Manual to the

2D Castle Pack - tiles and props

“Just Place and Stretch”

Summary

1. What is it?
2. How to use it?
3. Sprites
    - Basic colored
    - Blurred
    - Grayscale
    - Shader textures
4. Shaders
5. Prefabs
6. Scripts
    - AppManager
    - Object Pooling
    - Cloud Spawner
    - Snapshot Saver
7. Demo

Before you begin

First of all you need to install **Universal Render Pipeline** for this package to work correctly.

Installing URP

1. In Unity, open your Project.
2. In the top navigation bar, select **Window > Package Manager** to open the **Package Manager** window.
3. Select the **All** tab. This tab displays the list of available packages for the version of Unity that you are currently running.
4. Select **Universal RP** from the list of packages.
5. In the bottom right corner of the Package Manager window, select **Install**. Unity installs URP directly into your Project.

Configuring URP

Before you can start using URP, you need to configure it. To do this, you need to add a Scriptable Render Pipeline Asset to your Graphics settings.

Adding the Asset to your Graphics settings

To use URP, you need to add the Universal Render Pipeline Asset to your Graphics settings in Unity. If you don't, Unity still tries to use the Built-in render pipeline.

To add the Universal Render Pipeline Asset to your Graphics settings:

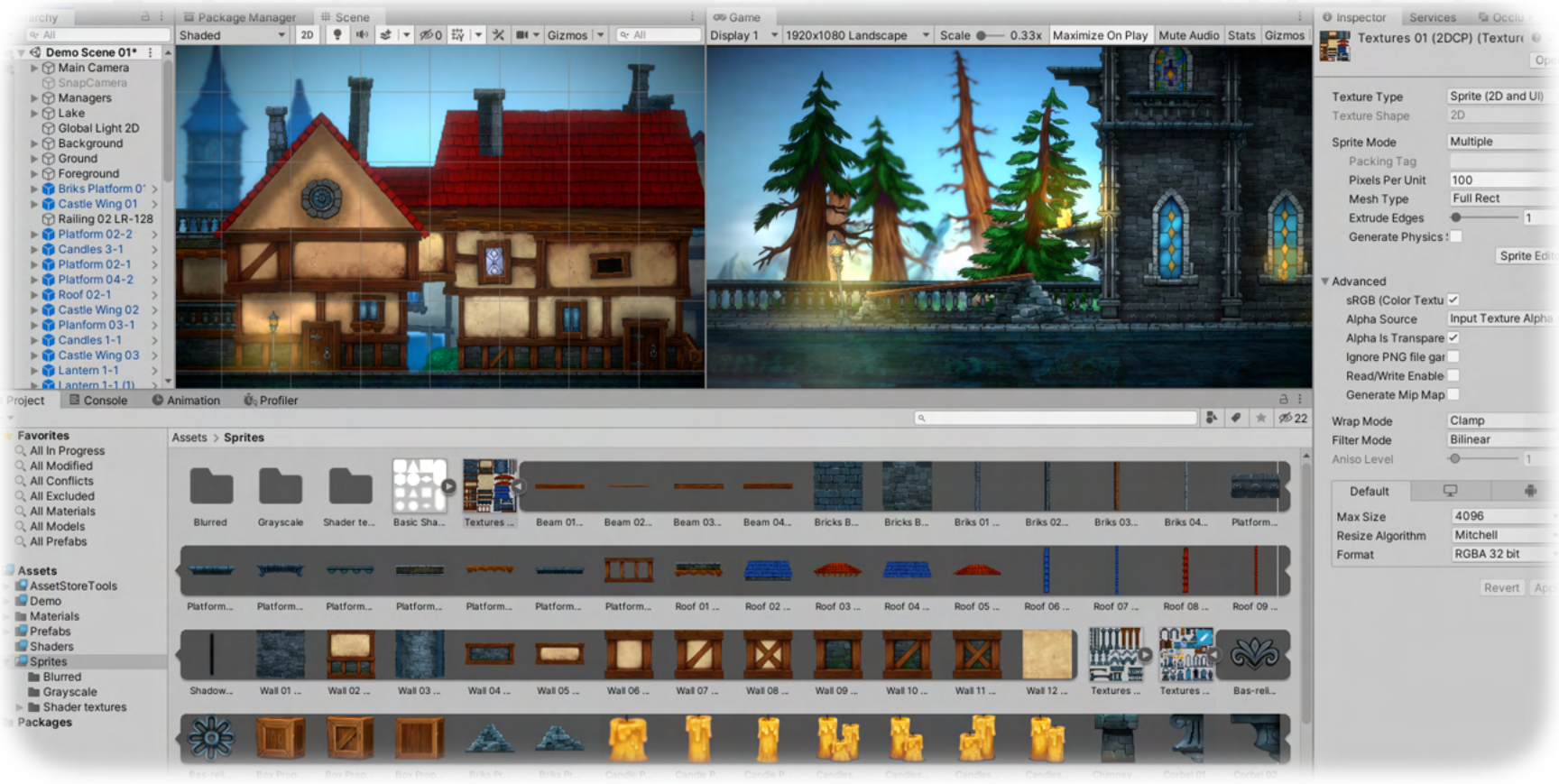
1. Navigate to **Edit** > **Project Settings...** > **Graphics**.
2. In the **Scriptable Render Pipeline Settings** field, add the Universal Render Pipeline Asset located in the folder **2DCP** > **Demo** > **Rendering**. When you add the Universal Render Pipeline Asset, the available Graphics settings immediately change. Your Project is now using URP.

What is it?

2D Castle Pack - tiles and props it's a set of **high definition seamless stylized sprites** of walls, roofs, doors, windows and a lot of other props to build castles, buildings and cities for your games.

All textures are packed into **compact** and fully **configured spritesheets** that greatly **improves the performance** of your projects.

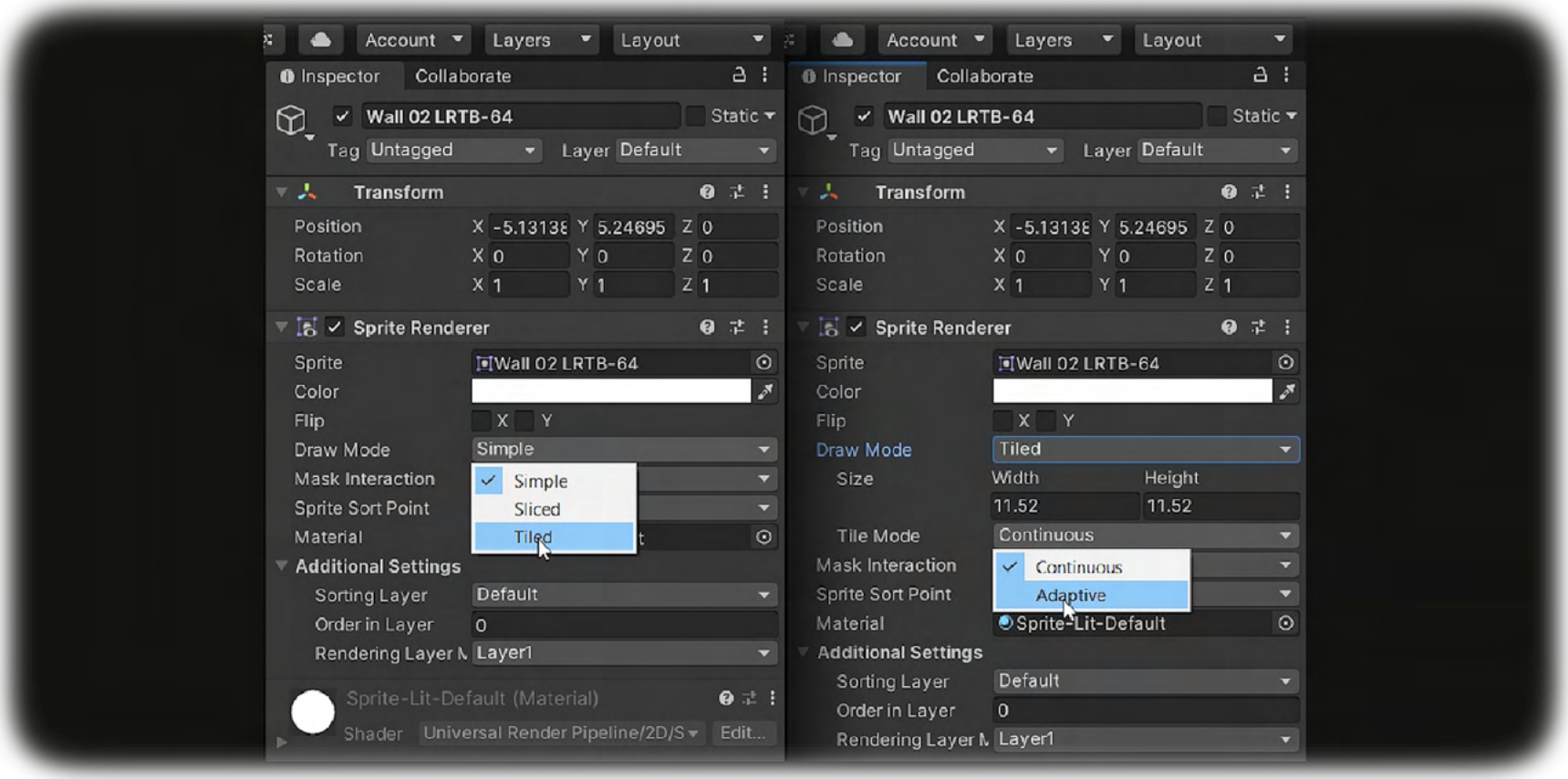
The set also contains some **shaders** and **scripts** that will simplify and speed up your work on level design.



How to use it?

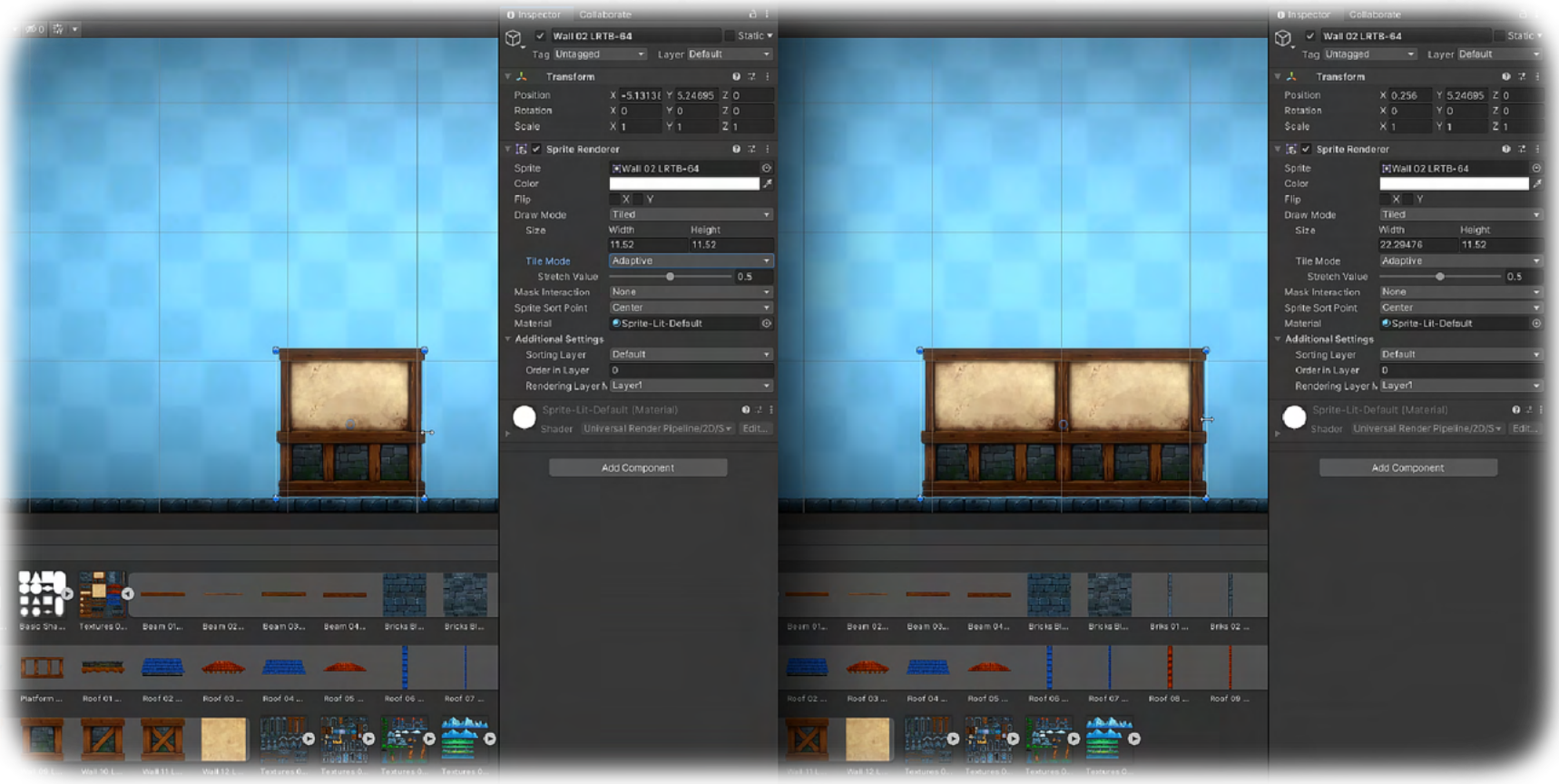
Place the seamless sprite on a scene

Set **Draw Mode** to **Tiled**



(you can choose between Continuous and Adaptive tile modes)

Stretch the sprite using Rect Tool (T shortcut)

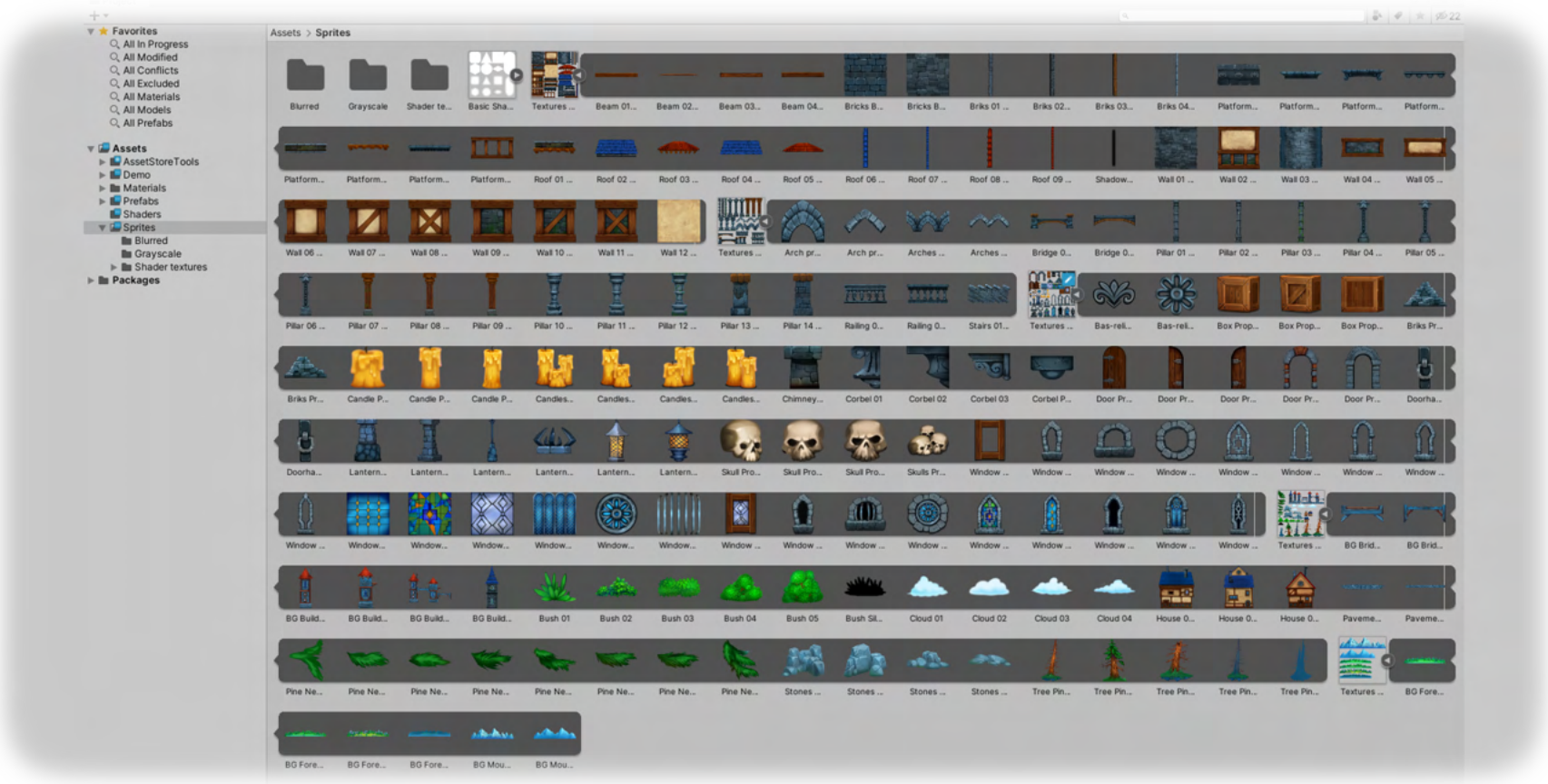


or set a **Width** and a **Height** values manually to change the texture look and size

Sprites

This sprite pack includes:

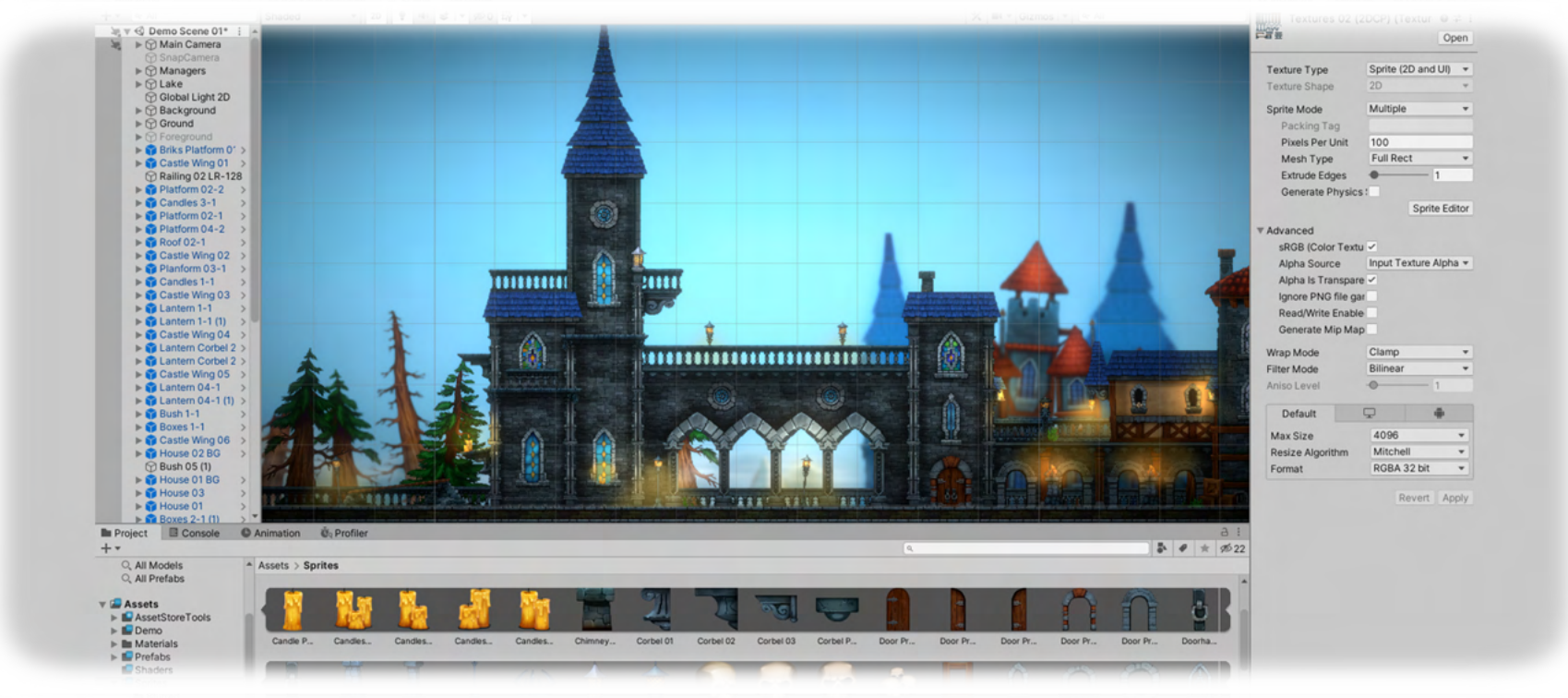
- **Over 180** high quality **sprites** packed into **6 spritesheets** in **PNG** format
- Three **blurred versions** of each sprite (blur radius 5px, 10px and 15px)
- **Grayscale version** of each sprite (for easy color variations of sprites with a low memory footprint)
- **Over 80 prefabs** of Buildings, Constructions, Platforms, Doors, Windows, Lanterns, Bushes, Trees and a lot of other props



Sprites - Basic colored textures

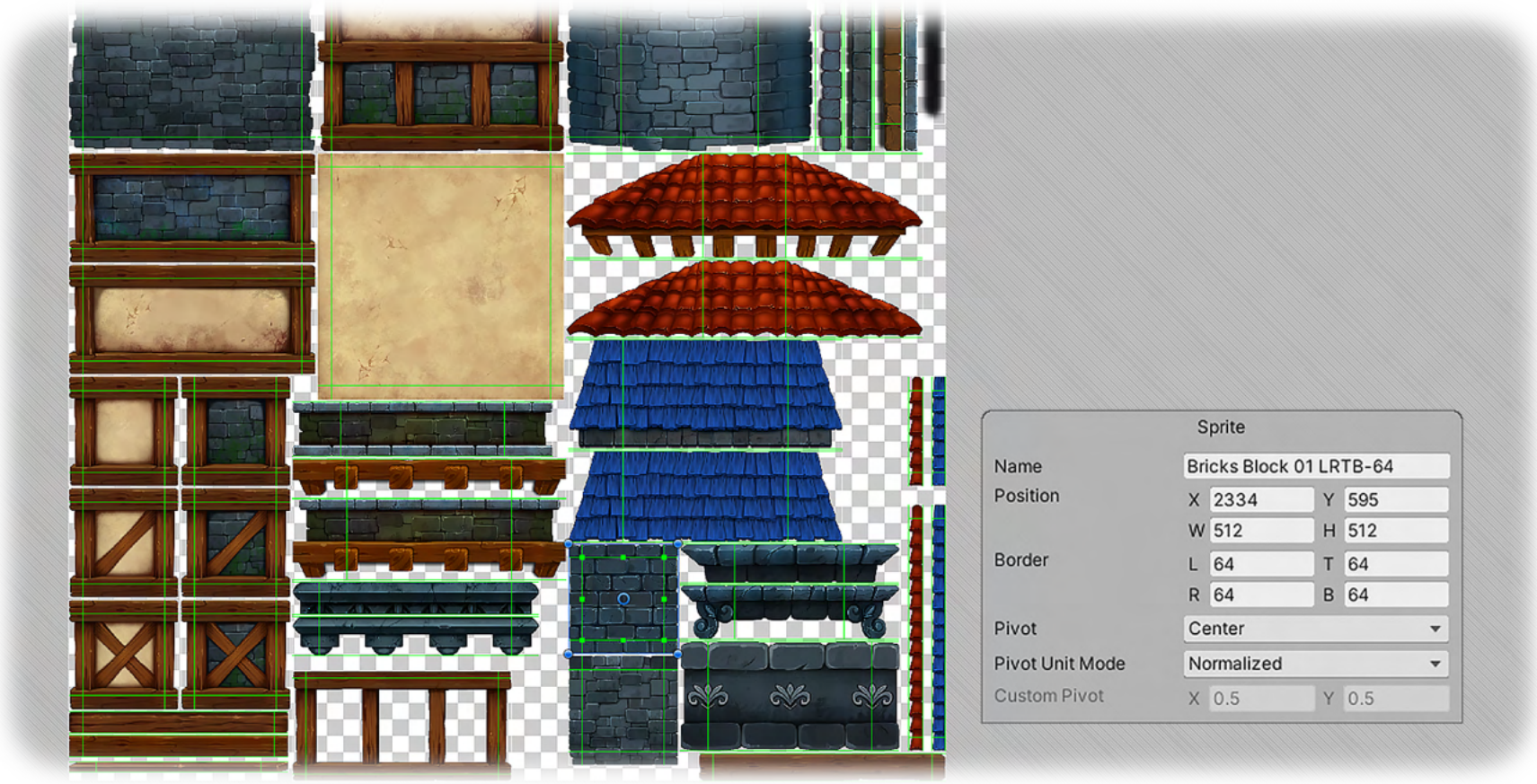
You can find **6 spritesheets** located in the **2DCP** > **Sprites** folder

- **Textures 01 (2DCP)** - contains seamless sprites of walls, roofs, bricks, platforms and wooden beams
- **Textures 02 (2DCP)** - contains seamless sprites of arches, bridges, pillars, railings and stairs
- **Textures 03 (2DCP)** - contains a lot of props as doors, windows, lanterns, candles, boxes, bricks and other
- **Textures 04 (2DCP)** - contains trees, bushes, vegetation, clouds, stones, background buildings and constructions
- **Textures 05 (2DCP)** - contains seamless sprites of woodlands and mountain ranges for creating of background
- **Basic Shapes (2DCP)** - contains the simple geometric shapes and their blurred versions (useful for masks, shadows and lights)



All sprites from spritesheets have their fully **setted names** and **tile borders**. If the sprite has **label** like **“LRTB-128”** or **“LR-64 TB-128”** in the name, it's mean **this sprite is seamless** and can be draw in tiled mode

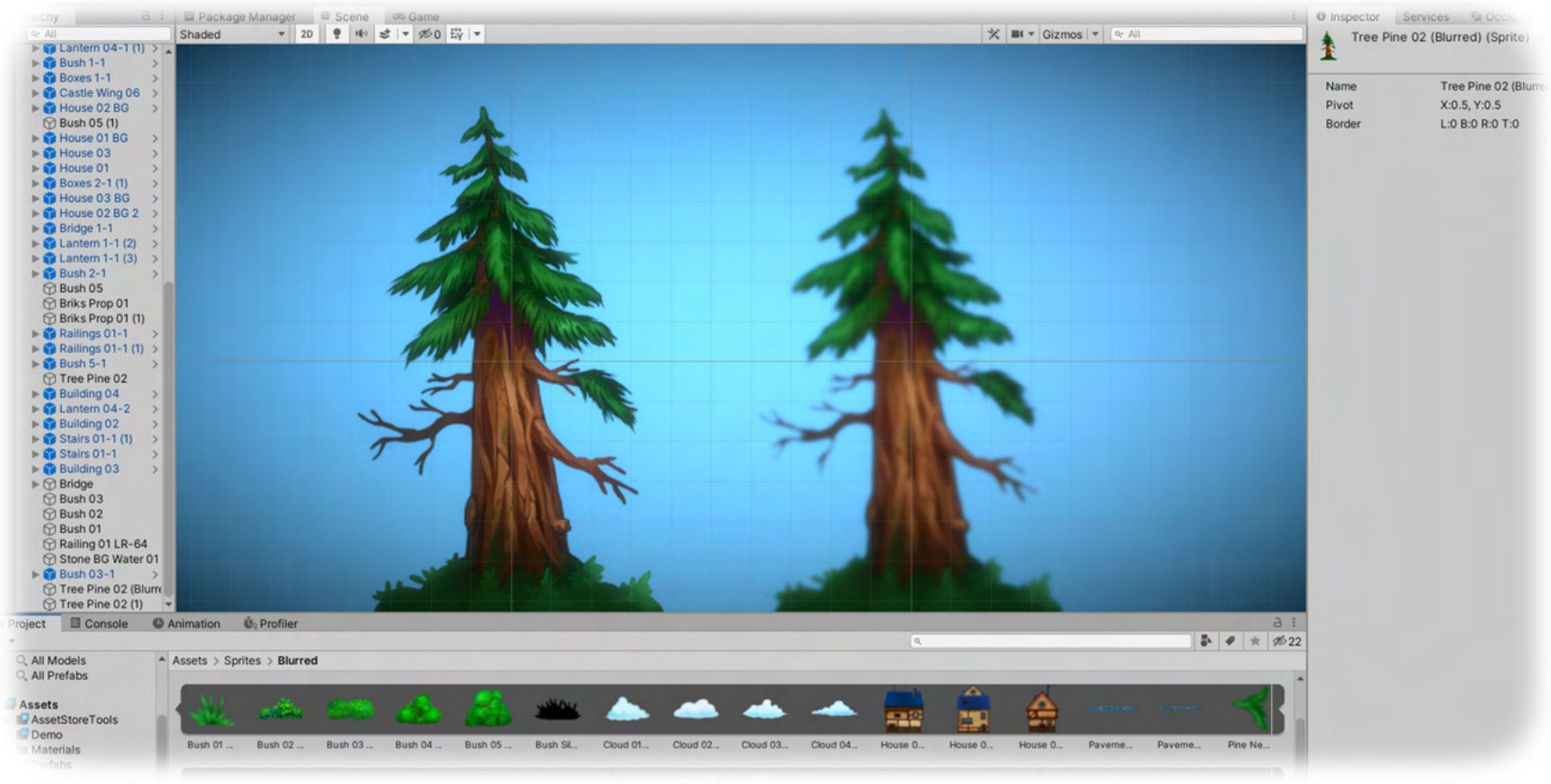
- **horizontally** if has **LR** label (indicates the value of the **Left** and the **Right** borders)
- **vertically** if has **TB** label (indicates the value of the **Top** and the **Bottom** borders)



The numbers after the dash in the name of sprite, for example **Bricks Block 01 LRTB-64**, shows the margin value of the border from the edge. **LRTB-64** means the left, right, top and bottom borders of the sprite should be set to **64 px** (since it is at this distance from the edges of the sprite that the lines of seamlessness was drew)

Sprites - Blurred textures

In the folder **2DCP** > **Sprites** > **Blurred** you can find three blurred versions of each basic colored spritesheet. These blurred versions differ from each other by **blur radius** 5px, 10px and 15px.

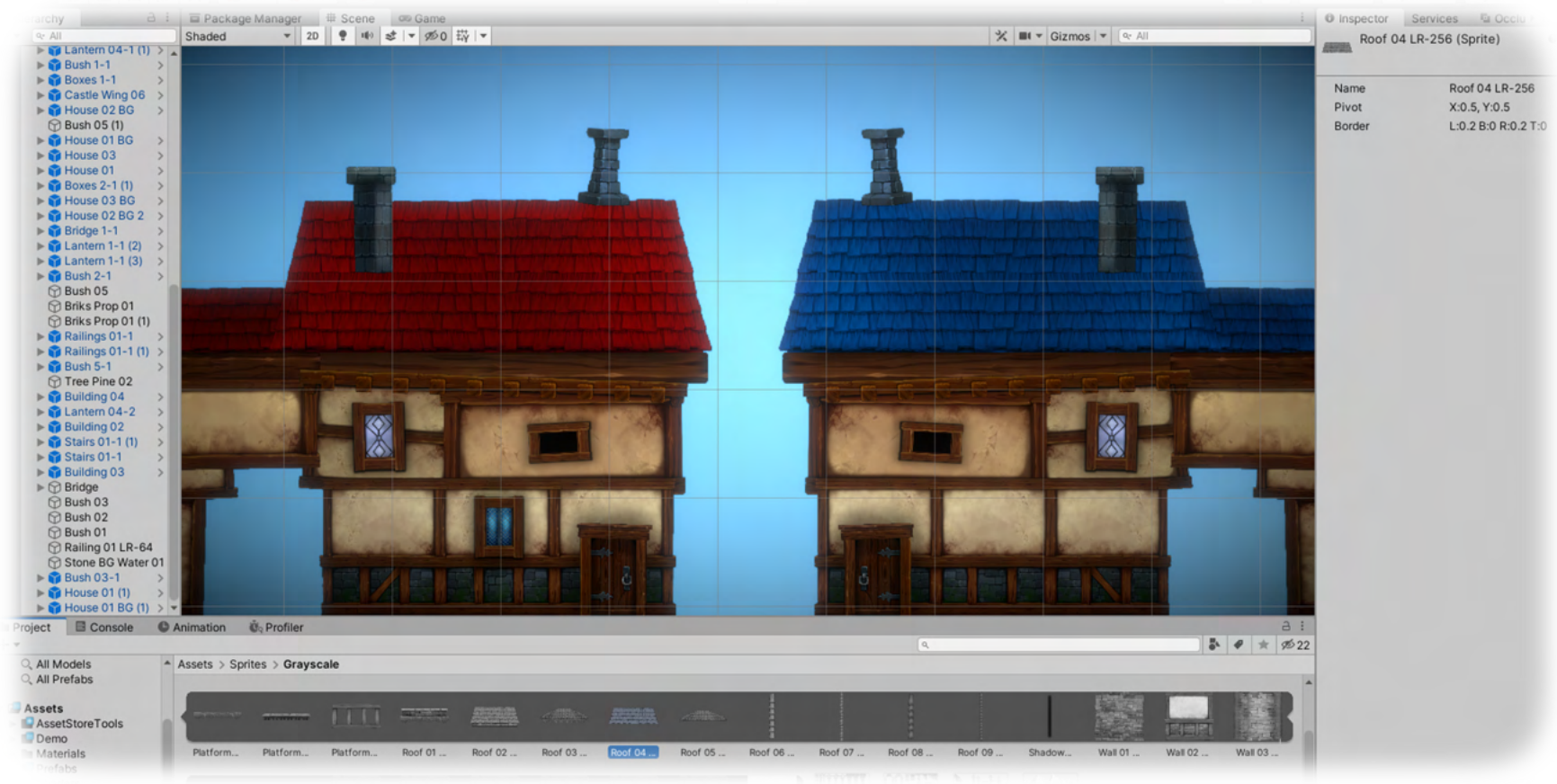


- All blurred sprites can be used for **creating** deep of field **effect manually**
- Also these blurred versions of the spritesheets are **used by the blur shader** Blur (2DCP) to create dynamic depth of field effect depends on the distance to the camera

Sprites - Grayscale textures

In the folder **2DCP** > **Sprites** > **Grayscale** you can find grayscale versions of each basic colored spritesheet. You can use these grayscale versions of sprites just like colored versions, but grayscale sprites have their advantages

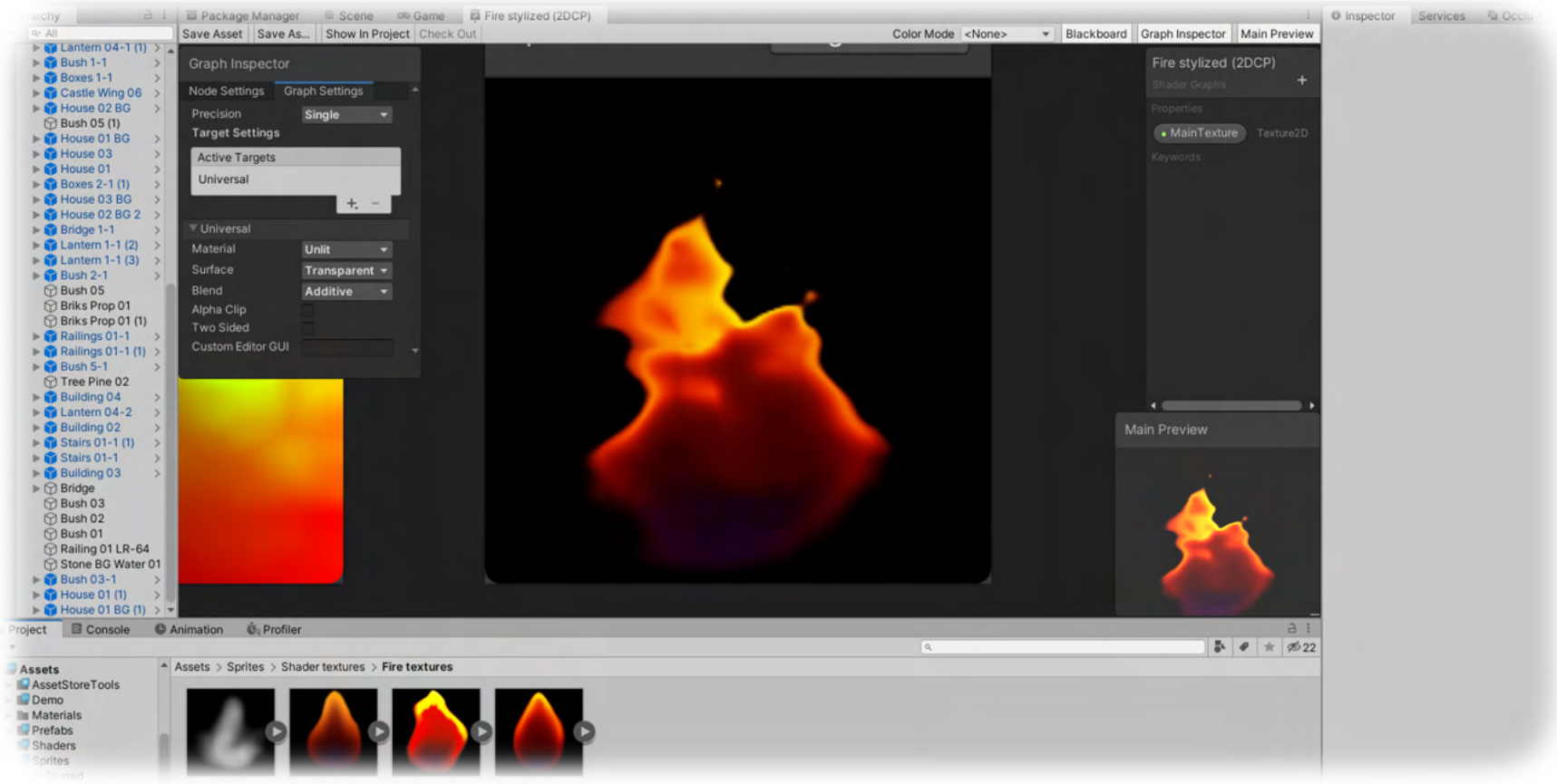
+ **Smallest** texture **size** (more optimized)
+ **Reusable** textures (a lot of color themes variations of each sprite)



Sprites - Shader textures

In the folder **2DCP** > **Sprites** > **Shader textures** you can find all images that used by shaders and particle systems in demo as like

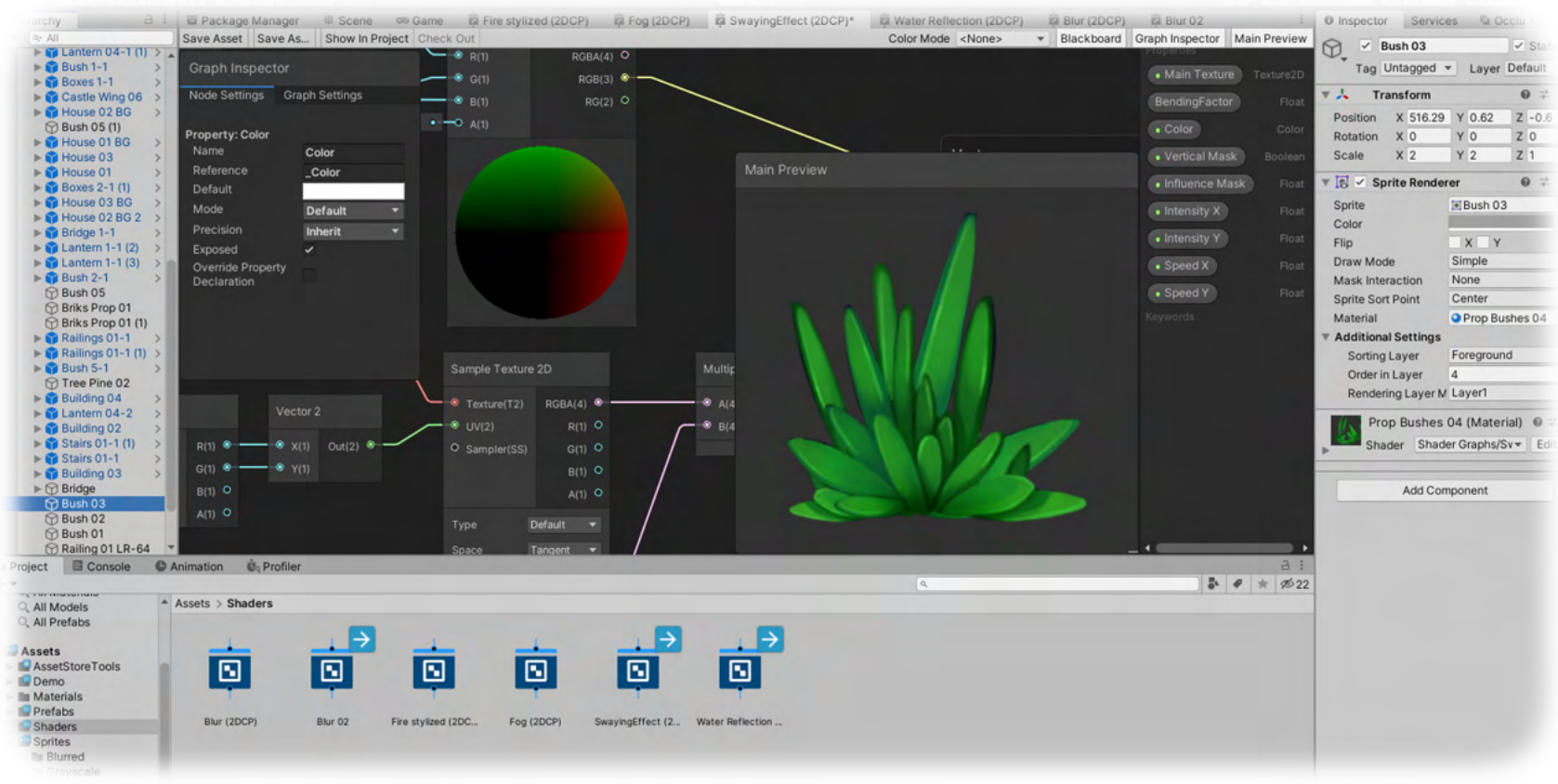
- **Fire textures** for **Fire stylized (2DCP)** shader
- **Water Render textures** for **Water Reflection (2DCP)** shader
- Textures for **masks** and **gradients**
- Placeholder images



Shaders

There are several useful shaders in the folder **2DCP** > **Shaders**

- **Blur (2DCP)** - replaces the sprite with its blurred version depending on the distance to the camera
- **Blur 02 (2DCP)** - creates texture sample multiple times with offset in different directions
- **Cloud stylized (2DCP)** - creates the effect of dynamic stylized clouds
- **Fire stylized (2DCP)** - just a beautiful stylized fire for your torches, lanterns and bonfires
- **Fog (2DCP)** - creates a couple of layers of floating fog
- **Swaying Effect (2DCP)** - distorts the texture horizontally and vertically, creating a swaying effect
- **Water Reflection (2DCP)** - creates water effects on the render texture captured from the camera

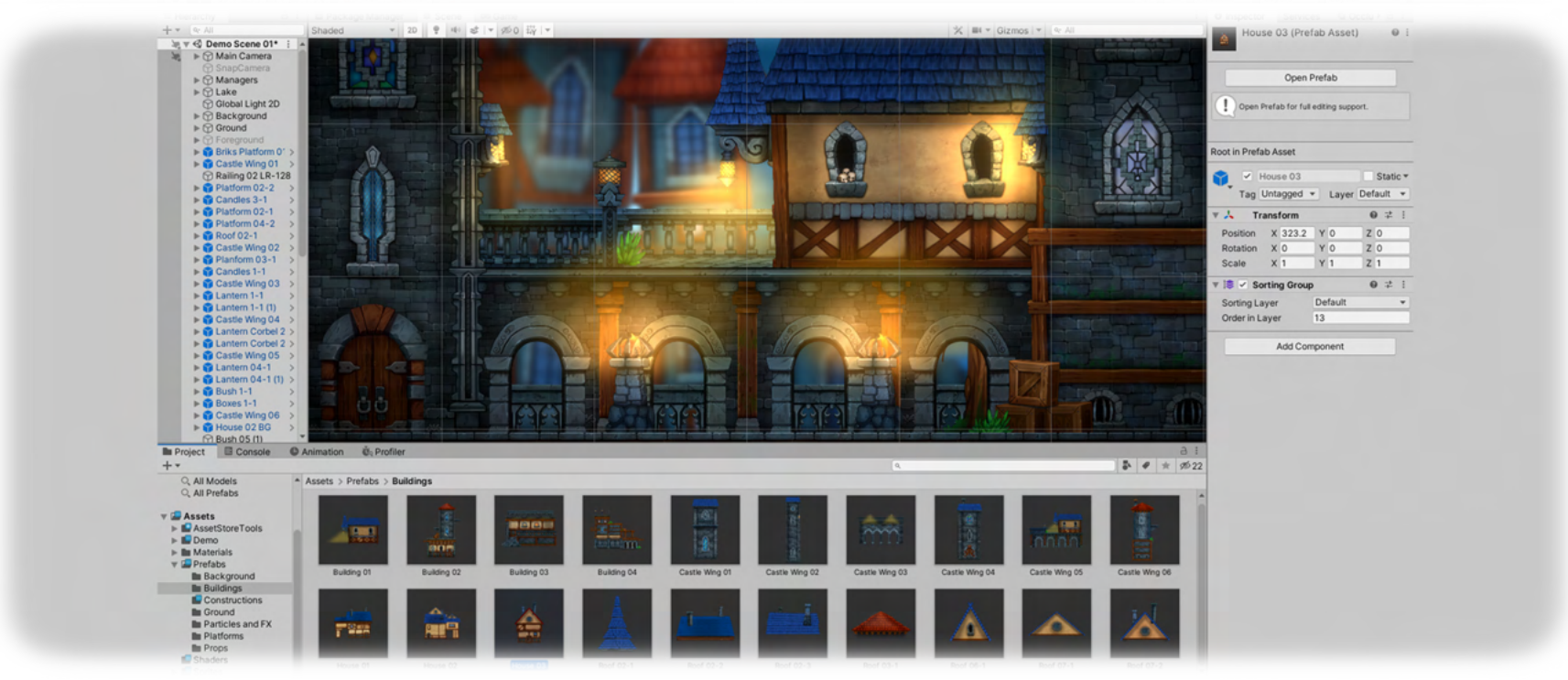


Prefabs

The **2DCP** > **Prefabs** folder contains many simple and complex objects composed of multiple sprites, masks, particles and scripts.

This folder contains **individual folders** for different **types** of objects

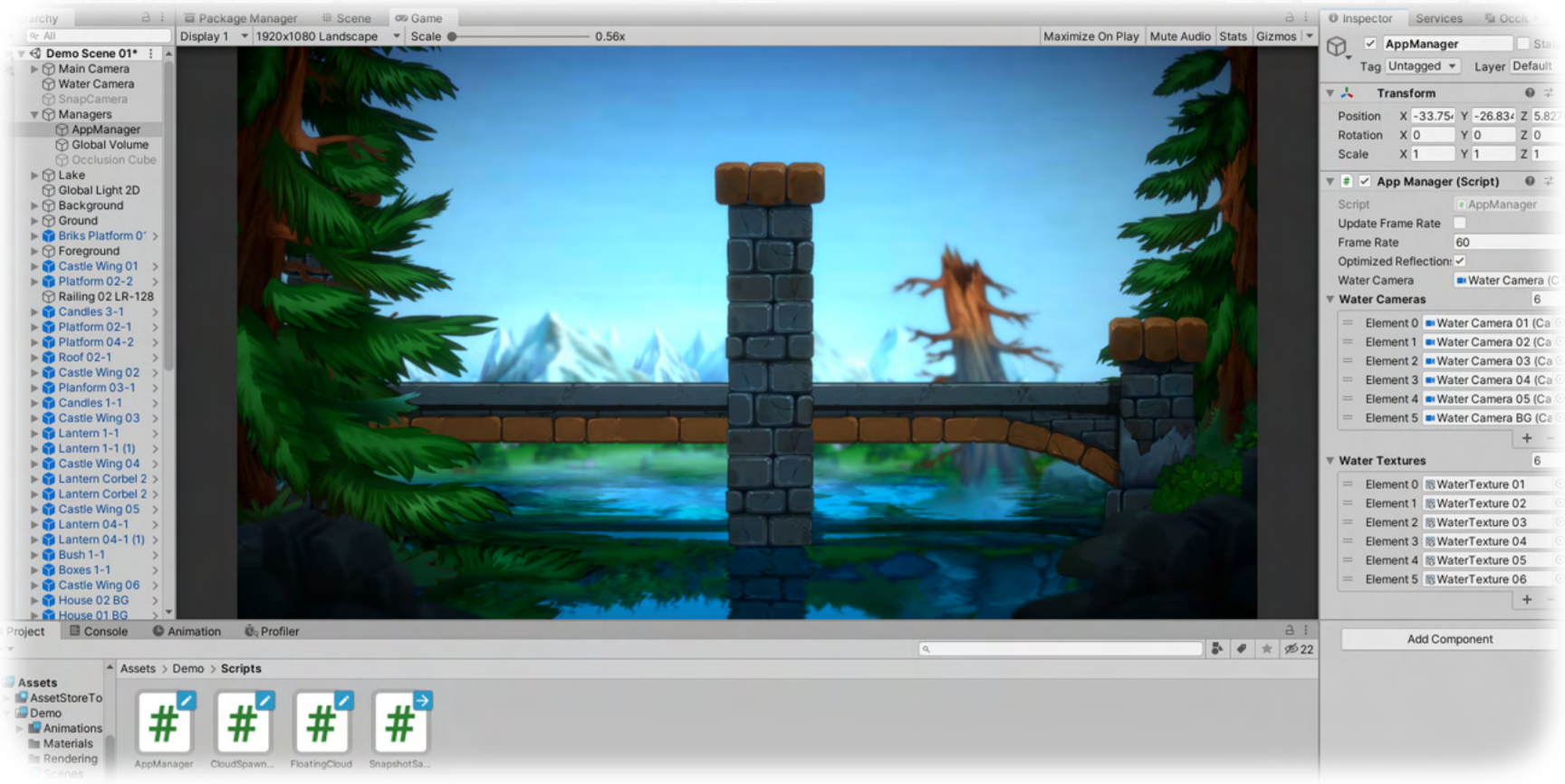
- Background elements
- Buildings
- Constructions
- Ground elements
- Particles and FX
- Platforms
- Props



Scripts

2DCP > **Demo** > **Scripts** folder contains some simple but useful tools that will make your level design more **handy**, **modular** and **flexible**

AppManager.cs - simple manager will allow you to **control** a **frame rate** of your app and **optimize** all water reflections (improves performance on mobile devices)



Object Pooling

Directory: **2DCP** > **Demo** > **Scripts** > **LDT** > **Fallencake** > **Tools** > **ObjectPool**

FCObjectPooler.cs - a basic pooler class, uses to be extended for **SingleTypeObjectPooler** and **MultiTypeObjectPooler** classes

FCSingleTypeObjectPooler.cs - a simple object pooler that can output only a **single type** of **objects**

FCMultiTypeObjectPooler.cs - a complex object pooler that can output multiple types of objects

FCPoolableObject.cs - this class should be added to the object that you need to be pooled from Object Pooler

How to Get a GameObject from the Pool ?

- Use the method **GetPooledGameObject()** to return a pooled gameobject

How to Destroy / Deactivate a pooled GameObject ?

- Use the method **Destroy()** to deactivate the instance to reuse it

Life Time parameter (**float lifeTime**) - the time, in seconds, that the pooled object will be active on scene. If set to any positive value, it'll be set inactive after that time. Value of 0 means the object will live forever.

MultiTypeObjectPooler Methods you can pull objects from the pool with:

- **SequentialOrder** - the pooler will get all object of current type before moving to the next type object

- **InTurnTypeBased** - objects will be spawned in the order has setted in the inspector (from top to bottom)

- **InTurnPriorityBased** - tries to get an object from the pool in order based on the type Priority value, probability to be get picked depends on the Priority value too (the larger the Priority value, the higher the probability it'll be chosen)

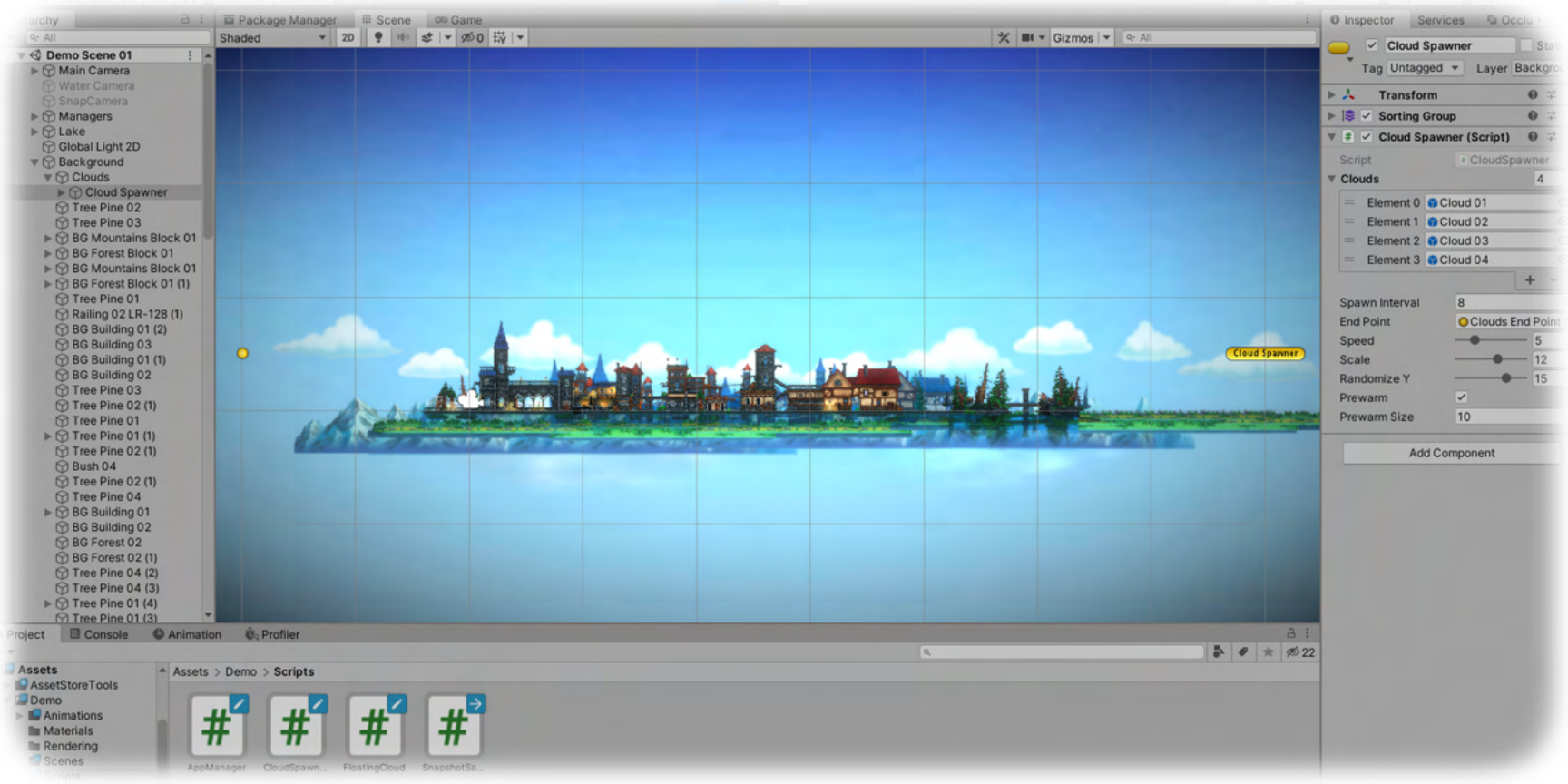
- **RandomPoolBased** - the pooler will get one object from the whole pool, at random, each object has equal chances to be chosen

- **RandomTypeBased** - randomly chooses the type of the object (the larger the amount of object of the specific type, the higher the chances it'll be chosen)

- **RandomPriorityBased** - randomly chooses one object from the pool, based on Priority value of its type probability (the larger the Priority, the higher the chances it'll be chosen)

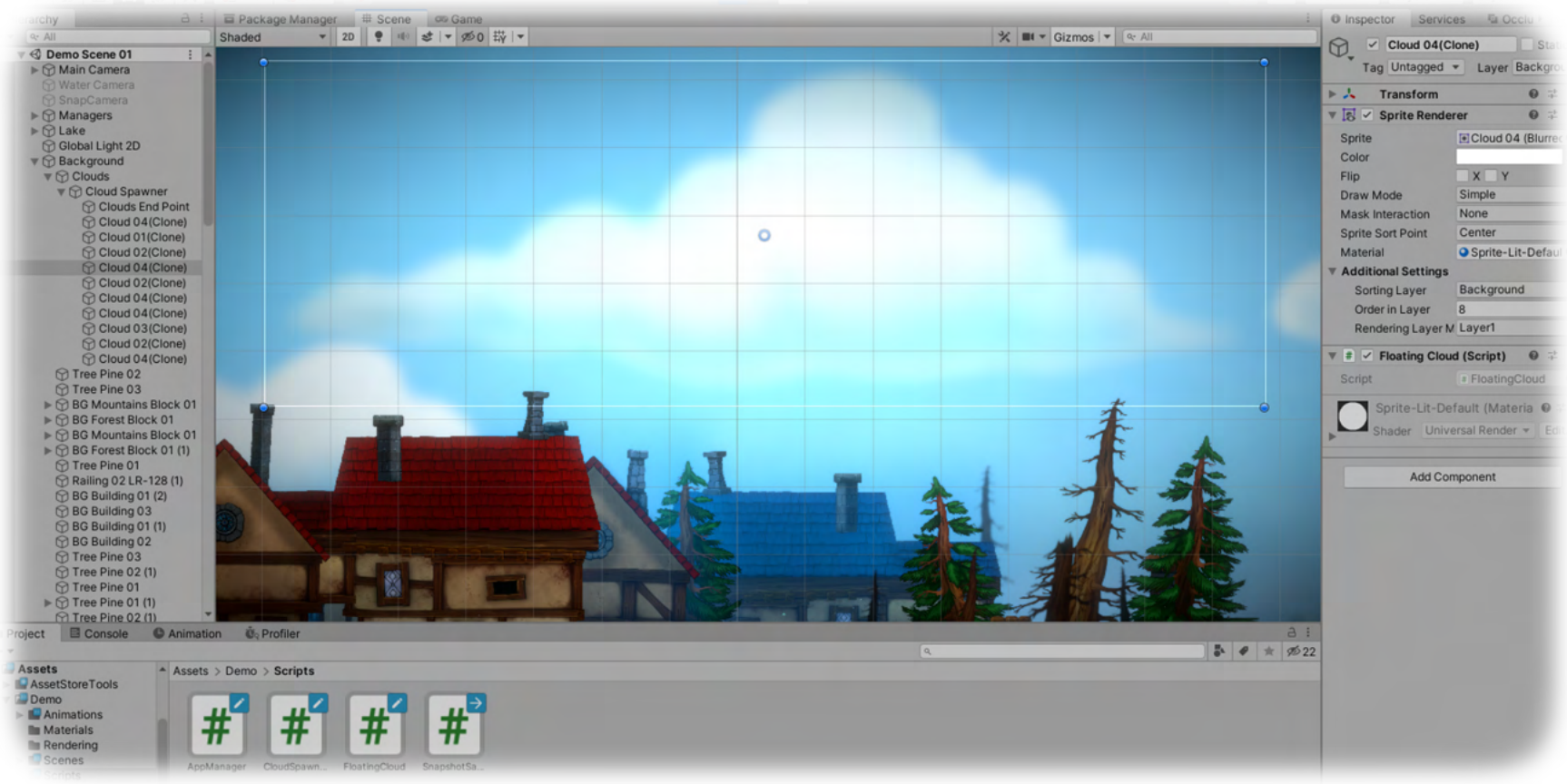
Cloud Spawner

CloudSpawner.cs - spawning system, based on the **FCMultiTypeObjectPooler** class. Allow you to spawn prefabs of floating clouds on the scene and control their speed, scale, grouping, spawn and unspawn points



Cloud Spawner

FloatingCloud.cs - a script, based on the **FCPoolableObject** class, that translating the cloud gameobject at a specific speed and unspawning it at the endpoint

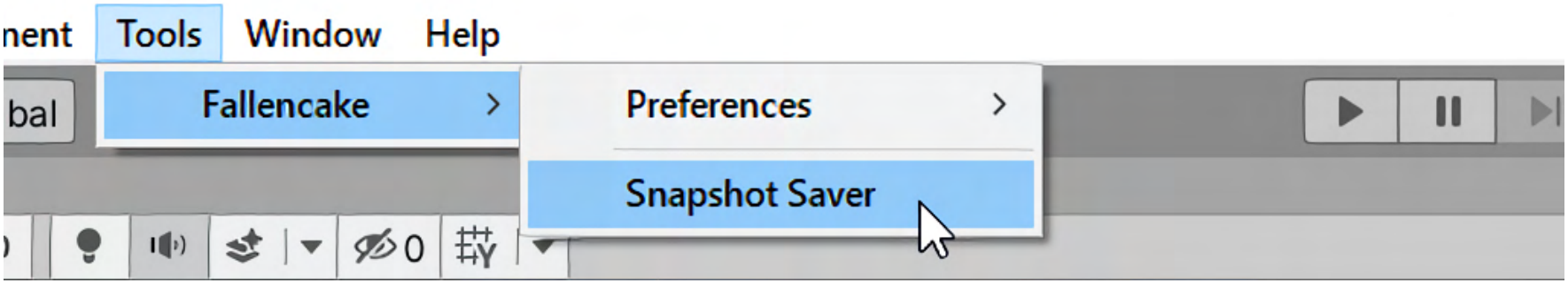


Material of the clouds based on the **Cloud stylized (2DCP)** shader.

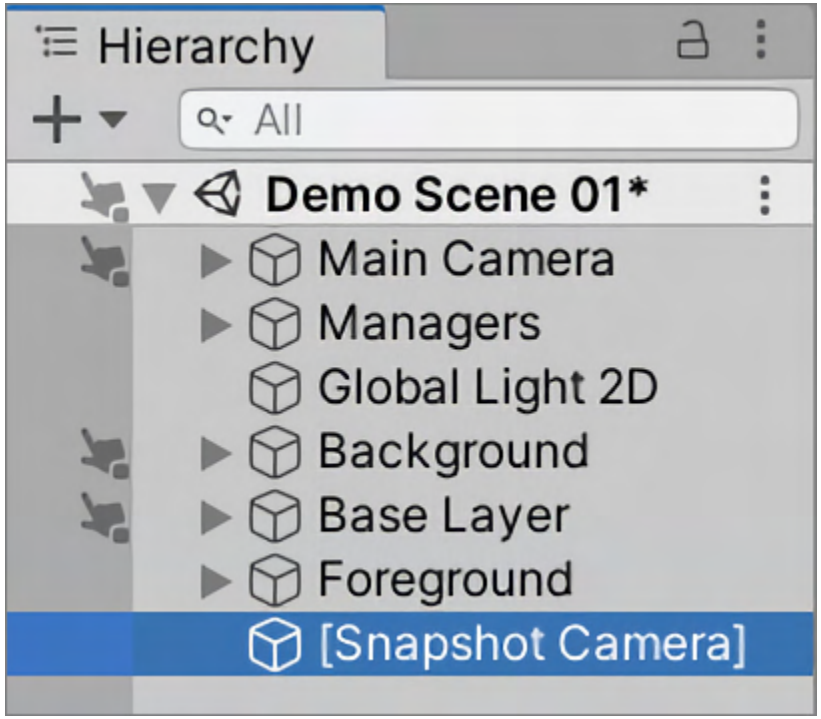
Snapshot Saver

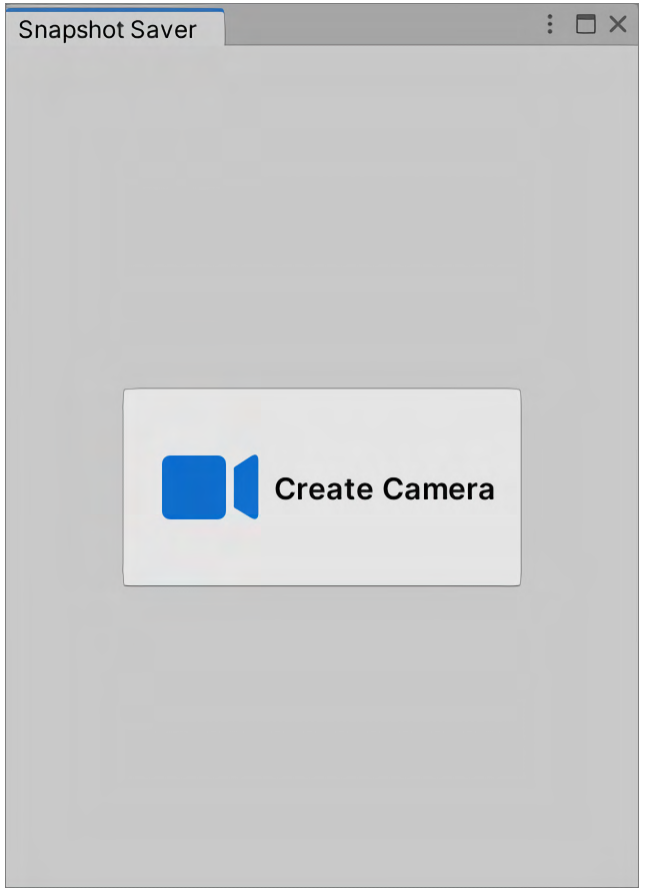
SnapshotWindow.cs - very useful and powerful tool that will help you **save** any composed **GameObject** or **prefab** to a single png **sprite**. This can significantly **improve** the **performance** of your application!

1. Click on the **Tools** > **Fallencake** > **Preferences** > **Snapshot Saver** item on the top Unity Menu to open Snapshot Saver window



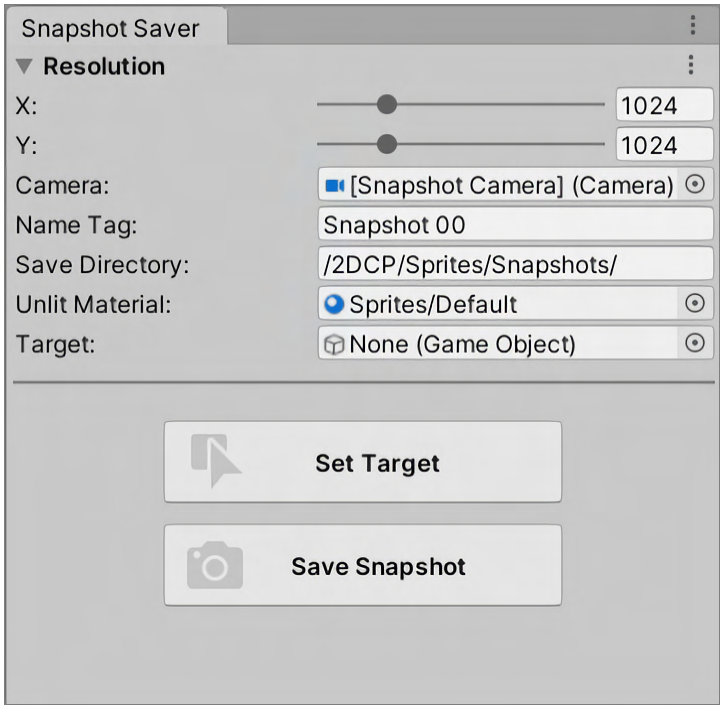
2. Now you need to click on the **Create Camera** button. The program will create a new [Snapshot Camera] gameobject at the bottom of the hierarchy, add all the required components and set all the necessary settings.





Snapshot Saver

Snapshot Saver window contains the following **components**:



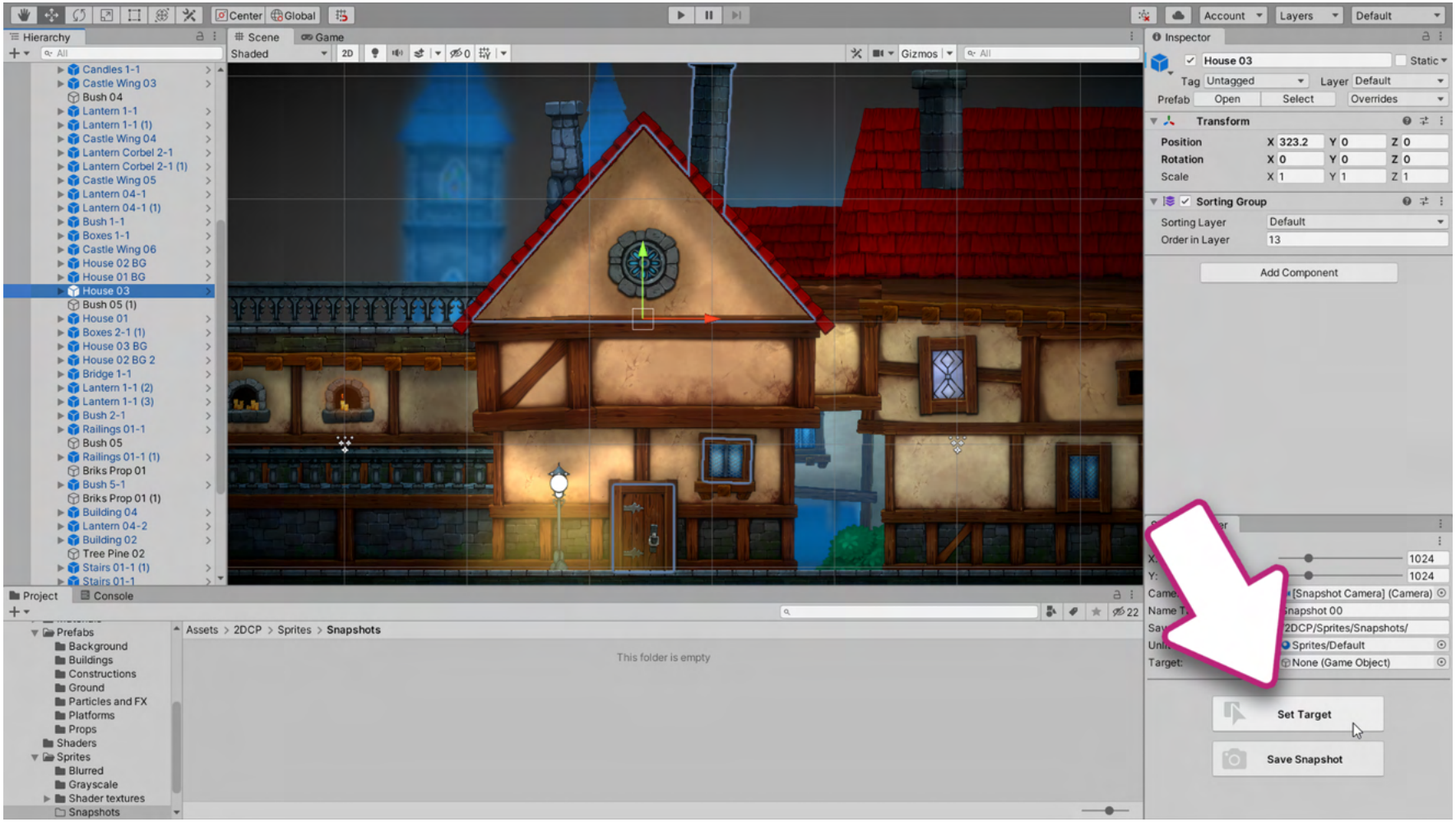
- **Resolution** Sliders - allow you to change the size of the image (X slider - width, Y slider - height in pixels). Also these sliders control the viewport size of the **Camera**
- **Camera** - the camera that will be used for the image rendering (you can recreate it by setting the value of this field to None, or delete the current camera and then click on the **Create Camera** button again)
- **Name Tag** - use this string parameter to specify the images names (numeral ending will be added to this Name tag depending on the number of saved images in the folder)
- **Save Directory** - path to the save folder (all snapshots you have saved will be stored in this folder)
- **Unlit Material** - this material will be applied to all sprites of the **Target** gameobject (unlit material is necessary for the correct transparency of the saved image)
- **Target** - gameobject you want to save as a single PNG image. Just select a gameobject you want and click on the **Set Target** button. The program will create a clone of this gameobject, place it in the field of view of the camera and prepare it for taking a snapshot.

Snapshot Saver

Composed GameObjects, made up of many sprites, often **reduce frame rates**, especially on mobile platforms. Below you can see an example of such a complex gameobject.

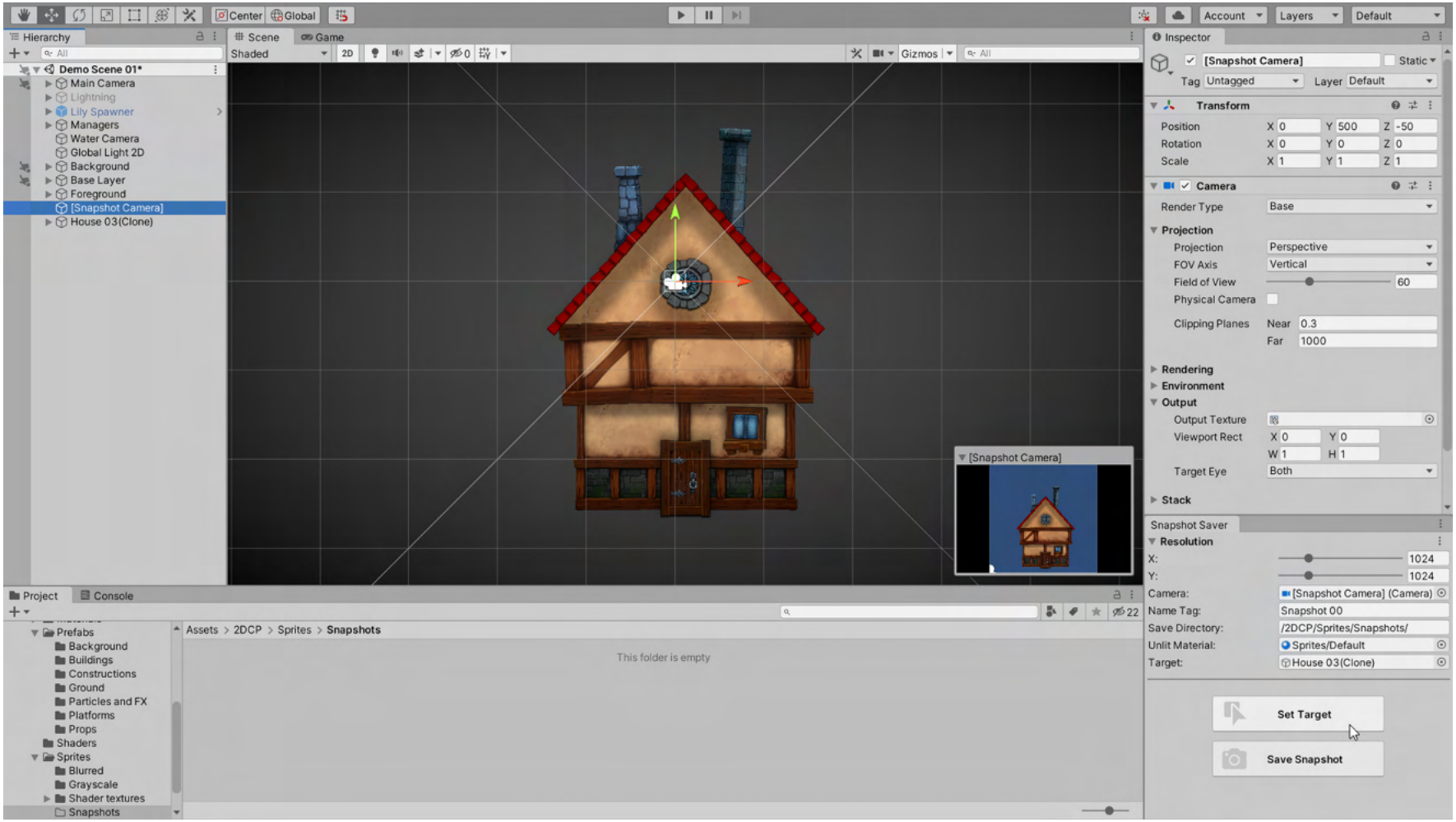
Snapshot Saver helps you optimize your level design and **improve performance** of your application a lot with just a few clicks!

3. **Select** any **gameobject** you need to save as PNG and **click** on the **“Set Target”** button.

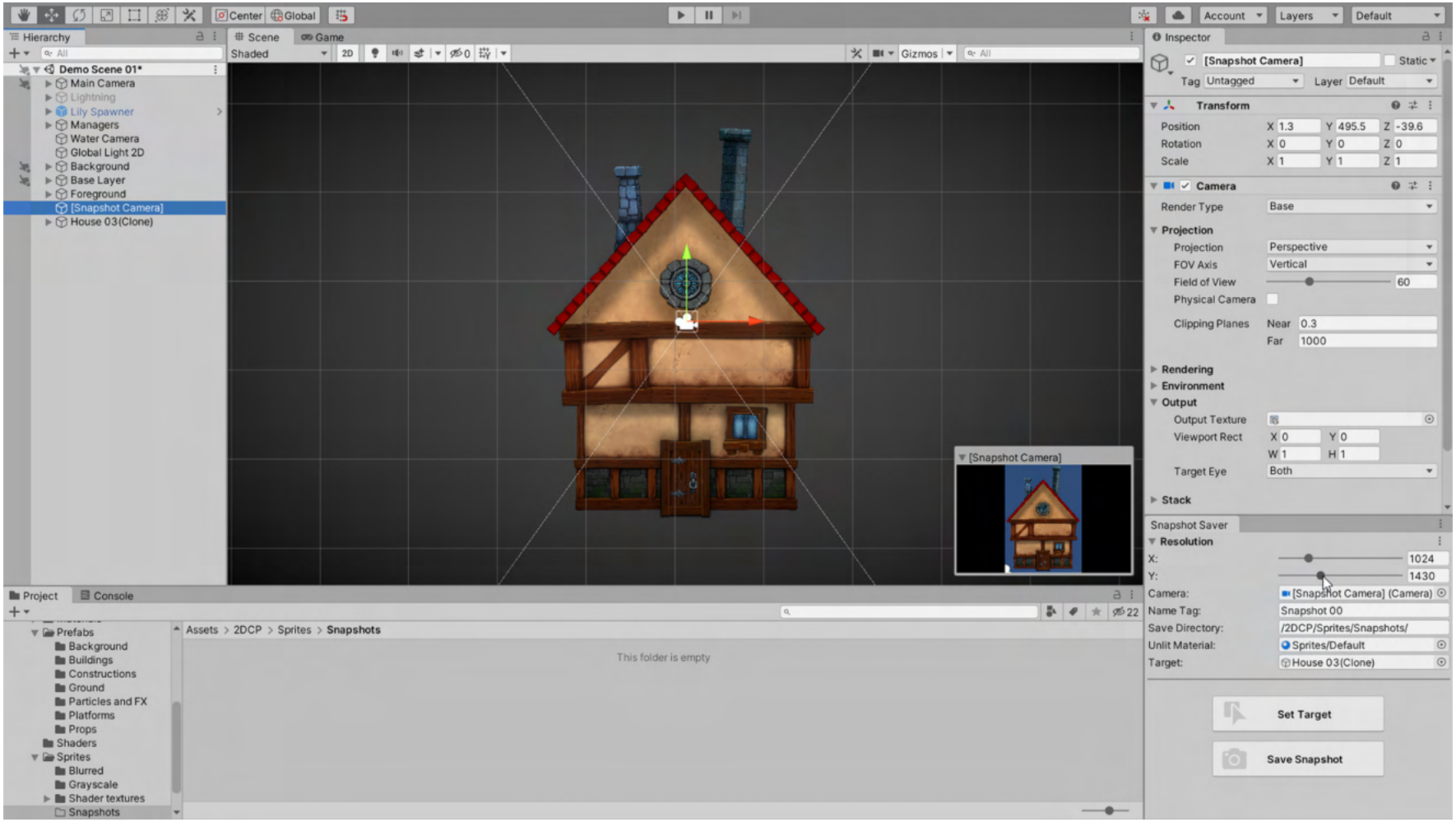


Snapshot Saver

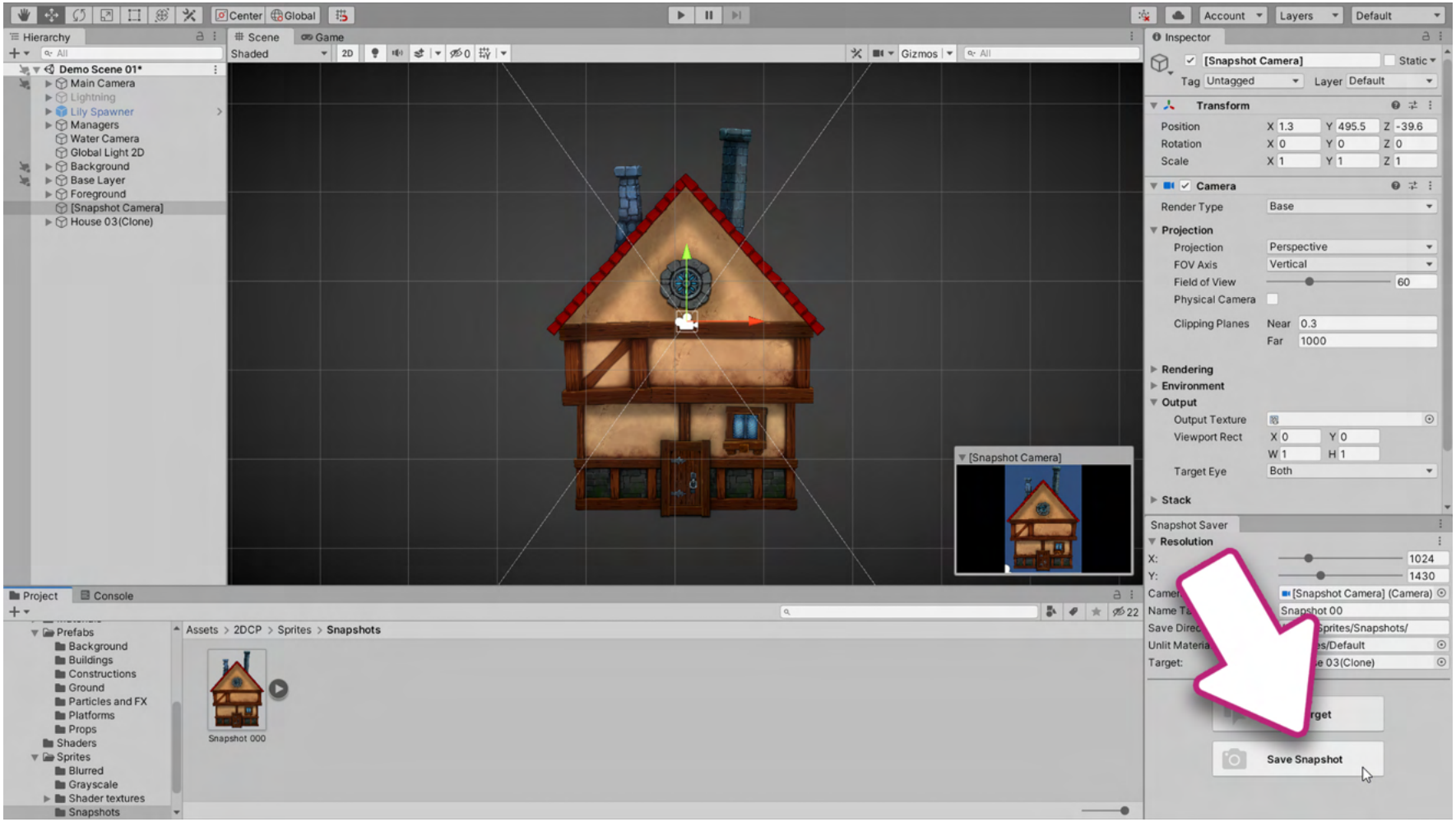
Selected gameobject will be duplicated and the clone will be placed in the [Snapshot Camera] field of view. This clone will be set as a Target in the Snapshot Saver window.



4. **Set position** of the **camera** to find the best view point on the target. Also you can **change the Resolution** value to make the target fit better in the frame.

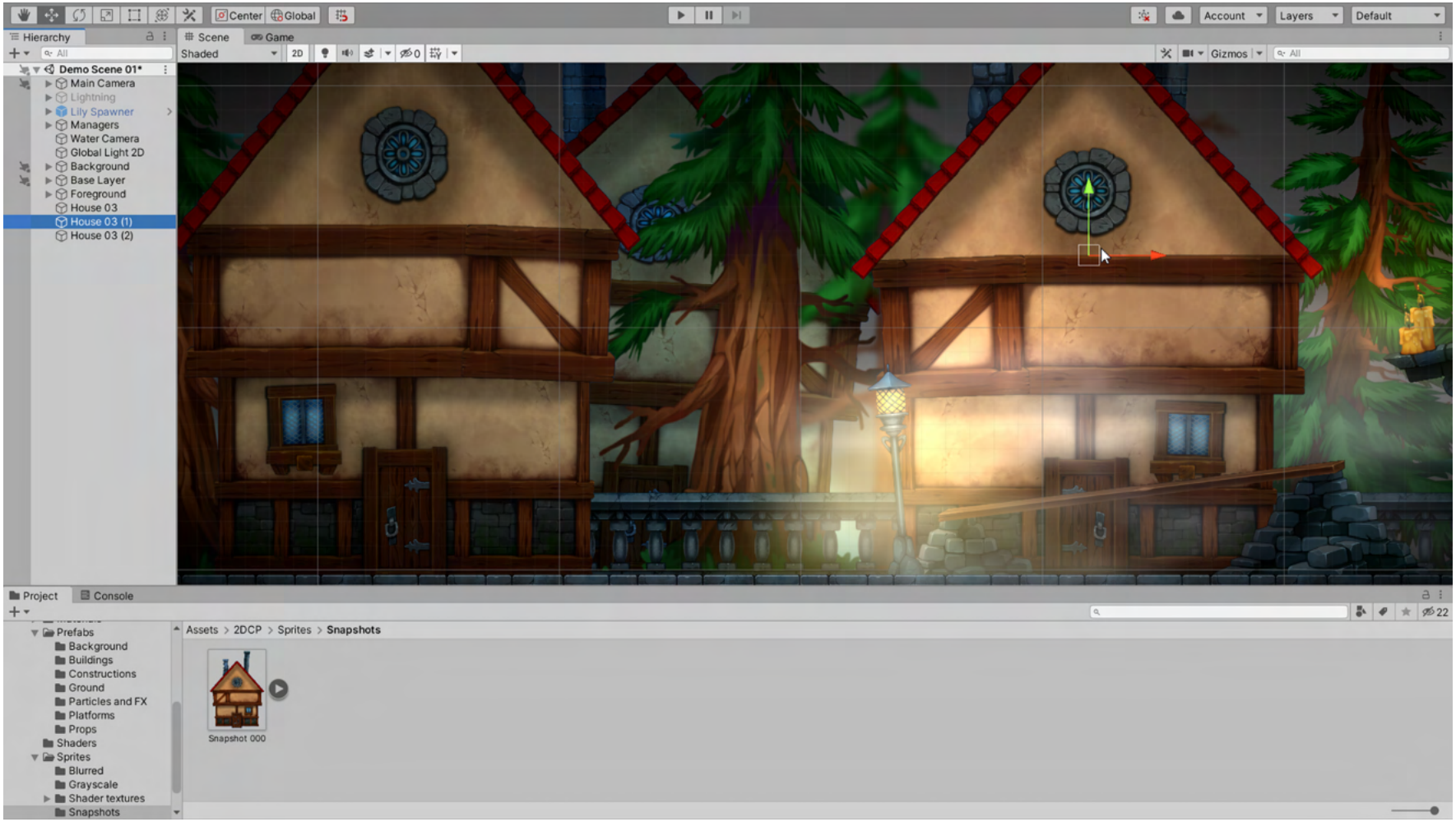


5. And now **click** on the **“Save Snapshot”** button to **save** your PNG **image**.



You will immediately see your new image appear in the saving folder. This image is completely ready to use in level design!

Using such images is much more efficient and better for performance than using complex game objects.



You can also **combine** the **snapshots** you created into a single **sprite sheet** to improve the performance of your app even more!

Demo

The **2DCP** > **Demo** folder contains Animations, Materials, Scripts, Universal Render Pipeline assets that are required for the package to work correctly.

The **Demo Scene 01** in the folder **2DCP** > **Demo** > **Scenes** showcases the usage of the sprites, shaders and scripts to help you to **create the gorgeous worlds!**