Celestial Rewind

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:

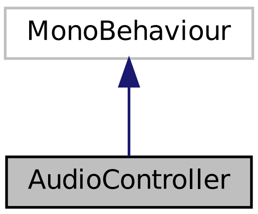
Chapter 3

Class Documentation

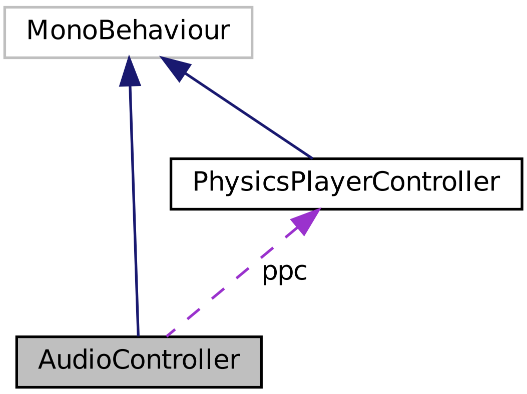
3.1 AudioController Class Reference

A controller for player audio.

Inheritance diagram for AudioController:



Collaboration diagram for AudioController:



Public Member Functions

- IEnumerator WaitForNextStep ()

  A coroutine method that waits until the next step audio should be played
- IEnumerator WaitTilOpen ()

  A coroutine method that waits until the ttOpenCloseSounds has finished, then starts the PlayTimeRewind method.
- IEnumerator **WaitForRewind** ()

Public Attributes

- AudioSource **audioSource**
- AudioClip[ ] **walkSounds**
- AudioClip **landSound**
- AudioSource **timeAudio**
- AudioClip **timeRewindSound**
- AudioClip **ttOpenCloseSounds**
- AudioClip **rewindStartSound**
- AudioClip **rewindLimitWarningSound**
- float **waitStep** = .5f
- PhysicsPlayerController **ppc**

Private Member Functions

- void Update ()

  Controls player walking and time rewind sounds.
- void TimeTurnerOpen ()

  a method triggered by pressing the time rewind button that plays the ttOpenCloseSounds sound effect
- void TimeTurnerClose ()

  A method triggered by releasing the time rewind button that plays the ttOpenCloseSounds sound effect
- void **PlayTimeRewind** ()
- void **PlayRewindSFX** ()

Private Attributes

- int **walkSoundIndex**
- bool **playSound** = true
- bool **ttclosed** = true
- bool **landed**

3.1.1 Detailed Description

A controller for player audio.

3.1.2 Member Function Documentation

3.1.2.1 TimeTurnerClose()

```
void AudioController.TimeTurnerClose ( )  [inline], [private]
```

A method triggered by releasing the time rewind button that plays the ttOpenCloseSounds sound effect

3.1.2.2 TimeTurnerOpen()

```
void AudioController.TimeTurnerOpen ( )  [inline], [private]
```

a method triggered by pressing the time rewind button that plays the ttOpenCloseSounds sound effect

3.1.2.3 Update()

```
void AudioController.Update ( )  [inline], [private]
```

Controls player walking and time rewind sounds.

3.1.2.4 WaitForNextStep()

```
IEnumerator AudioController.WaitForNextStep ( )  [inline]
```

A coroutine method that waits until the next step audio should be played

Returns

A WaitForSeconds object.

3.1.2.5 WaitTilOpen()

```
IEnumerator AudioController.WaitTilOpen ( )  [inline]
```

A coroutine method that waits until the ttOpenCloseSounds has finished, then starts the PlayTimeRewind method.

Returns

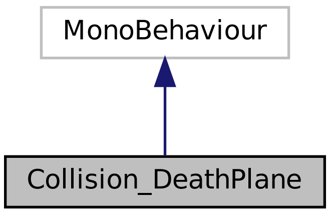
>A WaitForSeconds object.

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

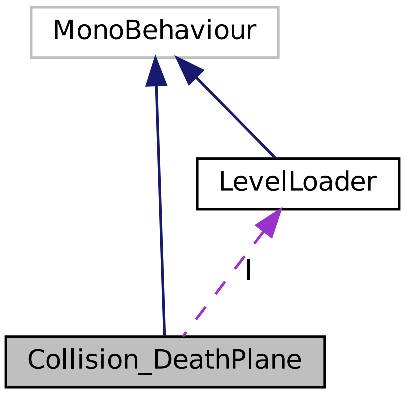
- Assets/Scripts/AudioController.cs

3.2 Collision_DeathPlane Class Reference

Inheritance diagram for Collision_DeathPlane:



Collaboration diagram for Collision_DeathPlane:



Public Member Functions

- void **Start** ()
- void **OnCollisionEnter** (Collision collision)

Private Attributes

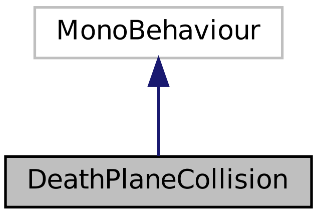
- LevelLoader **l**
- GameObject **p**

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

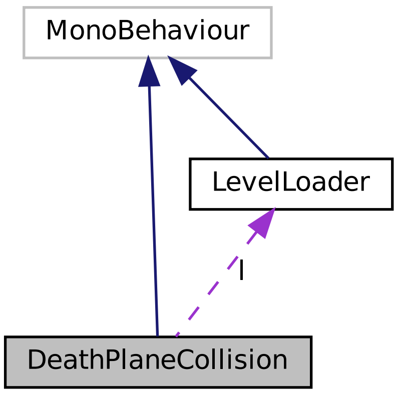
- Assets/Scripts/Collision_DeathPlane.cs

3.3 DeathPlaneCollision Class Reference

Inheritance diagram for DeathPlaneCollision:



Collaboration diagram for DeathPlaneCollision:



Public Member Functions

- void **Start** ()
- void **OnTriggerEnter** (Collider other)

Private Attributes

- LevelLoader **l**
- GameObject **p**

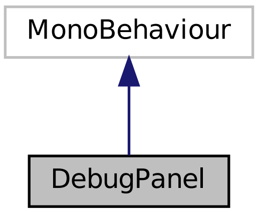
The documentation for this class was generated from the following file:

- Assets/Scripts/DeathPlaneResart.cs

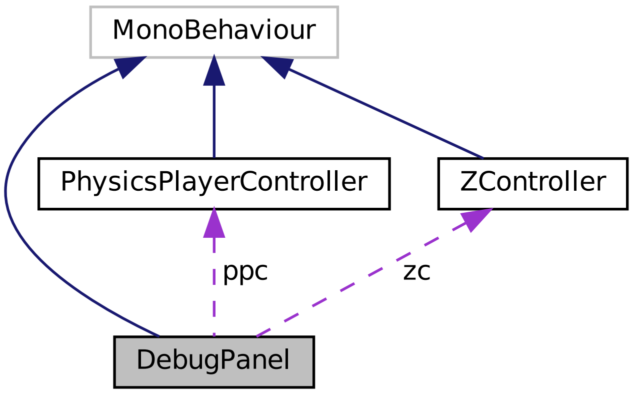
3.4 DebugPanel Class Reference

A controller for the debug settings. Currently contains a manually maintained map of UI elements to the corresponding object value.

Inheritance diagram for DebugPanel:



Collaboration diagram for DebugPanel:



Public Attributes

- TMP_InputField Gravity

  A text field representing gravity's value
- TMP_InputField RewindSpeed

  A text field representing how often per second a rewind capture is performed
- TMP_InputField Drag

  A text field representing drag
- List< TMP_InputField > Move

A list of input fields corresponding to the three values of a player's movement acceleration

- List< TMP_InputField > Speed

  A list of input fields corresponding to the three values of a player's max velocity
- TMP_InputField RLimit

  A text field representing how long rewind can be held before time resumes.
- TMP_InputField RCooldown

  A text field representing how long the player must wait after burning out the rewind before they can use it again.
- TMP_InputField Leniency

  The leniency for the ZController's check for no motion.
- PhysicsPlayerController ppc

  The PhysicsPlayerController the fields are referencing.
- ZController zc

  The ZController the fields are referencing
- GameObject panel

  The parent container for all the text fields.
- Toggle UseForce

  the toggle to choose to used force movement over kinematic movement;

Private Member Functions

- void Start ()

  Set up all of the elements. If they have not been set in the editor, they will be extracted from the children of the panel GameObject. For each element, the corresponding value is pulled from game data and populated into the text fields.
- void Update ()

  Every frame, so long as the panel itself is active, all values in the gui will be copied into their corresponding fields in the game. This is done through TryParse calls, where a failure will result in the original value being maintained. If the key (f1) for the debug is called, the panel will be toggled.

3.4.1 Detailed Description

A controller for the debug settings. Currently contains a manually maintained map of UI elements to the corresponding object value.

Not as automated as I'd like but that would be difficult to do without reflection, which is already used slightly in instantiation.

3.4.2 Member Function Documentation

3.4.2.1 Start()

```
void DebugPanel.Start ( )  [inline], [private]
```

Set up all of the elements. If they have not been set in the editor, they will be extracted from the children of the panel GameObject. For each element, the corresponding value is pulled from game data and populated into the text fields.

3.4.2.2 Update()

```
void DebugPanel.Update ( ) [inline], [private]
```

Every frame, so long as the panel itself is active, all values in the gui will be copied into their corresponding fields in the game. This is done through TryParse calls, where a failure will result in the original value being maintained. If the key (f1) for the debug is called, the panel will be toggled.

3.4.3 Member Data Documentation

3.4.3.1 Drag

```
TMP_InputField DebugPanel.Drag
```

A text field representing drag

Possibly could be removed now

3.4.3.2 Gravity

```
TMP_InputField DebugPanel.Gravity
```

A text field representing gravity's value

3.4.3.3 Leniency

```
TMP_InputField DebugPanel.Leniency
```

The leniency for the ZController's check for no motion.

3.4.3.4 Move

```
List<TMP_InputField> DebugPanel.Move
```

A list of input fields corresponding to the three values of a player's movement acceleration

3.4.3.5 panel

```
GameObject DebugPanel.panel
```

The parent container for all the text fields.

3.4.3.6 ppc

```
PhysicsPlayerController DebugPanel.ppc
```

The PhysicsPlayerController the fields are referencing.

3.4.3.7 RCooldown

```
TMP_InputField DebugPanel.RCooldown
```

A text field representing how long the player must wait after burning out the rewind before they can use it again.

3.4.3.8 RewindSpeed

```
TMP_InputField DebugPanel.RewindSpeed
```

A text field representing how often per second a rewind capture is performed

3.4.3.9 RLimit

```
TMP_InputField DebugPanel.RLimit
```

A text field representing how long rewind can be held before time resumes.

3.4.3.10 Speed

```
List<TMP_InputField> DebugPanel.Speed
```

A list of input fields corresponding to the three values of a player's max velocity

3.4.3.11 UseForce

```
Toggle DebugPanel.UseForce
```

the toggle to choose to used force movement over kinematic movement;

3.4.3.12 zc

```
ZController DebugPanel.zc
```

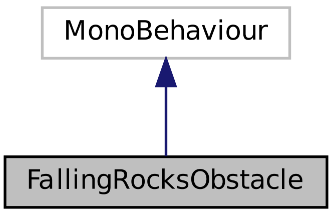
The ZController the fields are referencing

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

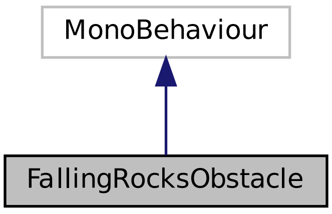
- Assets/Scripts/UI/DebugPanel.cs

3.5 FallingRocksObstacle Class Reference

Inheritance diagram for FallingRocksObstacle:



Collaboration diagram for FallingRocksObstacle:



Public Attributes

- float **moveSpeed** = 0.05f
- float **timeout** = 2f

Private Member Functions

- void Start ()

  place script on the prefab that SpawningRocks script will spawn.
- void **FixedUpdate** ()
- IEnumerator Timeout ()

3.5.1 Member Function Documentation

3.5.1.1 Timeout()

```
IEnumerator FallingRocksObstacle.Timeout ( )  [inline], [private]
```

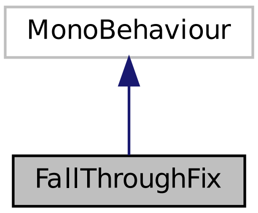
destroys obstacles after they have already fallen, after they already

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

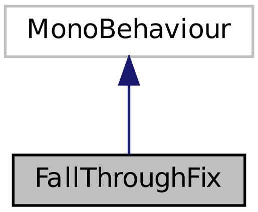
- Assets/Scripts/FallingRocksObstacle.cs

3.6 FallThroughFix Class Reference

Inheritance diagram for FallThroughFix:



Collaboration diagram for FallThroughFix:



Public Attributes

- float **UnitsToMoveUp** = 1

Private Member Functions

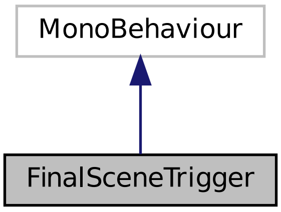
- void **OnTriggerEnter** (Collider other)

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

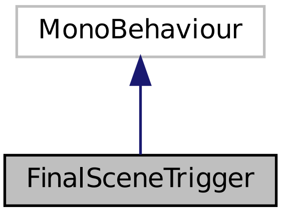
- Assets/Scripts/FallThroughFix.cs

3.7 FinalSceneTrigger Class Reference

Inheritance diagram for FinalSceneTrigger:



Collaboration diagram for FinalSceneTrigger:



Public Attributes

- GameObject[ ] **targets**
- bool **Disable** = false

Private Member Functions

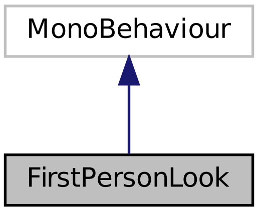
- void **OnTriggerEnter** (Collider other)

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

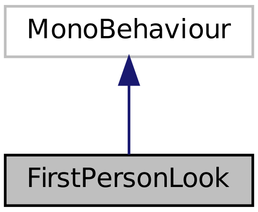
- Assets/Scripts/FinalSceneTrigger.cs

3.8 FirstPersonLook Class Reference

Inheritance diagram for FirstPersonLook:



Collaboration diagram for FirstPersonLook:



Public Attributes

- float **turnSpeed** = 1
- float **mouseXSensitiviy** = 1
- float **mouseYSensitiviy** = 1
- Transform **cam**
- Transform **orientation**
- Transform **player**
- Transform **playerObj**

Private Member Functions

- void **Start** ()
- void **Update** ()
- void **MyInput** ()
- void **rotateCamera** ()
- void **TurnPlayer2** ()
- void **TurnPlayer1** ()
- void **resetTurn** ()

Private Attributes

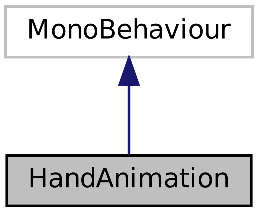
- Vector2 **turn**
- float **horizontalInput**
- float **veritcalInput**

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

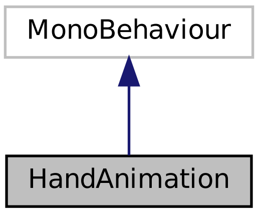
- Assets/Scripts/TestMovement/FirstPersonLook.cs

3.9 HandAnimation Class Reference

Inheritance diagram for HandAnimation:



Collaboration diagram for HandAnimation:



Public Attributes

- Animator timeRewindAnimator

  The animation controller for the chronometer
- Animator handRewindAnimator

  The animation controller for the player hand
- GameObject goodParticlePrefab

  The particle system for active rewindi
- GameObject **badParticlePrefab**
- KeyCode **rewindPrimary** = KeyCode.Mouse0
- KeyCode **rewindSecondary** = KeyCode.Q

Private Member Functions

- void **Update** ()
- void **setParticlesActive** ()

3.9.1 Member Data Documentation

3.9.1.1 goodParticlePrefab

`GameObject HandAnimation.goodParticlePrefab`

The particle system for active rewindi

3.9.1.2 handRewindAnimator

`Animator HandAnimation.handRewindAnimator`

The animation controller for the player hand

3.9.1.3 timeRewindAnimator

`Animator HandAnimation.timeRewindAnimator`

The animation controller for the chronometer

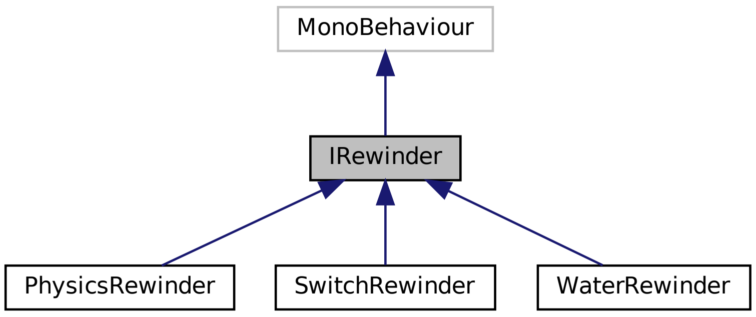
The documentation for this class was generated from the following file:

- Assets/Scripts/HandAnimation.cs

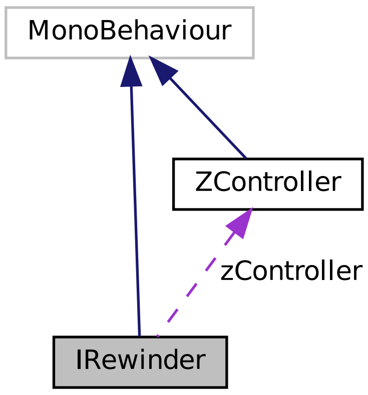
3.10 IRewinder Class Reference

An abstraction of the rewinder functionality for use with a ZController to rewind an object.

Inheritance diagram for IRewinder:



Collaboration diagram for IRewinder:



Public Member Functions

- virtual void Start ()

  Initializes SnapState list, sets initial position and rotation, and extracts the ZController from parent if it was not explicitly set. Finally, pauses the object, awaiting the ZController's signal.
- virtual bool NeedUpdate ()

  Checks to see if the object has moved since the last state capture. Uses ZController.Approximate(Vector3, Vector3) to compare the previous state's position with the current's. If they are not equivalent within the range of leniency, then a signal is sent stating that an update is required.
- virtual void Store ()

  Stores the current position and rotation of the object as well as the current frame number on the rewind stack if the object needs an update.
- virtual SnapState RewindState (int count)

  Rewinds the states from the stack frame one state at a time, allowing for easier speed-up of rewind. If there are no states left, the function returns null
- virtual void Reset ()

  Resets the object to its initial position and rotation and clears the SnapState stack.
- abstract void Play ()

  Continues time wherever it was left off, however the concrete implementation needs to accomplish this.
- abstract void Pause ()

  Pauses the movement of the object however the concrete implementation needs to accomplish this.
- bool HasStates ()

  Checks whether the SnapState stack contains elements.

Public Attributes

- ZController zController

  Parent ZController of the current GameObject; Primarily for use of ZController.Approximate(Vector3, Vector3) method.

Protected Attributes

- Stack< SnapState > states

  Stack of SnapState objects; used to stash and then unwind different points within the IRewinder's lifecycle over the course of the ZController's rewind capture.
- int frameNumber

  A variable to keep track of what state the IRewinder is currently on. Increments with each call to store and decrements as the object is rewound.
- bool printDebug

  Debug variable to toggle output in the debug console; in child methods, whenever debug is required, wrap it with an if statment checking this field.

Properties

- Vector3 startPos `[get, protected set]`

  Initial position state of the attachee GameObject
- Quaternion startRot `[get, protected set]`

  Initial rotation state of the attachee GameObject

3.10.1 Detailed Description

An abstraction of the rewinder functionality for use with a ZController to rewind an object.

3.10.2 Member Function Documentation

3.10.2.1 HasStates()

```
bool IRewinder.HasStates ( )  [inline]
```

Checks whether the SnapState stack contains elements.

Returns

3.10.2.2 NeedUpdate()

```
virtual bool IRewinder.NeedUpdate ( )  [inline], [virtual]
```

Checks to see if the object has moved since the last state capture. Uses ZController.Approximate(Vector3, Vector3) to compare the previous state's position with the current's. If they are not equivalent within the range of leniency, then a signal is sent stating that an update is required.

Returns

True if the object has moved

3.10.2.3 Pause()

```
abstract void IRewinder.Pause ( )  [pure virtual]
```

Pauses the movement of the object however the concrete implementation needs to accomplish this.

Implemented in SwitchRewinder, PhysicsRewinder, and WaterRewinder.

3.10.2.4 Play()

```
abstract void IRewinder.Play ( )  [pure virtual]
```

Continues time wherever it was left off, however the concrete implementation needs to accomplish this.

Implemented in SwitchRewinder, PhysicsRewinder, and WaterRewinder.

3.10.2.5 Reset()

```
virtual void IRewinder.Reset ( )  [inline], [virtual]
```

Resets the object to its initial position and rotation and clears the SnapState stack.

3.10.2.6 RewindState()

```
virtual SnapState IRewinder.RewindState (
             int count )  [inline], [virtual]
```

Rewinds the states from the stack frame one state at a time, allowing for easier speed-up of rewind. If there are no states left, the function returns null

Parameters

count	The number of frames that should be undone in a single call to this method. <remark>Can make the animation choppy if this number is too high!</remark>

Returns

The popped SnapState or null if there are no states remaining.

Reimplemented in SwitchRewinder, and PhysicsRewinder.

3.10.2.7 Start()

```
virtual void IRewinder.Start ( )  [inline], [virtual]
```

Initializes SnapState list, sets initial position and rotation, and extracts the ZController from parent if it was not explicitly set. Finally, pauses the object, awaiting the ZController's signal.

Reimplemented in PhysicsRewinder, SwitchRewinder, and WaterRewinder.

3.10.2.8 Store()

```
virtual void IRewinder.Store ( )  [inline], [virtual]
```

Stores the current position and rotation of the object as well as the current frame number on the rewind stack if the object needs an update.

Reimplemented in PhysicsRewinder, and SwitchRewinder.

3.10.3 Member Data Documentation

3.10.3.1 frameNumber

```
int IRewinder.frameNumber  [protected]
```

A variable to keep track of what state the IRewinder is currently on. Increments with each call to store and decrements as the object is rewound.

3.10.3.2 printDebug

```
bool IRewinder.printDebug  [protected]
```

Debug variable to toggle output in the debug console; in child methods, whenever debug is required, wrap it with an if statment checking this field.

3.10.3.3 states

```
Stack<SnapState> IRewinder.states  [protected]
```

Stack of SnapState objects; used to stash and then unwind different points within the IRewinder's lifecycle over the course of the ZController's rewind capture.

3.10.3.4 zController

```
ZController IRewinder.zController
```

Parent ZController of the current GameObject; Primarily for use of ZController.Approximate(Vector3, Vector3) method.

3.10.4 Property Documentation

3.10.4.1 startPos

```
Vector3 IRewinder.startPos  [get], [protected set]
```

Initial position state of the attachee GameObject

3.10.4.2 startRot

```
Quaternion IRewinder.startRot  [get], [protected set]
```

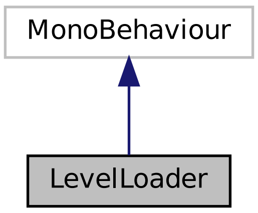
Initial rotation state of the attachee GameObject

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

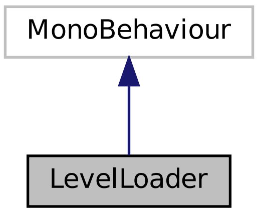
- Assets/Scripts/Rewinders/IRewinder.cs

3.11 LevelLoader Class Reference

Inheritance diagram for LevelLoader:



Collaboration diagram for LevelLoader:



Public Member Functions

- IEnumerator LoadLevel (int sceneIndex)

  Triggers the actual change in scenes after validation from previous method and after scene transition animation is played
- void ChangeScene (int sceneIndex)

  Level loader mode if public Target Bool is enabled Allows a designer to chose which scene to warp to for any given level loader The chosen stage must be in the build settings to be a valid warp. Gice bad warp error if index is not in build settings
- void NextScene ()

  Level loader default A transition trigger will default to loading the next level in the build settings and will loop back to index 0 at after the final scene
- void PreviousScene ()

  A transition that will load the previous scene and then loop to the last scene if at scene index 0
- void ReloadScene ()

  Simply reloads the scene while providing the transition amimation

Public Attributes

- List< Scene > **SceneList**
- Animator **Transition**
- float **TransitionTime** = 1

Private Member Functions

- void Start ()

  Start method gets all available scenes for checking if they are available to warp to in ChangeScene()

3.11.1 Member Function Documentation

3.11.1.1 ChangeScene()

```
void LevelLoader.ChangeScene (
            int sceneIndex )  [inline]
```

Level loader mode if public Target Bool is enabled Allows a designer to chose which scene to warp to for any given level loader The chosen stage must be in the build settings to be a valid warp. Gice bad warp error if index is not in build settings

3.11.1.2 LoadLevel()

```
IEnumerator LevelLoader.LoadLevel (
            int sceneIndex )  [inline]
```

Triggers the actual change in scenes after validation from previous method and after scene transition animation is played

3.11.1.3 NextScene()

```
void LevelLoader.NextScene ( )  [inline]
```

Level loader default A transition trigger will default to loading the next level in the build settings and will loop back to index 0 at after the final scene

3.11.1.4 PreviousScene()

```
void LevelLoader.PreviousScene ( )  [inline]
```

A transition that will load the previous scene and then loop to the last scene if at scene index 0

3.11.1.5 ReloadScene()

```
void LevelLoader.ReloadScene ( )  [inline]
```

Simply reloads the scene while providing the transition amimation

3.11.1.6 Start()

```
void LevelLoader.Start ( )  [inline], [private]
```

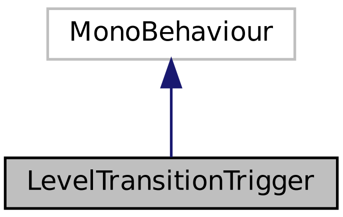
Start method gets all available scenes for checking if they are available to warp to in ChangeScene()

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

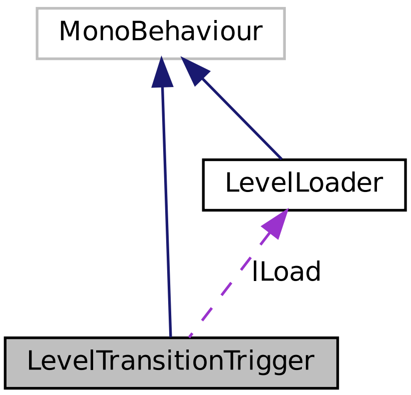
- Assets/Scripts/Transitions/LevelLoader.cs

3.12 LevelTransitionTrigger Class Reference

Inheritance diagram for LevelTransitionTrigger:



Collaboration diagram for LevelTransitionTrigger:



Public Member Functions

- void OnTriggerEnter (Collider other)

    Detects if colliding with player and then triggers the level loader to begin transitioning to the next stage

Public Attributes

- LevelLoader **lLoad**
- bool **TargetScene**
- int **SelectedScene** = 0

3.12.1 Member Function Documentation

3.12.1.1 OnTriggerEnter()

```
void LevelTransitionTrigger.OnTriggerEnter (
             Collider other )  [inline]
```

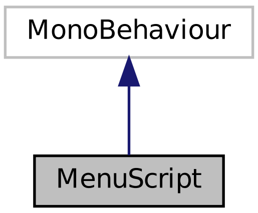
Detects if colliding with player and then triggers the level loader to begin transitioning to the next stage

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

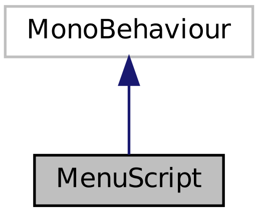
- Assets/Scripts/Transitions/LevelTransitionTrigger.cs

3.13 MenuScript Class Reference

Inheritance diagram for MenuScript:



Collaboration diagram for MenuScript:



Public Attributes

- GameObject **panel**

Private Member Functions

- void Update ()

  Function to display a small menu when the Escape key is pressed. Allows the player to restart a scene if they get stuck.

3.13.1 Member Function Documentation

3.13.1.1 Update()

```
void MenuScript.Update ( )  [inline], [private]
```

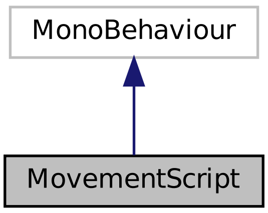
Function to display a small menu when the Escape key is pressed. Allows the player to restart a scene if they get stuck.

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

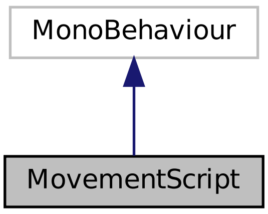
- Assets/Scripts/UI/MenuScript.cs

3.14 MovementScript Class Reference

Inheritance diagram for MovementScript:



Collaboration diagram for MovementScript:



Public Member Functions

- IEnumerator **WaitForNextStep** ()

Public Attributes

- Transform **orientation**
- Rigidbody **rb**
- AudioSource **audioSource**
- AudioSource **timeAudio**
- float **moveSpeed** = 1
- float **jumpForce**
- float **jumpCooldown**
- float **airMultiplier**
- bool **readyToJump**
- AudioClip[ ] **walkSounds**
- AudioClip **timeRewindSound**

- float **waitStep** = .5f
- KeyCode **jumpKey** = KeyCode.Space
- float **playerHeight**
- float **groundDrag**
- LayerMask **whatIsGround**
- bool **grounded**

Private Member Functions

- void **Start** ()
- void **Update** ()
- void **FixedUpdate** ()
- void **MyInput** ()
- void **MovePlayer** ()
- void **SpeedControl** ()
- void **Jump** ()
- void **ResetJump** ()

Private Attributes

- int **walkSoundIndex**
- bool **playSound** = true
- float **horizontalInput**
- float **verticalInput**
- Vector3 **moveDir**

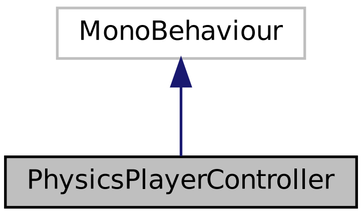
The documentation for this class was generated from the following file:

- Assets/Scripts/TestMovement/MovementScript.cs

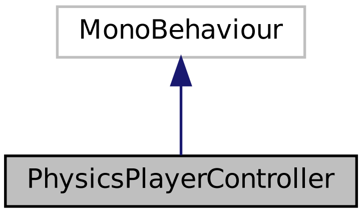
3.15 PhysicsPlayerController Class Reference

A Rigidbody based player controller.

Inheritance diagram for PhysicsPlayerController:



Collaboration diagram for PhysicsPlayerController:



Public Member Functions

- void UpdateCamera ()

  Updates camera rotations using player mouse input. Player GameObject is also transformed for y rotations, allowing forward to always be where the player is facing.
- void DoMovement (ref float xVel, ref float zVel)

  Calculates the player's x and z direction movement. Movement is applied using Rigidbody.MovePosition by applying the playerAcceleration with the second equation of motion. If the current velocity is higher than the max, however, it will be reduced to the max and stored in xVel and zVel .
- void DoJump (ref float yVel)

  Calculates the player's vertical velocity and processes jump input. Checks for ground using a raycast, then if the player is on the ground and jump is pressed enacts an impulse force upwards using the value specified in playerAcceleration. If not on the ground and the velocity is greater than the max speed, yVel will be set to the value in maxSpeed.
- void OnCollisionEnter (Collision collision)

  Detects a collision with the player and calculates the force of the impact. If the impact is too high, the player will be killed.

Public Attributes

- Vector3 playerAcceleration = new Vector3(10, 5, 10)

  A Vector3 containing the accelleration value for the player in each direction.
- Vector3 maxSpeed = new Vector3(4, 15, 4)

  A Vector3 containing the maximum absolute velocity that the player can go in any direction.
- bool grounded

  A bool representing whether or not the player is currently touching a ground.
- bool useForce = true

  A bool representing whether or not the player is using force to move.
- Transform cam

  The Transform for the player's main Camera.
- float mouseXSensitiviy = 1

  The mouse sensitivity for the camera going up and down.
- float mouseYSensitiviy = 1

  The mouse sensitivity for the camera going side to side.
- bool jump

  Caches whether or not the jump button is pressed for use between Update and FixedUpdate

Private Member Functions

- void Start ()

  Attempts to set the Rigidbody and camera Transform if not set within the editor, and locks the CursorState.
- void Update ()

  Calls UpdateCamera and captures player input.
- void FixedUpdate ()

  Updates the player's movement based on input; DoMovement(ref float, ref float) and DoJump(ref float). Also applies velocity cap.
- void PlayerMove ()

  Calculates the players x and z movement using rigidBody.AddForce by multiplying the playerAcceleration by the mass of the player. If there is no input, the velocity of the player is devided by 1.1 every physics update.

Private Attributes

- Rigidbody rigidBody

  The attachee GameObject's Rigidbody.
- LayerMask **ground**
- Vector2 turn

  A Vector2 keeping track of the current x and y rotations of the player camera.
- Vector3 horizontalInput

  Caches the player input as a variable for use between Update and FixedUpdate
- float **SlowMultiplier** = 1.1f

3.15.1 Detailed Description

A Rigidbody based player controller.

3.15.2 Member Function Documentation

3.15.2.1 DoJump()

```
void PhysicsPlayerController.DoJump (
            ref float yVel )  [inline]
```

Calculates the player's vertical velocity and processes jump input. Checks for ground using a raycast, then if the player is on the ground and jump is pressed enacts an impulse force upwards using the value specified in playerAcceleration. If not on the ground and the velocity is greater than the max speed, *yVel* will be set to the value in maxSpeed.

Parameters

yVel	A reference to the z velocity of the player; will be capped to the maxSpeed value if too high

3.15.2.2 DoMovement()

```
void PhysicsPlayerController.DoMovement (
            ref float xVel,
            ref float zVel )  [inline]
```

Calculates the player's x and z direction movement. Movement is applied using Rigidbody.MovePosition by applying the playerAcceleration with the second equation of motion. If the current velocity is higher than the max, however, it will be reduced to the max and stored in *xVel* and *zVel* .

Parameters

xVel	A reference to the x velocity of the player; will be capped to the maxSpeed value if too high
zVel	A reference to the z velocity of the player; will be capped to the maxSpeed value if too high

3.15.2.3 FixedUpdate()

```
void PhysicsPlayerController.FixedUpdate ( )  [inline], [private]
```

Updates the player's movement based on input; DoMovement(ref float, ref float) and DoJump(ref float).

Also applies velocity cap.

3.15.2.4 OnCollisionEnter()

```
void PhysicsPlayerController.OnCollisionEnter (
            Collision collision )  [inline]
```

Detects a collision with the player and calculates the force of the impact. If the impact is too high, the player will be killed.

3.15.2.5 PlayerMove()

```
void PhysicsPlayerController.PlayerMove ( )  [inline], [private]
```

Calculates the players x and z movement using rigidBody.AddForce by multiplying the playerAcceleration by the mass of the player. If there is no input, the velocity of the player is devided by 1.1 every physics update.

3.15.2.6 Start()

```
void PhysicsPlayerController.Start ( )  [inline], [private]
```

Attempts to set the Rigidbody and camera Transform if not set within the editor, and locks the CursorState.

3.15.2.7 Update()

```
void PhysicsPlayerController.Update ( )  [inline], [private]
```

Calls UpdateCamera and captures player input.

3.15.2.8 UpdateCamera()

```
void PhysicsPlayerController.UpdateCamera ( )  [inline]
```

Updates camera rotations using player mouse input. Player GameObject is also transformed for y rotations, allowing forward to always be where the player is facing.

3.15.3 Member Data Documentation

3.15.3.1 cam

```
Transform PhysicsPlayerController.cam
```

The Transform for the player's main Camera.

3.15.3.2 grounded

```
bool PhysicsPlayerController.grounded
```

A bool representing whether or not the player is currently touching a ground.

3.15.3.3 horizontalInput

```
Vector3 PhysicsPlayerController.horizontalInput  [private]
```

Caches the player input as a variable for use between Update and FixedUpdate

3.15.3.4 jump

```
bool PhysicsPlayerController.jump
```

Caches whether or not the jump button is pressed for use between Update and FixedUpdate

3.15.3.5 maxSpeed

```
Vector3 PhysicsPlayerController.maxSpeed = new Vector3(4, 15, 4)
```

A Vector3 containing the maximum absolute velocity that the player can go in any direction.

3.15.3.6 mouseXSensitiviy

```
float PhysicsPlayerController.mouseXSensitiviy = 1
```

The mouse sensitivity for the camera going up and down.

3.15.3.7 mouseYSensitiviy

```
float PhysicsPlayerController.mouseYSensitiviy = 1
```

The mouse sensitivity for the camera going side to side.

3.15.3.8 playerAcceleration

```
Vector3 PhysicsPlayerController.playerAcceleration = new Vector3(10, 5, 10)
```

A Vector3 containing the accelleration value for the player in each direction.

3.15.3.9 rigidBody

```
Rigidbody PhysicsPlayerController.rigidBody  [private]
```

The attachee GameObject's Rigidbody.

3.15.3.10 turn

```
Vector2 PhysicsPlayerController.turn  [private]
```

A Vector2 keeping track of the current x and y rotations of the player camera.

3.15.3.11 useForce

```
bool PhysicsPlayerController.useForce = true
```

A bool representing whether or not the player is using force to move.

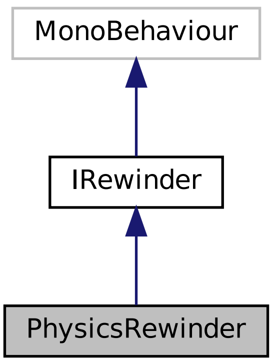
The documentation for this class was generated from the following file:

- Assets/Scripts/PhysicsPlayerController.cs

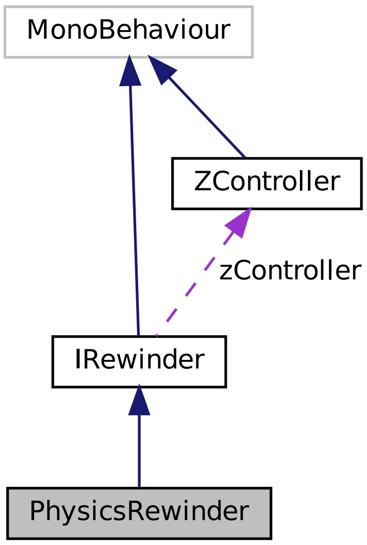
3.16 PhysicsRewinder Class Reference

An implementation of IRewinder made for operating with Rigidbody physics.

Inheritance diagram for PhysicsRewinder:



Collaboration diagram for PhysicsRewinder:



Public Member Functions

- override void Start ()

  Initializes the Rigidbody if not set in the editor.
  Initializes SnapState list, sets initial position and rotation, and extracts the ZController from parent if it was not explicitly set. Finally, pauses the object, awaiting the ZController's signal.
- override void Store ()

  Stores the current position and rotation of the object as well as the current frame number on the rewind stack if the object needs an update.
  Also stores the Rigidbody's current velocity and current angular velocity.
- override SnapState RewindState (int count)

  Rewinds the states from the stack frame one state at a time, allowing for easier speed-up of rewind. If there are no states left, the function returns null
  Sets the rewind velocity to the popped SnapState's velocity.
- void Update ()

  Updates the rewindVelocity every frame, so long as the the Rigidbody is not in kinematic mode.
- override void Play ()

  Continues physics for the object, and begins by applying the force if there is no history within the SnapState stack. If there is a state within the stack, attempts to resume movement by applying the previous state's velocity
- override void Pause ()

  Halts physics for the object and sets the Rigidbody to kinematic mode.
- void OnCollisionStay (Collision collision)

  In the case of collision while the GameObject is rewinding, a force needs to be applied to the colliding object since the Rigidbody cannot do it for itself in kinematic mode.

Public Attributes

- Vector3 startForce = Vector3.zero

  The initial force that should be applied to the Rigidbody when time starts playing.
- Rigidbody rb

  The attachee GameObject's attached Rigidbody.

Private Attributes

- Vector3 rewindVelocity

  A cached state of the previously popped SnapState's velocity; used for calculating the force imparted onto any non-rewound object while rewinding time (primarily the plyaer)

Additional Inherited Members

3.16.1 Detailed Description

An implementation of IRewinder made for operating with Rigidbody physics.

3.16.2 Member Function Documentation

3.16.2.1 OnCollisionStay()

```
void PhysicsRewinder.OnCollisionStay (
            Collision collision )  [inline]
```

In the case of collision while the GameObject is rewinding, a force needs to be applied to the colliding object since the Rigidbody cannot do it for itself in kinematic mode.

Using the difference between the next state's velocity and the cached rewind velocity, the delta is multiplied by the Rigidbody's mass and applied to the colliding Rigidbody as a force. In this implementation, the change in time is ignored because it produces better results.

Parameters

collision	A Collision containing data about the colliding body.

3.16.2.2 Pause()

```
override void PhysicsRewinder.Pause ( )  [inline], [virtual]
```

Halts physics for the object and sets the Rigidbody to kinematic mode.

Implements IRewinder.

3.16.2.3 Play()

```
override void PhysicsRewinder.Play ( )  [inline], [virtual]
```

Continues physics for the object, and begins by applying the force if there is no history within the SnapState stack. If there is a state within the stack, attempts to resume movement by applying the previous state's velocity

Implements IRewinder.

3.16.2.4 RewindState()

```
override SnapState PhysicsRewinder.RewindState (
             int count )  [inline], [virtual]
```

Rewinds the states from the stack frame one state at a time, allowing for easier speed-up of rewind. If there are no states left, the function returns null

Sets the rewind velocity to the popped SnapState's velocity.

Parameters

count	Rewinds the states from the stack frame one state at a time, allowing for easier speed-up of rewind. If there are no states left, the function returns null

Returns

Rewinds the states from the stack frame one state at a time, allowing for easier speed-up of rewind. If there are no states left, the function returns null

Reimplemented from IRewinder.

3.16.2.5 Start()

```
override void PhysicsRewinder.Start ( )  [inline], [virtual]
```

Initializes the Rigidbody if not set in the editor.

Initializes SnapState list, sets initial position and rotation, and extracts the ZController from parent if it was not explicitly set. Finally, pauses the object, awaiting the ZController's signal.

Reimplemented from IRewinder.

3.16.2.6 Store()

```
override void PhysicsRewinder.Store ( )  [inline], [virtual]
```

Stores the current position and rotation of the object as well as the current frame number on the rewind stack if the object needs an update.

Also stores the Rigidbody's current velocity and current angular velocity.

Reimplemented from IRewinder.

3.16.2.7 Update()

```
void PhysicsRewinder.Update ( )  [inline]
```

Updates the rewindVelocity every frame, so long as the the Rigidbody is not in kinematic mode.

3.16.3 Member Data Documentation

3.16.3.1 rb

```
Rigidbody PhysicsRewinder.rb
```

The attachee GameObject's attached Rigidbody.

3.16.3.2 rewindVelocity

```
Vector3 PhysicsRewinder.rewindVelocity  [private]
```

A cached state of the previously popped SnapState's velocity; used for calculating the force imparted onto any non-rewound object while rewinding time (primarily the plyaer)

3.16.3.3 startForce

```
Vector3 PhysicsRewinder.startForce = Vector3.zero
```

The initial force that should be applied to the Rigidbody when time starts playing.

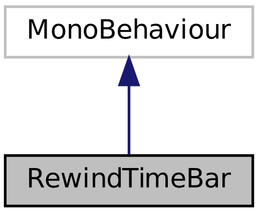
The documentation for this class was generated from the following file:

- Assets/Scripts/Rewinders/PhysicsRewinder.cs

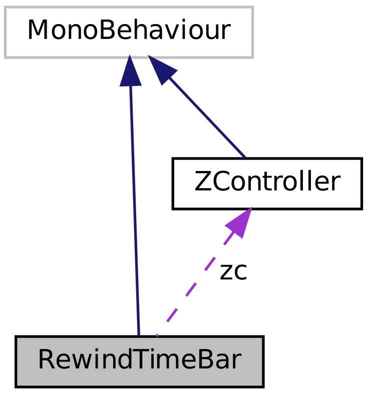
3.17 RewindTimeBar Class Reference

A simple script to synchronize the amount of time the player has left to use in the current ZController with an indication bar at the top of the screen.

Inheritance diagram for RewindTimeBar:



Collaboration diagram for RewindTimeBar:



Public Attributes

- ZController zc

  The ZController that the bar is working off of.

Private Member Functions

- void Start ()

  Attempt to set zc, backgroundBar, and progressBar from the scene if they have not been set already.
- void Update ()

  Sets the location of the top right corner of the progressBar based on the percentage of time remaining within the ZController for any given level. Hides the bars if ZController.timeAllowance is set to zero.

Private Attributes

- Image backgroundBar

  The backdrop image and mask of the bar.
- Image progressBar

  The actual progress bar itself.

3.17.1 Detailed Description

A simple script to synchronize the amount of time the player has left to use in the current ZController with an indication bar at the top of the screen.

3.17.2 Member Function Documentation

3.17.2.1 Start()

```
void RewindTimeBar.Start ( )  [inline], [private]
```

Attempt to set zc, backgroundBar, and progressBar from the scene if they have not been set already.

3.17.2.2 Update()

```
void RewindTimeBar.Update ( )  [inline], [private]
```

Sets the location of the top right corner of the progressBar based on the percentage of time remaining within the ZController for any given level. Hides the bars if ZController.timeAllowance is set to zero.

3.17.3 Member Data Documentation

3.17.3.1 backgroundBar

```
Image RewindTimeBar.backgroundBar  [private]
```

The backdrop image and mask of the bar.

3.17.3.2 progressBar

```
Image RewindTimeBar.progressBar  [private]
```

The actual progress bar itself.

3.17.3.3 zc

```
ZController RewindTimeBar.zc
```

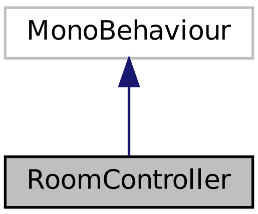
The ZController that the bar is working off of.

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

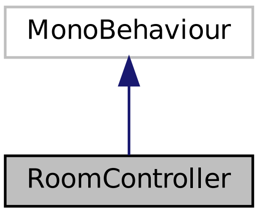
- Assets/Scripts/UI/RewindTimeBar.cs

3.18 RoomController Class Reference

Inheritance diagram for RoomController:



Collaboration diagram for RoomController:



Private Member Functions

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

Private Attributes

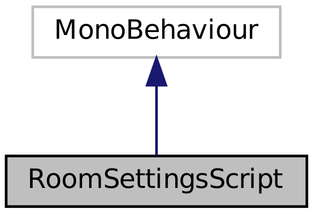
- IEnumerable< Animator > **animators**
- float **animTime**
- float **maxTime**

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

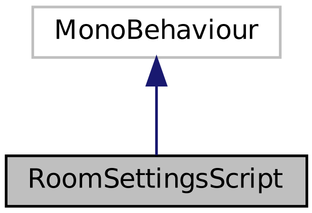
- Assets/Scripts/RoomController.cs

3.19 RoomSettingsScript Class Reference

Inheritance diagram for RoomSettingsScript:



Collaboration diagram for RoomSettingsScript:



Private Member Functions

- void **Start** ()

Private Attributes

- Color **fogColor**

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

- Assets/Scripts/RoomSettingsScript.cs

3.20 SnapState Class Reference

A representation of a gameObject's current state.

Public Attributes

- int **frameNumber**
- Vector3 **position**
- Quaternion **rotation**
- Vector3 **velocity**
- Vector3 **angularVelocity**
- bool **isOn**

3.20.1 Detailed Description

A representation of a gameObject's current state.

Currently contains

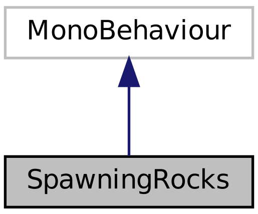
- Frame Number
- Position
- Rotation
- Velocity
- Angular Velocity
- is On?

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

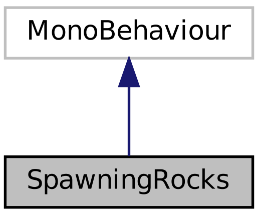
- Assets/Scripts/Rewinders/IRewinder.cs

3.21 SpawningRocks Class Reference

Inheritance diagram for SpawningRocks:



Collaboration diagram for SpawningRocks:



Public Attributes

- GameObject **rockPrefab**

Private Member Functions

- void **Start** ()
- IEnumerator **SpawnFalling** ()

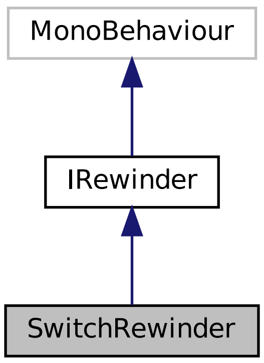
The documentation for this class was generated from the following file:

- Assets/Scripts/SpawningRocks.cs

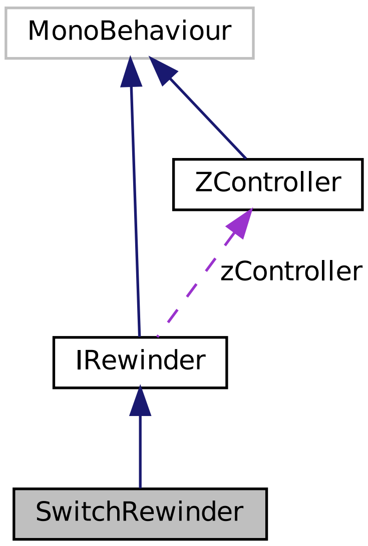
3.22 SwitchRewinder Class Reference

An implementation of IRewinder made for operating with binary switches.

Inheritance diagram for SwitchRewinder:



Collaboration diagram for SwitchRewinder:



Public Member Functions

- override void Store ()

Store the current state of the switch when recording.
- override SnapState RewindState (int count)

  Restore the state of the switch when rewinding.
- void OnTriggerEnter (Collider other)

  Check for input when player is near switch.
- void **OnTriggerExit** (Collider other)
- override void Play ()

  Continues time wherever it was left off, however the concrete implementation needs to accomplish this.
- override void Pause ()

  Pauses the movement of the object however the concrete implementation needs to accomplish this.

Public Attributes

- GameObject objToToggle

  The GameObject to toggle active when the switch is interacted with.
- bool isOn

  Boolean value. True if the switch is currently on (objToToggle inactive).
- bool **canToggle** = false
- Animator **animator**

Private Member Functions

- void Start ()

  Initializes the state of the switch.
- void **Update** ()

Additional Inherited Members

3.22.1 Detailed Description

An implementation of IRewinder made for operating with binary switches.

3.22.2 Member Function Documentation

3.22.2.1 OnTriggerEnter()

```
void SwitchRewinder.OnTriggerEnter (
             Collider other )  [inline]
```

Check for input when player is near switch.

3.22.2.2 Pause()

```
override void SwitchRewinder.Pause ( )  [inline], [virtual]
```

Pauses the movement of the object however the concrete implementation needs to accomplish this.

Implements IRewinder.

3.22.2.3 Play()

```
override void SwitchRewinder.Play ( )  [inline], [virtual]
```

Continues time wherever it was left off, however the concrete implementation needs to accomplish this.

Implements IRewinder.

3.22.2.4 RewindState()

```
override SnapState SwitchRewinder.RewindState (
             int count )  [inline], [virtual]
```

Restore the state of the switch when rewinding.

Reimplemented from IRewinder.

3.22.2.5 Store()

```
override void SwitchRewinder.Store ( )  [inline], [virtual]
```

Store the current state of the switch when recording.

Reimplemented from IRewinder.

3.22.3 Member Data Documentation

3.22.3.1 isOn

```
bool SwitchRewinder.isOn
```

Boolean value. True if the switch is currently on (objToToggle inactive).

3.22.3.2 objToToggle

```
GameObject SwitchRewinder.objToToggle
```

The GameObject to toggle active when the switch is interacted with.

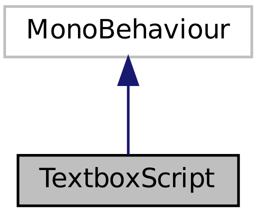
The documentation for this class was generated from the following file:

- Assets/Scripts/Rewinders/SwitchRewinder.cs

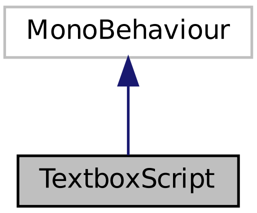
3.23 TextboxScript Class Reference

A controller for textboxes. Part of the textboxes prefab. Textbox appears on canvas layer when collision with player detected. Has support for multiple lines one after another, advanced with the Space key. Does not reappear if trigger is collided with again.

Inheritance diagram for TextboxScript:



Collaboration diagram for TextboxScript:



Public Attributes

- List< string > text

  A list of strings to display in the textbox. They are advanced through with the Space key.
- float timeToAdvance = 5

  Time in seconds before the textbox auto-advances
- bool hasBeenSeen = false

  A flag to keep track of if the textbox has been seen. Prevents the textbox from reoccuring if the collider is triggered again.

Private Member Functions

- void **Start** ()
- void OnTriggerEnter (Collider other)

  Function for when the player collides with the trigger.
- IEnumerator **DisplayTextCoroutine** ()

Private Attributes

- GameObject TextboxLayer

  The GameObject containing the Textbox Canvas conponent
- TextMeshProUGUI Textbox

  The Textbox in the Canvas UI Layer to write the text to

3.23.1 Detailed Description

A controller for textboxes. Part of the textboxes prefab. Textbox appears on canvas layer when collision with player detected. Has support for multiple lines one after another, advanced with the Space key. Does not reappear if trigger is collided with again.

3.23.2 Member Function Documentation

3.23.2.1 OnTriggerEnter()

```
void TextboxScript.OnTriggerEnter (
             Collider other )  [inline], [private]
```

Function for when the player collides with the trigger.

3.23.3 Member Data Documentation

3.23.3.1 hasBeenSeen

```
bool TextboxScript.hasBeenSeen = false
```

A flag to keep track of if the textbox has been seen. Prevents the textbox from reoccuring if the collider is triggered again.

3.23.3.2 text

```
List<string> TextboxScript.text
```

A list of strings to display in the textbox. They are advanced through with the Space key.

3.23.3.3 Textbox

```
TextMeshProUGUI TextboxScript.Textbox  [private]
```

The Textbox in the Canvas UI Layer to write the text to

3.23.3.4 TextboxLayer

```
GameObject TextboxScript.TextboxLayer  [private]
```

The GameObject containing the Textbox Canvas conponent

3.23.3.5 timeToAdvance

```
float TextboxScript.timeToAdvance = 5
```

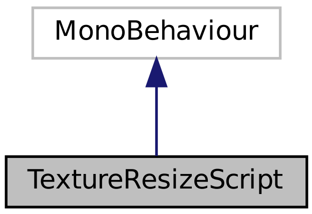
Time in seconds before the textbox auto-advances

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

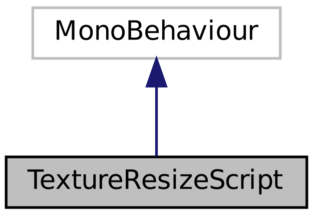
- Assets/Scripts/UI/TextboxScript.cs

3.24 TextureResizeScript Class Reference

Inheritance diagram for TextureResizeScript:



Collaboration diagram for TextureResizeScript:



Private Member Functions

- void **Start** ()

Private Attributes

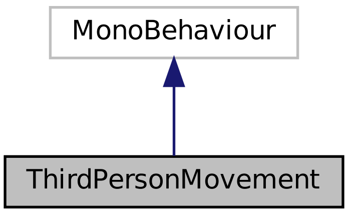
- float **resize** = 4
- float **scale** = 1
- Transform **t**
- Renderer **r**
- Material **m**

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

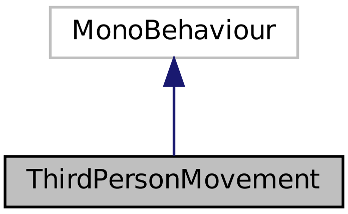
- Assets/Scripts/TextureResizeScript.cs

3.25 ThirdPersonMovement Class Reference

Inheritance diagram for ThirdPersonMovement:



Collaboration diagram for ThirdPersonMovement:



Public Attributes

- CharacterController **_controller**
- Transform **_cam**
- float **walkSpeed** = 5f
- float **turnSmoothnessTime** = 0.1f

Private Member Functions

- void **Update** ()

Private Attributes

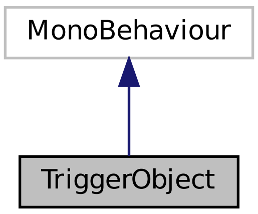
- float **turnSmoothSpeed**

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

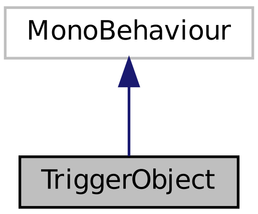
- Assets/Scripts/ThirdPersonMovement.cs

3.26 TriggerObject Class Reference

Inheritance diagram for TriggerObject:



Collaboration diagram for TriggerObject:



Public Attributes

- GameObject[ ] **targets**
- bool **Disable** = false

Private Member Functions

- void **OnTriggerEnter** (Collider other)

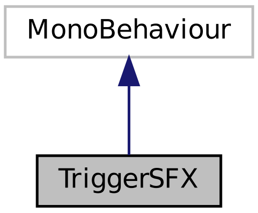
The documentation for this class was generated from the following file:

- Assets/Scripts/TriggerObject.cs

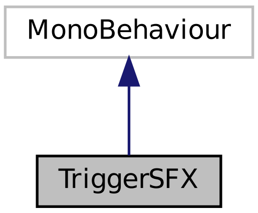
3.27 TriggerSFX Class Reference

A script for SFX sounds that are only heard when the player interacts with something, like walking over a pressure plate. Put this script on an object with a trigger that you you want to play a sound when the player walks into it. Add the sound's audio source in the script's inspector.

Inheritance diagram for TriggerSFX:



Collaboration diagram for TriggerSFX:



Public Attributes

- AudioSource **triggerSound**

Private Member Functions

- void **OnTriggerEnter** (Collider other)

3.27.1 Detailed Description

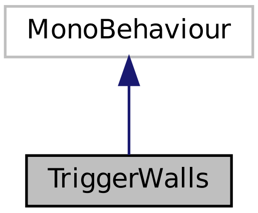
A script for SFX sounds that are only heard when the player interacts with something, like walking over a pressure plate. Put this script on an object with a trigger that you you want to play a sound when the player walks into it. Add the sound's audio source in the script's inspector.

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

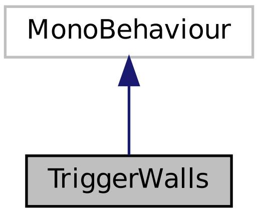
- Assets/Scripts/TriggerSFX.cs

3.28 TriggerWalls Class Reference

Inheritance diagram for TriggerWalls:



Collaboration diagram for TriggerWalls:



Public Attributes

- GameObject[ ] **targets**
- bool **Disable** = false

Private Member Functions

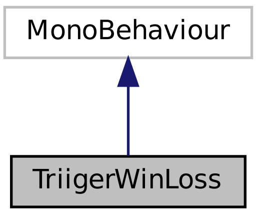
- void **OnTriggerEnter** (Collider other)

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

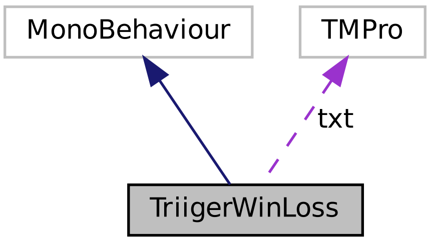
- Assets/Scripts/TriggerWalls.cs

3.29 TriigerWinLoss Class Reference

Inheritance diagram for TriigerWinLoss:



Collaboration diagram for TriigerWinLoss:



Public Attributes

- GameObject **WinLossScreen**
- bool **WinTrigger** = false
- bool **LoseTrigger** = false
- string **TriggerMessage** = "Unspecified Trigger"

Private Member Functions

- void **Start** ()
- void **OnTriggerEnter** (Collider other)

Private Attributes

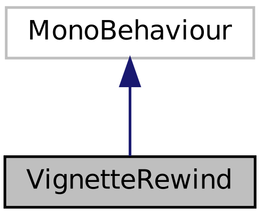
- TMPro.TextMeshProUGUI **txt**

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

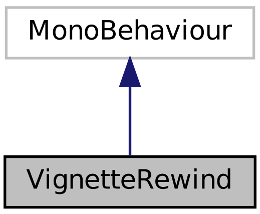
- Assets/Scripts/TriigerWinLoss.cs

3.30 VignetteRewind Class Reference

Inheritance diagram for VignetteRewind:



Collaboration diagram for VignetteRewind:



Private Member Functions

- void **Update** ()
- void **VignetteOn** ()
- void **VignetteOff** ()

Private Attributes

- Volume **volume**
- float **_intensity**

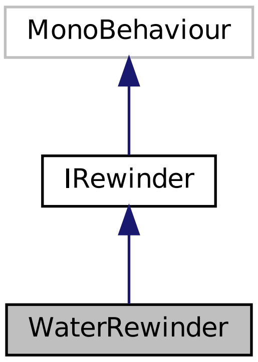
The documentation for this class was generated from the following file:

- Assets/Scripts/VignetteRewind.cs

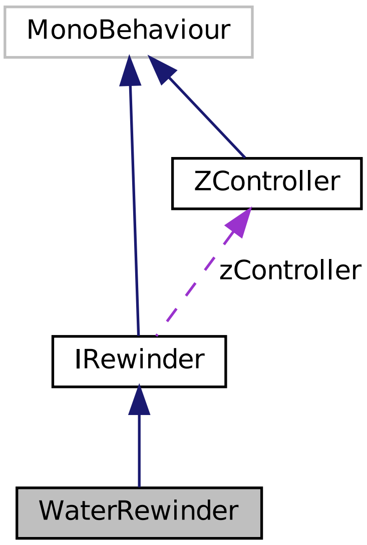
3.31 WaterRewinder Class Reference

An implementation of IRewinder made for operating with the water kill plane.

Inheritance diagram for WaterRewinder:



Collaboration diagram for WaterRewinder:



Public Member Functions

- override void Start ()

Initializes SnapState list, sets initial position and rotation, and extracts the ZController from parent if it was not explicitly set. Finally, pauses the object, awaiting the ZController's signal.

- void Update ()

  Updates the plane's y position by waterRiseSpeed / 100 units every frame
- override void Play ()

  Resume normal water rising
- override void Pause ()

  Sets the waterRiseLockout and prevents water from rising

Public Attributes

- float waterRiseSpeed = 10

  Speed of the rising water kill plane
- bool waterRiseLockout = false

  Lockout to prevent the water rising during rewind and pause

Additional Inherited Members

3.31.1 Detailed Description

An implementation of IRewinder made for operating with the water kill plane.

3.31.2 Member Function Documentation

3.31.2.1 Pause()

```
override void WaterRewinder.Pause ( )  [inline], [virtual]
```

Sets the waterRiseLockout and prevents water from rising

Implements IRewinder.

3.31.2.2 Play()

```
override void WaterRewinder.Play ( )  [inline], [virtual]
```

Resume normal water rising

Implements IRewinder.

3.31.2.3 Start()

```
override void WaterRewinder.Start ( )  [inline], [virtual]
```

Initializes SnapState list, sets initial position and rotation, and extracts the ZController from parent if it was not explicitly set. Finally, pauses the object, awaiting the ZController's signal.

Reimplemented from IRewinder.

3.31.2.4 Update()

```
void WaterRewinder.Update ( )  [inline]
```

Updates the plane's y position by waterRiseSpeed / 100 units every frame

3.31.3 Member Data Documentation

3.31.3.1 waterRiseLockout

```
bool WaterRewinder.waterRiseLockout = false
```

Lockout to prevent the water rising during rewind and pause

3.31.3.2 waterRiseSpeed

```
float WaterRewinder.waterRiseSpeed = 10
```

Speed of the rising water kill plane

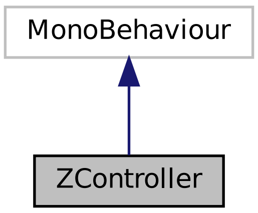
The documentation for this class was generated from the following file:

- Assets/Scripts/Rewinders/WaterRewinder.cs

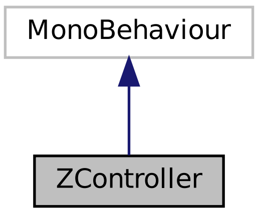
3.32 ZController Class Reference

A script to control all IRewinders in the current GameObject's children and process their state capture, motion, and rewind.

Inheritance diagram for ZController:



Collaboration diagram for ZController:



Public Member Functions

- void Pause ()

  Pauses all IRewinders under this controller's control.
- bool Approximate (Vector3 a, Vector3 b)

  Checks if two vectors are approximately equivalent with approximateLeniency leniency.
- void ReduceDeltaTime (float timeInc)

  Resets the reduces the delta time checking how long the players been holding the rewind by given value
- void ModifyTimeScale ()

  Increases or decreses the current time rewind scale. When rewind scale is less than one the value wil be fractional instead of entering negatives

Public Attributes

- float recordInterval = .2f

  The interval at which a record state should be taken in seconds.
- float rewindScale = 1f

  A measure of how quickly the rewinds should occur as a multiplier
- float rewindLimit = -1f

  How long in seconds a rewind can be held at a fully rewound state before "burning out" and needing to cooldown. -1 is disabled
- float rewindCooldown = 2f

  How long the player should be made to wait until they can use the rewind again after burning it out.
- float approximateLeniency = .01f

  A measure of how close to zero a vector needs to be for a child IRewinder to consider it to be zero; use primarily for IRewinder.NeedUpdate.
- float timeAllowance = 0f

  How much time in seconds the player has with this ZController before they can't use it anymore. 0 means disabled.
- float timeRemaining

  The internal tracking of how much time the player has with the ZController
- bool active

  A boolean to determine whether or not this controller is active.

Private Member Functions

- void Start ()

  Initialize the time and extract the children IRewinders. If the rewind scale is below zero, set it to 1 to prevent crashes.
- void Update ()

  Captures the input for rewinding and continues playing the scene if the keys released.
- void FixedUpdate ()

  Handle all rewind code.
  - *If the player is doing a rewind, then iterate through the rewind states for each IRewinder*
  - *If all states have been rewound and the rewind limit is reached, set the cooldown and play motion.*
  - *If there is an IRewinder that is still moving, all IRewinders capture their current states.*
- void OnTriggerEnter (Collider other)

  Allows for multiple controllers to exist within the same scene, setting activation to a bound set of triggers. Whenever the trigger attached to this GameObject is entered, all other controllers will be paused.

Private Attributes

- IEnumerable< IRewinder > rewinds

  A collection of all IRewinders within this controller's influence.
- float deltaTime

  The amount of time that has passed since an update; used to calculate the record interval and the rewind limit
- bool rewinding

  A variable to keep track of if the rewind buttons are being pressed;
- float cooldown = 0f

  Keeps track of how long the cooldown has left before players can use rewind again.

3.32.1 Detailed Description

A script to control all IRewinders in the current GameObject's children and process their state capture, motion, and rewind.

Because like, control + z, get it? Because it undoes?

3.32.2 Member Function Documentation

3.32.2.1 Approximate()

```
bool ZController.Approximate (
            Vector3 a,
            Vector3 b )  [inline]
```

Checks if two vectors are approximately equivalent with approximateLeniency leniency.

Parameters

a	First vector
b	Second vector

Returns

True if the difference between the vectors is less than approximateLeniency

3.32.2.2 FixedUpdate()

```
void ZController.FixedUpdate ( )  [inline], [private]
```

Handle all rewind code.

- If the player is doing a rewind, then iterate through the rewind states for each IRewinder
- If all states have been rewound and the rewind limit is reached, set the cooldown and play motion.
- If there is an IRewinder that is still moving, all IRewinders capture their current states.

3.32.2.3 ModifyTimeScale()

```
void ZController.ModifyTimeScale ( )  [inline]
```

Increases or decreses the current time rewind scale. When rewind scale is less than one the value wil be fractional instead of entering negatives

3.32.2.4 OnTriggerEnter()

```
void ZController.OnTriggerEnter (
            Collider other )  [inline], [private]
```

Allows for multiple controllers to exist within the same scene, setting activation to a bound set of triggers. Whenever the trigger attached to this GameObject is entered, all other controllers will be paused.

3.32.2.5 Pause()

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

Pauses all IRewinders under this controller's control.

3.32.2.6 ReduceDeltaTime()

```
void ZController.ReduceDeltaTime (
            float timeInc )  [inline]
```

Resets the reduces the delta time checking how long the players been holding the rewind by given value

3.32.2.7 Start()

```
void ZController.Start ( )  [inline], [private]
```

Initialize the time and extract the children IRewinders. If the rewind scale is below zero, set it to 1 to prevent crashes.

3.32.2.8 Update()

```
void ZController.Update ( )  [inline], [private]
```

Captures the input for rewinding and continues playing the scene if the keys released.

3.32.3 Member Data Documentation

3.32.3.1 active

```
bool ZController.active
```

A boolean to determine whether or not this controller is active.

The Unity version won't work because it disables the update methods when the object is inactive.

3.32.3.2 approximateLeniency

```
float ZController.approximateLeniency = .01f
```

A measure of how close to zero a vector needs to be for a child IRewinder to consider it to be zero; use primarily for IRewinder.NeedUpdate.

3.32.3.3 cooldown

```
float ZController.cooldown = 0f  [private]
```

Keeps track of how long the cooldown has left before players can use rewind again.

3.32.3.4 deltaTime

```
float ZController.deltaTime  [private]
```

The amount of time that has passed since an update; used to calculate the record interval and the rewind limit

3.32.3.5 recordInterval

```
float ZController.recordInterval = .2f
```

The interval at which a record state should be taken in seconds.

3.32.3.6 rewindCooldown

```
float ZController.rewindCooldown = 2f
```

How long the player should be made to wait until they can use the rewind again after burning it out.

3.32.3.7 rewinding

```
bool ZController.rewinding [private]
```

A variable to keep track of if the rewind buttons are being pressed;

3.32.3.8 rewindLimit

```
float ZController.rewindLimit = -1f
```

How long in seconds a rewind can be held at a fully rewound state before "burning out" and needing to cooldown. -1 is disabled

3.32.3.9 rewinds

```
IEnumerable<IRewinder> ZController.rewinds [private]
```

A collection of all IRewinders within this controller's influence.

3.32.3.10 rewindScale

```
float ZController.rewindScale = 1f
```

A measure of how quickly the rewinds should occur as a multiplier

3.32.3.11 timeAllowance

```
float ZController.timeAllowance = 0f
```

How much time in seconds the player has with this ZController before they can't use it anymore. 0 means disabled.

3.32.3.12 timeRemaining

```
float ZController.timeRemaining
```

The internal tracking of how much time the player has with the ZController

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

- Assets/Scripts/ZController.cs

Index