User Guide

Ultimate Replay 3.0

A simple and effective state-based replay system for Unity

Trivial Interactive

Version 3.0.x

Ultimate Replay 3.0 is a complete state-based replay system ideally suited to kill cams or action replay applications. Due to the state-based nature of the system, it is possible to view replays from any angle or even fly around the scene as playback occurs.

Recommended Uses:

- Action replays in sports of similar games (Demo included).
- Kill cams / Death cams in shooting games (Demo included).
- Ghost vehicles in racing games (Demo included).
- Show reel / highlights montage.
- Multiple angle replays where the same recording is viewed from several different vantage points in succession.
- Many more uses…

Features

- Quick and easy setup / integration into existing projects.
- Simple API for replay and playback control means that very little scripting knowledge is required.
- Record and replay as many different objects as you want simultaneously.
- Uses a state-based replay system meaning that you can view playback from different camera angles or even fly around as playback occurs.
- Full support for instantiation or destruction of objects during recording.
- Fully interpolated playback means that you can record at ultra-low frame rates (5fps and less) and retain smooth and accurate replays.
- Supports playback at any speed from ultra-slow motion to 2 or 4x.
- Supports reverse playback which can be used to produce a rewind effect.
- Playback can be paused and resumed at a later date.
- Full playback seek support allows you to jump to any point in a recording.

- Full control over recording frame rate to make sure you capture the best quality recording at the smallest memory cost.
- Recording an object is as simple as attaching a replay component.
- Built-in support for recording transform, audio, particles, animation, and more.
- Easily create your own component recorders to expand the capabilities.
- Memory recording can be setup as continuous or as a rolling buffer configuration.
- File support means that you can create persistent replays for later game sessions.
- Highly optimized file format allows for lengthy replays with minimal file size and high-performance streaming.
- ReplayVars allow script variables to be recorded simply by adding an attribute.
- Get useful hints about the storage space requirements for all replay objects.
- Includes example GUI controls for playback manipulation.
- Comprehensive .chm documentation of the API for quick and easy reference.
- Fully commented C# source code included.

Getting started

To get up and running as quick as possible check out the Quick Start guide which will cover the basic setup required.

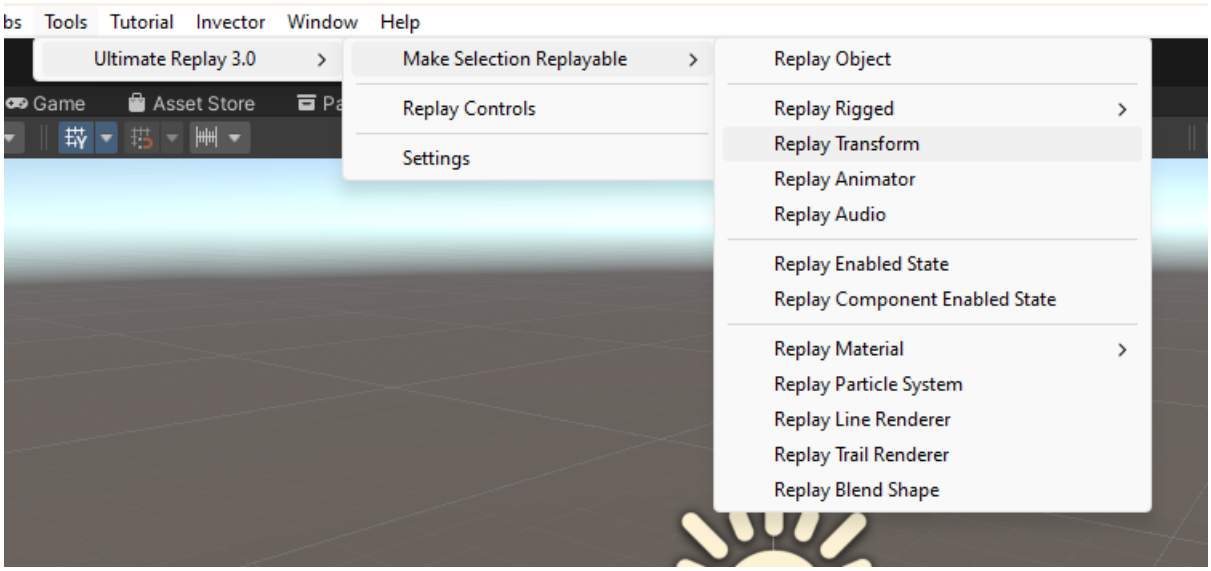
Contents

Upgrade Guide

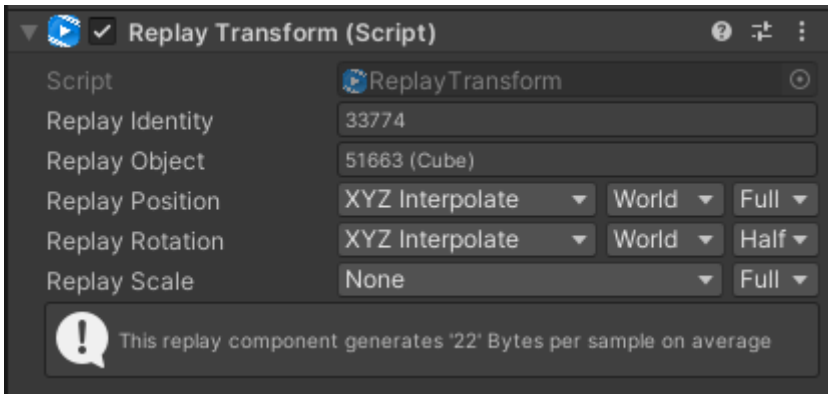
This section will cover some of the major changes between version 2.0 and 3.0 and what you will need to do in order to upgrade an existing project from 2.0 to 3.0.

Setup Replay Objects

Setting up replay objects in version 3.0 is almost identical to version 2.0, where most users will use the tools menu to quickly setup game objects in the scene or prefabs. The only minor changes you may notice is that some components may be named slightly differently, and the menu may also be organised in a different way.



You will see that the components are added to the game object in much the same way as version 2.0, and no further setup is required for non-prefab objects. Note that replay component inspectors may appear different.

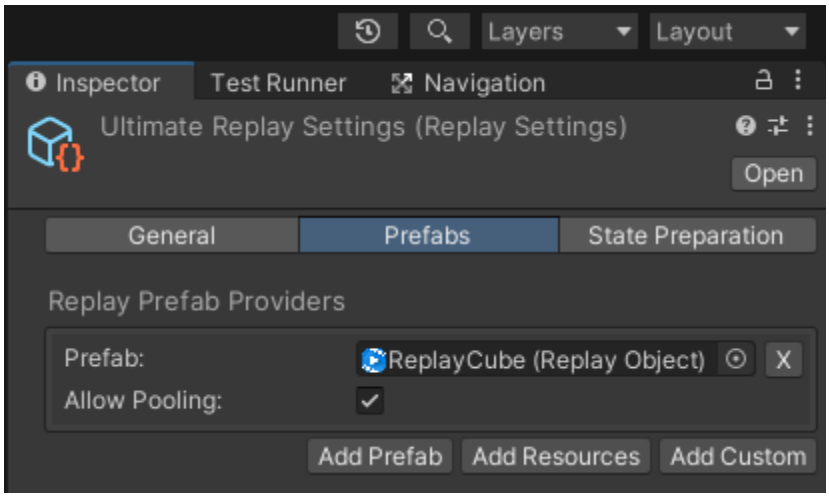


Setup Replay Prefabs

Replay prefabs are slightly different in version 3.0 because the replay system now supports Object lifecycle Providers which are special scripts that manage the creation and destruction of a particular replay object instance during playback.

Open the replay settings window as done in previous version via the menu Tools -> Ultimate Replay 3.0 -> Settings and then select the Prefabs tab. You will see a list view with the options to AddPrefab, AddResources and AddCustom lifecycle providers. The button you use will depend upon where the prefab is located, but for most users AddPrefab will be suitable.

If you have a replay object located inside the Resources folder, then you would likely want to use AddResources option which can lazy load the asset on demand. AddCustom is more advanced and will be covered in a dedicate topic.



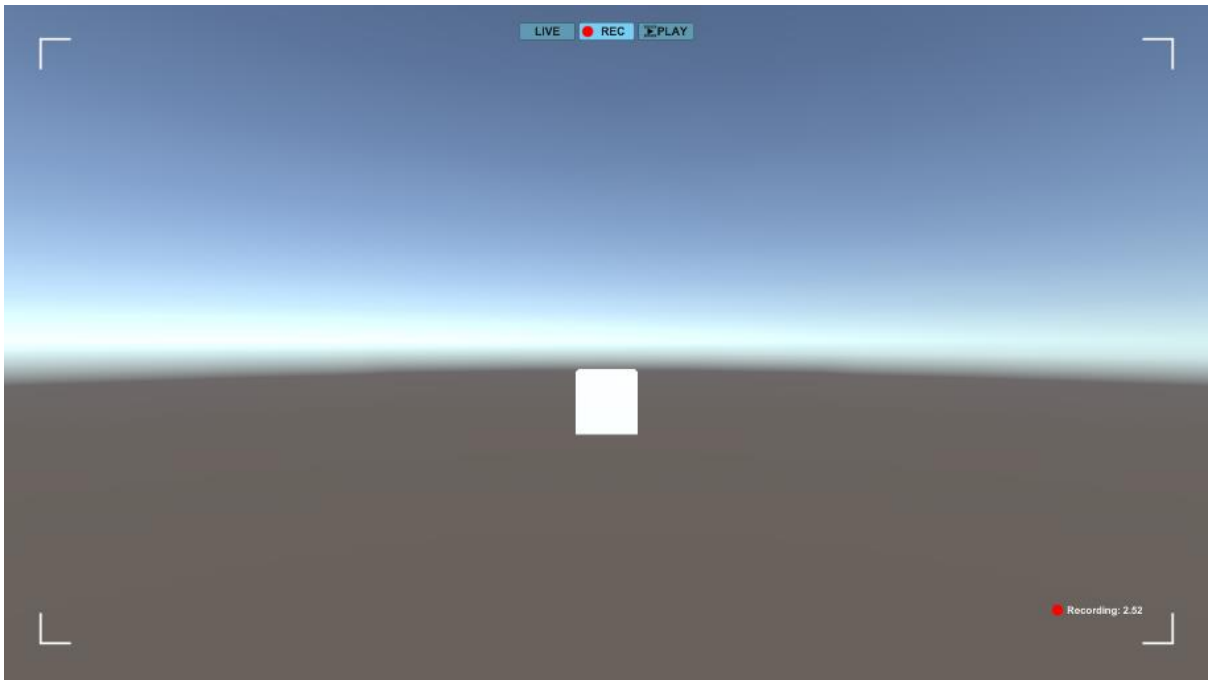
You will see that there is now a slot to drop your Replay Object into, as well as the option to enable or disable pooling.

Ultimate Replay 3.0 now has built in support for pooling using UnityEngine.Pool namespace and is enabled by default for optimal performance. Custom pooling implementations can be used by creating a custom lifecycle provider, to be covered in a future topic.

Replay Controls

The replay controls UI that is useful for quick testing or as a base to expand upon is much the same in terms of functionality between versions. The main difference is that for version 3.0, the UI is now implemented with UGUI rather than the legacy Gui. There is no other real difference to mention apart from the fact that the new ReplayControls prefab will require an event system to be in the scene in order to function correctly, just like any other UGUI components. The ReplayControls script will give out a useful warning though if there is no event system present, so you aren't left wondering why the buttons are not responding.

As in the previous version, you can add the default ReplayControls UI to your current scene using the menu Tools -> Ultimate Replay 3.0 -> Replay Controls



Replay Manager API

The other main change for version 3.0 is the ReplayManager API which is the main part of the asset that you will use when scripting your game. The previous version of Ultimate Replay used the ReplayHandle concept to represent a specific record or replay operation. Version 3.0 has now been redesigned to be more object oriented and user friendly, so replay handles are no longer a thing. Instead, when you start a replay operation using either ReplayManager.BeginRecording or ReplayManager.BeginPlayback, you will now get either a ReplayRecordOperation or ReplayPlaybackOperation object back in return. From that object you can access all information associated with the replay operation. here are some quick examples:

Start recording example

A quick example to show the differences between versions when starting a new record operation.

Ultimate Replay 2.0

C# Code

```
1  // Create some storage
2  ReplayStorageTarget storage = new ReplayMemoryTarget();
3
4  // Start recording
5  ReplayHandle recordHandle = ReplayManager.BeginRecording(storage);
```

Ultimate Replay 3.0

C# Code

```
1  // Create some storage
2  ReplayStorage storage = new ReplayMemoryStorage("MyReplay");
3
4  // Start recording
5  ReplayRecordOperation recordOp =
6  ReplayManager.BeginRecording(storage);
```

Seek playback example

A quick example to show the differences between versions when starting a new playback operation and then seeking to a specific time offset.

Ultimate Replay 2.0

C# Code

```
1  ReplayStorageTarget storage = ...
2
3  // Start playback
4  ReplayHandle playbackHandle = ReplayManager.BeginPlayback(storage);
5
6  // Seek to specified time (5 seconds)
7  ReplayManager.SetPlaybackTime(playbackHandle, 5f);
```

Ultimate Replay 3.0

C# Code

```
ReplayStorage storage = ...

// Start playback
ReplayPlaybackOperation playbackOp =
ReplayManager.BeginPlayback(storage);

// Seek to specified time (5 seconds)
playbackOp.SeekPlayback(5f);
```

You can see from the above examples that version 3.0 no longer uses ReplayHandle's but instead uses replay operations which we think is more intuitive and easier for beginners to understand. Similarly to the above, all other record and replay operations that you would expect such as pause, resume, playback time, playback timescale, and more can be accessed via the ReplayRecordOperation and ReplayPlaybackOperation objects respectively.

Quick Start

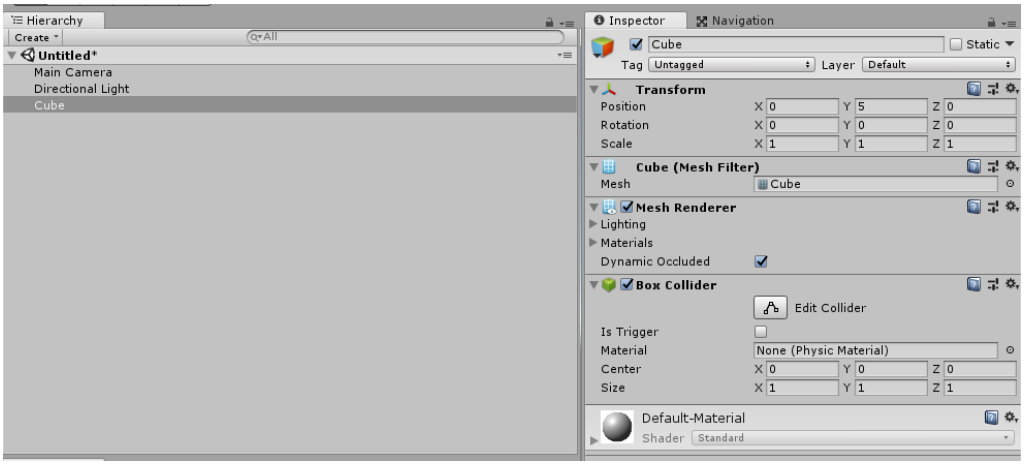
This section will walk you through the bare minimum setup required to get a simple replay scene up and running as quick as possible. As this is a quick start guide, many of the replay system concepts will not be covered here, so it is recommended that you check out the following sections for more information: Replay Considerations, Replay Concepts

> **Note:** *The completed demo scene which will be created by following the guide below is included in the asset for convenience. You can find this scene at the path 'Assets/Ultimate Replay 3.0/Demo/QuickStart.unity'*

This guide will assume that you have successfully imported the Ultimate Replay 3.0 asset into your Unity project and that you are starting with a blank scene 'File -> New Scene'. If you have trouble importing the asset for any reason, then please contact us for support. Contact information can be found at the end of this document.

1. The first step is to create a game object that we would like to be recorded and replayed by the replay system. For this example, we will use a falling physics cube as a demonstration, but really the object could be anything you like.

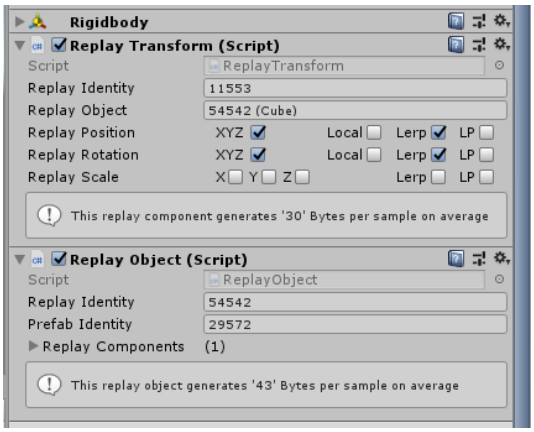
   Add a cube game object to the scene by going to 'GameObject -> 3D Object -> Cube' and position the object at **(0, 5, 0)**. We will also set the rotation of the cube to **(50, 60, 0**) because it will cause the cube to fall on its corner and tumble around a bit, so it will make the replay a little more interesting.



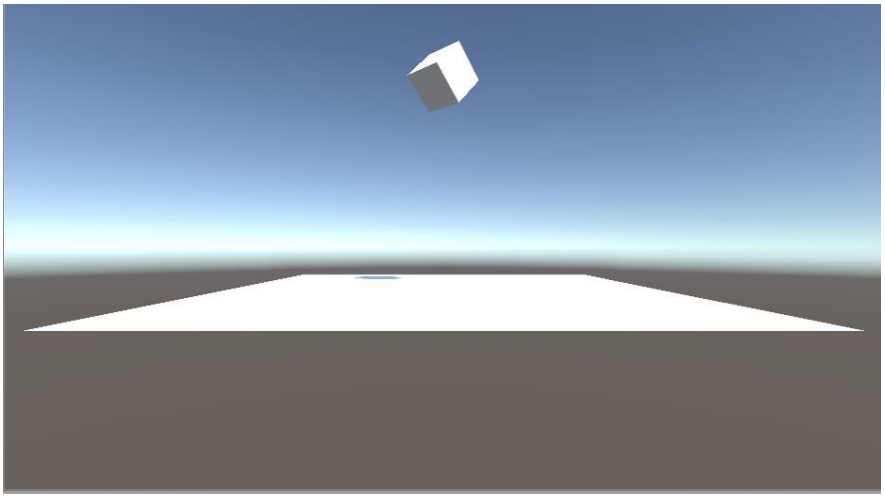
   With the cube object still selected, add a RigidBody component by going to 'Add Component -> Physics -> Rigidbody'. This will allow the cube to fall with gravity so that we have a moving object to record.

2. Next, we will want to make the cube object replayable as you might expect. To do this we can simply add a replay component to the cube and then the object will be fully replayable. Ultimate Replay 3.0 includes several built-in replay components, but for this example we will want to use the `ReplayTransform` to record and replay the object position and rotation.

Add a `ReplayTransform` component by going to `Tools -> Ultimate Replay 3.0 -> Make Selection Replayable -> Replay Transform`. You will see that 2 new components are added to the cube object.



3. Once we have setup the cube object, we will now add a ground plane to the scene so that the cube has something to collide with. Go to 'GameObject -> 3D Object -> Plane' to create a ground plane. Make sure the plane position is set to **(0, 0, 0).**



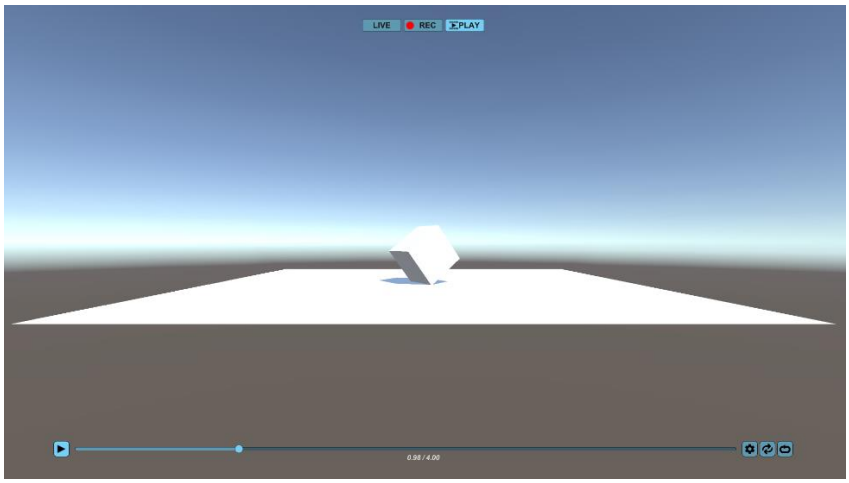
4. That is all the setup complete. We could now start recording using the Ultimate Replay API and we would capture a valid replay of the cube falling and tumbling as it contacts the ground plane. There is however an easier way to test the scene without the need for scripting. The ReplayControls is a UI prefab that allows you to quickly test out replays by providing a video player style interface for ease of use.

   Add a ReplayControls component by going to 'Tools -> Ultimate Replay 3.0 -> ReplayControls '. This will add a new game object to the scene named 'ReplayControls' which has a ReplayControls script component attached to it.

   Since the replay controls uses uGUI, you will also need to make sure an event system is present in the scene. Go to `GameObject -> UI -> Event System` to create one.

5. We can now test our replay scene by entering play mode in the editor to start the game. The ReplayControls will start recording automatically as soon as the game starts and the cube will fall to the ground and tumble. Once the cube has settled, click the 'Play' button of the ReplayControls UI in the top button group to enter playback mode. You will see that the

replay begins, and the cubes actions are replayed smoothly and accurately. You can also use the playback slider to jump to different positions in the replay as well as the settings menu which contains options for playback speed and direction.



Note: *While in playback mode, you can use the free-cam controls of WASD + RMB to fly the camera around the scene as the replay is running.*

Congratulations! You have successfully setup your first replay system using Ultimate Replay 3.0 and can now move on to bigger and better things. At this stage it may be worth taking a look at the following sections to get a better understanding of the asset and how it works:

- Replay Considerations – There are a few things to consider when using Ultimate Replay 3.0 in your game. This section will highlight things that you should be aware of.
- Replay Concepts – Learn about the key concepts used by Ultimate Replay 3.0
- Replay a Game Object – Learn how to record and replay a game object in depth.
- Replay Manager – Learn more about the heart of the replay system.
- Replay Prefabs – Need to instantiate or destroy objects in your game? This section will tell you how it can be achieved.
- Replay Recorder Components – Take a look in detail at the built-in recorder components offered by Ultimate Replay 3.0. Also learn how you could create your own recorder components.

There are also a number of demo scenes included with the asset which may be worth taking a look at. All demo scenes are located inside the demo folder at 'Assets/Ultimate Replay 3.0/Demo':

- CubeTest – A basic demo scene that spawns a large amount of physics cubes over a period of a few seconds. A useful demo if you want to instantiate or destroy recorded game objects.
- GhostCarDemo – This demo shows how a ghost vehicle in a racing game could be implemented. This demo uses more advanced techniques like multiple simultaneous record and replay operations, replay identity transfer and multi storage management.
- ActionReplayDemo – This demo shows how to implement a post-match action replay showing a montage of highlights from the recorded gameplay. This demo makes use of several different cameras to achieve the montage effect while tracking a racing car lap.

- KillcamDemo – Demonstrates how a first person killcam could be implemented in a multiplayer game. This demo features ragdoll, particle, audio recording and more. It also demonstrates how a replay can be viewed from the other players perspective (as the shooter) for a true killcam point of view.

As well as the included demos and documentation provided in this guide, there is also a dedicated samples repo on GitHub containing many useful code examples and additional documentation for working with Ultimate Replay 3.0. You can check it out here.

Replay Considerations

Ultimate Replay 3.0 uses a state-based storage technique to record game data at fixed intervals as specified by the user. These data samples are known as snapshots and can accurately describe the state of a scene at a given time offset known as the time stamp. Ultimate Replay 3.0 uses these snapshots to recreate the scene during playback using a slideshow effect to show these states quickly to give the illusion of seamless animation. There are a few things to consider when implementing Ultimate Replay 3.0 into your game project:

- **Replays are state-based:** Ultimate Replay 3.0 uses a state-based approach to record and replay the scene. This means that many snapshots will be captured per second during recording which contain enough state data to be able to recreate the scene at a later time. As a result, there are a few things to consider:
    - Replays are created by restoring these snapshots in quick succession in order to create a smooth playback sequence from a series of snapshots.
    - It is not possible to store recordings as popular video formats such as MP4 because the screen pixel data is simply not captured.
    - Replays are rendered in realtime by the active camera allowing you to switch cameras during playback, add or remove post processing effects during, create a highlight reel with multiple camera angles and more.
    - Your game does **NOT** need to be deterministic to support replays. Other replay systems that work by recording inputs must force all aspects of the game to be deterministic which can be very limiting and challenging to implement. Ultimate Replay 3.0 does not have this limitation due to the state-based nature so will be more suited and easier to integrate for a wider variety of games.
- **Scripts don't run during playback:** For the replay system to accurately recreate the scene as it was recorded, scripts may be disabled so that they cannot move or otherwise manipulate objects during playback which would cause inaccurate results. Only scripts attached to a replay object will be disabled so this will not affect standalone game systems such as game managers which are not repayable. Scripts will be enabled and disabled automatically by the replay system via the active replay preparer. Note that you can also disable or modify this behaviour if required by creating a custom replay preparer script. See the Replay Preparers section for more information. Note that scripts deriving from ReplayBehaviour are treated specially and will be allowed to run during playback.
- **Physics components are inactive during playback:** Physics components will also be disabled during playback mode to prevent inaccurate replays. The reason for this is that much like scripts, physics components can cause objects to be moved during playback as a result of rigid body updates or collision resolution. This would cause playback to become inaccurate, so the components are disabled or deactivated using the active replay preparer.
- **Storage targets cannot be used in multiple record operations simultaneously:** If you attempt to start more than one record or replay operation using the same storage target, you will receive an exception as this behaviour is not supported. By design, storage targets can only be in write or read mode at any given time. Even then, only a single replay operation can access the target to avoid many seek operations which could cause potential performance issues, or worse, multiple write operations which could corrupt the data stream.

Replay Concepts

This section will cover some of the essential concepts used by Ultimate Replay 3.0 and how they are used to affect recording and replay behaviour.

Ultimate Replay 3.0 is a state-based replay system meaning that the scene is sampled multiple times during recording to create a series of snapshots. These snapshots contain the necessary state data of all ReplayObject s in the scene such as position and rotation which are used during playback to reconstruct the scene exactly how it was during recording. By reconstructing these snapshots in order very quickly, it is possible to create the illusion of a seamless replay, a bit like a flipbook animation. This state-based approach has a few benefits over traditional screen recording techniques:

1. The replay can be viewed from different angles or cameras since the replays are rendered as they are running. You can also move the camera during playback and apply new post processing effects to change the appearance of a replay if required.
2. The state data recorded per scene sample is actually a very small amount of data when compared with screen recording techniques where the screen pixels need to be stored. This means that memory usage and replay file sizes can be very small allowing for long and complex replays to be recorded, especially with the compression techniques offered by Ultimate Replay 3.0.

Replay Manager

The replay manager is the heart of the replay system and is main interface used to interact with Ultimate Replay in your game from a scripting perspective. It contains all the methods to start record or replay operations which will then provide a means to interact with the replay system in various ways, from pausing/resuming to seeking during playback and much more. Check out the Replay Manager section for more detail.

Replay Operation

A replay operation is created when you start recording or when you start replaying a scene and will either be a `ReplayRecordOperation` or a `ReplayPlaybackOperation`. Any number of replay operations can run at the same time, although it is not possible to replay the same object multiple times for example.

- **ReplayRecordOperation**: Represents a single record operation which may operate on any number of replay objects. Any number of record operations can run in parallel, and it is possible to record the same replay object with multiple record operations. Note that each record operation must have its own dedicated `ReplayStorage`. Use this operation to control the record process by pausing, resuming, or stopping, as well as accessing other information such as current record duration.
- **ReplayPlaybackOperation**: Represents a single playback operation which may operate on any number of replay objects. Any number of playback operations can run in parallel, however it is not possible to replay the same object multiple times. Multiple playback operations can however share the same replay storage if required. Use this operation to control the playback process by pausing, resuming, stopping, or seeking, as well as accessing other information like current playback time, playback speed, etc.

Replay Identity

A ReplayIdentity is a unique serialized id value that all replay components are assigned by the replay system when created. The ReplayIdentity is used to identify each replay object, component, data segment etc so that the recorded data is restored to the correct objects. ReplayIdentities will be generated automatically by the replay system but there are occasions when you may like a particular replay object to take on the identity of a different object. This can be useful to record data from one object but replay on a different object as used in some use cases such as ghost vehicles. Take a look at identity transfer for more information.

Replay Scene

Replay State

A replay state is a high-performance storage object that is passed to all recorder components and is used to store primitive data types into a data stream. Some Unity types such as Color and Vectors are also supported for ease of use. Any time you need to manually record or restore any replay data, a ReplayState will be passed which you can use to write to or read from.

ReplaySnapshot

A ReplaySnapshot contains the entire state data of the active ReplayScene at a given time stamp. Multiple snapshots stored in order can be used to recreate the scene over a period of time much like a slideshow or keyframe animation. These snapshots are used as a higher-level storage device and can be stored to and fetched from a ReplayStorage directly. You will not need to deal with ReplaySnapshots directly but it is a good idea to understand what they are and their purpose.

Replay Scene

A ReplayScene represents a collection of ReplayObject s which should be used in a record or replay operation. By default, Ultimate Replay 3.0 will use all ReplayObject s in the active scene for all record and replay operations unless you manually specify a custom replay scene. Replay scenes are also responsible for a process called state preparation which is required for playback. Essentially, any components that could interfere with playback such as rigid bodies, colliders or scripts need to be prepared before playback commences.

Replay Storage

`ReplayStorage` is an end storage device used to store the data that is recorded by the replay system. The data that is generated by the replay system is in the form of a ReplaySnapshot which is then flushed to the active ReplayStorage when it should be stored. The StorageTarget could be anything from a memory storage target to file storage or more. You may also create custom replay storage if you needed to store the data in some other form such as in a database.

Replay Recorder Component

Recorder components are used to record and replay an associated Unity component. For example, the ReplayTransform component is intended to record and replay the Unity Transform component of a specific object. There are many other recorder components built for things like animation and audio, and it is also possible to create your own recorder components for unsupported or 3rd party components.

Replay a Game Object

In order to create a replay for your game using Ultimate Replay 3.0, you will need to decide which game objects should be replayable so that the necessary replay components can be added. The objects which are replayable will depend largely on the specific game but generally you will want to add replay components to any game objects which move, are animated, have effects such as particle systems etc. There are some exceptions but again this will depend on your game. An example would be an animated crowd in a sports game. It may not be necessary to record the whole crowd as you could just continue playing the animations during the replay.

Good candidates for replay objects are game objects that are core to the gameplay such as the player, enemies, projectiles, effects, sound effects, etc. Basically, any game object which is not stationary and whose behaviour does not consist of static continuous animation or similar. In some cases, you may need to replay custom gameplay elements which are not supported by built in components. For example: a GUI overlay which displays chat text or similar.

Once you have determined which game objects should be replayable, you then need to add the appropriate replay components. There are many replay components built into Ultimate Replay 3.0 which are covered in the Recorder Components section. For game objects that move, you will want to add a ReplayTransform component so that the game object transform will be recorded and replayed. For Animator components you will want to add a ReplayAnimator component and so on.

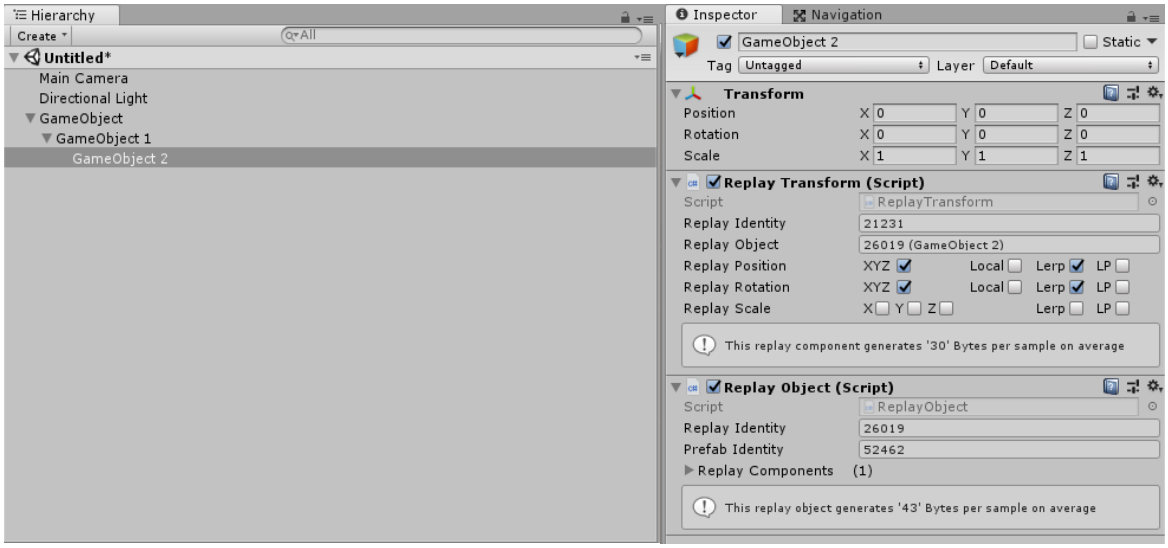
> **Note:** *A ReplayObject component will automatically be added when you attach a replay component to a game object. This is a required component and will manage one or more recorder components.*

Once you have added the necessary replay components, you should have a working replay system which will record and replay the state of all observed components. You just need to use the ReplayManager API to start and stop recording and playback. To quickly test out your newly added replay components, you can use the built in Replay Controls interface which will handle the record and replay API calls while providing a simple and easy to use interface.

Take a look at the included demo scenes to see how various replay techniques can be implemented and which replay components are used to do so.

Game Object Hierarchy

If you have a game object with one or more children that need to be recorded, then there are some things to consider. It is important that a ReplayObject component is attached to the highest-level game object that has a replay component attached. The ReplayObject component will usually be added automatically when attaching a replay component to a game object but in some cases it may not be added at the correct level. Take the following examples:



As you can see from the screenshots, 'GameObject' 2 has a ReplayTransform component added which causes a ReplayObject component to be added automatically at the same level. This is fine because there are no replay components attached to game objects higher in the hierarchy.

If we wanted to add a ReplayTransform component to 'GameObject 1' then this would cause an issue. If we selected 'GameObject 1' and then went to 'Tools -> Make Selection Replayable -> Replay Transform' then we would end up with the same components as 'GameObject 2'. The problem is that both objects will have a ReplayObject component attached which is inefficient and may cause issues. In scenarios like this, it is enough to simply remove the ReplayObject component attached to 'GameObject 2' and now the hierarchy setup is perfectly valid. The highest-level replay component in the hierarchy has the only ReplayObject component attached which will now manage both ReplayTransform components.

Note: *An exception to this rule is if 'GameObject 2' in the above example was a prefab instance, in which case multiple ReplayObjects would be the way to go in order to support dynamic creation and destrouction.*

Replay Manager

The Replay Manager is the main interface for Ultimate Replay and is used to control and query all replay operations using the static API. If you are coming from Ultimate Replay 1.0 you may already be familiar with the Replay Manager however in version 2.0 there are some major differences. It may be worth taking a look at the Upgrade Guide section if you haven't already as this section covers some of the major changes from the original asset.

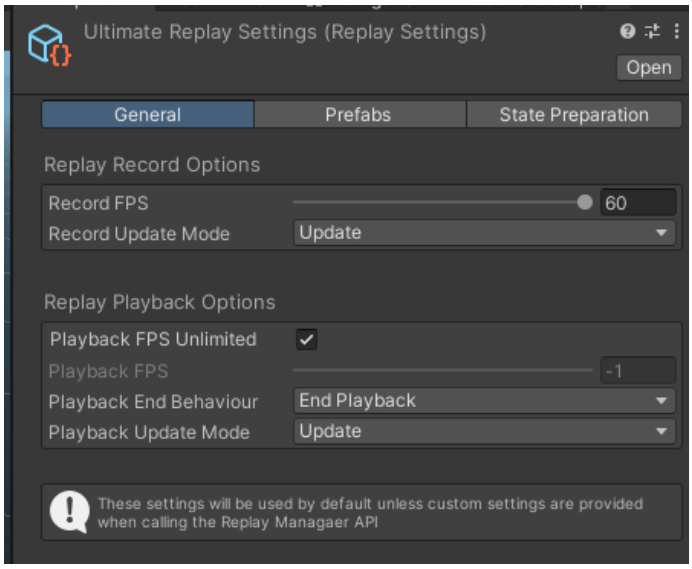
The ReplayManager is a type that defines the main API of Ultimate Replay 3.0 and will be used by your games scripts to control the replay system. The ReplayManager is implemented as a static API for ease of use, meaning that you can call its methods from any script without requiring an object reference. This means that there is no scene representation of the ReplayManager unlike the original asset. This means that scene changes do not cause issues or interfere with the replay system in any way.

Replay Settings

Ultimate Replay 3.0 has a number of global settings which affect various aspects of the asset. Usually the default settings will be OK for most games but they can easily be changed by going to 'Tools -> Ultimate Replay -> Settings' which will open the global settings in the inspector window. The settings window is also where you will add prefab references to objects which may be destroyed or instantiated dynamically while recording. Take a look at the Replay Prefabs section for more information.

General

The following section will cover the main settings which can be found on the default 'General' tab:

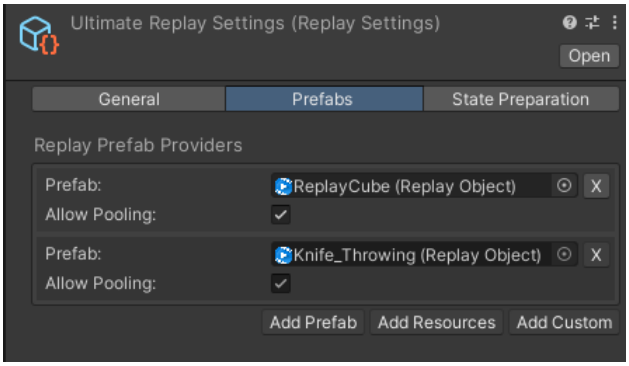


- **Record FPS:** The record frame rate used to determine how many snapshot frames are captured per second. The default value is '12' which will usually be more than enough for most games when interpolation is used.

- **Record Update Method:** The Unity update event used to update the entire replay system during the recording phase. This option is mostly for compatibility reasons and allows replay captures to occur at different points in the game loop. The default option is Update which will be fine for most games and all supported options are:
    - **Update:** Use the main Unity update method to update the recording phase of the replay system.
    - **Late Update:** Use the Unity late update method to update the recording phase of the replay system.
    - **Fixed Update:** Use the Unity physics update method to update the recording phase. This is not recommended unless you are having issues with the first 2 update methods.
- **Playback FPS Unlimited:** Enable this if there should not be any cap placed upon the playback update speed while a repay is occurring. Unlimited FPS means that the replay system will update replays as fast as possible up to your current game frame rate. It is recommended for most use cases to enable this option, but you can also disable and then specify a fixed fps value if required if for example you want updates to occur at a lower update rate: 30fps for example.
- **Playback FPS:** The replay frame rate used to determine how often playback operations are updated. Higher playback frame rates will result in smooth replays, even if the record fps is low. This is because playback runs interpolation on the recorded snapshots meaning that it is possible to create virtual snapshots in between recorded snapshots with the effect of interpolated smoothing, much like in keyframe animation systems. The default value is '-1' which means that playback will run at the game FPS which is recommended. Lower playback FPS values can be used if you experience playback performance issues but playback frame rates lower than the recorded FPS are not recommended.
- **Playback End Behaviour:** Determines what happens when a replay reaches the end of its recording. Options are:
    - **End Playback:** The playback operation will automatically finish and cause the associated ReplayObject s to be prepared for gameplay by switching to live mode.
    - **Stop Playback:** The replay will stop on the very last frame of the recording but will remain in playback mode. This means that playback operations such as seeking will still work as expected and you will need to manually call StopPlayback to exit replay mode.
    - **Loop Playback:** The replay will loop around to the start when the end frame is reached resulting in an infinite looping replay. You will need to call StopPlayback manually to exit replay mode. This mode is useful for after gameplay highlights that run until player input is received.
- **Playback Update Method:** The Unity update event used to update the entire replay system during the replay phase. This option is mostly for compatibility reasons and allows replay updates to occur at different points in the game loop. The default option is Update which will be fine for most games and all supported options are:
    - **Update:** Use the main Unity update method to update the playback phase of the replay system.
    - **Late Update:** Use the Unity late update method to update the playback phase of the replay system.
    - **Fixed Update:** Use the Unity physics update method to update the playback phase. This is not recommended unless you are having issues with the first 2 update methods.

Prefabs

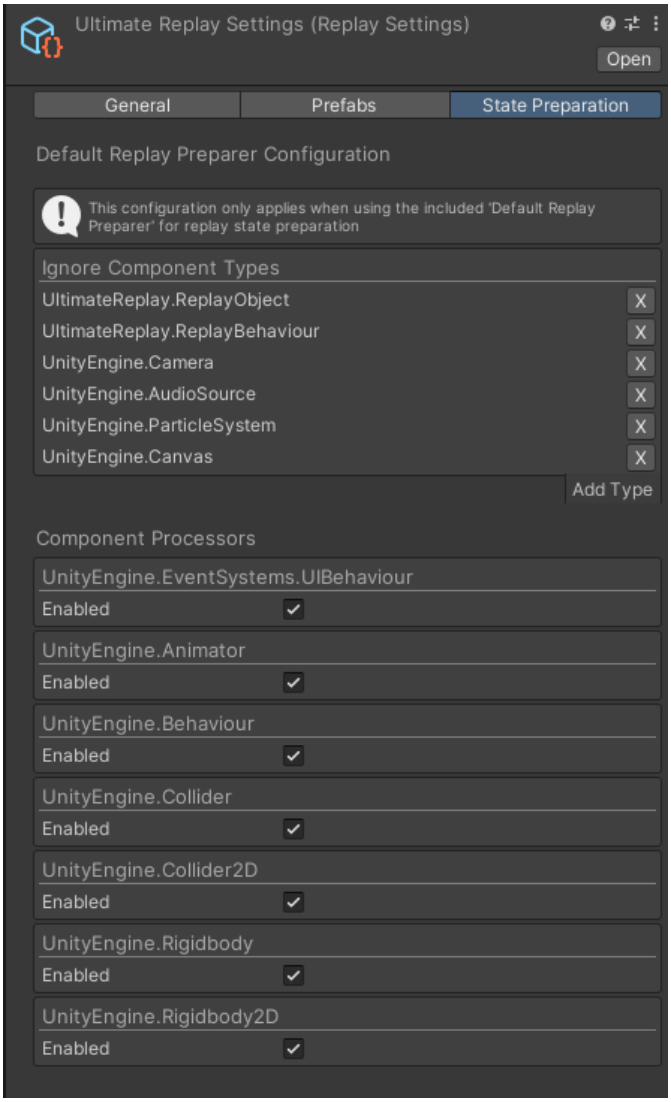
Replay prefabs must be registered in the settings if you need to instantiate or destroy an object during recording. In such a case the replay system will then be capable of instantiating or destroying a prefab instance during playback mode on demand to create an accurate replay.



Replay Prefab Providers: A collection of prefab references to ReplayObject s that may be instantiated or destroyed during recording. In order for Ultimate Replay 3.0 to support dynamic object creation and destruction during recording, you will need to inform the replay system about any prefabs that may be dynamic by adding them to the prefabs collection. Instantiating or destroying a prefab instance during recording that is not added to this collection will result in playback accuracy issues along with a warning message. Take a look at the Replay Prefabs section for more information.

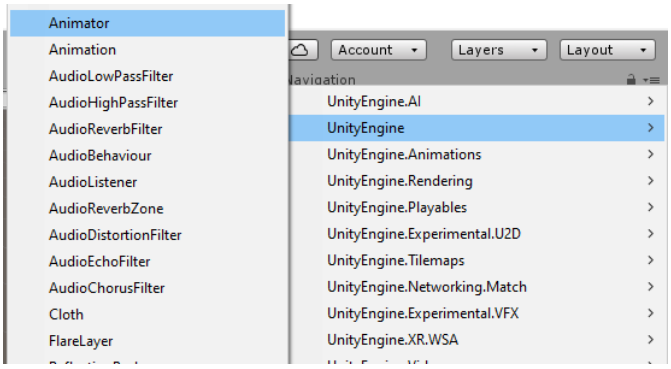
State Preparation

The next section will cover additional settings which can be found on the 'State Preparation' tab and relate to the built in Default Replay Preparer.



- **Ignore Component Types:** A collection of component types which will be ignored by the default replay preparer. This means that they will not be processed or affected in any way when switching between playback and live modes.

  This collection will already contain a few types which are added by default and we recommend that they remain in most cases. Adding new types can be done easily by clicking the ‘Add Type’ button and selecting your desired component type from the resulting context menu. Types are organised by ‘[namespace] -> [type name]’ to make them easier to find:



- **Component Processors:** This section contains settings for each component processor that is used by the default replay preparer. A component processor is simply a special type which prepares a specific component for either playback or live modes. Typically, these processors will either deactivate or disable their target component when entering playback mode and restore the target component to their original state when exiting playback mode.
  Each component processor has the following options:

    - **Enabled:** Is the component processor enabled. A disabled component processor will not run and as a result, will have no effect on the target component when the scene is being prepared.

Replay Scene

A replay scene is used to represent a collection of ReplayObject s which should be recorded or replayed. When you start a record or replay operation using the ReplayManager , you will usually need to pass a ReplayScene instance which will describe which objects should be included in the operation. You can create any number or ReplayScene s at a given time although you should take care not to start multiple operations on a single scene at the same time as this is not supported. For example, multiple record and replay operations can occur at the same time but not with the same scene instance. The ReplayManager will throw an exception is a scene instance is already in use.

Replay Scene Mode

A replay scene is a state object and can either be in live or playback modes. Changing the scene mode will cause all registered ReplayObject s to be prepared using the active replay preparer so that they are ready to receive record or replay updates.

- **Live Mode:** All ReplayObject s are reset to their initial state using the active replay preparer. This means that all scripts, physics components etc. are restored to their initial state. Usually meaning that they are re-enabled or re-activated so that they can interact with the game as usual.
- **Playback Mode:** All ReplayObject s are prepared for replay updates by the active replay preparer. Scripts and physics components will be disabled to prevent them from manipulating the objects during playback allowing the replay system to replicate the recording accurately.

Replay Preparers

A replay preparer is a special script which is executed on every ReplayObject in the ReplayScene when the scene mode is changed. The purpose of the replay preparer is to find and modify any components on the specified ReplayObject which could potentially interfere with playback accuracy. Components such as Rigidbodies or scripts are likely candidates as they could potentially move an object during a replay causing it to become out of place according to the recorded data. The state of such components is then saved or restored by the preparer depending upon the state change so that the component can be deactivated during playback.

By default, a built-in replay preparer called the DefaultReplayPreparer will be used to prepare ReplayObject s but it is possible to create your own replay preparer script if required. The default preparer will potentially affect the following component types:

- **MonoBehaviour scripts (Unless they derive from ReplayBehaviour):** Scripts will be disabled so that the 'Update' methods do not run.
- **Physics rigid body 2D / 3D:** Rigid bodies will be set to kinematic mode so that forces such as gravity do not affect the object.
- **Physics colliders 2D / 3D:** Colliders will be disabled so that collision resolution cannot affect playback.
- **Animator:** Animators will be disabled so that animation poses cannot be applied during playback.

Note: *Script components deriving from ReplayBehaviour will not be modified by the default replay preparer. You can derive from this class if you need to prevent a script from being disabled during playback on a particular replay object.*

Note: *Replay preparers work on a per object basis and will only affect ReplayObjects which were added to a ReplayScene. Entering or exiting playback mode will trigger the preparer to run on all ReplayObjects which are added to the active ReplayScene instance.*

Custom Replay Preparer

If you find that the default replay preparer is affecting components that you do not want it to, you could implement your own replay preparer so that you have full control over which components are affected.

Creating a custom replay preparer is as simple as implementing an interface and then registering it with the replay system. First you will need to implement the 'UltimateReplay.Core.IReplayPreparer' interface which has 2 methods:

- **PrepareForPlayback:** This method will be invoked when potential interfering components should be deactivated because the replay system is entering playback mode. The method will be invoked a number of times for each ReplayObject in the scene. You will need to use the Unity API to find such components and handle them accordingly.
- **PrepareForGameplay:** This method will be invoked when exiting playback mode as a result of calling StopPlayback and will run on all ReplayObject s in the associated ReplayScene . This method should be used to restore any modified components to their initial state.

If you wish to implement a custom ReplayPreparer then it may be worth taking a look at how the default preparer is implemented by examining the source code. You can find the default preparer implementation in the following source file (Not available in the trial version): 'Assets/Ultimate Replay 3.0/Scripts/Core/DefaultReplayPreparer.cs'. It may also be worth checking out the dedicated component preparers which can be found inside the 'StatePreparation' folder.

C# Code

```
1  class ExamplePreparer : IReplayPreparer
2  {
3      public void PrepareForPlayback(ReplayObject replayObject)
4      {
5          foreach(Collider collider in
6  replayObject.GetComponents<ReplayObject>())
7              collider.enabled = false;
8      }
9
10     public void PrepareForGameplay(ReplayObject replayObject)
11     {
12         foreach(Collider collider in
13 replayObject.GetComponents<ReplayObject>())
14             collider.enabled = true;
15     }
}
```

Once you have implemented a custom replay preparer, the next step is to register it so that the replay system can make use of it. A replay preparer instance is associated with every ReplayScene instance as the preparer needs to be run on every ReplayObject in the ReplayScene . This means

that you can provide an IReplayPreparer implementation in the constructor of the ReplayScene type. If no preparer is passed, then the default preparer is used automatically.

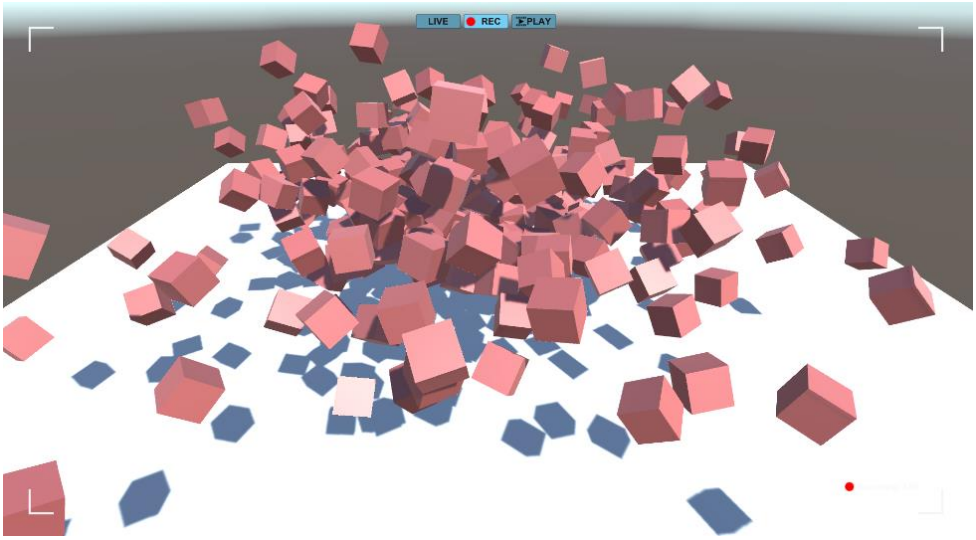
C# Code

```
1  class Example : MonoBehaviour
2  {
3      void Start()
4      {
5          ReplayScene scene = new ReplayScene(new ExamplePreparer());
6          scene.AddReplayObject(...);
7
8          ReplayManager.BeginPlayback(null, scene);
9      }
   }
```

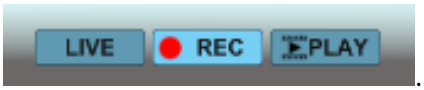
Note: *It is possible to implement different replay preparers for different playback operations if required. For example: you may want some replays to have colliders enabled so that the player can interact with them while you may want another replay to be unaffected. This is possible by creating multiple ReplayScene instances.*

Replay Controls

Ultimate Replay 3.0 includes a simple replay controls UI just like in the original asset. This UI is implemented using the legacy Unity immediate mode GUI and is intended for quick testing and demonstration purposes. If you need a similar in game UI then we recommend that you create your own using your favourite UI package or asset since the immediate mode GUI is now depreciated. Here you can see the replay controls UI in record mode:



The replay controls UI has 3 different modes which can be used to switch between different states in the replay system. Ultimate Replay 3.0 supports an unlimited number of simultaneous record and replay operations; however, the replay controls can only be used to control a single replay operation at any given time. This means that the controls can be used to either record, replay or remain idle. The replay state can be changed at any time using the controls in the upper left corner and is also indicated by the selected button:

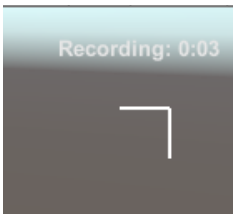


- **Live Mode:** Live mode allows gameplay to continue as expected. All replay objects have their components restored gameplay mode and physics and animation systems can control objects as normal.
- **Record Mode:** All replay objects are recorded at a fixed rate based upon the recording interval as specified via the settings window. All active ReplayObject s in the scene will be recorded to a memory storage target.
- **Playback Mode:** Playback mode allows you to view the recorded data and see the replay as it was recorded. All replay objects will be prepared for playback which involves disabling various game systems such as physics and scripts which could otherwise cause the object to move out of playback position. The replay system will then proceed to recreate the recording by restoring scene snapshots or key frames.

Record Mode

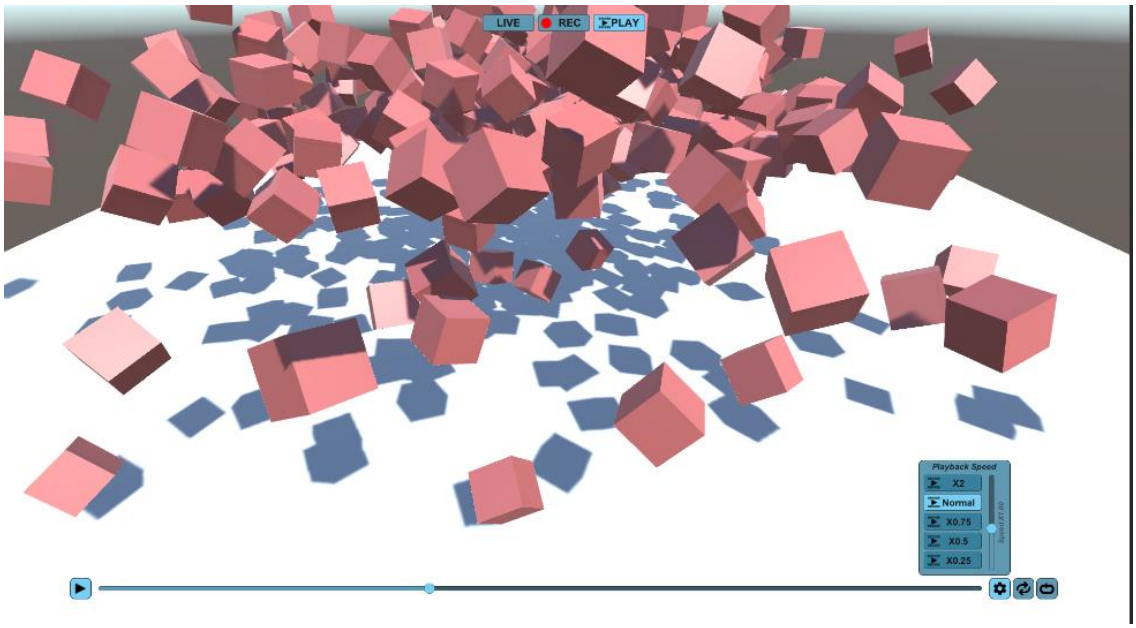
The replay controls include a record mode which is used to record replay objects in the scene over a period of time. The record state is indicated by the framing borders being displayed as part of the UI along with the current record duration in seconds. The replay controls will always store its recorded data in a memory target with an unlimited size meaning that long recordings are possible.

You can start recording with the replay controls UI by selecting the 'Rec' mode button which will begin capturing the scene.



Playback Mode

The replay controls also provide a convenient way to view the last recorded segment with full control via a playback seek control as well as speed and direction controls. The replay controls will also offer free cam perspective during playback meaning that you can move the camera around the scene using navigation keys to view the replay from any perspective.



- **State Controls:** As previously mentioned, the state controls allow you to switch between Live, Record and Playback modes offering full testing abilities.
- **Free Cam Hint:** The free cam hint is only displayed while in playback mode and indicates that you are able to move the camera around using the navigation controls. Keyboard and mouse control hits will also be displayed here.
- **Playback Speed:** The speed that the replay will be played at. The speed slider allows speeds between 0-2 to be specified where 1 is the default playback speed. A value of 0 would cause the playback to halt whereas a value of 2 would cause playback to run twice as fast. Note that the GUI slider is limited to 0-2 for ease of use but the replay system accepts much larger values via code.

- **Playback Direction:** Used to toggle between forward and reverse playback.
- **Playback Time:** Displays the current playback time value in seconds for the recording (Also indicated by the seek slider position) along with the total duration of the recording in seconds.
- **Playback Settings:** Used to show/hide the playback options popup containing the speed and direction controls.
- **Playback Slider:** The playback slider indicates the current playback position in relation to the overall recording. The slider can also be used to seek to different points in the replay by dragging or clicking along the slider bar. Note that interpolation is not available while seeking so snapping or jumping may occur when slowly scrubbing.
- **Play/Pause:** Allows playback to be paused or resumed at any point.

Free Cam Mode

One of the advantages of using a state-based replay system is that you are able to view the replay from any camera angle, or even multiple cameras in succession in order to create a highlights reel or similar. This is possible because the replay is rendered in real-time using the active camera.

The ReplayControls component makes use of this feature by allowing a free cam mode during playback which allows you to fly the camera around the scene as a replay is running. While in playback mode, you will see in the upper right corner that 'Free Cam' mode is enabled, meaning that you can manipulate the camera using the following controls:

- **W:** Move the camera forward relative to the current camera heading.
- **S:** Move the camera backwards relative to the current camera heading.
- **A:** Move the camera left relative to the current camera heading.
- **D:** Move the camera right relative to the current camera heading.
- **RMB + Drag:** Pan / tilt the camera angle based upon the mouse movement.

Exiting and re-entering playback mode will cause the free cam to be reset to its initial position which will be the position of the active rendering camera when entering playback mode.

> **Note:** *in order to preserve any gameplay cameras in the scene, the replay system will create its own camera that will be used during free cam mode which will adopt the position and rotation of the active scene camera. This will give the effect of moving the current scene camera but in actual fact, scene cameras will be left untouched.*

Replay Techniques

Replay Animation

Recording and replaying animated objects is a common use case of Ultimate Replay 3.0 and as a result we have added support for replaying Animator components, as well as support for IK animations via an alternative approach.

The ReplayAnimator component can be attached to a game object in order to record its animations. You will need to assign the ObservedAnimator property of the ReplayAnimator component to the Animator that you want to record and replay. After that, your animations should be recorded and replayed seamlessly by the replay system. Take a look at the ReplayAnimator section for more detail.

Some games may make use of IK animation to position bones via scripts to reach a target pose. Ultimate Replay 3.0 can also support IK animation, although a different approach needs to be used to setup the object. Essentially, each bone in the object skeleton that you wish to record should have a ReplayTransform component attached, and a single managing ReplayObject component at the root. If this sounds too complicated then don't worry, we have created a editor tools to help setup these replay component properly. The Replay Humanoid Configurator can be used to add the necessary replay components. Take a look at the Replay Humanoid Configurator section for more information.

Replay Ragdolls

Some games may make use of physics-based ragdolls for enemy deaths or similar which can also be recorded and replayed by Ultimate Replay 3.0. The process of recording a ragdoll character or similar is much the same as recording IK animation and requires that each bone in the skeleton has a ReplayTransform component attached. If your ragdoll has a humanoid structure, then setup is made quite simple by using either **Replay Humanoid Rig** or **Replay Generic Rig** replay components which make this process simpler by offering a single component to record the entire hierarchy.

These simple rules will ensure that your replay components are placed on the correct objects in the ragdoll hierarchy:

1. A ReplayObject component is required on the very root of the ragdoll object. This will usually be the highest object in the hierarchy.
2. A ReplayTransform should be added to every bone in the hierarchy. An easy way to do this to setup your ragdoll using the Unity ragdoll window, and then add a ReplayTransform component to every object in the hierarchy that has a 'Character Joint' component attached.
3. Any ReplayTransform components that are not attached to the very root of the object should have their position and rotation options set to record in local space.

Here is an example setup to give you a better idea (This example assumes that you have already setup your ragdoll in Unity):

-Root	ReplayObject , ReplayTransform *(World Space)*
--Bone 1	*Character Joint, Replay Transform (Local Space)*
--Bone 2	*Character Joint, Replay Transform (Local Space)*
---Bone 3	*Character Joint, Replay Transform (Local Space)*
------Bone 4	*Character Joint, Replay Transform (Local Space)*

After attaching replay components following the structure above, you can immediately test the scene to ensure that everything is working correctly. It may also be worth taking a look at the included killcam demo scene which uses the ragdoll replay technique. The demo scene can be found at 'Assets/Ultimate Replay 3.0/Demo/Killcam.unity'

Killcams

Killcams are another use case that Ultimate Replay 3.0 fully supports. A killcam is usually used to replay the last few seconds of the game when a player is killed so that they can see the death from the point of view of the shooter. If we break down this problem, we can see that the following things are required in order to create a working killcam system:

- Continuous recording of the last (n)seconds of gameplay.
- The ability to view the replay from different perspectives.
- A scene to playback the recording without outside influences. For example, other players in a networked game.

Ultimate Replay 3.0 has support for endless continuous recording when using a ReplayMemoryTarget as the storage device. Generally, a killcam will only need to use memory storage as any old data is no longer relevant and can be discarded. A ReplayMemoryTarget has a time value in seconds that can be assigned to which represents the maximum length of recording that can be stored. This is not a normal limit though as the time value is a negative offset. Ie. This time value is used to constrain the recorded data to the last x amount of seconds so that the smallest amount of memory is used and endless recording is possible without running into memory usage issues.

Note that some games may like to use a killcam but also have the ability to record complete game sessions at the same time. Ultimate Replay 3.0 can now support an unlimited number of simultaneous replay operations, so it is possible to both record in memory for the in-game killcam, and stream the entire game session to file or another storage device if required.

Another key aspect of a killcam is that the replay viewpoint is from the perspective of the shooter. This means that you get to view the actions of another enemy/player leading up to the death as if you were in their shoes. Luckily, this is something that is very easy to achieve using Ultimate Replay 3.0 due to the state-based approach used for replays. Ultimate Replay 3.0 renders all replays in realtime using the active camera. This camera can be positioned at any location and moved as required in order to view the replay from any location.

The easiest way to achieve this would be to attach a secondary camera to the enemy character model assuming a first person killcam is required. This camera should be setup as a first-person view for the enemy but will not be activated until you need to view the replay. You will also need to record the transform of this camera so that the players movements are captured. Then when entering playback mode, it is simply a case of switching cameras to view the replay from the shooter perspective. This step can be repeated for each enemy/player in the game so that all potential shooters can be used as the viewing perspective.

One final thing to consider when implementing a killcam is where a replay will be constructed. Ultimate Replay 3.0 uses the recorded scene objects to reconstruct the scene which could potentially cause issues. For example: If you have a networked multiplayer game where you send player updates across the network, when you switch to replay mode, other clients may receive the results of the replay rather than the gameplay. In this scenario, you would need to suspend network

updates while the replay is running so that the local scene is not synced to other clients, and also that other clients do not affect the local scene used for the replay which could cause inaccurate playback. It is not a major issue, but something to take into account when designing a killcam system.

Ghost Car

Ultimate Replay 3.0 can be used to create ghost vehicles for a racing game, and it is now much easier and better supported than in the original asset. A ghost vehicle is used in racing games to show the player their previous best racing line/time, usually via a semi-transparent non-collidable car. To setup a ghost vehicle system there are a few things to consider:

- The player vehicle is used for recording.
- Usually a different ghost vehicle object is used for playback.
- The player could beat their previous time and the ghost vehicle should display the fastest lap only.

Recording the player vehicle is straight forward and can be achieved with the built-in recorder components, namely the ReplayTransform component. This would record the vehicles transform as it drives around the track. The problem comes when we need to replay the recording. Usually with Ultimate Replay 3.0 you could just call BeginPlayback and the replay would run just fine. The issue though is that the replay would be played back on the player car instead of the ghost vehicle which is not desirable. This can be resolved quite easily though using identity transfer to allow the ghost vehicle object to take on the identity of the player car for playback purposes. Take a look at the identity transfer section for more information.

Using the identity transfer technique, we can allow the player car to record the information as it drives around the track, and then replay that information onto a different ghost vehicle car. This means that the player car is not taken over by the replay and is free to drive another lap. It also means that a completely different game object usually with a different visual appearance can be used for the ghost vehicle which is an ideal solution.

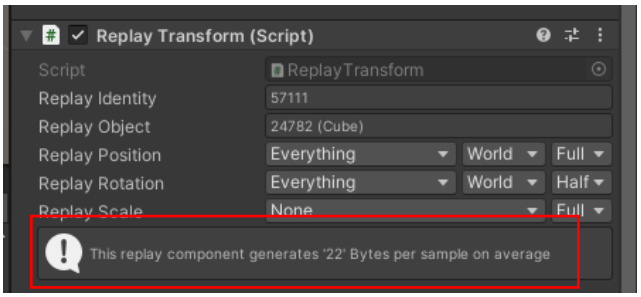
Another thing to consider when implementing a ghost vehicle system is that the player could beat their previous lap and the ghost vehicle would need to use this latest replay to reflect this. This means that we need to record every lap that the player car completes just in case they posted a faster time. This is quite simple to do using the following rules:

1. Start recording the player car when they cross the start/finish line and wait for the lap to complete.
2. Check if we have any previous times posted.
    a. If yes, then we check the newly posted lap time to see if it was faster and store the recording if so. The recording can safely be discarded if the time was not a new record as a previous recording will exist.
    b. If no, then we keep a reference to the storage target that contains the recording and start recording the player car again using a new recording target.
3. Create a ghost vehicle using the identify transfer process and replay the saved storage target which contains the fastest lap time.
4. Loop back to step 2 or until the player quits the game.

Replay Statistics

Ultimate Replay 3.0 uses a state-based approach and needs to store data for each recorder component of every ReplayObject in the scene in order for replays to be captured. On top of this, snapshot frames are captured in quick succession, often at over 16 frames per second which can result in quite a bit of data to store. Ultimate Replay 3.0 features some highly effective lossless compression techniques to keep that figure low but you can also save storage space on a per component basis. Many recorder components such as ReplayTransform have inspector properties which control which data is recorded and how accurately. By tuning these components to only record what is needed, you will be able to reduce the overall storage requirements for your replays.

It is important to note that saving a couple of bytes per component may not seem like much, but at over 16FPS and many recorder components in the scene, it does in fact add up to make quite a difference to the overall size. You will notice that any component deriving from ReplayBehaviour will display useful stats in the inspector window indicating how much data is generated per average sample. This figure is essentially the amount of data you can expect to be produced by the component for every recording sample prior to compression techniques.



Note: *Some recorder components may only be able to provide accurate data statistics while in play mode. An example would be the ReplayAnimator component which displays a sample size of '0' while the game is not running, but displays accurate statistics in play mode.*

When you change the properties of the recorder components, you will see that the statistics data updates in real time to give you immediate feedback on the storage requirements for the component. This is highly useful to fine tune a recorder component for playback accuracy vs storage size.

Report a Bug

At Trivial Interactive we test our assets thoroughly to ensure that they are fit for purpose and ready for use in games, but it is often inevitable that a bug may sneak into a release version and only expose itself under a strict set of conditions.

If you feel you have exposed a bug within the asset and want to get it fixed, then please let us know and we will do our best to resolve it. We would ask that you describe the scenario in which the bug occurs along with instructions on how to reproduce the bug so that we have the best possible chance of resolving the issue and releasing a patch update.

http://trivialinteractive.co.uk/bug-report/

Request a Feature

Ultimate Replay was designed as a complete replay system, however if you feel that it should contain a feature that is not currently incorporated then you can request to have it added into the next release. If there is enough demand for a specific feature, then we will do our best to add it into a future version. Please note, requested features should fall within the scope of the asset and unrelated or overreaching features will not be added.

http://trivialinteractive.co.uk/feature-request/

Contact Us

Feel free to contact us if you are having trouble with the asset and need assistance. Contact can either be made by the contact options on the asset store or via the link below.

Please attempt to describe the problem as best you can so we can fully understand the issue you are facing and help you come to a resolution. Help us to help you :-)

http://trivialinteractive.co.uk/contact-us/