Search

This document is about: **QUANTUM 2**

SWITCH TO

Baking Unity Animation Curves

Even though Quantum's `FPAnimationCurve` already does a conversion between Unity's non deterministic type under the hood through the curves editor, sometimes it might be useful to convert from an `AnimationCurve` when there is no automatic-in-editor conversion available.

One example is the conversion of curves which comes from Unity'a animation Clips, to create a deterministic version of them which could be used on the simulation.

Here are the snippets needed to create an `FPAnimationCurve` from an `AnimationCurve` :

C#

```csharp
public FPAnimationCurve ConvertAnimationCurve(AnimationCurve anim
{
    // Get UNITY keyframes
    Keyframe[] unityKeys = animationCurve.keys;

    // Prepare QUANTUM curves and keyframes to receive the info
    FPAnimationCurve fpCurve = new FPAnimationCurve();
    fpCurve.Keys = new FPAnimationCurve.Keyframe[unityKeys.Length

    // Get the Unity Start and End time for this specific curve
    float startTime = animationCurve.keys.Length == 0 ? 0.0f : fl
    float endTime = animationCurve.keys.Length == 0 ? 1.0f : floa

    // Set the resolution for the curve, which informs how detail
    fpCurve.Resolution = 32;

    for (int i = 0; i < unityKeys.Length; i++)
    {
        fpCurve.Keys[i].Time = FP.FromFloat_UNSAFE(unityKeys[i].t
        if (float.IsInfinity(unityKeys[i].inTangent) == false)
        {
            fpCurve.Keys[i].InTangent = FP.FromFloat_UNSAFE(unity
        }
        else
        {
            fpCurve.Keys[i].InTangent = FP.SmallestNonZero;
        }

        if (float.IsInfinity(unityKeys[i].outTangent) == false)
        {
            fpCurve.Keys[i].OutTangent = FP.FromFloat_UNSAFE(unit
        }
        else
        {
            fpCurve.Keys[i].OutTangent = FP.SmallestNonZero;
        }

        fpCurve.Keys[i].TangentModeLeft = (byte)AnimationUtility.
        fpCurve.Keys[i].TangentModeRight = (byte)AnimationUtility

        startTime = Mathf.Min(startTime, animationCurve[i].time);
        endTime = Mathf.Max(endTime, animationCurve[i].time);
    }

    fpCurve.StartTime = FP.FromFloat_UNSAFE(startTime);
    fpCurve.EndTime = FP.FromFloat_UNSAFE(endTime);

    fpCurve.PreWrapMode = (int)animationCurve.preWrapMode;
    fpCurve.PostWrapMode = (int)animationCurve.postWrapMode;

    // Actually save the many points of the unity curve into the
    SaveQuantumCurve(animationCurve, 32, ref fpCurve, startTime,
    return fpCurve;
}

private void SaveQuantumCurve(AnimationCurve animationCurve, int
        {
            if (resolution <= 0)
                return;
            fpCurve.Samples = new FP[resolution + 1];
            {
                var time = startTime + deltaTime * i;
                var fp = FP.FromFloat_UNSAFE(animationCurve.Evalu
                fpCurve.Samples[i].RawValue = fp.RawValue;
            }
        }
```

Back to top

We Make Multiplayer Simple

Products

Fusion

Quantum

Realtime

Chat

Voice

PUN

Memberships

Gaming Circle

Industries Circle

Support

Gaming Circle

Industries Circle

Circle Discord

Circle Stack Overflow

Connect

Documentation

Fusion

Quantum

Realtime

Chat

Voice

PUN

Bolt

Server

VR | AR | MR

Resources

Dashboard

Samples

SDK Downloads

Cloud Status

Languages

YouTube
Facebook
Twitter
Blog
Contact Us

日本語
한국어
简体中文
繁体中文

Terms Regulatory Privacy Policy Privacy Code of Conduct Cookie Settings