SuperTiled2Unity Documentation

Release 1.0.0.0

Seanba

Sep 15, 2023

User Manual

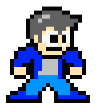
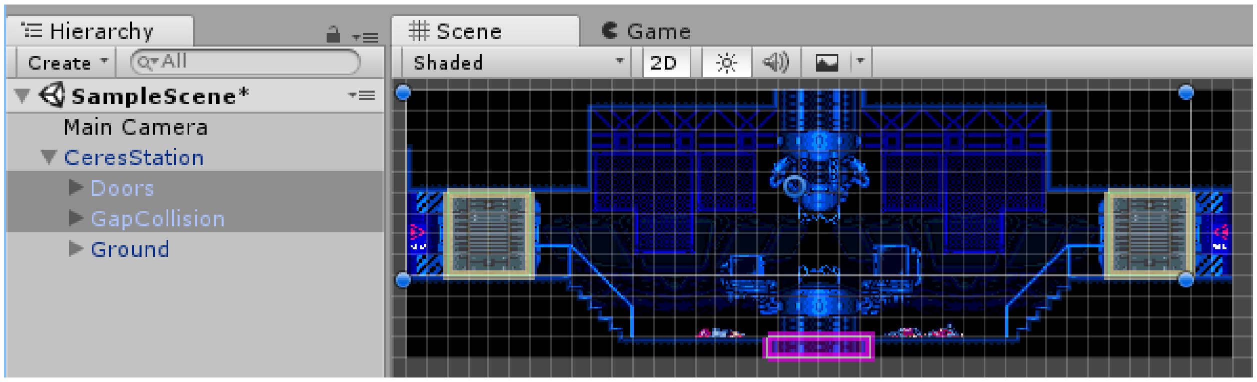


Fig. 1: By Sean Barton

SuperTiled2Unity is a collection of Unity scripts that import files from the popular Tiled Map Editor in your Unity projects. It is available for download on itch.io. The goal of SuperTiled2Unity is that **it just works**. Except for the most specialized cases users should be able to quickly and easily add tile-based 2D content to their Unity projects.

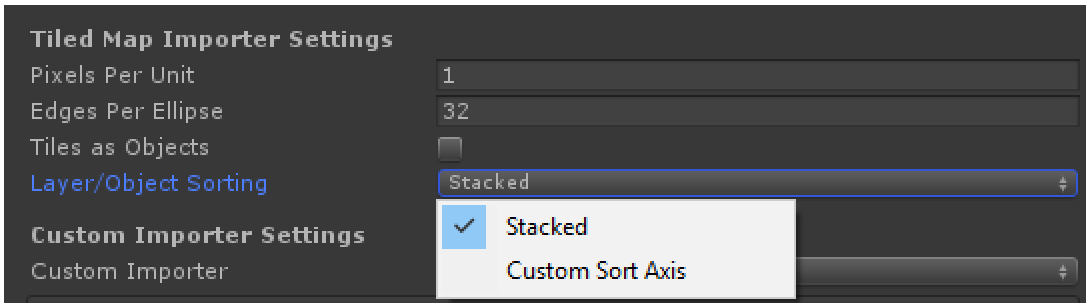
SuperTiled2Unity is currently distibuted as a Unity Package at any price you choose (including free). Additional donations can be made and **are always appreciated**.



CHAPTER 1

Sorting with SuperTile2Unity

At the map import level, SuperTiled2Unity has two options for sorting the layers (and objects) in your Tiled Map Editor file.



Tiled Map Importer Settings
Pixels Per Unit: 1
Edges Per Ellipse: 32
Tiles as Objects: ☐
Layer/Object Sorting: Stacked

Sorting Modes

Stacked	Default sorting. Matches the rendering order of layers and objects in Tiled.
Custom Sort Axis	Sorting is performed with the help of a Custom Sort Axis (a setting in Unity).

`Stacked` is a good default for side-scroller games where players and other game objects do not move about the map in ways that change their rendering order. Overhead-style games may prefer to use the `Custom Sort Axis` setting. This takes a little more work but will be necessary if you need the rendering order of game objets against tiles to be dynamic.

1.1 How SuperTiled2Unity Implements Sorting

In Unity, render order of sprite and tile assets is generally handled through two settings on the Sprite Renderer and Tilemap Renderer components:

Sorting Layer	Name of sorting layer. See the Tag Manager to manage these.
Order in Layer	How the renderer is sorted in the named layer.

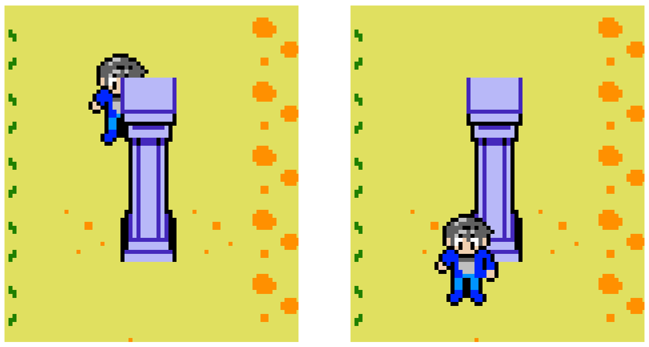


Fig. 1: The example that comes with SuperTiled2Unity uses a Custom Sort Axis so that our player can be rendered either in front of or behind these columns depending on his current y-position.

SuperTiled2Unity performs sorting almost primarily through manipulating the `Order in Layer` setting of the prefab components it creates during import. By default, all tile layers use the Unity's built-in `Default` sorting layer with ever increasing `Order in Layer` values.

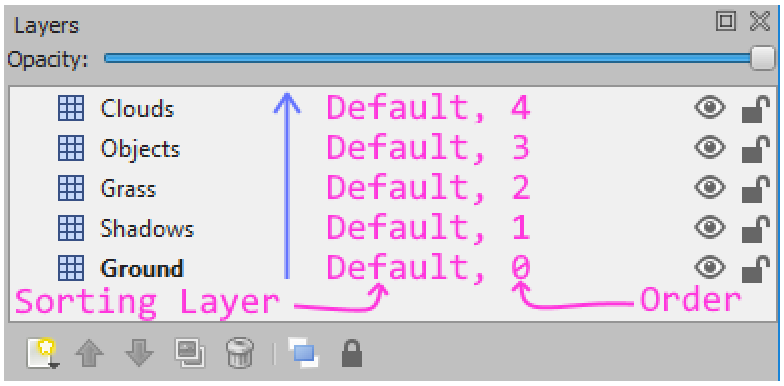


Fig. 2: Higher layers in Tiled use higher `Order in Layer` values in Unity so that rendering order is preserved.

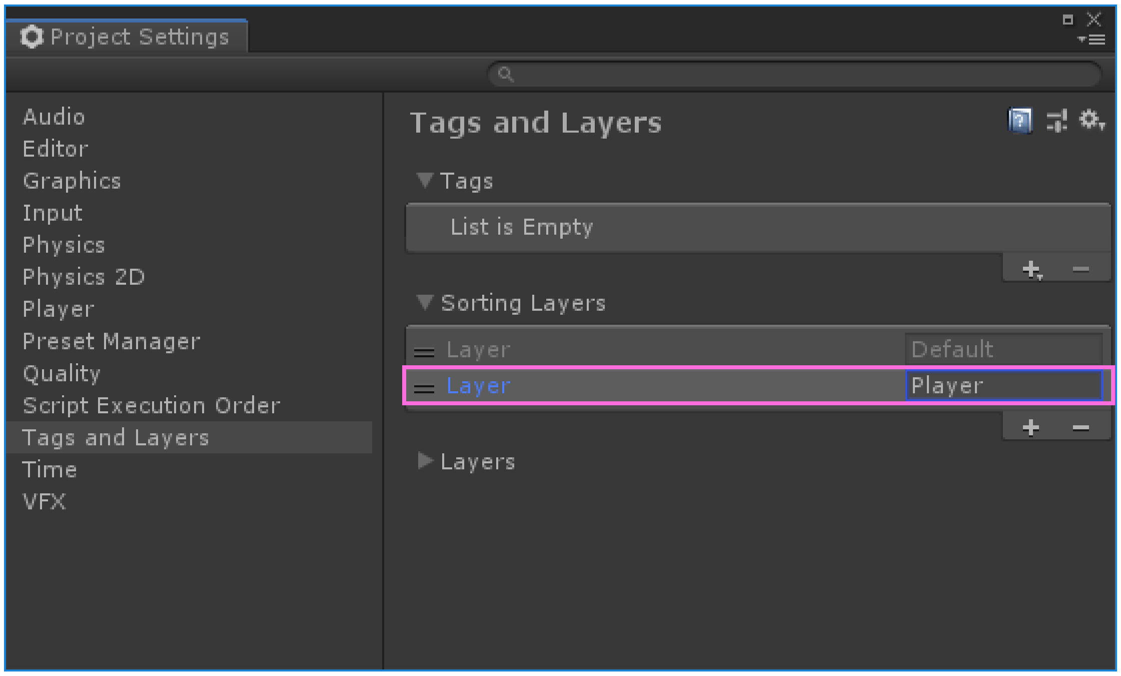
Most Unity projects, however, will have several custom `Sorting Layers` that we want a mix of tiles and sprites to share. In these cases, a specifically-named custom property, `unity:SortingLayer`, will direct SuperTiled2Unity further on how sorting fields are assigned.

This will result in our `Objects` tile layer breaking the chain of `Default` sorting layers.

Note that the `Clouds` layer will still be rendered on top of the `Objects` layer. You may wish to use yet another `unity:SortingLayer` for clouds to make it more explicit that these tiles are drawn on top of other tiles and sprites.

Now the cloud tiles will be rendered in order but on a sorting layer that is more aptly named for them and other tiles that may always appear above our map.

Tip: It takes a bit of work but in general it is a good idea to be explicit about what layers (in Tiled) are assigned to which sort layer (in Unity). Using the `unity:SortingLayer` in concert with Unity Sorting Layers *early* makes it easier to make sweeping sorting changes *later*.



1.2 Tile Objects and Sorting

Tiled allows you to place tiles in a an object layer as separate tile objects. During import, Tiled2Unity turns these tile objects into sprites that are not part of any `Tilemap`. If you import your map with the `Stacked` sorting mode then these sprites will also be assigned a sort order.

This may make it difficult to predict what sorting order is assigned to the layer that comes after `TileObjects` as it depends on the number of objects in that group. Using a custom property to set the sorting layer name on the next layer will help. (Note that for the `Custom Sort Axis` sort mode that each imported sprite is not assigned an incrememted sorting order as they will be sorted by their y-position instead.)

1.3 Dynamic Sorting with a Custom Sort Axis

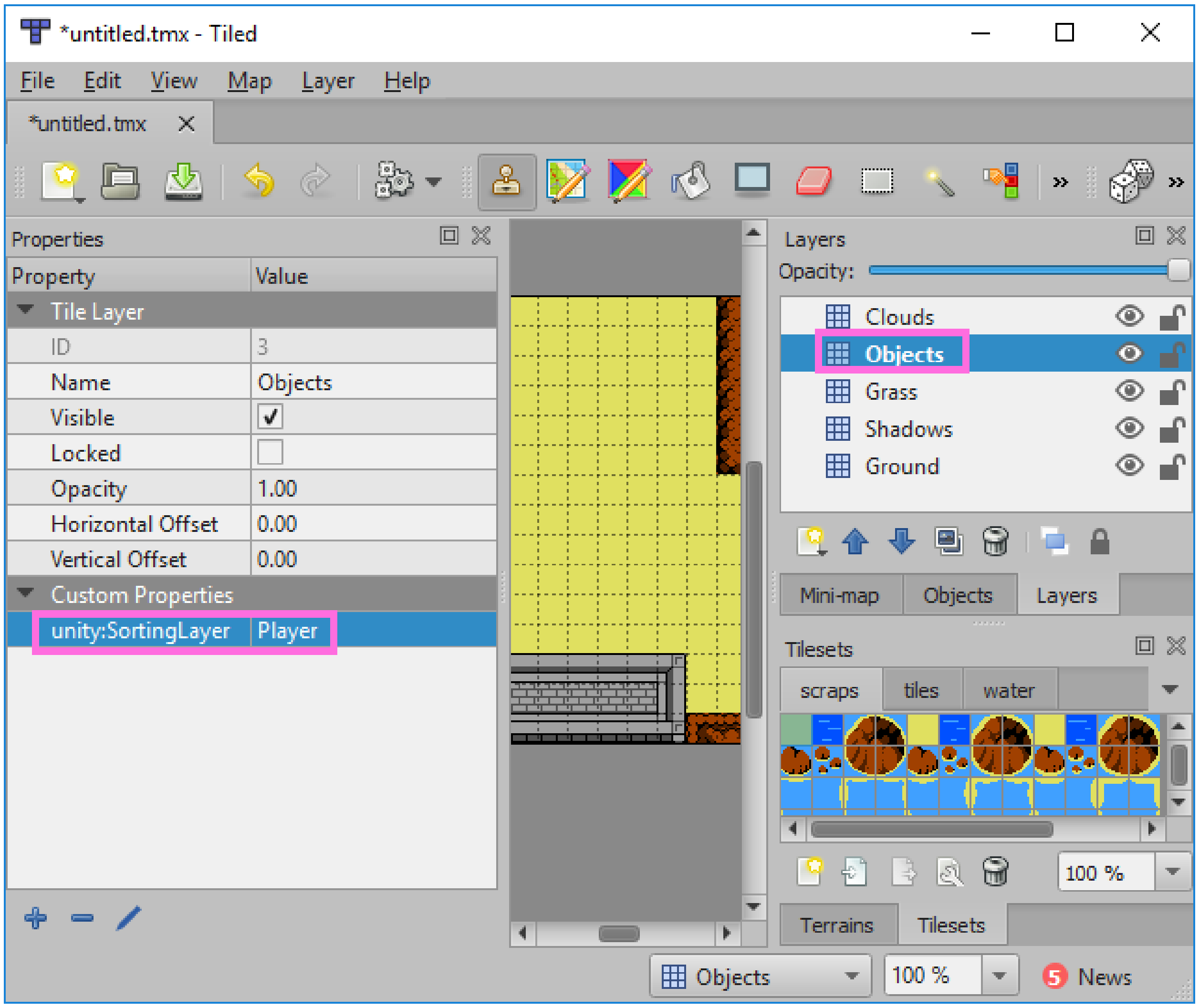
Games with an overhead view often have sprites that need to alter their rendering order with tile maps as they move around. The classic example is a sprite that may appear either in-front-of or behind a column based on their y-position.

In order to use custom axis sorting follow these three steps:

- Import your map with the `Custom Sort Axis` sort mode
- Modify `Transparency Sort Mode` and `Transparency Sort Axis` in your Unity Project Settings to sort against an axis with increasing y-value.
- Make sure your sprites (Unity) and tiles (Tiled) *that you want to sort dynamically* are assigned the same `Sorting Layer` and `Order in Layer` values. (This is the most common source of errors.)

Tip: Objects to be dynamically sorted by a Custom Sort Axis will need to have the same `Sorting Layer` and `Order in Layer` values of the tiles they are sorting against.

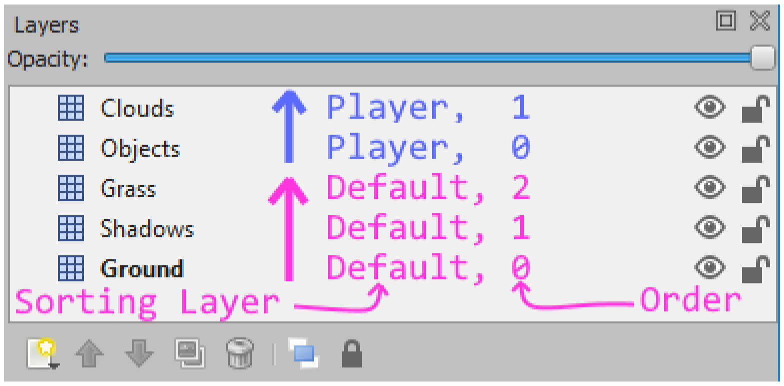
Note that you can also set the `Transparency Sort Axis` in script if you wish.



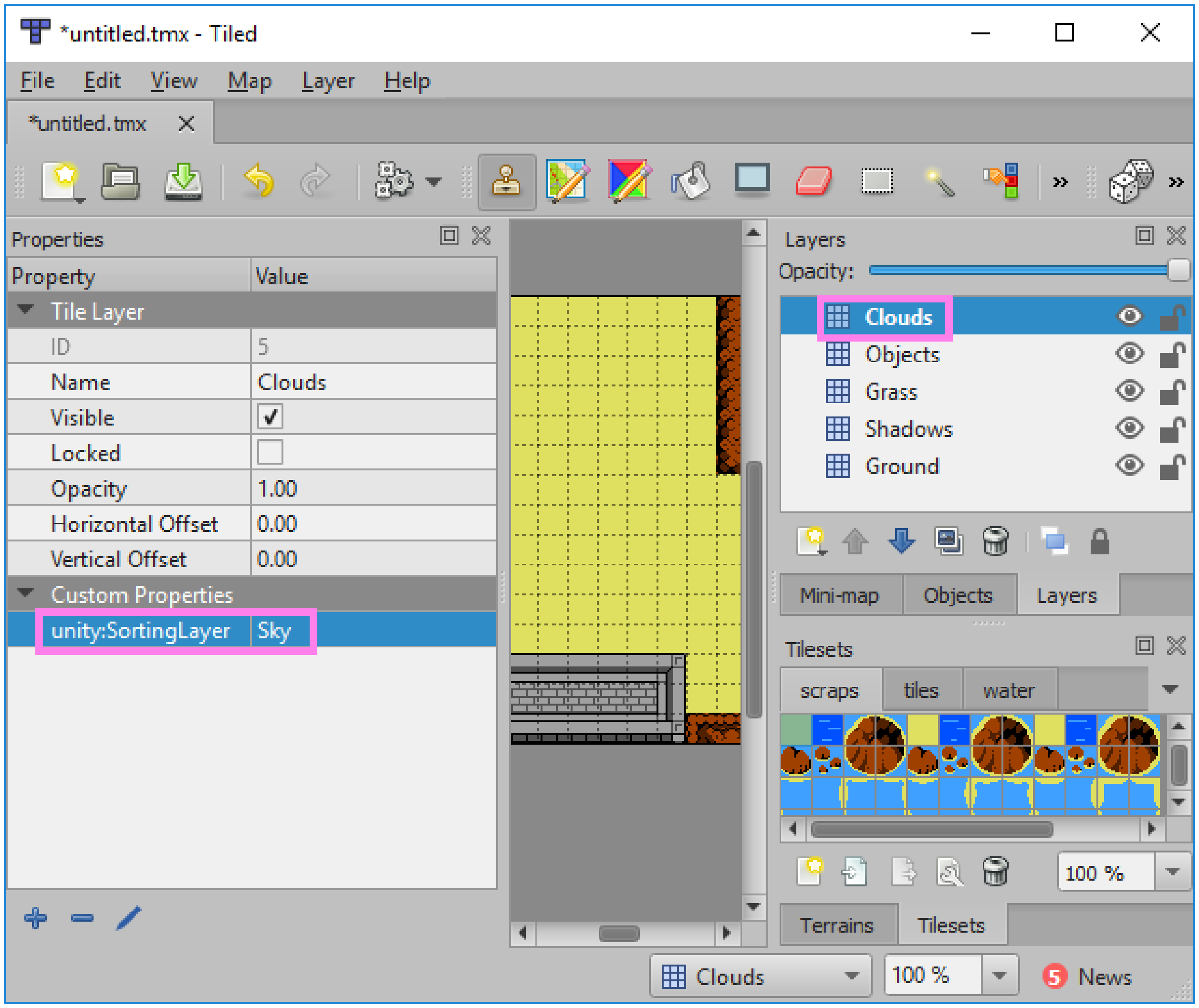
Properties

Property	Value
Tile Layer	
ID	3
Name	Objects
Visible	✓
Locked	
Opacity	1.00
Horizontal Offset	0.00
Vertical Offset	0.00
Custom Properties	
unity:SortingLayer	Player

Layers: Clouds, Objects, Grass, Shadows, Ground

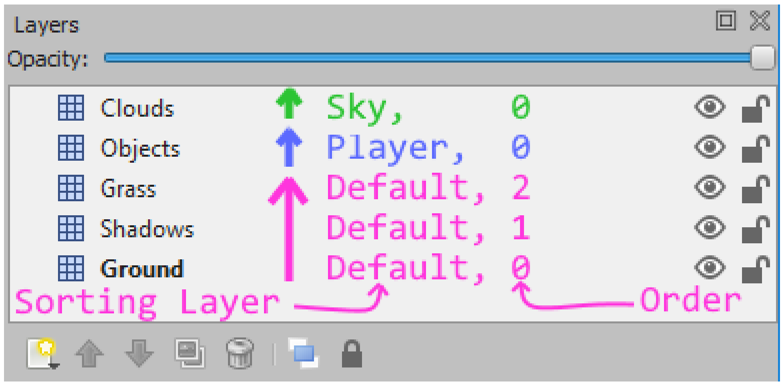


Layer	Sorting Layer	Order
Clouds	Player,	1
Objects	Player,	0
Grass	Default,	2
Shadows	Default,	1
Ground	Default,	0



Property	Value
Tile Layer	
ID	5
Name	Clouds
Visible	✓
Locked	
Opacity	1.00
Horizontal Offset	0.00
Vertical Offset	0.00
Custom Properties	
unity:SortingLayer	Sky

Layers: Clouds, Objects, Grass, Shadows, Ground



Clouds	Sky,	0
Objects	Player,	0
Grass	Default,	2
Shadows	Default,	1
Ground	Default,	0

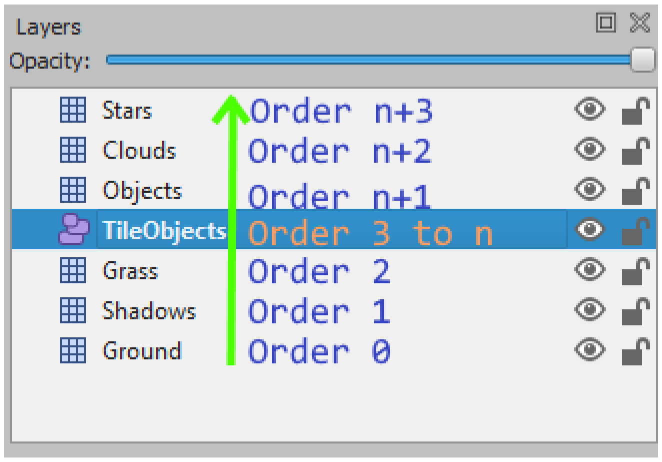
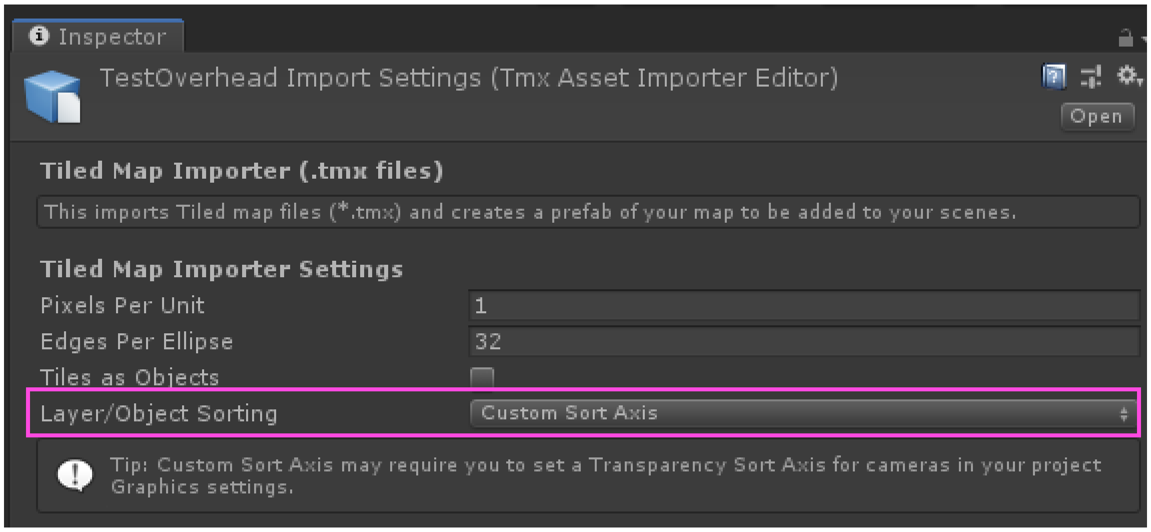


Fig. 3: The `overhead` scene included with SuperTiled2Unity may serve as a useful guide for others trying to achieve this effect.



TestOverhead Import Settings (Tmx Asset Importer Editor)

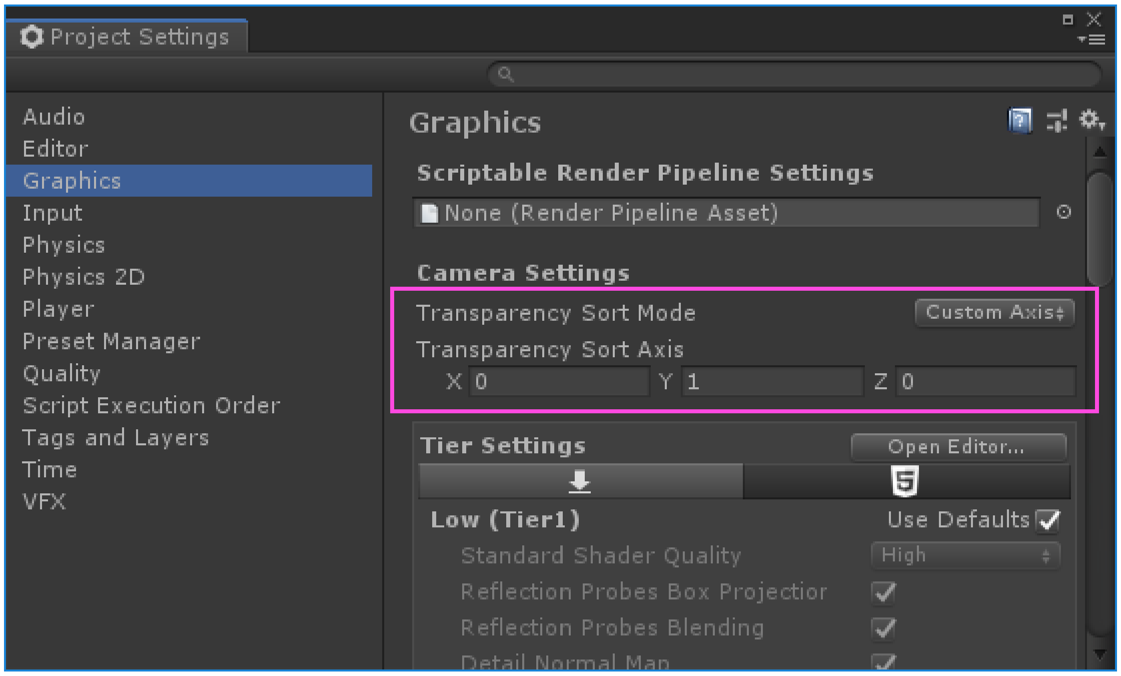
Tiled Map Importer (.tmx files)

This imports Tiled map files (*.tmx) and creates a prefab of your map to be added to your scenes.

Tiled Map Importer Settings

Pixels Per Unit	1
Edges Per Ellipse	32
Tiles as Objects	☐
Layer/Object Sorting	Custom Sort Axis

Tip: Custom Sort Axis may require you to set a Transparency Sort Axis for cameras in your project Graphics settings.



Scriptable Render Pipeline Settings: None (Render Pipeline Asset)

Camera Settings

Transparency Sort Mode: Custom Axis

Transparency Sort Axis: X 0 Y 1 Z 0

Tier Settings

Low (Tier1) Use Defaults ✓

Standard Shader Quality: High

Reflection Probes Box Projection ✓

Reflection Probes Blending ✓

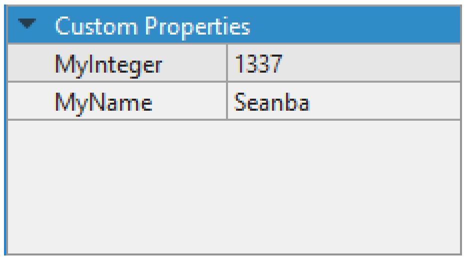
Detail Normal Map ✓

```
var camera = GameObject.FindGameObjectWithTag("MainCamera").GetComponent<Camera>();
camera.transparencySortMode = TransparencySortMode.CustomAxis;
camera.transparencySortAxis = Vector3.up;
```

CHAPTER 2

Custom Properties Support

The Tiled Map Editor has support for Custom Properties that allow you to include custom data or information along with components of your map.



Custom Properties	
MyInteger	1337
MyName	Seanba

Fig. 1: Tiled Custom Properties

These custom properties are also **supported by SuperTiled2Unity** and can be found on the `SuperCustomProperties` Monobehaviour component when imported in your Unity project.

2.1 Object Types Support

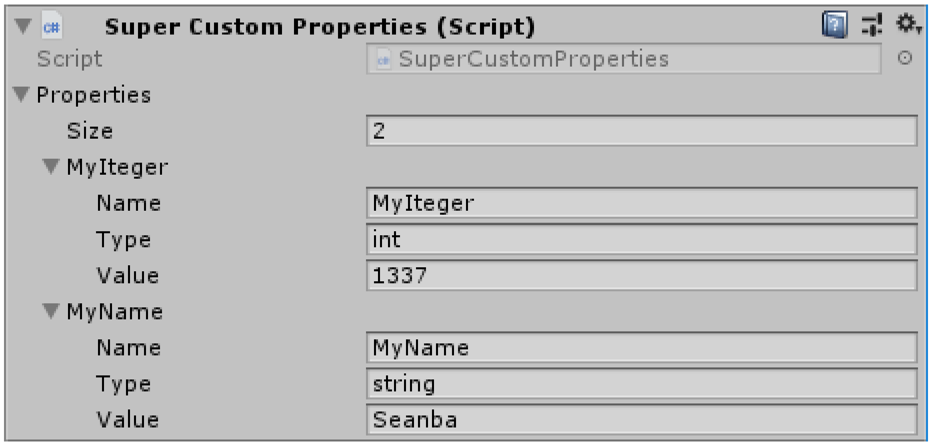
Tiled also has *predefined* properties that are described through the Object Types Editor.

This is a time-saving way to create classes or groups of properties. However, by default, SuperTiled2Unity has no way of being aware of these predefined properties. This can be resolved by **exporting the Object Types Xml file** to your Unity project.

First, select `Export Object Types...` from the `File` menu item.

This will bring up the save file dialog. Save your object types Xml file somewhere within your Unity project.

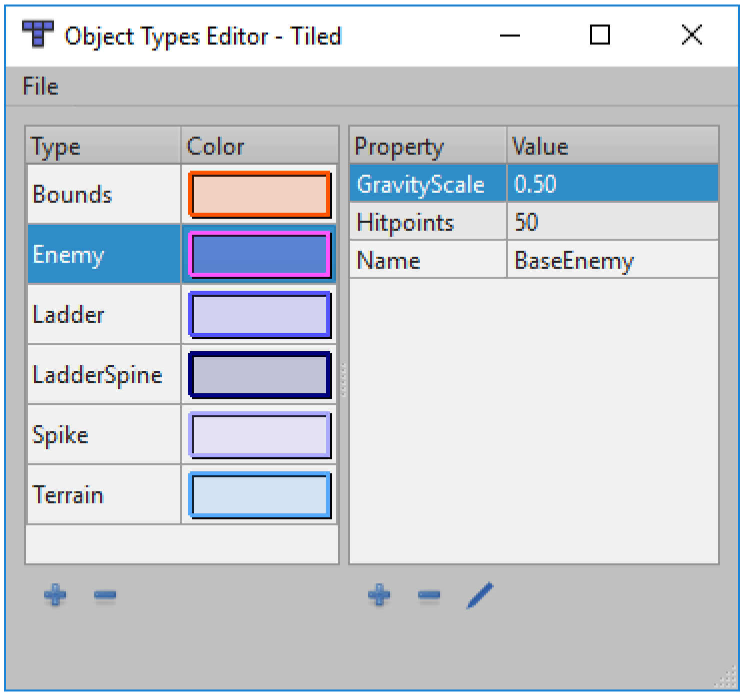
Tip: You can export your Object Types Xml file to any filename but make sure it is somewhere under your Unity project's `Assets` directory. This Xml file itself will need to be a Unity asset that is referenced by SuperTiled2Unity's



Super Custom Properties (Script)

Script	SuperCustomProperties
Properties	
Size	2
MyIteger	
Name	MyIteger
Type	int
Value	1337
MyName	
Name	MyName
Type	string
Value	Seanba

Fig. 2: SuperTiled2Unity Custom Properties



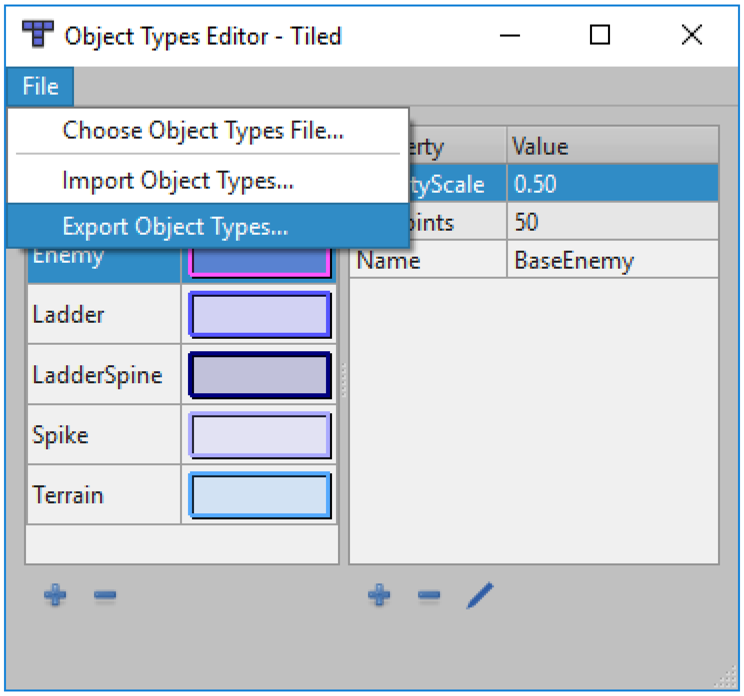
Object Types Editor - Tiled

File

Type	Color
Bounds	🟧
Enemy	🟦
Ladder	🟪
LadderSpine	🟪
Spike	⬜
Terrain	🟦

Property	Value
GravityScale	0.50
Hitpoints	50
Name	BaseEnemy

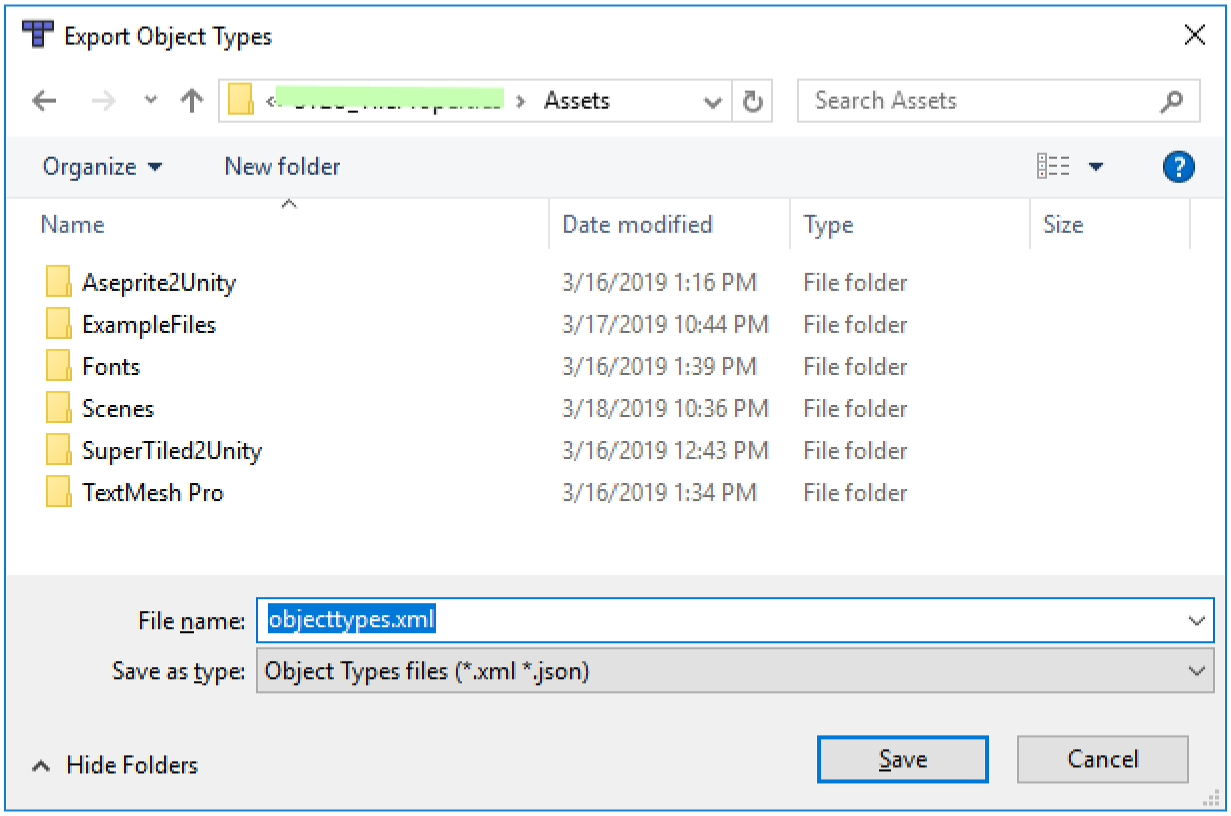
Fig. 3: Object Types Editor



Enemy
Ladder
LadderSpine
Spike
Terrain

erty	Value
tyScale	0.50
oints	50
Name	BaseEnemy

Fig. 4: Select **Export Object Types** from the File menu item



Name	Date modified	Type	Size
Aseprite2Unity	3/16/2019 1:16 PM	File folder	
ExampleFiles	3/17/2019 10:44 PM	File folder	
Fonts	3/16/2019 1:39 PM	File folder	
Scenes	3/18/2019 10:36 PM	File folder	
SuperTiled2Unity	3/16/2019 12:43 PM	File folder	
TextMesh Pro	3/16/2019 1:34 PM	File folder	

File name: objecttypes.xml
Save as type: Object Types files (*.xml *.json)

Fig. 5: Save Object Types Xml file to your Unity Project

settings.

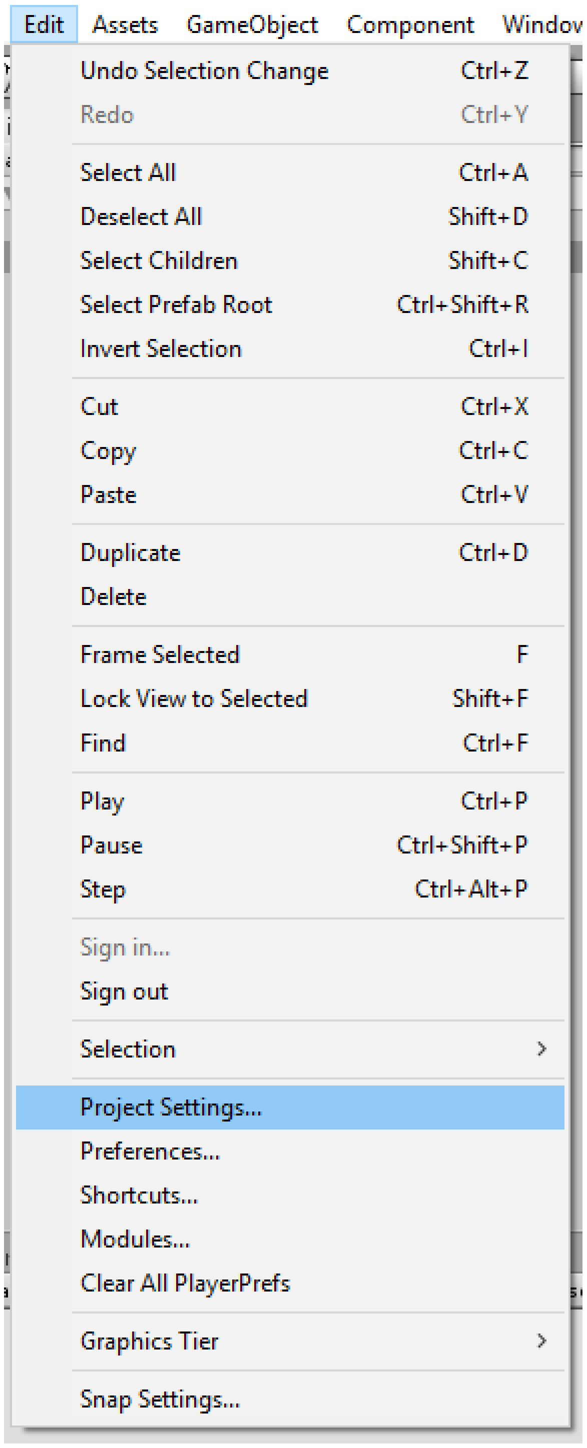
We now need to make SuperTiled2Unity aware of this exported Object Types Xml file through the `SuperTiled2Unity Project Settings`. These settings are found with your other project-wide settings (audio, graphics, etc.) through the `Edit -> Project Settings...` menu item.

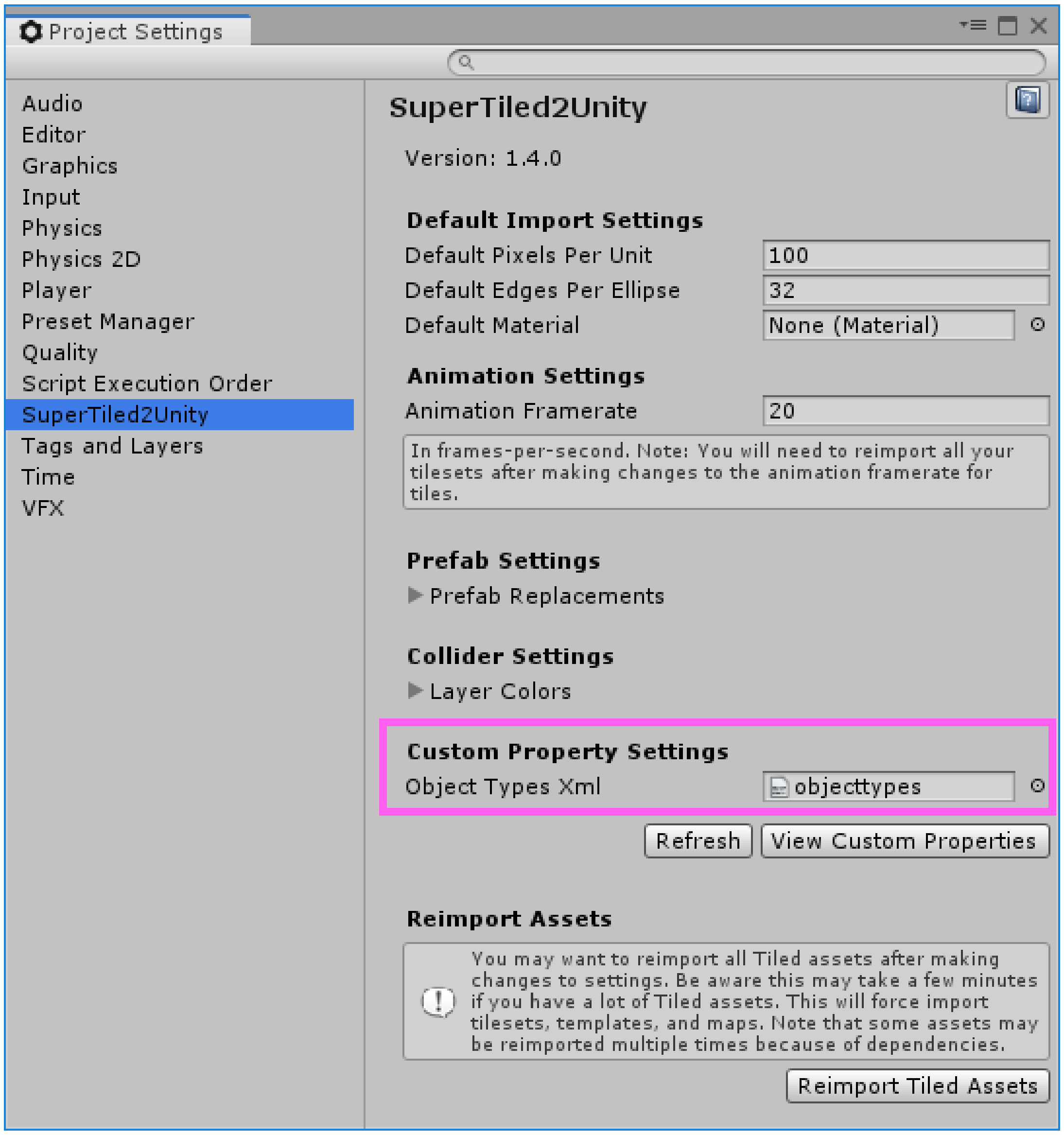
In the settings window you should see a field for `Object Types Xml`. Either drag and drop your recently exported Object Types Xml filed into this field or use the asset selector button to select the asset.

With the Object Types Xml file now set hit the `View Custom Properties` button just below. This will display the `Custom Object Types Properties` window which lists all the custom object types that were imported as well as their custom properties (if any) and custom color.

Now, any in your Unity project that are updated should have these predefined properties in the appropriate `SuperCustomProperties` instances.

Warning: Note that SuperTiled2Unity does not automatically update map assets when changes to the Object Types Xml are made. See the `Reimport Tiled Assets` button in the `SuperTiled2Unity Project Settings` inspector if you want to update all Tiled assets in your Unity project. **This may take some time** depending on the number and complexity of your Tiled assets, however.


Edit
Assets
GameObject
Component
Windov
Undo Selection Change Ctrl+Z
Redo Ctrl+Y
Select All Ctrl+A
Deselect All Shift+D
Select Children Shift+C
Select Prefab Root Ctrl+Shift+R
Invert Selection Ctrl+I
Cut Ctrl+X
Copy Ctrl+C
Paste Ctrl+V
Duplicate Ctrl+D
Delete
Frame Selected F
Lock View to Selected Shift+F
Find Ctrl+F
Play Ctrl+P
Pause Ctrl+Shift+P
Step Ctrl+Alt+P
Sign in...
Sign out
Selection
Project Settings...
Preferences...
Shortcuts...
Modules...
Clear All PlayerPrefs
Graphics Tier
Snap Settings...



SuperTiled2Unity

Version: 1.4.0

Default Import Settings

Default Pixels Per Unit 100

Default Edges Per Ellipse 32

Default Material None (Material)

Animation Settings

Animation Framerate 20

In frames-per-second. Note: You will need to reimport all your tilesets after making changes to the animation framerate for tiles.

Prefab Settings

Collider Settings

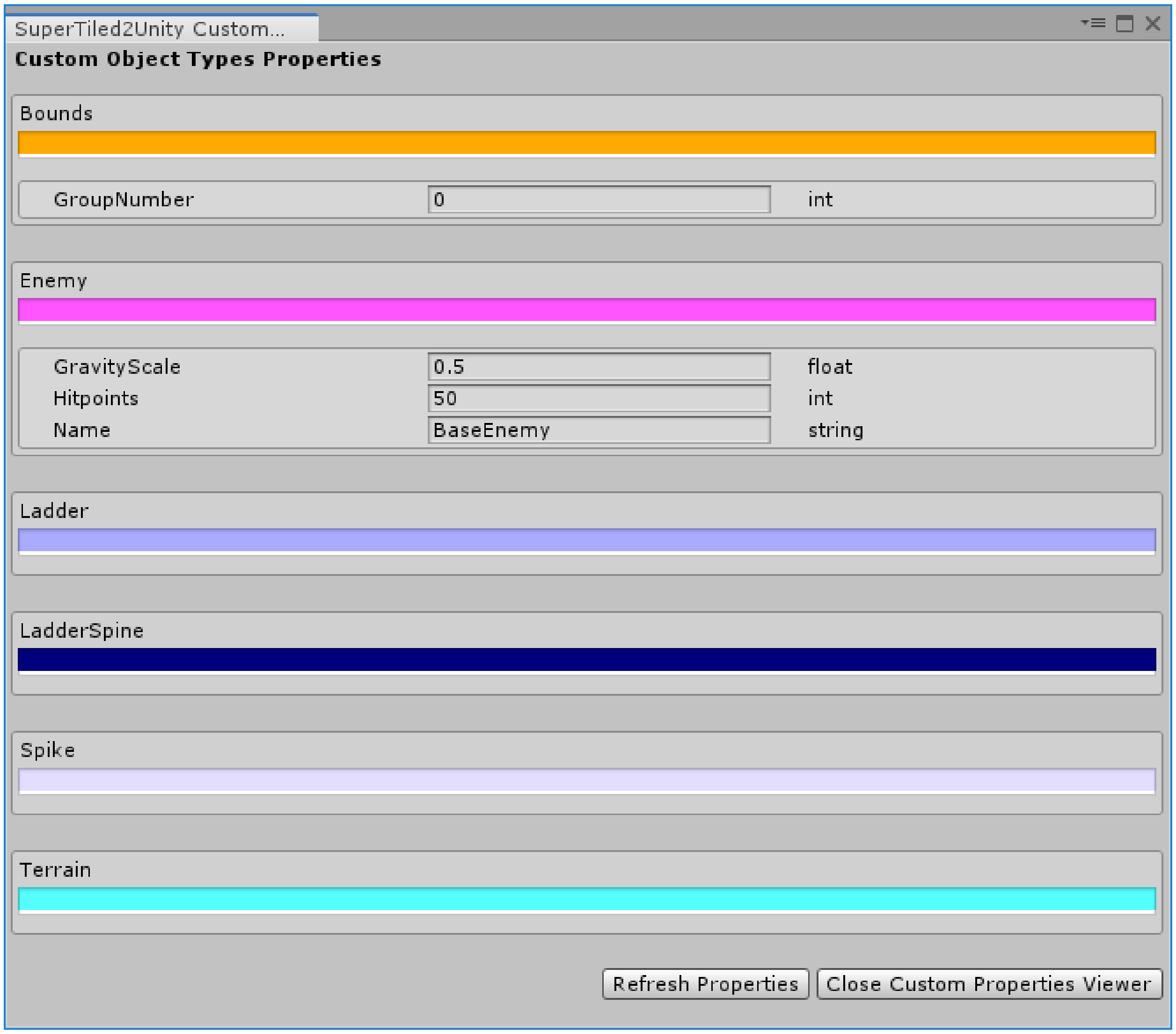
Custom Property Settings

Object Types Xml objecttypes

Reimport Assets

You may want to reimport all Tiled assets after making changes to settings. Be aware this may take a few minutes if you have a lot of Tiled assets. This will force import tilesets, templates, and maps. Note that some assets may be reimported multiple times because of dependencies.

Fig. 6: Drag and drop your exported object types Xml file into the **Object Types Xml** field



Custom Object Types Properties

Bounds

GroupNumber	0	int

Enemy

GravityScale	0.5	float
Hitpoints	50	int
Name	BaseEnemy	string

Ladder

LadderSpine

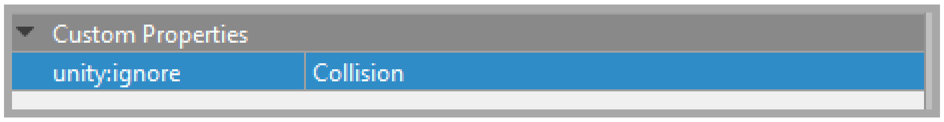
Spike

Terrain

Fig. 7: Custom properties for object types are displayed in this window.

CHAPTER 3

Custom Properties for Importing



Custom Properties

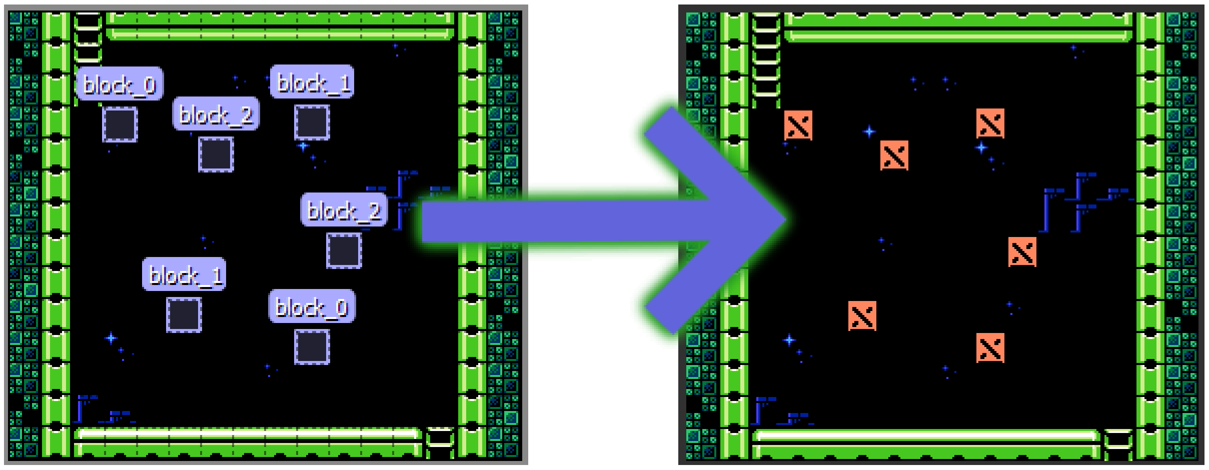
unity:ignore	Collision

You can customize how some portions of your Tiled files are imported into Unity prefabs through special custom properties. These custom properties always have a `unity:` prefix so that they do not collide with your own custom properties.

Property Name	Value Type	Used On	Purpose
`unity:ignore`	`string`	Layers	Portions of the Tiled layer can be ignored during the import process. This is useful if you tiles to appear for a layer but not the colliders, for instance. **Acceptable values:** • **False**: The layer is imported in full (default) • **True**: The layer is fully ignored (same result as having the layer invisible in Tiled) • **Visual**: The tiles are not imported for the layer. • **Collision**: The colliders are not imported for the layer.
`unity:IsTrigger`	`bool`	Layers, Tileset, Tile, Collider, Collider Group	Controls the `isTrigger` setting of the `Collider2D` object generated in the prefab. Add as a layer custom property to override all colliders for a given layer.
`unity:layer`	`string`	Layers, Tileset, Tile, Collider, Collider Group	Controls which Unity phyisics layer our generated colliders are assigned. The layer name must exist in your project's Tag Manager.
`unity:SortingLayer`	`string`	Layers, Tile Objects	Controls which sorting layer is assigned to the Unity `Renderer` component created for your tilemaps and sprites. The sorting layer name must exist in your project's Tag Manager.
`unity:SortingOrder`	`int`	Layers, Tile Objects	Controls which sorting order is applied to the Unity `Renderer` component created for your tilemaps and sprites.
`unity:Tag`	`string`	Layers, Objects	Controls which tag is applied to the `GameObject` created for your layers and

CHAPTER 4

Extending The SuperTiled2Unity Importer



SuperTiled2Unity strives to build prefabs out of your Tiled maps (*.txm files) with minimal input from users. This is achieved through Scripted Importers which was first added with Unity 2017.

Every game is different, however, and for projects of some complexity you'll want to extend the SuperTiled2Unity import pipeline to fit your needs.

SuperTiled2Unity gives you two ways to modify generated prefabs. Once hooked up these modifications will become an itegral part of the import process. This means the same modifications will applied to your maps, automatically, every time you resave in Tiled.

Tip: If you find you have to modify your imported Tiled maps by hand after every save then this is for you. Automating the pipeline saves you time and frustration.

SuperTiled2Unity provides two ways to modify the import pipeline:

- Prefab Replacements
- Custom Importers

4.1 Prefab Replacements

Prefab Replacements are the easiest way to modify your imported maps. In essence, they replace a `Tiled Object` in your map with a `Prefab Instance` based on a prefab in your project.

This Github Repo explains Prefab Replacements in greater detail and provides a working example.

Tip: **Prefab Replacements** are ideal for spawners. Also, any custom properties on your `Tiled Object` with a name that matches a field in your prefab's components will be automatically applied.

4.2 Custom Importers

Custom Importers are the most powerful way to modify your imported prefabs. They give you a place (in code) in the import process where you can completely transform the generated prefab. This power does come at a cost though as it requires you to write some code. This may be intimidating to people new to computer programming but if your project requires the kind of specialization affored by custom importers then it is highly recommended you invest the time needed to up your coding skills.

```
// The AutoCustomTmxImporterAttribute will force this importer to always be applied.
// Leave this attribute off if you want to choose a custom importer from a drop-down␣
↪list instead.
[AutoCustomTmxImporter()]
public class MyTmxImporter : CustomTmxImporter
{
    public override void TmxAssetImported(TmxAssetImportedArgs args)
    {
        // Note: args.ImportedSuperMap is the root of the imported prefab
        // You can modify the gameobjects and components any way you wish here
        // Howerver, the results must be deterministic (i.e. the same prefab is␣
↪created each time)
        var map = args.ImportedSuperMap;
        Debug.LogFormat("Map '{0}' has been imported.", map.name);
    }
}
```

This Github Repo has a working example of a custom importer (named `TrackCustomImporter`) in action. It takes a specially-marked `GameObject` and moves it along a track that was drawn in a Tiled map.

Tip: **Custom Importers** requires programming but they provide the ideal way to bend the import pipeline to your will.