EntityRef Hijacking

Introduction

EntityRef *Hijacking* is the term used for a known side-effect the predict-rollback model can have in the view.

Description

There are several moving pieces playing a role in this:

- Predict-rollback;
- Entity creation; and,
- Entity view.

The simulation is constantly predicting the next couple of frames. ***Predicted frames*** use non-verified player input; once the server verified input for all players has been received, Quantum simulates a verified frame. Should a predicted frame required an *Entity A* to be created, it proceed with it and subsequently assigned it an "EntityRef X". If the `EntityViewAsset` associated with it have its `Bind Behaviour` set to **Non Verified** , the view element will be created at the very next `OnUpdateView` callback in Unity.

When receiving confirmed inputs from the server, Quantum re-simulates the frame with the server-side validated information - these are the **Verified** frames. Here is where things get interesting.

Should another *Entity B* have been created before *Entity A* in a re-simulated frame that a predicted frame had not, "EntityRef X" will be reassigned since EntityRefs are deterministic and given out sequentially. In this case, *Entity A* is assigned a new "EntityRef Y".

1. *Entity A* and *Entity B* each use a **different** `EntityViewAsset` . The `EntityViewUpdater` script will detect the mismatch, destroy the link it had for *Entity A* as well as its associated View. Once the clean-up finished, it will then create a new View for *Entity A* and *Entity B*.
2. *Entity A* and *Entity B* use the **same** `EntityViewAsset` . The `EntityViewUpdater` script will detect the mismatch in the next update, break the link between *Entity A* and its view, link *Entity B* to the existing view and instantiate a new View for *Entity A*.
    - *Entity A* has a new View. The view uses the correct data.
    - *Entity B* has a view previously associated with *Entity A*. The view was initialized with the wrong data, and therefore needs to be updated.

Note EntityViewUpdater: The `EntityViewUpdater` script saves the link between an entity in the simulation and its view by caching an EntityRef and EntityViewAsset key-pair value. See `EntityViewUpdater.cs` for more information.

Note Bind Behaviour: If an Entity View uses the **Verified** Bind Behaviour, it will only be instantiated on a verified frame and thus the above would never happen to it. It is a trade-off, the "types" of an entity dictate which behaviour is more suitable (see the *General Advice* section below).

How-to deal with it

This situation can be easily resolved by either tweaking the simulation or view.

In Code

To ensure entities are only created on verified frames, you could either wrap the following conditional statement:

C#

```
if(f.IsVerified) {
    // Do stuff
}
```

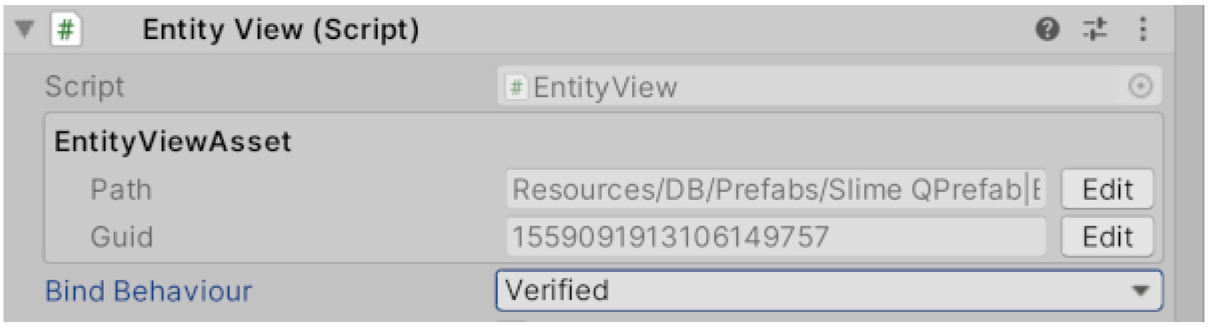
Or bail out early on non-verified, i.e. predicted, frames by using:

```
if (f.IsPredicted == false)
    return;
```

The API can also be used in `OnPlayerDataSet` if needed.

In Unity

In the view you can opt to either deal with the problem by modifying the initialization behaviour, or have the view update itself if it detects a mismatch in data it holds. The solution will depend on whether you set the Bind Behaviour to **Verified** or **Non-Verified**. To change the **Bind Behaviour** in Unity, simply navigate to the `Entity View` script located on the prefab associated with the `EntityViewAsset` and switch it from **Non Verified** to **Verified**.


Entity View (Script)
Script: EntityView
EntityViewAsset
Path: Resources/DB/Prefabs/Slime QPrefab|E
Guid: 1559091913106149757
Bind Behaviour: Verified

Editing the Bind Behaviour on the Entity View in Unity.

Bind Behaviour - Verified

In this case you are guaranteed to have the correct information from the get-go and you can trust the information available at the time of Unity's `Start()` method. It is also possible to use the `OnEntityInstantiated` EntityView event (see the Entity View script attached to the prefab).

Bind Behaviour - Non-Verified

In this case the information available to you at the time of instantiation may be erroneous.

To guarantee that the view will be able to reflect this change ensure the view can set itself from within the Unity `Update()` via polling. This way it will be able to switch its data if it happens to be wrong. Should the view be completely independent from the simulation and already hold of the

General Advice on Bind Behaviour

Both behaviours, **Verified** and **Non Verified**, have their advantages and draw-backs.

- `Verified` : Things that are not extremely sensitive to millisecond accurate instantiation, such as player characters.
- `Non Verified` : Things that are fast and players need time to react to, such as projectiles.

Back to top

Connect

Public Discord
YouTube
Facebook
Twitter
Blog
Contact Us

Languages

English
日本語
한국어
简体中文
繁体中文