

Celestial Rewind

Generated by Doxygen 1.8.17

1 Hierarchical Index	1
1.1 Class Hierarchy	1
2 Class Index	3
2.1 Class List	3
3 Class Documentation	5
3.1 AudioController Class Reference	5
3.1.1 Detailed Description	5
3.1.2 Member Function Documentation	5
3.1.2.1 WaitForNextStep()	6
3.2 DebugPanel Class Reference	6
3.2.1 Detailed Description	7
3.2.2 Member Data Documentation	7
3.2.2.1 Drag	7
3.2.2.2 Gravity	7
3.2.2.3 Leniency	7
3.2.2.4 Move	7
3.2.2.5 panel	8
3.2.2.6 ppc	8
3.2.2.7 RCooldown	8
3.2.2.8 RewindSpeed	8
3.2.2.9 RLimit	8
3.2.2.10 Speed	8
3.2.2.11 zc	9
3.3 FallingRocksObstacle Class Reference	9
3.4 FirstPersonLook Class Reference	9
3.5 IRewinder Class Reference	9
3.5.1 Detailed Description	10
3.5.2 Member Function Documentation	11
3.5.2.1 HasStates()	11
3.5.2.2 NeedUpdate()	11
3.5.2.3 Pause()	11
3.5.2.4 Play()	11
3.5.2.5 Reset()	12
3.5.2.6 RewindState()	12
3.5.2.7 Start()	12
3.5.2.8 Store()	12
3.5.3 Member Data Documentation	12
3.5.3.1 printDebug	13
3.5.3.2 states	13
3.5.3.3 zController	13
3.5.4 Property Documentation	13

3.5.4.1 startPos	13
3.5.4.2 startRot	13
3.6 LevelLoader Class Reference	14
3.7 LevelTransitionTrigger Class Reference	14
3.8 MenuScript Class Reference	14
3.9 MovementScript Class Reference	15
3.10 PhysicsPlayerController Class Reference	15
3.10.1 Detailed Description	16
3.10.2 Member Function Documentation	16
3.10.2.1 DoJump()	16
3.10.2.2 DoMovement()	16
3.10.2.3 OnCollisionEnter()	17
3.10.2.4 UpdateCamera()	17
3.10.3 Member Data Documentation	17
3.10.3.1 cam	17
3.10.3.2 grounded	17
3.10.3.3 maxSpeed	18
3.10.3.4 mouseXSensitivity	18
3.10.3.5 mouseYSensitivity	18
3.10.3.6 playerAcceleration	18
3.11 PhysicsRewinder Class Reference	18
3.11.1 Detailed Description	19
3.11.2 Member Function Documentation	19
3.11.2.1 OnCollisionStay()	19
3.11.2.2 Pause()	20
3.11.2.3 Play()	20
3.11.2.4 RewindState()	20
3.11.2.5 Start()	21
3.11.2.6 Store()	21
3.11.2.7 Update()	21
3.11.3 Member Data Documentation	21
3.11.3.1 rb	21
3.11.3.2 startForce	21
3.12 PlayerController Class Reference	22
3.12.1 Detailed Description	22
3.13 RoomController Class Reference	22
3.14 SnapState Class Reference	22
3.14.1 Detailed Description	23
3.15 SpawningRocks Class Reference	23
3.16 TextureResizeScript Class Reference	23
3.17 ThirdPersonMovement Class Reference	24
3.18 TriggerWalls Class Reference	24

3.19 TriigerWinLoss Class Reference	24
3.20 ZController Class Reference	24
3.20.1 Detailed Description	25
3.20.2 Member Function Documentation	25
3.20.2.1 Approximate()	25
3.20.2.2 Pause()	26
3.20.3 Member Data Documentation	26
3.20.3.1 active	26
3.20.3.2 approximateLeniency	26
3.20.3.3 recordInterval	26
3.20.3.4 rewindCooldown	26
3.20.3.5 rewindLimit	27
3.20.3.6 rewindScale	27
Index	29

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

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

MonoBehaviour	
AudioController	5
DebugPanel	6
FallingRocksObstacle	9
FirstPersonLook	9
IRewinder	9
PhysicsRewinder	18
LevelLoader	14
LevelTransitionTrigger	14
MenuScript	14
MovementScript	15
PhysicsPlayerController	15
PlayerController	22
RoomController	22
SpawningRocks	23
TextureResizeScript	23
ThirdPersonMovement	24
TriggerWalls	24
TriigerWinLoss	24
ZController	24
SnapState	22

Chapter 2

Class Index

2.1 Class List

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

AudioController	
A controller for player audio	5
DebugPanel	
A controller for the debug settings. Currently contains a manually maintained map of UI elements to the corresponding object value	6
FallingRocksObstacle	9
FirstPersonLook	9
IRewinder	
An abstraction of the rewinder functionality for use with a ZController to rewind an object	9
LevelLoader	14
LevelTransitionTrigger	14
MenuScript	14
MovementScript	15
PhysicsPlayerController	
A Rigidbody based player controller	15
PhysicsRewinder	
An implementation of IRewinder made for operating with Rigidbody physics	18
PlayerController	
Modified from unity docs: https://docs.unity3d.com/ScriptReference/CharacterController.Move.html To be removed in later versions	22
RoomController	22
SnapState	
A representation of a gameObject's current state	22
SpawningRocks	23
TextureResizeScript	23
ThirdPersonMovement	24
TriggerWalls	24
TriigerWinLoss	24
ZController	
A script to control all IRewinders in the current GameObject's children and process their state capture, motion, and rewind	24

Chapter 3

Class Documentation

3.1 AudioController Class Reference

A controller for player audio.

Public Member Functions

- IEnumerator [WaitForNextStep](#) ()
A coroutine method that waits until the next step audio should be played

Public Attributes

- AudioSource **audioSource**
- AudioClip[] **walkSounds**
- AudioSource **timeAudio**
- AudioClip **timeRewindSound**
- float **waitStep** = .5f
- [PhysicsPlayerController](#) **ppc**

3.1.1 Detailed Description

A controller for player audio.

3.1.2 Member Function Documentation

3.1.2.1 WaitForNextStep()

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

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

Returns

A WaitForSeconds object.

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

- Assets/Scripts/AudioController.cs

3.2 DebugPanel Class Reference

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

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.

3.2.1 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.

3.2.2 Member Data Documentation

3.2.2.1 Drag

```
TMP_InputField DebugPanel.Drag
```

A text field representing drag

Possibly could be removed now

3.2.2.2 Gravity

```
TMP_InputField DebugPanel.Gravity
```

A text field representing gravity's value

3.2.2.3 Leniency

```
TMP_InputField DebugPanel.Leniency
```

The leniency for the [ZController](#)'s check for no motion.

3.2.2.4 Move

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

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

3.2.2.5 panel

`GameObject DebugPanel.panel`

The parent container for all the text fields.

3.2.2.6 ppc

`PhysicsPlayerController DebugPanel.ppc`

The `PhysicsPlayerController` the fields are referencing.

3.2.2.7 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.

3.2.2.8 RewindSpeed

`TMP_InputField DebugPanel.RewindSpeed`

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

3.2.2.9 RLimit

`TMP_InputField DebugPanel.RLimit`

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

3.2.2.10 Speed

`List<TMP_InputField> DebugPanel.Speed`

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

3.2.2.11 **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`

3.3 FallingRocksObstacle Class Reference

Public Attributes

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

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

- `Assets/Scripts/FallingRocksObstacle.cs`

3.4 FirstPersonLook Class Reference

Public Attributes

- float **turnSpeed** = 1
- float **mouseXSensitivity** = 1
- float **mouseYSensitivity** = 1
- Transform **cam**
- Transform **orientation**
- Transform **player**
- Transform **playerObj**

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

- `Assets/Scripts/TestMovement/FirstPersonLook.cs`

3.5 IRewinder Class Reference

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

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 on the rewind stack.
- virtual [SnapState RewindState](#) ()
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.
- 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

3.5.1 Detailed Description

An abstraction of the rewinder functionality for use with a [ZController](#) to rewind an object.

3.5.2 Member Function Documentation

3.5.2.1 HasStates()

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

Checks whether the [SnapState](#) stack contains elements.

Returns

3.5.2.2 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

3.5.2.3 Pause()

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

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

Implemented in [PhysicsRewinder](#).

3.5.2.4 Play()

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

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

Implemented in [PhysicsRewinder](#).

3.5.2.5 Reset()

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

Resets the object to its initial position and rotation and clears the [SnapState](#) stack.

3.5.2.6 RewindState()

```
virtual SnapState IRewinder.RewindState ( ) [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

Returns

The popped [SnapState](#) or null if there are no states remaining.

Reimplemented in [PhysicsRewinder](#).

3.5.2.7 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](#).

3.5.2.8 Store()

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

Stores the current position and rotation of the object on the rewind stack.

Reimplemented in [PhysicsRewinder](#).

3.5.3 Member Data Documentation

3.5.3.1 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.

3.5.3.2 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.

3.5.3.3 zController

```
ZController IRewinder.zController
```

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

3.5.4 Property Documentation

3.5.4.1 startPos

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

Initial position state of the attachee GameObject

3.5.4.2 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

3.6 LevelLoader Class Reference

Public Member Functions

- IEnumerator **LoadLevel** (int sceneIndex)
- void **ChangeScene** (int sceneIndex)
- void **NextScene** ()
- void **ReloadScene** ()

Public Attributes

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

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

- Assets/Scripts/Transitions/LevelLoader.cs

3.7 LevelTransitionTrigger Class Reference

Public Member Functions

- void **OnTriggerEnter** (Collider other)

Public Attributes

- [LevelLoader](#) **ILoad**
- bool **TargetScene**
- int **SelectedScene** = 0

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

- Assets/Scripts/Transitions/LevelTransitionTrigger.cs

3.8 MenuScript Class Reference

Public Attributes

- GameObject **panel**

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

- Assets/Scripts/UI/MenuScript.cs

3.9 MovementScript Class Reference

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**

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

- Assets/Scripts/TestMovement/MovementScript.cs

3.10 PhysicsPlayerController Class Reference

A Rigidbody based player controller.

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 converted into a Rigidbody force using the mass and the [playerAcceleration](#) for that direction. 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, 10, 20)
A Vector3 containing the acceleration value for the player in each direction.
- Vector3 [maxSpeed](#) = new Vector3(3, 15, 5)
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.
- Transform [cam](#)
The Transform for the player's main Camera.
- float [mouseXSensitivity](#) = 1
The mouse sensitivity for the camera going up and down.
- float [mouseYSensitivity](#) = 1
The mouse sensitivity for the camera going side to side.

3.10.1 Detailed Description

A Rigidbody based player controller.

3.10.2 Member Function Documentation

3.10.2.1 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

<i>yVel</i>	A reference to the z velocity of the player; will be capped to the maxSpeed value if too high
-------------	---

3.10.2.2 DoMovement()

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

Calculates the player's x and z direction movement. Movement is converted into a Rigidbody force using the mass and the [playerAcceleration](#) for that direction. If the current velocity is higher than the max, however, it will be reduced to the max and stored in *xVel* and *zVel*.

Parameters

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

3.10.2.3 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.

3.10.2.4 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.

3.10.3 Member Data Documentation

3.10.3.1 cam

```
Transform PhysicsPlayerController.cam
```

The Transform for the player's main Camera.

3.10.3.2 grounded

```
bool PhysicsPlayerController.grounded
```

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

3.10.3.3 maxSpeed

```
Vector3 PhysicsPlayerController.maxSpeed = new Vector3(3, 15, 5)
```

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

3.10.3.4 mouseXSensitiviy

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

The mouse sensitivity for the camera going up and down.

3.10.3.5 mouseYSensitiviy

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

The mouse sensitivity for the camera going side to side.

3.10.3.6 playerAcceleration

```
Vector3 PhysicsPlayerController.playerAcceleration = new Vector3(10, 10, 20)
```

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

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

- Assets/Scripts/PhysicsPlayerController.cs

3.11 PhysicsRewinder Class Reference

An implementation of [IRewinder](#) made for operating with Rigidbody physics.

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 on the rewind stack.
 Also stores the Rigidbody's current velocity and current angular velocity.*
- override [SnapState](#) [RewindState](#) ()
*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.

Additional Inherited Members

3.11.1 Detailed Description

An implementation of [IRewinder](#) made for operating with Rigidbody physics.

3.11.2 Member Function Documentation

3.11.2.1 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

<i>collision</i>	A Collision containing data about the colliding body.
------------------	---

3.11.2.2 Pause()

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

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

Implements [IRewinder](#).

3.11.2.3 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](#).

3.11.2.4 RewindState()

```
override SnapState PhysicsRewinder.RewindState ( ) [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.

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](#).

3.11.2.5 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](#).

3.11.2.6 Store()

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

Stores the current position and rotation of the object on the rewind stack.

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

Reimplemented from [IRewinder](#).

3.11.2.7 Update()

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

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

3.11.3 Member Data Documentation

3.11.3.1 rb

Rigidbody PhysicsRewinder.rb

The attachee GameObject's attached Rigidbody.

3.11.3.2 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

3.12 PlayerController Class Reference

Modified from unity docs: <https://docs.unity3d.com/ScriptReference/CharacterController.Move.html> To be removed in later versions.

Public Member Functions

- IEnumerator **WaitForNextStep** ()

Public Attributes

- CharacterController **controller**
- float **playerSpeed** = 5.0f
- float **jumpHeight** = 2.0f
- float **MAX_VELOCITY_TOLERANCE** = 100
- Transform **cam**
- float **turnSmoothnessTime** = 0.1f
- float **mouseXSensitivity** = 1
- float **mouseYSensitivity** = 1
- AudioSource **audioSource**
- AudioClip[] **walkSounds**
- AudioSource **timeAudio**
- AudioClip **timeRewindSound**
- float **waitStep** = .5f

3.12.1 Detailed Description

Modified from unity docs: <https://docs.unity3d.com/ScriptReference/CharacterController.Move.html> To be removed in later versions.

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

- Assets/Scripts/PlayerController.cs

3.13 RoomController Class Reference

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

- Assets/Scripts/RoomController.cs

3.14 SnapState Class Reference

A representation of a gameObject's current state.

Public Attributes

- Vector3 **position**
- Quaternion **rotation**
- Vector3 **velocity**
- Vector3 **angularVelocity**

3.14.1 Detailed Description

A representation of a gameObject's current state.

Currently contains

- Position
- Rotation
- Velocity
- Angular Velocity

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

- Assets/Scripts/Rewinders/IRewinder.cs

3.15 SpawningRocks Class Reference

Public Attributes

- GameObject **rockPrefab**

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

- Assets/Scripts/SpawningRocks.cs

3.16 TextureResizeScript Class Reference

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

- Assets/Scripts/TextureResizeScript.cs

3.17 ThirdPersonMovement Class Reference

Public Attributes

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

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

- Assets/Scripts/ThirdPersonMovement.cs

3.18 TriggerWalls Class Reference

Public Attributes

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

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

- Assets/Scripts/TriggerWalls.cs

3.19 TriigerWinLoss Class Reference

Public Attributes

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

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

- Assets/Scripts/TriigerWinLoss.cs

3.20 ZController Class Reference

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

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.

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](#).
- bool [active](#)
A boolean to determine whether or not this controller is active.

3.20.1 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?

3.20.2 Member Function Documentation

3.20.2.1 Approximate()

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

Checks if two vectors are approximately equivalent with [approximateLeniency](#) leniency.

Parameters

<i>a</i>	First vector
<i>b</i>	Second vector

Returns

True if the difference between the vectors is less than [approximateLeniency](#)

3.20.2.2 Pause()

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

Pauses all [IRewinders](#) under this controller's control.

3.20.3 Member Data Documentation

3.20.3.1 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.

3.20.3.2 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](#).

3.20.3.3 recordInterval

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

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

3.20.3.4 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.

3.20.3.5 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

3.20.3.6 rewindScale

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

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

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

- Assets/Scripts/ZController.cs

Index

- active
 - ZController, [26](#)
- Approximate
 - ZController, [25](#)
- approximateLeniency
 - ZController, [26](#)
- AudioController, [5](#)
 - WaitForNextStep, [5](#)
- cam
 - PhysicsPlayerController, [17](#)
- DebugPanel, [6](#)
 - Drag, [7](#)
 - Gravity, [7](#)
 - Leniency, [7](#)
 - Move, [7](#)
 - panel, [7](#)
 - ppc, [8](#)
 - RCooldown, [8](#)
 - RewindSpeed, [8](#)
 - RLimit, [8](#)
 - Speed, [8](#)
 - zc, [8](#)
- DoJump
 - PhysicsPlayerController, [16](#)
- DoMovement
 - PhysicsPlayerController, [16](#)
- Drag
 - DebugPanel, [7](#)
- FallingRocksObstacle, [9](#)
- FirstPersonLook, [9](#)
- Gravity
 - DebugPanel, [7](#)
- grounded
 - PhysicsPlayerController, [17](#)
- HasStates
 - IRewinder, [11](#)
- IRewinder, [9](#)
 - HasStates, [11](#)
 - NeedUpdate, [11](#)
 - Pause, [11](#)
 - Play, [11](#)
 - printDebug, [12](#)
 - Reset, [11](#)
 - RewindState, [12](#)
 - Start, [12](#)
 - startPos, [13](#)
 - startRot, [13](#)
 - states, [13](#)
 - Store, [12](#)
 - zController, [13](#)
- Leniency
 - DebugPanel, [7](#)
- LevelLoader, [14](#)
- LevelTransitionTrigger, [14](#)
- maxSpeed
 - PhysicsPlayerController, [17](#)
- MenuScript, [14](#)
- mouseXSensitiviy
 - PhysicsPlayerController, [18](#)
- mouseYSensitiviy
 - PhysicsPlayerController, [18](#)
- Move
 - DebugPanel, [7](#)
- MovementScript, [15](#)
- NeedUpdate
 - IRewinder, [11](#)
- OnCollisionEnter
 - PhysicsPlayerController, [17](#)
- OnCollisionStay
 - PhysicsRewinder, [19](#)
- panel
 - DebugPanel, [7](#)
- Pause
 - IRewinder, [11](#)
 - PhysicsRewinder, [20](#)
 - ZController, [26](#)
- PhysicsPlayerController, [15](#)
 - cam, [17](#)
 - DoJump, [16](#)
 - DoMovement, [16](#)
 - grounded, [17](#)
 - maxSpeed, [17](#)
 - mouseXSensitiviy, [18](#)
 - mouseYSensitiviy, [18](#)
 - OnCollisionEnter, [17](#)
 - playerAcceleration, [18](#)
 - UpdateCamera, [17](#)
- PhysicsRewinder, [18](#)
 - OnCollisionStay, [19](#)
 - Pause, [20](#)
 - Play, [20](#)

- rb, [21](#)
 - RewindState, [20](#)
 - Start, [20](#)
 - startForce, [21](#)
 - Store, [21](#)
 - Update, [21](#)
- Play
 - IRewinder, [11](#)
 - PhysicsRewinder, [20](#)
- playerAcceleration
 - PhysicsPlayerController, [18](#)
- PlayerController, [22](#)
- ppc
 - DebugPanel, [8](#)
- printDebug
 - IRewinder, [12](#)
- rb
 - PhysicsRewinder, [21](#)
- RCooldown
 - DebugPanel, [8](#)
- recordInterval
 - ZController, [26](#)
- Reset
 - IRewinder, [11](#)
- rewindCooldown
 - ZController, [26](#)
- rewindLimit
 - ZController, [26](#)
- rewindScale
 - ZController, [27](#)
- RewindSpeed
 - DebugPanel, [8](#)
- RewindState
 - IRewinder, [12](#)
 - PhysicsRewinder, [20](#)
- RLimit
 - DebugPanel, [8](#)
- RoomController, [22](#)
- SnapState, [22](#)
- SpawningRocks, [23](#)
- Speed
 - DebugPanel, [8](#)
- Start
 - IRewinder, [12](#)
 - PhysicsRewinder, [20](#)
- startForce
 - PhysicsRewinder, [21](#)
- startPos
 - IRewinder, [13](#)
- startRot
 - IRewinder, [13](#)
- states
 - IRewinder, [13](#)
- Store
 - IRewinder, [12](#)
 - PhysicsRewinder, [21](#)
- TextureResizeScript, [23](#)
- ThirdPersonMovement, [24](#)
- TriggerWalls, [24](#)
- TriigerWinLoss, [24](#)
- Update
 - PhysicsRewinder, [21](#)
- UpdateCamera
 - PhysicsPlayerController, [17](#)
- WaitForNextStep
 - AudioController, [5](#)
- zc
 - DebugPanel, [8](#)
- ZController, [24](#)
 - active, [26](#)
 - Approximate, [25](#)
 - approximateLeniency, [26](#)
 - Pause, [26](#)
 - recordInterval, [26](#)
 - rewindCooldown, [26](#)
 - rewindLimit, [26](#)
 - rewindScale, [27](#)
- zController
 - IRewinder, [13](#)