glue Documentation

Release 0.11.1

Jorge Bastida

July 12, 2016

Contents

Glue is a simple command line tool to generate sprites:

```
$ glue source output
```

- Automatic Sprite (Image + Metadata) creation including:
  - css (less, scss)
  - cocos2d
  - json (array, hash)
  - CAAT
- Automatic multi-dpi retina sprite creation.
- Support for multi-sprite projects.
- Create sprites from multiple folders (recursively).
- Multiple algorithms available.
- Automatic crop of unnecessary transparent borders around source images.
- Configurable paddings and margin per image, sprite or project.
- Watch option to keep glue running watching for file changes.
- Project-, Sprite- and Image-level configuration via static config files.
- Customizable output using jinja templates.
- CSS: Optional .less/.scss output format.
- CSS: Configurable cache busting for sprite images.
- CSS: Customizable class names.

CHAPTER 1

Documentation

1.1 Installing Glue

Glue only depends on one external library, Pillow. a friendly fork of PIL.

This libraries require some external codes in order to manipulate `jpeg` images. These codecs aren't available by default in some Linux distributions neither OSX, so it's necessary to install them manually.

1.1.1 OSX

If you are using OSX, the easiest way to install the jpeg decoder is using Homebrew. Before installing `Homebrew` you'll need to install Xcode, then you can follow these steps:

```
$ brew install jpeg
$ sudo pip install glue
```

Note: If you are using Snow Leopard (10.6) you should install glue using `sudo env ARCHFLAGS='-arch i386 -arch x86_64' pip install glue`

1.1.2 Debian/Ubuntu

If you are using Debian/Ubuntu installing `glue` is really easy:

```
$ apt-get install libjpeg62 libjpeg62-dev zlib1g-dev python-dev
$ sudo pip install glue
```

You can also install it using the native package:

```
$ apt-get install glue-sprite
```

1.1.3 Windows

1. Install Python, if not yet available. Python 2.7.2 Windows installer.
2. Install PIL, check this website for a matching version (PIL-1.1.7 for Python 2.7)
3. Install Python's `easy_install` easy_install installer for Python 2.7.

4. Add Python's Scripts dir to your Path. Add `;C:\Python27\Scripts` to the end of the line.
5. Start the cmd and type

```
$ easy_install glue
```

6. Easy isn't?

1.1.4 Development version

The source code of Glue is available on Github https://github.com/jorgebastida/glue/.

1.2 Quickstart

After installing Glue you will have a new command named `glue`. You can check if it was correctly installed calling `glue` with the `--help` option to get the list of all the available command line arguments:

```
$ glue --help
```

If `glue` was correctly installed... Let's create your first sprite!

1.2.1 Your first sprite

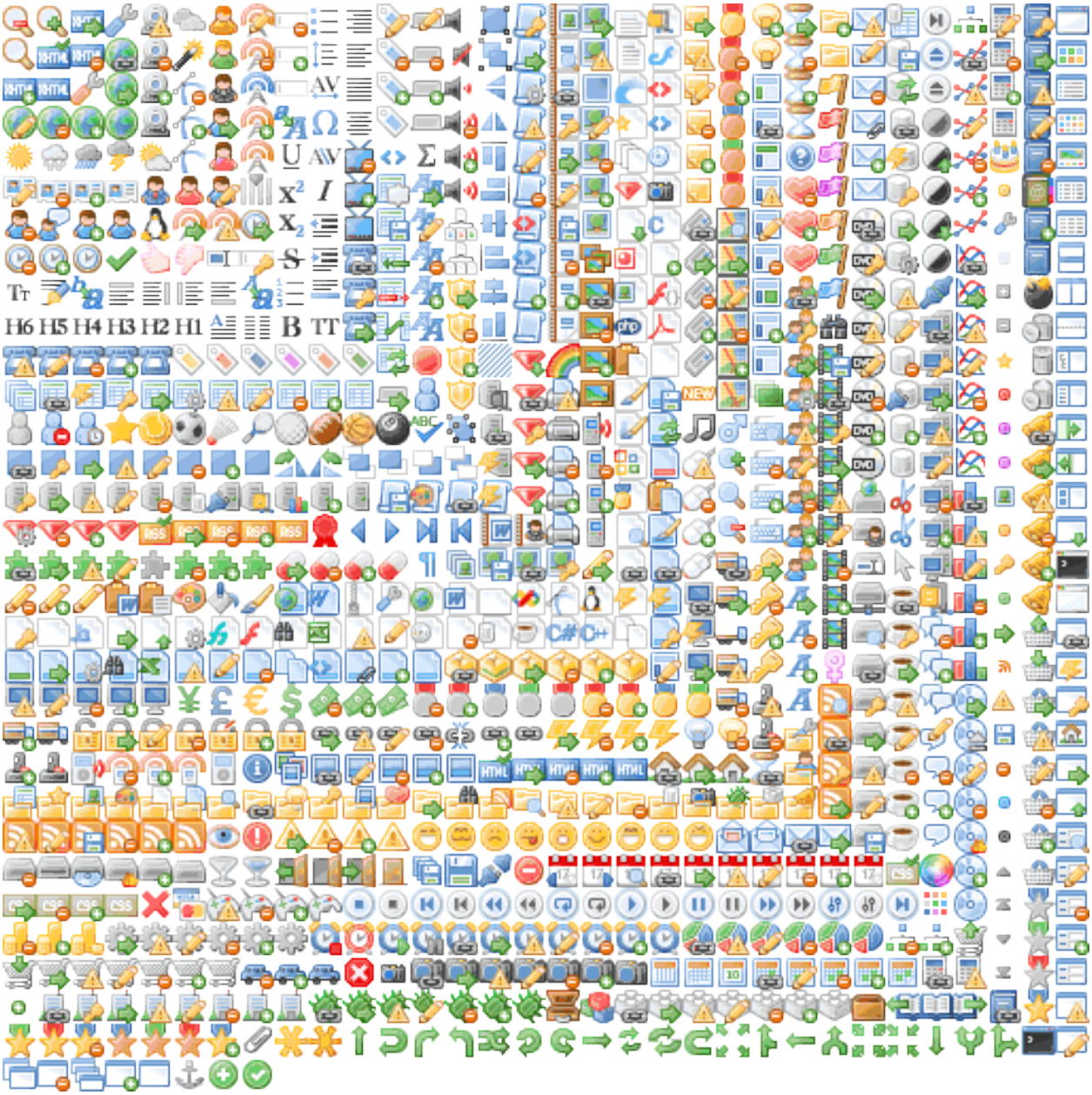
Create a new folder (`icons` in this example), and add as many images as you want. Then you can simply run the following command:

```
$ glue icons sprites
```

Glue will create a new folder named `sprites` with the following structure:

```
sprites
    -- icons.css
    -- icons.png
```

For example using the gorgeous famfamfam icons (4.2Mb) you will get the following `icons.png` (401Kb).



The other file, `icons.css` will have all the necessary css classes for this sprite:

```
.sprite-icons-zoom_out{ background:url('sprites/icons/icons.png'); top:0; left:0; no-repeat;}
.sprite-icons-zoom_in{ background:url('sprites/icons/icons.png'); top:0; left:-16; no-repeat;}
.sprite-icons-zoom{ background:url('sprites/icons/icons.png'); top:-16; left:0; no-repeat;}
.sprite-icons-xhtml_valid{ background:url('sprites/icons/icons.png'); top:-16; left:-16; no-repeat;}
...
```

1.2.2 And why those CSS class names?

As you can see, `glue` will use both the filename and the sprite name as part of the css class name. You can generate the sprite as many times as you want on any computer and the CSS class related to a specific image will always be the same, so you can create these sprites safely as part of your deployment process without being worried about css class name changes/collisions.

Note: `glue` will only get alphanumeric, _ and - characters from the filename to create the CSS class name.

For example, an imaginary `animals` sprite with 5 images will have the following class names.

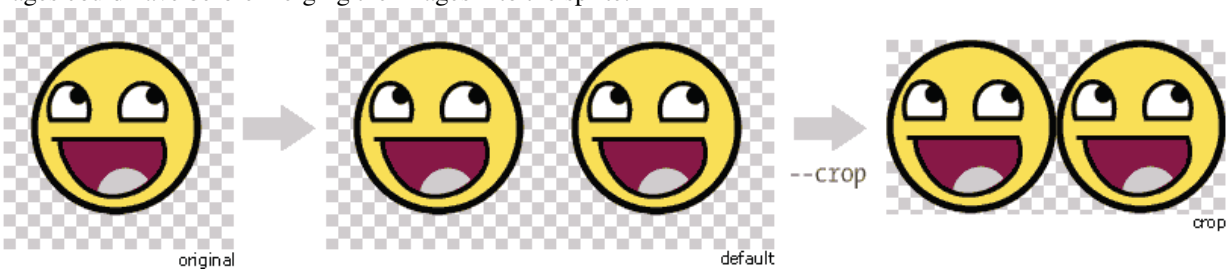
filename	css class name
cat.png	.sprite-animals-cat
dog2.png	.sprite-animals-dog2
cat_blue.png	.sprite-animals-cat_blue
dog-red.png	.sprite-animals-dog-red
dog_(white).png	.sprite-animals-dog_white

If for any reason two images are to generate the same CSS class name, an error will be raised.

Note: All CSS class names generated by `glue` will also have a namespace `sprite-` in the beginning. This namespace could be changed using the `--namespace` option.

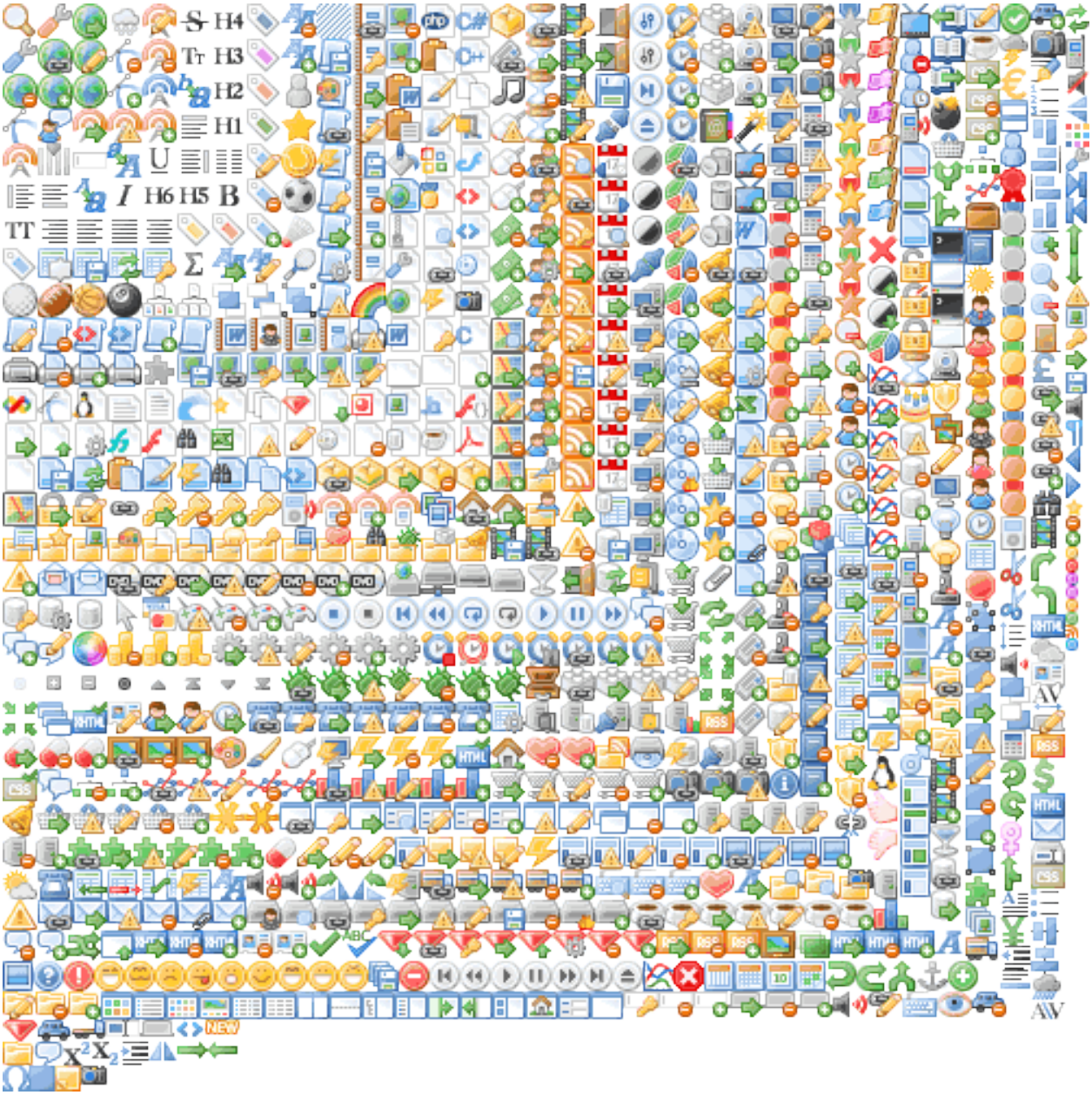
1.2.3 Crop unnecessary transparent spaces

Usually designers add some unnecessary transparent space around the images because it is easier for them to work with a larger canvas. `glue` can optimize our sprite by cropping all the unnecessary transparent spaces that the original images could have before merging the images into the sprite.




```
$ glue icons sprites --crop
```

The new `icons.png` (348Kb) will be 53Kb smaller.



Now, the css file will have the new coordinates but using the same css class names!

```
.sprite-icons-zoom{ background:url('sprites/icons/icons.png'); top:0; left:0; no-repeat;}
.sprite-icons-wrench_orange{ background:url('sprites/icons/icons.png'); top:0; left:-16; no-repeat;}
.sprite-icons-wrench{ background:url('sprites/icons/icons.png'); top:-16; left:0; no-repeat;}
.sprite-icons-world_link{ background:url('sprites/icons/icons.png'); top:-16; left:-16; no-repeat;}
...
```

1.2.4 What about if I need to generate multiple sprites?

Usually an app has more than one sprite and generate all of them one by one could be annoying.

The suggested setup is to create a new folder for every sprite, and add inside all the images you need for each one. `glue` will create a new sprite for every folder if you use the `--project` argument:

```
images
-- actions
|   -- add.png
|   -- remove.png
-- borders
|   -- top_left.png
|   -- top_right.png
-- icons
    -- comment.png
    -- new.png
    -- rss.png
```

So now, running:

```
$ glue images sprites --project
```

Will generate a new `sprites` folder with the images and the css inside:

```
sprites
    -- actions.png
    -- actions.css
    -- borders.png
    -- borders.css
    -- icons.png
    -- icons.css
```

1.2.5 And now?

`glue` have some more magical powers inside!

- Retina sprites: Do you want to make your sprites look good on any device? Read the ratios documentation.
- Glue can also read the configuration from static config files.
- We support less! It's easy, add `--less` and `glue` will generate the CSS file with the `.less` extension.
- Cache Busting? Yes! Add `--cachebuster` and `glue` will add the `SHA1` of the PNG sprite as a queryarg on the CSS files. Read the Command line arguments page.
- Still hungry? Read the Command line arguments page to discover all the available settings.

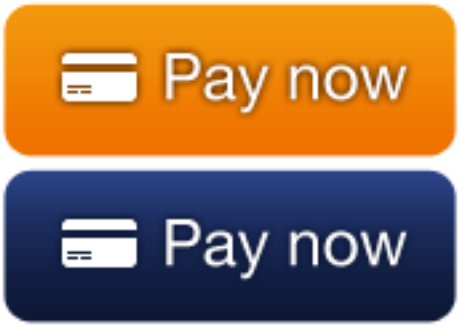
1.3 Pseudo Classes

1.3.1 Using the filename

Using the filename of the source images you can customize the pseudo class related to the images, so if you simply append `__hover` to the filename `glue` will add `:hover` to the CSS class name:

```
buttons
-- pay.png
-- pay__hover.png
```

Using this simple convention you can create for example create button sprites like:



And generate automatically the following css:

```
.sprite-buttons-pay{background-image:url(buttons.png);background-repeat:no-repeat}
.sprite-buttons-pay:hover{background-position:0px 0px;width:174px;height:62px;}
.sprite-buttons-pay{background-position:0px -62px;width:174px;height:62px;}
```

Note: You can use multiple pseudo-classes at the same time `__hover__before.png`

Note: pseudo-class separator use to be _. Since `glue 0.9` it is __. If you don't want / you can'e rename all your files, you can use `--pseudo-class-separator=_` in order to make `glue` work in legacy mode.

1.3.2 Available pseudo classes

Glue will only detect the following pseudo-classes: `link`, `visited`, `active`, `hover`, `focus`, `first-letter`, `first-line`, `first-child`, `before` and `after`.

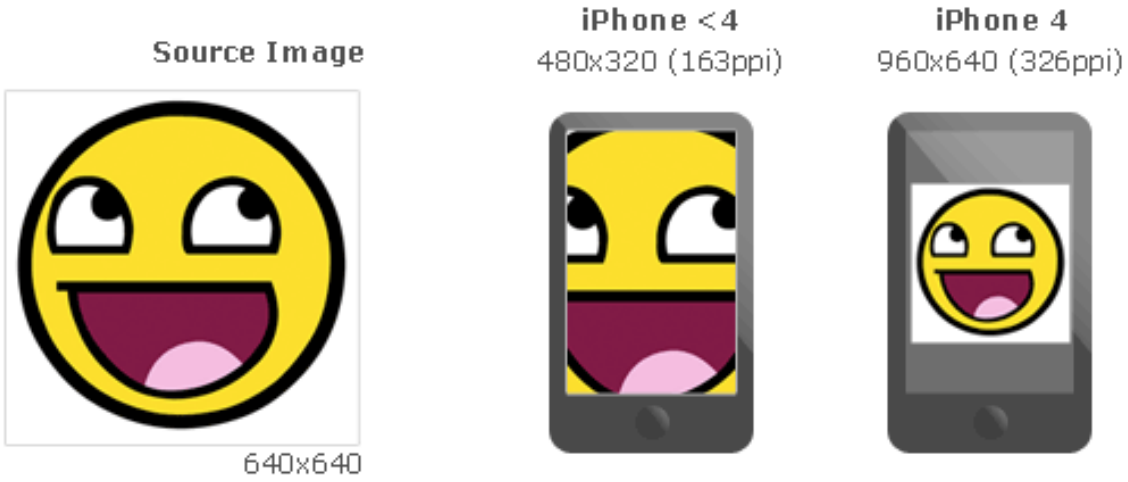
1.4 Retina Sprites: Ratios

1.4.1 What is this for?

These days, some devices like the iPhone 4, or the new Macbook Pro have screens with higher pixel density than usual. For example iPhone4's Retina Display doubles the pixel density we use to see on handheld devices.

This change improves the sharpness of vector graphics, but not images... Why? These devices scale vector graphics like text without losing quality, but in order to make images bigger, images are automatically pixel-doubled like in the following example.

How can we solve this problem with high DPI devices? Basically, provide two different version of each image.



And... how can we detect wich image should we use? CSS Media Queries. Modern browsers (anything after IE 8.0) supports them, and they allow us to specify different styles based on the `device-pixel-ratio` of the browser.

Can glue help? Yes, using `--ratios` you can choose different ratios you want to build of each sprite. Glue will create one sprite for each ratio and will add all the neccesary CSS magic to make the browser use the high DPI image if the browser needs it. You can also use `--retina`, it's a shortcut for `--ratios=2,1`.

1.4.2 How –retina and –ratios work?

As `glue` cannot do magic scaling up the source images, **it assumes that these images are the biggests you want to serve**. *(i.e. For iPhone 4 Retina these images should be 2x the final size you want)*, then glue will create one sprite for each ratio you set in the command line or only `2x` if you use `--retina`:

```
$ glue icons sprites --retina
```

This command will generate the following files:

```
sprites
-- icons.css
```

```
-- icons.png
-- icons@2x.png
```

Fig. 1.1: icons.png

Fig. 1.2: icons@2x.png

And this will be the content of `icons.css`:

```
.sprite-sprites-loopback,
.sprite-sprites-weather,
.sprite-sprites-magnify,
.sprite-sprites-chat{
    background-image:url(sprites.png);
    background-repeat:no-repeat
}

.sprite-sprites-loopback{ background-position:-1px -1px;width:32px;height:21px; }
.sprite-sprites-weather{ background-position:-1px -24px;width:24px;height:26px; }
.sprite-sprites-magnify{ background-position:-35px -1px;width:24px;height:24px; }
.sprite-sprites-chat{ background-position:-35px -27px;width:24px;height:22px; }

@media only screen and (-webkit-min-device-pixel-ratio: 2.0),
       only screen and (min--moz-device-pixel-ratio: 2.0),
       only screen and (-o-min-device-pixel-ratio: 200/100),
       only screen and (min-device-pixel-ratio: 2.0) {
        .sprite-sprites-loopback,
        .sprite-sprites-weather,
        .sprite-sprites-magnify,
        .sprite-sprites-chat{
            background-image:url(sprites@2x.png);
            -webkit-background-size: 60px 51px;
            -moz-background-size: 60px 51px;
            background-size: 60px 51px;
        }
}
```

1.4.3 What about if I need some other ratios?

The option `--retina` is only a shortcut for `--ratios=2,1`. You can manually use `--ratios=A,B,C...` to create different ones. For example you can use `--ratios=2,1.5,1` to make glue build three diferent sprites:

```
sprites
-- icons.css
-- icons.png
-- icons@1.5.png
-- icons@2x.png
```

1.4.4 Wich ratios should I target?

Is up to you, but using `2` and `1.5` should be enough for most of the devices.

Here you have a list of suggested ratios for some famous devices, (full list):

Device	Screen size	dpi	Suggested ratio
iPad	**2048 × 1536**	**264ppi**	**2**
iPhone 5/5S/5C	**1136 × 640**	**326ppi**	**2**
iPhone 4	**960 × 640**	**326ppi**	**2**
iPhone 4S	**960 × 640**	**326ppi**	**2**
iPad (3rd gen)	**2048 × 1536**	**264ppi**	**2**
MacBook Retina	**2880 x 1800**	**220ppi**	**2**
Xperia S	**720 × 1280**	**342ppi**	**2**
One X	**720 × 1280**	**312ppi**	**2**
EVO LTE	**720 × 1280**	**312ppi**	**2**
Galaxy Note	**800 × 1280**	**285ppi**	**2**
Galaxy SIII	**720 × 1280**	**306ppi**	**2**
Galaxy S4	**1080 × 1920**	**441ppi**	**3**
Galaxy Nexus	**720 × 1280**	**316ppi**	**2**
Nexus 4	**768 × 1280**	**320ppi**	**2**
Nexus 5	**1920 x 1080**	**445ppi**	**3**
Kindle Fire HDX 8.9	**2560 x 1600**	**339ppi**	**1.5**
Kindle Fire HD 8.9	**1920 x 1200**	**254ppi**	**1.5**
HTC Desire	480 × 800	252ppi	1.5
Nexus One	480 × 800	252ppi	1.5
Sensation	960 × 540	256ppi	1.5
Evo 3D	960 × 540	256ppi	1.5
Sensation XE	960 × 540	256ppi	1.5
LG Optimus 2X	480 × 800	233ppi	1.5
Defy+	854 × 480	265ppi	1.5
Milestone	480 × 854	265ppi	1.5
Nexus S SAMOLED	480 × 800	235ppi	1.5
Nexus S LCD	480 × 800	235ppi	1.5
Galaxy S Plus	480 x 800	233ppi	1.5
Galaxy SII	480 × 800	219ppi	1.5
Galaxy Tab	600 × 1024	171ppi	1.5
iPad mini	1024 × 768	163ppi	1
iPhone	480 × 320	163ppi	1
iPhone 3G	480 × 320	163ppi	1
iPhone 3GS	480 × 320	163ppi	1

Continued on next page

Table 1.1 – continued from previous page

Device	Screen size	dpi	Suggested ratio
iPad (1st gen)	1024 × 768	132ppi	1
iPad 2	1024 × 768	132ppi	1
Kidle Fire	1024 × 600	169ppi	1
Galaxy Y (S5360)	240 × 320	133ppi	0.75

1.5 Configuration files

1.5.1 Introduction

`glue` has around 30 command line options. Remember all of them every time you need to rebuild your sprites could be really annoying. If you are using glue as part of your assets rebuild process and you want consistent executions over time, using configuration files could be a good idea.

The only thing you need to do is create a file named `sprite.conf` inside your sprite folder (or project folder if you want to apply this settings to your entire project) and glue will override your command line options using these settings. Project-level and sprite-level configuration files can coexist:

```
sprites
    -- actions
    |   -- add.png
    |   -- remove.png
    |   -- sprite.conf
    -- icons
    |   -- comment.png
    |   -- new.png
    |   -- rss.png
    -- sprite.conf
```

If for example you want to change the namespace and the default padding to all your sprites you can add this to your project-level `sprite.conf`:

```
[sprite]
namespace=my-sprites
padding=20
```

If the `actions` images needs to be cropped and have a different padding, you can create add the following settings to your new `actions/sprite.conf` file:

```
[sprite]
crop=true
padding=10
```

If the `remove.png` image needs to have `10px` margin and `0px` padding you can append a new section to your `actions/sprite.conf` like the following:

```
[remove.png]
margin=10
padding=0
```

This will override any previous setting about `margin` or `padding` affecting `remove.png`.

Note: project-level, sprite-level and image-level settings override any environmnet or command-line settings. More information in the settings section

---

1.5.2 Available configuration

Configuration File setting	Project-level	Sprite-level	Image-level
source			
output			
quiet			
watch			
project			
recursive	X	X	
follow_links	X	X	
force	X	X	
algorithm	X	X	
algorithm_ordering	X	X	
css_dir	X	X	
css_namespace	X	X	
css_sprite_namespace	X	X	
css_url	X	X	
css_cachebuster	X	X	
css_cachebuster_filename	X	X	
css_separator	X	X	
css_template	X	X	
css_pseudo_class_separator	X	X	
less_dir	X	X	
scss_dir	X	X	
img_dir	X	X	
generate_image	X	X	
png8	X	X	
ratios	X	X	
html_dir	X	X	
cocos2d_dir	X	X	
caat_dir	X	X	
json_dir	X	X	
json_format	X	X	
crop	X	X	X
padding	X	X	X
margin	X	X	X

Note: You can't enable output formats using configurations files. If for example you add `less_dir` to your `sprite.conf` this would only override `less` output fodler if less is already an enabled output format.

1.6 Templates

1.6.1 Introduction

`glue` formats based on templates can be customized usign your own templates. By convention, every format (i.e `css`) will define an optional `--css-template` with wich you can override the default template.

These templates are simple Jinja2 templates, so you can customize a far as you want using the following context variables.

Note: By default glue will use it's own internal templates, so you don't need to provide a template unless you want to super-customize glue's output.

Note: If you don't know if you need a custom template, you **don't** need a custom template.

1.6.2 BaseTextFormat

Global

Variable	Value
version	Glue version
hash	Hash of the sprite
name	Name of the sprite
sprite_path	Sprite path
sprite_filename	Sprite filename
width	Sprite width
height	Sprite height
images	List of `Images` inside the sprite
ratios	List of the `Ratios` inside

Image

Variable	Value
filename	Image original filename
last	Last Image in the sprite
x	X position within the sprite
y	Y position within the sprite
width	Image width
height	Image height

Ratio

Variable	Value
ratio	Ratio value
fraction	Nearest fraction for this ratio
sprite_path	Sprite Image path for this ratio

1.6.3 CssFormat

Image

Variable	Value
label	CSS label for this image
pseudo	CSS pseudo class (if any)

1.6.4 HtmlFormat

Global

Variable	Value
css_path	Path where the css file is

1.6.5 Templates Examples

If you are going to create a new template from scratch or if you want to do some changes to an existing output, a good starting point would be to read some of the existing templates in the `formats` folder.

1.7 Command line arguments

1.7.1 -a –algorithm

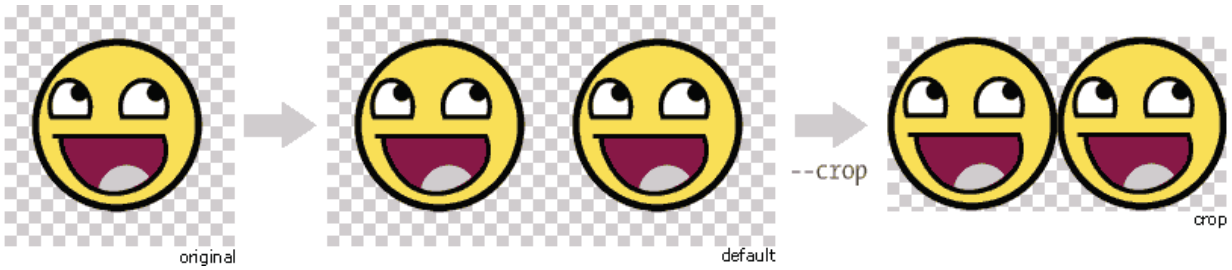
The criteria that `glue` uses to order the images before adding them to the canvas can be tunned. By default the algorithm is *square*, but in some situations using another ordering like *vertical* or *horizontal* could be useful depending on the kind of images you are spriting.

- The *square* algorithm was inspired by the Binary Tree Bin Packing Algorithm Article by Jake Gordon.
- The *vertical* one allocates the images vertically aligning them to the left of the sprite.
- The *vertical-right* one allocates the images vertically aligning them to the right of the sprite.
- The *horizontal* one allocates the images aligning them to the top of the sprite.
- The *horizontal-bottom* one allocates the images aligning them to the bottom of the sprite.
- The *diagonal* one allocates the images diagonally. It was inspired by the Diagonal CSS Sprites Article by Aaron Barker.

```
$ glue source output --algorithm=[square|vertical|hortizontal|diagonal|vertical-right|horizontal-bott
```

1.7.2 -c –crop

Usually designers add some unnecessary transparent space around the images because it is easier for them to work with a larger canvas. `glue` can optimize our sprite by croping all the unnecessary transparent spaces that the original images could have.




```
$ glue source output --crop
```

1.7.3 –caat

Using the `--caat` option, `Glue` will generate both a sprite image and a caat metadata file.

```
$ glue source output --caat
```

1.7.4 –cachebuster

If you decide to add an expires header to your static resources (and if you haven't already you really should), you need to worry about cache busting these resources every time you change one of them.

Cache busting is a technique that prevents a browser from reusing a resource that was already downloaded and cached. Cache in general is good, but in some situations could be annoying if it's duration is too long and we want to update a resource **now**.

This technique adds a flag to every url that links an external resource (PNG in this case). This flag usually is the last modified time or the `hash` of the file.

`glue` can use this technique to automatically add the `hash` of the PNG file to the CSS url, so as soon as the file change (add/remove an image) the `hash` will be different and the browser will re-download the image.

```
$ glue source output --cachebuster
```

Original css:

```
.sprite-icons-zoom{ background:url('sprites/icons/icons.png'); top:0; left:0; no-repeat;}
.sprite-icons-wrench_orange{ background:url('sprites/icons/icons.png'); top:0; left:-16; no-repeat;}
...
```

After –cachebuster:

```
.sprite-icons-zoom{ background:url('sprites/icons/icons.png?p3c54d'); top:0; left:0; no-repeat;}
.sprite-icons-wrench_orange{ background:url('sprites/icons/icons.png?p3c54d'); top:0; left:-16; no-re
...
```

1.7.5 –cachebuster-filename

This option has the same purpose than `--cachebuster` but insted of using the hash of the PNG as a queryarg it uses it as part of the filename.

```
$ glue source output --cachebuster-filename
```

Original css:

```
.sprite-icons-zoom{ background:url('sprites/icons/icons.png'); top:0; left:0; no-repeat;}
.sprite-icons-wrench_orange{ background:url('sprites/icons/icons.png'); top:0; left:-16; no-repeat;}
...
```

After –cachebuster:

```
.sprite-icons-zoom{ background:url('sprites/icons/icons_p3c54d.png'); top:0; left:0; no-repeat;}
.sprite-icons-wrench_orange{ background:url('sprites/icons/icons_p3c54d.png'); top:0; left:-16; no-re
...
```

1.7.6 –cachebuster-filename-only-sprites

Unlike `--cachebuster-filename`, glue will only apply filename cachebusting to the sprite image and not to both the `CSS` and the sprite image.

```
$ glue source output --cachebuster-filename-only-sprites
```

Note: New in version 0.9.2

1.7.7 –cocos2d

Using the `--cocos2d` option, `Glue` will generate both a sprite image and a xml metadata file compatible with cocos2d.

```
$ glue source output --cocos2d
```

Note: New in version 0.9

Note: The output of this format has not been deeply tested and we are looking for a cocos2d-champion who can sponsor this feature.

1.7.8 –css –img

Usually both CSS and PNG files reside on different folders, e.g. *css* and *img*. If you want to choose an individual folder for each type of file you can use the `--img=<dir> --css=<dir>` options together to customize where the output files will be created.

```
$ glue source --img=images/compiled --css=css/compiled
```

1.7.9 –css-template

While using `--css` you can use your own css template using `--css-template=<FILE>`.

Note: By default glue will use it's own internal `css` template, so this command **is not required** unless you want to super-customize glue's `css` output using **your own** template. You can find further documentation about how templates work in the templates documentation page.

```
$ glue source output --css-template=my_template.jinja
```

1.7.10 –force

By default `glue` store some metadata inside the generated sprites in order to not rebuild it again if the source images and settings are the same. Glue set two different keys, `glue` with the version number the sprite was build and `hash`, generated using the source images data, name and all the relevant sprite settings like padding, margin etc...

In order to avoid this behaviour you can use `--force` and `glue` will always build the sprites.

```
$ glue source output --force
```

1.7.11 –follow-links

Follow symbolic links.

```
$ glue source output --follow-links
```

Note: Be aware that following links can lead to infinite recursion if a link points to a parent directory of itself. `glue` does not keep track of the directories it visited already.

1.7.12 –html

Using the `--html` option, `Glue` will also generate a test html per sprite using all the available CSS classes. This option is only useful for testing purposes. Glue generate the `html` file in the same directory as the CSS file.

```
$ glue source output --html
```

1.7.13 –json

Using the `--json` option, `Glue` will generate both a sprite image and a json metadata file.

```
$ glue source output --json
```

1.7.14 –json-format

Using the `--json-format` option you can customize how the generated `JSON` will look. You can choose between `array` and `hash`.

```
$ glue source output --json --json-format=hash
```

Example `array` output:

```
{"frames": [{"filename": "apple.png", width": 128, "height": 128, ...}, {...}], "meta": {...}}
```

Example `hash` output:

```
{"frames": {"apple.png": {"width": 128, "height": 128, ...}, "orange.png": {...}, "meta": {...}}
```

1.7.15 -l –less

less is a dynamic stylesheet language that extends CSS with dynamic behaviors. `glue` can also create `.less` files adding the `--less` option. This files contain exactly the same CSS code. This option only changes the file format.

```
$ glue source output --less
```

1.7.16 –less-template

While using `--less` you can use your own less template using `--less-template=<FILE>`.

Note: By default glue will use it's own internal `less` template, so this command **is not required** unless you want to super-customize glue's `less` output using **your own** template. You can find further documentation about how templates work in the templates documentation page.

```
$ glue source output --less-template=my_template.jinja
```

Note: New in version 0.9.2

1.7.17 –margin

If you want to spread the images around the sprite but you don't want to count this space as image width/height (as happens using *–padding*'), you can use the `--margin` option followed by the margin you want to add:

```
$ glue source output --margin=10
$ glue source output --margin='10 20'
$ glue source output --margin='10 20 30 40'
```

Note: New in version 0.9

1.7.18 –namespace

By default `glue` adds the namespace `sprite` to all the generated CSS class names. If you want to use your own namespace you can override the default one using the `--namespace` option.

```
$ glue source output --namespace=my-namespace
```

If you want to completely remove the namespace (both the global and the sprite part) you can use:

```
$ glue source output --sprite-namespace= --namespace=
```

1.7.19 –no-img

Don't create any sprite image.

```
$ glue source output --no-img
```

1.7.20 –no-css

Don't create any CSS file.

```
$ glue source output --no-css
```

1.7.21 –ordering

Before processing the images using the *algorithm* glue orders the images. The default ordering is *maxside* but you can configure it using the `--ordering` option.

```
$ glue source output --ordering=[maxside|width|height|area|filename]
```

You can reverse how any of the available algorithms works prepending a -.

```
$ glue source output --ordering=[-maxside|-width|-height|-area|-filename]
```

1.7.22 -p –padding

If you want to add the same padding around all images you can use the `--padding` option:

```
$ glue source output --padding=10
$ glue source output --padding=10 20
$ glue source output --padding=10 20 30 40
```

1.7.23 –png8

By using the flag `png8` the output image format will be png8 instead of png32.

```
$ glue source output --png8
```

Note: This feature is unstable in OSX > 10.7 because a bug in PIL.

1.7.24 –project

As it's explained at the quickstart page the default behaviour of `glue` is to handle one unique sprite folder. If you need to generate several sprites for a project, you can use the `--project` option to handle multiple folders with only one command.

The suggested setup is to create a new folder for every sprite, and add inside all the images you need for each one. `glue` will create a new sprite for every folder:

```
images
-- actions
|   -- add.png
|   -- remove.png
-- borders
|   -- top_left.png
|   -- top_right.png
-- icons
    -- comment.png
```

```
    -- new.png
    -- rss.png
```

```
$ glue source output --project
```

1.7.25 –pseudo-class-separator

As it's explained at the pseudo-classes page using the filename of the source images you can customize the pseudo class related to the images, so if you simply append `__hover` to the filename `glue` will add `:hover` to the CSS class name.

Since `glue 0.9` this separator is `__` but for previous version it use to be only `_`. In order to not make `glue < 0.9` users rename their images, `glue 0.9` introduces this new option so you can customize the separator.

```
$ glue source output --pseudo-class-separator=_
```

1.7.26 -q –quiet

This flag will make `glue` suppress all console output.

```
$ glue source output -q
```

1.7.27 -r –recursive

Read directories recursively and add all the images to the same sprite.

Example structure:

```
source
-- actions
|   -- add.png
|   -- remove.png
-- borders
|   -- top_left.png
|   -- top_right.png
-- icons
    -- comment.png
    -- new.png
    -- rss.png
    -- blog
        -- rss.png
        -- atom.png
```

If you want to create only one sprite image you should use.

```
$ glue source output --recursive
```

On the other hand if you want to create three different sprites (one per folder) you can combine `--project` and `--recursive`.

```
$ glue source output --recursive --project
```

1.7.28 –ratios

`Glue` can automatically scale down your sprites to automatically fit them into low-dpi devices. `Glue` assumes that the source images are the biggests you want to serve, then `glue` will create one sprite for each ratio you set in this command. For more information, read Retina Sprites: Ratios.

```
$ glue source output --ratios=2,1
$ glue source output --ratios=2,1.5,1
```

1.7.29 –retina

The option `--retina` is only a shortcut for `--ratios=2,1`.

```
$ glue source output --retina
```

1.7.30 -s –source -o –output

There are two ways to choose which are the `source` and the `output` directories. Using the first and the second positional arguments is the traditional way of using `glue` but in order to standardize how configuration is handled `glue 0.9` intruduced these two new options.

```
$ glue output --source=DIR --output=DIR
```

1.7.31 –scss

scss/sass is another dynamic stylesheet language that extends CSS with dynamic behaviors. `glue` can also create `.scss` files adding the `--scss` option. This files contain exactly the same CSS code. This option only changes the file format.

```
$ glue source output --scss
```

Note: New in version 0.9

1.7.32 –scss-template

While using `--scss` you can use your own less template using `--scss-template=<FILE>`.

Note: By default glue will use it's own internal `scss` template, so this command **is not required** unless you want to super-customize glue's `scss` output using **your own** template. You can find further documentation about how templates work in the templates documentation page.

```
$ glue source output --scss-template=my_template.jinja
```

Note: New in version 0.9.2

1.7.33 –separator

`glue` by default uses `-` as separator for the CSS class names. If you want to customize this behaviour you can use `--separator` to specify your own one:

```
$ glue source output --separator=_
```

If you want to use camelCase instead of a separator, choose `camelcase` as separator.

```
$ glue source output --separator=camelcase
```

1.7.34 –sprite-namespace

By default `glue` adds the sprite's name as past of the CSS class namespace. If you want to use your own namespace you can override the default one using the `--sprite-namespace` option.

```
$ glue source output --sprite-namespace=custom
```

As part of the new sprite namespace you can use the key `%(sprite)s` to refer to the original namespace.

If you want to completely remove the namespace (both the global and the sprite part) you can use:

```
$ glue source output --sprite-namespace= --namespace=
```

1.7.35 -u –url

By default `glue` adds to the PNG file name the relative url between the CSS and the PNG file. If for any reason you need to change this behaviour, you can use `url=<your-static-url-to-the-png-file>` and `glue` will replace its suggested one with your url.

```
$ glue source output --url=http://static.example.com/
```

1.7.36 –watch

While you are developing a site it could be quite frustrating running `Glue` once and another every time you change a source image or a filename. `--watch` will allow you to keep `Glue` running in the background and it'll rebuild the sprite every time it detects changes on the source directory.

```
$ glue source output --watch
```

1.8 Settings

1.8.1 Settings Priority

Remember that environment variables would override glue's defaults but their priority is lower than command line options.

From higher priority to lower priority:

1. Image settings (from configuration file)
2. Sprite settings (from configuration file)

3. Command line settings
4. Environment variables
5. Default settings

Every command-line option available in glue is configurable using environment variables.

Note: New in version 0.9

1.8.2 Settings Map

Command-line arg	Environment Variable	Configuration File setting
–source	GLUE_SOURCE	source
–output	GLUE_OUTPUT	output
-q –quiet	GLUE_QUIET	quiet
-r –recursive	GLUE_RECURSIVE	recursive
–follow-links	GLUE_FOLLOW_LINKS	follow_links
-f –force	GLUE_FORCE	force
-w –watch	GLUE_WATCH	watch
–project	GLUE_PROJECT	project
-a –algorithm	GLUE_ALGORITHM	algorithm
–ordering	GLUE_ORDERING	algorithm_ordering
–css	GLUE_CSS	css_dir
–less	GLUE_LESS	less_dir
–less-template	GLUE_LESS_TEMPLATE	less_template
–scss	GLUE_SCSS	scss_format
–scss-template	GLUE_SCSS_TEMPLATE	scss_template
–namespace	GLUE_CSS_NAMESPACE	css_namespace
–sprite-namespace	GLUE_CSS_SPRITE_NAMESPACE	css_sprite_namespace
-u –url	GLUE_CSS_URL	css_url
–cachebuster	GLUE_CSS_CACHEBUSTER	css_cachebuster
–cachebuster-filename	GLUE_CSS_CACHEBUSTER	css_cachebuster_filename
–separator	GLUE_CSS_SEPARATOR	css_separator
–css-template	GLUE_CSS_TEMPLATE	css_template
–pseudo-class-separator	GLUE_CSS_PSEUDO_CLASS_SEPARATOR	css_pseudo_class_separator
–img	GLUE_IMG	img_dir
–no-img	GLUE_GENERATE_IMG	generate_image
–no-css	GLUE_GENERATE_CSS	generate_css
-c –crop	GLUE_CROP	crop
-p –padding	GLUE_PADDING	padding
–margin	GLUE_MARGIN	margin
–png8	GLUE_PNG8	png8
–ratios	GLUE_RATIOS	ratios
–retina	GLUE_RETINA	ratios
–html	GLUE_HTML	html_dir
–cocos2d	GLUE_COCOS2D	cocos2d_dir
–json	GLUE_JSON	json_dir
–json-format	GLUE_JSON_FORMAT	json_format
–caat	GLUE_CAAT	caat_dir

1.9 FAQ

1.9.1 Errors compiling PIL in Snow Leopard

```
/usr/libexec/gcc/powerpc-apple-darwin10/4.2.1/as: assembler (/usr/bin/../libexec/as/ppc/as or
/usr/bin/../local/libexec/as/ppc/as) for architecture ppc not installed

Installed assemblers are:

/usr/bin/../libexec/as/x86_64/as for architecture x86_64

/usr/bin/../libexec/as/i386/as for architecture i386

/usr/bin/../libexec/as/arm/as for architecture arm

_imaging.c:3017: warning: initialization from incompatible pointer type

_imaging.c:3077: warning: initialization from incompatible pointer type

_imaging.c:3281: fatal error: error writing to -: Broken pipe

compilation terminated.

_imaging.c:3017: warning: initialization from incompatible pointer type

_imaging.c:3077: warning: initialization from incompatible pointer type

lipo: can't open input file: /var/tmp//ccsCS1Iv.out (No such file or directory)

error: command 'gcc-4.2' failed with exit status 1
```

The reason for this error is that Apple has removed from Xcode the assembler for PPC, while the core system retains their PPC images in the fat binaries. If you run `file /usr/bin/python` you will likely find the following:

```
/usr/bin/python: Mach-O universal binary with 3 architectures
/usr/bin/python (for architecture x86_64):  Mach-O 64-bit executable x86_64
/usr/bin/python (for architecture i386):    Mach-O executable i386
/usr/bin/python (for architecture ppc7400): Mach-O executable ppc
```

Python compiles C extensions with the same compiler flags that Python itself was compiled with.

The solution? Install `glue` using this line:

```
$ sudo env ARCHFLAGS='-arch i386 -arch x86_64' pip install glue
```

1.10 Changelog

1.10.1 0.11.1

- Fix packaging issue. #192

1.10.2 0.11.0

- Fix incorrect order in JSON output #186

- Stop using utf-8-sig #162
- Fix several typos in the documentation #165 #181 #182

1.10.3 0.10.0

- Added support to `python 3.3+`

1.10.4 0.9.4

- Make glue only require argparse if sys.version_info is < 2.7 (Thanks Lorenzo)
- Make glue read project-level configuration files #139 (Thanks Ady Liu)
- Fix `--less-template` and `--scss-template` options.

1.10.5 0.9.3

- Make the css output prettier #137 (Thanks uberrobert).

1.10.6 0.9.2

- Fix transparent images cropping #133 (Thanks lzubiaur).
- Fix CSS format paths on windows #132 (Thanks Most).
- Fix –cachebuster-filename #135 (Thanks monsanto).
- New image ordering `filename`.
- Fix less retina output #122.
- New option: `--cachebuster-filename-only-sprites` #113.
- Fix existing `--cachebuster-filename` as it was not cachebusting css files since `0.9` #113.

1.10.7 0.9.1

- Fix issue with cocos2d output format #131 (Thanks lzubiaur).

1.10.8 0.9

- This version solves lots of design flaws glue had since it was created two years ago. At the very beginning, glue was created as a single-file script to create css sprites. After two years of development, an outstanding support from the community and an incredible support from lots of companies it's time to refurbish glue from its foundations ensuring it lasts over time.
- This version introduce a new concept named `Formats`. Every output file generated by glue now is created using a format. This change decouples glue's core (as far as possible) from how the output is created allowing glue developers to create different formats without messing glue's core. Fixing #64
- Glue now support two new output formats (cocos2d, json and CAAT). CSS will still be the default output.
- New project layout, algorithms formats and manager now have their own packages.

- Glue now uses `argparse` instead of `optparse`.
- Glue now requires `Pillow>=2.2`. Fixing #99
- Glue now requires `Jinja2>=2.7`.
- Glue now uses utf-8 as default encoding for css files. #65
- New (fully rewritten) test suite and test framework in order to make writing tests easier.
- `--margin` now supports more than only one margin, you can now use it with (e.g.) `--margin=10 20 10 20`. Fixes #101
- New `setup.py` fixing #110
- New options `--source` and `--output` will now complement the first and second positional argument.
- New option `--css-template` will allow you to choose your own css output jinja template.
- New option `--scss` will use `scss` as file extension instead of `css`. Fixing #86
- New option `--json-format` will allow to customize the json structure generated by `--json`.
- New feature: Every settting is now configurable using environment variables and configuration files.
- New option `--pseudo-class-separator=_`.
- Glue now support multiple css pseudo-classes applied to the same image #87.
- Glue css pseudo-class separator is now ___ instead of _. If you want to make `glue` work in legacy mode use the new option `--pseudo-class-separator=_`.
- Glue will not validate css output if `--no-css` is present #78.
- Glue CSS media-queries now includes `min-resolution` See https://developer.mozilla.org/en-US/docs/Web/Guide/CSS/Media_queries#-moz-device-pixel-ratio.
- Several bugs fixed like #70 #93 #94 #108 #109 #110 #115
- This version includes **several** backward incompatible changes - **read this document carefully before upgrading**.
  - Glue will **not** extract padding information from filenames. If you want to customize paddings individualy (per image), you should use configuration files.
  - `--imagemagick` and `--imagemagickpath` are now deprecated. These options were introduced as a workaround in order to solve some scaling glitches caused by `PIL`. Pillow `2.2.0` solves these issues.
  - `--each-template`, `--global-template` and `--ratio-template` are now deprecated. CSS output customization can be easily done now using `--css-template=template.jinja`.
  - `--ignore-filename-paddings` is now deprecated. Per-file customization can be easily (and more scalably) done now using configuration files.
  - `--optipng` and `--optipngpath` are now deprecated. There are lots of png optimization libraries and it would be silly to support all of them. Feel free to optimize your png files using your favourite one.
  - `--debug` is now deprecated. If an unhandled Exception is triggered, glue will now automatically show some friendly debug information.

1.10.9 0.4.1

- Make glue require Pillow >= 2.2.2 in order to make it work on Mavericks (Thanks wyuenho).

1.10.10 0.4

- This version is a transition between glue `0.3` and glue `0.9`.
- The following arguments will now show a deprecation warning:
  - `--imagemagick`
  - `--imagemagickpath`
  - `--global-template`
  - `--each-template`
  - `--ratio-template`
  - `--ignore-filename-paddings`
  - `--optipng`
  - `--optipngpath`
  - `--debug`

1.10.11 0.3

- New `--imagemagick` option. If present, glue will use ImageMagick to scale down retina sprites instead of Pillow #72.
- New `--imagemagickpath` option #72.
- Soft 2px default for margin no longer exists #73.
- Fix how glue choose which classes to add to the global scope in order to add pseudo-classes if needed #77.
- Fix camelcase separator as it wasn't preserving original case #74.
- Fix sprites containing images with filenames included in PSEUDO_CLASSES #59.

1.10.12 0.2.9.1

- Fix ProjectSpriteManager issues.

1.10.13 0.2.9

- Improve error messages.
- Added variable `identifier` to `--each-template`.
- Glue now require `Pillow==1.7.8`

1.10.14 0.2.8.1

- Fix maximum recursion depth issues in `ConfigManager`
- Update Documentation.

1.10.15 0.2.8

- New `--recursive` option.
- New `--follow-links` option.
- New `--sprite-namespace` option.
- Speed up improvement: Glue is now 1.3x faster in a cold run.
- Speed up improvement: Glue is now 14x faster for already created sprites.
- Glue now store some metadata inside the generated sprites in order to not rebuild them again if the source images and settings are the same.
- New `--force` option to make glue rebuild the sprites.
- New `--no-img` and `--no-css` options.
- Fix some CSS aligment issues related with odd sized images.
- A soft default of 2px of margin is going to be added while using glue with `--ratios` or `--retina` in order to fix scaling noise.
- Fix `--url` in order to override relative path calculated by `--img` and `--css`.

1.10.16 0.2.7

- Glue now require Pillow instead of PIL (http://pypi.python.org/pypi/Pillow/)
- Improve compatibility with less allowing variables in the urls (Thanks rafeca).
- Fix cachebuster issues with –retina and –url

1.10.17 0.2.6.1

- Fix bug with images that only contain digits like. Thanks to Russ Ferriday and Paul Hallett.
- Make possible read optipng related configuration from static configuration files.

1.10.18 0.2.6

- Added support for multi-dpi (retina) sprite creation.
- New `--ratios` and `--retina` options.
- New option `--debug`
- Performance improvements. ~10% on big sprites.

1.10.19 0.2.5

- New `--watch` option to keep glue running in the background watching file changes.
- New option `--html` that generates a html using all the available css classes.
- New option `--margin` that adds margins around the sprited images. This margin doesn't count as image size.
- Add MANIFEST.in and tune the setup.py preparing the Debian/Ubuntu package.

- Fix _locate_images to be deterministic.
- Add support to Travis CI.
- Fix 8bit B/W images bug.

1.10.20 0.2.4

- Better error handling: Glue will now return non zero return codes if something goes wrong.

1.10.21 0.2.3

- Fix `--version`
- Fix the camelcase `--separator` to not lowercase the filename before the capitalization.

1.10.22 0.2.2

- New feature: Per-file pseudo-class customization.
- Added support for 8bit bg images.
- Added support for digit-only images.
- Fix newline characters support on `--global-template` and `--each-template`.
- New algoritms `vertical-right` and `horizontal-bottom`.
- New option `--separator`: Customizable CSS class name separator.

1.10.23 0.2.1

- New command line argument `--global-template`.
- New command line argument `--each-template`.
- `-z` and `--no-size` arguments are now deprecated.

1.10.24 0.2

- The default behaviour of glue is now the old `--simple` one.
- The old default behaviour (multiple-sprites) is now accesible using –project
- `--simple` argument is now deprecated
- New ordering algorithms square, horizontal, vertical and diagonal.
- New command line argument `--ordering`.
- New command line argument `--cachebuster-filename`.
- Old algorithms maxside, width, height and area are now orderings.
- Glue now ignore folders that start with a '.'
- CSS files will now avoid using quotes around the sprite filename.
- New `-v`, `--version` option.

- Fix bugs.
- New test suite.

1.10.25 0.1.9

- New command line argument `-z`, `--no-size` to avoid adding the image width and height to the sprite.
- New command line argument `--png8` forces the output image format to be png8 instead of png32.
- Improve CSS parsing performance removing bloat in the CSS.
- Improved documentation.

CHAPTER 2

Indices and tables

- genindex
- modindex
- search