie otwarta.
- void CloseChest ()

  Zamyka skrzynię i jej panel z poziomu zewnętrznych systemów (np. menu pauzy).

Atrybuty publiczne

- GameObject chestPanel

  Referencja do panelu skrzyni w UI.
- ChestController chestController

  Referencja do skryptu kontrolującego logikę skrzyni.

Statyczne atrybuty publiczne

- static ChestPanelManager Instance

  Statyczna instancja singletonu ChestPanelManager.

6.12.1 Opis szczegółowy

Globalny menedżer odpowiedzialny za stan panelu skrzyni.

Umożliwia sprawdzenie, czy skrzynia jest aktualnie otwarta, oraz jej zamknięcie z zewnętrznych skryptów (np. z menu pauzy). Implementuje wzorzec singletonu do globalnego dostępu.

Autor

Julia Bigaj

6.12.2 Dokumentacja funkcji składowych

6.12.2.1 CloseChest()

```
void ChestPanelManager.CloseChest () [inline]
```

Zamyka skrzynię i jej panel z poziomu zewnętrznych systemów (np. menu pauzy).

6.12.2.2 IsChestOpen()

```
bool ChestPanelManager.IsChestOpen () [inline]
```

Sprawdza, czy skrzynia jest aktualnie otwarta.

Zwraca

`true` jeśli panel skrzyni jest aktywny, w przeciwnym razie `false`.

6.12.3 Dokumentacja atrybutów składowych

6.12.3.1 chestController

```
ChestController ChestPanelManager.chestController
```

Referencja do skryptu kontrolującego logikę skrzyni.

6.12.3.2 chestPanel

```
GameObject ChestPanelManager.chestPanel
```

Referencja do panelu skrzyni w UI.

6.12.3.3 Instance

```
ChestPanelManager ChestPanelManager.Instance  [static]
```

Statyczna instancja singletonu ChestPanelManager.

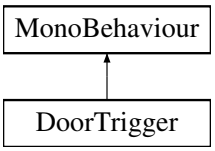
Dokumentacja dla tej klasy została wygenerowana z pliku:

- Assets/Scripts/Menu HUDs/ChestPanelManager.cs

6.13 Dokumentacja klasy DoorTrigger

Skrypt obsługujący teleportację gracza po naciśnięciu klawisza, gdy znajduje się przy drzwiach.

Diagram dziedziczenia dla DoorTrigger



Atrybuty publiczne

- GameObject promptText
- Sprite openDoorSprite
- Transform teleportTarget

6.13.1 Opis szczegółowy

Skrypt obsługujący teleportację gracza po naciśnięciu klawisza, gdy znajduje się przy drzwiach.

Po wejściu gracza w obszar drzwi, pojawia się tekst podpowiedzi. Jeśli gracz naciśnie F, gracz zostaje teleportowany, a drzwi zmieniają wygląd.

Autor

Julia Bigaj

6.13.2 Dokumentacja atrybutów składowych

6.13.2.1 openDoorSprite

```
Sprite DoorTrigger.openDoorSprite
```

6.13.2.2 promptText

```
GameObject DoorTrigger.promptText
```

6.13.2.3 teleportTarget

```
Transform DoorTrigger.teleportTarget
```

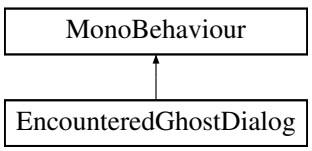
Dokumentacja dla tej klasy została wygenerowana z pliku:

- Assets/Scripts/Doors/DoorTrigger.cs

6.14 Dokumentacja klasy EncounteredGhostDialog

Obsługuje dialog pomiędzy graczem a napotkanym duszkiem.

Diagram dziedziczenia dla EncounteredGhostDialog



Atrybuty publiczne

- string dialog1 = "Kim jesteś?"
- string dialog2 = "Kimś kto pomoże ci wydostać się z tego zamku. Tak się składa, że jestem jednym z mieszkańców"
- string dialog3 = "Tak po prostu mi pomożesz? Nie mam nic co mógłbym ci dać w zamian"
- string dialog4 = "Masz ale jeszcze o tym nie wiesz. Chodź, musimy sie spieszyć"
- AudioClip dialog1Audio
- AudioClip dialog2Audio
- AudioClip dialog3Audio
- AudioClip dialog4Audio
- GameObject encounteredGhost

  Duszek do ukrycia po zakończeniu dialogu.
- GameObject ghostWithCam

  Duszek z kamerą do pokazania po dialogu.

6.14.1 Opis szczegółowy

Obsługuje dialog pomiędzy graczem a napotkanym duszkiem.

Wyświetla sekwencję tekstów z opcjonalnym dźwiękiem, zatrzymując ruch gracza. Po zakończeniu dialogu jeden duszek znika, a inny (z kamerą) się pojawia.

Autor

Filip Kudła

6.14.2 Dokumentacja atrybutów składowych

6.14.2.1 dialog1

```
string EncounteredGhostDialog.dialog1 = "Kim jesteś?"
```

6.14.2.2 dialog1Audio

```
AudioClip EncounteredGhostDialog.dialog1Audio
```

6.14.2.3 dialog2

```
string EncounteredGhostDialog.dialog2 = "Kimś kto pomoże ci wydostać się z tego zamku. Tak się składa, że jestem jednym z mieszkańców"
```

6.14.2.4 dialog2Audio

```
AudioClip EncounteredGhostDialog.dialog2Audio
```

6.14.2.5 dialog3

```
string EncounteredGhostDialog.dialog3 = "Tak po prostu mi pomożesz? Nie mam nic co mógłbym ci dać w zamian"
```

6.14.2.6 dialog3Audio

```
AudioClip EncounteredGhostDialog.dialog3Audio
```

6.14.2.7 dialog4

```
string EncounteredGhostDialog.dialog4 = "Masz ale jeszcze o tym nie wiesz. Chodź, musimy sie spieszyć"
```

6.14.2.8 dialog4Audio

```
AudioClip EncounteredGhostDialog.dialog4Audio
```

6.14.2.9 encounteredGhost

```
GameObject EncounteredGhostDialog.encounteredGhost
```

Duszek do ukrycia po zakończeniu dialogu.

6.14.2.10 ghostWithCam

```
GameObject EncounteredGhostDialog.ghostWithCam
```

Duszek z kamerą do pokazania po dialogu.

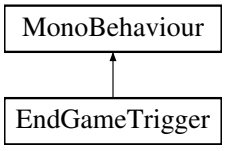
Dokumentacja dla tej klasy została wygenerowana z pliku:

- Assets/Scripts/Ghost/EncounteredGhostDialog.cs

6.15 Dokumentacja klasy EndGameTrigger

Obsługuje zakończenie gry po dotarciu gracza do punktu końcowego.

Diagram dziedziczenia dla EndGameTrigger



Atrybuty publiczne

- CanvasGroup fadeOverlay

  Referencja do `CanvasGroup` odpowiedzialnego za przyciemnienie ekranu.
- float fadeDuration = 1.5f

  Czas trwania animacji przyciemnienia.
- string mainMenuSceneName = "MainMenu"

  Nazwa sceny głównego menu, do której następuje powrót po zakończeniu gry.

6.15.1 Opis szczegółowy

Obsługuje zakończenie gry po dotarciu gracza do punktu końcowego.

Po wejściu gracza w trigger, rozpoczyna animację przyciemniania ekranu (poprzez zmianę `alpha` komponentu `CanvasGroup`) i ładuje scenę głównego menu.

Wymaga na scenie obiektu z czarnym `Image` (UI) oraz komponentem `CanvasGroup`.

Autor

Julia Bigaj

6.15.2 Dokumentacja atrybutów składowych

6.15.2.1 fadeDuration

```
float EndGameTrigger.fadeDuration = 1.5f
```

Czas trwania animacji przyciemnienia.

6.15.2.2 fadeOverlay

```
CanvasGroup EndGameTrigger.fadeOverlay
```

Referencja do `CanvasGroup` odpowiedzialnego za przyciemnienie ekranu.

6.15.2.3 mainMenuSceneName

```
string EndGameTrigger.mainMenuSceneName = "MainMenu"
```

Nazwa sceny głównego menu, do której następuje powrót po zakończeniu gry.

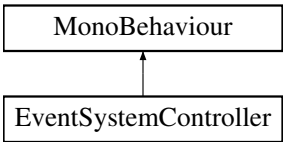
Dokumentacja dla tej klasy została wygenerowana z pliku:

- Assets/Scripts/Menu HUDs/EndGameTrigger.cs

6.16 Dokumentacja klasy EventSystemController

Kontroluje istnienie tylko jednego aktywnego EventSystemu po zmianie sceny.

Diagram dziedziczenia dla EventSystemController



6.16.1 Opis szczegółowy

Kontroluje istnienie tylko jednego aktywnego EventSystemu po zmianie sceny.

Skrypt zapobiega duplikowaniu EventSystemów po wczytywaniu scen i dba o ich poprawną aktywację.

Autor

Julia Bigaj

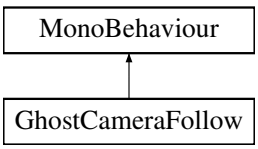
Dokumentacja dla tej klasy została wygenerowana z pliku:

- Assets/Scripts/Core/EventSystemController.cs

6.17 Dokumentacja klasy GhostCameraFollow

Prosty skrypt do podążania obiektu (np. kamery) za wskazanym celem.

Diagram dziedziczenia dla GhostCameraFollow



Atrybuty publiczne

- Transform mainCamera

6.17.1 Opis szczegółowy

Prosty skrypt do podążania obiektu (np. kamery) za wskazanym celem.

Może być użyty przez ducha lub inny obiekt śledzący pozycję gracza.

Autor

Filip Kudła

6.17.2 Dokumentacja atrybutów składowych

6.17.2.1 mainCamera

```
Transform GhostCameraFollow.mainCamera
```

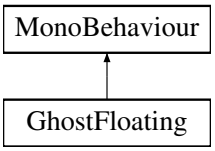
Dokumentacja dla tej klasy została wygenerowana z pliku:

- Assets/Scripts/Core/GhostCameraFollow.cs

6.18 Dokumentacja klasy GhostFloating

Odpowiada za animację 'unoszenia się' duszka w górę i w dół.

Diagram dziedziczenia dla GhostFloating



Atrybuty publiczne

- float smoothSpeed = 2f

  Prędkość płynnego przejścia do nowej pozycji.
- float floatAmplitude = 0.5f

  Amplituda unoszenia się w osi Y.
- float floatFrequency = 1f

  Częstotliwość unoszenia się.

6.18.1 Opis szczegółowy

Odpowiada za animację 'unoszenia się' duszka w górę i w dół.

Skrypt dodaje efekt pływania dla obiektu (najczęściej duszka), poruszając go sinusoidalnie w osi Y. Może być używany np. do ozdobnych duszków czy elementów interaktywnych.

Autor

Filip Kudła

6.18.2 Dokumentacja atrybutów składowych

6.18.2.1 floatAmplitude

```
float GhostFloating.floatAmplitude = 0.5f
```

Amplituda unoszenia się w osi Y.

6.18.2.2 floatFrequency

```
float GhostFloating.floatFrequency = 1f
```

Częstotliwość unoszenia się.

6.18.2.3 smoothSpeed

```
float GhostFloating.smoothSpeed = 2f
```

Prędkość płynnego przejścia do nowej pozycji.

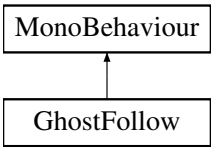
Dokumentacja dla tej klasy została wygenerowana z pliku:

- Assets/Scripts/Ghost/GhostFloating.cs

6.19 Dokumentacja klasy GhostFollow

Skrypt do płynnego śledzenia gracza przez duszka.

Diagram dziedziczenia dla GhostFollow



Atrybuty publiczne

- Transform player

  Transform gracza do śledzenia.
- Vector3 offset

  Względna pozycja duszka nad graczem.
- float smoothSpeed = 2f

  Prędkość płynnego podążania.
- float floatAmplitude = 0.5f

  Amplituda 'pływania' duszka.
- float floatFrequency = 1f

  Częstotliwość pływania.

6.19.1 Opis szczegółowy

Skrypt do płynnego śledzenia gracza przez duszka.

Obiekt przypisany do tego skryptu podąża za graczem, unosząc się nad nim i zmieniając kierunek w zależności od skali gracza. Dodaje także efekt sinusoidalnego 'pływania' w osi Y.

Autor

Filip Kudła

6.19.2 Dokumentacja atrybutów składowych

6.19.2.1 floatAmplitude

```
float GhostFollow.floatAmplitude = 0.5f
```

Amplituda 'pływania' duszka.

6.19.2.2 floatFrequency

```
float GhostFollow.floatFrequency = 1f
```

Częstotliwość pływania.

6.19.2.3 offset

```
Vector3 GhostFollow.offset
```

Względna pozycja duszka nad graczem.

6.19.2.4 player

```
Transform GhostFollow.player
```

Transform gracza do śledzenia.

6.19.2.5 smoothSpeed

```
float GhostFollow.smoothSpeed = 2f
```

Prędkość płynnego podążania.

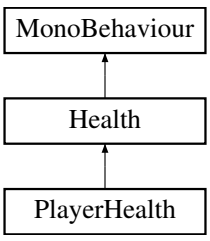
Dokumentacja dla tej klasy została wygenerowana z pliku:

- Assets/Scripts/Ghost/GhostFollow.cs

6.20 Dokumentacja klasy Health

Bazowa klasa obsługująca zdrowie dla wszelkich istot w grze.

Diagram dziedziczenia dla Health



Metody publiczne

- virtual void TakeDamage (float amount)
- virtual void Die ()
- virtual void Heal (float amount)
- virtual void SetBarsValue (float value)

Metody chronione

- virtual void Awake ()
- virtual IEnumerator DamageCooldownCoroutine ()

Atrybuty chronione

- float startingHealth

  Początkowa wartość zdrowia.
- MicroBar healthBar

  Pasek zdrowia.
- float damageCooldown = 0.75f

  Czas nieśmiertelności po otrzymaniu obrażeń.
- float currentHealth
- Animator anim
- SpriteRenderer spriteRend
- bool canTakeDamage = true

6.20.1 Opis szczegółowy

Bazowa klasa obsługująca zdrowie dla wszelkich istot w grze.

Oferuje podstawową logikę odbierania/leczenia obrażeń, obsługę nieśmiertelności oraz pasek zdrowia. Może być dziedziczona przez inne klasy (np. `PlayerHealth`, `EnemyHealth`).

Autor

Filip Kudła

6.20.2 Dokumentacja funkcji składowych

6.20.2.1 Awake()

```
virtual void Health.Awake ()  [inline], [protected], [virtual]
```

Inicjalizuje komponenty i pasek zdrowia.

Reimplementowana w PlayerHealth.

6.20.2.2 DamageCooldownCoroutine()

```
virtual IEnumerator Health.DamageCooldownCoroutine ()  [inline], [protected], [virtual]
```

Coroutine - krótki czas nieśmiertelności po otrzymaniu obrażeń.

6.20.2.3 Die()

```
virtual void Health.Die ()  [inline], [virtual]
```

Domyślna logika śmierci - usunięcie obiektu.

Reimplementowana w PlayerHealth.

6.20.2.4 Heal()

```
virtual void Health.Heal (
            float amount)  [inline], [virtual]
```

Leczy postać o podaną wartość.

6.20.2.5 SetBarsValue()

```
virtual void Health.SetBarsValue (
            float value)  [inline], [virtual]
```

Ustawia konkretną wartość na pasku zdrowia.

Reimplementowana w PlayerHealth.

6.20.2.6 TakeDamage()

```
virtual void Health.TakeDamage (
            float amount)  [inline], [virtual]
```

Odbiera obrażenia, aktualizuje pasek zdrowia i wywołuje śmierć, jeśli HP spadnie do zera.

Reimplementowana w PlayerHealth.

6.20.3 Dokumentacja atrybutów składowych

6.20.3.1 anim

```
Animator Health.anim  [protected]
```

6.20.3.2 canTakeDamage

```
bool Health.canTakeDamage = true  [protected]
```

6.20.3.3 currentHealth

```
float Health.currentHealth  [protected]
```

6.20.3.4 damageCooldown

```
float Health.damageCooldown = 0.75f  [protected]
```

Czas nieśmiertelności po otrzymaniu obrażeń.

6.20.3.5 healthBar

```
MicroBar Health.healthBar  [protected]
```

Pasek zdrowia.

6.20.3.6 spriteRend

```
SpriteRenderer Health.spriteRend  [protected]
```

6.20.3.7 startingHealth

```
float Health.startingHealth  [protected]
```

Początkowa wartość zdrowia.

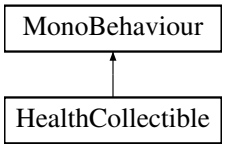
Dokumentacja dla tej klasy została wygenerowana z pliku:

- Assets/Scripts/Health/Health.cs

6.21 Dokumentacja klasy HealthCollectible

Skrypt odpowiedzialny za przedmiot przywracający zdrowie graczowi.

Diagram dziedziczenia dla HealthCollectible



6.21.1 Opis szczegółowy

Skrypt odpowiedzialny za przedmiot przywracający zdrowie graczowi.

Po zebraniu przez gracza, przywraca określoną ilość zdrowia i usuwa się ze sceny.

Autor

Filip Kudła

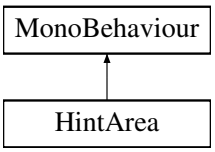
Dokumentacja dla tej klasy została wygenerowana z pliku:

- Assets/Scripts/Health/HealthCollectible.cs

6.22 Dokumentacja klasy HintArea

Wyświetla wiadomość podpowiedzi, gdy gracz znajdzie się w określonym obszarze.

Diagram dziedziczenia dla HintArea



Atrybuty publiczne

- string message

  Tekst wiadomości, która ma być wyświetlona graczowi.

6.22.1 Opis szczegółowy

Wyświetla wiadomość podpowiedzi, gdy gracz znajdzie się w określonym obszarze.

Wykrywa obecność gracza w triggerze i przekazuje tekst do `HintController`, który zarządza wyświetlaniem wiadomości na ekranie.

Wspiera mechaniki eksploracji i nawigacji poprzez wskazówki środowiskowe.

Autor

Julia Bigaj

6.22.2 Dokumentacja atrybutów składowych

6.22.2.1 message

```
string HintArea.message
```

Tekst wiadomości, która ma być wyświetlona graczowi.

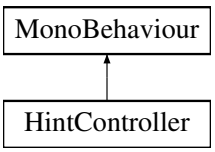
Dokumentacja dla tej klasy została wygenerowana z pliku:

- Assets/Scripts/Menu HUDs/HintArea.cs

6.23 Dokumentacja klasy HintController

Zarządza wyświetlaniem tekstowych podpowiedzi w interfejsie użytkownika.

Diagram dziedziczenia dla HintController



Metody publiczne

- void ShowHint (string message)

  Wyświetla wiadomość podpowiedzi na ekranie.
- void HideHint ()

  Ukrywa aktualnie wyświetlaną podpowiedź.

Atrybuty publiczne

- TextMeshProUGUI hintText

  Referencja do komponentu TextMeshProUGUI wyświelającego podpowiedzi.

6.23.1 Opis szczegółowy

Zarządza wyświetlaniem tekstowych podpowiedzi w interfejsie użytkownika.

Współpracuje z komponentami takimi jak `HintArea`, umożliwiając dynamiczne pokazywanie i ukrywanie wiadomości dla gracza.

Autor

Julia Bigaj

6.23.2 Dokumentacja funkcji składowych

6.23.2.1 HideHint()

```
void HintController.HideHint ()  [inline]
```

Ukrywa aktualnie wyświetlaną podpowiedź.

6.23.2.2 ShowHint()

```
void HintController.ShowHint (
            string message)  [inline]
```

Wyświetla wiadomość podpowiedzi na ekranie.

Parametry

	message	Tekst wiadomości do wyświetlenia.

6.23.3 Dokumentacja atrybutów składowych

6.23.3.1 hintText

```
TextMeshProUGUI HintController.hintText
```

Referencja do komponentu TextMeshProUGUI wyświelającego podpowiedzi.

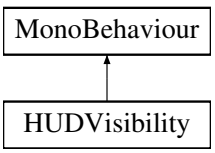
Dokumentacja dla tej klasy została wygenerowana z pliku:

- Assets/Scripts/Menu HUDs/HintController.cs

6.24 Dokumentacja klasy HUDVisibility

Ukrywa interfejs HUD w scenie menu głównego.

Diagram dziedziczenia dla HUDVisibility



6.24.1 Opis szczegółowy

Ukrywa interfejs HUD w scenie menu głównego.

Skrypt automatycznie dezaktywuje obiekt HUD, jeśli aktualnie aktywna scena to `"MainMenu"`. Zapewnia, że elementy interfejsu nie są widoczne podczas przebywania w menu.

Autor

Julia Bigaj

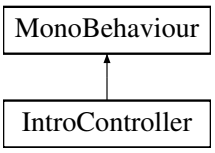
Dokumentacja dla tej klasy została wygenerowana z pliku:

- Assets/Scripts/Menu HUDs/HUDVisibility.cs

6.25 Dokumentacja klasy IntroController

Odpowiada za odtwarzanie sekwencji wprowadzenia do gry (intro).

Diagram dziedziczenia dla IntroController



Atrybuty publiczne

- GameObject image1

  Obraz nr 1 wyświetlany na początku sekwencji.
- GameObject image3

  Obraz nr 3 wyświetlany w dalszej części sekwencji.
- GameObject image4

  Obraz nr 4 wyświetlany w dalszej części sekwencji.
- GameObject image5

  Obraz nr 5 – końcowy, na którym wykonywany jest efekt zoom i fade out.
- float zoomDuration = 8f

  Czas trwania efektu zoom.
- float zoomScale = 1.3f

  Współczynnik przybliżenia końcowego obrazu.

6.25.1 Opis szczegółowy

Odpowiada za odtwarzanie sekwencji wprowadzenia do gry (intro).

Skrypt pokazuje kolejne obrazy, odtwarza narrację i dźwięki za pomocą `MusicManager`, a na końcu wykonuje efekt przybliżenia i zanikania ostatniego obrazu, po czym ładuje scenę `"Level 1 - Cave"`.

Przeznaczony do użycia wyłącznie w scenie `"Intro"`.

Autor

Julia Bigaj

6.25.2 Dokumentacja atrybutów składowych

6.25.2.1 image1

```
GameObject IntroController.image1
```

Obraz nr 1 wyświetlany na początku sekwencji.

6.25.2.2 image3

```
GameObject IntroController.image3
```

Obraz nr 3 wyświetlany w dalszej części sekwencji.

6.25.2.3 image4

```
GameObject IntroController.image4
```

Obraz nr 4 wyświetlany w dalszej części sekwencji.

6.25.2.4 image5

```
GameObject IntroController.image5
```

Obraz nr 5 – końcowy, na którym wykonywany jest efekt zoom i fade out.

6.25.2.5 zoomDuration

```
float IntroController.zoomDuration = 8f
```

Czas trwania efektu zoom.

6.25.2.6 zoomScale

```
float IntroController.zoomScale = 1.3f
```

Współczynnik przybliżenia końcowego obrazu.

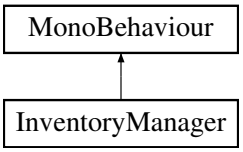
Dokumentacja dla tej klasy została wygenerowana z pliku:

- Assets/Scripts/Music&Sound/IntroController.cs

6.26 Dokumentacja klasy InventoryManager

Zarządza systemem ekwipunku gracza.

Diagram dziedziczenia dla InventoryManager



Metody publiczne

- void AddItem (string item)

  Dodaje przedmiot do listy przedmiotów.
- void RemoveItem (string item)

  Usuwa przedmiot z listy przedmiotów.
- void AddLetter (LetterData newLetter)

  Dodaje list do listy zebranych listów.

Atrybuty publiczne

- List< string > items = new List<string> ()

  Lista nazw zwykłych przedmiotów w ekwipunku.
- List< LetterData > collectedLetters = new List<LetterData>()

  Lista zebranych listów (LetterData).

Statyczne atrybuty publiczne

- static InventoryManager Instance

  Singleton umożliwiający dostęp do instancji klasy z innych skryptów.

6.26.1 Opis szczegółowy

Zarządza systemem ekwipunku gracza.

Przechowuje zebrane przedmioty oraz listy, zapewnia singleton do globalnego dostępu i pozwala na ich dodawanie i usuwanie.

Autor

Julia Bigaj

6.26.2 Dokumentacja funkcji składowych

6.26.2.1 AddItem()

```
void InventoryManager.AddItem (
            string item)  [inline]
```

Dodaje przedmiot do listy przedmiotów.

Parametry

item	Nazwa przedmiotu do dodania.

6.26.2.2 AddLetter()

```
void InventoryManager.AddLetter (
            LetterData newLetter)  [inline]
```

Dodaje list do listy zebranych listów.

Parametry

newLetter	Obiekt LetterData reprezentujący nowy list.

6.26.2.3 RemoveItem()

```
void InventoryManager.RemoveItem (
            string item)  [inline]
```

Usuwa przedmiot z listy przedmiotów.

Parametry

item	Nazwa przedmiotu do usunięcia.

6.26.3 Dokumentacja atrybutów składowych

6.26.3.1 collectedLetters

```
List<LetterData> InventoryManager.collectedLetters = new List<LetterData>()
```

Lista zebranych listów (LetterData).

6.26.3.2 Instance

```
InventoryManager InventoryManager.Instance  [static]
```

Singleton umożliwiający dostęp do instancji klasy z innych skryptów.

6.26.3.3 items

```
List<string> InventoryManager.items = new List<string> ()
```

Lista nazw zwykłych przedmiotów w ekwipunku.

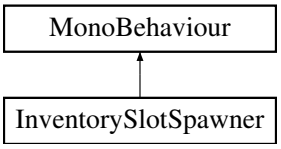
Dokumentacja dla tej klasy została wygenerowana z pliku:

- Assets/Scripts/Inventory/InventoryManager.cs

6.27 Dokumentacja klasy InventorySlotSpawner

Generuje dynamiczne sloty listów w UI ekwipunku.

Diagram dziedziczenia dla InventorySlotSpawner



Metody publiczne

- void RefreshSlots ()

  Odświeża wszystkie sloty z listami w interfejsie użytkownika.

Atrybuty publiczne

- GameObject slotPrefab

  Prefab pojedynczego slotu listu.
- Transform slotsParent

  Rodzic (kontener) dla wygenerowanych slotów.

6.27.1 Opis szczegółowy

Generuje dynamiczne sloty listów w UI ekwipunku.

Usuwa stare sloty, tworzy nowe na podstawie zebranych listów oraz przypisuje odpowiednie przyciski do wyświetlania ich zawartości.

Autor

Julia Bigaj

6.27.2 Dokumentacja funkcji składowych

6.27.2.1 RefreshSlots()

```
void InventorySlotSpawner.RefreshSlots ()  [inline]
```

Odświeża wszystkie sloty z listami w interfejsie użytkownika.

Usuwa poprzednie sloty i generuje nowe na podstawie aktualnego stanu ekwipunku.

6.27.3 Dokumentacja atrybutów składowych

6.27.3.1 slotPrefab

```
GameObject InventorySlotSpawner.slotPrefab
```

Prefab pojedynczego slotu listu.

6.27.3.2 slotsParent

```
Transform InventorySlotSpawner.slotsParent
```

Rodzic (kontener) dla wygenerowanych slotów.

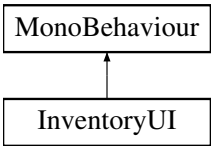
Dokumentacja dla tej klasy została wygenerowana z pliku:

- Assets/Scripts/Inventory/InventorySlotSpawner.cs

6.28 Dokumentacja klasy InventoryUI

Zarządza interfejsem ekwipunku oraz wyświetlaniem treści listów.

Diagram dziedziczenia dla InventoryUI



Metody publiczne

- void ShowLetterContent (LetterData letterData)

  Wyświetla treść wybranego listu.

Atrybuty publiczne

- GameObject inventoryPanel

  Panel główny UI ekwipunku.
- Image letterContentImage

  Obraz prezentujący treść listu.
- InventorySlotSpawner slotSpawner

  Komponent odpowiedzialny za generowanie slotów.

Statyczne atrybuty publiczne

- static InventoryUI Instance

  Singleton zapewniający dostęp do UI ekwipunku.

6.28.1 Opis szczegółowy

Zarządza interfejsem ekwipunku oraz wyświetlaniem treści listów.

Umożliwia otwieranie/zamykanie ekwipunku, odświeżanie jego zawartości oraz pokazywanie treści wybranego listu (LetterData).

Autor

Julia Bigaj

6.28.2 Dokumentacja funkcji składowych

6.28.2.1 ShowLetterContent()

```
void InventoryUI.ShowLetterContent (
            LetterData letterData)  [inline]
```

Wyświetla treść wybranego listu.

Parametry

	letter	Obiekt `LetterData`, którego treść ma zostać pokazana.

6.28.3 Dokumentacja atrybutów składowych

6.28.3.1 Instance

```
InventoryUI InventoryUI.Instance  [static]
```

Singleton zapewniający dostęp do UI ekwipunku.

6.28.3.2 inventoryPanel

```
GameObject InventoryUI.inventoryPanel
```

Panel główny UI ekwipunku.

6.28.3.3 letterContentImage

```
Image InventoryUI.letterContentImage
```

Obraz prezentujący treść listu.

6.28.3.4 slotSpawner

```
InventorySlotSpawner InventoryUI.slotSpawner
```

Komponent odpowiedzialny za generowanie slotów.

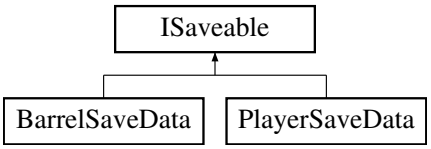
Dokumentacja dla tej klasy została wygenerowana z pliku:

- Assets/Scripts/Inventory/InventoryUI.cs

6.29 Dokumentacja interfejsu ISaveable

Interfejs do obsługi zapisu i odczytu stanu obiektów w grze.

Diagram dziedziczenia dla ISaveable



Metody publiczne

- object CaptureState ()

  Zbiera dane reprezentujące stan obiektu do zapisania.
- void RestoreState (object state)

  Przywraca stan obiektu na podstawie danych z zapisu.

6.29.1 Opis szczegółowy

Interfejs do obsługi zapisu i odczytu stanu obiektów w grze.

Każdy obiekt, który implementuje ten interfejs, może być automatycznie zapisany i odtworzony przez globalny system zapisu. Przechowywany stan musi być możliwy do serializacji.

Przykład użycia:

```
public class MyObject : MonoBehaviour, ISaveable
{
    public object CaptureState()
    {
        return myData;
    }

    public void RestoreState(object state)
    {
        myData = (int)state;
    }
}
```

Autor

Filip Kudła

6.29.2 Dokumentacja funkcji składowych

6.29.2.1 CaptureState()

```
object ISaveable.CaptureState ()
```

Zbiera dane reprezentujące stan obiektu do zapisania.

Zwraca

Obiekt serializowalny reprezentujący aktualny stan.

Implementowany w BarrelSaveData i PlayerSaveData.

6.29.2.2 RestoreState()

```
void ISaveable.RestoreState (
            object state)
```

Przywraca stan obiektu na podstawie danych z zapisu.

Parametry

	state	Stan obiektu odczytany z pliku zapisu.

Implementowany w BarrelSaveData i PlayerSaveData.

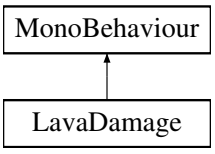
Dokumentacja dla tego interfejsu została wygenerowana z pliku:

- Assets/Scripts/Save/ISaveable.cs

6.30 Dokumentacja klasy LavaDamage

Skrypt zadający obrażenia graczowi przy kontakcie z lawą.

Diagram dziedziczenia dla LavaDamage



6.30.1 Opis szczegółowy

Skrypt zadający obrażenia graczowi przy kontakcie z lawą.

Po wejściu w strefę lawy, odejmuje graczowi określoną ilość zdrowia i odrzuca go.

Autor

Filip Kudła

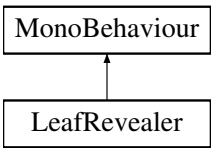
Dokumentacja dla tej klasy została wygenerowana z pliku:

- Assets/Scripts/Health/LavaDamage.cs

6.31 Dokumentacja klasy LeafRevealer

Odsłania ukrytą lokalizację, gdy gracz wejdzie w obszar kolizji z liśćmi.

Diagram dziedziczenia dla LeafRevealer



Atrybuty publiczne

- GameObject hiddenLocation

  Obiekt, który zostanie ujawniony po wejściu gracza w trigger (np. ukryta ścieżka).

6.31.1 Opis szczegółowy

Odsłania ukrytą lokalizację, gdy gracz wejdzie w obszar kolizji z liśćmi.

Skrypt przypisany do obiektu z liśćmi. Gdy gracz wejdzie w trigger, zostaje aktywowany wskazany obiekt `hidden↩Location` (np. ukryte przejście). Może być częścią zagadki logicznej lub ukrytego obszaru w grze.

Autor

Julia Bigaj

6.31.2 Dokumentacja atrybutów składowych

6.31.2.1 hiddenLocation

```
GameObject LeafRevealer.hiddenLocation
```

Obiekt, który zostanie ujawniony po wejściu gracza w trigger (np. ukryta ścieżka).

Dokumentacja dla tej klasy została wygenerowana z pliku:

- Assets/Scripts/Map and locations/LeafRevealer.cs

6.32 Dokumentacja klasy LetterData

Dane pojedynczego listu kolekcjonerskiego.

Atrybuty publiczne

- Sprite icon

  Ikona reprezentująca list.
- Sprite content

  Treść listu do wyświetlenia.

6.32.1 Opis szczegółowy

Dane pojedynczego listu kolekcjonerskiego.

Zawiera ikonę i treść listu – używane do wyświetlania w ekwipunku.

Autor

Julia Bigaj

6.32.2 Dokumentacja atrybutów składowych

6.32.2.1 content

```
Sprite LetterData.content
```

Treść listu do wyświetlenia.

6.32.2.2 icon

`Sprite LetterData.icon`

Ikona reprezentująca list.

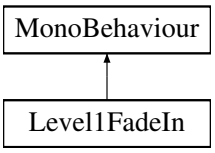
Dokumentacja dla tej klasy została wygenerowana z pliku:

- Assets/Scripts/Inventory/LetterData.cs

6.33 Dokumentacja klasy Level1FadeIn

Realizuje efekt płynnego zanikania czarnego ekranu po uruchomieniu poziomu.

Diagram dziedziczenia dla Level1FadeIn



Atrybuty publiczne

- CanvasGroup fadeOverlay

  Referencja do `CanvasGroup`, który kontroluje przezroczystość czarnego overlaya.
- float fadeDuration = 1f

  Czas trwania efektu zanikania (w sekundach).

6.33.1 Opis szczegółowy

Realizuje efekt płynnego zanikania czarnego ekranu po uruchomieniu poziomu.

Skrypt przy starcie uruchamia animację `fade-in` przy użyciu `CanvasGroup`, stopniowo ujawniając widok gry. Po zakończeniu efektu wyłącza obiekt z komponentem `CanvasGroup`.

Przeznaczony do użytku jako efekt przejścia przy rozpoczęciu sceny (np. poziomu 1).

Autor

Julia Bigaj

6.33.2 Dokumentacja atrybutów składowych

6.33.2.1 fadeDuration

`float Level1FadeIn.fadeDuration = 1f`

Czas trwania efektu zanikania (w sekundach).

6.33.2.2 fadeOverlay

```
CanvasGroup Level1FadeIn.fadeOverlay
```

Referencja do `CanvasGroup`, który kontroluje przezroczystość czarnego overlaya.

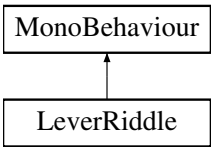
Dokumentacja dla tej klasy została wygenerowana z pliku:

- Assets/Scripts/Menu HUDs/Level1FadeIn.cs

6.34 Dokumentacja klasy LeverRiddle

Sprawdza poprawność ułożenia dźwigni i aktywuje ukrytą platformę.

Diagram dziedziczenia dla LeverRiddle



Metody publiczne

- void CheckCorrectness ()
  Sprawdza aktualny stan dźwigni i aktywuje ukryte kafelki, jeśli warunek jest spełniony.

6.34.1 Opis szczegółowy

Sprawdza poprawność ułożenia dźwigni i aktywuje ukrytą platformę.

Gdy dźwignie zostaną ustawione w odpowiedniej kombinacji, aktywuje ukryte kafelki (np. kamień, most).

Autor

Filip Kudła

6.34.2 Dokumentacja funkcji składowych

6.34.2.1 CheckCorrectness()

```
void LeverRiddle.CheckCorrectness ()  [inline]
```

Sprawdza aktualny stan dźwigni i aktywuje ukryte kafelki, jeśli warunek jest spełniony.

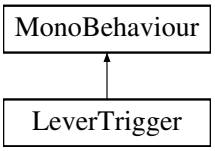
Dokumentacja dla tej klasy została wygenerowana z pliku:

- Assets/Scripts/Interactive/LeverRiddle.cs

6.35 Dokumentacja klasy LeverTrigger

Obsługuje interakcję gracza z dźwignią i zmienia jej stan.

Diagram dziedziczenia dla LeverTrigger



Atrybuty publiczne

- Animator leverAnimator

  Animator przypisany do dźwigni (obsługuje stan On/Off).
- bool leverIsOn

  Aktualny stan dźwigni (czy włączona).

6.35.1 Opis szczegółowy

Obsługuje interakcję gracza z dźwignią i zmienia jej stan.

Gracz może aktywować lub dezaktywować dźwignię po naciśnięciu klawisza F, jeśli znajduje się w zasięgu. Dźwignia uruchamia animację i dźwięk oraz przekazuje informację do skryptu zagadki.

Autor

Julia Bigaj

6.35.2 Dokumentacja atrybutów składowych

6.35.2.1 leverAnimator

```
Animator LeverTrigger.leverAnimator
```

Animator przypisany do dźwigni (obsługuje stan On/Off).

6.35.2.2 leverIsOn

```
bool LeverTrigger.leverIsOn
```

Aktualny stan dźwigni (czy włączona).

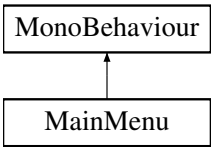
Dokumentacja dla tej klasy została wygenerowana z pliku:

- Assets/Scripts/Interactive/LeverTrigger.cs

6.36 Dokumentacja klasy MainMenu

Obsługuje przyciski menu głównego: rozpoczęcie gry, wczytanie stanu i wyjście.

Diagram dziedziczenia dla MainMenu



Metody publiczne

- void StartNewGame ()

  Rozpoczyna nową grę, ładując scenę wprowadzającą.
- void LoadGame ()

  Wczytuje grę i ustawia callback na zakończenie ładowania sceny.
- void QuitGame ()

  Zamyka aplikację.

6.36.1 Opis szczegółowy

Obsługuje przyciski menu głównego: rozpoczęcie gry, wczytanie stanu i wyjście.

Klasa odpowiada za przechodzenie do odpowiednich scen oraz inicjalizację systemu zapisu podczas wczytywania gry. Umożliwia również zakończenie działania aplikacji.

Autor

Julia Bigaj

6.36.2 Dokumentacja funkcji składowych

6.36.2.1 LoadGame()

```
void MainMenu.LoadGame ()  [inline]
```

Wczytuje grę i ustawia callback na zakończenie ładowania sceny.

Po załadowaniu sceny "`Level 1 - Cave`" następuje automatyczne wywołanie systemu zapisu.

6.36.2.2 QuitGame()

```
void MainMenu.QuitGame ()  [inline]
```

Zamyka aplikację.

Wywołuje `Application.Quit()` i wypisuje debug log.

6.36.2.3 StartNewGame()

```
void MainMenu.StartNewGame ()  [inline]
```

Rozpoczyna nową grę, ładując scenę wprowadzającą.

Zamiast poziomu głównego ładowana jest scena `"Intro"`.

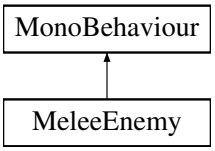
Dokumentacja dla tej klasy została wygenerowana z pliku:

- Assets/Scripts/Menu HUDs/MainMenu.cs

6.37 Dokumentacja klasy MeleeEnemy

Skrypt przeciwnika atakującego wręcz, wykrywającego gracza za pomocą BoxCast.

Diagram dziedziczenia dla MeleeEnemy



6.37.1 Opis szczegółowy

Skrypt przeciwnika atakującego wręcz, wykrywającego gracza za pomocą BoxCast.

Przeciwnik patroluje do momentu wykrycia gracza, a następnie wykonuje animację ataku z określonym cooldownem.

Autor

Filip Kudła

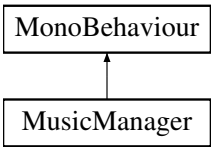
Dokumentacja dla tej klasy została wygenerowana z pliku:

- Assets/Scripts/Enemy/MeleeEnemy.cs

6.38 Dokumentacja klasy MusicManager

Globalny menedżer muzyki i narracji działający między scenami.

Diagram dziedziczenia dla MusicManager



Metody publiczne

- void PlayMenuMusic ()

  Odtwarza muzykę menu, jeśli jeszcze nie jest aktywna.
- void PlayGameplayMusic (float volume=1f)

  Odtwarza muzykę tła do rozgrywki z domyślną lub określoną głośnością.
- void PlayIntroAudio ()

  Odtwarza narrację i muzykę w scenie Intro.
- void StopMusic ()

  Zatrzymuje odtwarzanie aktualnej muzyki i narracji.

Atrybuty publiczne

- AudioSource audioSource

  Źródło audio odpowiedzialne za tło muzyczne.
- AudioSource narrationSource

  Źródło audio odpowiedzialne za narrację (np. intro).
- AudioClip menuMusic

  Muzyka odtwarzana w menu głównym.
- AudioClip gameplayMusic

  Muzyka odtwarzana podczas gry.
- AudioClip introNarration

  Narracja głosowa odtwarzana w scenie Intro.
- AudioClip introMusic

  Muzyka odtwarzana w scenie Intro.

Właściwości

- static MusicManager Instance `[get]`

  Instancja singletonu.

6.38.1 Opis szczegółowy

Globalny menedżer muzyki i narracji działający między scenami.

Zarządza odtwarzaniem odpowiednich ścieżek dźwiękowych i narracji w zależności od aktualnie załadowanej sceny. Implementuje wzorzec singletonu, aby zapewnić dostępność między scenami (`DontDestroyOnLoad`).

Obsługuje oddzielne źródła dźwięku: jedno dla muzyki (`audioSource`), drugie dla narracji (`narration↩Source`).

Autor

Julia Bigaj

6.38.2 Dokumentacja funkcji składowych

6.38.2.1 PlayGameplayMusic()

```
void MusicManager.PlayGameplayMusic (
            float volume = 1f)  [inline]
```

Odtwarza muzykę tła do rozgrywki z domyślną lub określoną głośnością.

Parametry

	volume	Poziom głośności muzyki (domyślnie 1f).

6.38.2.2 PlayIntroAudio()

```
void MusicManager.PlayIntroAudio ()  [inline]
```

Odtwarza narrację i muzykę w scenie Intro.

6.38.2.3 PlayMenuMusic()

```
void MusicManager.PlayMenuMusic ()  [inline]
```

Odtwarza muzykę menu, jeśli jeszcze nie jest aktywna.

6.38.2.4 StopMusic()

```
void MusicManager.StopMusic ()  [inline]
```

Zatrzymuje odtwarzanie aktualnej muzyki i narracji.

6.38.3 Dokumentacja atrybutów składowych

6.38.3.1 audioSource

```
AudioSource MusicManager.audioSource
```

Źródło audio odpowiedzialne za tło muzyczne.

6.38.3.2 gameplayMusic

```
AudioClip MusicManager.gameplayMusic
```

Muzyka odtwarzana podczas gry.

6.38.3.3 introMusic

```
AudioClip MusicManager.introMusic
```

Muzyka odtwarzana w scenie Intro.

6.38.3.4 introNarration

```
AudioClip MusicManager.introNarration
```

Narracja głosowa odtwarzana w scenie Intro.

6.38.3.5 menuMusic

```
AudioClip MusicManager.menuMusic
```

Muzyka odtwarzana w menu głównym.

6.38.3.6 narrationSource

```
AudioSource MusicManager.narrationSource
```

Źródło audio odpowiedzialne za narrację (np. intro).

6.38.4 Dokumentacja właściwości

6.38.4.1 Instance

```
MusicManager MusicManager.Instance  [static], [get]
```

Instancja singletonu.

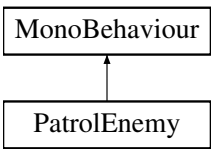
Dokumentacja dla tej klasy została wygenerowana z pliku:

- Assets/Scripts/Music&Sound/MusicManager.cs

6.39 Dokumentacja klasy PatrolEnemy

Skrypt odpowiedzialny za patrolowanie przeciwnika między dwoma punktami.

Diagram dziedziczenia dla PatrolEnemy



6.39.1 Opis szczegółowy

Skrypt odpowiedzialny za patrolowanie przeciwnika między dwoma punktami.

Przeciwnik przemieszcza się w lewo i prawo między dwoma granicami. Zatrzymuje się na chwilę na krańcach i zmienia kierunek.

Autor

Filip Kudła

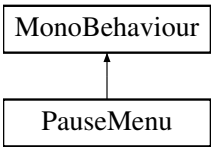
Dokumentacja dla tej klasy została wygenerowana z pliku:

- Assets/Scripts/Enemy/PatrolEnemy.cs

6.40 Dokumentacja klasy PauseMenu

Zarządza stanem pauzy i końca gry w trakcie rozgrywki.

Diagram dziedziczenia dla PauseMenu



Metody publiczne

- void Resume ()

  Wznawia grę po pauzie.
- void LoadGame ()

  Wczytuje bieżącą scenę i odtwarza zapisany stan.
- void Pause ()

  Wstrzymuje grę i aktywuje menu pauzy.
- void QuitGame ()

  Kończy grę i wraca do menu głównego.
- void SaveGame ()

  Zapisuje bieżący stan gry.
- IEnumerator ShowGameOver ()

  Coroutine wyświetlająca ekran końca gry po krótkim opóźnieniu.

Atrybuty publiczne

- GameObject pauseMenuUI

  Panel UI menu pauzy.
- GameObject gameOverUI

  Panel UI ekranu końca gry.
- AudioSource gameOverAudio

  Źródło dźwięku odtwarzanego przy przegranej.

Statyczne atrybuty publiczne

- static bool isGameOver = false

  Czy gra zakończyła się (np. przegrana).
- static bool isPaused = false

  Czy gra jest aktualnie zapauzowana.

6.40.1 Opis szczegółowy

Zarządza stanem pauzy i końca gry w trakcie rozgrywki.

Obsługuje wstrzymywanie i wznawianie gry, zapisywanie/wczytywanie stanu, powrót do menu głównego oraz ekran końca gry. Blokuje czas gry i interakcje UI w odpowiednich momentach. Działa tylko w scenie "`Level 1 - Cave`".

Autor

Julia Bigaj

6.40.2 Dokumentacja funkcji składowych

6.40.2.1 LoadGame()

```
void PauseMenu.LoadGame ()  [inline]
```

Wczytuje bieżącą scenę i odtwarza zapisany stan.

6.40.2.2 Pause()

```
void PauseMenu.Pause ()  [inline]
```

Wstrzymuje grę i aktywuje menu pauzy.

6.40.2.3 QuitGame()

```
void PauseMenu.QuitGame ()  [inline]
```

Kończy grę i wraca do menu głównego.

6.40.2.4 Resume()

```
void PauseMenu.Resume ()  [inline]
```

Wznawia grę po pauzie.

6.40.2.5 SaveGame()

```
void PauseMenu.SaveGame ()  [inline]
```

Zapisuje bieżący stan gry.

6.40.2.6 ShowGameOver()

```
IEnumerator PauseMenu.ShowGameOver ()  [inline]
```

Coroutine wyświetlająca ekran końca gry po krótkim opóźnieniu.

Zwraca

Enumerator dla StartCoroutine.

6.40.3 Dokumentacja atrybutów składowych

6.40.3.1 gameOverAudio

```
AudioSource PauseMenu.gameOverAudio
```

Źródło dźwięku odtwarzanego przy przegranej.

6.40.3.2 gameOverUI

```
GameObject PauseMenu.gameOverUI
```

Panel UI ekranu końca gry.

6.40.3.3 isGameOver

```
bool PauseMenu.isGameOver = false  [static]
```

Czy gra zakończyła się (np. przegrana).

6.40.3.4 isPaused

```
bool PauseMenu.isPaused = false  [static]
```

Czy gra jest aktualnie zapauzowana.

6.40.3.5 pauseMenuUI

`GameObject PauseMenu.pauseMenuUI`

Panel UI menu pauzy.

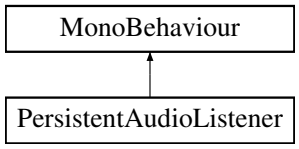
Dokumentacja dla tej klasy została wygenerowana z pliku:

- Assets/Scripts/Menu HUDs/PauseMenu.cs

6.41 Dokumentacja klasy PersistentAudioListener

Zapewnia, że w scenie znajduje się tylko jeden aktywny `AudioListener`.

Diagram dziedziczenia dla PersistentAudioListener



6.41.1 Opis szczegółowy

Zapewnia, że w scenie znajduje się tylko jeden aktywny `AudioListener`.

Skrypt usuwa się automatycznie, jeśli w scenie istnieje już inny `AudioListener`, zapobiegając konfliktom audio w Unity (komunikat: "There are 2 AudioListeners"). Ustawiony jako trwały między scenami dzięki `DontDestroy↩OnLoad`.

Autor

Julia Bigaj

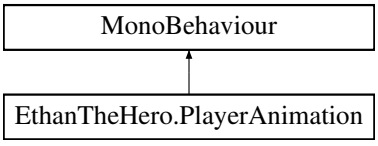
Dokumentacja dla tej klasy została wygenerowana z pliku:

- Assets/Scripts/Music&Sound/PersistentAudioListener.cs

6.42 Dokumentacja klasy EthanTheHero.PlayerAnimation

Steruje animacjami gracza na podstawie jego stanu ruchu i fizyki.

Diagram dziedziczenia dla EthanTheHero.PlayerAnimation



6.42.1 Opis szczegółowy

Steruje animacjami gracza na podstawie jego stanu ruchu i fizyki.

Komunikuje się z komponentem Animator, by dynamicznie ustawiać animacje biegu, skoku, dasza, zsuwania się po ścianie i innych zachowań. Pobiera dane z klas `PlayerMovement`, `Rigidbody2D` i `PlayerAttackMethod`.

Obsługuje również logikę sprawdzania zakończenia animacji przejścia oraz może być rozszerzony o animacje obrażeń i śmierci.

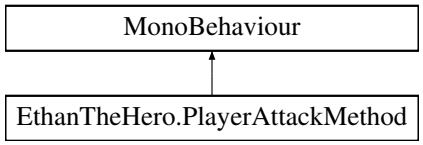
Dokumentacja dla tej klasy została wygenerowana z pliku:

- Assets/Scripts/Player/PlayerAnimation.cs

6.43 Dokumentacja klasy EthanTheHero.PlayerAttackMethod

Obsługuje podstawowy system ataku postaci gracza (combo 3-atakowe).

Diagram dziedziczenia dla EthanTheHero.PlayerAttackMethod



Atrybuty publiczne

- float basicAttack01Power = 0.5f

  Siła przesunięcia gracza podczas 1 ataku.
- float basicAttack02Power = 0.5f

  Siła przesunięcia gracza podczas 2 ataku.
- float basicAttack03Power = 0.9f

  Siła przesunięcia gracza podczas 3 ataku.

Statyczne atrybuty publiczne

- static bool isPaused = false

  Czy gra jest zapauzowana (globalnie).

6.43.1 Opis szczegółowy

Obsługuje podstawowy system ataku postaci gracza (combo 3-atakowe).

Umożliwia wykonywanie sekwencyjnych ataków (combo), reagując na kliknięcia myszy w odpowiednim czasie trwania animacji. Obsługuje również dźwięki i przesunięcia gracza podczas ataku.

Używa animatora oraz komponentów PlayerMovement i Rigidbody2D.

6.43.2 Dokumentacja atrybutów składowych

6.43.2.1 basicAttack01Power

```
float EthanTheHero.PlayerAttackMethod.basicAttack01Power = 0.5f
```

Siła przesunięcia gracza podczas 1 ataku.

6.43.2.2 basicAttack02Power

```
float EthanTheHero.PlayerAttackMethod.basicAttack02Power = 0.5f
```

Siła przesunięcia gracza podczas 2 ataku.

6.43.2.3 basicAttack03Power

```
float EthanTheHero.PlayerAttackMethod.basicAttack03Power = 0.9f
```

Siła przesunięcia gracza podczas 3 ataku.

6.43.2.4 isPaused

```
bool EthanTheHero.PlayerAttackMethod.isPaused = false  [static]
```

Czy gra jest zapauzowana (globalnie).

Dokumentacja dla tej klasy została wygenerowana z pliku:

- Assets/Scripts/Player/PlayerAttackMethod.cs

6.44 Dokumentacja struktury PlayerData

Struktura danych zawierająca informacje do zapisu o graczu.

Atrybuty publiczne

- float hp

  Aktualny poziom zdrowia gracza.

6.44.1 Opis szczegółowy

Struktura danych zawierająca informacje do zapisu o graczu.

Aktualnie przechowuje jedynie ilość punktów życia gracza (`hp`), ale można ją łatwo rozszerzyć o inne dane (np. poziom, pozycję, ekwipunek).

Autor

Filip Kudła

6.44.2 Dokumentacja atrybutów składowych

6.44.2.1 hp

```
float PlayerData.hp
```

Aktualny poziom zdrowia gracza.

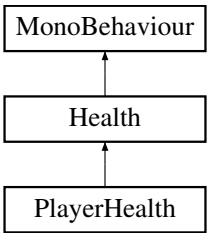
Dokumentacja dla tej struktury została wygenerowana z pliku:

- Assets/Scripts/Save/PlayerSaveData.cs

6.45 Dokumentacja klasy PlayerHealth

Klasa zarządzająca zdrowiem i wytrzymałością gracza.

Diagram dziedziczenia dla PlayerHealth



Metody publiczne

- override void Die ()
- void TakeStamina (float amount)
- override void TakeDamage (float amount)
- void HealStamina (float amount)
- override void SetBarsValue (float value)

Metody publiczne dziedziczone z Health

- virtual void Heal (float amount)

Metody chronione

- override void Awake ()

Metody chronione dziedziczone z Health

- virtual IEnumerator DamageCooldownCoroutine ()

Właściwości

- float currentStamina [get]

  Aktualna ilość staminy (dostępna tylko do odczytu).

Dodatkowe dziedziczone składowe

Atrybuty chronione dziedziczone z Health

- float startingHealth

  Początkowa wartość zdrowia.
- MicroBar healthBar

  Pasek zdrowia.
- float damageCooldown = 0.75f

  Czas nieśmiertelności po otrzymaniu obrażeń.
- float currentHealth
- Animator anim
- SpriteRenderer spriteRend
- bool canTakeDamage = true

6.45.1 Opis szczegółowy

Klasa zarządzająca zdrowiem i wytrzymałością gracza.

Dziedziczy po klasie `Health` i rozszerza ją o obsługę paska staminy, jej zużywania i regeneracji. Integruje się z paskami zdrowia i staminy z Microlight.MicroBar, a także zatrzymuje ruch gracza po śmierci.

Autor

Filip Kudła

6.45.2 Dokumentacja funkcji składowych

6.45.2.1 Awake()

```
override void PlayerHealth.Awake ()  [inline], [protected], [virtual]
```

Inicjalizuje zdrowie i staminy oraz komponent ruchu.

Reimplementowana z Health.

6.45.2.2 Die()

```
override void PlayerHealth.Die ()  [inline], [virtual]
```

Przeciąża metodę śmierci: animacja i ekran końcowy.

Reimplementowana z Health.

6.45.2.3 HealStamina()

```
void PlayerHealth.HealStamina (
            float amount)  [inline]
```

Lepsza wersja odzyskiwania staminy.

6.45.2.4 SetBarsValue()

```
override void PlayerHealth.SetBarsValue (
            float value)  [inline], [virtual]
```

Ustawia wartość pasków zdrowia i staminy.

Reimplementowana z Health.

6.45.2.5 TakeDamage()

```
override void PlayerHealth.TakeDamage (
            float amount)  [inline], [virtual]
```

Odbieranie HP wraz z zaimplementowanym dźwiękiem.

Reimplementowana z Health.

6.45.2.6 TakeStamina()

```
void PlayerHealth.TakeStamina (
            float amount)  [inline]
```

Odbiera określoną ilość staminy, aktualizując pasek.

6.45.3 Dokumentacja właściwości

6.45.3.1 currentStamina

```
float PlayerHealth.currentStamina  [get]
```

Aktualna ilość staminy (dostępna tylko do odczytu).

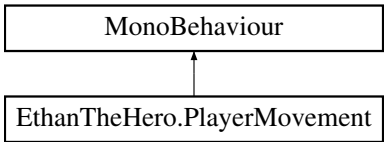
Dokumentacja dla tej klasy została wygenerowana z pliku:

- Assets/Scripts/Health/PlayerHealth.cs

6.46 Dokumentacja klasy EthanTheHero.PlayerMovement

Odpowiada za ruch postaci gracza (bieganie, skok, dash, wall slide, wall jump).

Diagram dziedziczenia dla EthanTheHero.PlayerMovement



Atrybuty publiczne

- Vector2 move

  Kierunek ruchu gracza.
- bool isDashing
- bool grounded
- bool isJumping
- bool wallSlidingEnabled = true
- bool wallJump
- bool wallSliding

6.46.1 Opis szczegółowy

Odpowiada za ruch postaci gracza (bieganie, skok, dash, wall slide, wall jump).

Bazuje na danych z `PlayerMovementData`. Obsługuje również dźwięki (kroki, dash, skok), stan UI oraz interakcję z kolizjami gruntu i ścian.

6.46.2 Dokumentacja atrybutów składowych

6.46.2.1 grounded

```
bool EthanTheHero.PlayerMovement.grounded
```

6.46.2.2 isDashing

```
bool EthanTheHero.PlayerMovement.isDashing
```

6.46.2.3 isJumping

```
bool EthanTheHero.PlayerMovement.isJumping
```

6.46.2.4 move

```
Vector2 EthanTheHero.PlayerMovement.move
```

Kierunek ruchu gracza.

6.46.2.5 wallJump

```
bool EthanTheHero.PlayerMovement.wallJump
```

6.46.2.6 wallSliding

```
bool EthanTheHero.PlayerMovement.wallSliding
```

6.46.2.7 wallSlidingEnabled

```
bool EthanTheHero.PlayerMovement.wallSlidingEnabled = true
```

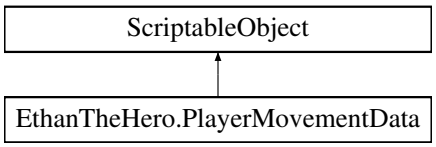
Dokumentacja dla tej klasy została wygenerowana z pliku:

- Assets/Scripts/Player/PlayerMovement/PlayerMovement.cs

6.47 Dokumentacja klasy EthanTheHero.PlayerMovementData

ScriptableObject przechowujący dane konfiguracyjne ruchu gracza.

Diagram dziedziczenia dla EthanTheHero.PlayerMovementData



Atrybuty publiczne

- float runMaxSpeed

  Docelowa maksymalna prędkość biegu gracza.
- float runAcceleration

  Czas, po którym gracz powinien osiągnąć prędkość maksymalną (z pozycji spoczynkowej).
- float runAccelAmount

  Obliczona siła przyspieszenia (na podstawie `runAcceleration`).
- float runDecceleration

  Czas potrzebny do wytracenia prędkości (zatrzymania).
- float runDeccelAmount

  Obliczona siła wytracenia prędkości (na podstawie `runDecceleration`).
- float accelInAir

  Mnożnik przyspieszenia w powietrzu.
- float deccelInAir

  Mnożnik wytracenia prędkości w powietrzu.
- bool doConserveMomentum

  Czy gracz ma zachowywać pęd przy nagłych zmianach kierunku.
- float jumpHeight

  Wysokość skoku gracza.
- float dashPower = 30f

  Siła dasza (poziomy impuls).
- float dashingCoolDown = 1f

  Czas oczekiwania na ponowne użycie dasza.
- float dashingTime = 0.2f

  Czas trwania dasza.
- float dashCost = 20f

  Koszt dasza w punktach staminy.
- float wallDistance = 0.05f

  Odległość od ściany, przy której wykrywana jest możliwość zsuwania się.
- float wallJumpTime = 0.2f

  Czas okna wejścia w wall jump.
- float wallSlideSpeed = 0.3f

  Maksymalna prędkość zsuwania się po ścianie.
- float wallJumpingYPower = 6.5f

  Siła skoku w pionie przy wall jumpie.
- float wallJumpingXPower = 5f

  Siła skoku w poziomie przy wall jumpie.
- float WallJumpTimeInSecond = 0.1f

  Czas trwania animacji/efektu wall jumpa.

6.47.1 Opis szczegółowy

ScriptableObject przechowujący dane konfiguracyjne ruchu gracza.

Zawiera ustawienia prędkości biegu, przyspieszenia, skoku, daszu oraz interakcji ze ścianami. Obliczenia siły przyspieszenia i wytracania prędkości wykonywane są automatycznie w metodzie `OnValidate()`. Umożliwia łatwe dostosowanie balansu postaci bez modyfikacji kodu.

6.47.2 Dokumentacja atrybutów składowych

6.47.2.1 accelInAir

```
float EthanTheHero.PlayerMovementData.accelInAir
```

Mnożnik przyspieszenia w powietrzu.

6.47.2.2 dashCost

```
float EthanTheHero.PlayerMovementData.dashCost = 20f
```

Koszt dasza w punktach staminy.

6.47.2.3 dashingCoolDown

```
float EthanTheHero.PlayerMovementData.dashingCoolDown = 1f
```

Czas oczekiwania na ponowne użycie dasza.

6.47.2.4 dashingTime

```
float EthanTheHero.PlayerMovementData.dashingTime = 0.2f
```

Czas trwania dasza.

6.47.2.5 dashPower

```
float EthanTheHero.PlayerMovementData.dashPower = 30f
```

Siła dasza (poziomy impuls).

6.47.2.6 deccelInAir

```
float EthanTheHero.PlayerMovementData.deccelInAir
```

Mnożnik wytracenia prędkości w powietrzu.

6.47.2.7 doConserveMomentum

```
bool EthanTheHero.PlayerMovementData.doConserveMomentum
```

Czy gracz ma zachowywać pęd przy nagłych zmianach kierunku.

6.47.2.8 jumpHeight

```
float EthanTheHero.PlayerMovementData.jumpHeight
```

Wysokość skoku gracza.

6.47.2.9 runAccelAmount

```
float EthanTheHero.PlayerMovementData.runAccelAmount
```

Obliczona siła przyspieszenia (na podstawie `runAcceleration`).

6.47.2.10 runAcceleration

```
float EthanTheHero.PlayerMovementData.runAcceleration
```

Czas, po którym gracz powinien osiągnąć prędkość maksymalną (z pozycji spoczynkowej).

6.47.2.11 runDeccelAmount

```
float EthanTheHero.PlayerMovementData.runDeccelAmount
```

Obliczona siła wytracenia prędkości (na podstawie `runDecceleration`).

6.47.2.12 runDecceleration

```
float EthanTheHero.PlayerMovementData.runDecceleration
```

Czas potrzebny do wytracenia prędkości (zatrzymania).

6.47.2.13 runMaxSpeed

```
float EthanTheHero.PlayerMovementData.runMaxSpeed
```

Docelowa maksymalna prędkość biegu gracza.

6.47.2.14 wallDistance

```
float EthanTheHero.PlayerMovementData.wallDistance = 0.05f
```

Odległość od ściany, przy której wykrywana jest możliwość zsuwania się.

6.47.2.15 wallJumpingXPower

```
float EthanTheHero.PlayerMovementData.wallJumpingXPower = 5f
```

Siła skoku w poziomie przy wall jumpie.

6.47.2.16 wallJumpingYPower

```
float EthanTheHero.PlayerMovementData.wallJumpingYPower = 6.5f
```

Siła skoku w pionie przy wall jumpie.

6.47.2.17 wallJumpTime

```
float EthanTheHero.PlayerMovementData.wallJumpTime = 0.2f
```

Czas okna wejścia w wall jump.

6.47.2.18 WallJumpTimeInSecond

```
float EthanTheHero.PlayerMovementData.WallJumpTimeInSecond = 0.1f
```

Czas trwania animacji/efektu wall jumpa.

6.47.2.19 wallSlideSpeed

```
float EthanTheHero.PlayerMovementData.wallSlideSpeed = 0.3f
```

Maksymalna prędkość zsuwania się po ścianie.

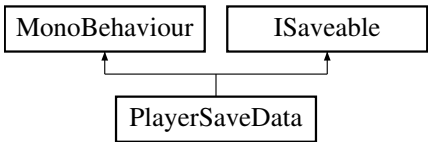
Dokumentacja dla tej klasy została wygenerowana z pliku:

- Assets/Scripts/Player/PlayerMovement/PlayerMovementData.cs

6.48 Dokumentacja klasy PlayerSaveData

Klasa odpowiedzialna za zapis i odczyt stanu gracza.

Diagram dziedziczenia dla PlayerSaveData



Metody publiczne

- object CaptureState ()

  Tworzy obiekt stanu gracza do zapisania.
- void RestoreState (object state)

  Przywraca stan gracza na podstawie zapisanych danych.

6.48.1 Opis szczegółowy

Klasa odpowiedzialna za zapis i odczyt stanu gracza.

Implementuje interfejs `ISaveable`, integrując się z globalnym systemem zapisu gry. Pobiera i przywraca stan komponentu `Health` przypisanego do gracza.

Nota

Wymaga obecności komponentu `Health` w tym samym obiekcie.

Autor

Filip Kudła

6.48.2 Dokumentacja funkcji składowych

6.48.2.1 CaptureState()

```
object PlayerSaveData.CaptureState ()  [inline]
```

Tworzy obiekt stanu gracza do zapisania.

Zwraca

Obiekt `PlayerData` reprezentujący aktualne dane gracza.

Implementuje ISaveable.

6.48.2.2 RestoreState()

```
void PlayerSaveData.RestoreState (
            object state)  [inline]
```

Przywraca stan gracza na podstawie zapisanych danych.

Parametry

	state	Obiekt w formacie JSON zawierający dane gracza.

Implementuje ISaveable.

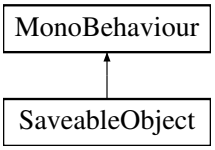
Dokumentacja dla tej klasy została wygenerowana z pliku:

- Assets/Scripts/Save/PlayerSaveData.cs

6.49 Dokumentacja klasy SaveableObject

Przypisuje unikalny identyfikator (`UniqueId`) do obiektu gry, aby umożliwić jego zapis i odtworzenie.

Diagram dziedziczenia dla SaveableObject



Właściwości

- string UniqueId `[get]`

  Publiczny dostęp do unikalnego ID.

6.49.1 Opis szczegółowy

Przypisuje unikalny identyfikator (`UniqueId`) do obiektu gry, aby umożliwić jego zapis i odtworzenie.

Klasa ta zapewnia, że każdy obiekt w scenie posiada unikalny identyfikator, który może być wykorzystany w systemie zapisu stanu gry. Jeśli identyfikator nie jest unikalny, zostaje wygenerowany nowy. Działa tylko w edytorze Unity — podczas działania gry identyfikator nie jest zmieniany.

Nota

Współpracuje z interfejsem `ISaveable`.

`DisallowMultipleComponent` gwarantuje, że komponent nie zostanie dodany wielokrotnie do jednego obiektu.

Autor

Filip Kudła

6.49.2 Dokumentacja właściwości

6.49.2.1 UniqueId

```
string SaveableObject.UniqueId  [get]
```

Publiczny dostęp do unikalnego ID.

Zwraca

Niezmienny identyfikator GUID.

Dokumentacja dla tej klasy została wygenerowana z pliku:

- Assets/Scripts/Save/SaveableObject.cs

6.50 Dokumentacja klasy SaveSystem.SaveData

(Nieużywana) Lista wpisów `SaveEntry` — używana w alternatywnym podejściu.

Atrybuty publiczne

- List< SaveEntry > entries = new List<SaveEntry>()

6.50.1 Opis szczegółowy

(Nieużywana) Lista wpisów `SaveEntry` — używana w alternatywnym podejściu.

6.50.2 Dokumentacja atrybutów składowych

6.50.2.1 entries

```
List<SaveEntry> SaveSystem.SaveData.entries = new List<SaveEntry>()
```

Dokumentacja dla tej klasy została wygenerowana z pliku:

- Assets/Scripts/Save/SaveSystem.cs

6.51 Dokumentacja klasy SaveSystem.SaveEntry

(Nieużywana) Struktura zapisu jednego obiektu z danymi jako JSON.

Atrybuty publiczne

- string id
- string jsonData
- string type

6.51.1 Opis szczegółowy

(Nieużywana) Struktura zapisu jednego obiektu z danymi jako JSON.

6.51.2 Dokumentacja atrybutów składowych

6.51.2.1 id

```
string SaveSystem.SaveEntry.id
```

6.51.2.2 jsonData

```
string SaveSystem.SaveEntry.jsonData
```

6.51.2.3 type

```
string SaveSystem.SaveEntry.type
```

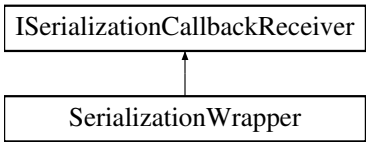
Dokumentacja dla tej klasy została wygenerowana z pliku:

- Assets/Scripts/Save/SaveSystem.cs

6.52 Dokumentacja klasy SerializationWrapper

Pomocnicza klasa do serializacji słownika `Dictionary<string, object>` w Unity.

Diagram dziedziczenia dla SerializationWrapper



Metody publiczne

- void OnBeforeSerialize ()

  Metoda wywoływana automatycznie przed serializacją.
- void OnAfterDeserialize ()

  Metoda wywoływana automatycznie po deserializacji.

Atrybuty publiczne

- List< string > keys = new List<string>()

  Klucze (typów komponentów ISaveable) jako listy stringów.
- List< string > jsonValues = new List<string>()

  Dane w postaci stringów (JSON) powiązane z każdym kluczem.
- Dictionary< string, object > data = new Dictionary<string, object>()

  Faktyczne dane zmapowane przez typy komponentów jako string (key) i obiekt (value).

6.52.1 Opis szczegółowy

Pomocnicza klasa do serializacji słownika `Dictionary<string, object>` w Unity.

Unity nie obsługuje bezpośrednio serializacji słowników i typów ogólnych (generics), dlatego `SerializationWrapper` konwertuje dane do dwóch list: `keys` i `jsonValues`.

Umożliwia bezpieczne zapisanie i odczytanie złożonych danych w systemie zapisu gry.

Zawiera implementację interfejsu `ISerializationCallbackReceiver` do obsługi procesów serializacji i deserializacji w edytorze i czasie działania.

Zobacz również

SaveSystem

Autor

Filip Kudła

6.52.2 Dokumentacja funkcji składowych

6.52.2.1 OnAfterDeserialize()

```
void SerializationWrapper.OnAfterDeserialize () [inline]
```

Metoda wywoływana automatycznie po deserializacji.

Odtwarza strukturę słownika z danych zawartych w `keys` i `jsonValues`. Wartości pozostają jako string (JSON) — konwersja do konkretnego typu musi być wykonana później ręcznie.

6.52.2.2 OnBeforeSerialize()

```
void SerializationWrapper.OnBeforeSerialize () [inline]
```

Metoda wywoływana automatycznie przed serializacją.

Przekształca dane z `Dictionary<string, object>` do dwóch list: `keys` i `jsonValues`, gdzie każdy obiekt serializowany jest jako JSON.

6.52.3 Dokumentacja atrybutów składowych

6.52.3.1 data

```
Dictionary<string, object> SerializationWrapper.data = new Dictionary<string, object>()
```

Faktyczne dane zmapowane przez typy komponentów jako string (key) i obiekt (value).

To pole jest oznaczone jako [NonSerialized] i uzupełniane podczas `OnAfterDeserialize`.

6.52.3.2 jsonValues

```
List<string> SerializationWrapper.jsonValues = new List<string>()
```

Dane w postaci stringów (JSON) powiązane z każdym kluczem.

6.52.3.3 keys

```
List<string> SerializationWrapper.keys = new List<string>()
```

Klucze (typów komponentów ISaveable) jako listy stringów.

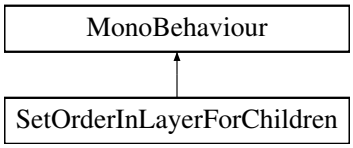
Dokumentacja dla tej klasy została wygenerowana z pliku:

- Assets/Scripts/Save/SerializationWrapper.cs

6.53 Dokumentacja klasy SetOrderInLayerForChildren

Ustawia warstwę rysowania (`sortingOrder`) dla wszystkich dzieci posiadających komponent SpriteRenderer.

Diagram dziedziczenia dla SetOrderInLayerForChildren



Atrybuty publiczne

- int orderInLayer = 2

  Wartość warstwy renderowania przypisywana wszystkim dzieciom.

6.53.1 Opis szczegółowy

Ustawia warstwę rysowania (`sortingOrder`) dla wszystkich dzieci posiadających komponent SpriteRenderer.

Skrypt przeznaczony do kontrolowania kolejności renderowania sprite'ów w hierarchii obiektów. Może być używany np. na obiekcie nadrzędnym zawierającym elementy dekoracyjne.

Autor

Julia Bigaj

6.53.2 Dokumentacja atrybutów składowych

6.53.2.1 orderInLayer

```
int SetOrderInLayerForChildren.orderInLayer = 2
```

Wartość warstwy renderowania przypisywana wszystkim dzieciom.

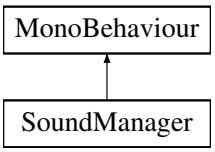
Dokumentacja dla tej klasy została wygenerowana z pliku:

- Assets/Scripts/Map and locations/SetOrderInLayerForChildren.cs

6.54 Dokumentacja klasy SoundManager

Centralny menedżer efektów dźwiękowych w grze.

Diagram dziedziczenia dla SoundManager



Metody publiczne

- void PlaySound (AudioClip clip)

  Odtwarza pojedynczy dźwięk z podanego AudioClip.
- void StartSteps ()

  Rozpoczyna zapętlone odtwarzanie dźwięku kroków.
- void StopSteps ()

  Zatrzymuje zapętlony dźwięk kroków.
- void PlayStep ()

  Odtwarza dźwięk kroku.
- void PlayChest ()

  Odtwarza dźwięk otwierania skrzyni.
- void PlayJump ()

  Odtwarza dźwięk skoku.
- void PlayDash ()

  Odtwarza dźwięk dashowania.
- void PlayHurt ()

  Odtwarza dźwięk obrażeń.
- void PlayDoor ()

  Odtwarza dźwięk otwierania drzwi.
- void PlayBarrel ()

  Odtwarza dźwięk zniszczenia beczki.
- void PlayLightAttack ()

  Odtwarza dźwięk lekkiego ataku.
- void PlayHeavyAttack ()

  Odtwarza dźwięk ciężkiego ataku.
- void PlayLever ()

  Odtwarza dźwięk użycia dźwigni.
- void PlayStone ()

  Odtwarza dźwięk przesuwania kamienia.

Atrybuty publiczne

- AudioClip doorOpenSound
  Dźwięk otwierania drzwi.
- AudioClip chestOpenSound
  Dźwięk otwierania skrzyni.
- AudioClip destroyBarrelSound
  Dźwięk zniszczenia beczki.
- AudioClip lightAttackSound
  Dźwięk lekkiego ataku.
- AudioClip heavyAttackSound
  Dźwięk ciężkiego ataku.
- AudioClip jumpSound
  Dźwięk skoku.
- AudioClip dashSound
  Dźwięk dashowania.
- AudioClip hurtSound
  Dźwięk otrzymania obrażeń.
- AudioClip stepSound
  Dźwięk kroków bohatera.
- AudioClip leverPullSound
  Dźwięk pociągnięcia za dźwignię.
- AudioClip moveStoneSound
  Dźwięk przesuwania kamienia.

Statyczne atrybuty publiczne

- static SoundManager Instance
  Instancja singletonu SoundManager.

6.54.1 Opis szczegółowy

Centralny menedżer efektów dźwiękowych w grze.

Implementuje wzorzec singletonu. Udostępnia metody do odtwarzania konkretnych efektów dźwiękowych, takich jak ataki, otwieranie skrzyni, dźwięki łamigłówek czy kroki bohatera. Kroki obsługiwane są osobnym źródłem dźwięku (`stepSource`) jako dźwięk zapętlony.

Autor

Julia Bigaj

6.54.2 Dokumentacja funkcji składowych

6.54.2.1 PlayBarrel()

```
void SoundManager.PlayBarrel ()
```

Odtwarza dźwięk zniszczenia beczki.

6.54.2.2 PlayChest()

```
void SoundManager.PlayChest ()
```

Odtwarza dźwięk otwierania skrzyni.

6.54.2.3 PlayDash()

```
void SoundManager.PlayDash ()
```

Odtwarza dźwięk dashowania.

6.54.2.4 PlayDoor()

```
void SoundManager.PlayDoor ()
```

Odtwarza dźwięk otwierania drzwi.

6.54.2.5 PlayHeavyAttack()

```
void SoundManager.PlayHeavyAttack ()
```

Odtwarza dźwięk ciężkiego ataku.

6.54.2.6 PlayHurt()

```
void SoundManager.PlayHurt ()
```

Odtwarza dźwięk obrażeń.

6.54.2.7 PlayJump()

```
void SoundManager.PlayJump ()
```

Odtwarza dźwięk skoku.

6.54.2.8 PlayLever()

```
void SoundManager.PlayLever ()
```

Odtwarza dźwięk użycia dźwigni.

6.54.2.9 PlayLightAttack()

```
void SoundManager.PlayLightAttack ()
```

Odtwarza dźwięk lekkiego ataku.

6.54.2.10 PlaySound()

```
void SoundManager.PlaySound (
            AudioClip clip)  [inline]
```

Odtwarza pojedynczy dźwięk z podanego AudioClip.

Parametry

	clip	Dźwięk do odtworzenia.

6.54.2.11 PlayStep()

```
void SoundManager.PlayStep ()
```

Odtwarza dźwięk kroku.

6.54.2.12 PlayStone()

```
void SoundManager.PlayStone ()
```

Odtwarza dźwięk przesuwania kamienia.

6.54.2.13 StartSteps()

```
void SoundManager.StartSteps ()  [inline]
```

Rozpoczyna zapętlone odtwarzanie dźwięku kroków.

6.54.2.14 StopSteps()

```
void SoundManager.StopSteps ()  [inline]
```

Zatrzymuje zapętlony dźwięk kroków.

6.54.3 Dokumentacja atrybutów składowych

6.54.3.1 chestOpenSound

```
AudioClip SoundManager.chestOpenSound
```

Dźwięk otwierania skrzyni.

6.54.3.2 dashSound

```
AudioClip SoundManager.dashSound
```

Dźwięk dashowania.

6.54.3.3 destroyBarrelSound

```
AudioClip SoundManager.destroyBarrelSound
```

Dźwięk zniszczenia beczki.

6.54.3.4 doorOpenSound

```
AudioClip SoundManager.doorOpenSound
```

Dźwięk otwierania drzwi.

6.54.3.5 heavyAttackSound

```
AudioClip SoundManager.heavyAttackSound
```

Dźwięk ciężkiego ataku.

6.54.3.6 hurtSound

```
AudioClip SoundManager.hurtSound
```

Dźwięk otrzymania obrażeń.

6.54.3.7 Instance

```
SoundManager SoundManager.Instance  [static]
```

Instancja singletonu SoundManager.

6.54.3.8 jumpSound

```
AudioClip SoundManager.jumpSound
```

Dźwięk skoku.

6.54.3.9 leverPullSound

```
AudioClip SoundManager.leverPullSound
```

Dźwięk pociągnięcia za dźwignię.

6.54.3.10 lightAttackSound

```
AudioClip SoundManager.lightAttackSound
```

Dźwięk lekkiego ataku.

6.54.3.11 moveStoneSound

`AudioClip SoundManager.moveStoneSound`

Dźwięk przesuwania kamienia.

6.54.3.12 stepSound

`AudioClip SoundManager.stepSound`

Dźwięk kroków bohatera.

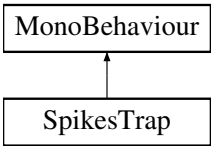
Dokumentacja dla tej klasy została wygenerowana z pliku:

- Assets/Scripts/Music&Sound/SoundManager.cs

6.55 Dokumentacja klasy SpikesTrap

Pułapka kolców zadająca obrażenia i odpychająca gracza po wejściu w trigger.

Diagram dziedziczenia dla SpikesTrap



6.55.1 Opis szczegółowy

Pułapka kolców zadająca obrażenia i odpychająca gracza po wejściu w trigger.

Po kolizji z graczem:

- zadaje określoną liczbę obrażeń,
- anuluje jego bieżący ruch,
- odpycha go w kierunku od centrum kolców.

Wymaga komponentu `Rigidbody2D` na graczu i skryptu `Health`.

Autor

Filip Kudła

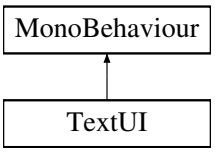
Dokumentacja dla tej klasy została wygenerowana z pliku:

- Assets/Scripts/Traps/SpikesTrap.cs

6.56 Dokumentacja klasy TextUI

Wyświetla tymczasowe wiadomości dialogowe na ekranie w komponentach TextMeshProUGUI.

Diagram dziedziczenia dla TextUI



Metody publiczne

- void ShowTextDialog (string text, float duration=3f, int boxIndex=0)

  Wyświetla wiadomość tekstową w określonym oknie dialogowym na określony czas.

Atrybuty publiczne

- TextMeshProUGUI[ ] dialogBoxes

  Tablica okien dialogowych (TextMeshProUGUI), w których może pojawić się tekst.

6.56.1 Opis szczegółowy

Wyświetla tymczasowe wiadomości dialogowe na ekranie w komponentach TextMeshProUGUI.

Umożliwia prezentację tekstu w jednym z wielu okien dialogowych (np. powiadomień, opisów, wskazówek). Tekst znika automatycznie po określonym czasie.

Autor

Filip Kudła

6.56.2 Dokumentacja funkcji składowych

6.56.2.1 ShowTextDialog()

```
void TextUI.ShowTextDialog (
            string text,
            float duration = 3f,
            int boxIndex = 0)  [inline]
```

Wyświetla wiadomość tekstową w określonym oknie dialogowym na określony czas.

Parametry

	text	Tekst do wyświetlenia.
	duration	Czas trwania wyświetlania tekstu (domyślnie 3 sekundy).
	boxIndex	Indeks okna dialogowego z tablicy `dialogBoxes`.

6.56.3 Dokumentacja atrybutów składowych

6.56.3.1 dialogBoxes

```
TextMeshProUGUI [] TextUI.dialogBoxes
```

Tablica okien dialogowych (TextMeshProUGUI), w których może pojawić się tekst.

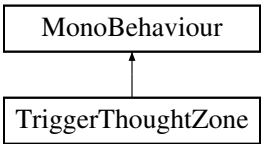
Dokumentacja dla tej klasy została wygenerowana z pliku:

- Assets/Scripts/Menu HUDs/TextUI.cs

6.57 Dokumentacja klasy TriggerThoughtZone

Wyświetla myśl bohatera po wejściu do specjalnej strefy w grze.

Diagram dziedziczenia dla TriggerThoughtZone



Atrybuty publiczne

- string thought = "Nie ma innego wyjścia... \r\nChyba muszę po prostu tam skoczyć i zobaczyć co się stanie"
  Treść myśli wyświetlanej po wejściu do triggera.

6.57.1 Opis szczegółowy

Wyświetla myśl bohatera po wejściu do specjalnej strefy w grze.

Skrypt przypisany do triggera środowiskowego – po wejściu gracza:

- wyłącza ślizganie po ścianach (jeśli dotyczy),
- uruchamia wiadomość dialogową przez `TextUI`, wyświetlaną nad głową bohatera.

Myśl znika automatycznie po kilku sekundach.

Autor

Julia Bigaj

6.57.2 Dokumentacja atrybutów składowych

6.57.2.1 thought

```
string TriggerThoughtZone.thought = "Nie ma innego wyjścia... \r\nChyba muszę po prostu tam
skoczyć i zobaczyć co się stanie"
```

Treść myśli wyświetlanej po wejściu do triggera.

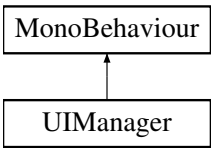
Dokumentacja dla tej klasy została wygenerowana z pliku:

- Assets/Scripts/Menu HUDs/TriggerThoughtZone.cs

6.58 Dokumentacja klasy UIManager

Singleton odpowiedzialny za zarządzanie elementami interfejsu użytkownika w całej grze.

Diagram dziedziczenia dla UIManager



Właściwości

- static UIManager Instance `[get]`
  Statyczna instancja singletonu UIManager.

6.58.1 Opis szczegółowy

Singleton odpowiedzialny za zarządzanie elementami interfejsu użytkownika w całej grze.

Obiekt jest przenoszony między scenami za pomocą `DontDestroyOnLoad`, zapewniając trwałość UI i centralny dostęp do zarządzania elementami interfejsu w wielu scenach.

Autor

Julia Bigaj

6.58.2 Dokumentacja właściwości

6.58.2.1 Instance

```
UIManager UIManager.Instance  [static], [get]
```

Statyczna instancja singletonu UIManager.

Dokumentacja dla tej klasy została wygenerowana z pliku:

- Assets/Scripts/Menu HUDs/UIManager.cs

Rozdział 7

Dokumentacja plików

7.1 Dokumentacja pliku Assets/Scripts/Attack/AttackController.cs

Komponenty

- class AttackController

  Klasa odpowiedzialna za zarządzanie atakiem postaci (tworzeniem hitboxów, cooldownem).

7.2 Dokumentacja pliku Assets/Scripts/Attack/AttackData.cs

Komponenty

- class AttackData

  ScriptableObject przechowujące dane dotyczące ataku.

7.3 Dokumentacja pliku Assets/Scripts/Attack/AttackHitbox.cs

Komponenty

- class AttackHitbox

  Odpowiada za detekcję kolizji i zadawanie obrażeń.

7.4 Dokumentacja pliku Assets/Scripts/Core/CameraController.cs

Komponenty

- class CameraController

  Klasa odpowiadająca za płynne podążanie kamery za graczem.

7.5 Dokumentacja pliku Assets/Scripts/Core/EventSystemController.cs

Komponenty

- class EventSystemController

  Kontroluje istnienie tylko jednego aktywnego EventSystemu po zmianie sceny.

7.6 Dokumentacja pliku Assets/Scripts/Core/GhostCameraFollow.cs

Komponenty

- class GhostCameraFollow

  Prosty skrypt do podążania obiektu (np. kamery) za wskazanym celem.

7.7 Dokumentacja pliku Assets/Scripts/Doors/DoorTrigger.cs

Komponenty

- class DoorTrigger

  Skrypt obsługujący teleportację gracza po naciśnięciu klawisza, gdy znajduje się przy drzwiach.

7.8 Dokumentacja pliku Assets/Scripts/Enemy/MeleeEnemy.cs

Komponenty

- class MeleeEnemy

  Skrypt przeciwnika atakującego wręcz, wykrywającego gracza za pomocą BoxCast.

7.9 Dokumentacja pliku Assets/Scripts/Enemy/PatrolEnemy.cs

Komponenty

- class PatrolEnemy

  Skrypt odpowiedzialny za patrolowanie przeciwnika między dwoma punktami.

7.10 Dokumentacja pliku Assets/Scripts/Ghost/EncounteredGhostDialog.cs

Komponenty

- class EncounteredGhostDialog

  Obsługuje dialog pomiędzy graczem a napotkanym duszkiem.

7.11 Dokumentacja pliku Assets/Scripts/Ghost/GhostFloating.cs

Komponenty

- class GhostFloating

  Odpowiada za animację 'unoszenia się' duszka w górę i w dół.

7.12 Dokumentacja pliku Assets/Scripts/Ghost/GhostFollow.cs

Komponenty

- class GhostFollow

  Skrypt do płynnego śledzenia gracza przez duszka.

7.13 Dokumentacja pliku Assets/Scripts/Health/Health.cs

Komponenty

- class Health

  Bazowa klasa obsługująca zdrowie dla wszelkich istot w grze.

7.14 Dokumentacja pliku Assets/Scripts/Health/HealthCollectible.cs

Komponenty

- class HealthCollectible

  Skrypt odpowiedzialny za przedmiot przywracający zdrowie graczowi.

7.15 Dokumentacja pliku Assets/Scripts/Health/LavaDamage.cs

Komponenty

- class LavaDamage

  Skrypt zadający obrażenia graczowi przy kontakcie z lawą.

7.16 Dokumentacja pliku Assets/Scripts/Health/PlayerHealth.cs

Komponenty

- class PlayerHealth

  Klasa zarządzająca zdrowiem i wytrzymałością gracza.

7.17 Dokumentacja pliku Assets/Scripts/Interactive/Barrel.cs

Komponenty

- class Barrel

  Obsługuje niszczenie beczki po uderzeniu atakiem gracza.

7.18 Dokumentacja pliku Assets/Scripts/Interactive/LeverRiddle.cs

Komponenty

- class LeverRiddle

  Sprawdza poprawność ułożenia dźwigni i aktywuje ukrytą platformę.

7.19 Dokumentacja pliku Assets/Scripts/Interactive/LeverTrigger.cs

Komponenty

- class LeverTrigger

  Obsługuje interakcję gracza z dźwignią i zmienia jej stan.

7.20 Dokumentacja pliku Assets/Scripts/Inventory/ChestController.cs

Komponenty

- class ChestController

  Otwiera skrzynię po interakcji gracza i dodaje przedmiot (list) do ekwipunku.

7.21 Dokumentacja pliku Assets/Scripts/Inventory/InventoryManager.cs

Komponenty

- class InventoryManager

  Zarządza systemem ekwipunku gracza.

7.22 Dokumentacja pliku Assets/Scripts/Inventory/InventorySlotSpawner.cs

Komponenty

- class InventorySlotSpawner

  Generuje dynamiczne sloty listów w UI ekwipunku.

7.23 Dokumentacja pliku Assets/Scripts/Inventory/InventoryUI.cs

Komponenty

- class InventoryUI

  Zarządza interfejsem ekwipunku oraz wyświetlaniem treści listów.

7.24 Dokumentacja pliku Assets/Scripts/Inventory/LetterData.cs

Komponenty

- class LetterData

  Dane pojedynczego listu kolekcjonerskiego.

7.25 Dokumentacja pliku Assets/Scripts/Map and locations/LeafRevealer.cs

Komponenty

- class LeafRevealer

  Odsłania ukrytą lokalizację, gdy gracz wejdzie w obszar kolizji z liśćmi.

7.26 Dokumentacja pliku Assets/Scripts/Map and locations/SetOrderInLayerForChildren.cs

Komponenty

- class SetOrderInLayerForChildren

  Ustawia warstwę rysowania (`sortingOrder`) dla wszystkich dzieci posiadających komponent SpriteRenderer.

7.27 Dokumentacja pliku Assets/Scripts/Menu HUDs/Bootstrapper.cs

Komponenty

- class Bootstrapper

  Ładuje sceny startowe gry przy uruchomieniu aplikacji.

7.28 Dokumentacja pliku Assets/Scripts/Menu HUDs/ButtonHoverUnderline.cs

Komponenty

- class ButtonHoverUnderline

  Dodaje i usuwa podkreślenie tekstu przycisku podczas najechania kursorem myszy.

7.29 Dokumentacja pliku Assets/Scripts/Menu HUDs/ChestPanelManager.cs

Komponenty

- class ChestPanelManager

  Globalny menedżer odpowiedzialny za stan panelu skrzyni.

7.30 Dokumentacja pliku Assets/Scripts/Menu HUDs/EndGameTrigger.cs

Komponenty

- class EndGameTrigger

  Obsługuje zakończenie gry po dotarciu gracza do punktu końcowego.

7.31 Dokumentacja pliku Assets/Scripts/Menu HUDs/HintArea.cs

Komponenty

- class HintArea

  Wyświetla wiadomość podpowiedzi, gdy gracz znajdzie się w określonym obszarze.

7.32 Dokumentacja pliku Assets/Scripts/Menu HUDs/HintController.cs

Komponenty

- class HintController

  Zarządza wyświetlaniem tekstowych podpowiedzi w interfejsie użytkownika.

7.33 Dokumentacja pliku Assets/Scripts/Menu HUDs/HUDVisibility.cs

Komponenty

- class HUDVisibility

  Ukrywa interfejs HUD w scenie menu głównego.

7.34 Dokumentacja pliku Assets/Scripts/Menu HUDs/Level1FadeIn.cs

Komponenty

- class Level1FadeIn

  Realizuje efekt płynnego zanikania czarnego ekranu po uruchomieniu poziomu.

7.35 Dokumentacja pliku Assets/Scripts/Menu HUDs/MainMenu.cs

Komponenty

- class MainMenu

  Obsługuje przyciski menu głównego: rozpoczęcie gry, wczytanie stanu i wyjście.

7.36 Dokumentacja pliku Assets/Scripts/Menu HUDs/PauseMenu.cs

Komponenty

- class PauseMenu

  Zarządza stanem pauzy i końca gry w trakcie rozgrywki.

7.37 Dokumentacja pliku Assets/Scripts/Menu HUDs/TextUI.cs

Komponenty

- class TextUI

  Wyświetla tymczasowe wiadomości dialogowe na ekranie w komponentach TextMeshProUGUI.

7.38 Dokumentacja pliku Assets/Scripts/Menu HUDs/TriggerThoughtZone.cs

Komponenty

- class TriggerThoughtZone

  Wyświetla myśl bohatera po wejściu do specjalnej strefy w grze.

7.39 Dokumentacja pliku Assets/Scripts/Menu HUDs/UIManager.cs

Komponenty

- class UIManager

  Singleton odpowiedzialny za zarządzanie elementami interfejsu użytkownika w całej grze.

7.40 Dokumentacja pliku Assets/Scripts/Menu HUDs/UIStateManager.cs

Komponenty

- class **UIStateManager**

  Przechowuje globalny stan interfejsu użytkownika (UI).

7.41 Dokumentacja pliku Assets/Scripts/Music&Sound/ButtonHoverSound.cs

Komponenty

- class ButtonHoverSound

  Odtwarza dźwięk najechania na przycisk w interfejsie użytkownika.

7.42 Dokumentacja pliku Assets/Scripts/Music&Sound/IntroController.cs

Komponenty

- class IntroController

  Odpowiada za odtwarzanie sekwencji wprowadzenia do gry (intro).

7.43 Dokumentacja pliku Assets/Scripts/Music&Sound/MusicManager.cs

Komponenty

- class MusicManager

  Globalny menedżer muzyki i narracji działający między scenami.

7.44 Dokumentacja pliku Assets/Scripts/Music&Sound/PersistentAudioListener.cs

Komponenty

- class PersistentAudioListener

  Zapewnia, że w scenie znajduje się tylko jeden aktywny `AudioListener`.

7.45 Dokumentacja pliku Assets/Scripts/Music&Sound/SoundManager.cs

Komponenty

- class SoundManager

  Centralny menedżer efektów dźwiękowych w grze.

7.46 Dokumentacja pliku Assets/Scripts/Player/PlayerAnimation.cs

Komponenty

- class EthanTheHero.PlayerAnimation

  Steruje animacjami gracza na podstawie jego stanu ruchu i fizyki.

Przestrzenie nazw

- namespace EthanTheHero

7.47 Dokumentacja pliku Assets/Scripts/Player/PlayerAttackMethod.cs

Komponenty

- class EthanTheHero.PlayerAttackMethod

  Obsługuje podstawowy system ataku postaci gracza (combo 3-atakowe).

Przestrzenie nazw

- namespace EthanTheHero

7.48 Dokumentacja pliku Assets/Scripts/Player/PlayerMovement/PlayerMovement.cs

Komponenty

- class EthanTheHero.PlayerMovement

  Odpowiada za ruch postaci gracza (bieganie, skok, dash, wall slide, wall jump).

Przestrzenie nazw

- namespace EthanTheHero

7.49 Dokumentacja pliku Assets/Scripts/Player/PlayerMovement/PlayerMovementData.cs

Komponenty

- class EthanTheHero.PlayerMovementData

  ScriptableObject przechowujący dane konfiguracyjne ruchu gracza.

Przestrzenie nazw

- namespace EthanTheHero

7.50 Dokumentacja pliku Assets/Scripts/Save/BarrelSavaData.cs

Komponenty

- class BarrelSaveData

  Odpowiada za trwałe przechowywanie informacji o zniszczonych beczkach między sesjami gry.

7.51 Dokumentacja pliku Assets/Scripts/Save/ISaveable.cs

Komponenty

- interface ISaveable

  Interfejs do obsługi zapisu i odczytu stanu obiektów w grze.

7.52 Dokumentacja pliku Assets/Scripts/Save/PlayerSaveData.cs

Komponenty

- struct PlayerData

  Struktura danych zawierająca informacje do zapisu o graczu.
- class PlayerSaveData

  Klasa odpowiedzialna za zapis i odczyt stanu gracza.

7.53 Dokumentacja pliku Assets/Scripts/Save/SaveableObject.cs

Komponenty

- class SaveableObject

  Przypisuje unikalny identyfikator (`UniqueId`) do obiektu gry, aby umożliwić jego zapis i odtworzenie.

7.54 Dokumentacja pliku Assets/Scripts/Save/SaveSystem.cs

Komponenty

- class **SaveSystem**

  Statyczna klasa odpowiedzialna za zapis i odczyt stanu sceny w grze.
- class SaveSystem.SaveEntry

  (Nieużywana) Struktura zapisu jednego obiektu z danymi jako JSON.
- class SaveSystem.SaveData

  (Nieużywana) Lista wpisów `SaveEntry` — używana w alternatywnym podejściu.

7.55 Dokumentacja pliku Assets/Scripts/Save/SerializationWrapper.cs

Komponenty

- class SerializationWrapper

  Pomocnicza klasa do serializacji słownika `Dictionary<string, object>` w Unity.

7.56 Dokumentacja pliku Assets/Scripts/Traps/AxeTrap.cs

Komponenty

- class AxeTrap

  Obsługuje ruchome pułapki typu wahadłowy topór.

7.57 Dokumentacja pliku Assets/Scripts/Traps/SpikesTrap.cs

Komponenty

- class SpikesTrap

  Pułapka kolców zadająca obrażenia i odpychająca gracza po wejściu w trigger.