Introduction

Easy Replay System allows you to record any kind of information (positions, rotations, scales, states, field values, etc) and then play it back at a given playback rate or even seek to an exact moment. It comes with some built-in trackables and It is highly extensible so you can implement whatever you need to be recorded and played back.

Main Components

Easy Replay Systems relies on two main components to work.
The Tracker, which is the manager in charge of recording and replaying all the action in a lapse of time.
The Trackables, that is, each of the elements that will be recorded by the Tracker.

Basic usage

First of all, you need to add the Tracker to the scene.
The Tracker may be used in two main ways. Please, decide which one better suits your needs:

- You may drag and drop ERS Tracker prefab in your scene. Check the “Debug Mode” checkbox to receive tracking feedback on the console panel.

or

- You may use the System right away, and the ERS Tracker will be instantiated on the fly

In either way, the tracker must be present before adding the Trackables.

The second thing you’ll need is to add your target trackables, either by dropping the Trackable component of your choice in the inspector or by adding it at runtime in your own script.
The Trackable component is an abstract class, all of the specific Trackable components must inherit Trackable. You have two built-in trackable components (TrackableTransform and TrackableSprite), more components will be added on further releases as they are needed. Feel free to implement any other functionality you may need by inheriting the Trackable class.

To start recording, simply call:

```
Tracker.Instance.Record();
```

To stop:

```
Tracker.Instance.Stop();
```

To play back:
```
Tracker.Instance.Play();
```

To pause when playing:
```
Tracker.Instance.Pause();
```

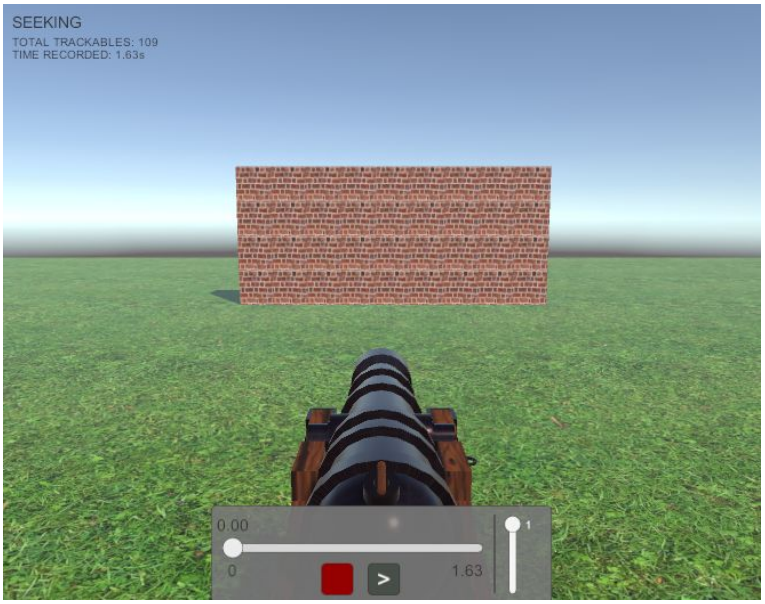
This are the basic commands. For more methods and customization read the Script documentation section below.

ERS Tracker Debugger

For your convenience, we included a ERS Tracker Debugger to let you test your scene before adding your own scripts. It allows you to record, stop, replay, change play speed (even negative, to rewind) and seek your track. To use it, simply drop it in the scene along with the ERS Tracker and the Trackable gameobjects and hit Play in the editor.

Demos

Three demo scenes are included in the package:

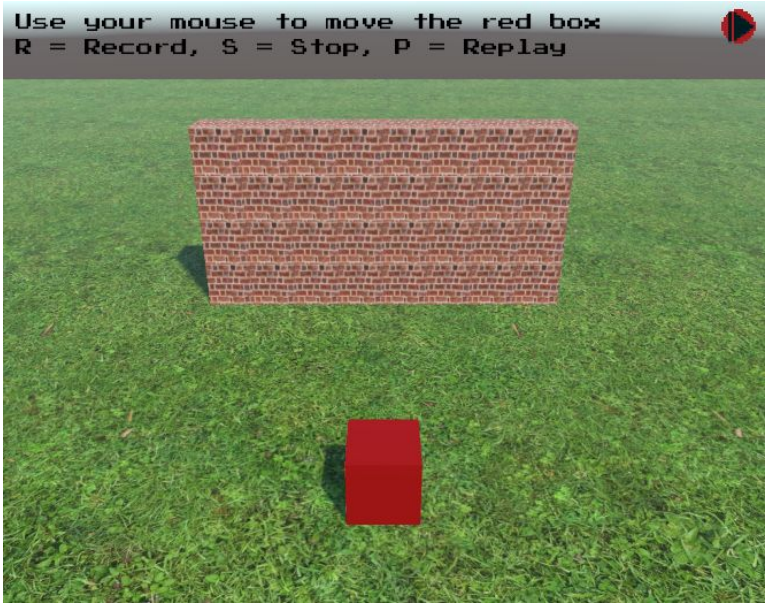


Cannon

In this demo scene you'll find the ERS Tracker and the ERS Tracker Debugger, as well as several rigidbodies with the TrackableTransform component added.
The cannon listens for the ERS Tracker Debugger events to launch a ball at the wall of rigidbodies.
Have fun recording, replaying, seeking and changing playback speed!

Runner

In this demo scene the Tracker is created on the fly. You'll find an example of the usage of the TrackableSprite, as well as triggering a record call at a given point and replaying a few seconds after the stop call.



FreeRecord

In this demo scene the player records, stops and replays whenever a key is hit. The box is moved with the mouse freely.

Script documentation

Although all the code is fairly well documented, here you have a selection of the main methods and parameters of the Tracker:

Methods:
`GetDebugInfo` → Returns some information about the state of the Tracker
`AddTrackable(Trackable t)` → Adds the Trackable to the list of Trackables to be managed by the Tracker
`RemoveTrackable(Trackable t)` → Removes the Trackable from the list of Trackables managed by the Tracker
`Record()` → Starts recording (by default at 10FPS)
`Record(float recordingRate)` → Records at a given rate. Default is 0.1 (FPS_10). But for non interpolatable trackables (eg: sprite frames) you may shorten it. eg: 0.033 (FPS_30) or 0.017 (FPS_60) use sparingly because it will affect performance
`Play()` → Plays back the last recording
`Play(float speed)` → Plays back the last recording at a given speed (1 = 100%)
`Pause()` → Pauses the playback
`Stop()` → Stops the recording or the playback
`Seek(float time)` → Seeks the track to the given time
`SeekToStart()` → Seeks to the start of the track
`SeekToEnd()` → Seeks to the end of the track

Status:
`IsPlaying()`
`IsPaused()`
`IsRecording()`

UnityEvents:
`onRecordStarted()`
`onRecordFinished()`
`onPlaybackStarted()`
`opPlaybackStopped()`
`onPlaybackFinished()`

Extending the Trackable component

To extend the system and implement your own recordable data you must extend the Trackable component. You may need to override some or all of the methods, it depends on the type of data you need to record. But in most cases you will be good to go overriding these:

`SetToIDLE()`
Here you should set all of the data to its initial states, like if it was just created

`PrepareForRecord()`
Here you should set all of the data to its initial states, just before the recording starts

`RecordAt(float time)`
Here, a new TrackDataEntry should be added to the Track at the given time

`PrepareForPlayback()`
Here you should set all of the data to its initial states, just before the replay starts

`PlaybackAt(float time)`
Here, the TrackDataEntry for the given time is used to restore the state of the Trackable entity

`Seek(float time)`
Here, the TrackDataEntry for the given time is used to restore the state of the Trackable entity