This page is a work in progress and could be pending updates.

Extending Assets for Unity

Overview

Quantum assets can be extended with Unity-specific data not relevant for the simulation like data for the UI (colors, texts, icons...). This is done with the use of *partial classes*.

Example

Let's take the `CharacterSpec` asset as an example. Its `ScriptableObject`-based wrapper in Unity is called `CharacterSpecAsset` and is the type which needs to extended.

C#

```
public partial class CharacterSpecAsset {
  [Header("Unity")]
  public Sprtie Icon;
  public Color Color;
```

photon

> These fields can only be accessed in the View (Unity) and cannot be accessed or used in the simulation (Quantum).

The newly created partial class needs to be added to the same assembly as the original definition of `CharacterSpecAsset`. By default, all Unity-side Quantum code belongs to the `PhotonQuantum` assembly.

To ensure the partial class belongs to the correct assembly use one of the following approaches:

- Save the class in `Assets/Photon/Quantum/User` directory.
- Save the class in any directory that has an *AssemblyDefinitionReference* asset pointing to the `PhotonQuantum` assembly.
- Delete `Assets/Photon/Quantum/PhotonQuantum.asmdef`. This will make Quantum a part of the main assembly. Note that this step needs to be repeated after each Quantum SDK update.

Access at Runtime

To access the extra fields at runtime, use the `UnityDB.FindAsset<T>()` method.

C#

```
CharacterSpecAsset characterSpecAsset = UnityDB.FindAsset<Charact
Debug.Log(characterSpecAsset.DisplayName);
```

Alternatively, the code-generated `GetUnityAsset()` extension methods can be used:

C#

```
CharacterSpec characterSpec = frame.FindAsset<CharacterSpec>(guid
CharacterSpecAsset characterSpecAsset = characterSpec.GetUnityAss
Debug.Log(characterSpecAsset.DisplayName);
```

appropriate method, as discussed here: *Resources, Addressables and Asset Bundles*.

Access at Edit-time

To load an asset using its path while in the Unity Editor, the `UnityEditor.AssetDataBase.LoadAssetAtPath<T>()` method can be used.

C#

```
CharacterSpecAsset characterSpecAsset = UnityEditor.AssetDatabase
Debug.Log(characterSpecAsset.DisplayName);
```

Alternatively, the asset can be loaded using its `AssetGuid` via the `UnityDB.FindAssetForInspector()` method and casting the result to the correct type.

C#

```
CharacterSpecAsset characterSpecAsset = (CharacterSpecAsset)Unity
Debug.Log(characterSpecAsset.DisplayName);
```

Back to top

photon

We Make Multiplayer Simple

Products

Fusion

Quantum

Realtime

Documentation

Fusion

Quantum

Realtime

photon

PUN

Memberships

Gaming Circle
Industries Circle

Support

Gaming Circle
Industries Circle
Circle Discord
Circle Stack Overflow

Connect

Public Discord
YouTube
Facebook
Twitter
Blog
Contact Us

PUN
Bolt
Server
VR | AR | MR

Resources

Dashboard
Samples
SDK Downloads
Cloud Status

Languages

English
日本語
한국어
简体中文
繁体中文

Terms Regulatory Privacy Policy Privacy Code of Conduct Cookie Settings