Architecture Map

for World Builder VR

This document contains diagrams and descriptions for the 2 different core architectures, regarding handling of Assets. The first one being the fully sustainable architecture, allowing to CRUD asset in-app (used by World Assets), while the second one requires the use of Unity Engine, due to it using ScriptableObjects.

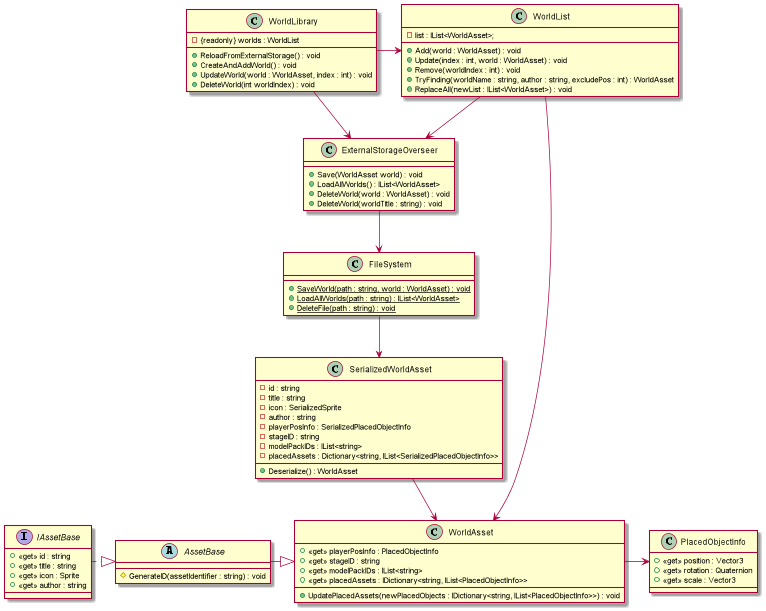
Obsah

[1. Standalone Asset Architecture 3](#_Toc90836563)

[2. Depended Asset Architecture 4](#_Toc90836564)

# Standalone Asset Architecture

This is the architecture of handling World Assets in the app. They are not tied to working with the engine and are fully independent from it.



**IAssetBase** is a base for all assets in the app.

**AssetBase** is a base for all assets, that work independently of Unity Engine (not ScriptableObjects).

**PlacedObjectInfo** stores all relevant positional values about a give GameObject.

**WorldAsset** is the main class representing Worlds. It stores all the necessary data a World would need to properly function.

**WorldLibrary** contains a list of all the Worlds currently loaded by the app. This is also the place to add, update and remove worlds.

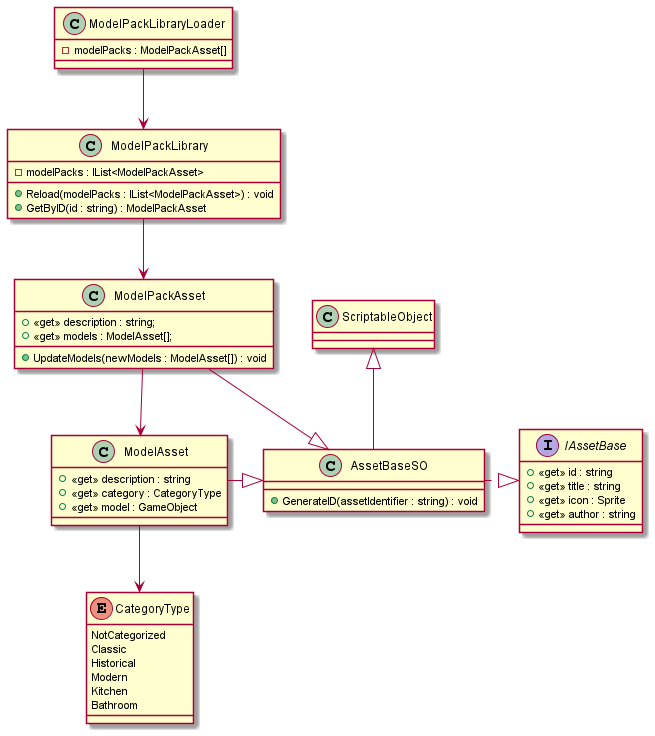
**WorldList** is a variation of a normal List, where adding/removing files is synced with external storage. It also has searching capabilities and is the storing unit for the Worlds.

**ExtrernalStorageOverseer** prepares Worlds for proper saving, loading, and removing from the correct path on external storage and is the place to call for doing such actions.

**FileSystem** is a static class that saves, loads, and removes Worlds/Files from a given path on external storage.

**SerializedWorldAsset** is the WorldAsset, except in a format, that can be serialized into a binary file. Its constructor converts a normal WorldAsset into itself and can be later deserialized.

# Depended Asset Architecture



**IAssetBase** is a base for all assets in the app.

**AssetBaseSO** is a base for all assets, that dependent on Unity Engine. That’s where they have to be created, updated and deleted. They are also all ScriptableObjects.

**CategoryType** is an enum storing all the possible categories a model can have.

**ModelAsset** is the main class representing Models. It stores all the necessary data a Model needs to properly operate.

**ModelPackAsset** is the main class representing Model Packs. It stores all the necessary data a Model Pack needs to properly operate and also an array of models it contains.

**ModelPackLibrary** contains a list of all the Model Packs currently loaded by the app. This is the place to get access to them.

**ModelPackLibraryLoader** has only 1 job and that is to load the ModelPackLibrary with Model Packs. These are set through the inspector.