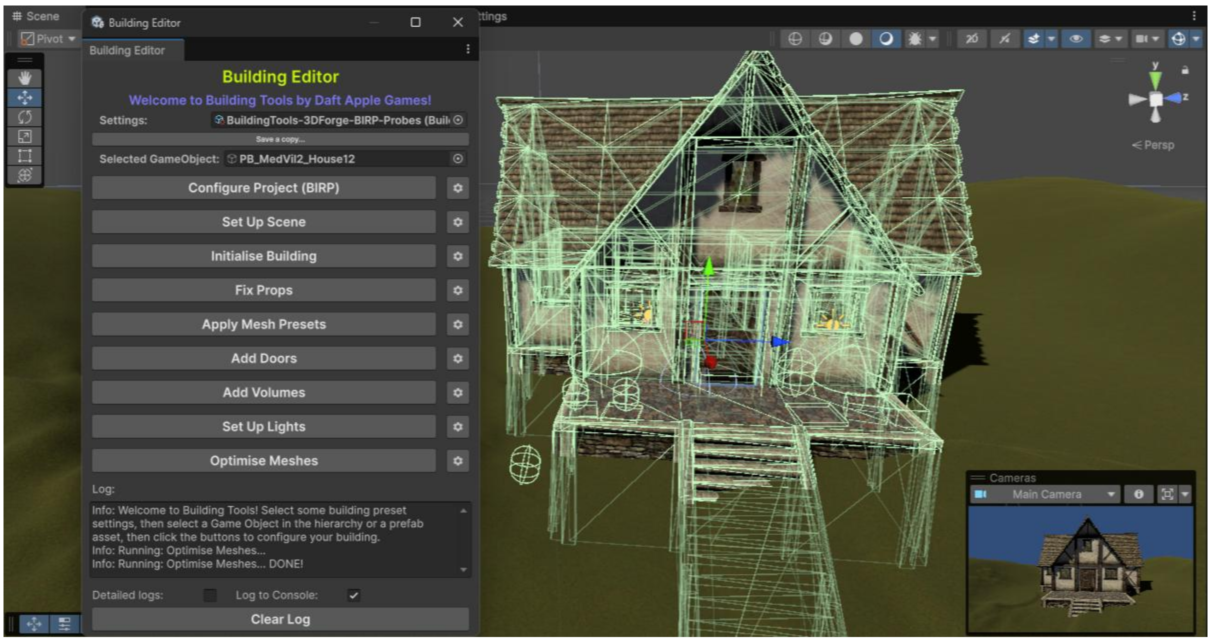
Daft Apple Games

Unity Building Tools



Settings: BuildingTools-3DForge-BIRP-Probes (Buil

Selected GameObject: PB_MedVil2_House12

Log:

```
Info: Welcome to Building Tools! Select some building preset
settings, then select a Game Object in the hierarchy or a prefab
asset, then click the buttons to configure your building.
Info: Running: Optimise Meshes...
Info: Running: Optimise Meshes... DONE!
```

Detailed logs: ☐ Log to Console: ✓

Offline Documentation

Latest documentation can always be found here:

https://mroshaw.github.io/unity-framework-docs/buildingtools/

Contents

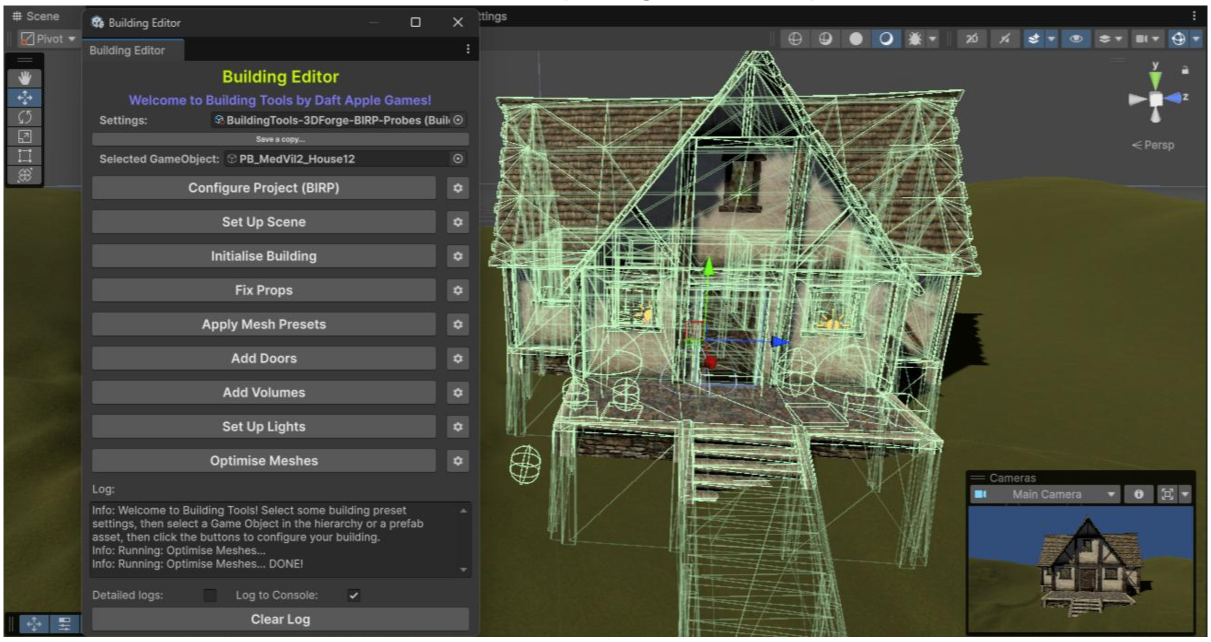
About Daft Apple Building Tools

Daft Apple Building Tools provides editor windows, editor tools, and run-time components to allow quick, consistent configuration of building assets in a Unity project. Features include consistent configuration of mesh renders, setting up efficient and effective lighting, fixing colliders and aligning props on the terrain, automatically adding working doors, and mesh optimisation. The package comes with pre-sets and examples for configuring the excellent 3D Forge village assets with specific presets for each of the 3 Unity render pipelines. Everything is extensible and configurable and you are encouraged to create your own presets and make the most out of the tools for your particular project and use cases.

You can find a short video showcasing the main features of the package on my YouTube channel.

Important

Third party assets are not required to use the tools in this package. No unlicensed 3rd party assets whatsoever are included in either the package or the samples.



Settings: BuildingTools-3DForge-BIRP-Probes (Buil

Selected GameObject: PB_MedVil2_House12

Log:

```
Info: Welcome to Building Tools! Select some building preset
settings, then select a Game Object in the hierarchy or a prefab
asset, then click the buttons to configure your building.
Info: Running: Optimise Meshes...
Info: Running: Optimise Meshes... DONE!
```

Detailed logs: ☐ Log to Console: ✓

Important

Building Tools is supported in **Unity 6 LTS ONLY**! A number of useful changes came into Unity 6 and I’ve made use of these where possible in this package. I have tested in Unity 6 ONLY and would expect compilation errors and general failures if used with an earlier version of Unity.

Support

Limited support for these tools will be provided via Issues in the GitHub repository.

Important

This package, and Daft Apple Games, are in no way affiliated with 3D Forge. Please do not ask them for support for these tools!

About Daft Apple Games

I am a hobby game developer who just loves to code! All of my game dev stuff I do for fun, and I like to give back to the community who helped me grow my knowledge and experience of the art of game development. I dabble in Unity, and have released one full game with another in active development.

Daft Apple
Games

Daft Apple Games Design Philosophy

The packages that I build are always designed to be as useful and innocuous as possible. I aim to follow these core principles in all the tools and code that I publish:

- All of my code and packages are free and open source. Everything is free for you to use, change, make your own, do anything you like.
- All Run Time components are granular, optional and non-intrusive. Nothing is enforced by a package, and your hierarchy, project, and scene setup is not tampered with in any way unless it's specifically a documented function of a tool or component.
- Any project level changes made by the tools are documented and reversible and detailed uninstallation instructions are always provided.
- I try to keep my code as clean and as error and warning free as possible, and I follow best practice principles as best I can. I employ Rider and SonarQube code quality checking to ensure optimum code and avoid "code smells".
- Assembly definitions are used where appropriate, to keep code clean, decoupled, safe, and ensure your project compile times are kept down.
- All Editor specific script is contained within "Editor Only" assembly definitions. My packages won't add unnecessary code to your game builds and you won't find my tools taking up valuable time in your update loop.

Package Overview

Building Tools is in active development, completely free, and open source. I'm tracking features and bugs in a GitHub project KanBan, so you can see exactly what's in the pipeline by checking out the Building Tools Project Page.

Core Functions

The latest version of Building Tools provides these functions:

- **Initialise Building**
  - Adds a new Building component, providing core component functionality to your building.
  - Shifts the pivot of the building so that it is easier to place on an uneven surface, such as a terrain.
- **Configure Meshes**
  - Applies configurable MeshRenderer preset properties to all Exterior, Interior and Props meshes.
    - For example, presets can be created for "Lightmapped Meshes" with properties set for consistent lighting to be applied to all "Outdoor Props"
- **Configure Props**
  - Adds any missing Colliders to props, such as barrels, crates, carpets, chairs, tables, etc.
  - Aligns any outdoor props to the terrain, such as barrels, stables, and fences.
- **Configure Doors**
  - Adds a Door Controller component and configures Door components on all door mesh games objects, giving you lots of control over the state of doors within a building.
  - Adds Door Trigger components that open and close doors when a player enters the trigger, plays audio effects, and provides Unity Events to hook in your own functionality.
- **Configure Lighting**
  - Adds a BuildingLightManager game object to your active scene and adds a BuildingLightController component and configures BuildingLight components on all lights within your building Game Object. This gives you loads of control over light states, with specific public methods for turning on and off candles, fires, and outdoor lights.
  - Preconfigure light properties, allowing you to apply consistent preset light properties (think intensity, range, temperature etc), across all similar light types.
  - Replaces unsupported LensFlare components with SRPLensFlare components. (**URP and HDRP only**)

    - Optionally adds a OnDemandShadowMapUpdate component, allowing you to fine tune shadow updates on individual lights, offering significant performance gains. (**HDRP only**)
    - Samples include a very simple TimeOfDay component, demonstrating how lights can be turned on and off as time progresses.
- **Configure Volumes**
    - Adds a lighting Volume with an "interior volume profile", allowing you to fine tune lighting within the bounds of your building. (**URP and HDRP only**)
    - Adds an InteriorAudioFilter component that allows you to "muffle" ambient / outdoor audio when the player is in your building.
    - Components provide UnityEvents that you can hook into to trigger your own functionality when a player or character enters and leaves the building.
- **Mesh Optimisation**
    - Combines exterior, interior and prop meshes into merge meshes within your building. Greatly reduces draw calls and shadow casters and can offer huge performance gains.

Samples

The package comes with options samples, with preconfigured settings and sample assets to get you started. Included are some presets for working with 3D Forge Village Interiors and Exteriors assets, which is my primary use case when developing these tools.

Support and How to Contribute

Contributing to the Repository

The source code for this package is completely open source and is made available under the MIT License. Please feel free to fork the code, make changes, and contribute any changes via a Pull Request.

Please ensure any code changes are aligned to the coding style, standards and architecture of the project. All PRs will be carefully reviewed and only merge if they provide benefit while maintaining the code quality and standards set in the baseline code.

Getting Support

All issues and suggestions should be submitted via the Issues tab in the GitHub repository. Please take time to give issues a descriptive title and as detailed a description as you possible can. Please include these details at a minimum:

- Full Version of Unity (e.g. 6000.0.47f1)
- Render Pipeline (e.g. HDRP, URP, Built In)
- For issues, include:
    - Numbered steps to reproduce.
    - Expected outcome.
    - Actual outcome.
- For suggestions, please include:
    - Description of the function or change.
    - Details of the benefit this will bring to the package.

Important

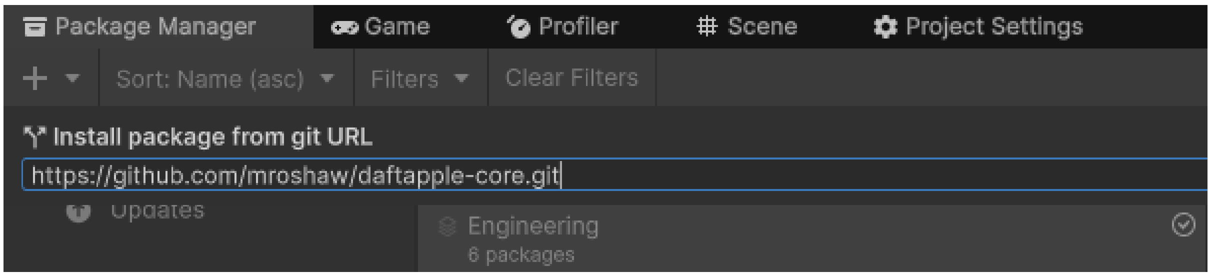
Any ticket raised that does not include the minimum required info will be closed!

Installation

Package Package and Dependencies

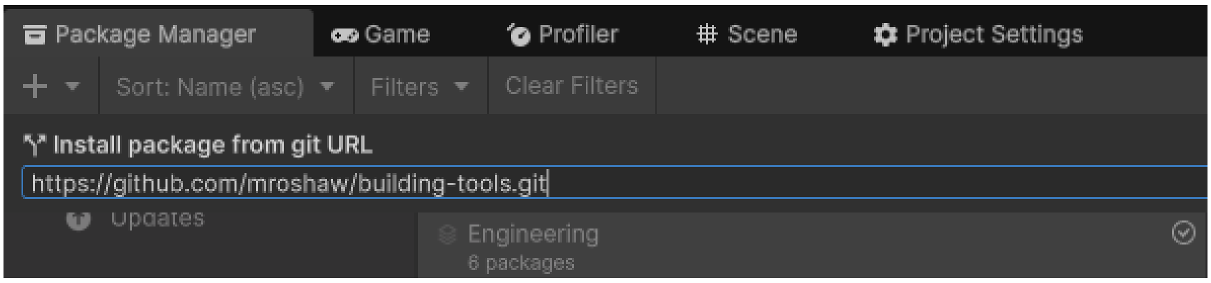
The package relies on a number of scripts and resources in the “daftapple-core” package. You must install this first through package manager:

1. Open the package manager window.
2. From the menu in the top left, pick “Install package from git URL…”
3. Copy and paste this URL: https://github.com/mroshaw/daftapple-core.git



https://github.com/mroshaw/daftapple-core.git

4. Click install.
5. You can now install the “building-tools” package by following the same process with this URL: https://github.com/mroshaw/building-tools.git



https://github.com/mroshaw/building-tools.git

Project Settings

In order to implement some of the functionality, the package requires specific Layers and RenderingLayers to be present in your project settings.

You can either add these manually, or using the “Set up” button in the main editor window. Note that this automated processed will look for available slots for tags and layers and won’t overwrite what’s there. Rendering Layer names are, however, overwritten.

To configure these manually, go to Edit > Project Settings > Tags and Layers…

Add the following Layers - it doesn’t matter at what index they are added:

- BuildingExterior
- BuildingInterior
- ExteriorProps
- InteriorProps

Update the following Rendering Layers, setting the name for the given layer index:

- 1 - External
- 2 - Internal

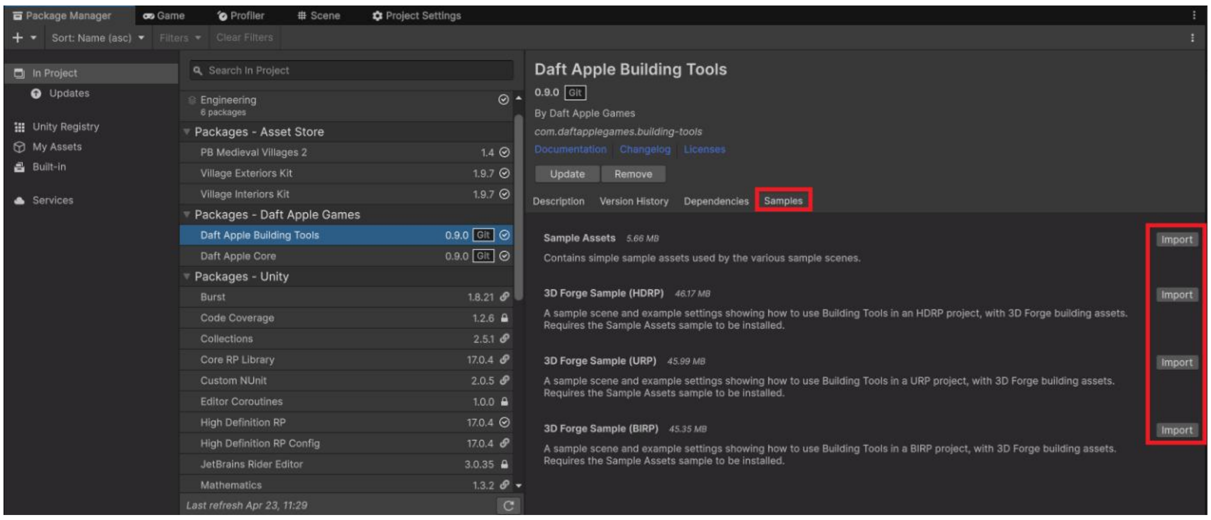
Install Samples

The package comes with a sample folder contain some example assets, such as door sounds and an AudioMixer, and presets for the "3D Forge" building assets. Due to the way lighting and meshes work in Unity, there are sample packages for each of the three render pipelines, and you should import the one relevant to your project. The render pipeline specific packages rely on shared assets from the "Sample Assets" package, so please install this first.

Important

No 3rd party assets whatsoever are included in the samples. They provide a staging scene, with a terrain and pipeline configured lighting, a simple "Fly Cam" attached to the camera, and some sample ambient audio and effects. They include a set of "example" tools configurations to help you get started, setup to work with 3D Forge building prefabs. You can using the staging scene to drag in your buildings and experiment with the tools and settings.

You can install the samples via the Package Manager by selecting the "Daft Apple Building Tools" package, clicking "Samples" and clicking the "Import" button next to the packages you want to install:



Engineering 6 packages	
Packages - Asset Store	
PB Medieval Villages 2	1.4
Village Exteriors Kit	1.9.7
Village Interiors Kit	1.9.7
Packages - Daft Apple Games	
Daft Apple Building Tools	0.9.0 Git
Daft Apple Core	0.9.0 Git
Packages - Unity	
Burst	1.8.21
Code Coverage	1.2.6
Collections	2.5.1
Core RP Library	17.0.4
Custom NUnit	2.0.5
Editor Coroutines	1.0.0
High Definition RP	17.0.4
High Definition RP Config	17.0.4
JetBrains Rider Editor	3.0.35
Mathematics	1.3.2

Last refresh Apr 23, 11:29

Daft Apple Building Tools
0.9.0 Git
By Daft Apple Games
com.daftapplegames.building-tools
Documentation | Changelog | Licenses

Sample Assets *5.66 MB*
Contains simple sample assets used by the various sample scenes.

3D Forge Sample (HDRP) *46.17 MB*
A sample scene and example settings showing how to use Building Tools in an HDRP project, with 3D Forge building assets. Requires the Sample Assets sample to be installed.

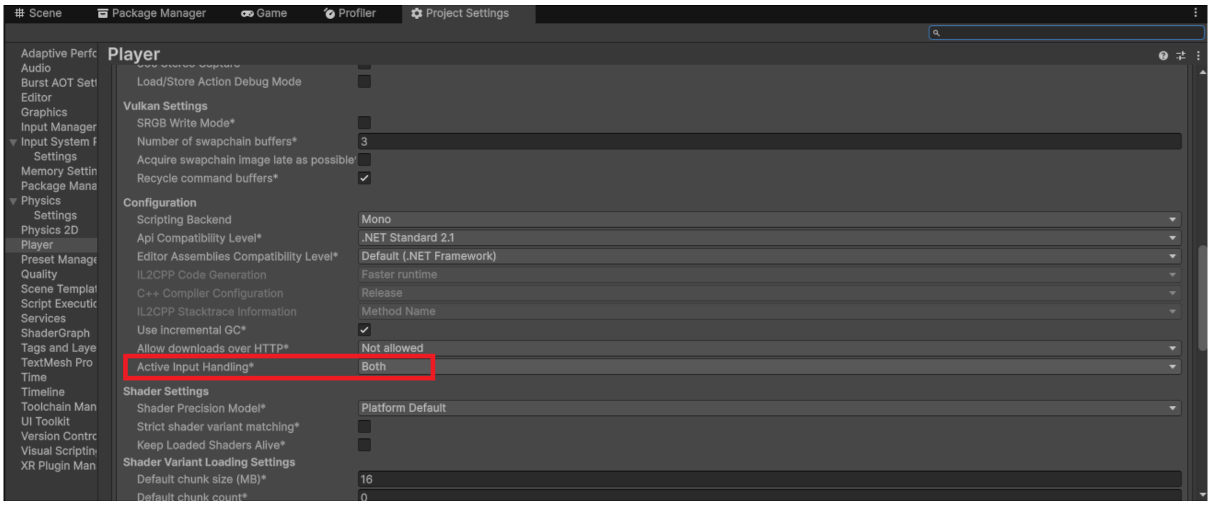
3D Forge Sample (URP) *45.99 MB*
A sample scene and example settings showing how to use Building Tools in a URP project, with 3D Forge building assets. Requires the Sample Assets sample to be installed.

3D Forge Sample (BIRP) *45.35 MB*
A sample scene and example settings showing how to use Building Tools in a BIRP project, with 3D Forge building assets. Requires the Sample Assets sample to be installed.

To use the SimpleFlyCamera component in the samples, you'll need to set the project player input settings to handle both the "old" and "new" input systems:

Project Settings > Player > Configuration

Set "Active Input Handling" to "Both":



Player

Load/Store Action Debug Mode

Vulkan Settings

SRGB Write Mode*

Number of swapchain buffers* 3

Acquire swapchain image late as possible*

Recycle command buffers* ✓

Configuration

Scripting Backend	Mono
Api Compatibility Level*	.NET Standard 2.1
Editor Assemblies Compatibility Level*	Default (.NET Framework)
IL2CPP Code Generation	Faster runtime
C++ Compiler Configuration	Release
IL2CPP Stacktrace Information	Method Name
Use incremental GC*	✓
Allow downloads over HTTP*	Not allowed
Active Input Handling*	Both

Shader Settings

Shader Precision Model*	Platform Default
Strict shader variant matching*	
Keep Loaded Shaders Alive*	

Shader Variant Loading Settings

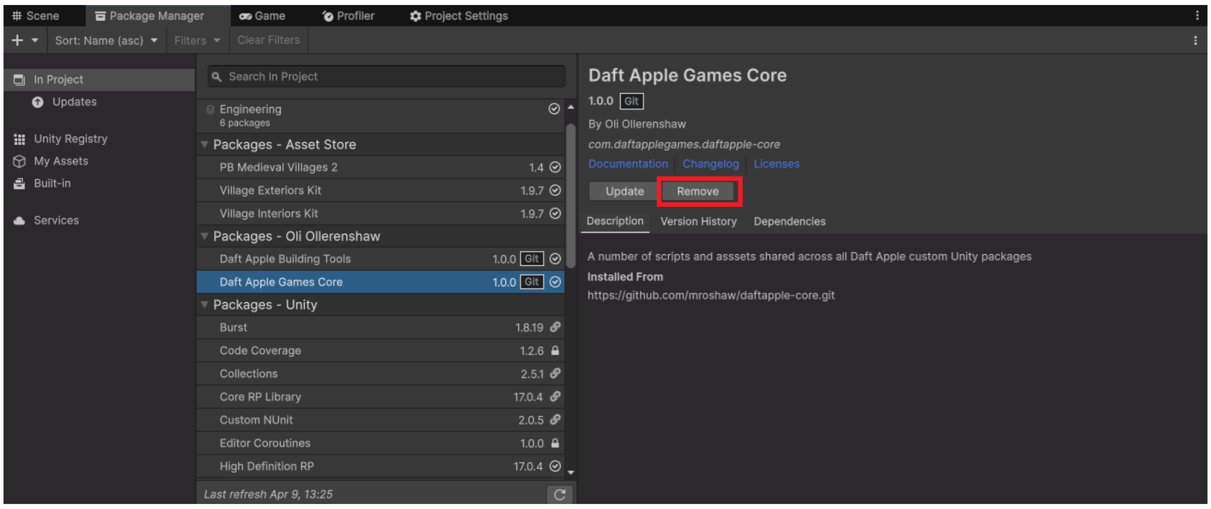
Default chunk size (MB)*	16
Default chunk count*	0

Uninstalling

A small number of steps are required to completely uninstall the Building Tools package.

Remove the packages

You can uninstall the packages by selecting the package in Package Manager, under “In Project”, and clicking the “Remove” button. You should remove the Building Tools package first, then the Core package:



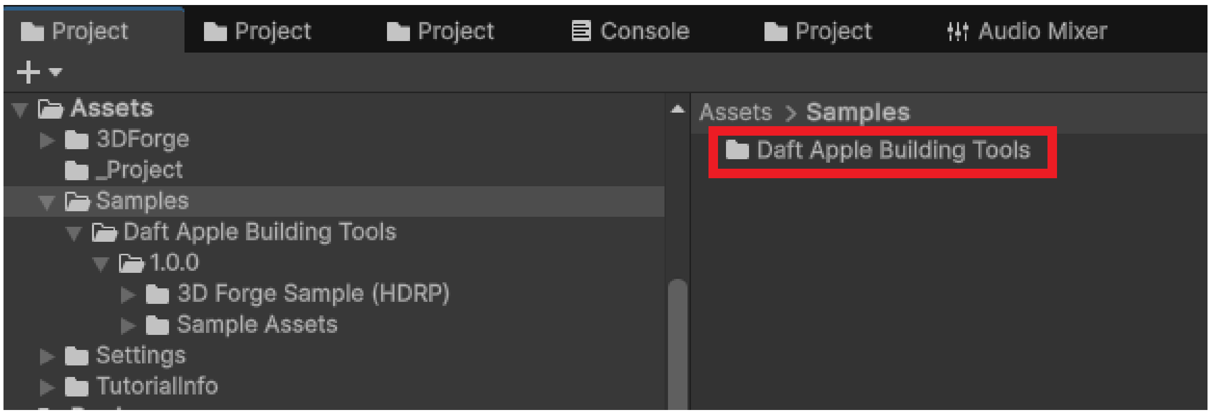
Engineering 8 packages	
Packages - Asset Store	
PB Medieval Villages 2	1.4
Village Exteriors Kit	1.9.7
Village Interiors Kit	1.9.7
Packages - Oli Ollerenshaw	
Daft Apple Building Tools	1.0.0 Git
Daft Apple Games Core	1.0.0 Git
Packages - Unity	
Burst	1.8.19
Code Coverage	1.2.6
Collections	2.5.1
Core RP Library	17.0.4
Custom NUnit	2.0.5
Editor Coroutines	1.0.0
High Definition RP	17.0.4

Daft Apple Games Core
1.0.0 Git
By Oli Ollerenshaw
com.daftapplegames.daftapple-core
Documentation Changelog Licenses

A number of scripts and assets shared across all Daft Apple custom Unity packages
Installed From
https://github.com/mroshaw/daftapple-core.git

Delete any installed samples

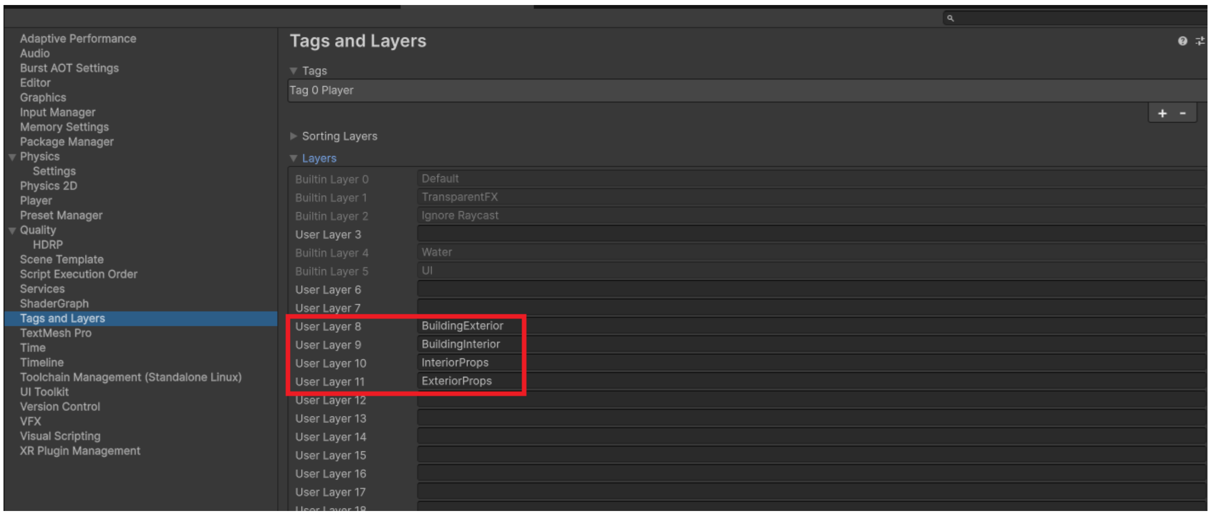
You can very easily remove the samples by simply deleting the folder that is automatically created by the package manager:



Restore Layers and Rendering Layers

Layers

If you clicked the “Set Up” button, a number of Layers are created. You can undo this by going to your “Project Settings > Tags and Layers” and removing / updating the entries:



Adaptive Performance
Audio
Burst AOT Settings
Editor
Graphics
Input Manager
Memory Settings
Package Manager
Physics
Settings
Physics 2D
Player
Preset Manager
Quality
HDRP
Scene Template
Script Execution Order
Services
ShaderGraph
Tags and Layers
TextMesh Pro
Time
Timeline
Toolchain Management (Standalone Linux)
UI Toolkit
Version Control
VFX
Visual Scripting
XR Plugin Management

Tags and Layers

Tags

Tag 0 Player

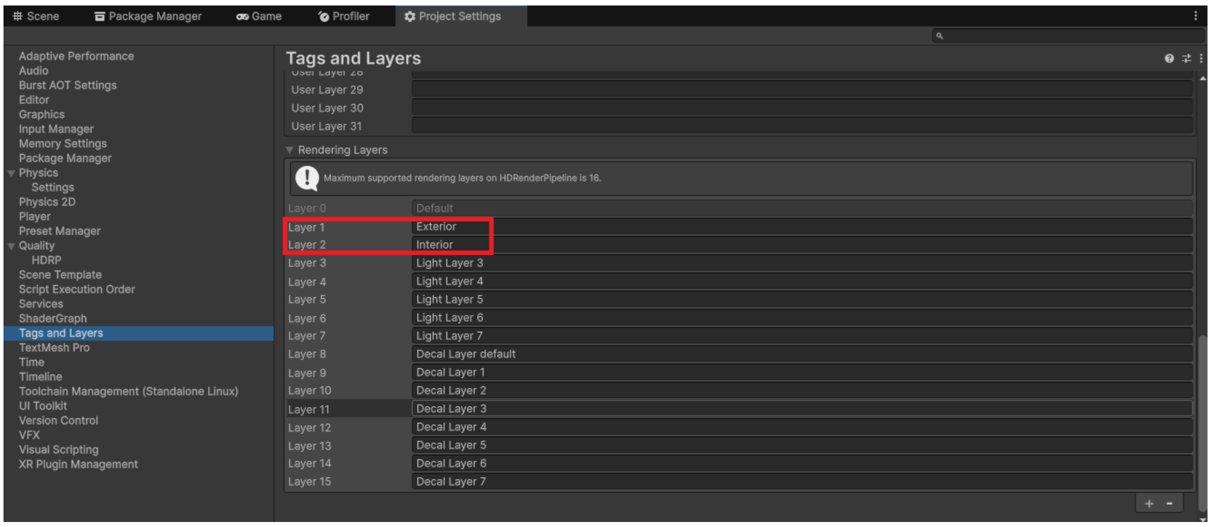
Sorting Layers

Layers

Builtin Layer 0	Default
Builtin Layer 1	TransparentFX
Builtin Layer 2	Ignore Raycast
User Layer 3	
Builtin Layer 4	Water
Builtin Layer 5	UI
User Layer 6	
User Layer 7	
User Layer 8	BuildingExterior
User Layer 9	BuildingInterior
User Layer 10	InteriorProps
User Layer 11	ExteriorProps
User Layer 12	
User Layer 13	
User Layer 14	
User Layer 15	
User Layer 16	
User Layer 17	

Rendering Layers

Again, only if you clicked "Set Up", two of the Rendering Layers have been renamed. You can undo this via "Project Settings > Tags and Layers":



Scene Package Manager Game Profiler Project Settings

Adaptive Performance
Audio
Burst AOT Settings
Editor
Graphics
Input Manager
Memory Settings
Package Manager
Physics
Settings
Physics 2D
Player
Preset Manager
Quality
HDRP
Scene Template
Script Execution Order
Services
ShaderGraph
Tags and Layers
TextMesh Pro
Time
Timeline
Toolchain Management (Standalone Linux)
UI Toolkit
Version Control
VFX
Visual Scripting
XR Plugin Management

Tags and Layers

User Layer 29	
User Layer 30	
User Layer 31	

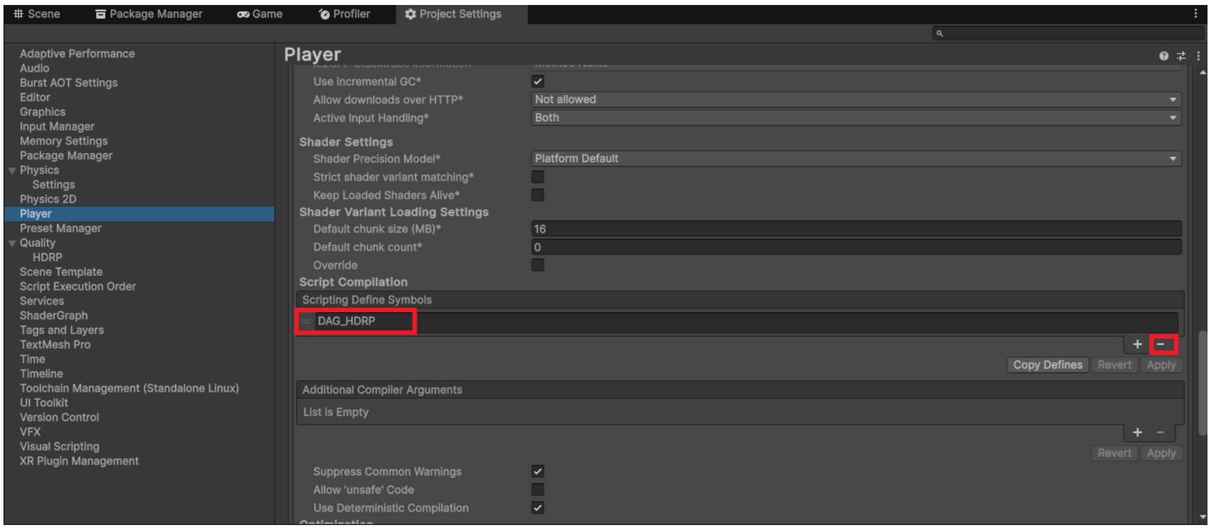
Rendering Layers

Maximum supported rendering layers on HDRenderPipeline is 16.

Layer 0	Default
Layer 1	Exterior
Layer 2	Interior
Layer 3	Light Layer 3
Layer 4	Light Layer 4
Layer 5	Light Layer 5
Layer 6	Light Layer 6
Layer 7	Light Layer 7
Layer 8	Decal Layer default
Layer 9	Decal Layer 1
Layer 10	Decal Layer 2
Layer 11	Decal Layer 3
Layer 12	Decal Layer 4
Layer 13	Decal Layer 5
Layer 14	Decal Layer 6
Layer 15	Decal Layer 7

Script Define Symbols

When first installing the "Daft Apple Core" package, it sets a Script Define Symbol to allow compilation of code specific to your render pipeline. To remove this, go to "Project Settings > Player" and simply delete the script define symbol with the prefix "DAG":


Scene | Package Manager | Game | Profiler | Project Settings

Adaptive Performance
Audio
Burst AOT Settings
Editor
Graphics
Input Manager
Memory Settings
Package Manager
Physics
  Settings
Physics 2D
Player
Preset Manager
Quality
  HDRP
Scene Template
Script Execution Order
Services
ShaderGraph
Tags and Layers
TextMesh Pro
Time
Timeline
Toolchain Management (Standalone Linux)
UI Toolkit
Version Control
VFX
Visual Scripting
XR Plugin Management

Player

Use incremental GC* ✓
Allow downloads over HTTP* Not allowed
Active Input Handling* Both

Shader Settings
Shader Precision Model* Platform Default
Strict shader variant matching* ☐
Keep Loaded Shaders Alive* ☐

Shader Variant Loading Settings
Default chunk size (MB)* 16
Default chunk count* 0
Override ☐

Script Compilation
Scripting Define Symbols
DAG_HDRP
+ −
Copy Defines | Revert | Apply

Additional Compiler Arguments
List is Empty
+ −
Revert | Apply

Suppress Common Warnings ✓
Allow 'unsafe' Code ☐
Use Deterministic Compilation ✓

Building Editor Tool Window

Introduction

The Building Editor Tool is designed as a "one stop shop" for applying a number of changes to a building model or prefab instance.

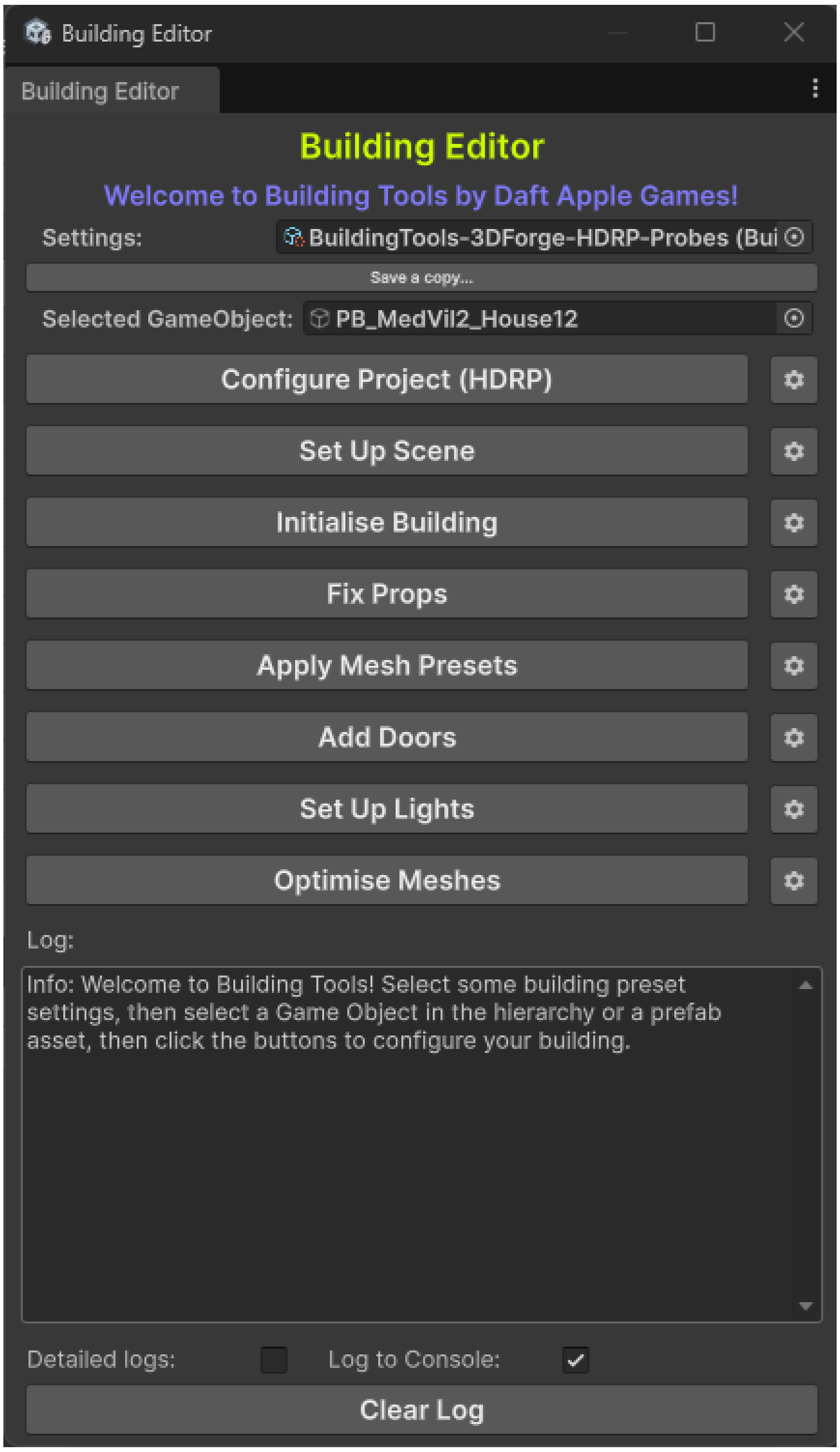
You'll find specific details and instructions for each of the core tools in the links on the left.

The Building Editor Window

The main editor window can be accessed via the Unity editor menu:

Tools > Daft Apple Games > Building Tools > Building Editor

The window looks like this:



Settings: BuildingTools-3DForge-HDRP-Probes (Bui

Selected GameObject: PB_MedVil2_House12

Log:

```
Info: Welcome to Building Tools! Select some building preset
settings, then select a Game Object in the hierarchy or a prefab
asset, then click the buttons to configure your building.
```

Detailed logs: ☐ Log to Console: ✓

Each button represents a specific “Tool” in the package and will apply it’s changes to the Game Object that is currently selected in the scene hierarchy. You can click the cog icon to select the tools configuration Scriptable Object instance in the project. You can find details of each of the tools using the navigation links on this page. All actions are applied to a single building Game Object that you select in the scene hierarchy. Support for applying changes directly to prefab assets is in the works.

Tip

All of the tools use the “Undo” feature of the Unity editor. Having run any one of the tools, you can completely undo any changes made via the “Undo History” menu item.

Note

Generally, the tools will work fine with prefab variants in the scene. In some cases, such as where you need to move prop Game Objects within the parent Game Object structure, you may need to unpack the prefab.

Configure Project Tool

This tool simply sets up various required properties in your project, required by some of the other tools.

Quick Overview

The Set Up tool:

- Adds 4 new “Layers” into your project, which are used to configure discrete “buckets” to manage interior and exterior building meshes, and interior and exterior props.
- Renames the existing “Rendering Layers”, in index locations 1 and 2, used by the “Lighting Tools” to give you more control over how lighting effects various meshes in your building model.

Parameters

These parameters can be configured on an instance of this tool:

Parameter	**Purpose**
Exterior Layer Name	Name of the layer on which building exteriors (think outer walls, etc) will sit.
Interior Layer Name	Name of the layer on which building interiors (think inner walls, etc) will sit.
Interior Props Layer Name	Name of the layer on which interior props (think tables, chairs, rugs, etc) will sit.
Exterior Props Layer Name	Name of the layer on which exterior props (think barrels, carts, stables, etc) will sit.
Exterior Renderer Layer Name	HDRP and URP Render Pipelines only. Name of Rendering Layer that will be influenced by Exterior lights.
Interior Renderer Layer Name	HDRP and URP Render Pipelines only. Name of Rendering Layer that will be influenced by Interior lights.

Set Up Scene Tool

This tool sets up dependent objects and components that are active in your currently open scene or scenes.

Quick Overview

The Set Up Scene tool:

- Looks for a single "Directional" light in your scene and configures the "Lighting Layers", "Rendering Layers", or "Culling Layers", depending on which render pipeline is currently in use for the open project. This effectively tells the main scene Light (for example, the sun), exactly what meshes are influenced by it's light.
- Adds an OnDemandShadowMapUpdate component, if configured to do so, that helps control the frequency of shadow updates triggered by the directional light. This can significantly improve the performance of shadow rendering in the scene.

Parameters

These parameters can be configured on an instance of this tool:

Parameter	**Purpose**
Directional Light Rendering Layer Mask	HDRP and URP Render Pipelines only. Determines which meshes are influence by the main directional light, driven by "Rendering Layers", which is a concept only supported in the "Scriptable" rendering pipelines.
Directional Light Culling Layer Mask	Built In Render Pipeline only. Determines which meshes are influence by the main directional light, driven by the mesh renderer Game Object layers.
Add On Demand Shadow Map Component	HDRP and URP Render Pipelines only. If true, an OnDemandShadowMapUpdate component will be added to your main Directional Light. This can be configured to update the shadow on the light every n frames or m seconds. This can be a significant performance improvement if you are currently updating shadows every frame.
Shadow Refresh Rate	HDRP and URP Render Pipelines only. Sets the number of frames to wait before refreshing the shadows of a light, driven by the new OnDemandShadowUpdate component.

Initialize Building Tool

This tool gets your selected Game Object ready for use by other tools in the package, and makes positioning the building on uneven surfaces a little easier.

Quick Overview

The Add Building Component tool:

- Adds a “Building” component to your selected Game Object, if such a component does not already exist.
- Moves the children of the Game Object so that the effective building pivot allows for an amount of plinth to be shown. This makes positioning the building on an uneven surface, such as a Terrain, a little easier.

Important

Having run the tool, there is a manual step you must complete before using any of the other tools!

You must configure the component by dragging child Game Objects from your building into the slots to define:

- Groups of meshes representing the “Exterior” of your building.
- Groups of meshes representing the “Interior” of your building.
- Groups of meshes representing any “Indoor Props” that exist within your building.
- Groups of meshes representing any “Outdoor Props” that exist within your building instance or prefab.

Parameters

These parameters can be configured on an instance of this tool:

Parameter	**Purpose**
Adjust Pivot Height	How far “up” all direct child Game Objects are raised above the current pivot position of the main building Game Object. Set this to “0” if you don’t want to apply any change to the pivot position.

Fix Props Tool

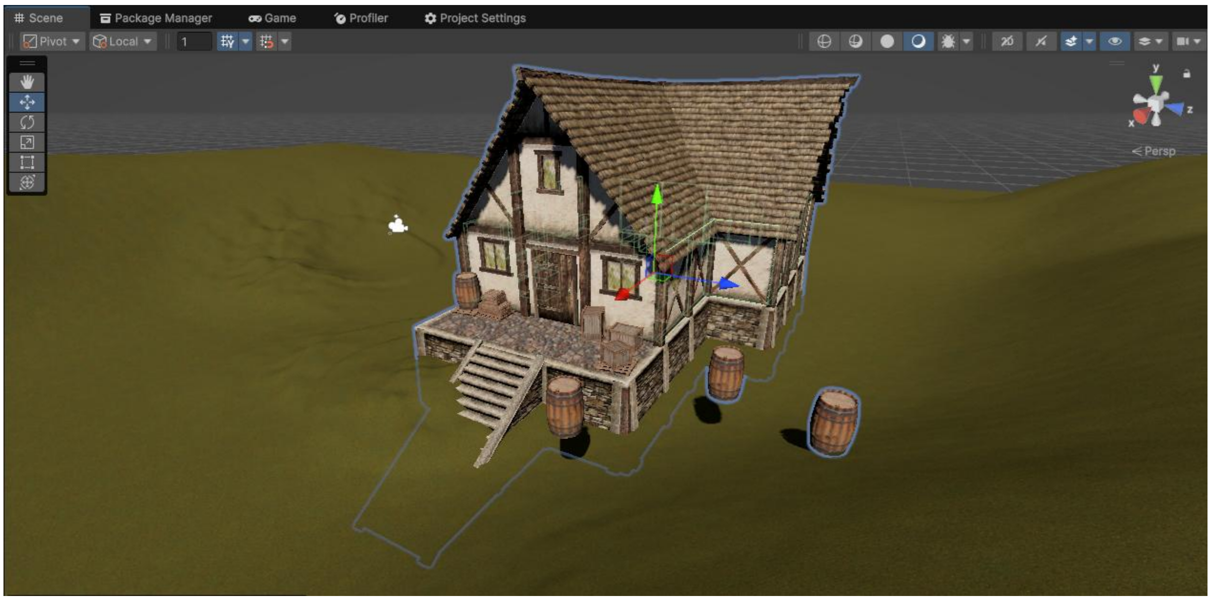
This tool configures props, like barrels, crates, etc, within your building structures.

Quick Overview

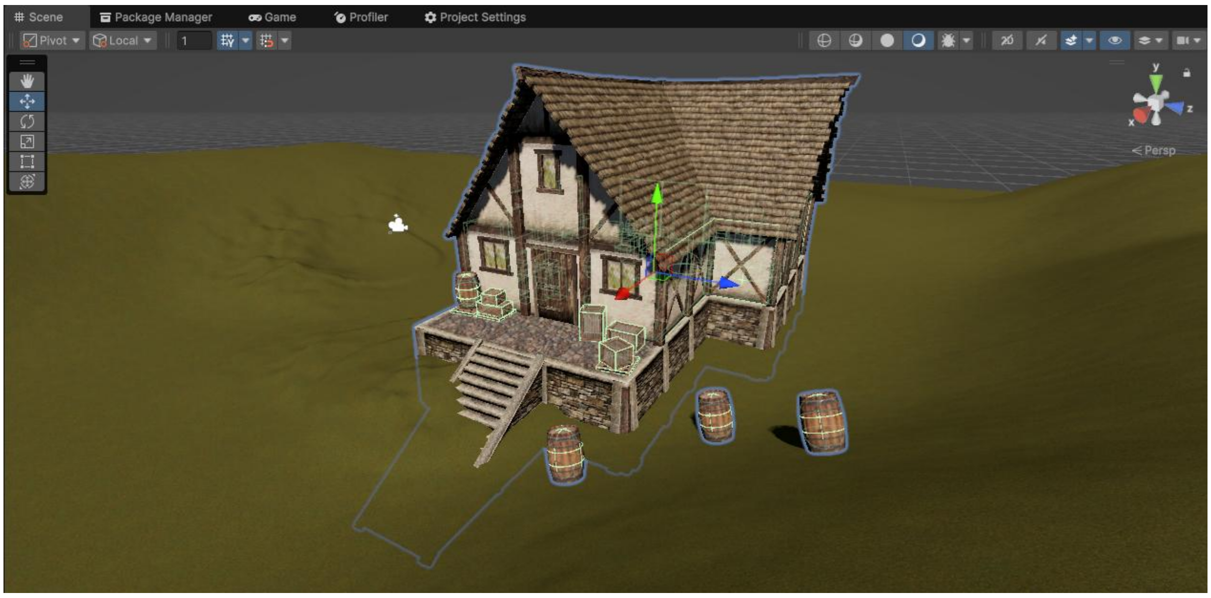
The Fix Props tool:

- Searches for mesh game objects by name and adds any missing Colliders.
    - For example:
        - Crates, chairs and tables get a BoxCollider
        - Barrells get a CapsuleCollider
        - Rugs and carpets get a MeshCollider
- Where exterior props sit directly on a Terrain in your scene, the props are aligned correctly to the terrain.

Props are identified by comparing Game Object names against the list provided in the configuration. The process does a Contains check, so the full Game Object name need only a partial match. For example, the name string ‘crate’ will match a Game Object called ‘fe_vill_crate_11’. Unity automatically scales the collider to fit the mesh, but some minor adjustments may be required.



You’ll see above that there are no colliders highlighted on the visible props, and some of the exterior props are hovering above the terrain. This is almost always going to be the case for assets, as the authors have to assume alignment to a flat surface. A click of a button on the fix props tool results in this:



You'll see the colliders have been added and are highlighted in green by Unity. The barrels have been aligned to the terrain, not just matching the y axis, but aligned also to the slope of the terrain. The axis used in alignment can be modified by setting parameters, as described below.

Parameters

These parameters can be configured on an instance of this tool:

Parameter	**Purpose**
Box Collider Names	Game Objects with this string in their name, that contain a Mesh Renderer, will have a BoxCollider added, if one does not exist.
Sphere Collider Names	Game Objects with this string in their name, that contain a Mesh Renderer, will have a SphereCollider added, if one does not exist.
Capsule Collider Names	Game Objects with this string in their name, that contain a Mesh Renderer, will have a CapsuleCollider added, if one does not exist.
Mesh Collider Names	Game Objects with this string in their name, that contain a Mesh Renderer, will have a MeshCollider added, if one does not exist.
Terrain Align Position	If ticked, Game Objects on the terrain will be lowered to sit flush on the terrain.
Terrain Align Rotation	If ticked, Game Objects will be rotated to align with the contact point on the terrain.
Terrain Align X	If ticked, align rotation will affect this axis. Untick to preserve the axis as is.
Terrain Align Y	If ticked, align rotation will affect this axis. Untick to preserve the axis as is.
Terrain Align Z	If ticked, align rotation will affect this axis. Untick to preserve the axis as is.

Add Doors Tool

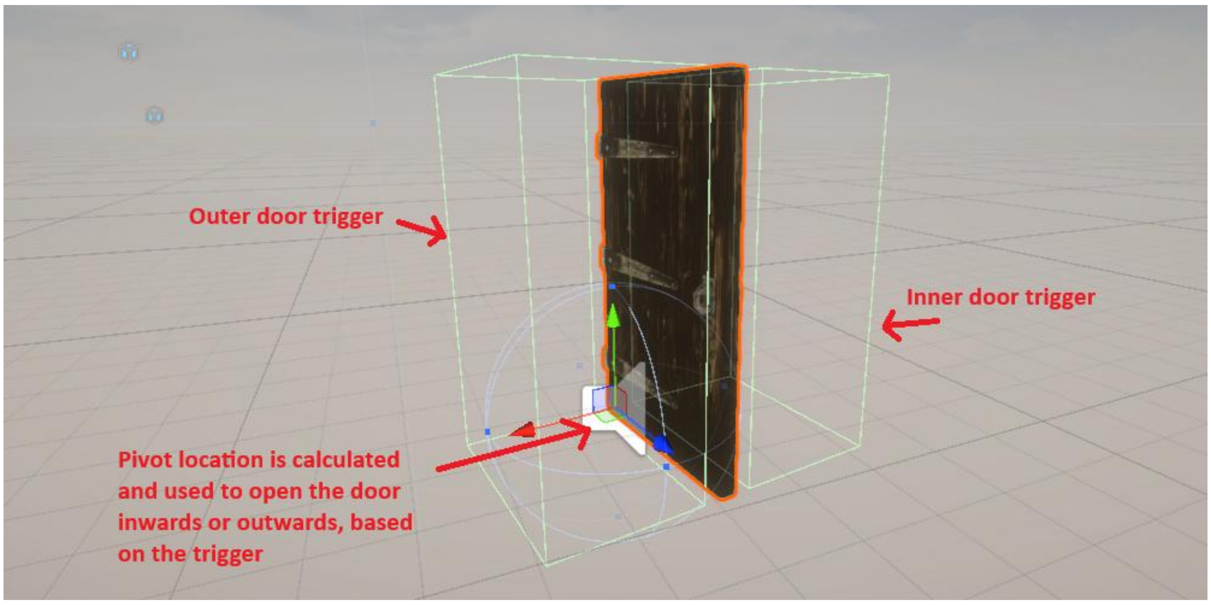
This tool helps you to quickly add functioning doors to your building Game Object.

Quick Overview

The Add Doors tool:

- Adds a “Door” component to mesh Game Objects found by searching for particular name patterns.
- Adds “Door Triggers” to the door, allow it to open and close as the player approaches and leaves the area of the door.
- Adds Audio Clips that play when the door is opening and closing.
- Ensures the “Static Flags” are set to allow the door to move.

The door tool creates two DoorTrigger and corresponding BoxCollidercomponents, in Game Objects on either side of the door. These are sized and positioned according to the door, and tagged as “Inside” or “Outside” triggers based on the forward transform of the door. The pivot location is derived and stored against the Door component and is used, along with the trigger location, to open the door in the correct direction depending on whether it was triggered from outside (the door opens inwards) or inside (the door opens outwards).


Outer door trigger
Inner door trigger
Pivot location is calculated and used to open the door inwards or outwards, based on the trigger

Parameters

These parameters can be configured on an instance of this tool:

Parameter	Purpose
Door Names	Game Objects that contain any of these strings will be configured as doors.
Door Opening Clips	Optional audio clip(s) that play when the door starts opening. If multiple clips are added, a random clip will be played.

Parameter	Purpose
Door Open Clips	Optional audio clip(s) that play when the door completes opening. If multiple clips are added, a random clip will be played.
Door Closing Clips	Optional audio clip(s) that play when the door starts closing. If multiple clips are added, a random clip will be played.
Door Closed Clips	Optional audio clip(s) that play when the door completes closing. If multiple clips are added, a random clip will be played.
Door SFX Group	The AudioMixerGroup used to play the door audio sounds.
Door Trigger Layer Mask	When a collider enters the door trigger, only colliders on any of the identified layers will be considered for triggering the door.
Door Trigger Tags	When a collider enters the door trigger, only colliders with any of the identified tags will be considered for triggering the door.
Override Collider Height	By default, the colliders on either side of the door will match the height of the door mesh. If you want to override this, check this box and enter a value in the override.
Door Collider Override Height	The height of the colliders created on either side of the door, if you've chosen to override the default.
Door Collider Depth	The depth (distance from the door) of the colliders created on either side of the door.
Override Collider Width	By default, the trigger width is calculated based on the width of the door. Tick this to override that with a fixed value.
Door Collider Width Override	The override width of the colliders, if the override flag is set.
Moveable Mesh Static Flags	Static flag mask to be applied to the door. Generally "Nothing" to allow the door to be animated.
Outside Trigger Name	The name of the Game Object created to hold the "outside" door trigger and collider.
Inside Trigger Name	The name of the Game Object created to hold the "inside" door trigger and collider.

Apply Mesh Pre-sets Tool

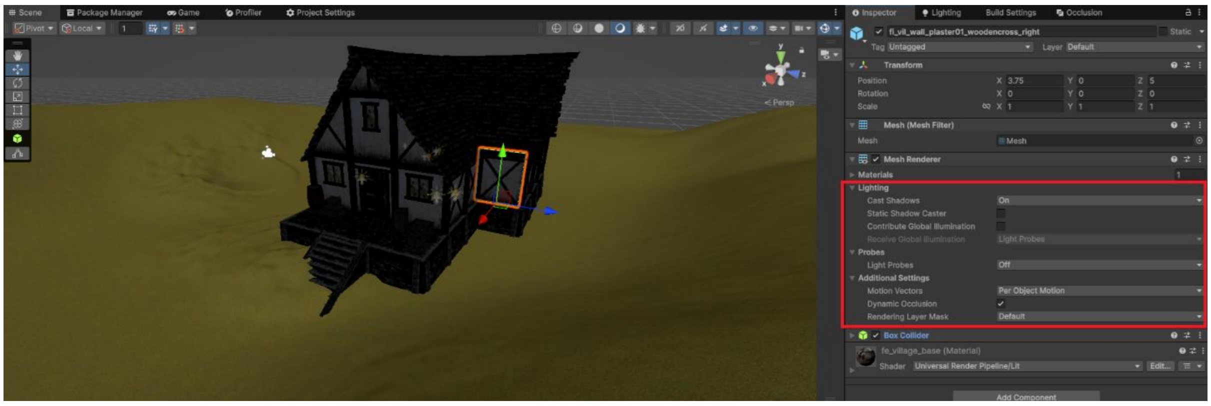
This tool configures meshes within your selected building Game Object, giving you control over various aspects of the building.

Quick Overview

The Apply Mesh Pre-sets tool:

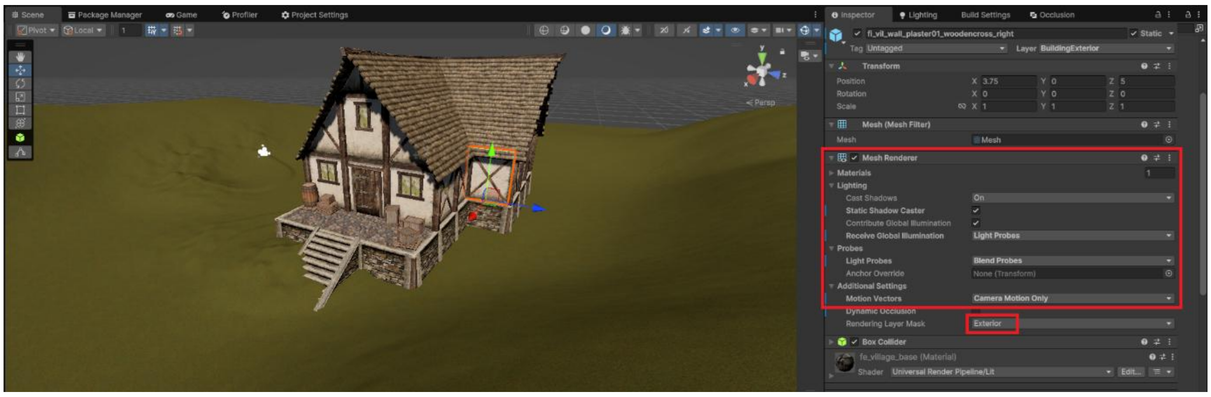
- Takes a set of preconfigured MeshRenderer parameter settings and applies them to all Interior, Exterior and Prop meshes in the selected game object.
    - All MeshRenderer properties can be configured, and support for pipeline specific properties is in the works.

The principle behind this tool is that you can configure a number of mesh "presets", for a number of use cases, and apply those presets consistently across all of the MeshRenderer components in a building. The main use case that I've catered for is how the meshes respond to lighting. For example, I can create pre-sets for "lightmap baked" mesh renderers, with specific shadow casting properties, and than apply those settings to all of the exterior meshes of my building. Doing this manually is painstaking, especially when you want to try different variations of settings on buildings with a large number of mesh renderers.



```
fi_vil_wall_plaster01_woodencross_right            Static
Tag  Untagged                    Layer  Default
Transform
Position     X 3.75   Y 0   Z 5
Rotation     X 0      Y 0   Z 0
Scale        X 1      Y 1   Z 1
Mesh (Mesh Filter)
Mesh                               Mesh
Mesh Renderer
Materials                                          1
Lighting
  Cast Shadows                     On
  Static Shadow Caster
  Contribute Global Illumination
  Receive Global Illumination      Light Probes
Probes
  Light Probes                     Off
Additional Settings
  Motion Vectors                   Per Object Motion
  Dynamic Occlusion                ✓
  Rendering Layer Mask             Default
Box Collider
fe_village_base (Material)
Shader  Universal Render Pipeline/Lit       Edit...
```

In the screenshot above, you'll see that all of the exterior meshes are configured to receive light from the "Default" render layer. Light probes are off, and there are some other settings applied to the meshes. Clicking the mesh tool button results in some changes:



```
fi_vil_wall_plaster01_woodencross_right          ✓ Static
Tag  Untagged                    Layer  BuildingExterior
Transform
Position     X 3.75   Y 0   Z 5
Rotation     X 0      Y 0   Z 0
Scale        X 1      Y 1   Z 1
Mesh (Mesh Filter)
Mesh                               Mesh
Mesh Renderer
Materials                                          1
Lighting
  Cast Shadows                     On
  Static Shadow Caster             ✓
  Contribute Global Illumination   ✓
  Receive Global Illumination      Light Probes
Probes
  Light Probes                     Blend Probes
  Anchor Override                  None (Transform)
Additional Settings
  Motion Vectors                   Camera Motion Only
  Dynamic Occlusion
  Rendering Layer Mask             Exterior
Box Collider
fe_village_base (Material)
Shader  Universal Render Pipeline/Lit       Edit...
```

As you can see, all of the exterior meshes are now receiving light from the “Exterior” layer, which happens to be the layer that the directional light is influencing. All meshes are now configured to cast static shadows and receive indirect light from light probes.

Parameters

These parameters can be configured on an instance of this tool:

Parameter	**Purpose**
Interior Mesh Settings	Mesh presets to be applied to all interior building mesh renderers.
Exterior Mesh Settings	Mesh presets to be applied to all exterior building mesh renderers.
Interior Prop Mesh Settings	Mesh presets to be applied to all interior prop mesh renderers.
Exterior Prop Mesh Settings	Mesh presets to be applied to all exterior prop mesh renderers.

Mesh Pre-sets

Each of the “Mesh Settings” properties of this tool are themselves Scriptable Object instances. Each instance has these parameters, which are derived directly from the “Mesh Renderer” component in Unity. Therefore, you can refer to the Unity Mesh API page for details of each of the properties.

Currently supported properties are:

Parameter	**Notes**
Layer Name	
Static Editor Flags	Includes what is surfaced as “Contribute GI” in the Unity Inspector.
Shadow Casting Mode	
Static Shadow Caster	
Receive GI	
Light Probe Usage	
Reflection Probe Usage	Built In Render Pipeline only.
Scale in Lightmap	HDRP and URP only.
Motion Vectors	
Dynamic Occlusion	
Rendering Layer Mask	HDRP and URP only.
Priority	HDRP and URP only.

Add Volumes Tool

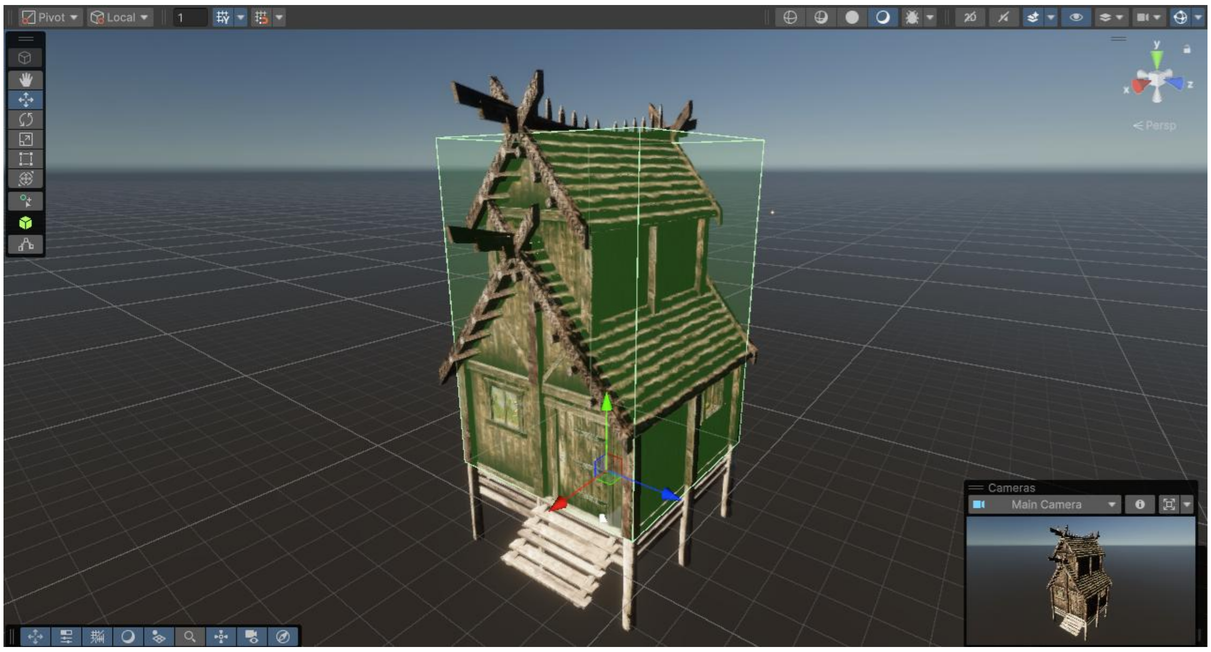
This tool sets up “Volumes”, which are areas that encompass your selected building Game Object, allowing you to configure various functions that trigger as a player moves up and out of the building.

Quick Overview

The Add Volumes tool:

- Adds an “Interior Volume” (HDRP and URP only) that gives you control over lighting when the player enters the building.
- Adds an “Interior Audio Filter” component, that allows you to create a “muffled” sound for outdoor/ambient audio when the player enters the building.

The “bounds” of the building are calculated by combining the bound extends of all meshes within a configurable layer - this is the “BuildingExterior” layer by default. You can fine tune this by excluding specific mesh renderer in Game Object names containing any number of strings, all using the parameters described below. The tool uses this to create a BoxCollider, set as a trigger, that is used to control components that do different things when a collider (e.g. the player) enters and leaves this collider.



Due to certain mesh renderers, such as over-hangs on ceilings, you may have to tweak the final collider size manually, to contain only the “interior” areas of your building.

The layers of colliders checked, and tags that will cause the triggers to fire, are configurable via the parameters described below. They can also be modified directly on the corresponding components that are attached to the “Interior Volume” Game Object that is created by this tool.

Important

You should add and configure volumes before optimising the building meshes. Otherwise, the calculation of the building bounds will not be able to consider smaller meshes for exclusion, and the result may be that the volume encompasses a larger area than desired. You can manually re-size the collider after it has been created.

Parameters

These parameters can be configured on an instance of this tool:

Parameter	**Purpose**
Mesh Size Ignore Names	When determining the “bounds” of a building, any mesh renderer Game Objects containing these names are ignored by the calculation, as are any of their children. Can be used to remove “invisible” meshes, such as those under the ground or that “overhang” the interior space, from the calculation.
Mesh Size Include Layers	LayerMask containing layers that will be considered by the mesh size calculation. Only mesh renderers within these layers will be considered.
Interior Volume Game Object Name	Name of the Game Object created, if not there already, that contains the various volume components.
Interior Volume Profile (HDRP and URP only)	Selected Volume Profile that will be applied when the player/camera enters the building.
Indoor Snapshot	AudioMixer Snapshot corresponding to the AudioMixer profile that is applied when the player/camera enters the building. Typically contains “Lowbypass/Highbypass” filters to “muffle” the audio.
Outdoor Snapshot	AudioMixer Snapshot corresponding to the AudioMixer profile that is applied when the player/camera exists the building. Typically a “clean” snapshot profile.
Volume Trigger Tags	When a collider enters the volume trigger collider, only colliders on Game Objects with one of these tags will trigger the volume effect.
Volume Trigger Layer	When a collider enters the volume trigger collider, only colliders on Game Objects on one of these layerswill trigger the volume effect.

Set Up Lighting Tool

This tool configures lights within your selecting building Game Object and adds components to give you control over their function.

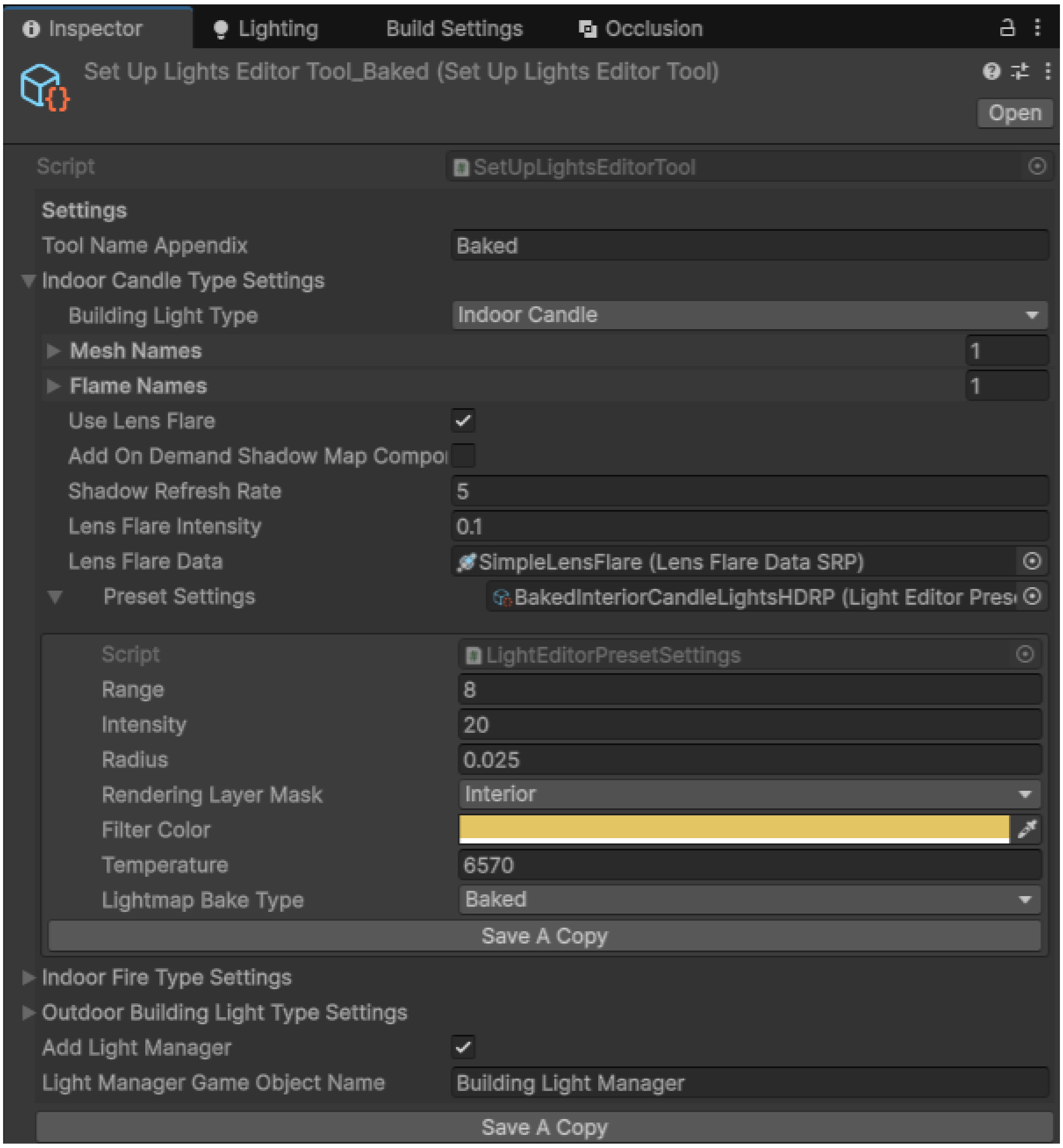
Quick Overview

The Set Up Lighting tool:

- Adds a BuildingLight component to each light within the building, giving more granular control over it's function.
- Adds a BuildingLightController to the building, allowing you to control groups of lights within the building structure.
- Configures lights for any one of three types, identified by the containing Game Object name:
  - Indoor candles
  - Indoor fires (cookers, fireplaces)
  - Outdoor lights
- Configures the range, intensity, and colour depending on the type of light.
- Adds a BuildingLightManager singleton component, which gives you complete control of all building lights in the scene.
- Removes unsupported "built in" Flare components when using URP or HDRP pipelines.
- Replaces "built in" Flare components with SRP Flare components, along with a preconfigured sample flare setup.
- Configures the RenderingLayerMask or CullingLayerMask, depending on the active render pipeline.
  - This gives you control over the influence of exterior, interior and lights based on where a mesh is located inside or outside your building.
- Adds an OnDemandShadowMapUpdate component, if configured to do so, that helps control the frequency of shadow updates triggered by the light. This can significantly improve the performance of shadow rendering in the scene. Works on HDRP only, for the moment.

There are two primary goals of this tool:

Consistent Lighting - apply consistent lighting settings across different types of light in your building. For example, you may want all candles within buildings to share intensity, range, and colour temperature. You may want all interior lights to only impact interior meshes. You can do this by applying "lighting presets" configured within the tools:

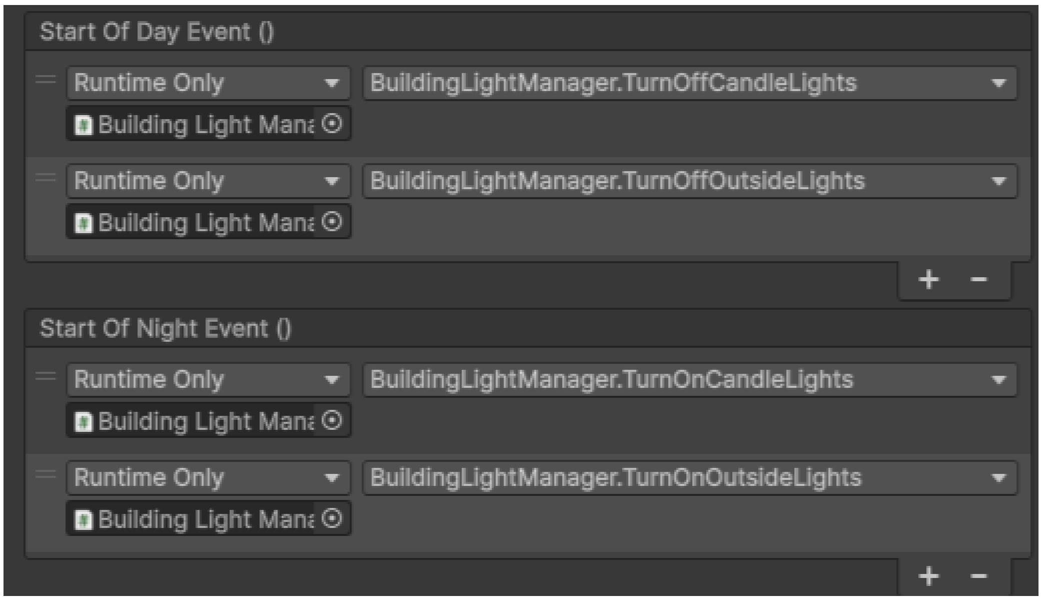

Inspector | Lighting | Build Settings | Occlusion

Set Up Lights Editor Tool_Baked (Set Up Lights Editor Tool)

Open

Script	SetUpLightsEditorTool
Settings	
Tool Name Appendix	Baked
▼ Indoor Candle Type Settings	
Building Light Type	Indoor Candle
▶ **Mesh Names**	1
▶ **Flame Names**	1
Use Lens Flare	✓
Add On Demand Shadow Map Compo	
Shadow Refresh Rate	5
Lens Flare Intensity	0.1
Lens Flare Data	SimpleLensFlare (Lens Flare Data SRP)
▼ Preset Settings	BakedInteriorCandleLightsHDRP (Light Editor Pres
Script	LightEditorPresetSettings
Range	8
Intensity	20
Radius	0.025
Rendering Layer Mask	Interior
Filter Color	🟨
Temperature	6570
Lightmap Bake Type	Baked
Save A Copy	
▶ Indoor Fire Type Settings	
▶ Outdoor Building Light Type Settings	
Add Light Manager	✓
Light Manager Game Object Name	Building Light Manager
Save A Copy	

Granular Lighting Control - provide components and methods to allow complete, and granular, control over the state of building lighting within open scenes. For example, using the sample SimpleTimeOfDay component in the samples, we can use events to turn all building lights on and off as the time of day changes:


Start Of Day Event ()

Runtime Only | BuildingLightManager.TurnOffCandleLights
Building Light Mana

Runtime Only | BuildingLightManager.TurnOffOutsideLights
Building Light Mana

Start Of Night Event ()

Runtime Only | BuildingLightManager.TurnOnCandleLights
Building Light Mana

Runtime Only | BuildingLightManager.TurnOnOutsideLights
Building Light Mana

Building Light Components

There is a hierarchy of components, all optional, that can be added to your scene to give you control over lights in and around your buildings.

- Building Light Manager (scene wide control)
    - Building Light Controller (building wide control)
        - Building Light (individual light control)

The Building Light Manager is a singleton and so it's methods can be called from script anywhere using:

BuildingLightManager.Instance

For example:

BuildingLightManager.Instance.TurnOnCandleLights();

This will turn on all indoor candle lights in all buildings in the current scene.

If you are using additive scenes, you can also add a BuildingLightManagerMethods component into any scene, and hook that into Unity events. So you can trigger an event in one scene, calling a BuildingLightManagerMethod method that calls the singleton method, even though the manager itself is in a different scene. Just ensure you load and activate the scene with the manager before calling the event methods.

Important

If you are loading scenes additively, remember to call the BuildingLightManager.RefreshLightControllers() method once all scenes have been loaded. This will cause the manager to find all controllers across all open scenes.

Parameters

These parameters can be configured on an instance of this tool:

Parameter	**Purpose**
Indoor Candle Light Settings	Pre-sets to be applied to all "indoor candle" lights within the building.
Indoor Fire Light Settings	Pre-sets to be applied to all "indoor fire" lights (for example, cookers, hearth fires etc) within the building.
Outdoor Building Light Settings	Pre-sets to be applied to all "outdoor" lights within the building Game Object hierarchy.
Add Light Manager	If checked, will add a Light Manager Game Object and singleton component if one does not already exist in the scene.
Light Manager Game Object Name	If a new Light Manager is created, it will be created in a Game Object at the root of your active scene, and given this name.

Each of these properties is itself an instance of a class with the following properties:

Parameter	Purpose
Building Light Type	The type of light to which these settings apply. You should only have one instance of each light type in a given tool configuration.
Mesh Names	Game Objects that contain any of these strings will be configured as Lights by the tool.
Flame Names	Game Objects that contain any of these strings will be configured as flames by the tool. This is then used to toggle flames on and off along with the light itself.
Use Lens Flare	Determines whether Lens Flare components should be added and configured for this light type.
Lens Flare Intensity	The intensity of the lens flare, if one is added.
Lens Flare Data	HDRP and URP only. SRP Lens Flare configuration for the lens flare, if one is added.
Add On Demand Shadow Map Component	HDRP only. If true, an OnDemandShadowMapUpdate component will be added to the light. This can be configured to update the shadow on the light every n frames or m seconds. This can be a significant performance improvement if you are currently updating shadows every frame.
Shadow Refresh Rate	HDRP only. Sets the number of frames to wait before refreshing the shadows of a light, driven by the new OnDemandShadowUpdate component.
Preset Settings	Light properties to be applied to each light. See the "Light Presets" section below.

Light Presets

Each of the "Preset Settings" properties of this tool are themselves Scriptable Object instances. Each instance has these parameters, which are derived directly from the "Light" component in Unity. Therefore, you can refer to the Unity Light API page for details of each of the properties.

Currently supported properties are:

Parameter	Notes
Range	
Intensity	
Radius	HDRP and URP only.
Rendering Layer Mask	HDRP and URP only.
Culling Layer Mask	Built In Render Pipeline only.

Parameter	Notes
Filter Color	
Temperature	
Lightmap Bake Type	

Optimise Meshes Tool

This tool helps to reduce the number of draw calls and shadow casters on your selected building Game Object. This can greatly improve the performance of your project.

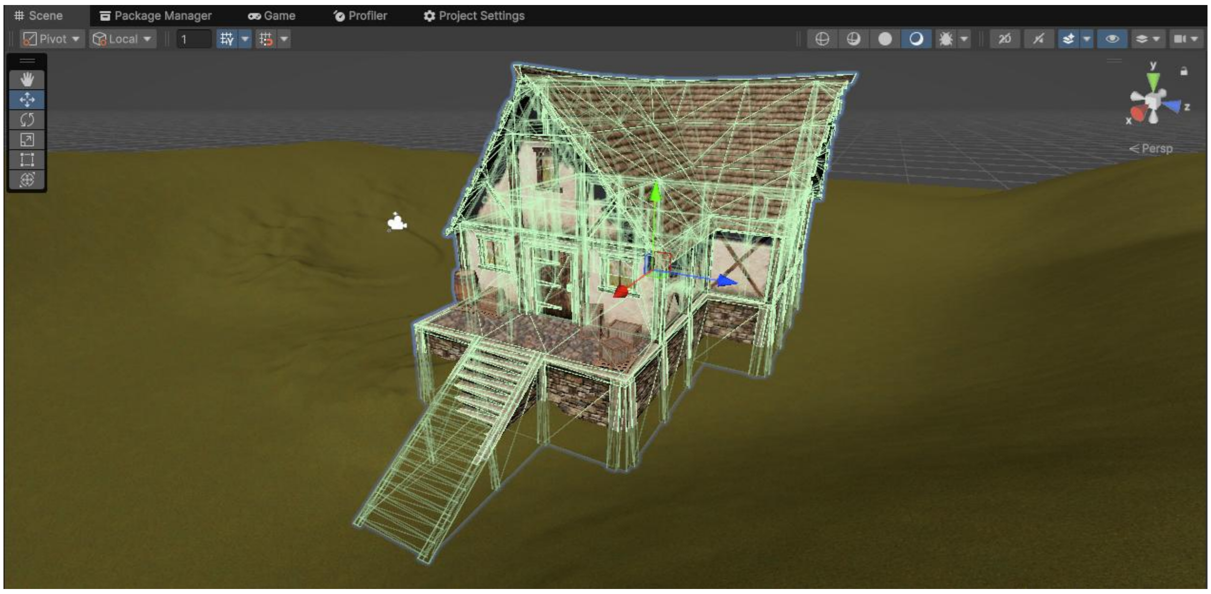
Quick Overview

The Optimise Meshes Tool:

- Combines meshes from interior, exterior, interior prop and exterior prop layers respectively, generating one mesh per material for each group.
- Ignores meshes that have been flagged by the "Ignore Mesh Combine" component, where dynamic behaviour is required. For example, moving doors.
- Saves the combined meshes as assets and reconfigures the Game Object to use the new meshes.

You can also opt to merge the whole building, by unticking the "Combine Meshes by Layer" parameter. This means meshes that share materials across interior, exterior and prop layers will be merged, giving you less control over the behaviour of light on the resulting mesh renderers.

The resulting merge meshes are saved as assets, to be folder specified in the parameters - the folder is created in the "Assets" directory in your project, if it doesn't already exist.



In the image above, you can see that all of the exterior meshes have been combined into a small number of larger meshes, one mesh for each unique material.

Once you've run the tool against a building, you'll notice a new component has been added to the Game Object, called MeshCombineRollBack. The purpose of this component is, unsurprisingly, to allow you to undo the process and restore the Game Object. It does this by storing an "Audit" of the changes made by the optimisation process to that specific building Game Object, and provides a button that allows you to undo those changes. Specifically, this component will:

1. Re-enable all of the original Mesh Renderers that it deactivated.

2. Delete the merge mesh Game Objects from your building.
3. Delete the mesh and Game Object prefab assets that were created.
4. Delete the MeshCombineRollBack component itself.

You can optionally retain any or all of the deleted objects by unchecking the corresponding checkbox in the component inspector, prior to clicking the button. You can also remove the audit altogether, if it’s starting to take up too much space in your scene asset for example, by simply removing the component from the Game Object. You will, of course, then lose the ability to easily roll back.

Parameters

These parameters can be configured on an instance of this tool:

Parameter	**Purpose**
Combine Meshes By Layer	If checked, meshes will be combined separately for each of the BuildngExterior, BuildingInterior, ExteriorProps and InteriorProps layers. If unchecked, the whole building will be merged as one.
Asset Output Folder	Folder in the “Assets” folder where combined meshes and prefabs will be saved.
Asset File Name Prefix	Prefix that will be added to each generated combined mesh and prefab.
Create Output Folder	If the Asset Output Folder does not exist, checking this will cause the tool to create it.
Is 32Bit	If checked, forces the created mesh to use a 32-bit index format. Otherwise, it will use a 16-bit index format. See Mesh.indexFormat for more details.
Generate Secondary UVs	Forces the tool to generate the UVs for the new combined meshes.
Combined Mesh Presets	This is an instance of the Apply Mesh Presets Tool, and is used to apply pre-set parameters to the newly created merged meshes. Note that in the case of a “whole building” merge (if “Combine Meshes By Layer” is unchecked), then the resulting meshing will have the “Building Exterior” mesh pre-sets applied.

Building Runtime Components Overview

Components are provided for use both in the "core" package, to be shared and used across all Daft Apple Unity Framework packages, and within the package itself.

Building Components

The following runtime components are contained in the package, some of which are demonstrated in the sample scenes.

Building

Mainly provides data storage for persistent properties of a building.

BuildingLight

Controls a single Light component within a building.

BuildingLightController

Controls all instances of BuildingLights within a building.

BuildingLightManager

Controls all instances of BuildingLightController across all open scenes.

BuildingLightManagerMethods

Provides access to methods on the BuildingLightManager singleton from event handlers in any open scenes.

Door

Provides methods and properties for opening and closing doors.

DoorController

Controls all doors within a building

DoorTrigger

Uses a BoxCollider to trigger the opening and closing of doors.

DynamicMeshRenderer

A simple "placeholder" component used to prevent merging of dynamic meshes and to ensure appropriate static flags are applied.

InteriorAudioFilter

Uses a BoxCollider to apply a "muffled" sound to a given AudioMixerGroup when the collider is triggered.

MeshCombineRollBack

Provides data storage and functions to rollback combined mesh changes.

WindMill

Provides a simple component to rotate the blades of a windmill at a configurable speed.

Core Components

A number of “core” components are provided in the core package that provide useful functions outside of the building tools package.

Documentation for those components can be found on the Core Runtime Components page.