Pixelborne

Generated by Doxygen 1.8.17

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 2

Class Index

2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

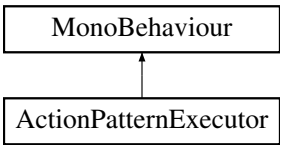
Chapter 3

Class Documentation

3.1 ActionPatternExecutor Class Reference

Executes automatically actions on objects that have a proper implementation of the IEnemyActions interface. The entity that is executed by this class should have an attack and sight range.

Inheritance diagram for ActionPatternExecutor:



Private Types

- enum **EntityMode** { **IN_SIGHT_RANGE** = 0, **OUT_OF_SIGHT_RANGE** = 1, **IN_ATTACK_RANGE** = 2 }

Private Member Functions

- void **Start** ()
- Tuple< int, float >[ ][ ] **parseAttackPatterns** ()
- void **ResetActionPattern** ()
- void **Update** ()
- void **RandomlySelectNextAttackPatternInRange** ()
- Tuple< int, float >[ ] **ParseAttackPattern** (string attackPatternString)
- void **PrepareAttackPatternParsingDict** ()

Private Attributes

- string[ ] **m_attackPatternStringsWhileInAttackRange**
- string **m_attackPatternStringWhileInSightRange**
- string **m_attackPatternStringWhileOutOfSight**
- IEnemyActions **m_entityAttackAndMovement**
- List< Action > **m_actions**
- Random **random** = new Random()
- int **m_nextAttackPatternIndex**
- float **m_timeToWaitUntilNextAction**
- bool **m_isWaitingOnBeingGrounded**
- Tuple< int, float >[ ][ ] **m_attackPatterns**
- int **m_currentAttackPatternListIndex**
- EntityMode **m_currentEntityMode** = EntityMode.OUT_OF_SIGHT_RANGE
- Dictionary< string, Tuple< int, float > > **m_attackPatternStringToInternalIdentifications**

Static Private Attributes

- static readonly string **ATTACK_UP_IDENTIFICATION** = "AU"
- static readonly string **ATTACK_MID_IDENTIFICATION** = "AM"
- static readonly string **ATTACK_DOWN_IDENTIFICATION** = "AD"
- static readonly string **JUMP_IDENTIFICATION** = "JUMP"
- static readonly string **START_FOLLOW_PLAYER_IDENTIFICATION** = "STARTF"
- static readonly string **STOP_FOLLOW_PLAYER_IDENTIFICATION** = "STOPF"
- static readonly string **START_AUTO_JUMPING_IDENTIFICATION** = "STARTAUTOJUMP"
- static readonly string **STOP_AUTO_JUMPING_IDENTIFICATION** = "STOPAUTOJUMP"
- static readonly string **SEPERATION_IDENTIFICATION** = "|"

3.1.1 Detailed Description

Executes automatically actions on objects that have a proper implementation of the IEnemyActions interface. The entity that is executed by this class should have an attack and sight range.

The actions are divided into 3 pattern. The first pattern is the m_attackPatternStringWhileOutOfSight. It is executed if not IsPlayerInSightRange(). The second pattern is the m_attackPatternStringWhileInSightRange. It is executed if IsPlayerInoAttackRange() and not IsPlayerInSAttackRange. The last pattern is the m_attackPatternStringWhile↩ InAttackRange. It is executed if IsPlayerInAttackRange(). This pattern is actually a list of individual patterns. After one individual pattern has finished the next one is chosen randomly.

Each pattern is provided as a string with the grammar below. It basically contains a series of actions that are looped infinitely. The identifications of these actions can be found below. After each action a waiting time can be specified. If no waiting time is specified, the duration of that action is taken as the waiting time.

When the attack pattern changes the currently executed action is finished and then the new attack pattern starts from the beginning.

The attack pattern need to be set in the unity editor.

```
ATTACK PATTERN GRAMMAR: ATTACK_PATTERN = ATTACK_TOKEN ATTACK_PATTERN_1 or
epsilon ATTACK_PATTERN_1 = |ATTACK_TOKEN or epsilon ATTACK_TOKEN = ATTACK_↩
INSTRUCTION or ATTACK_INSTRUCTION|TIMEOUT TIMEOUT = float ATTACK_INSTRUCTION
= one of the constant strings below
Example assignment of the attack pattern in the unity editor.
m_attackPatternStringWhileOutOfSight = "STOPF";
m_attackPatternStringWhileInSightRange = "STARTF";
m_attackPatternStringWhileInAttackRange = ["AU|AM|AD|2", "AD|JUMP|0.5|AD|3|AU"];
```

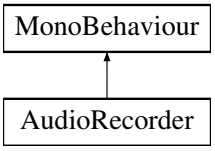
The documentation for this class was generated from the following file:

- C:/X_PIXELS/pixelborne/Assets/Scripts/Enemies/ActionPatternExecutor.cs

3.2 AudioRecorder Class Reference

Is responsible for recording and saving the 10 seconds long audio clips.

Inheritance diagram for AudioRecorder:



Public Member Functions

- bool MicrophoneAvailable ()

  Returns if a microphone is available.
- void Record ()

  Initiates the recording of a 10 seconds long audio clip if no recording is already running.

Private Member Functions

- void **Start** ()
- void **FixedUpdate** ()
- void **SaveRecording** ()

Private Attributes

- AudioClip **m_microphoneClip**
- bool **m_isRecording** = false
- Stopwatch **m_stopwatchForRecording** = new Stopwatch()
- string **m_filedir**
- string **m_selectedDevice**

Static Private Attributes

- static readonly int **RECORD_DURATION** = 10000
- static readonly string **AUDIO_RECORD_DIR** = "records"

3.2.1 Detailed Description

Is responsible for recording and saving the 10 seconds long audio clips.

3.2.2 Member Function Documentation

3.2.2.1 MicrophoneAvailable()

```
bool AudioRecorder.MicrophoneAvailable ( )
```

Returns if a microphone is available.

Returns

Is a microphone available.

3.2.2.2 Record()

```
void AudioRecorder.Record ( )
```

Initiates the recording of a 10 seconds long audio clip if no recording is already running.

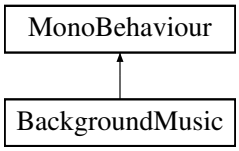
The documentation for this class was generated from the following file:

- C:/X_PIXELS/pixelborne/Assets/Scripts/Recording/AudioRecorder.cs

3.3 BackgroundMusic Class Reference

Handles the volume of the default background music.

Inheritance diagram for BackgroundMusic:



Static Public Member Functions

- static void SetVolume (float value)
  Sets the volume.

Private Member Functions

- void **Start** ()

Static Private Attributes

- static AudioSource **s_player**

3.3.1 Detailed Description

Handles the volume of the default background music.

3.3.2 Member Function Documentation

3.3.2.1 SetVolume()

```
static void BackgroundMusic.SetVolume (
            float value )  [static]
```

Sets the volume.

Parameters

value	The volume.

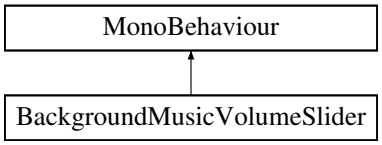
The documentation for this class was generated from the following file:

- C:/X_PIXELS/pixelborne/Assets/Scripts/Audio/BackgroundMusic.cs

3.4 BackgroundMusicVolumeSlider Class Reference

Sets the value of the volume slider in the settings. Gets this value from the SettingsContainer.

Inheritance diagram for BackgroundMusicVolumeSlider:



Private Member Functions

- void **Start** ()

3.4.1 Detailed Description

Sets the value of the volume slider in the settings. Gets this value from the SettingsContainer.

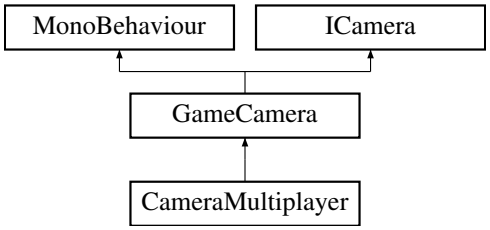
The documentation for this class was generated from the following file:

- C:/X_PIXELS/pixelborne/Assets/Scripts/Menu/BackgroundMusicVolumeSlider.cs

3.5 CameraMultiplayer Class Reference

It controls the camera movement and fades to black of the multiplayer scene camera.

Inheritance diagram for CameraMultiplayer:



Public Member Functions

- void SetPosition (int index)

  Moves the center of both the camera and the fade to black canvas object to the given position while retaining the z-position.

Protected Member Functions

- override void FadedOut ()

  Is called when the camera faded out and invokes the FadedOut-Methdod on the current Multiplayer instance.
- override void FadedIn ()

  Is called when the camera faded in and invokes the FadedIn-Methdod on the current Multiplayer instance.

Properties

- IList< Vector2 > Positions `[get, set]`

  Gets or sets the camera spawn positions from outer left to outer right stage as they are in the scene.

Private Member Functions

- void **Awake** ()
- void **Start** ()

Private Attributes

- Transform **m_cameraPositionsTransform**

Additional Inherited Members

3.5.1 Detailed Description

It controls the camera movement and fades to black of the multiplayer scene camera.

3.5.2 Member Function Documentation

3.5.2.1 FadedIn()

```
override void CameraMultiplayer.FadedIn ( )  [protected], [virtual]
```

Is called when the camera faded in and invokes the FadedIn-Methdod on the current Multiplayer instance.

Implements GameCamera.

3.5.2.2 FadedOut()

```
override void CameraMultiplayer.FadedOut ( )  [protected], [virtual]
```

Is called when the camera faded out and invokes the FadedOut-Methdod on the current Multiplayer instance.

Implements GameCamera.

3.5.2.3 SetPosition()

```
void CameraMultiplayer.SetPosition (
            int index )
```

Moves the center of both the camera and the fade to black canvas object to the given position while retaining the z-position.

Parameters

index	The index in the camera spawn positions list.

3.5.3 Property Documentation

3.5.3.1 Positions

```
IList<Vector2> CameraMultiplayer.Positions  [get], [set]
```

Gets or sets the camera spawn positions from outer left to outer right stage as they are in the scene.

The positions.

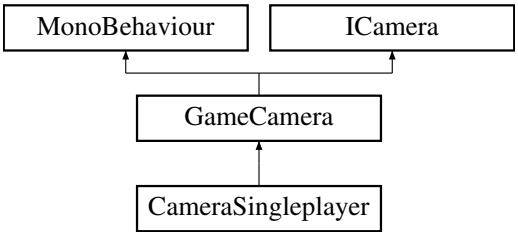
The documentation for this class was generated from the following file:

- C:/X_PIXELS/pixelborne/Assets/Scripts/Camera/CameraMultiplayer.cs

3.6 CameraSingleplayer Class Reference

Controls the camera of the singleplayer scene.

Inheritance diagram for CameraSingleplayer:



Protected Member Functions

- override void Update ()

  Updates this instance by setting the position to the player position.
- override void FadedOut ()

  Is called when the camera faded out and invokes the FadedOut-Methdod on the current Singleplayer instance.
- override void FadedIn ()

  Is called when the camera faded in and invokes the FadedIn-Methdod on the current Singleplayer instance.

Private Member Functions

- void **Start** ()

Additional Inherited Members

3.6.1 Detailed Description

Controls the camera of the singleplayer scene.

3.6.2 Member Function Documentation

3.6.2.1 FadedIn()

```
override void CameraSingleplayer.FadedIn ( )  [protected], [virtual]
```

Is called when the camera faded in and invokes the FadedIn-Methdod on the current Singleplayer instance.

Implements GameCamera.

3.6.2.2 FadedOut()

```
override void CameraSingleplayer.FadedOut ( )  [protected], [virtual]
```

Is called when the camera faded out and invokes the FadedOut-Methdod on the current Singleplayer instance.

Implements GameCamera.

3.6.2.3 Update()

```
override void CameraSingleplayer.Update ( )  [protected], [virtual]
```

Updates this instance by setting the position to the player position.

Reimplemented from GameCamera.

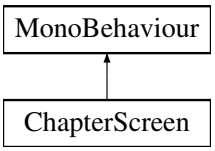
The documentation for this class was generated from the following file:

- C:/X_PIXELS/pixelborne/Assets/Scripts/Camera/CameraSingleplayer.cs

3.7 ChapterScreen Class Reference

Handles the displaying of a chapter screen.

Inheritance diagram for ChapterScreen:



Private Member Functions

- void **Start** ()
- void **Update** ()

Private Attributes

- float **m_displayTime** = 1500
- Stopwatch **m_stopwatch** = new Stopwatch()

3.7.1 Detailed Description

Handles the displaying of a chapter screen.

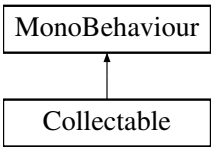
The documentation for this class was generated from the following file:

- C:/X_PIXELS/pixelborne/Assets/Scripts/Story/ChapterScreen.cs

3.8 Collectable Class Reference

Handles the collection of a collectable GameObject.

Inheritance diagram for Collectable:



Private Member Functions

- void **Start** ()
- void **Update** ()
- void **OnTriggerEnter2D** (Collider2D collider)

Private Attributes

- AudioSource **m_audioPlayer**
- bool **m_isCollected** = false

3.8.1 Detailed Description

Handles the collection of a collectable GameObject.

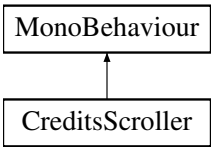
The documentation for this class was generated from the following file:

- C:/X_PIXELS/pixelborne/Assets/Scripts/Trigger/Collectable.cs

3.9 CreditsScroller Class Reference

Handles the behaviour of the credits in the menu.

Inheritance diagram for CreditsScroller:



Private Member Functions

- void **Awake** ()
- void **OnEnable** ()
- void **Update** ()
- void **OnDisable** ()

Private Attributes

- GameObject **m_credits**
- GameObject **m_mainMenu**
- Vector3 **m_originalPos**

Static Private Attributes

- const float **m_SCROLL_SPEED** = 0.05f
- const int **m_CREDITS_SCREEN_BORDER_Y** = 15

3.9.1 Detailed Description

Handles the behaviour of the credits in the menu.

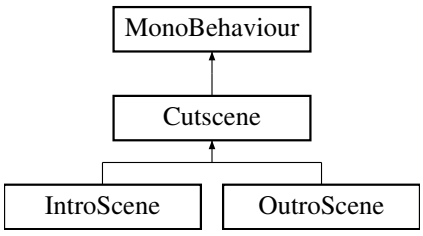
The documentation for this class was generated from the following file:

- C:/X_PIXELS/pixelborne/Assets/Scripts/Menu/CreditsScroller.cs

3.10 Cutscene Class Reference

Implements basic functionality of a cutscene.

Inheritance diagram for Cutscene:



Protected Types

- enum **CutSceneMode** {
  FadeImage, **DisplayText**, **AnimateImages**, **Done**,
  Nothing }

Protected Member Functions

- virtual void **Start** ()
- virtual void **Update** ()
- abstract CutSceneMode **ChangeStoryPart** ()

Protected Attributes

- float **m_fadeTime** = 3000
- Image **m_backgroundImage**
- TextMeshProUGUI **m_story**
- CutSceneMode **m_mode**
- int **m_storyPart** = 0
- int **m_textPart** = 0
- Stopwatch **m_stopwatch** = new Stopwatch()

Properties

- virtual string[ ][ ] **StoryHolder** `[get, set]`

3.10.1 Detailed Description

Implements basic functionality of a cutscene.

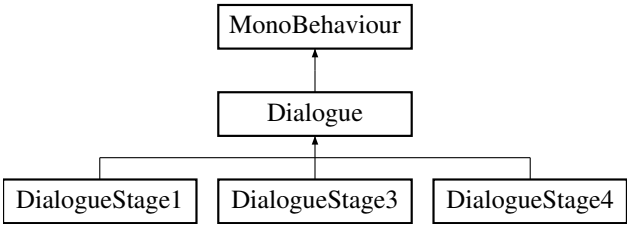
The documentation for this class was generated from the following file:

- C:/X_PIXELS/pixelborne/Assets/Scripts/Story/CutScene.cs

3.11 Dialogue Class Reference

Implements basic functionality of a dialogue.

Inheritance diagram for Dialogue:



Protected Member Functions

- virtual void **Start** ()
- virtual void **SetDialogueVisibility** (bool isVisible)
- void **InsertName** ()

Protected Attributes

- int **m_displayTime** = 3000
- Image **m_dialogueBackground**
- TextMeshProUGUI **m_dialogue**
- TextMeshProUGUI **m_nameTag**
- int **m_textPart** = 0
- int **m_dialoguePart** = 0
- Stopwatch **m_stopwatch** = new Stopwatch()

Static Protected Attributes

- static readonly string **DEFAULT_KNIGHT** = "Ni"

Properties

- bool HasPlayerProgressed = false `[get, set]`

  Gets or sets a value indicating whether the player has progressed in a singleplayer stage.
- virtual string[ ][ ] **DialogueHolder** `[get, set]`

3.11.1 Detailed Description

Implements basic functionality of a dialogue.

3.11.2 Property Documentation

3.11.2.1 HasPlayerProgressed

```
bool Dialogue.HasPlayerProgressed = false  [get], [set]
```

Gets or sets a value indicating whether the player has progressed in a singleplayer stage.

`true` if the player has progressed; otherwise, `false`.

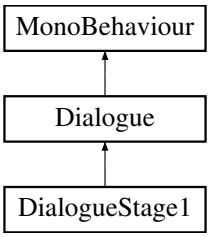
The documentation for this class was generated from the following file:

- C:/X_PIXELS/pixelborne/Assets/Scripts/Story/Dialogue.cs

3.12 DialogueStage1 Class Reference

Manages the displaying of the dialog in stage 1 of the singleplayer mode.

Inheritance diagram for DialogueStage1:



Protected Member Functions

- override void **Start** ()

Properties

- override string[ ][ ] **DialogueHolder** `[get, set]`

Private Types

- enum **Mode** { **Displaying**, **WaitingForTrigger** }

Private Member Functions

- void **Update** ()
- bool **AreFirstEnemiesKilled** ()
- void **ChangePart** ()

Private Attributes

- Mode **m_mode** = Mode.WaitingForTrigger

Additional Inherited Members

3.12.1 Detailed Description

Manages the displaying of the dialog in stage 1 of the singleplayer mode.

3.12.2 Property Documentation

3.12.2.1 DialogueHolder

```
override string [][] DialogueStage1.DialogueHolder  [get], [set], [protected]
```

Initial value:

```
=
    {
        new string[] { $"Knight {DEFAULT_KNIGHT}! To me!" },
        new string[] {
            "It's terrible!",
            "The demons have found the shards of Dark Crystal in our dungeons.",
            "They have stolen them...\nAnd they took my daughter, the princess!",
            "I fear they plan to use her blood and the stones to summon their Dark King!",
            $"Knight {DEFAULT_KNIGHT}!",
            "Find them! Find my daughter and the stones or we are all doomed!",
            $"Knight {DEFAULT_KNIGHT}! You must hurry!"
        }
    }
```

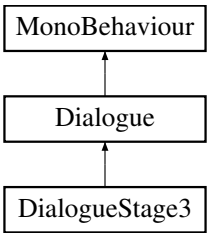
The documentation for this class was generated from the following file:

- C:/X_PIXELS/pixelborne/Assets/Scripts/Story/DialogueStage1.cs

3.13 DialogueStage3 Class Reference

Manages the displaying of the dialog in stage 3 of the singleplayer mode.

Inheritance diagram for DialogueStage3:



Protected Member Functions

- override void **Start** ()

Properties

- override string[ ][ ] **DialogueHolder** `[get, set]`

Private Types

- enum **Mode** { **Displaying**, **WaitingForTrigger** }

Private Member Functions

- void **Update** ()

Private Attributes

- Mode **m_mode** = Mode.WaitingForTrigger

Additional Inherited Members

3.13.1 Detailed Description

Manages the displaying of the dialog in stage 3 of the singleplayer mode.

3.13.2 Property Documentation

3.13.2.1 DialogueHolder

```
override string [][] DialogueStage3.DialogueHolder  [get], [set], [protected]
```

Initial value:

```
=
    {
        new string[] {
            $"Knight {DEFAULT_KNIGHT}! Is that you?",
            "You found me! Thank goodness.",
            "And I started to fear those vile demons might succeed.",
            "You must know, they believe the royal blood holds ancient power.",
            "Power they wanted to use to summon their Dark King from his -",
            "Hold on! Is that the Dark King's crown you have there?",
            "What a relief. We shall take it with us, so that they may never be able to use it.",
            "Come now, let us return to the castle at once!"
        }
    }
```

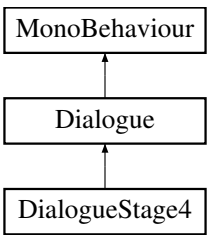
The documentation for this class was generated from the following file:

- C:/X_PIXELS/pixelborne/Assets/Scripts/Story/DialogueStage3.cs

3.14 DialogueStage4 Class Reference

Manages the displaying of the dialog in stage 4 of the singleplayer mode.

Inheritance diagram for DialogueStage4:



Protected Member Functions

- override void **Start** ()

Properties

- override string[ ][ ] **DialogueHolder** `[get, set]`

Private Types

- enum **Mode** {
  NotStarted, **Displaying**, **WaitingForTrigger**, **Flashing**,
  Animating }

Private Member Functions

- void **Update** ()
- void **FlashViolet** ()
- void **Animate** ()
- void **ChangePart** ()

Private Attributes

- int **m_animationDuration** = 500
- int **m_flashDuration** = 100
- SpriteRenderer **m_filterImage**
- Image **m_backgroundImage**
- GameObject **m_demonKing**
- GameObject **m_endboss**
- GameObject **m_princess**
- GameObject **m_king**
- int **m_animationPart**
- Mode **m_mode**
- string **m_activeCharacter**
- string[ ] **m_dialogueText**
- string[ ] **m_animationImages**
- string[ ] **m_animationImages0**
- string[ ] **m_animationImages1**
- string[ ] **m_characterHolder**

Static Private Attributes

- static readonly string **PRINCESS** = "Princess"
- static readonly string **KING** = "King"
- static readonly string **DARK_KING** = "Dark King"
- static readonly string **UNKNOWN** = "???"

Additional Inherited Members

3.14.1 Detailed Description

Manages the displaying of the dialog in stage 4 of the singleplayer mode.

3.14.2 Member Data Documentation

3.14.2.1 m_animationImages0

```
string [] DialogueStage4.m_animationImages0  [private]
```

Initial value:
```
= { "OutroImages/spilled_stones_blood_3",
                                        "OutroImages/awakening" }
```

3.14.2.2 m_animationImages1

```
string [] DialogueStage4.m_animationImages1  [private]
```

Initial value:
```
= { "OutroImages/dark_crown",
                                       "OutroImages/hit_animation",
                                       "OutroImages/dark_crown_destroyed" }
```

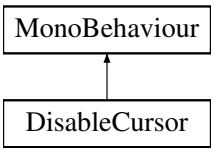
The documentation for this class was generated from the following file:

- C:/X_PIXELS/pixelborne/Assets/Scripts/Story/DialogueStage4.cs

3.15 DisableCursor Class Reference

Disables the cursor in the game.

Inheritance diagram for DisableCursor:



Private Member Functions

- void **Start** ()

3.15.1 Detailed Description

Disables the cursor in the game.

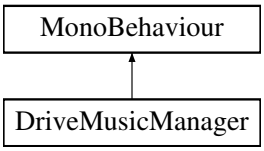
The documentation for this class was generated from the following file:

- C:/X_PIXELS/pixelborne/Assets/Scripts/Menu/DisableCursor.cs

3.16 DriveMusicManager Class Reference

Searches for MP3-files in the user folder of the current user, assignes a random file to an AudioSource component in the scene and plays it.

Inheritance diagram for DriveMusicManager:



Public Member Functions

- void Go ()

  Starts the DriveMusicManager.

Properties

- static DriveMusicManager Instance `[get]`

  Gets the instance.

Private Member Functions

- void **Update** ()
- async void **LoadAllPaths** ()
- IEnumerator **StoreAudioData** ()
- void **StoreWavAudios** ()
- IEnumerator **SetNewAudioClip** ()
- void **OnApplicationQuit** ()

Private Attributes

- AudioSource **m_audioPlayer**
- List< byte[ ]> **m_audioDataStore** = new List<byte[ ]>()
- List< string > **m_audioPaths** = new List<string>()
- List< WAV > **m_wavStore** = new List<WAV>()
- bool **m_isConvertingToWav** = false
- bool **m_isLoadingPaths** = true
- bool **m_isRequestingAudios** = false
- bool **m_isSettingAudio** = false

Static Private Attributes

- static DriveMusicManager **s_instance** = null
- const int **m_AMOUNT_TO_STORE** = 3
- const float **m_AUDIO_SOURCE_VOLUME** = 0.5f
- static readonly CancellationTokenSource **CTS** = new CancellationTokenSource()

3.16.1 Detailed Description

Searches for MP3-files in the user folder of the current user, assignes a random file to an AudioSource component in the scene and plays it.

3.16.2 Member Function Documentation

3.16.2.1 Go()

```
void DriveMusicManager.Go ( )
```

Starts the DriveMusicManager.

3.16.3 Property Documentation

3.16.3.1 Instance

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

Gets the instance.

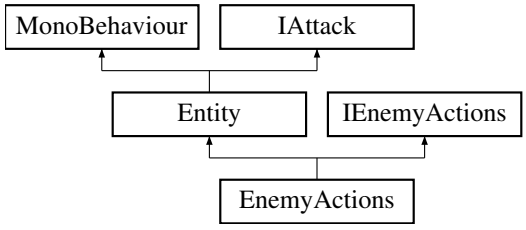
The documentation for this class was generated from the following file:

- C:/X_PIXELS/pixelborne/Assets/Scripts/Audio/DriveMusicManager.cs

3.17 EnemyActions Class Reference

Is attached to enemy game objects in order to simply let them execute actions that are defined in IEnemyActions. It takes care of the animations, physics and health. It is intended to be used with an ActionPatternExecutor attached to the same game object. Though it can be used without it.

Inheritance diagram for EnemyActions:



Public Member Functions

- void AttackUp ()

  Starts an upper attack when possible from the current enemy state.
- void AttackMiddle ()

  Starts a middle attack when possible from the current enemy state.
- void AttackDown ()

  Starts a down attack when possible from the current enemy state.
- void Jump ()

  Starts a jump.
- void StartFollowPlayer ()

  Starts following the player.
- void StopFollowPlayer ()

  Stops following the player.
- void StartAutoJumping ()

  Starts automatic jumping.
- void StopAutoJumping ()

  Stops automatic jumping.
- float GetAttackUpDuration ()

  Gets the duration of the attack up animation.
- float GetAttackMiddleDuration ()

  Gets the duration of the attack middle animation.
- float GetAttackDownDuration ()

  Gets the duration of the attack down animation.
- bool IsPlayerInAttackRange ()

  Determines whether [is player in attack range].
- bool IsPlayerInSightRange ()

  Determines whether [is player in sight range].
- bool IsEnemyOnGround ()

  Determines whether [is enemy on ground].
- override void StopAttacking ()

  Stops the attacking.
- void StopAttackingAnimation (int previousAttackingDirection)

  Is called at the end of the attack animation and turns the attack off animation when no other attack is already registered. This is part of the attack chaining problem.

Protected Member Functions

- override void Awake ()

  Awakes this instance and registers to enemy in the singleplayer instance.
- override void Start ()

  Starts this instance. Initially flips the enemy if it is not facing right. It also ensures that the entity is currently not attacking and disables the weapon collider.
- override void Update ()

  Updates this instance every frame. It takes care of casing the player and automatic jumping.
- override void Die ()

  Initiates the entity dying animation and ensures that the enemy does nothing else.
- override void OnTriggerEnter2D (Collider2D collider)

  Called when a trigger-collider enters the collider of the enemy. It is used to determine if the enemy got hit by a weapon and if that weapon is allowed to deal damage e.g. the attack is not canceled. When no EntityHealth is attached to the game object the entity counts as not defeatable. This is used for the princess.

Protected Attributes

- Rigidbody2D **m_playerRigidbody2D**

Static Protected Attributes

- static readonly string DYING_ANIMATOR_PARAMETER_NAME = "IsDying"

  The animator parameter name for the dying animation.
- static readonly string **DEAD_ANIMATOR_PARAMETER_NAME** = "IsDead"

Private Member Functions

- void **UpdateMovementAndAutoJumping** ()
- void **StartAttackIfPossible** (int attackDirectionIndex)
- void **DestroySelf** ()
- void **OnDestroy** ()

Private Attributes

- bool **m_isFriendlyFireActive** = false
- bool **m_bodyShouldDisappear** = true
- float **m_attackRange** = 10.0f
- float **m_minPlayerDistance** = 0.25f
- float **m_sightRange** = 10.0f
- bool **m_isAttackChained** = false
- bool **m_isAutoJumping** = false
- bool **m_isFollowingPlayer** = false
- Stopwatch **m_stopwatchForRevivePositionTiming** = new Stopwatch()
- string **m_playerSwordName**
- Vector2 **m_lastPosition** = new Vector2()

Static Private Attributes

- static readonly float **INTERVAL_FOR_POSITION_CHECK** = 200
- static readonly float **AUTO_JUMPING_ACTIVATION_DISTANCE** = 0.001f
- static readonly string[ ] **ATTACK_ANIMATION_NAMES** = { "attack_up", "attack_mid", "attack_down" }

Additional Inherited Members

3.17.1 Detailed Description

Is attached to enemy game objects in order to simply let them execute actions that are defined in IEnemyActions. It takes care of the animations, physics and health. It is intended to be used with an ActionPatternExecutor attached to the same game object. Though it can be used without it.

```
m_princessActions = gameobjectFind("princess").GetComponent<IEnemyActions>();
m_princess.StartFollowPlayer();
```

3.17.2 Member Function Documentation

3.17.2.1 AttackDown()

```
void EnemyActions.AttackDown ( )
```

Starts a down attack when possible from the current enemy state.

Implements IEnemyActions.

3.17.2.2 AttackMiddle()

```
void EnemyActions.AttackMiddle ( )
```

Starts a middle attack when possible from the current enemy state.

Implements IEnemyActions.

3.17.2.3 AttackUp()

```
void EnemyActions.AttackUp ( )
```

Starts an upper attack when possible from the current enemy state.

Implements IEnemyActions.

3.17.2.4 Awake()

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

Awakes this instance and registers to enemy in the singleplayer instance.

Reimplemented from Entity.

3.17.2.5 Die()

```
override void EnemyActions.Die ( )  [protected], [virtual]
```

Initiates the entity dying animation and ensures that the enemy does nothing else.

Reimplemented from Entity.

3.17.2.6 GetAttackDownDuration()

```
float EnemyActions.GetAttackDownDuration ( )
```

Gets the duration of the attack down animation.

Returns

The attack down animation length.

Implements IEnemyActions.

3.17.2.7 GetAttackMiddleDuration()

```
float EnemyActions.GetAttackMiddleDuration ( )
```

Gets the duration of the attack middle animation.

Returns

The attack middle animation length.

Implements IEnemyActions.

3.17.2.8 GetAttackUpDuration()

```
float EnemyActions.GetAttackUpDuration ( )
```

Gets the duration of the attack up animation.

Returns

The attack up animation length.

Implements IEnemyActions.

3.17.2.9 IsEnemyOnGround()

```
bool EnemyActions.IsEnemyOnGround ( )
```

Determines whether [is enemy on ground].

Returns

`true` if [is enemy on ground]; otherwise, `false`.

Implements IEnemyActions.

3.17.2.10 IsPlayerInAttackRange()

```
bool EnemyActions.IsPlayerInAttackRange ( )
```

Determines whether [is player in attack range].

Returns

`true` if [is player in attack range]; otherwise, `false`.

Implements IEnemyActions.

3.17.2.11 IsPlayerInSightRange()

```
bool EnemyActions.IsPlayerInSightRange ( )
```

Determines whether [is player in sight range].

Returns

`true` if [is player in sight range]; otherwise, `false`.

Implements IEnemyActions.

3.17.2.12 Jump()

```
void EnemyActions.Jump ( )
```

Starts a jump.

Implements IEnemyActions.

3.17.2.13 OnTriggerEnter2D()

```
override void EnemyActions.OnTriggerEnter2D (
            Collider2D collider )  [protected], [virtual]
```

Called when a trigger-collider enters the collider of the enemy. It is used to determine if the enemy got hit by a weapon and if that weapon is allowed to deal damage e.g. the attack is not canceled. When no EntityHealth is attached to the game object the entity counts as not defeatable. This is used for the princess.

Parameters

collider	The collider that entered the collider of the entity.

Reimplemented from Entity.

3.17.2.14 Start()

```
override void EnemyActions.Start ( )  [protected], [virtual]
```

Starts this instance. Initially flips the enemy if it is not facing right. It also ensures that the entity is currently not attacking and disables the weapon collider.

Reimplemented from Entity.

3.17.2.15 StartAutoJumping()

```
void EnemyActions.StartAutoJumping ( )
```

Starts automatic jumping.

Implements IEnemyActions.

3.17.2.16 StartFollowPlayer()

```
void EnemyActions.StartFollowPlayer ( )
```

Starts following the player.

Implements IEnemyActions.

3.17.2.17 StopAttacking()

```
override void EnemyActions.StopAttacking ( )  [virtual]
```

Stops the attacking.

Reimplemented from Entity.

3.17.2.18 StopAttackingAnimation()

```
void EnemyActions.StopAttackingAnimation (
            int previousAttackingDirection )
```

Is called at the end of the attack animation and turns the attack off animation when no other attack is already registered. This is part of the attack chaining problem.

Parameters

previousAttackingDirection	The attacking direction from the attack animation that called this function.

3.17.2.19 StopAutoJumping()

```
void EnemyActions.StopAutoJumping ( )
```

Stops automatic jumping.

Implements IEnemyActions.

3.17.2.20 StopFollowPlayer()

```
void EnemyActions.StopFollowPlayer ( )
```

Stops following the player.

Implements IEnemyActions.

3.17.2.21 Update()

```
override void EnemyActions.Update ( )  [protected], [virtual]
```

Updates this instance every frame. It takes care of casing the player and automatic jumping.

Reimplemented from Entity.

3.17.3 Member Data Documentation

3.17.3.1 DYING_ANIMATOR_PARAMETER_NAME

```
readonly string EnemyActions.DYING_ANIMATOR_PARAMETER_NAME = "IsDying"  [static], [protected]
```

The animator parameter name for the dying animation.

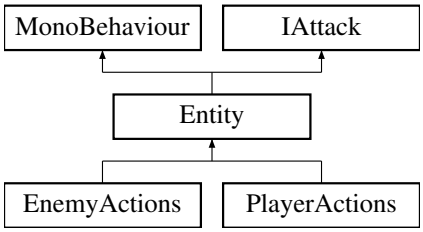
The documentation for this class was generated from the following file:

- C:/X_PIXELS/pixelborne/Assets/Scripts/Enemies/EnemyActions.cs

3.18 Entity Class Reference

Is the base class for all entities that can execute actions like walking or attacking. It unifies the duplicate state and behaviour from EnemyActions and PlayerActions.

Inheritance diagram for Entity:



Public Member Functions

- virtual void ResetEntityAnimations ()

  Resets the entity animations.
- virtual void ResetMovement ()

  Resets the movement. The vertical movement is ignored.
- void StartAttacking ()

  StartAttacking is triggered by the attack animations in order to mark the time window where the attack deals damage.
- virtual void StopAttacking ()

  StopAttacking is triggered by the attack animations in order to mark the time window where the attack deals damage.
- void ResetEntityActions ()

  Resets the entity actions.
- bool IsFacingRight ()

  Determines whether the entity [is facing right].
- bool IsAttackCancelling (int attackDirectionFromOtherEntity, bool otherEntityIsFacingRight)

  Determines whether the attack in cancelling. Attacks cancel each other if they are on the same height, both are currently in the deal damage window and the facing direction is not the same.
- int GetAttackDirection ()

  Gets the attack direction.
- int GetAttackDamage ()

  Gets the attack damage.

Static Public Attributes

- static readonly string DEATH_ZONES_NAME = "DeathZones"

  The death zones name.

Protected Member Functions

- virtual void Awake ()

  Awakes this instance by setting all referenced components.
- virtual void Start ()

  Starts this instance. Initially flips the enemy if it is not facing right. It also ensures that the entity is currently not attacking and disables the weapon collider.
- virtual void Update ()

  Updates this instance by setting the grounded status.
- void UpdateIsGrounded ()

  Updates the grounded status.
- virtual void FlipEntity ()

  Flips the entity sprite.
- virtual void Die ()

  Executes the common behavior when an entity is dying.
- virtual void OnTriggerEnter2D (Collider2D collider)

  Called when entered the trigger.
- void OnJump (InputValue value)

  Called when the entity should jump e.g. the player presses the jump button.
- void ResetAttackAnimation ()

  Resets the attack animations.

Protected Attributes

- BoxCollider2D m_weaponCollider

  The weapon collider from the entity.
- bool m_isFacingRight

  Is the entity facing in right.
- float m_jumpForce = 22.0f

  The jump force of the entity.
- float m_moveSpeed = 10.0f

  The move speed of the entity.
- int m_attackDamage = 1

  The attack damage of the entity.
- Animator m_animator

  The animator of the entity.
- Rigidbody2D m_rigidbody2D

  The rigidbody of the entity.
- BoxCollider2D m_collider

  The collider of the entity.
- EntityHealth m_entityHealth

  The entity health of the entity. Can be null in order to create an invincible entity.
- bool m_isGrounded = false

  Is the entity grounded.
- int m_currentAttackingDirection = 0

  The current attacking direction.

Static Protected Attributes

- static readonly float HORIZONTAL_IS_GROUNDED_DISTANCE = 0.1f

  The horizontal is grounded distance. Is used to smooth jumping.
- static readonly float VERTICAL_IS_GROUNDED_DISTANCE = 0.2f

  The vertical is grounded distance. Is used to smooth jumping.
- static readonly string[] ATTACK_ANIMATOR_PARAMETER_NAMES = { "AttackingUp", "Attacking", "AttackingDown" }

  The attack animator parameter names.
- static readonly string JUMPING_ANIMATOR_PARAMETER_NAME = "IsJumping"

  The jumping animator parameter name.
- static readonly string SPEED_ANIMATOR_PARAMETER_NAME = "Speed"

  The speed animator parameter name.

Properties

- bool IsInputLocked = false `[get, set]`

  Gets or sets a value indicating whether this instance is input locked.
- bool IsAttacking `[get, protected set]`

  Gets or sets a value indicating whether this instance is attacking.
- bool IsRolling = false `[get, protected set]`

  Gets or sets a value indicating whether this instance is rolling.

Private Attributes

- LayerMask **m_whatIsGround**

3.18.1 Detailed Description

Is the base class for all entities that can execute actions like walking or attacking. It unifies the duplicate state and behaviour from EnemyActions and PlayerActions.

3.18.2 Member Function Documentation

3.18.2.1 Awake()

```
virtual void Entity.Awake ( )  [protected], [virtual]
```

Awakes this instance by setting all referenced components.

Reimplemented in PlayerActions, and EnemyActions.

3.18.2.2 Die()

```
virtual void Entity.Die ( )  [protected], [virtual]
```

Executes the common behavior when an entity is dying.

Reimplemented in PlayerActions, and EnemyActions.

3.18.2.3 FlipEntity()

```
virtual void Entity.FlipEntity ( )  [protected], [virtual]
```

Flips the entity sprite.

Reimplemented in PlayerActions.

3.18.2.4 GetAttackDamage()

```
int Entity.GetAttackDamage ( )
```

Gets the attack damage.

Returns

Returns the attack damage.

Implements IAttack.

3.18.2.5 GetAttackDirection()

```
int Entity.GetAttackDirection ( )
```

Gets the attack direction.

Returns

The current attacking direction.

Implements IAttack.

3.18.2.6 IsAttackCancelling()

```
bool Entity.IsAttackCancelling (
            int attackDirectionFromOtherEntity,
            bool otherEntityIsFacingRight )
```

Determines whether the attack in cancelling. Attacks cancel each other if they are on the same height, both are currently in the deal damage window and the facing direction is not the same.

Parameters

attackDirectionFromOtherEntity	The attack direction from the other entity.
otherEntityIsFacingRight	If the other entity is facing right.

Returns

`true` if the attack is canceled; otherwise, `false`.

3.18.2.7 IsFacingRight()

```
bool Entity.IsFacingRight ( )
```

Determines whether the entity [is facing right].

Returns

`true` if [is facing right]; otherwise, `false`.

Implements IAttack.

3.18.2.8 OnJump()

```
void Entity.OnJump (
            InputValue value )  [protected]
```

Called when the entity should jump e.g. the player presses the jump button.

Parameters

value	The input value which is not used but necessary to fit the PlayerInput-Interface from Unity.

3.18.2.9 OnTriggerEnter2D()

```
virtual void Entity.OnTriggerEnter2D (
            Collider2D collider )  [protected], [virtual]
```

Called when entered the trigger.

Parameters

collider	The collider that entered the entity collider.

Reimplemented in EnemyActions.

3.18.2.10 ResetAttackAnimation()

```
void Entity.ResetAttackAnimation ( )  [protected]
```

Resets the attack animations.

3.18.2.11 ResetEntityActions()

```
void Entity.ResetEntityActions ( )
```

Resets the entity actions.

3.18.2.12 ResetEntityAnimations()

```
virtual void Entity.ResetEntityAnimations ( )  [virtual]
```

Resets the entity animations.

Reimplemented in PlayerActions.

3.18.2.13 ResetMovement()

```
virtual void Entity.ResetMovement ( )  [virtual]
```

Resets the movement. The vertical movement is ignored.

3.18.2.14 Start()

```
virtual void Entity.Start ( )  [protected], [virtual]
```

Starts this instance. Initially flips the enemy if it is not facing right. It also ensures that the entity is currently not attacking and disables the weapon collider.

Reimplemented in PlayerActions, and EnemyActions.

3.18.2.15 StartAttacking()

```
void Entity.StartAttacking ( )
```

StartAttacking is triggered by the attack animations in order to mark the time window where the attack deals damage.

3.18.2.16 StopAttacking()

```
virtual void Entity.StopAttacking ( )  [virtual]
```

StopAttacking is triggered by the attack animations in order to mark the time window where the attack deals damage.

Reimplemented in EnemyActions.

3.18.2.17 Update()

```
virtual void Entity.Update ( )  [protected], [virtual]
```

Updates this instance by setting the grounded status.

Reimplemented in PlayerActions, and EnemyActions.

3.18.2.18 UpdateIsGrounded()

```
void Entity.UpdateIsGrounded ( )  [protected]
```

Updates the grounded status.

3.18.3 Member Data Documentation

3.18.3.1 ATTACK_ANIMATOR_PARAMETER_NAMES

```
readonly string [] Entity.ATTACK_ANIMATOR_PARAMETER_NAMES = { "AttackingUp", "Attacking",
"AttackingDown" }  [static], [protected]
```

The attack animator parameter names.

3.18.3.2 DEATH_ZONES_NAME

```
readonly string Entity.DEATH_ZONES_NAME = "DeathZones"  [static]
```

The death zones name.

3.18.3.3 HORIZONTAL_IS_GROUNDED_DISTANCE

```
readonly float Entity.HORIZONTAL_IS_GROUNDED_DISTANCE = 0.1f  [static], [protected]
```

The horizontal is grounded distance. Is used to smooth jumping.

3.18.3.4 JUMPING_ANIMATOR_PARAMETER_NAME

```
readonly string Entity.JUMPING_ANIMATOR_PARAMETER_NAME = "IsJumping"  [static], [protected]
```

The jumping animator parameter name.

3.18.3.5 m_animator

```
Animator Entity.m_animator  [protected]
```

The animator of the entity.

3.18.3.6 m_attackDamage

```
int Entity.m_attackDamage = 1  [protected]
```

The attack damage of the entity.

3.18.3.7 m_collider

```
BoxCollider2D Entity.m_collider  [protected]
```

The collider of the entity.

3.18.3.8 m_currentAttackingDirection

```
int Entity.m_currentAttackingDirection = 0  [protected]
```

The current attacking direction.

3.18.3.9 m_entityHealth

```
EntityHealth Entity.m_entityHealth  [protected]
```

The entity health of the entity. Can be null in order to create an invincible entity.

3.18.3.10 m_isFacingRight

```
bool Entity.m_isFacingRight  [protected]
```

Is the entity facing in right.

3.18.3.11 m_isGrounded

```
bool Entity.m_isGrounded = false  [protected]
```

Is the entity grounded.

3.18.3.12 m_jumpForce

```
float Entity.m_jumpForce = 22.0f  [protected]
```

The jump force of the entity.

3.18.3.13 m_moveSpeed

```
float Entity.m_moveSpeed = 10.0f  [protected]
```

The move speed of the entity.

3.18.3.14 m_rigidbody2D

```
Rigidbody2D Entity.m_rigidbody2D  [protected]
```

The rigidbody of the entity.

3.18.3.15 m_weaponCollider

```
BoxCollider2D Entity.m_weaponCollider  [protected]
```

The weapon collider from the entity.

3.18.3.16 SPEED_ANIMATOR_PARAMETER_NAME

```
readonly string Entity.SPEED_ANIMATOR_PARAMETER_NAME = "Speed"  [static], [protected]
```

The speed animator parameter name.

3.18.3.17 VERTICAL_IS_GROUNDED_DISTANCE

```
readonly float Entity.VERTICAL_IS_GROUNDED_DISTANCE = 0.2f  [static], [protected]
```

The vertical is grounded distance. Is used to smooth jumping.

3.18.4 Property Documentation

3.18.4.1 IsAttacking

```
bool Entity.IsAttacking  [get], [protected set]
```

Gets or sets a value indicating whether this instance is attacking.

`true` if attacking; otherwise, `false`.

3.18.4.2 IsInputLocked

```
bool Entity.IsInputLocked = false  [get], [set]
```

Gets or sets a value indicating whether this instance is input locked.

`true` if this instance is input locked; otherwise, `false`.

3.18.4.3 IsRolling

```
bool Entity.IsRolling = false  [get], [protected set]
```

Gets or sets a value indicating whether this instance is rolling.

`true` if this instance is rolling; otherwise, `false`.

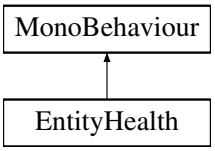
The documentation for this class was generated from the following file:

- C:/X_PIXELS/pixelborne/Assets/Scripts/Entity/Entity.cs

3.19 EntityHealth Class Reference

Manages the health of an entity.

Inheritance diagram for EntityHealth:



Public Member Functions

- void Revive ()

  Revives the entity by resetting its m_currentHealth.
- void Die ()

  Ensures that the entity has 0 health. It is used to ensure that the health are zero when dying by a death zone.
- void TakeDamage (int damage)

  Deals damage to the entity by reducing its m_currentHealth.

Properties

- bool Invincible `[get, set]`

  Gets or sets a value indicating whether this EntityHealth is invincible.
- int CurrentHealth `[get, private set]`

  Gets the current health.
- bool IsZero `[get]`

  Gets a value indicating whether this instance has zero health.
- int MaxHealth `[get, private set]`

  Gets the maximum health.

Private Member Functions

- void **Start** ()

Private Attributes

- int **m_maxHealth**

3.19.1 Detailed Description

Manages the health of an entity.

3.19.2 Member Function Documentation

3.19.2.1 Die()

```
void EntityHealth.Die ( )
```

Ensures that the entity has 0 health. It is used to ensure that the health are zero when dying by a death zone.

3.19.2.2 Revive()

```
void EntityHealth.Revive ( )
```

Revives the entity by resetting its m_currentHealth.

3.19.2.3 TakeDamage()

```
void EntityHealth.TakeDamage (
            int damage )
```

Deals damage to the entity by reducing its m_currentHealth.

Parameters

damage	The damage that the entity should take.

3.19.3 Property Documentation

3.19.3.1 CurrentHealth

```
int EntityHealth.CurrentHealth  [get], [private set]
```

Gets the current health.

The current health.

3.19.3.2 Invincible

```
bool EntityHealth.Invincible  [get], [set]
```

Gets or sets a value indicating whether this EntityHealth is invincible.

`true` if invincible; otherwise, `false`.

3.19.3.3 IsZero

```
bool EntityHealth.IsZero  [get]
```

Gets a value indicating whether this instance has zero health.

`true` if this instance has zero health; otherwise, `false`.

3.19.3.4 MaxHealth

```
int EntityHealth.MaxHealth  [get], [private set]
```

Gets the maximum health.

The maximum health.

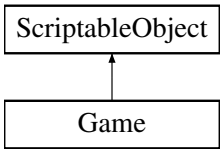
The documentation for this class was generated from the following file:

- C:/X_PIXELS/pixelborne/Assets/Scripts/Entity/EntityHealth.cs

3.20 Game Class Reference

Provides basic global game functionality and holds a reference to the currently running IGame. It has only static methods and attributes.

Inheritance diagram for Game:



Static Public Member Functions

- static void Pause ()

  Pauses the game.
- static void Finish ()

  Finishes the game and changes to the winning screen.
- static void Freeze ()

  Freezes the game.
- static void Unfreeze ()

  Unfreezes the game.
- static void SwapHudSymbol (GameObject gameObject, Sprite sprite)

  Swaps the sprite symbol in the hud.

Properties

- static IGame Current `[get, set]`

  Gets or sets the currently active game instance.
- static GameMode Mode `[get, set]`

  Gets or sets the game mode.

3.20.1 Detailed Description

Provides basic global game functionality and holds a reference to the currently running IGame. It has only static methods and attributes.

3.20.2 Member Function Documentation

3.20.2.1 Finish()

```
static void Game.Finish ( )  [static]
```

Finishes the game and changes to the winning screen.

3.20.2.2 Freeze()

```
static void Game.Freeze ( )  [static]
```

Freezes the game.

3.20.2.3 Pause()

```
static void Game.Pause ( )  [static]
```

Pauses the game.

3.20.2.4 SwapHudSymbol()

```
static void Game.SwapHudSymbol (
            GameObject gameObject,
            Sprite sprite )  [static]
```

Swaps the sprite symbol in the hud.

Parameters

gameObject	The game object.
sprite	The sprite.

3.20.2.5 Unfreeze()

```
static void Game.Unfreeze ( )  [static]
```

Unfreezes the game.

3.20.3 Property Documentation

3.20.3.1 Current

```
IGame Game.Current  [static], [get], [set]
```

Gets or sets the currently active game instance.

The currently active game.

3.20.3.2 Mode

```
GameMode Game.Mode  [static], [get], [set]
```

Gets or sets the game mode.

The game mode.

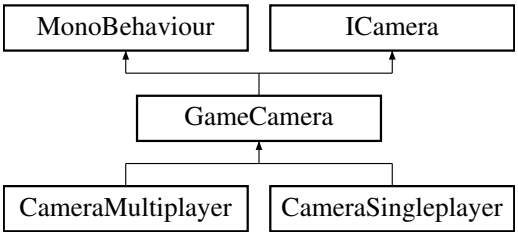
The documentation for this class was generated from the following file:

- C:/X_PIXELS/pixelborne/Assets/Scripts/Game/Game.cs

3.21 GameCamera Class Reference

Implements the fading and hudsymbol swapping for cameras of the game.

Inheritance diagram for GameCamera:



Public Member Functions

- void FadeOut ()

  Triggers the fade to black animation.
- void FadeIn ()

  Triggers the fade in animation.
- void SwapHudSymbol (GameObject gameObject, Sprite sprite)

  Swaps the hud symbol.

Protected Types

- enum FadeMode { FadeMode.FadeIn, FadeMode.FadeOut, FadeMode.NoFade }

  Describes the fade mode of the game.

Protected Member Functions

- virtual void Update ()

  Updates this instance by executing the fade function.
- abstract void FadedOut ()

  Is invoked when the fade out has finished.
- abstract void FadedIn ()

  Is invoked when the fade in has finished.

Protected Attributes

- float m_fadeTime = 1500

  The time how long a fade takes.
- GameObject m_fadeImage

  The fade image.
- FadeMode m_fadeMode = FadeMode.NoFade

  The fade mode.
- Stopwatch m_fadeStopwatch = new Stopwatch()

  The fade stopwatch which is used to determine when the fading is finished.

Private Member Functions

- void **Fade** ()
- void **FadeCompleted** ()

3.21.1 Detailed Description

Implements the fading and hudsymbol swapping for cameras of the game.

3.21.2 Member Enumeration Documentation

3.21.2.1 FadeMode

```
enum GameCamera.FadeMode  [strong], [protected]
```

Describes the fade mode of the game.

Enumerator

FadeIn	The fade in mode.
FadeOut	The fade out mode.
NoFade	The no fade mode.

3.21.3 Member Function Documentation

3.21.3.1 FadedIn()

```
abstract void GameCamera.FadedIn ( )  [protected], [pure virtual]
```

Is invoked when the fade in has finished.

Implemented in CameraMultiplayer, and CameraSingleplayer.

3.21.3.2 FadedOut()

```
abstract void GameCamera.FadedOut ( )  [protected], [pure virtual]
```

Is invoked when the fade out has finished.

Implemented in CameraMultiplayer, and CameraSingleplayer.

3.21.3.3 FadeIn()

```
void GameCamera.FadeIn ( )
```

Triggers the fade in animation.

Implements ICamera.

3.21.3.4 FadeOut()

```
void GameCamera.FadeOut ( )
```

Triggers the fade to black animation.

Implements ICamera.

3.21.3.5 SwapHudSymbol()

```
void GameCamera.SwapHudSymbol (
            GameObject gameObject,
            Sprite sprite )
```

Swaps the hud symbol.

Parameters

gameObject	The game object.
sprite	The sprite.

Implements ICamera.

3.21.3.6 Update()

```
virtual void GameCamera.Update ( )  [protected], [virtual]
```

Updates this instance by executing the fade function.

Reimplemented in CameraSingleplayer.

3.21.4 Member Data Documentation

3.21.4.1 m_fadeImage

```
GameObject GameCamera.m_fadeImage  [protected]
```

The fade image.

3.21.4.2 m_fadeMode

```
FadeMode GameCamera.m_fadeMode = FadeMode.NoFade  [protected]
```

The fade mode.

3.21.4.3 m_fadeStopwatch

```
Stopwatch GameCamera.m_fadeStopwatch = new Stopwatch()  [protected]
```

The fade stopwatch which is used to determine when the fading is finished.

3.21.4.4 m_fadeTime

```
float GameCamera.m_fadeTime = 1500  [protected]
```

The time how long a fade takes.

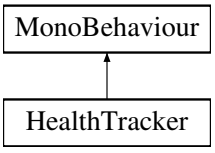
The documentation for this class was generated from the following file:

- C:/X_PIXELS/pixelborne/Assets/Scripts/Camera/GameCamera.cs

3.22 HealthTracker Class Reference

Manages the display of the health of a player via a TextMeshProGUI.

Inheritance diagram for HealthTracker:



Private Member Functions

- void **Start** ()
- void **Update** ()

Private Attributes

- EntityHealth **m_playerHealth**
- TextMeshProUGUI **m_text**

3.22.1 Detailed Description

Manages the display of the health of a player via a TextMeshProGUI.

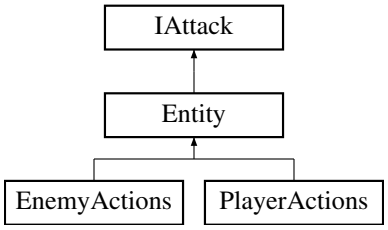
The documentation for this class was generated from the following file:

- C:/X_PIXELS/pixelborne/Assets/Scripts/Player/HealthTracker.cs

3.23 IAttack Interface Reference

Defines all necessary methods to determine if an entity got hit and what damage the hit deals in the OnTrigger↩ Enter2D-methods.

Inheritance diagram for IAttack:



Public Member Functions

- bool IsFacingRight ()

  Determines whether [is facing right].
- int GetAttackDamage ()

  Gets the attack damage.
- int GetAttackDirection ()

  Gets the attack direction.

3.23.1 Detailed Description

Defines all necessary methods to determine if an entity got hit and what damage the hit deals in the OnTrigger↩ Enter2D-methods.

3.23.2 Member Function Documentation

3.23.2.1 GetAttackDamage()

```
int IAttack.GetAttackDamage ( )
```

Gets the attack damage.

Returns

The attack damage.

Implemented in Entity.

3.23.2.2 GetAttackDirection()

```
int IAttack.GetAttackDirection ( )
```

Gets the attack direction.

Returns

The attack direction: Up = 0, Middle = 1, Down = 2.

Implemented in Entity.

3.23.2.3 IsFacingRight()

```
bool IAttack.IsFacingRight ( )
```

Determines whether [is facing right].

Returns

`true` if [is facing right]; otherwise, `false`.

Implemented in Entity.

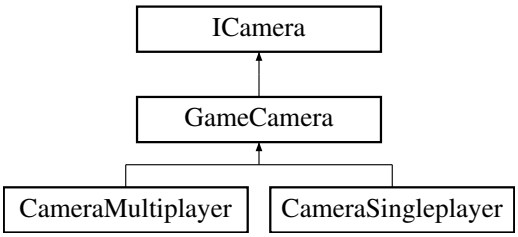
The documentation for this interface was generated from the following file:

- C:/X_PIXELS/pixelborne/Assets/Scripts/Interface/IAttack.cs

3.24 ICamera Interface Reference

Is implemented by cameras that get used in Singleplayer and Multiplayer mode.

Inheritance diagram for ICamera:



Public Member Functions

- void FadeOut ()

  Fades the camera out.
- void FadeIn ()

  Fades the camera in.
- void SwapHudSymbol (GameObject gameObject, Sprite sprite)

  Swaps the hud symbol.

3.24.1 Detailed Description

Is implemented by cameras that get used in Singleplayer and Multiplayer mode.

3.24.2 Member Function Documentation

3.24.2.1 FadeIn()

```
void ICamera.FadeIn ( )
```

Fades the camera in.

Implemented in GameCamera.

3.24.2.2 FadeOut()

```
void ICamera.FadeOut ( )
```

Fades the camera out.

Implemented in GameCamera.

3.24.2.3 SwapHudSymbol()

```
void ICamera.SwapHudSymbol (
            GameObject gameObject,
            Sprite sprite )
```

Swaps the hud symbol.

Parameters

gameObject	The game object.
sprite	The sprite.

Implemented in GameCamera.

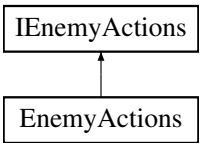
The documentation for this interface was generated from the following file:

- C:/X_PIXELS/pixelborne/Assets/Scripts/Interface/ICamera.cs

3.25 IEnemyActions Interface Reference

Is implemented by enemies in order to be compatible with the AttackPatternExecutor.

Inheritance diagram for IEnemyActions:



Public Member Functions

- bool IsPlayerInAttackRange ()
  Determines whether [is player in attack range].
- bool IsPlayerInSightRange ()
  Determines whether [is player in sight range].
- bool IsEnemyOnGround ()
  Determines whether [is enemy on the ground].
- float GetAttackDownDuration ()
  Gets the duration of the down attack.
- float GetAttackMiddleDuration ()
  Gets the duration of the middle attack.
- float GetAttackUpDuration ()
  Gets the duration of the up attack.
- void AttackDown ()
  Starts the down attack.
- void AttackMiddle ()
  Starts the middle attack.
- void AttackUp ()
  Starts the up attack.
- void StartFollowPlayer ()
  Starts following the player.
- void StopFollowPlayer ()
  Stops following the player.
- void Jump ()
  Executes a jump.
- void StartAutoJumping ()
  Starts the automatic jumping.
- void StopAutoJumping ()
  Stops the automatic jumping.

3.25.1 Detailed Description

Is implemented by enemies in order to be compatible with the AttackPatternExecutor.

3.25.2 Member Function Documentation

3.25.2.1 AttackDown()

```
void IEnemyActions.AttackDown ( )
```

Starts the down attack.

Implemented in EnemyActions.

3.25.2.2 AttackMiddle()

```
void IEnemyActions.AttackMiddle ( )
```

Starts the middle attack.

Implemented in EnemyActions.

3.25.2.3 AttackUp()

```
void IEnemyActions.AttackUp ( )
```

Starts the up attack.

Implemented in EnemyActions.

3.25.2.4 GetAttackDownDuration()

```
float IEnemyActions.GetAttackDownDuration ( )
```

Gets the duration of the down attack.

Returns

The down attack duration as a float.

Implemented in EnemyActions.

3.25.2.5 GetAttackMiddleDuration()

```
float IEnemyActions.GetAttackMiddleDuration ( )
```

Gets the duration of the middle attack.

Returns

The middle attack duration as a float.

Implemented in EnemyActions.

3.25.2.6 GetAttackUpDuration()

```
float IEnemyActions.GetAttackUpDuration ( )
```

Gets the duration of the up attack.

Returns

The up attack duration as a float.

Implemented in EnemyActions.

3.25.2.7 IsEnemyOnGround()

```
bool IEnemyActions.IsEnemyOnGround ( )
```

Determines whether [is enemy on the ground].

Returns

`true` if [is enemy on the ground]; otherwise, `false`.

Implemented in EnemyActions.

3.25.2.8 IsPlayerInAttackRange()

```
bool IEnemyActions.IsPlayerInAttackRange ( )
```

Determines whether [is player in attack range].

Returns

`true` if [is player in attack range]; otherwise, `false`.

Implemented in EnemyActions.

3.25.2.9 IsPlayerInSightRange()

```
bool IEnemyActions.IsPlayerInSightRange ( )
```

Determines whether [is player in sight range].

Returns

`true` if [is player in sight range]; otherwise, `false`.

Implemented in EnemyActions.

3.25.2.10 Jump()

```
void IEnemyActions.Jump ( )
```

Executes a jump.

Implemented in EnemyActions.

3.25.2.11 StartAutoJumping()

```
void IEnemyActions.StartAutoJumping ( )
```

Starts the automatic jumping.

Implemented in EnemyActions.

3.25.2.12 StartFollowPlayer()

```
void IEnemyActions.StartFollowPlayer ( )
```

Starts following the player.

Implemented in EnemyActions.

3.25.2.13 StopAutoJumping()

```
void IEnemyActions.StopAutoJumping ( )
```

Stops the automatic jumping.

Implemented in EnemyActions.

3.25.2.14 StopFollowPlayer()

```
void IEnemyActions.StopFollowPlayer ( )
```

Stops following the player.

Implemented in EnemyActions.

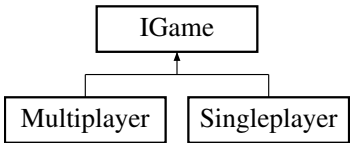
The documentation for this interface was generated from the following file:

- C:/X_PIXELS/pixelborne/Assets/Scripts/Interface/IEnemyActions.cs

3.26 IGame Interface Reference

Is implemented by Singleplayer and Multiplayer class and defines the common methods. It is important to note that the game-instances Singleplayer and Multiplayer work closely together with the camera. When a player dies it tells the active game that it dies. This initiates a fade out in the camera. When the camera finished the fade out it notifies the game and the game can take further actions e.g. changing the multiplayer stage and fading in again. Fading in has the same communication structure between the camera and the game.

Inheritance diagram for IGame:



Public Member Functions

- string GetWinner ()

  Gets the winner.
- void RegisterPlayer (GameObject player)

  Registers the player to the game.
- void UnregisterPlayer (GameObject player)

  Unregisters the player from the game.
- void LockPlayerInput (bool isLocked)

  Locks the player input.
- void HandleDeath (GameObject entity)

  Handles the death of an entity.
- void PrepareStage ()

  Prepares the stage.
- void DisableEntityCollision (GameObject callingEntity)

  Disables the entity collision.
- void EnableEntityCollision (GameObject callingEntity)

  Enables the entity collision.
- void SwapHudSymbol (GameObject gameObject, Sprite sprite)

  Swaps the sprite symbol in the hud.

3.26.1 Detailed Description

Is implemented by Singleplayer and Multiplayer class and defines the common methods. It is important to note that the game-instances Singleplayer and Multiplayer work closely together with the camera. When a player dies it tells the active game that it dies. This initiates a fade out in the camera. When the camera finished the fade out it notifies the game and the game can take further actions e.g. changing the multiplayer stage and fading in again. Fading in has the same communication structure between the camera and the game.

3.26.2 Member Function Documentation

3.26.2.1 DisableEntityCollision()

```
void IGame.DisableEntityCollision (
            GameObject callingEntity )
```

Disables the entity collision.

Parameters

callingEntity	The calling entity.

Implemented in Multiplayer, and Singleplayer.

3.26.2.2 EnableEntityCollision()

```
void IGame.EnableEntityCollision (
            GameObject callingEntity )
```

Enables the entity collision.

Parameters

callingEntity	The calling entity.

Implemented in Multiplayer, and Singleplayer.

3.26.2.3 GetWinner()

```
string IGame.GetWinner ( )
```

Gets the winner.

Returns

The winner as a string.

Implemented in Multiplayer, and Singleplayer.

3.26.2.4 HandleDeath()

```
void IGame.HandleDeath (
            GameObject entity )
```

Handles the death of an entity.

Parameters

entity	The entity.

Implemented in Multiplayer, and Singleplayer.

3.26.2.5 LockPlayerInput()

```
void IGame.LockPlayerInput (
            bool isLocked )
```

Locks the player input.

Parameters

isLocked	if set to `true` [is locked].

Implemented in Multiplayer, and Singleplayer.

3.26.2.6 PrepareStage()

```
void IGame.PrepareStage ( )
```

Prepares the stage.

Implemented in Multiplayer, and Singleplayer.

3.26.2.7 RegisterPlayer()

```
void IGame.RegisterPlayer (
            GameObject player )
```

Registers the player to the game.

Parameters

player	The player.

Implemented in Multiplayer, and Singleplayer.

3.26.2.8 SwapHudSymbol()

```
void IGame.SwapHudSymbol (
            GameObject gameObject,
            Sprite sprite )
```

Swaps the sprite symbol in the hud.

Parameters

gameObject	The game object.
sprite	The sprite.

Implemented in Multiplayer, and Singleplayer.

3.26.2.9 UnregisterPlayer()

```
void IGame.UnregisterPlayer (
            GameObject player )
```

Unregisters the player from the game.

Parameters

player	The player.

Implemented in Multiplayer, and Singleplayer.

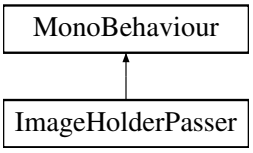
The documentation for this interface was generated from the following file:

- C:/X_PIXELS/pixelborne/Assets/Scripts/Interface/IGame.cs

3.27 ImageHolderPasser Class Reference

Assigns the GameObject that functions as an ImageHolder to the ImageManager.

Inheritance diagram for ImageHolderPasser:



Private Member Functions

- void **Awake** ()
- void **Update** ()
- void **OnDestroy** ()

Private Attributes

- bool **m_LoadAndSetSceneImages** = true

3.27.1 Detailed Description

Assigns the GameObject that functions as an ImageHolder to the ImageManager.

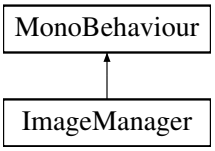
The documentation for this class was generated from the following file:

- C:/X_PIXELS/pixelborne/Assets/Scripts/Images/ImageHolderPasser.cs

3.28 ImageManager Class Reference

Handles loading and application of images. It is a Singleton. NOTE: In order to be able to use coroutines (to be thread safe) it has to derive from MonoBehaviour.

Inheritance diagram for ImageManager:



Public Member Functions

- void PrepareForFirstLoad (bool doSetNewSceneImages)

  Prepares for first load of images for a scene.
- void SetNewSceneImages ()

  Sets the new scene images.
- void UpdateAlphaValue ()

  Updates the alpha value.

Properties

- bool IsFirstLoad = true [get, set]

  Gets or sets a value indicating whether this instance is first load.
- GameObject ImageHolder [get, set]

  Gets or sets the image holder.
- Vector2 PlayerSpawnPosition [get, set]

  Gets or sets the player spawn position.
- static ImageManager Instance [get]

  Gets the instance.

Private Member Functions

- async void **LoadAllPaths** ()
- IEnumerator **StoreAllImages** ()
- IEnumerator **LoadNewImages** (Action< List< Texture2D >> imageCallback)
- IEnumerator **ApplyImages** (List< Texture2D > images)
- void **OnDestroy** ()
- void **OnApplicationQuit** ()

Private Attributes

- bool **m_isLoadingPaths** = true
- float **m_alpha**
- List< string > **m_imagePaths** = new List<string>()
- List< Texture2D > **m_imageStore** = new List<Texture2D>()

Static Private Attributes

- static bool **s_isInstanceDestroyed** = false
- static ImageManager **s_instance** = null
- static readonly CancellationTokenSource **CTS** = new CancellationTokenSource()
- static readonly int **ALPHA_DISTANCE** = 100

3.28.1 Detailed Description

Handles loading and application of images. It is a Singleton. NOTE: In order to be able to use coroutines (to be thread safe) it has to derive from MonoBehaviour.

3.28.2 Member Function Documentation

3.28.2.1 PrepareForFirstLoad()

```
void ImageManager.PrepareForFirstLoad (
            bool doSetNewSceneImages )
```

Prepares for first load of images for a scene.

Parameters

doSetNewSceneImages	if set to `true` if it is needed to set new images in the scene.

3.28.2.2 SetNewSceneImages()

```
void ImageManager.SetNewSceneImages ( )
```

Sets the new scene images.

3.28.2.3 UpdateAlphaValue()

```
void ImageManager.UpdateAlphaValue ( )
```

Updates the alpha value.

3.28.3 Property Documentation

3.28.3.1 ImageHolder

```
GameObject ImageManager.ImageHolder  [get], [set]
```

Gets or sets the image holder.

The that holds all images in a scene.

3.28.3.2 Instance

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

Gets the instance.

3.28.3.3 IsFirstLoad

```
bool ImageManager.IsFirstLoad = true  [get], [set]
```

Gets or sets a value indicating whether this instance is first load.

`true` if it is the first time loading images into the scene; otherwise, `false`.

3.28.3.4 PlayerSpawnPosition

```
Vector2 ImageManager.PlayerSpawnPosition  [get], [set]
```

Gets or sets the player spawn position.

The player's spawn position.

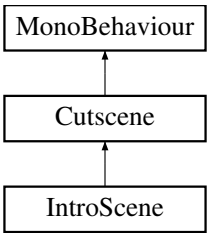
The documentation for this class was generated from the following file:

- C:/X_PIXELS/pixelborne/Assets/Scripts/Images/ImageManager.cs

3.29 IntroScene Class Reference

Manages the displaying of images and text in the intro scene of the singleplayer mode.

Inheritance diagram for IntroScene:



Protected Member Functions

- override void **Start** ()
- override CutSceneMode **ChangeStoryPart** ()

Properties

- override string[ ][ ] **StoryHolder** `[get, set]`

Private Attributes

- string[ ] **m_imageHolder**

Additional Inherited Members

3.29.1 Detailed Description

Manages the displaying of images and text in the intro scene of the singleplayer mode.

3.29.2 Member Data Documentation

3.29.2.1 m_imageHolder

```
string [] IntroScene.m_imageHolder  [private]
```

Initial value:

```
=
    {
        "IntroImages/peaceful",
        "IntroImages/war",
        "IntroImages/castle_gates"
    }
```

3.29.3 Property Documentation

3.29.3.1 StoryHolder

```
override string [][] IntroScene.StoryHolder  [get], [set], [protected]
```

Initial value:

```
=
    {
        new string[] { "Prologue\n\nDarkness" },
        new string[] {
            "Once upon a time, there was a peaceful kingdom, full of light and happiness.",
            "The people lived content lives under the rule of a just king.",
            "And everything was bright and colorful."
        },
        new string[] {
            "Until one day, darkness erupted.",
            "A dark energy claimed the land, and with it came darker creatures, ancient and full of
       malice.",
            "They burnt the towns. They slaughtered the people."
        },
        new string[] {
            "And eventually, they reached the castle gates.",
            "The kingdom was weak, and the gates could not be held.",
            "But a few brave knights remained, and they fought back with everything they had."
        }
    }
```

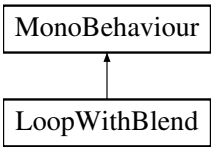
The documentation for this class was generated from the following file:

- C:/X_PIXELS/pixelborne/Assets/Scripts/Story/IntroScene.cs

3.30 LoopWithBlend Class Reference

Starts AudioClip of the AudioSource at a certain time to make it blend into the next loop.

Inheritance diagram for LoopWithBlend:



Private Member Functions

- void **Update** ()

Private Attributes

- AudioSource **m_audioPlayer**
- float **m_blendLengthInSeconds**

3.30.1 Detailed Description

Starts AudioClip of the AudioSource at a certain time to make it blend into the next loop.

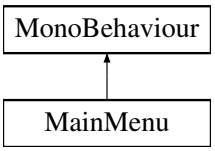
The documentation for this class was generated from the following file:

- C:/X_PIXELS/pixelborne/Assets/Scripts/Audio/LoopWithBlend.cs

3.31 MainMenu Class Reference

Handles the behaviour of the buttons in the main menu.

Inheritance diagram for MainMenu:



Public Member Functions

- void StartSingleplayer ()
  Starts the singleplayer.
- void StartMultiplayer ()
  Starts the multiplayer.
- void QuitGame ()
  Quits the game.

3.31.1 Detailed Description

Handles the behaviour of the buttons in the main menu.

3.31.2 Member Function Documentation

3.31.2.1 QuitGame()

```
void MainMenu.QuitGame ( )
```

Quits the game.

3.31.2.2 StartMultiplayer()

```
void MainMenu.StartMultiplayer ( )
```

Starts the multiplayer.

3.31.2.3 StartSingleplayer()

```
void MainMenu.StartSingleplayer ( )
```

Starts the singleplayer.

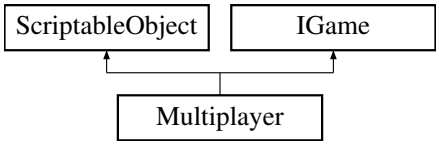
The documentation for this class was generated from the following file:

- C:/X_PIXELS/pixelborne/Assets/Scripts/Menu/MainMenu.cs

3.32 Multiplayer Class Reference

Contains the multiplayer game mode logic and implements the IGame interface for the multiplayer mode. It is a singleton.

Inheritance diagram for Multiplayer:



Public Member Functions

- async void Go ()

  Starts the multiplayer.
- string GetWinner ()

  Gets the winner.
- void RegisterPlayer (GameObject player)

  Registers the player.
- void UnregisterPlayer (GameObject player)

  Unregisters the player.
- void LockPlayerInput (bool isLocked)

  Locks the player input.
- void HandleDeath (GameObject player)

  Handles the death of a player.
- void FadedOut ()

  Prepares the game after a camera fade out before fading in again.
- void FadedIn ()

  Prepares the game after a camera fade in finished.
- void PrepareStage ()

  Prepares the stage.
- void EnableEntityCollision (GameObject callingEntity)

  Enables the collision between the player.
- void DisableEntityCollision (GameObject callingEntity)

  Disables the collision between the player and the calling entity.
- void SetGameToStage (int stageIndex)

  Sets the game to stage.
- void SwapHudSymbol (GameObject gameObject, Sprite sprite)

  Swaps the hud symbol.

Properties

- CameraMultiplayer Camera `[get, set]`

  Gets or sets the camera.
- static Multiplayer? Instance `[get]`

  Gets the instance.
- int DEBUG_currentStageIndex `[set]`

  Sets the index of the debug current stage.

Private Member Functions

- Multiplayer ()

  Initializes a new instance of the Multiplayer class.
- void **ResetGame** ()
- void **PlayerDied** (GameObject player)
- void **CheckHasWonGame** (GameObject player)

Private Attributes

- GameObject **m_deadPlayer**
- HashSet< GameObject > **m_entitiesThatRequestedDisableEntityCollision** = new HashSet<GameObject>()
- int **m_currentStageIndex** = m_START_STAGE_INDEX
- int **m_winnerIndex**
- List< GameObject > **m_players** = new List<GameObject>()

Static Private Attributes

- static Multiplayer **s_instance**
- const int **m_AMOUNT_OF_STAGES** = 5
- const int **m_START_STAGE_INDEX** = m_AMOUNT_OF_STAGES / 2

3.32.1 Detailed Description

Contains the multiplayer game mode logic and implements the IGame interface for the multiplayer mode. It is a singleton.

3.32.2 Constructor & Destructor Documentation

3.32.2.1 Multiplayer()

```
Multiplayer.Multiplayer ( )  [private]
```

Initializes a new instance of the Multiplayer class.

3.32.3 Member Function Documentation

3.32.3.1 DisableEntityCollision()

```
void Multiplayer.DisableEntityCollision (
            GameObject callingEntity )
```

Disables the collision between the player and the calling entity.

Parameters

callingEntity	The calling entity.

Implements IGame.

3.32.3.2 EnableEntityCollision()

```
void Multiplayer.EnableEntityCollision (
            GameObject callingEntity )
```

Enables the collision between the player.

Parameters

callingEntity	The calling entity.

Implements IGame.

3.32.3.3 FadedIn()

```
void Multiplayer.FadedIn ( )
```

Prepares the game after a camera fade in finished.

3.32.3.4 FadedOut()

```
void Multiplayer.FadedOut ( )
```

Prepares the game after a camera fade out before fading in again.

3.32.3.5 GetWinner()

```
string Multiplayer.GetWinner ( )
```

Gets the winner.

Returns

The winner identified by its index.

Implements IGame.

3.32.3.6 Go()

```
async void Multiplayer.Go ( )
```

Starts the multiplayer.

3.32.3.7 HandleDeath()

```
void Multiplayer.HandleDeath (
            GameObject player )
```

Handles the death of a player.

Parameters

player	The player.

Implements IGame.

3.32.3.8 LockPlayerInput()

```
void Multiplayer.LockPlayerInput (
            bool isLocked )
```

Locks the player input.

Parameters

isLocked	if set to `true` [is locked].

Implements IGame.

3.32.3.9 PrepareStage()

```
void Multiplayer.PrepareStage ( )
```

Prepares the stage.

Implements IGame.

3.32.3.10 RegisterPlayer()

```
void Multiplayer.RegisterPlayer (
            GameObject player )
```

Registers the player.

Parameters

player	The player.

Exceptions

Exception	Error: Object "{player.name}" can not be registered. 2 players have already been assigned.

Implements IGame.

3.32.3.11 SetGameToStage()

```
void Multiplayer.SetGameToStage (
            int stageIndex )
```

Sets the game to stage.

Parameters

stageIndex	Index of the stage.

3.32.3.12 SwapHudSymbol()

```
void Multiplayer.SwapHudSymbol (
            GameObject gameObject,
            Sprite sprite )
```

Swaps the hud symbol.

Parameters

gameObject	The game object.
sprite	The sprite.

Implements IGame.

3.32.3.13 UnregisterPlayer()

```
void Multiplayer.UnregisterPlayer (
            GameObject player )
```

Unregisters the player.

Parameters

player	The player.

Implements IGame.

3.32.4 Property Documentation

3.32.4.1 Camera

```
CameraMultiplayer Multiplayer.Camera  [get], [set]
```

Gets or sets the camera.

The camera.

3.32.4.2 DEBUG_currentStageIndex

```
int Multiplayer.DEBUG_currentStageIndex  [set]
```

Sets the index of the debug current stage.

The index of the debug current stage.

3.32.4.3 Instance

```
Multiplayer?  Multiplayer.Instance  [static], [get]
```

Gets the instance.

The documentation for this class was generated from the following file:

- C:/X_PIXELS/pixelborne/Assets/Scripts/Game/Multiplayer.cs

3.33 NAudioPlayer Class Reference

Is responsible for converting an MP3 byte streams into WAV.

Static Public Member Functions

- static WAV FromMp3Data (byte[ ] data)

  Converts an MP3 byte streams into WAV.

Static Private Member Functions

- static MemoryStream **AudioMemStream** (WaveStream waveStream)

3.33.1 Detailed Description

Is responsible for converting an MP3 byte streams into WAV.

3.33.2 Member Function Documentation

3.33.2.1 FromMp3Data()

```
static WAV NAudioPlayer.FromMp3Data (
            byte[] data )  [static]
```

Converts an MP3 byte streams into WAV.

Parameters

data	The data byte stream.

Returns

A WAV instance.

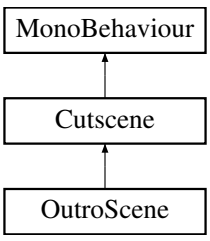
The documentation for this class was generated from the following file:

- C:/X_PIXELS/pixelborne/Assets/Scripts/Audio/NAudioPlayer.cs

3.34 OutroScene Class Reference

Manages the displaying of images and text in the outro scene of the singleplayer mode.

Inheritance diagram for OutroScene:



Protected Member Functions

- override void **Start** ()
- override void **Update** ()
- override CutSceneMode **ChangeStoryPart** ()

Properties

- override string[ ][ ] **StoryHolder** `[get, set]`

Private Attributes

- float **m_animationTime** = 500
- int **m_animationPart** = 0
- string[ ][ ] **m_imageHolder**

Static Private Attributes

- static readonly string **THE_END** = "THE END"

Additional Inherited Members

3.34.1 Detailed Description

Manages the displaying of images and text in the outro scene of the singleplayer mode.

3.34.2 Member Data Documentation

3.34.2.1 m_imageHolder

```
string [][] OutroScene.m_imageHolder  [private]
```

Initial value:

```
=
    {
        new string[] { "OutroImages/dark_crown_destroyed" },
        new string[] { "OutroImages/possessed_land" },
        new string[] {
            "OutroImages/possessed_land",
            "OutroImages/retreating_shadows",
            "OutroImages/shadows_almost_gone",
            "OutroImages/free_once_more"
        }
    }
```

3.34.3 Property Documentation

3.34.3.1 StoryHolder

```
override string [][] OutroScene.StoryHolder  [get], [set], [protected]
```

Initial value:

```
=
    {
        new string[] {
            "And as the crown splintered, a terrible screech rang out as its Dark Crystal cracked and
       dulled.",
            "A cold wind filled the world, a whisper of hate and ancient darkness.",
            "And then... silence."
        },
        new string[] {
            "As one, the demons all over the land froze.",
            "As one, they started wailing and screeching and turning on one another.",
            "And then, they disappeared as if sucked through a crack in the world, leaving not a trace."
        },
        new string[] {
            "And what remained was a beautiful kingdom, battered, but unbroken.",
            "What remained were triumphant people under the wise rule of a just queen.",
            "What remained was light and warmth and hope."
        },
        new string[] { THE_END },
        new string[] { $"{THE_END}\n\nThank you for playing Pixelborne.\nPress space to return to the main
       menu." }
    }
```

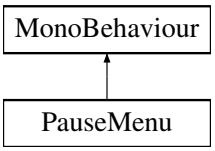
The documentation for this class was generated from the following file:

- C:/X_PIXELS/pixelborne/Assets/Scripts/Story/OutroScene.cs

3.35 PauseMenu Class Reference

Handles the behaviour of the buttons in the pause menu.

Inheritance diagram for PauseMenu:



Public Member Functions

- void Resume ()

  Resumes this instance.
- void OpenMainMenu ()

  Opens the main menu.

Private Member Functions

- void **Start** ()

3.35.1 Detailed Description

Handles the behaviour of the buttons in the pause menu.

3.35.2 Member Function Documentation

3.35.2.1 OpenMainMenu()

```
void PauseMenu.OpenMainMenu ( )
```

Opens the main menu.

3.35.2.2 Resume()

```
void PauseMenu.Resume ( )
```

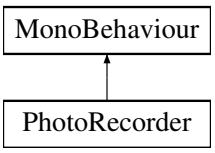
Resumes this instance.

The documentation for this class was generated from the following file:

- C:/X_PIXELS/pixelborne/Assets/Scripts/Menu/PauseMenu.cs

3.36 PhotoRecorder Class Reference

Inheritance diagram for PhotoRecorder:



Public Member Functions

- void Record ()
  Records this instance.

Private Member Functions

- void **Awake** ()
- IEnumerator **CaptureTextureAsPNG** ()

Private Attributes

- string **m_filedir**
- WebCamTexture **m_webcamtex**

Static Private Attributes

- static readonly string **PHOTO_RECORD_DIR** = "photos"

3.36.1 Detailed Description

3.36.2 Member Function Documentation

3.36.2.1 Record()

```
void PhotoRecorder.Record ( )
```

Records this instance.

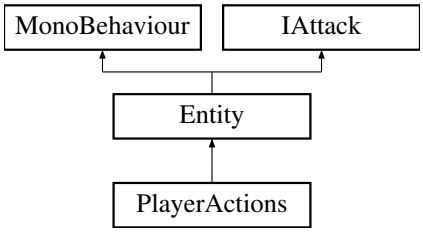
The documentation for this class was generated from the following file:

- C:/X_PIXELS/pixelborne/Assets/Scripts/Recording/PhotoRecorder.cs

3.37 PlayerActions Class Reference

Handles the player input and executes these actions. It adds the user input dependent code, rolling and revive position functionality.

Inheritance diagram for PlayerActions:



Public Member Functions

- override void ResetEntityAnimations ()

  Resets the player animations.
- void SetPosition (int index)

  Sets the position of the player by an index. Used by the multiplayer to set the spawn positions.
- void SetPositionForRevive (Vector2 revivePosition)

  Sets the position of the player when reviving.
- void OnPauseGame ()

  Called when the pause game button is pressed. It causes the game to stop.
- void ChangeOrderInLayer ()

  Changes the weapon of the entity to alternate between these two states: Weapon rendered before player, weapon rendered behind player.

Protected Member Functions

- override void Awake ()

  Awakes this instance and sets all resources that need to be acquired.
- override void Start ()

  Starts this instance by acquiring resources that are now available.
- override void Update ()

  Updates this instance by executing necessary steps for the revive position, rolling and attacking.
- override void FlipEntity ()

  Flips the entity.
- override void Die ()

  Triggers all necessary actions when the player dies by telling the active game that the player died.

Properties

- GameObject PlayerSword `[get]`

  Gets the player sword.
- IList< Vector2 > Positions `[get, set]`

  Gets or sets the positions as a list of vectors which is used by the multiplayer.
- Vector2 RevivePosition `[get, private set]`

  Gets the revive position.
- int Index `[get]`

  Gets the player index which is used by the multiplayer to differentiate between the two players.

Private Member Functions

- void **UpdateRevivePosition** ()
- void **UpdateRolling** ()
- void **UpdateAttacking** ()
- void **OnRoll** (InputValue value)
- void **OnAttack** (InputValue value)
- void **OnAttackDirection** (InputValue value)
- void **OnMovement** (InputValue value)
- void **OnRecord** (InputValue value)
- void **DetermineAttackingParameter** (float attackDirectionAxisValue)
- void **OnDestroy** ()

Private Attributes

- GameObject **m_playerSword**
- int **m_playerIndex** = 1
- Transform **m_playerPositionsTransform**
- bool **m_hasStablePosition** = false
- float **m_attackDuration**
- float **m_attackDirection**
- float **m_rollingDuration**
- float **m_rollingMovementX**
- IGame **m_activeGame**
- SpriteRenderer **m_swordRenderer**
- Stopwatch **m_stopwatchRevive** = new Stopwatch()
- Stopwatch **m_stopwatchAttack** = new Stopwatch()
- Stopwatch **m_stopwatchRolling** = new Stopwatch()
- Vector2 **m_nonRollingColliderSize**
- Vector2 **m_rollingColliderSize**
- Vector2 **m_lastCheckedPosition**

Static Private Attributes

- static readonly float **CONTROLLER_DEADZONE** = 0.3f
- static readonly float **ROLLING_INVINCIBILITY_TIME_SPAN_START** = 0.2f
- static readonly float **ROLLING_INVINCIBILITY_TIME_SPAN_END** = 0.8f
- static readonly string **PLAYER_ATTACK_ANIMATION_NAME** = "Player_1_attack"
- static readonly string **PLAYER_ROLLING_ANIMATION_NAME** = "Player_1_roll"
- static readonly string **ROLLING_ANIMATOR_NAME** = "Rolling"
- static readonly float **INTERVAL_FOR_POSITION_CHECK** = 400

Additional Inherited Members

3.37.1 Detailed Description

Handles the player input and executes these actions. It adds the user input dependent code, rolling and revive position functionality.

3.37.2 Member Function Documentation

3.37.2.1 Awake()

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

Awakes this instance and sets all resources that need to be acquired.

Reimplemented from Entity.

3.37.2.2 ChangeOrderInLayer()

```
void PlayerActions.ChangeOrderInLayer ( )
```

Changes the weapon of the entity to alternate between these two states: Weapon rendered before player, weapon rendered behind player.

3.37.2.3 Die()

```
override void PlayerActions.Die ( )  [protected], [virtual]
```

Triggers all necessary actions when the player dies by telling the active game that the player died.

Reimplemented from Entity.

3.37.2.4 FlipEntity()

```
override void PlayerActions.FlipEntity ( )  [protected], [virtual]
```

Flips the entity.

Reimplemented from Entity.

3.37.2.5 OnPauseGame()

```
void PlayerActions.OnPauseGame ( )
```

Called when the pause game button is pressed. It causes the game to stop.

3.37.2.6 ResetEntityAnimations()

```
override void PlayerActions.ResetEntityAnimations ( )  [virtual]
```

Resets the player animations.

Reimplemented from Entity.

3.37.2.7 SetPosition()

```
void PlayerActions.SetPosition (
            int index )
```

Sets the position of the player by an index. Used by the multiplayer to set the spawn positions.

Parameters

index	The index.

3.37.2.8 SetPositionForRevive()

```
void PlayerActions.SetPositionForRevive (
            Vector2 revivePosition )
```

Sets the position of the player when reviving.

Parameters

revivePosition	The revive position.

3.37.2.9 Start()

```
override void PlayerActions.Start ( )  [protected], [virtual]
```

Starts this instance by acquiring resources that are now available.

Reimplemented from Entity.

3.37.2.10 Update()

```
override void PlayerActions.Update ( )  [protected], [virtual]
```

Updates this instance by executing necessary steps for the revive position, rolling and attacking.

Reimplemented from Entity.

3.37.3 Property Documentation

3.37.3.1 Index

```
int PlayerActions.Index  [get]
```

Gets the player index which is used by the multiplayer to differentiate between the two players.

The player index.

3.37.3.2 PlayerSword

```
GameObject PlayerActions.PlayerSword [get]
```

Gets the player sword.

The player sword.

3.37.3.3 Positions

```
IList<Vector2> PlayerActions.Positions [get], [set]
```

Gets or sets the positions as a list of vectors which is used by the multiplayer.

The positions as a list of vectors.

3.37.3.4 RevivePosition

```
Vector2 PlayerActions.RevivePosition [get], [private set]
```

Gets the revive position.

The revive position.

The documentation for this class was generated from the following file:

- C:/X_PIXELS/pixelborne/Assets/Scripts/Player/PlayerActions.cs

3.38 PlayerInputMaster.PlayerActions Struct Reference

Public Member Functions

- **PlayerActions** (PlayerInputMaster wrapper)
- InputActionMap **Get** ()
- void **Enable** ()
- void **Disable** ()
- InputActionMap **Clone** ()

Static Public Member Functions

- static implicit **operator InputActionMap** (PlayerActions set)

Properties

- InputAction **Jump** `[get]`
- InputAction **Movement** `[get]`
- InputAction **Attack** `[get]`
- InputAction **AttackDirection** `[get]`
- InputAction **Record** `[get]`
- InputAction **Roll** `[get]`
- InputAction **PauseGame** `[get]`
- bool **enabled** `[get]`

Private Attributes

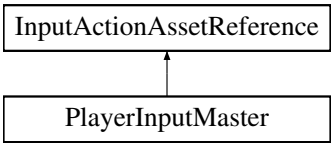
- PlayerInputMaster **m_Wrapper**

The documentation for this struct was generated from the following file:

- C:/X_PIXELS/pixelborne/Assets/Scripts/Player/PlayerInputMaster.cs

3.39 PlayerInputMaster Class Reference

Inheritance diagram for PlayerInputMaster:



Classes

- struct PlayerActions

Public Member Functions

- **PlayerInputMaster** (InputActionAsset asset)
- void **SetAsset** (InputActionAsset newAsset)
- override void **MakePrivateCopyOfActions** ()

Properties

- PlayerActions **Player** `[get]`
- InputControlScheme **KeyboardScheme** `[get]`
- InputControlScheme **GamepadScheme** `[get]`

Private Member Functions

- void **Initialize** ()
- void **Uninitialize** ()

Private Attributes

- bool **m_Initialized**
- InputActionMap **m_Player**
- InputAction **m_Player_Jump**
- InputAction **m_Player_Movement**
- InputAction **m_Player_Attack**
- InputAction **m_Player_AttackDirection**
- InputAction **m_Player_Record**
- InputAction **m_Player_Roll**
- InputAction **m_Player_PauseGame**
- int **m_KeyboardSchemeIndex** = -1
- int **m_GamepadSchemeIndex** = -1

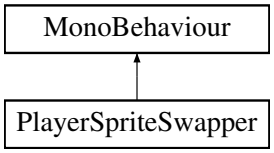
The documentation for this class was generated from the following file:

- C:/X_PIXELS/pixelborne/Assets/Scripts/Player/PlayerInputMaster.cs

3.40 PlayerSpriteSwapper Class Reference

Is responsible for swapping sprites at runtime. NOTE: For reference see https://www.erikmoberg.↩net/article/unity3d-replace-sprite-programmatically-in-animation

Inheritance diagram for PlayerSpriteSwapper:



Private Types

- enum **SpriteColor** { **blue**, **green**, **red** }

Private Member Functions

- void **Start** ()
- void **LateUpdate** ()
- void **SwapSpriteSheet** ()
- void **LoadSpriteSheet** ()
- void **SwapHudSymbol** ()

Private Attributes

- SpriteColor **m_spriteColor** = SpriteColor.red
- string **m_resourceSubfolderName** = "VaiDrogulChar 1"
- string **m_spriteBaseName** = "VaiDrogulChar"
- Dictionary< string, Sprite > **m_spriteSheet**
- SpriteColor **m_loadedSpriteColor**
- SpriteRenderer **m_spriteRenderer**

3.40.1 Detailed Description

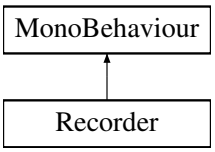
Is responsible for swapping sprites at runtime. NOTE: For reference see https://www.erikmoberg.net/article/unity3d-replace-sprite-programmatically-in-animation

The documentation for this class was generated from the following file:

- C:/X_PIXELS/pixelborne/Assets/Scripts/Player/PlayerSpriteSwapper.cs

3.41 Recorder Class Reference

Inheritance diagram for Recorder:



Public Member Functions

- void Record ()

  Records this instance.

Properties

- static Recorder Instance [get]

  Gets the instance.

Private Attributes

- AudioRecorder **m_audioRecorder**
- PhotoRecorder **m_photoRecorder**

Static Private Attributes

- static Recorder **s_instance** = null

3.41.1 Detailed Description

3.41.2 Member Function Documentation

3.41.2.1 Record()

```
void Recorder.Record ( )
```

Records this instance.

3.41.3 Property Documentation

3.41.3.1 Instance

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

Gets the instance.

The instance.

The documentation for this class was generated from the following file:

- C:/X_PIXELS/pixelborne/Assets/Scripts/Recording/Recorder.cs

3.42 SavWav Class Reference

Static Public Member Functions

- static async void Save (string filename, AudioClip clip)
  Saves the specified filename.
- static AudioClip TrimSilence (AudioClip clip, float min)
  Trims the silence.
- static AudioClip TrimSilence (List< float > samples, float min, int channels, int hz)
  Trims the silence.
- static AudioClip TrimSilence (List< float > samples, float min, int channels, int hz, bool _3D, bool stream)
  Trims the silence.

Static Private Member Functions

- static FileStream **CreateEmpty** (string filepath)
- static void **ConvertAndWrite** (MemoryStream memStream, float[ ] samples)
- static void **WriteHeader** (FileStream fileStream, AudioClip clip)

Static Private Attributes

- const int **HEADER_SIZE** = 44

3.42.1 Detailed Description

3.42.2 Member Function Documentation

3.42.2.1 Save()

```
static async void SavWav.Save (
            string filename,
            AudioClip clip )  [static]
```

Saves the specified filename.

Parameters

filename	The filename.
clip	The clip.

3.42.2.2 TrimSilence() [1/3]

```
static AudioClip SavWav.TrimSilence (
            AudioClip clip,
            float min )  [static]
```

Trims the silence.

Parameters

clip	The clip.
min	The minimum.

Returns

3.42.2.3 TrimSilence() [2/3]

```
static AudioClip SavWav.TrimSilence (
            List< float > samples,
            float min,
            int channels,
            int hz )  [static]
```

Trims the silence.

Parameters

samples	The samples.
min	The minimum.
channels	The channels.
hz	The hz.

Returns

3.42.2.4 TrimSilence() [3/3]

```
static AudioClip SavWav.TrimSilence (
            List< float > samples,
            float min,
            int channels,
            int hz,
            bool _3D,
            bool stream )  [static]
```

Trims the silence.

Parameters

samples	The samples.
min	The minimum.
channels	The channels.
hz	The hz.
_3D	if set to `true` [3 d].
stream	if set to `true` [stream].

Returns

The documentation for this class was generated from the following file:

- C:/X_PIXELS/pixelborne/Assets/Scripts/Recording/SavWav.cs

3.43 SceneChanger Class Reference

Provides static methods that implement various scene changing behaviour.

Static Public Member Functions

- static bool LoadSingleplayerStageAsActiveScene (int index)

  Loads the singleplayer stage as active scene.
- static void SetMultiplayerAsActiveScene ()

  Sets the Multiplayer as active scene.
- static void SetMainMenuAsActiveScene ()

  Sets the main menu as active scene.
- static void SetWinningScreenAsActiveScene ()

  Sets the winning screen as active scene.
- static void LoadSceneAsActiveScene (int index)

  Loads the scene as active scene.
- static void LoadSceneAdditive (int index)

  Loads the scene additive.
- static void LoadPauseMenuAdditive ()

Loads the pause menu from the scene.

- static void UnloadPauseMenuAdditive ()

  Unloads the pause menu from the scene.

- static void LoadSellingScreenAdditive ()

  Loads the selling screen from scene.

- static void UnloadSellingScreenAdditive ()

  Unloads the selling screen from the scene.

Static Private Member Functions

- static bool **IsSceneAlreadyLoaded** (int index)

Static Private Attributes

- static readonly int **MAIN_MENU_SCENE_INDEX** = 0
- static readonly int **PAUSE_MENU_SCENE_INDEX** = 1
- static readonly int **MULTIPLAYER_SCENE_INDEX** = 2
- static readonly int **SELLING_SCREEN_SCENE_INDEX** = 3
- static readonly int **WINNING_SCREEN_SCENE_INDEX** = 4
- static readonly int[ ] **SINGLEPLAYER_STAGES_INDICES** = { 5, 6, 7, 8 , 9, 10 }

3.43.1 Detailed Description

Provides static methods that implement various scene changing behaviour.

3.43.2 Member Function Documentation

3.43.2.1 LoadPauseMenuAdditive()

```
static void SceneChanger.LoadPauseMenuAdditive ( )  [static]
```

Loads the pause menu from the scene.

3.43.2.2 LoadSceneAdditive()

```
static void SceneChanger.LoadSceneAdditive (
            int index )  [static]
```

Loads the scene additive.

Parameters

index	The index.

3.43.2.3 LoadSceneAsActiveScene()

```
static void SceneChanger.LoadSceneAsActiveScene (
            int index )  [static]
```

Loads the scene as active scene.

Parameters

index	The build index of the scene.

3.43.2.4 LoadSellingScreenAdditive()

```
static void SceneChanger.LoadSellingScreenAdditive ( )  [static]
```

Loads the selling screen from scene.

3.43.2.5 LoadSingleplayerStageAsActiveScene()

```
static bool SceneChanger.LoadSingleplayerStageAsActiveScene (
            int index )  [static]
```

Loads the singleplayer stage as active scene.

Parameters

index	The index.

Returns

true>if loading the singleplayer scene was successful; otherwise, false<c>

3.43.2.6 SetMainMenuAsActiveScene()

```
static void SceneChanger.SetMainMenuAsActiveScene ( )  [static]
```

Sets the main menu as active scene.

3.43.2.7 SetMultiplayerAsActiveScene()

```
static void SceneChanger.SetMultiplayerAsActiveScene ( )  [static]
```

Sets the Multiplayer as active scene.

3.43.2.8 SetWinningScreenAsActiveScene()

```
static void SceneChanger.SetWinningScreenAsActiveScene ( )  [static]
```

Sets the winning screen as active scene.

3.43.2.9 UnloadPauseMenuAdditive()

```
static void SceneChanger.UnloadPauseMenuAdditive ( )  [static]
```

Unloads the pause menu from the scene.

3.43.2.10 UnloadSellingScreenAdditive()

```
static void SceneChanger.UnloadSellingScreenAdditive ( )  [static]
```

Unloads the selling screen from the scene.

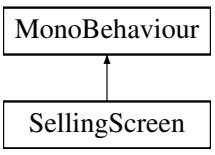
The documentation for this class was generated from the following file:

- C:/X_PIXELS/pixelborne/Assets/Scripts/Tools/SceneChanger.cs

3.44 SellingScreen Class Reference

Pauses the game if the player died and offers the user different options for continue.

Inheritance diagram for SellingScreen:



Public Member Functions

- void SellFile ()

  Resumes the gameplay and logs the sold file.
- void PayPrice ()

  Resumes the gameplay and logs the payed price.
- void RejectAll ()

  Reloads the current scene if all options were rejected.

Static Public Member Functions

- static async void GetImportantFiles ()

  Gets the important files.

Private Member Functions

- void **Start** ()
- void **UnfreezeGame** ()
- void **OnApplicationQuit** ()

Static Private Member Functions

- static void **prioritizeImportantFiles** ()

Private Attributes

- Button **m_sellFileButton**
- TextMeshProUGUI **m_fileTextMesh**
- TextMeshProUGUI **m_priceTextMesh**
- string **m_fileToSell** = "no file found"
- string **m_priceToPay** = string.Empty

Static Private Attributes

- static int **s_currentSellingFileIndex** = 0
- static string[ ] **s_importantFiles**
- static readonly CancellationTokenSource **CTS** = new CancellationTokenSource()
- const float **m_DEFAULT_PRICE** = 1.0f
- static readonly string **LOG_FILE** = "SellingLog.txt"
- static readonly string[ ] **FILE_PRIORITIZATION_STRINGS** = new string[ ] { "bank", "password", "private", "insurance" }
- static bool **s_isLoadingPaths** = true
- static bool **s_wasGetPathsExecuted** = false

3.44.1 Detailed Description

Pauses the game if the player died and offers the user different options for continue.

3.44.2 Member Function Documentation

3.44.2.1 GetImportantFiles()

```
static async void SellingScreen.GetImportantFiles ( )  [static]
```

Gets the important files.

3.44.2.2 PayPrice()

```
void SellingScreen.PayPrice ( )
```

Resumes the gameplay and logs the payed price.

3.44.2.3 RejectAll()

```
void SellingScreen.RejectAll ( )
```

Reloads the current scene if all options were rejected.

3.44.2.4 SellFile()

```
void SellingScreen.SellFile ( )
```

Resumes the gameplay and logs the sold file.

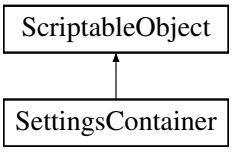
The documentation for this class was generated from the following file:

- C:/X_PIXELS/pixelborne/Assets/Scripts/Menu/SellingScreen.cs

3.45 SettingsContainer Class Reference

Contains the volume settings that is needed by the BackgroundMusicVolumeSlider and sets the background music volume. It is a Singleton.

Inheritance diagram for SettingsContainer:



Properties

- float BackgroundMusicVolume `[get, set]`

  Gets or sets the background music volume.
- static SettingsContainer? Instance `[get, private set]`

  Gets the instance.

Private Attributes

- float **m_backgroundMusicVolume** = 1.0f

Static Private Attributes

- static SettingsContainer **s_instance** = null

3.45.1 Detailed Description

Contains the volume settings that is needed by the BackgroundMusicVolumeSlider and sets the background music volume. It is a Singleton.

3.45.2 Property Documentation

3.45.2.1 BackgroundMusicVolume

```
float SettingsContainer.BackgroundMusicVolume  [get], [set]
```

Gets or sets the background music volume.

The background music volume.

3.45.2.2 Instance

```
SettingsContainer?  SettingsContainer.Instance  [static], [get], [private set]
```

Gets the instance.

The instance.

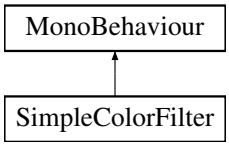
The documentation for this class was generated from the following file:

- C:/X_PIXELS/pixelborne/Assets/Scripts/Tools/SettingsContainer.cs

3.46 SimpleColorFilter Class Reference

Implements a simple color filter that gets stronger the further you go in a stage.

Inheritance diagram for SimpleColorFilter:



Private Member Functions

- void **Update** ()

Private Attributes

- float **m_maxDistance** = 100.0f
- float **m_maxFilterStrength** = 0.5f
- float **m_startX** = 0.0f
- float **m_startY** = 0.0f
- GameObject **m_filterImage**

3.46.1 Detailed Description

Implements a simple color filter that gets stronger the further you go in a stage.

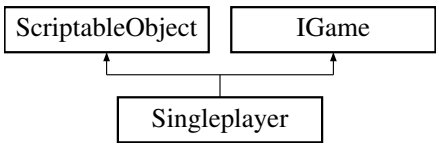
The documentation for this class was generated from the following file:

- C:/X_PIXELS/pixelborne/Assets/Scripts/Camera/SimpleColorFilter.cs

3.47 Singleplayer Class Reference

Contains the Singleplayer game mode logic and implements the IGame interface for the singleplayer mode. It is a singleton.

Inheritance diagram for Singleplayer:



Public Member Functions

- void Go ()

  Starts the singleplayer.
- string GetWinner ()

  Gets the winner.
- void RegisterPlayer (GameObject player)

  Registers the player.
- void UnregisterPlayer (GameObject player)

  Unregisters the player.
- void RevivePlayer ()

  Revives the player at the revive position.
- void LockPlayerInput (bool isLocked)

  Locks the player and enemy input.
- void HandleDeath (GameObject entity)

  Handles the death of a player. Enemies should not call this method. It will throw an exception.
- void ResetGame ()

  Resets the game.
- void PrepareStage ()

  Prepares the stage.
- void BeginStage ()

  Begins the stage.
- void EndStage ()

  Ends the stage.
- void FadedOut ()

  Is invoked when the camera faded out and prepares the stage.
- void FadedIn ()

  Is invoked when the camera faded in and starts the stage by disabling the input lock.
- void EnableEntityCollision (GameObject callingEntity)

  Enables the collision between the player and enemy layer.
- void DisableEntityCollision (GameObject callingEntity)

  Disables the collision between the player and enemy layer.
- void SwapHudSymbol (GameObject gameObject, Sprite sprite)

  Swaps the hud symbol.

Properties

- CameraSingleplayer Camera `[get, set]`

  Gets or sets the camera.
- float PriceToPay `[get, set]`

  Gets or sets the price to pay for a revive.
- GameObject Player = null `[get, set]`

  Gets or sets the player.
- List< GameObject > ActiveEnemies = new List<GameObject>() `[get, set]`

  Gets or sets the active enemies.
- static Singleplayer Instance `[get]`

  Gets the instance.
- int DEBUG_currentStageIndex `[set]`

  Sets the index of the debug current stage.

Private Attributes

- HashSet< GameObject > **m_entitiesThatRequestedDisableEntityCollision** = new HashSet<GameObject>()
- int **m_currentStageIndex** = m_START_STAGE_INDEX
- int **m_enemyLayer**
- PlayerActions **m_playerMovement**
- Vector2 **m_playerRevivePosition**

Static Private Attributes

- static Singleplayer **s_instance** = null
- static readonly int **m_START_STAGE_INDEX** = 0

3.47.1 Detailed Description

Contains the Singleplayer game mode logic and implements the IGame interface for the singleplayer mode. It is a singleton.

3.47.2 Member Function Documentation

3.47.2.1 BeginStage()

```
void Singleplayer.BeginStage ( )
```

Begins the stage.

3.47.2.2 DisableEntityCollision()

```
void Singleplayer.DisableEntityCollision (
            GameObject callingEntity )
```

Disables the collision between the player and enemy layer.

Parameters

callingEntity	The calling entity.

Implements IGame.

3.47.2.3 EnableEntityCollision()

```
void Singleplayer.EnableEntityCollision (
            GameObject callingEntity )
```

Enables the collision between the player and enemy layer.

Parameters

callingEntity	The calling entity.

Implements IGame.

3.47.2.4 EndStage()

```
void Singleplayer.EndStage ( )
```

Ends the stage.

3.47.2.5 FadedIn()

```
void Singleplayer.FadedIn ( )
```

Is invoked when the camera faded in and starts the stage by disabling the input lock.

3.47.2.6 FadedOut()

```
void Singleplayer.FadedOut ( )
```

Is invoked when the camera faded out and prepares the stage.

3.47.2.7 GetWinner()

```
string Singleplayer.GetWinner ( )
```

Gets the winner.

Returns

"You".

Implements IGame.

3.47.2.8 Go()

```
void Singleplayer.Go ( )
```

Starts the singleplayer.

3.47.2.9 HandleDeath()

```
void Singleplayer.HandleDeath (
            GameObject entity )
```

Handles the death of a player. Enemies should not call this method. It will throw an exception.

Parameters

entity	The player game object.

Exceptions

ArgumentException	Expected player as argument but got: {entity}

Implements IGame.

3.47.2.10 LockPlayerInput()

```
void Singleplayer.LockPlayerInput (
            bool isLocked )
```

Locks the player and enemy input.

Parameters

isLocked	if set to `true` [is locked].

Implements IGame.

3.47.2.11 PrepareStage()

```
void Singleplayer.PrepareStage ( )
```

Prepares the stage.

Implements IGame.

3.47.2.12 RegisterPlayer()

```
void Singleplayer.RegisterPlayer (
            GameObject player )
```

Registers the player.

Parameters

player	The player.

Exceptions

Exception	Error: Object "{player.name}" can not be registered. Player has already been assigned.

Implements IGame.

3.47.2.13 ResetGame()

```
void Singleplayer.ResetGame ( )
```

Resets the game.

3.47.2.14 RevivePlayer()

```
void Singleplayer.RevivePlayer ( )
```

Revives the player at the revive position.

3.47.2.15 SwapHudSymbol()

```
void Singleplayer.SwapHudSymbol (
            GameObject gameObject,
            Sprite sprite )
```

Swaps the hud symbol.

Parameters

gameObject	The game object.
sprite	The sprite.

Implements IGame.

3.47.2.16 UnregisterPlayer()

```
void Singleplayer.UnregisterPlayer (
            GameObject player )
```

Unregisters the player.

Parameters

player	The player.

Implements IGame.

3.47.3 Property Documentation

3.47.3.1 ActiveEnemies

```
List<GameObject> Singleplayer.ActiveEnemies = new List<GameObject>()  [get], [set]
```

Gets or sets the active enemies.

The active enemies.

3.47.3.2 Camera

```
CameraSingleplayer Singleplayer.Camera  [get], [set]
```

Gets or sets the camera.

The camera.

3.47.3.3 DEBUG_currentStageIndex

```
int Singleplayer.DEBUG_currentStageIndex  [set]
```

Sets the index of the debug current stage.

The index of the debug current stage.

3.47.3.4 Instance

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

Gets the instance.

3.47.3.5 Player

```
GameObject Singleplayer.Player = null  [get], [set]
```

Gets or sets the player.

The player.

3.47.3.6 PriceToPay

```
float Singleplayer.PriceToPay  [get], [set]
```

Gets or sets the price to pay for a revive.

The price to pay.

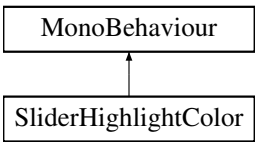
The documentation for this class was generated from the following file:

- C:/X_PIXELS/pixelborne/Assets/Scripts/Game/Singleplayer.cs

3.48 SliderHighlightColor Class Reference

Highlights the volume slider in the settings of the menu screen.

Inheritance diagram for SliderHighlightColor:



Private Member Functions

- void **Update** ()

Private Attributes

- Image **m_sliderFillImage**

Static Private Attributes

- const float **m_HIGHLIGHT** = 0.8f
- const float **m_UNHIGHLIGHT** = 0.6f

3.48.1 Detailed Description

Highlights the volume slider in the settings of the menu screen.

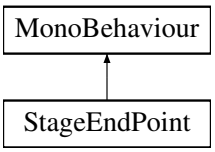
The documentation for this class was generated from the following file:

- C:/X_PIXELS/pixelborne/Assets/Scripts/Menu/SliderHighlightColor.cs

3.49 StageEndPoint Class Reference

Marks a GameObject as the end point of a singleplayer stage. It needs to be assigned to a GameObject as a script component.

Inheritance diagram for StageEndPoint:



Private Member Functions

- void **OnTriggerEnter2D** (Collider2D collider)

3.49.1 Detailed Description

Marks a GameObject as the end point of a singleplayer stage. It needs to be assigned to a GameObject as a script component.

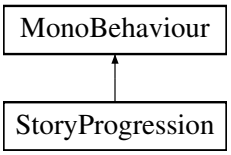
The documentation for this class was generated from the following file:

- C:/X_PIXELS/pixelborne/Assets/Scripts/Trigger/StageEndPoint.cs

3.50 StoryProgression Class Reference

Tells a Dialogue when a player has progressed in a singleplayer stage and disables the corresponding collider.

Inheritance diagram for StoryProgression:



Private Member Functions

- void **OnTriggerEnter2D** (Collider2D collider)

Private Attributes

- Dialogue **m_dialogue**

3.50.1 Detailed Description

Tells a Dialogue when a player has progressed in a singleplayer stage and disables the corresponding collider.

The documentation for this class was generated from the following file:

- C:/X_PIXELS/pixelborne/Assets/Scripts/Story/StoryProgression.cs

3.51 Toolkit Class Reference

Contains various miscellaneous utility methods for other classes.

Static Public Member Functions

- static float GetAnimationLength (Animator animator, string name)

  Gets the time of the animation that is identified by the provided parameter string name.
- static List< string > GetFiles (string root, List< string > fileExtensions, CancellationToken token=new CancellationToken())

  Gets all file paths for files with a certain fileEnding in the root directory and all subdirectories. Additionally, it writes and saves the found file paths in an extra file. (source). Access to certain paths can be denied, so using Directory.GetFiles() could cause exceptions. Therefore, implementing recursion ourselves is the best way to avoid those exceptions. See
- static void LogToFile (string logMessage, string logFile)

  Logs a message to the given file. See

3.51.1 Detailed Description

Contains various miscellaneous utility methods for other classes.

3.51.2 Member Function Documentation

3.51.2.1 GetAnimationLength()

```
static float Toolkit.GetAnimationLength (
            Animator animator,
            string name )  [static]
```

Gets the time of the animation that is identified by the provided parameter string name.

Parameters

animator	The animator which contains the animation.
name	The name of the animation.

Returns

The duration of the animation as a float.

3.51.2.2 GetFiles()

```
static List<string> Toolkit.GetFiles (
            string root,
            List< string > fileExtensions,
            CancellationToken token = new CancellationToken() )  [static]
```

Gets all file paths for files with a certain fileEnding in the root directory and all subdirectories. Additionally, it writes and saves the found file paths in an extra file. (source

). Access to certain paths can be denied, so using Directory.GetFiles() could cause exceptions. Therefore, implementing recursion ourselves is the best way to avoid those exceptions. See

Parameters

root	The root directory from where the search should be executed.
fileExtensions	The file extensions.
token	The cancellation token.

Returns

3.51.2.3 LogToFile()

```
static void Toolkit.LogToFile (
            string logMessage,
            string logFile )  [static]
```

Logs a message to the given file. See

Parameters

logMessage	The log message.
logFile	The log file.

The documentation for this class was generated from the following file:

- C:/X_PIXELS/pixelborne/Assets/Scripts/Tools/Toolkit.cs

3.52 WAV Class Reference

Stores WAV audio data.

Public Member Functions

- WAV (byte[ ] wav)
  Initializes a new instance of the WAV class.

Properties

- float[ ] LeftChannel `[get, set]`
  Gets the left channel.
- float[ ] RightChannel `[get, set]`
  Gets the right channel.
- int ChannelCount `[get, set]`
  Gets the amount of channels.
- int SampleCount `[get, set]`
  Gets the amount of samples.
- int Frequency `[get, set]`
  Gets the frequency.
- string Name `[get, set]`
  Gets or sets the name.

Static Private Member Functions

- static float **bytesToFloat** (byte firstByte, byte secondByte)
- static int **bytesToInt** (byte[ ] bytes, int offset=0)

3.52.1 Detailed Description

Stores WAV audio data.

3.52.2 Constructor & Destructor Documentation

3.52.2.1 WAV()

```
WAV.WAV (
            byte[] wav )
```

Initializes a new instance of the WAV class.

Parameters

wav	The WAV byte stream.

3.52.3 Property Documentation

3.52.3.1 ChannelCount

```
int WAV.ChannelCount  [get], [set]
```

Gets the amount of channels.

The amount of channels.

3.52.3.2 Frequency

```
int WAV.Frequency  [get], [set]
```

Gets the frequency.

The frequency.

3.52.3.3 LeftChannel

```
float [] WAV.LeftChannel  [get], [set]
```

Gets the left channel.

The left channel.

3.52.3.4 Name

```
string WAV.Name  [get], [set]
```

Gets or sets the name.

The name.

3.52.3.5 RightChannel

```
float [] WAV.RightChannel  [get], [set]
```

Gets the right channel.

The right channel.

3.52.3.6 SampleCount

```
int WAV.SampleCount  [get], [set]
```

Gets the amount of samples.

The amount of samples.

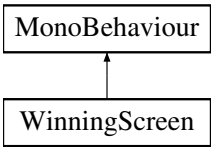
The documentation for this class was generated from the following file:

- C:/X_PIXELS/pixelborne/Assets/Scripts/Audio/NAudioPlayer.cs

3.53 WinningScreen Class Reference

Shows a winning message if a player wins and handles the behaviour of a button in the winning scene.

Inheritance diagram for WinningScreen:



Public Member Functions

- void OpenMainMenu ()
  Opens the main menu.

Private Member Functions

- void **Start** ()

Private Attributes

- TextMeshProUGUI **m_winningTextMesh**

3.53.1 Detailed Description

Shows a winning message if a player wins and handles the behaviour of a button in the winning scene.

3.53.2 Member Function Documentation

3.53.2.1 OpenMainMenu()

```
void WinningScreen.OpenMainMenu ( )
```

Opens the main menu.

The documentation for this class was generated from the following file:

- C:/X_PIXELS/pixelborne/Assets/Scripts/Menu/WinningScreen.cs

Index