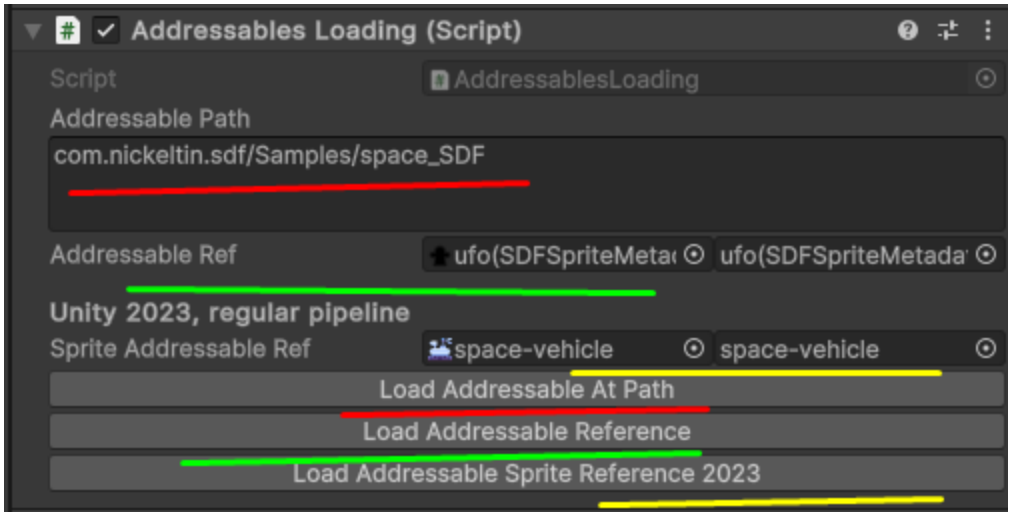
Addressables Support

Status	Up-to date

> ⚠ This sample is conditionally compiled, and only be fully functional if `com.unity.addressables` package is installed.

In 🎗 Runtime Sprite Change we covered loading sprite with `Resources.Load()`, but also its possible to use Addressables package to load sdf sprites.



Editor view

Decoupled pipeline

Decoupled pipeline designed mainly to allow for this kind of loading, with Resources or Addressables. That means in this section we will be loading `SDFSpriteMetadataAsset`'s from `SDFAsset`'s.

In unity 2023 it possible to load sprites from regular pipeline too, mentioned below.

Load Addressable At Path

This button will use *Addressable Path* `string` to load "***space_SDF.sdfasset***" sprite.

```
private void LoadAddressableAtPath()
{
    var sdf :AsyncOperationHandle<SDFSpriteMetadataAsset> = Addressables.LoadAssetAsync<SDFSpriteMetadataAsset>(_addressablePath);
    sdf.WaitForCompletion();
    Debug.Log( message: $"Addressable at {_addressablePath} loaded, status: {sdf.Status}, object: {sdf.Result}");
    SDFImage.SDFSpriteReference = new SDFSpriteReference(sdf.Result);
}
```

- Load addressable with static `Addressables.LoadAssetAsync()` call
  - in
    - addressable path, in this case **com.nickeltin.sdf/Samples/space_SDF**
  - return
    - Async operation that we complete synchronously
- Create `SDFSpriteReference` from loaded `SDFSpriteMetadataAsset`

> 💡 Advantages of loading by path is that you don't need to relay on serialized unity references. Useful if path to your icons comes from remote config file, for example, or you build addressables from other project.

Load Addressables Reference

This button is exactly the same, except it loads asset referenced with serialized `AssetReferenceT<SDFSpriteMetadataAsset>` - more reliable way to reference asset, since it

don’t care about addressable path and that it might change.

```
private void LoadAddressableReference()
{
    var sdf :AsyncOperationHandle<SDFSpriteMetadataAsset> =_addressableRef.LoadAssetAsync();
    sdf.WaitForCompletion();
    Debug.Log( message: $"Addressable with key {_addressableRef.RuntimeKey} loaded, sta
    SDFImage.SDFSpriteReference = new SDFSpriteReference(sdf.Result);
    _addressableRef.ReleaseAsset();
}
```

Process is pretty much the same as previously

- Create loading operation from reference
- Create `SDFSpriteReference` from loaded `SDFSpriteMetadatAsset`

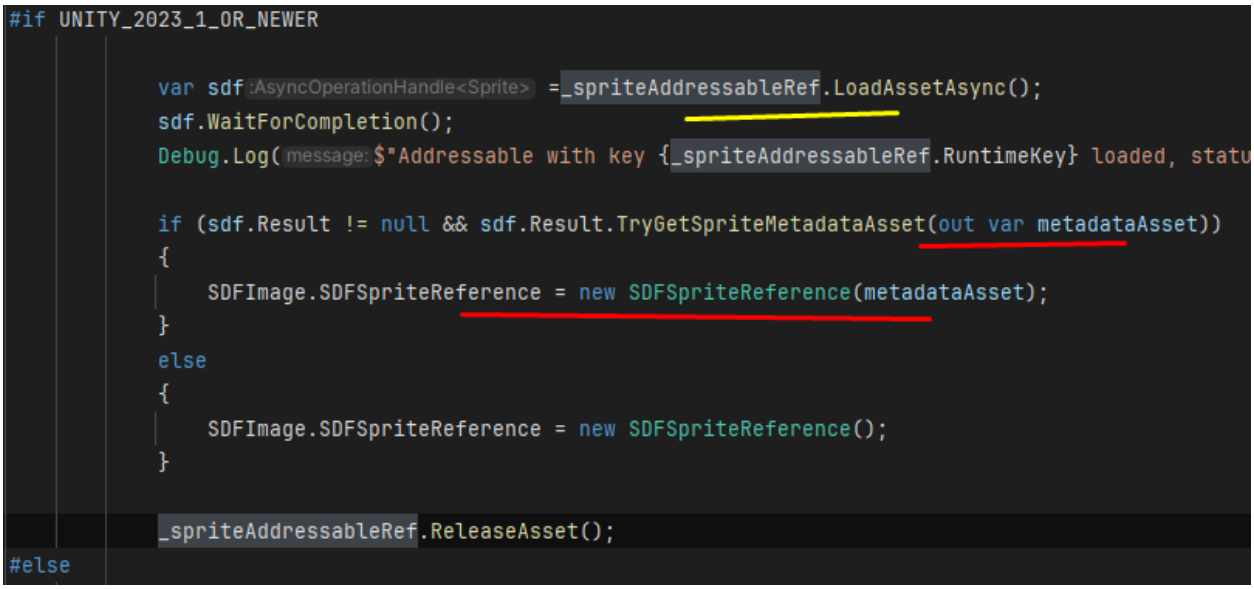
Regular pipeline

Only supported regular pipeline loading is in unity 2023. `SDFSpriteMetadatAsset`'s is hidden in regular pipeline, therefore can’t be loaded directly. However in unity 2023 you can load source sprite, and then extract `SDFSpriteMetadatAsset` from it.

Load Addressable Sprite Reference (Unity 2023)

This button will only function in unity 2023

```
[Header("Unity 2023, regular pipeline")]
[SerializeField] private AssetReferenceT<Sprite> _spriteAddressableRef;
```



```
#if UNITY_2023_1_OR_NEWER

        var sdf :AsyncOperationHandle<Sprite> =_spriteAddressableRef.LoadAssetAsync();
        sdf.WaitForCompletion();
        Debug.Log( message: $"Addressable with key {_spriteAddressableRef.RuntimeKey} loaded, statu

        if (sdf.Result != null && sdf.Result.TryGetSpriteMetadataAsset(out var metadataAsset))
        {
            SDFImage.SDFSpriteReference = new SDFSpriteReference(metadataAsset);
        }
        else
        {
            SDFImage.SDFSpriteReference = new SDFSpriteReference();
        }

        _spriteAddressableRef.ReleaseAsset();
#else
```

Process of loading addressable asset is the same, except were loading `Sprite` instead of `SDFSpriteMetadatAsset.`

- Create sprite loading operation from reference
- Try to get `SDFSpriteMetadatAsset` from sprite
- Create `SDFSpriteReference` from loaded `SDFSpriteMetadatAsset`

Notes

> 💡 All assets that is used as adressables is added to your default addressables settings trough script, ensuring that sample will work right away in any project with addressables. However something might go wrong and assets in folder **Scripting\4_Addressables Support\Sprites\Addressables** will not be addressable.