Replay Toolkit

Introduction:

The Replay Toolkit Plugin is a runtime replay management system developed for Unreal Engine that provides recording, playback, timeline scrubbing, replay management, and playback control functionalities through a centralized subsystem architecture.

The plugin is built on top of Unreal Engine's Replay System and utilizes the DemoNetDriver, NetworkReplayStreaming, and GameInstance frameworks to capture gameplay sessions and replay them later. It exposes a simplified interface for recording simulations, reviewing recorded sessions, controlling playback speed, pausing and resuming replays, performing timeline scrubbing, and managing stored replay files.

A key feature of the plugin is its support for smooth replay scrubbing by preserving critical actors such as spectator controllers, spectator pawns, and user-defined actors during seek operations. This helps maintain camera continuity and prevents unnecessary actor recreation while navigating through replay timelines.

The system is implemented as a UGameInstanceSubsystem, allowing replay functionality to remain globally accessible throughout the application's lifecycle while maintaining a clean separation from gameplay-specific logic.

How To Install:

1. Add the Asset Pack content to your project.
2. Restart the editor.
3. Add the widget “WBP_ReplayControls” to the view; this widget controls all the replay functionality of the plugin.

System Overview:

Replay Toolkit is an Unreal Engine subsystem-based plugin that provides functionality for:

- Recording gameplay sessions
- Playing back recorded sessions
- Timeline scrubbing
- Replay pause/resume
- Replay speed control
- Replay management (search, rename, delete)
- Actor preservation during replay scrubbing

The system is implemented through UReplayManagerSubsystem, which acts as a centralized interface for Unreal's Replay System and DemoNetDriver.

Architecture:

Core Class:

ReplayManagerSubsystem

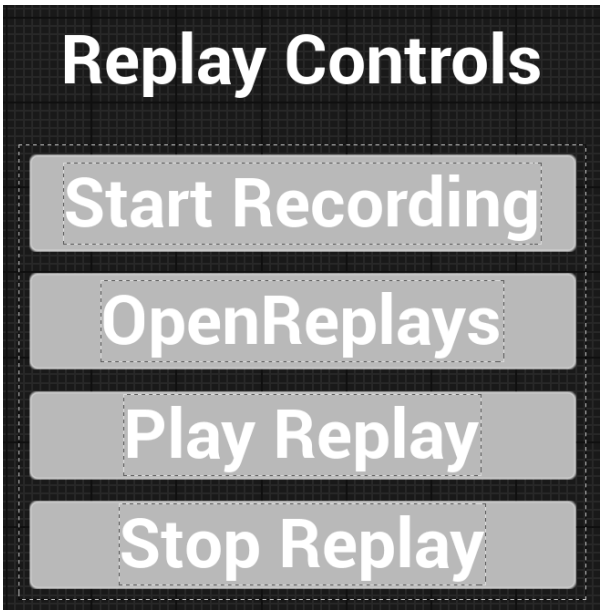
Inherits from:

UGameInstanceSubsystem

Responsibilities:

- Manage replay recording lifecycle
- Manage replay playback lifecycle
- Handle replay scrubbing
- Query replay metadata
- Enumerate saved replays
- Rename and delete replay files
- Preserve runtime actors during scrubbing

Recording System:



StartSessionRecording()

Purpose:

Starts a new replay recording session.

Configuration

Before recording begins, several replay console variables are configured:

demo.CheckpointUploadDelayInSeconds 5
demo.bSkipStartupActorRollback 1
demo.RecordHz 120

Process

1. Configure replay recording settings.
2. Retrieve current GameInstance.
3. Call:

StartRecordingReplay()

Parameters

Parameter	Description
InReplayName	Internal replay file name
InFriendlyName	Display name shown to users

StopRecordingSimulationReplay()

Purpose:

Stops active replay recording.

Process

StopRecordingReplay()

Replay Playback System:

PlayReplay()

Purpose:

Loads and plays a previously recorded replay.

Configuration

net.AllowAsyncLoading 1
net.DelayUnmappedRPCs 1

Process

PlayReplay()

Parameter

Parameter	Description
InReplayName	Replay file to load

Replay State Management:

PauseReplay()

Purpose:

Pauses or resumes replay playback.

Process

When paused:

1. Get local PlayerState.
2. Assign PlayerState to:

WorldSettings->SetPauserPlayerState( )

When resumed:

SetPauserPlayerState(nullptr)

Debug Output

Displays:

Replay Paused
Replay Resumed

StopReplay()

Purpose:

Stops active replay playback and resets replay state.

Process

1. Seek replay to time 0.
2. Pause replay.
3. Clear DemoNetDriver reference.

Internal Actions

GotoTimeInSeconds(`0.0f`)
PauseReplay(true)
DemoNetDriverRef = nullptr

Replay Time Control:

GetReplayCurrentTime()

Returns:

DemoNetDriver->GetDemoCurrentTime()

Used for:

- Timeline UI
- Replay HUD
- Scrubbing systems

GetReplayTotalTime()

Returns:

DemoNetDriver->GetDemoTotalTime()

Used for:

- Timeline maximum range
- Replay duration display

Replay Scrubbing System:

SeekInActiveReplay()

Purpose:

Jump to a specific timestamp.

Workflow

1. Preserve important actors.
2. Call:

`GotoTimeInSeconds()`

3. Register completion callback.

Parameter

Parameter	Description
InTimeSeconds	Target replay time

OnScrubFinished()

Purpose:

Receives notification after replay seek completes.

Event

OnSliderScrubFinished.Broadcast()

Used by UI systems.

Playback Speed System:

SetReplayPlaybackRate()

Purpose:

Adjust replay playback speed.

Process

DemoPlayTimeDilation = Rate

Examples:

Value	Result
0.25	Quarter speed
0.5	Half speed
1.0	Normal speed
2.0	Double speed

Replay Discovery System:

FindReplays()

Purpose:

Search local replay storage.

Process

1. Create ReplayList object.
2. Enumerate replay streams.
3. Populate replay entries.
4. Sort by timestamp.

Callback

OnEnumerateStreamsComplete()

Replay Sorting:

Replays are sorted by:

Timestamp.GetTicks()

Order:

Newest → Oldest

Replay Rename System:

RenameReplay()

Purpose:

Rename an existing replay.

Validation

Checks:

- New name is not empty.
- Replay list exists.
- Name does not already exist.

Operation

RenameReplay()

Callback

OnReplayRenamed()

Event

OnReplayRenamedSuccessfully.Broadcast()

Replay Delete System:

DeleteReplay()

Purpose:

Delete replay from storage.

Operation

DeleteFinishedStream()

Callback

OnReplayDeleted()

Event

OnReplayDeletedSuccessfully.Broadcast()

Replay Initialization:

IsReplayActive()

Purpose:

Checks if replay playback is active.

Additional Functionality

Automatically initializes:

DemoNetDriverRef

when replay playback begins.

Return

true

if replay is currently playing.

OnReplayStarted()

Purpose:

Stores DemoNetDriver reference after replay starts.

Actor Preservation System:

PrepareActorsForScrubbing()

Purpose:

Prevents actor destruction and recreation during timeline seeking.

Preserved Actors

Spectator Controller

GetSpectatorController()

Spectator Pawn

GetSpectatorController()->GetPawn()

User Defined Classes

Stored inside:

ClassesTobeAddedInNonQueueForScrubbing

Each matching actor is registered with:

AddNonQueuedActorForScrubbing()

Benefits

Prevents:

- Camera reset
- Spectator recreation
- UI actor recreation
- Manager actor destruction

Improves scrubbing performance and visual stability.

Replay Streamer Management:

GetStreamer()

Purpose:

Creates or retrieves replay streamer instance.

Implementation

```
FNetworkReplayStreaming::Get()
  .GetFactory()
  .CreateReplayStreamer()
```

Uses

- Replay discovery
- Rename operations
- Delete operations

Event System:

Broadcast Events

```
UDELEGATE()
DECLARE_DYNAMIC_MULTICAST_DELEGATE(FOnSliderScrubFinished);

UDELEGATE()
DECLARE_DYNAMIC_MULTICAST_DELEGATE_OneParam(FOnSearchFinished, UReplayList*, ResultList);

UDELEGATE()
DECLARE_DYNAMIC_MULTICAST_DELEGATE(FOnReplayDeletedSuccessfully);

UDELEGATE()
DECLARE_DYNAMIC_MULTICAST_DELEGATE(FOnReplayRenamedSuccessfully);
```

Replay Search

OnSearchFinishedDelegate

Replay Rename

OnReplayRenamedSuccessfully

Replay Delete

OnReplayDeletedSuccessfully

Scrub Finished

OnSliderScrubFinished

Dependencies:

Required Unreal Modules:

Engine
Core
CoreUObject
GameplayStatics
NetworkReplayStreaming
DemoNetDriver

Required Includes:

Engine.h
World.h
GameplayStatics.h
GameInstance.h
WorldSettings.h

TimerManager.h
NetworkReplayStreaming.h

Typical Usage Flow:

Recording:

1. StartSessionRecording()
2. Simulate gameplay
3. StopRecordingSimulationReplay()

Playback:

1. FindReplays()
2. PlayReplay()
3. PauseReplay()
4. SeekInActiveReplay()
5. SetReplayPlaybackRate()
6. StopReplay()

Management:

1. FindReplays()
2. RenameReplay()
3. DeleteReplay()

Conclusion:

The Replay Toolkit Plugin provides a comprehensive replay management solution built on Unreal Engine's native replay framework. By encapsulating recording, playback, timeline navigation, replay management, and actor preservation functionality within a centralized subsystem, the plugin simplifies the integration of replay capabilities into simulation and gameplay projects.

The architecture is designed to be modular, scalable, and extensible, allowing future enhancements such as advanced replay analytics, bookmarking, event tagging, multiplayer replay support, cloud-based replay storage, and enhanced UI integrations. Through its support for smooth timeline scrubbing, playback controls,

and replay lifecycle management, the system delivers a reliable foundation for reviewing, analysing, and revisiting recorded gameplay or simulation sessions.

Overall, the Replay Toolkit serves as a robust framework for applications that require replay functionality, enabling developers to provide users with an efficient and intuitive replay experience while leveraging Unreal Engine's existing networking and replay infrastructure.

Support & Customization:

This pack is designed to be a “living” asset and will continue to evolve over time. If you encounter any issues or have questions regarding implementation, please feel free to reach out through our support link or Discord server.

For faster support, we recommend joining our Discord server. Our team is ready to help and answer any questions you may have.

Level up your game's feel and performance today!

⭐⭐⭐⭐⭐ Your Review means the world to us!
Great feedback comes back as even better features.
If there are any suggestions or ideas for improvement, feel free to contact us.If possible, we will include your request in future updates.

📰 Follow us for updates and news!
Youtube | Facebook | Email | Linkedin

Website: 300mind.studio

Thank you for choosing the Replay Toolkit. We hope this documentation helps you. Happy Gaming!

Enjoy building your game! 🎮

THANK YOU