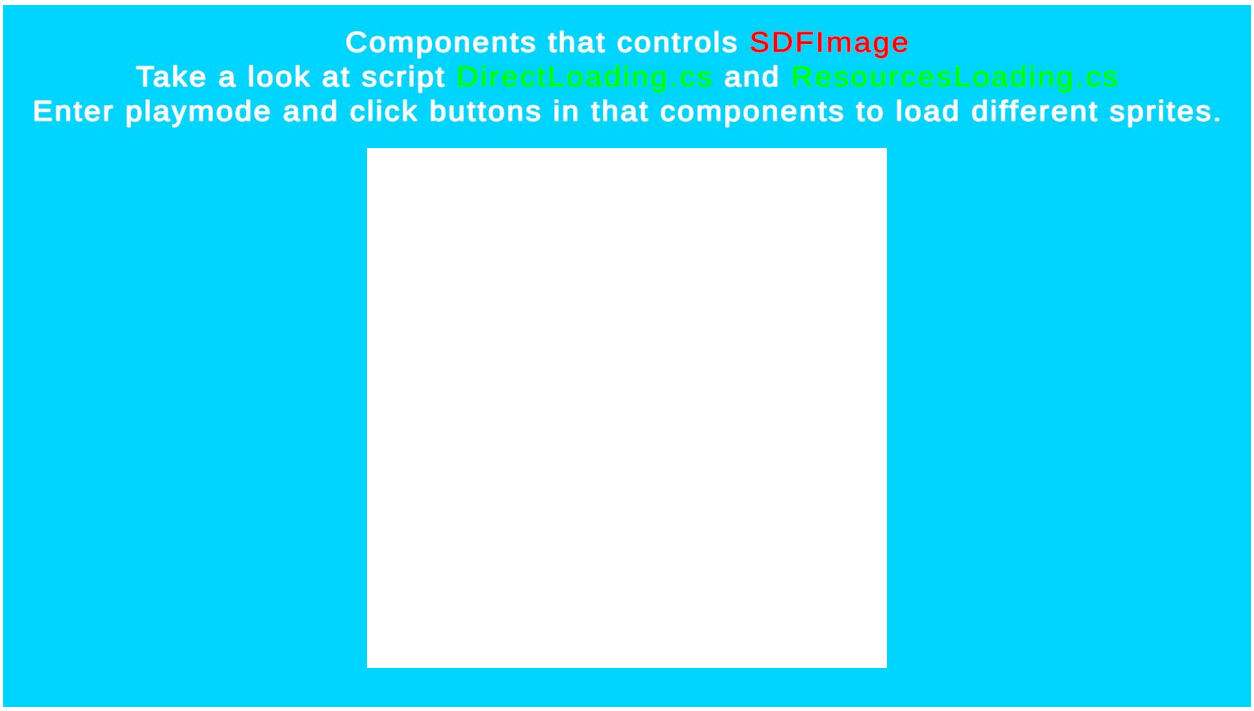
Runtime Sprite Change

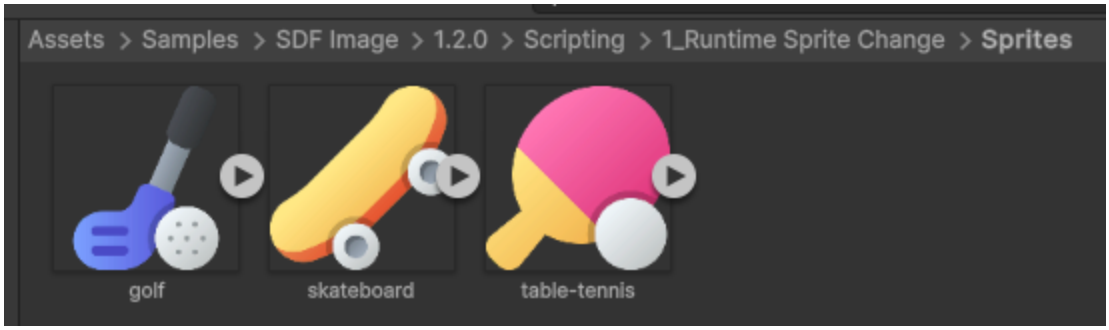
Status	Up-to date

Sample files overview



Scene view with empty SDFImage, you will be changing its sprite throughout sample

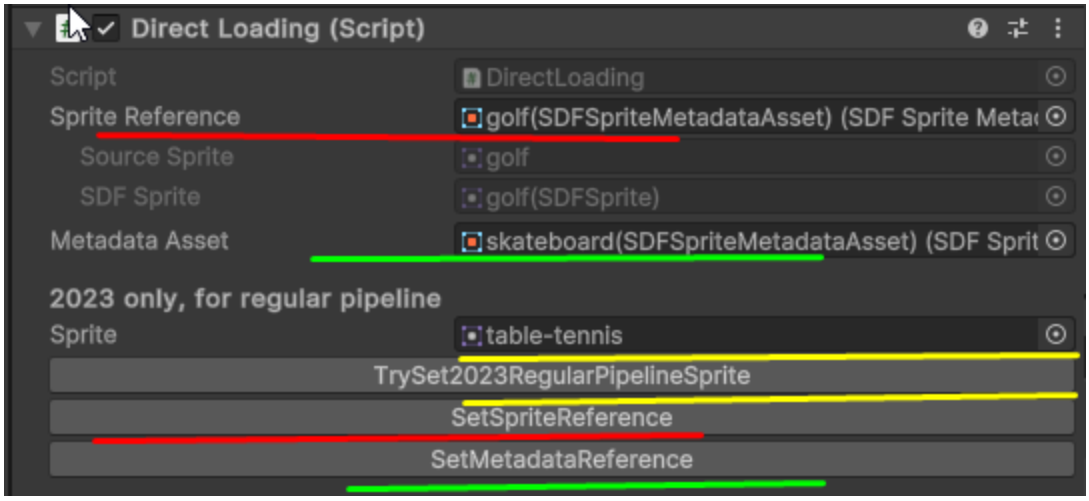
Resources loading section for both regular and decoupled pipeline



Sprites for direct loading

Direct loading

Shows how to set sprite to SDFImage as serialized field, there is a few kinds though.



Editor view of loading options

Set `SDFSpriteReference`

Most simple sprite setting way.

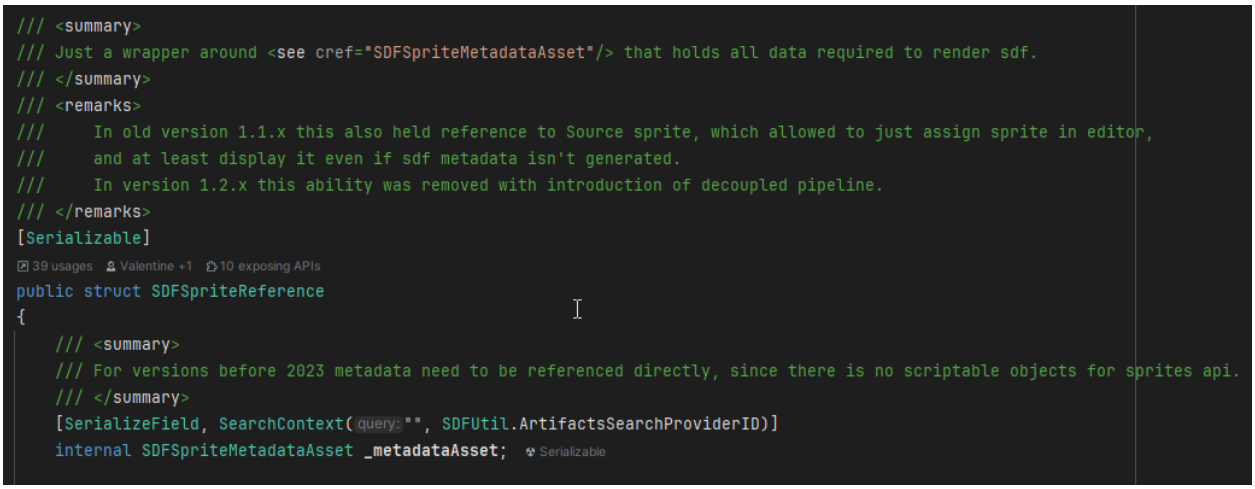
```
[SerializeField] private SDFSpriteReference _spriteReference;
```

Field

```
private void SetSpriteReference()
{
    SDFImage.SDFSpriteReference = _spriteReference;
}
```

Setting

This how you change sprites at runtime, by settings struct `SDFSpriteReference` to corresponding property.



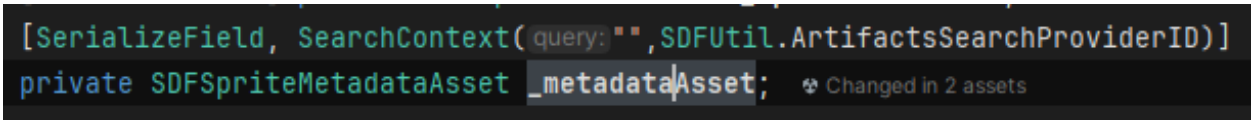
Docs with detailed explanation on why this struct exist

`SDFSpriteReference` is exist for multi-version compatibility, in versions before 1.2.x it was holding more properties, and handled some custom serialization, but right now its just an wrapper around `SDFSpriteMetadataAsset`.

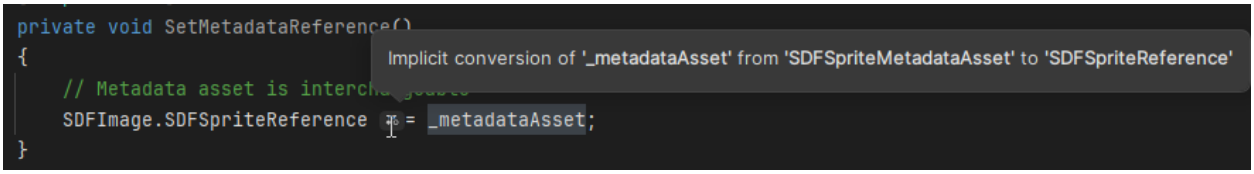
It will be kept in future versions as its useful to have some intermediate serialization layer, instead of plain object reference.

Set `SDFSpriteMetadataAsset`

`SDFSpriteMetadataAsset` was previously internal, but now is exposed to public, therefore you can use it directly as object field.



Field



Setting

Why is there additional attribute?

SearchContextAttribute is part of new UnityEngine.Search API, you can remove it and field will work perfectly fine, however it will only display `SDFSpriteMetadataAsset`'s from decoupled pipeline.

This happens due to `SDFSpriteMetadataAsset`'s being hidden in regular pipeline in editor, and default object picker can't display hidden assets. In decoupled pipeline they are, however, is visible.

> 💡 **Conclusion:** referencing `SDFSpriteMetadataAsset` is possible but with some nuances, learn more about them Decoupled Pipeline Scripting

Set sprite directly (Unity 2023)

There is another way to set sprite in unity 2023 with the introduction of Scriptable Objects for Sprites API. This way you can extract `SDFSpriteMetadataAsset` directly from sprite.

```
[Header("2023 only, for regular pipeline")]
[SerializeField] private Sprite _sprite;  Serializable
```

Field

```
        private void TrySet2023RegularPipelineSprite()
        {
#if UNITY_2023_1_OR_NEWER
            var found:bool = _sprite.TryGetSpriteMetadataAsset(out var metadata);
            Debug.Log(message:$"Sprite {_sprite}, metadata asset found: {found}, metadata: {metadata}");
            SDFImage.SDFSpriteReference = metadata;
#else
            Debug.LogError($"Unity before 2023 can't get sdf metadata directly from sprite");
#endif
        }
```

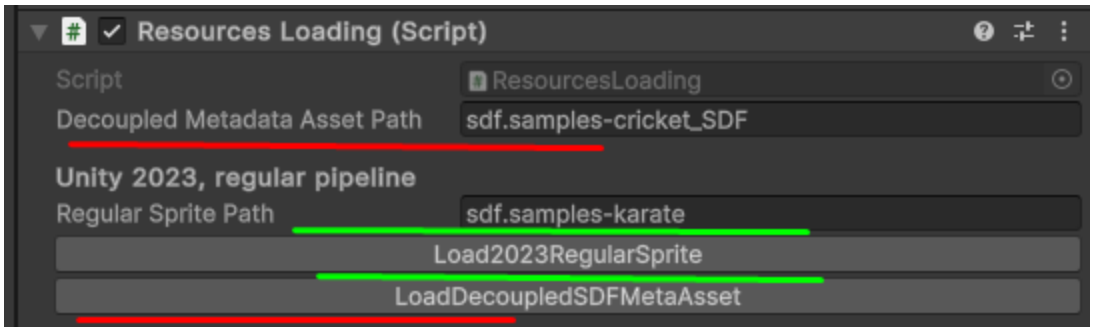
Setting

For unity 2023 `SDFUtil` class has some new public functions, the one use here is `TryGetSpriteMetadataAsset`. After getting meta asset it just assigned as in previous sections.

💡 See more about 2023 features  Unity 2023 Support

Resources loading

It possible to load sprites as resources, however its only possible for decoupled pipeline or in unity 2023.



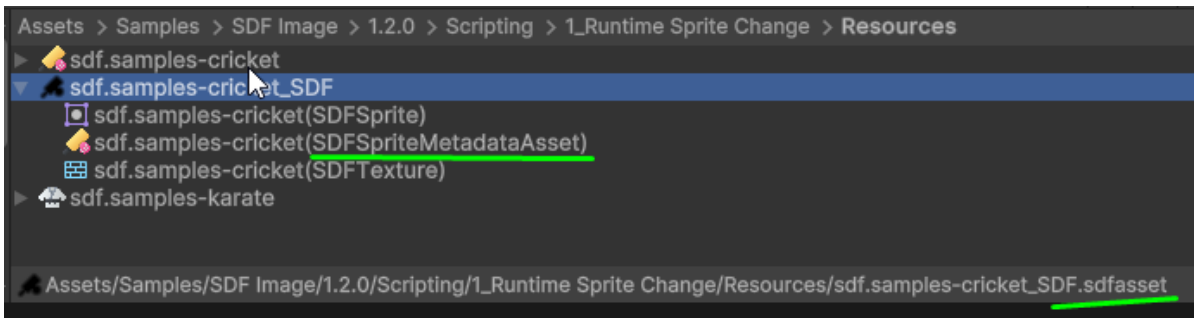
Editor view of loading options

Loading sprite from decoupled pipeline

- Locating `SDFAsset`

```
[SerializeField] private string _decoupledMetadataAssetPath;
```

Field with just a resources path to required SDFAsset



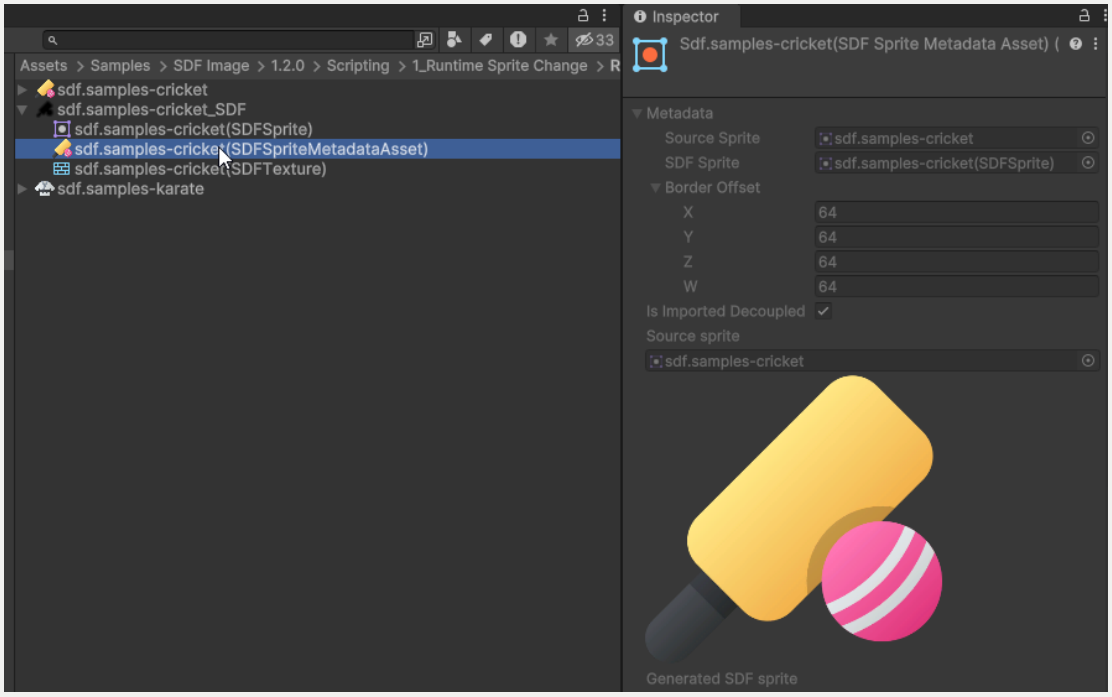
Path points to this SDFAsset

- Loading as resource

```
private void LoadDecoupledSDFMetaAsset()
{
    // For decoupled pipeline can load directly
    // For regular pipeline resources loading is not possible unless in unity 2023
    // due to SDFSpriteMetadataAsset's is hidden
    var metaAsset = Resources.Load<SDFSpriteMetadataAsset>(_decoupledMetadataAssetPath);
    SDFImage.SDFSpriteReference = metaAsset;
    Debug.Log(metaAsset);
}
```

This is regular procedure of loading an asset as resource. Then loaded `TryGetSpriteMetadataAsset` is assigned as in previous sections.

Loading as resources is possible due to `TryGetSpriteMetadataAsset` is visible in editor in `SDFAsset`. For regular pipeline they are hidden.



Load sprite directly (Unity 2023)

As we was extracting `TryGetSpriteMetadataAsset` from sprite it is possible to just load the sprite as resource, and then do the same.

```
        private void Load2023RegularSprite()
        {
            var sprite = Resources.Load<Sprite>(_regularSpritePath);
            Debug.Log(sprite);
#if UNITY_2023_1_OR_NEWER
            var found :bool = sprite.TryGetSpriteMetadataAsset(out var metadata);
            Debug.Log( message: $"Sprite {sprite}, metadata asset found: {found}, metadata: {metadata}");
            SDFImage.SDFSpriteReference = metadata;
#else
            Debug.LogError($"Unity before 2023 can't get sdf metadata directly from sprite");
#endif
        }
```

Loading regular sprite as resource, then extracting meta asset with 2023 API

Limitations

> ⚠ 2023 features won't work for source sprites of decoupled pipeline, see more in Unity 2023 Support and Decoupled Pipeline Scripting