Documentation

Summary

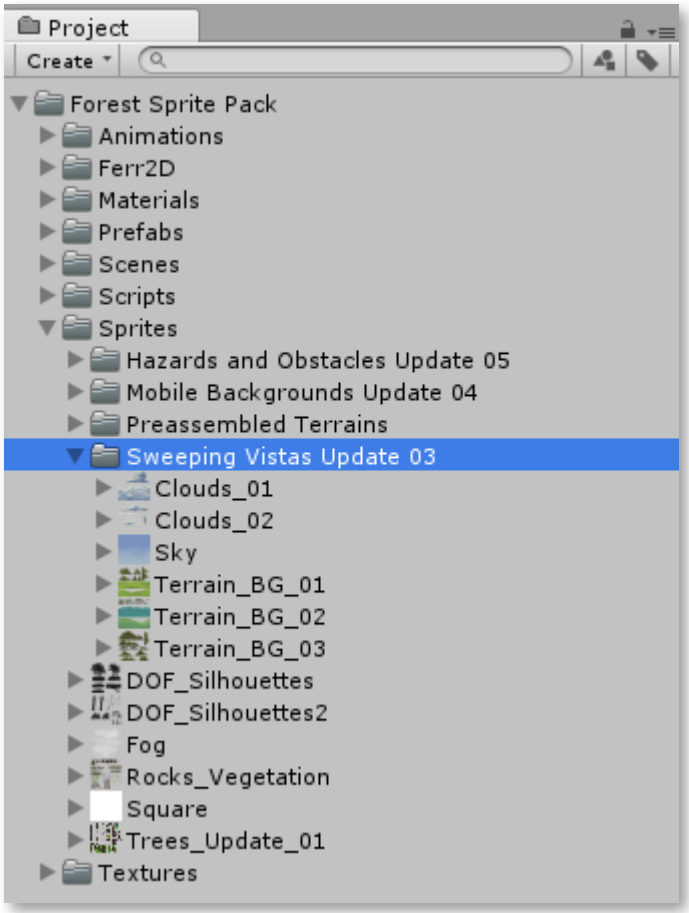
Folder Structure

The folder structure of the package should be self explanatory, each folder holds the kind of file it's name represents. On the Animation folder there are the animations for the panning cameras and for the animated objects in the package, the Scenes folder holds all the demo scenes, the Sprites folder all the sprites, and so forth.

Probably the most important folder for the user of the package is the Prefabs folder, where all the sprites and objects are set up for usage on your levels. More on the structure of this particular folder on the Prefabs section.

Since the Forest Sprite Pack gets updated overtime, the contents of new updates are stored into child folders inside the appropriate parents folders. That way you can easily find and import into your project just the necessary folders containing the new updates, preventing the importer from overwriting possible changes you might have made to the original files.

For example, the sprites that composes the Sweeping Vista Update, the third update of the package, are inside the folder Sprites / Sweeping Vistas Update 03.



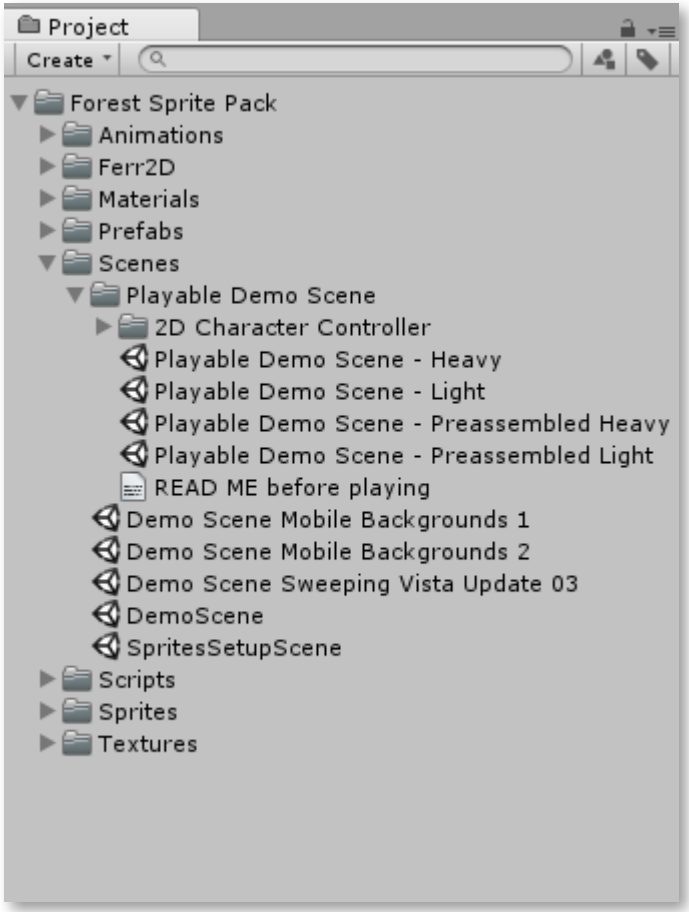
Sprites folder

Package Structure

Scenes

The scenes in the package showcases the usage of the sprites to create levels and backgrounds for your games.
With each new update a new demonstration scene has been added to the package, as a showcase of the new sprites in the update.

Inside the Scenes folder there's also the Playable Demo Scene folder, which contains a few playable demos, as well as the assets for the playable character in them.



Scenes folder

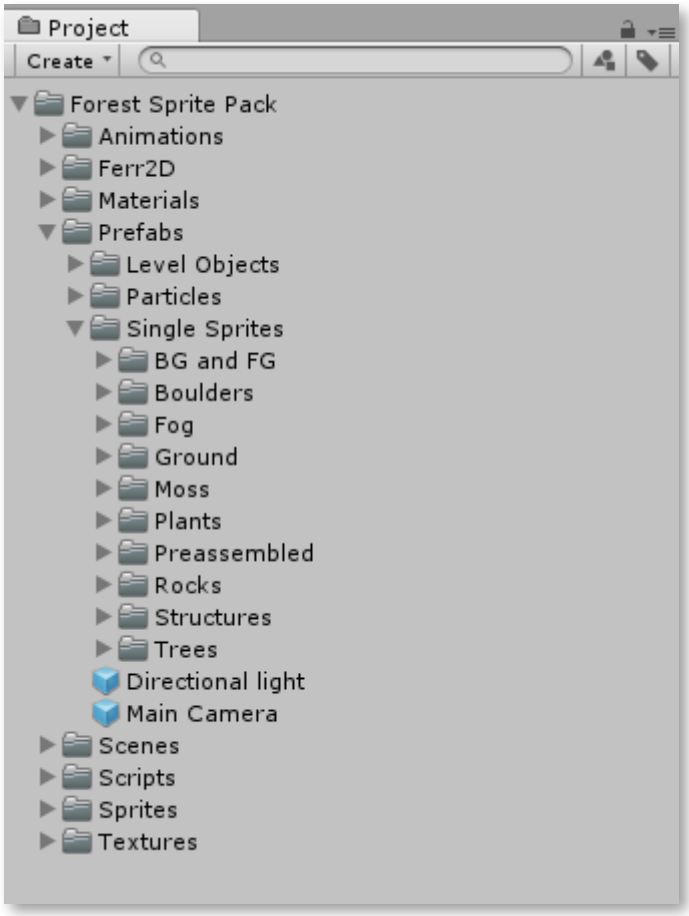
Package Structure

Prefabs

The Prefabs folder contains all the objects in the package, the prefab version of each sprite, as well as all the objects composed of multiple sprites and / or scripts, like grounds, platforms, trees, spikes, and many more.

This folder is divided into three main folders, the Single Sprites folder, which contains every sprite in the package, the Level Objects, which contain the assembled objects, and the Particles folder.

The Single Sprites folder is subdivided into categories, for ease of use. Inside the Ground folder there will be the ground related sprites, inside the Plants folder all the plants sprites, and so on. Both the Single Sprites and Level Objects folders are also subdivided by the updates, so if you're using an older version of the package and wishes to update it, should be easy to import just the new sprites without overriding the previous ones.



Prefabs folder

Using the Package

The Sprites Setup Scene

Inside the Scenes folder is set up a handy scene called SpritesSetupScene.
In this scene all the prefabs of the Prefabs folder are arranged into groups of their respective updates. You can use this quick access scene to visualize and drag and drop prefabs into your own scenes by having both open at the same time.

Organization

Plan ahead how you want to build your levels, for instance if you're building it with an orthographic camera moving parallax layers by script, or with a perspective camera, so parallax happens naturally with the distance from the camera, each different approach will display the sprites differently and will impact how you should handle their scales and positions.

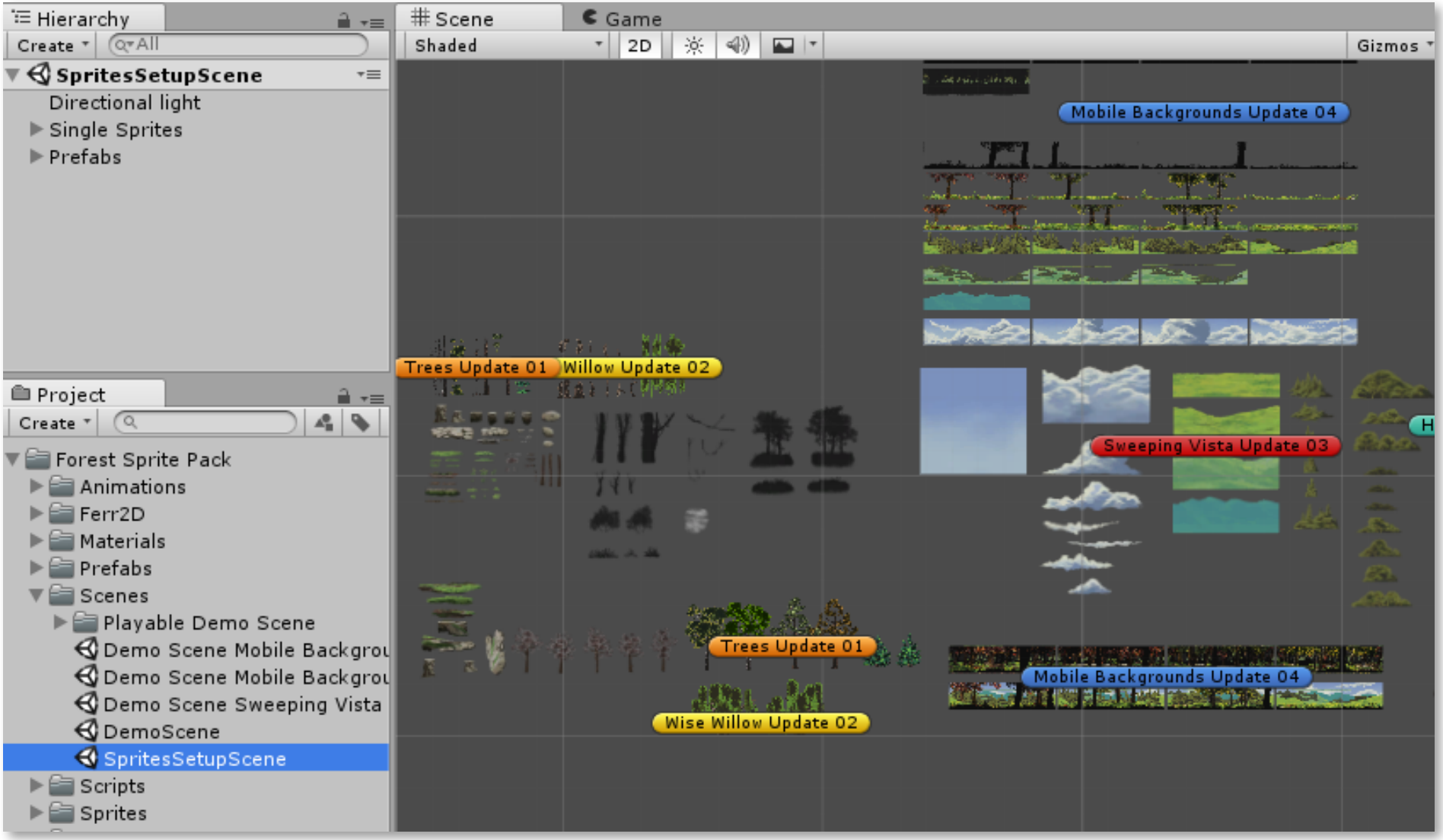
Setting up Sorting Layers will help tremendously to keep scene organization as tidy as possible. You can use a different sorting Layer for each layer of parallax, for example, making it easy to edit and maintain your levels.

Since the Forest Sprite Pack is a package of files that inhabit the Assets folder, the package itself won't contain any layers or sorting layers, which are setup at the project level. That means i wouldn't be able to provide a demo of how to organize sorting layers for building your scenes.

You'll notice all demo scenes sprites are setup inside the same Default sorting layer, which i don't recommend. It makes the maintenance of the level much more difficult and time consuming.

Setting up some sorting layers for better organization is a very simple process and will go a long way for productivity.

Using the Package



The SpritesSetupScene

Performance

Batching

One of the benefits of having sprites packed inside the same spritesheet (or texture atlas), is the possibility of having all these sprites batched together, reducing the overhead required for drawing each one on the screen. Unity's documentation details all the necessary requisites for batching objects together, but in essence there's a few things to mind about when building your levels, as follows:

Static and Dynamic batching.

In Unity it's possible to set gameobjects as Static or not. If a gameobject stays static and won't move, checking the Static box will change their batching from dynamic to static.
The difference between these two types of batching is that the static one groups meshes together, relieving processing from the cpu, but taking more memory space, while dynamic batching is done on the fly by the cpu, which is less performant.

Packing sprites inside the same spritesheet

Although the sprites provided in the package are already packed inside several spritesheets, they're manually placed with legibility in mind. That means there's room for improvement with a tighter packing. Depending on the version of Unity this can be set up differently. On older versions that's available through the sprite packer, and in newer version through the sprite atlas.
The returns of using the tighter, machine made, packing will be small, since the sprites are already packed. But if you want to squeeze that little bit more of performance, it won't hurt to setup these tools.

Using the same materials

Besides having packed sprites in the same texture, it's important that all the objects share the same material for batching to happen.
The demo scenes only uses two different materials, the Sprites-Default and Sprites-Diffuse, the former for unlit sprites and the later for using dynamic lights. These should be enough to cover most uses for this package, but, if for some reason you're using different materials, keep in mind that each material is basically a new batch.

Using the Package

Dynamic Lighting

Dynamic lighting is one of the most resource intensive things you can have on your scenes, but it's also one of the most visually impacting.

The same scene can have multiple moods depending on how it is lit. The thing to keep in mind is that each pixel light will add a new draw call for each sprite it lights, which can go out of proportions really easily.

Managing how many lights are in the scene, how much are pixel lights and / or vertex lights and where to put them is an strategic balance that entirely depends on the needs of the project and the platform it is being published on.

Inside the Scenes / Playable Demo Scene there are examples of scenes built with dynamic lights and without lighting at all, so you can assess an example of how these different approaches impact the performance of your game.

Dynamic Light

Using the Package

Single Sprites and Preassembled Sprites

The number of sprites being rendered every frame can have different impacts on the performance of the game. Every sprite will have some measure of Alpha Overdraw, since the nature of how the sprite's shaders render them on screen, and a certain amount of draw calls depending on how many dynamic lights are shining on them.

One way to mitigate the performance issues caused by the amount of sprites on screen, is to assemble the sprites provided by the package into larger terrains outside of Unity, and then reimport then back as a single sprite.
By doing this an object that would be composed of many sprites inside the scene will be composed of only one sprite, diminishing the performance cost of having lights shining on it. The drawback to this technique is the increased amount of video memory required to store the additional images that represent the preassembled terrains.
So, just like the other aspects of performance, what will define how to handle the number of sprites on screen is entirely dependent on the measurements of performance on your project and the limitations of the targeted platform.

Inside the Sprites / Preassembled Terrains folder there are a few spritesheets with an assortment of these preassembled objects. Their prefabs are store at the Prefabs / Single Sprites / Preassembled folder.
They should provide an example of how you can preassemble more terrains of your own with the sprites of the package, in case you wish to diversify even more the possibilities of the Forest Sprite Pack.

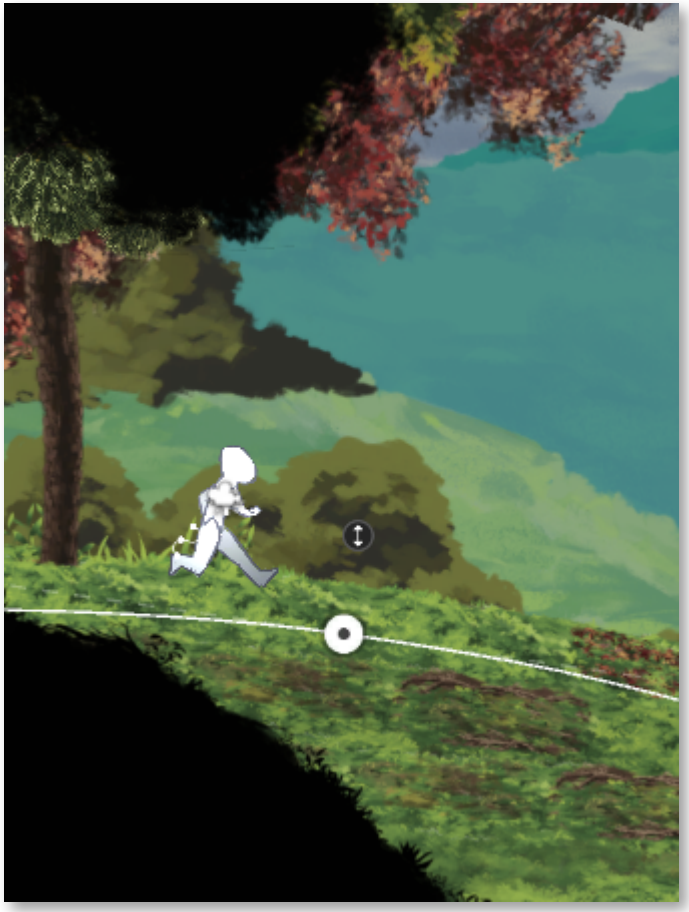
Inside the Scenes / Playable Demo Scene folder there are examples of scenes built entirely with the small modular sprites of the package and also with preassembled terrains, to provide an example of how the different approaches impact on the performance of the game.

Ferr2D Terrain Tool

The Ferr2D Terrain is a third party asset that lets you create 2D and 2.5D landscapes and levels with a path based tool.

Forest Sprite Pack has set up several Ferr2D Terrains inside the Ferr2D / Terrain Materials folder. So you can leverage the visual style of this package with this design tool that makes creation of levels much faster and easier.
Every Terrain Material is provided both with a Lit and an Unlit version, which are showcases inside the playable demo scene in the Ferr2D / Ferr2D Playable Demo folder.

Once both the Ferr2D Terrain Tool and the Forest Sprite Pack are imported into the project everything should work fine. If for some reason the Terrain Materials doesn't seem to be working properly, make sure to import the Ferr2D package before importing the Forest Sprite Pack / Ferr2D folder, as this seems to solve any issues.



Ferr2D Terrain Tool

Thanks for acquiring the Forest Sprite Pack!

Just wanted to ask you to spare a minute, whenever you can, to rate your experience with the package at the store. It helps the further development of this package and the creation of new ones.

You support is deeply appreciated!

Rate It!