General information

SpriteSharp is an efficient solution for generating better meshes for sprites, helping reduce your 2D scene total polygon count, draw call amount, overdraw and CPU load.

While Unity itself is capable of doing this, the possibilities are limited. There are no options to tweak to get the best possible performance, and sometimes meshes generated by Unity are so suboptimal you have to revert back to using Full Rect quads, losing the overdraw improvement.

This is why *SpriteSharp* was invented. It extends built-in Unity sprite mesh generation with various tweakable options, allowing you to get the best performance out of your 2D game. It is especially useful when working on complex scenes and developing for mobile platforms, where overdraw is often a problem.

Just a few more points:

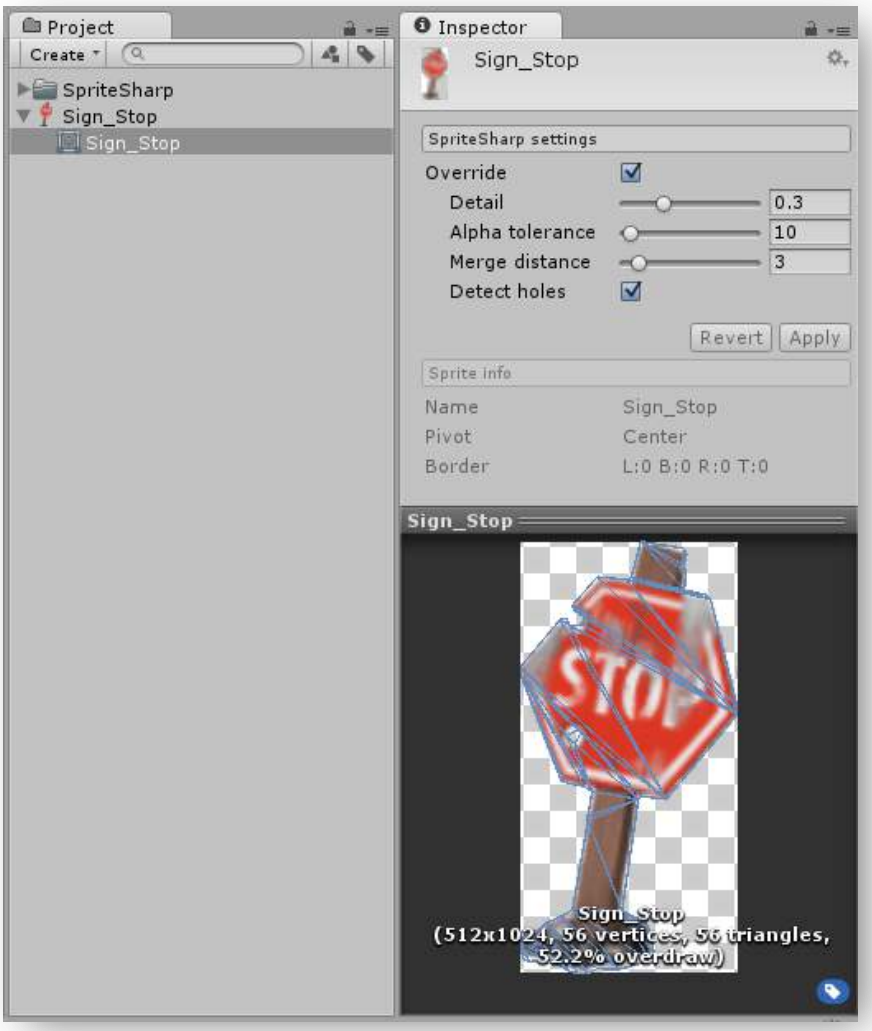
- If you are drawing a lot of sprites, the difference between 1000 and 100 triangles per sprite is going to end up a big deal.
- Objects with `SpriteRenderer` are only batched up to a certain amount of triangles, and the limit of triangles per batch is around 700. This is because sprites are batched on the CPU, and doing that has a slight overhead. Therefore, decreasing the polygon count also decreases the amount of draw calls and offloads the CPU for more interesting work, like running your game scripts.
- Unity has no prior knowledge about how sprites will be scaled in the scene. There is no point of making highly-tesselated mesh for an object that takes a small area on the screen. That's why manual control is needed.

Unity 5.0 and newer is supported, both Personal and Pro.

Usage

SpriteSharp integrates seamlessly into Unity and is very easy to use. Just make sure you have `Mesh Type` set to "`Tight`" in the texture settings, and select the sprite in your `Project` view. The *SpriteSharp* interface will show up in the `Inspector`.

Let's take a closer look on available options.



OVERRIDE

Enabled and disables *SpriteSharp* mesh generation. Default Unity algorithm will be used when this is turned off.

DETAIL

This value controls how detailed the sprite mesh must be. Lower values results in less triangles, but more overdraw, high values result in low overdraw, but at the cost of more triangles.

This is the value you'd want to tweak first to get a good balance of triangle count and overdraw. A value of 0.3 is a good starting point.

ALPHA TOLERANCE

Pixels with alpha less than this value will be ignored and not included into the sprite mesh. Value of 0 means all non-transparent pixels are included, and value of 254 means only fully opaque pixels will end up in the sprite mesh.

Value of 10-20 generally works well for sprites that are mostly opaque (characters, walls, etc.) without producing any artifacts. It is recommended to keep this value low for sprites representing special effects (smoke, explosions, etc.).

MERGE DISTANCE

Merging vertices that are very close to each other is efficient for decreasing the mesh complexity without sacrificing detail. This value controls the maximum distance at which vertices are merged.

Value of 3 works well for most sprites without producing any artifacts. Raising this value too high may result in parts of sprites being slightly cut off.

DETECT HOLES

This value controls whether inner holes in the texture will be detected and excluded from the sprite mesh, reducing the amount of overdraw.

It is recommended to keep this option enabled, since it pretty much has no drawbacks. However, you may want to disable it in some special cases.

Version control

All sprite mesh settings are stored in the file

```
Assets/SpriteSharp/Editor/Resources/SpriteSharpDatabase
```

This file has to be under control of your version control system in order to share sprite mesh settings across your team members.

Source code

SpriteSharp comes in both compiled DLL and source code forms, but after importing the package you'll have the DLL version. This is done to keep your project less cluttered and faster to compile. To get the source code, import the package

```
Assets/SpriteSharp/SpriteSharpSource
```

After importing the package, delete these files:

```
Assets/SpriteSharp/Editor/LostPolygon.SpriteSharp.dll
Assets/SpriteSharp/Editor/LostPolygon.SpriteSharp.dll.mdb
```

Note: switching to source code version or back to compiled DLL will reset the *SpriteSharp* database, so it's better to be done before adjusting sprite mesh settings.

Known issues

- Sprite mesh settings are not duplicated automatically when the texture is duplicated.
- Sprite mesh is not regenerated automatically when the texture is renamed. You have to use "Reimport" from the texture context menu after renaming the texture.

Contact

For any questions about this plugin, feel free to contact me at:

Unity forums thread: http://forum.unity3d.com/threads/spritesharp-sprite-mesh-optimizer.327997
e-mail: contact@lostpolygon.com
Skype: serhij.yolkin

Version history

1.0.0:

- Initial release.