www.learn-cocos2d.com - Tutorial: Bitmap Fonts and Hiero

www.learn-cocos2d.com - Tutorial: Bitmap Fonts and Hiero

Introduction to Bitmap Fonts

What is a bitmap font?

There are two prevalent types of fonts: truetype and bitmap fonts. Truetype fonts are a vector format which means that these fonts scale well from just one set of vector data, and anti-aliasing and other effects can be applied easily to improve readability or otherwise modify the font. The drawback of this approach is that rendering text using truetype fonts is relatively slow.

Bitmap fonts on the other hand have one concrete image, or a portion of a texture atlas, assigned to each letter used by the font. This makes the font's size fixed, it can scale up and down but not without loss of image quality. It also requires images for the letters to be in memory which naturally consume more memory than the vector data of Truetype fonts.

Advantages/Disadvantages

Wikipedia has sums it up nicely:

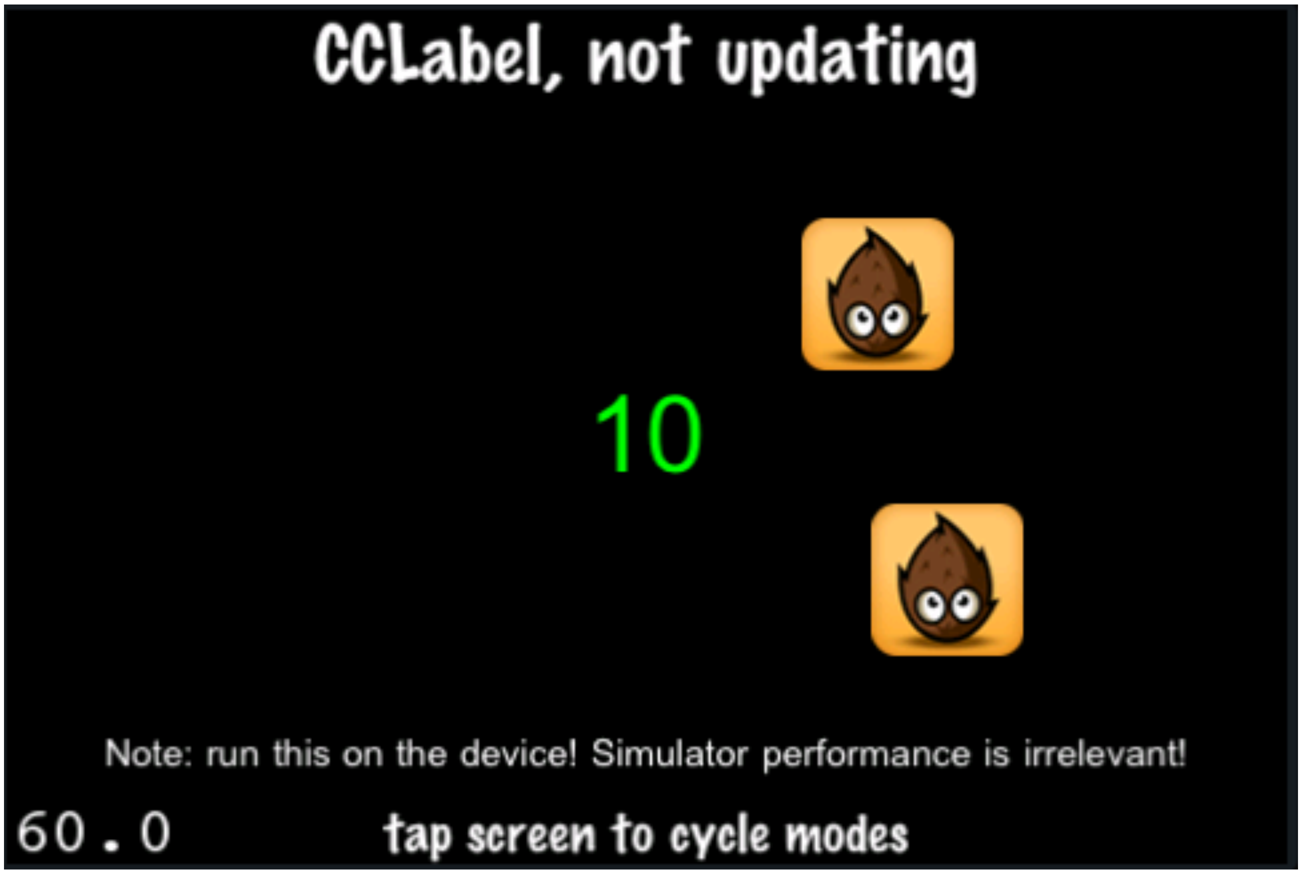
Advantages of bitmap fonts include:
- Extremely fast and simple to render
- Unscaled bitmap fonts always give exactly the same output
- Easier to create than other kinds.

The primary disadvantage of bitmap fonts is that the visual quality tends to be poor when scaled or otherwise transformed, compared to outline and stroke fonts, and providing many optimized and purpose-made sizes of the same font dramatically increases memory usage.

Measuring CCLabel performance (with sample code)

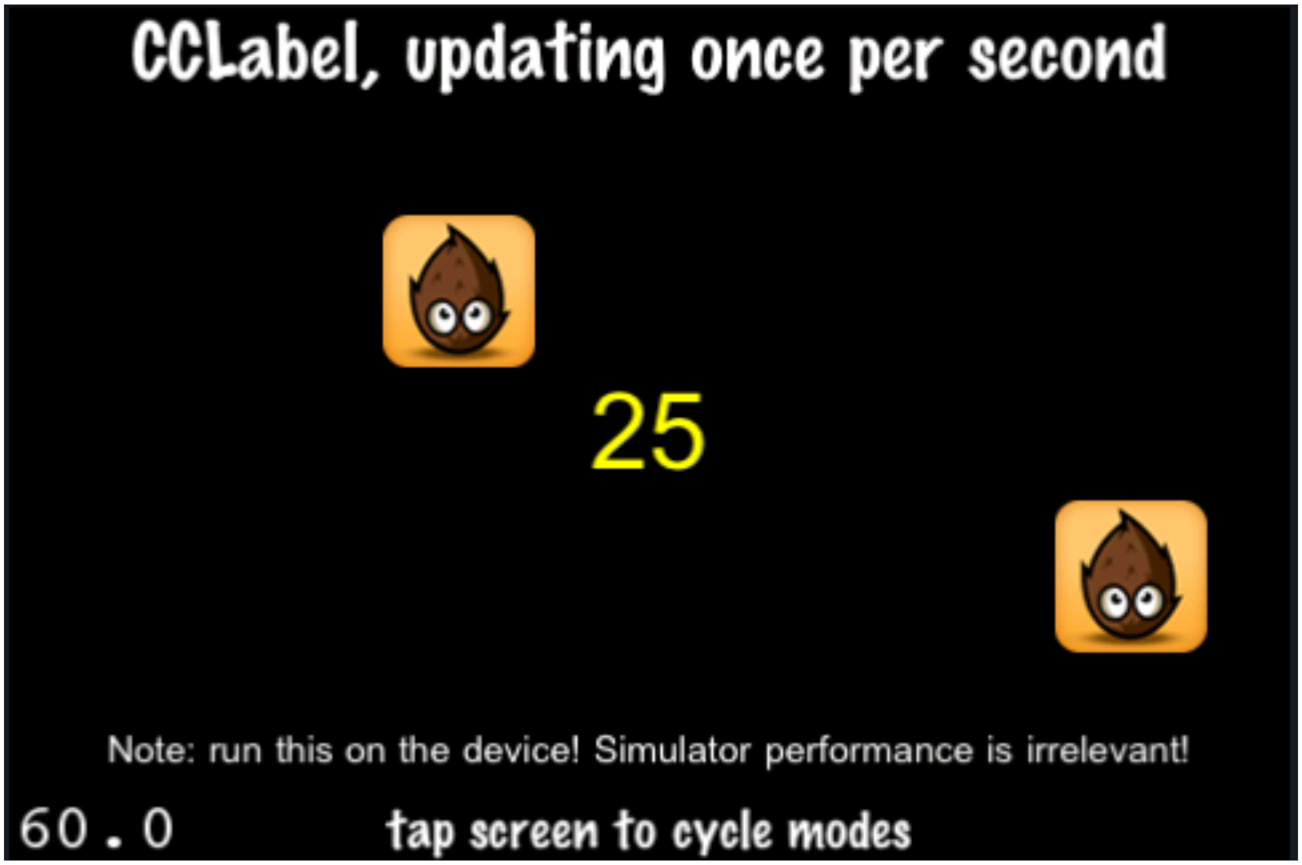
You may have read this statement in the cocos2d API reference: *CCLabel objects are slow. Consider using CCLabelAtlas or CCBitmapFontAtlas instead.*

Why CCLabels are actually fast!


CCLabel, not updating
10
Note: run this on the device! Simulator performance is irrelevant!
60.0
tap screen to cycle modes

If you create a CCLabel object once and then just display it without ever changing its text, it's going to be just as fast as any CCSprite of the same size!

Only changing the text of a CCLabel is slow!


CCLabel, updating once per second
25
Note: run this on the device! Simulator performance is irrelevant!
60.0
tap screen to cycle modes

If you use the

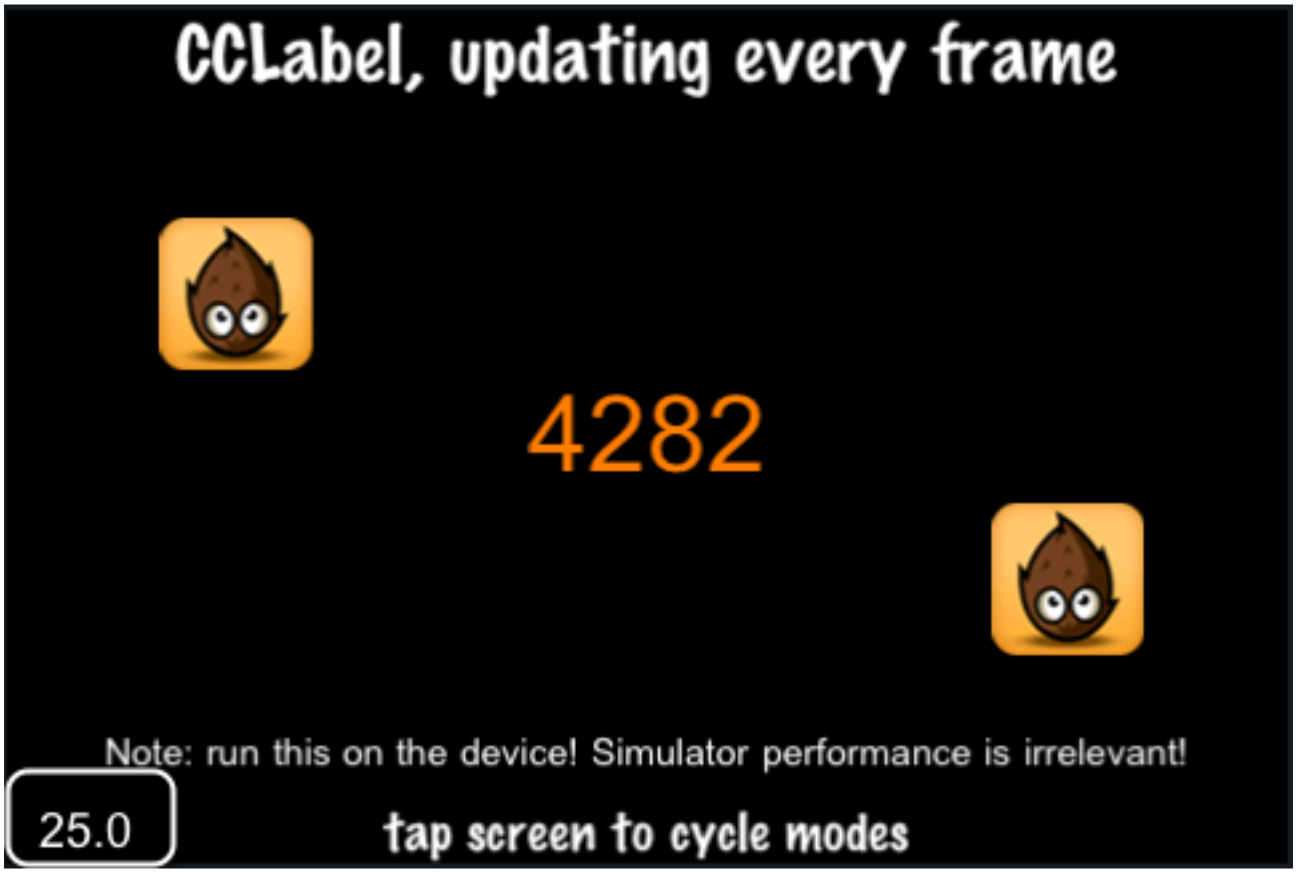
[label setString:@"Hello World!"];

method to change the string or text displayed by the CCLabel, that's where a CCLabel object is slow. It's also slow of course when you first create it, so creating and throwing away a CCLabel object is as slow as changing an existing label's text.

Whenever the text of a CCLabel is changed, it's existing texture with the existing text is thrown away. Then a new texture is generated and the new text is rendered onto the texture using the iPhone SDK's font rendering mechanisms. Altogether this is a process which isn't meant to be used by realtime applications.

In the Speedtest demo you'll notice the moving sprites will jump every second. The fps display is updated too slowly to catch these minute hickups but the human eye will see it!

When a CCLabel is really, really slow ...


CCLabel, updating every frame
4282
Note: run this on the device! Simulator performance is irrelevant!
25.0
tap screen to cycle modes

You may get away with CCLabels and changing their texts if that doesn't happen too often. But even changing a CCLabel once a second can cause regular framerate stuttering. Players will notice this by seeing otherwise smoothly moving objects suddenly jump for a small distance.

Once you get down to updating a CCLabel's text once per frame, even a single CCLabel of size 30 and just one letter can seriously kill your performance. On my iPhone 3GS the framerate drops to 30 fps or below when I update such a label's text every frame. Not good.

Note: *I badly faked the FPS display on the screenshot because the screenshot was taken in Simulator where my Mac is powerful enough to run it at 60 fps (darn, stupid computer, can't do a damn thing right). On my iPhone 3GS however I get about 25 fps. I swear! :)*

CCLabel Speed Test Xcode Project

Download the CCLabel Speedtest project.

Make sure to run it on the device. Measuring or even estimating performance in Simulator is futile. The device is where performance is accurate and measurable, and also where it counts.

I've included a 5 updates per frame mode too, in case the performance drop won't be easily noticeable on 3GS, iPad or iPhone 4 devices.

Measuring CCBitmapFontAtlas performance (with sample code)

Just let me tell you that the same code from the CCLabel performance measurements runs at 60 fps throughout all modes when using a bitmap font.

You can download the Xcode project using CCBitmapFontAtlas here.

You'll be surprised that I only needed to change two lines from CCLabel to CCBitmapFontAtlas and adding the .fnt and .png files for the bitmap fonts. It's really that simple. Included in the project is a switch so that you can try out one of the corrupt fonts with a vertically flipped image, just in case you want to see how that looks. In one word: garbled.

Working with Hiero v2.0 - Bitmap Font Tool

Download Hiero

The official download location is from slick.cokeandcode.com:

Download Hiero.jnlp

If for some reason the above download doesn't work, try this alternate download link:

Alternate Hiero.jnlp download

Save the .jnlp file to your hard drive and run it from there. Directly opening the .jnlp file may cause some malfunctions, I've heard.

Troubleshooting Hiero

Hiero can be a nasty little thing. It has bugs and it behaves different in some aspects than a regular, native Mac OS X app. This troubleshooting section answers these questions. If you don't have any problems, just skip over it.

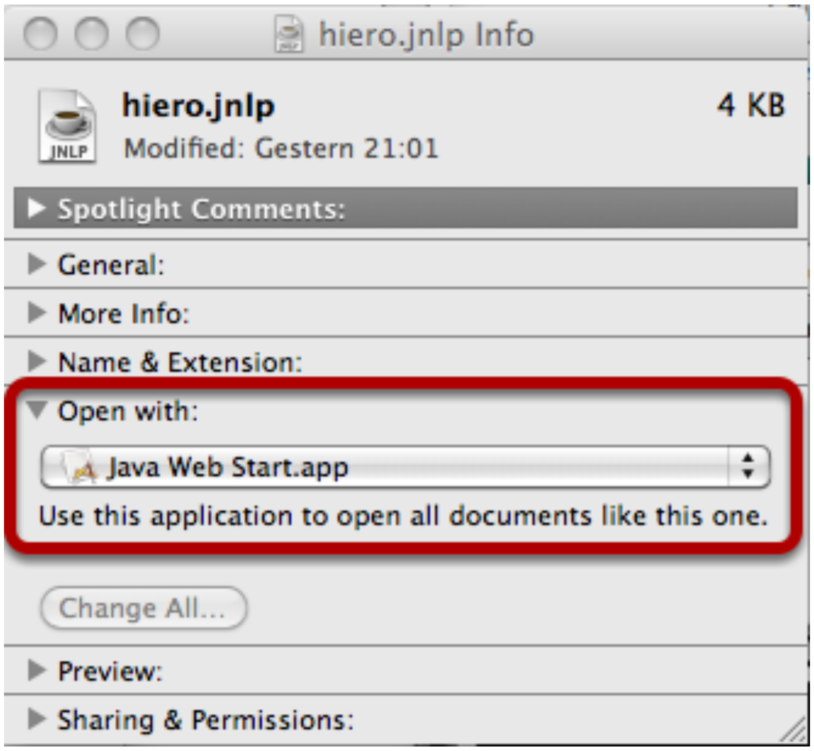
If you can't find the answer to your particular problem, please leave a comment describing your problem and I'll see what I can do to help.

Hiero is asking for security permissions

Yup, it does that. So far no one has noticed any bad behavior. If you don't trust the Hiero application your only option is to run it on a seperate, isolated, non-networked Mac computer and only transfer the .fnt and .png files. It's either that or not using Hiero at all.

If you ask me, I don't feel awfully good about it either. But I understand the problem and I'm willing to give it some trust. It costs money to get these certificates, and Apple knows the risk of running just any Java Webstart application on a computer unless it's signed with a certificate. So that's why there's a security warning.

Hiero won't start!



Modified: Gestern 21:01
Open with:
Java Web Start.app
Use this application to open all documents like this one.

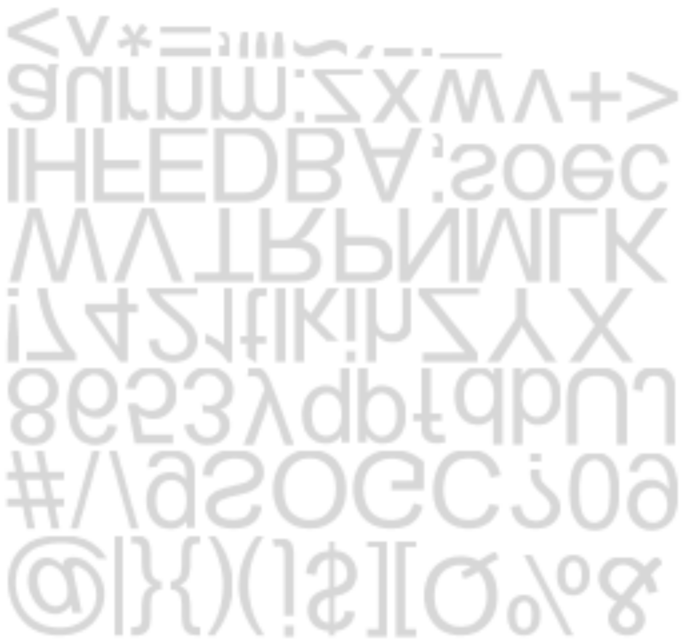
Download and save the hiero.jnlp file to your computer, preferably to your Applications folder. Just to avoid any oddities.
Then locate the file, right-click it and select Get Info.

In the Info dialog make sure it is set to "Open with:" the **Java Web Start.app**

If Java Web Start is not available or it still won't start, you may have to download and install the latest Java for Mac OS X Update. It should be available by choosing Software Update... from the Apple menu. If it's not, here's the download location for the Mac OS X 10.6 Java Update 2.

If it still won't start or crashes it'll be harder to figure out what's causing it. Please leave a comment and post your system details (Mac OS X version), what's happening when you start Hiero including any error message text that appears and what you've tried so far to resolve the problem. Note that I'm not the author of the Hiero program nor am I in contact with the author, I try to help but it's pretty limited what I can do about crashes.

Bitmap Font is a garbled mess or invisible



If the font's .png file looks something like the above picture with the letters mirrored and upside down, then that's the problem. Read the next step to learn how to fix this.

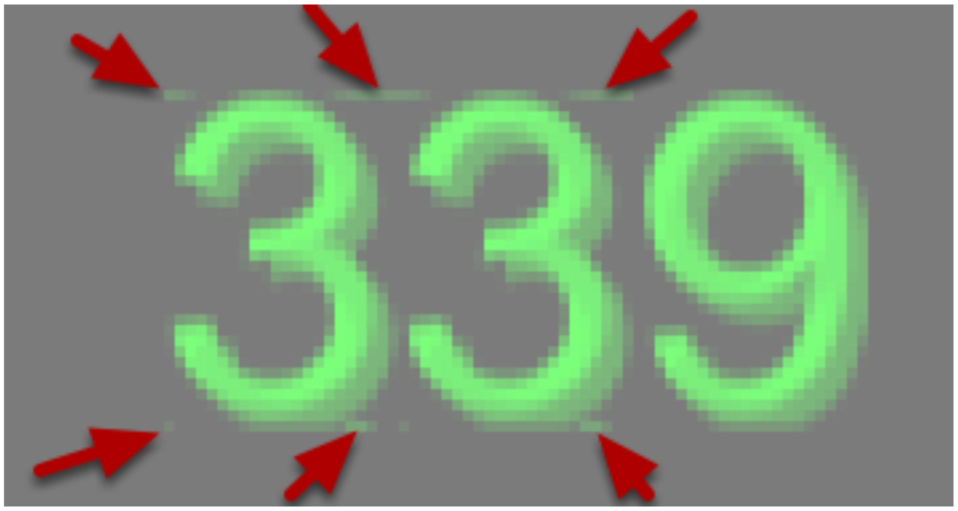
Bitmap Font .png file is flipped vertically (some say: upside-down)

I have no idea why Hiero sometimes creates those images. If you get such a result as the image in the above step, you'll have to fix the .png file manually until someone can figure out how to prevent this from happening.

If you have a suspicion why this is happening or even know how to circumvent it, please leave a comment!

The image file is flipped along the vertical axis. Use any image program (I use Seashore) to load the file, run the "flip vertically" (sometimes: "mirror vertically") function on it and re-save it.

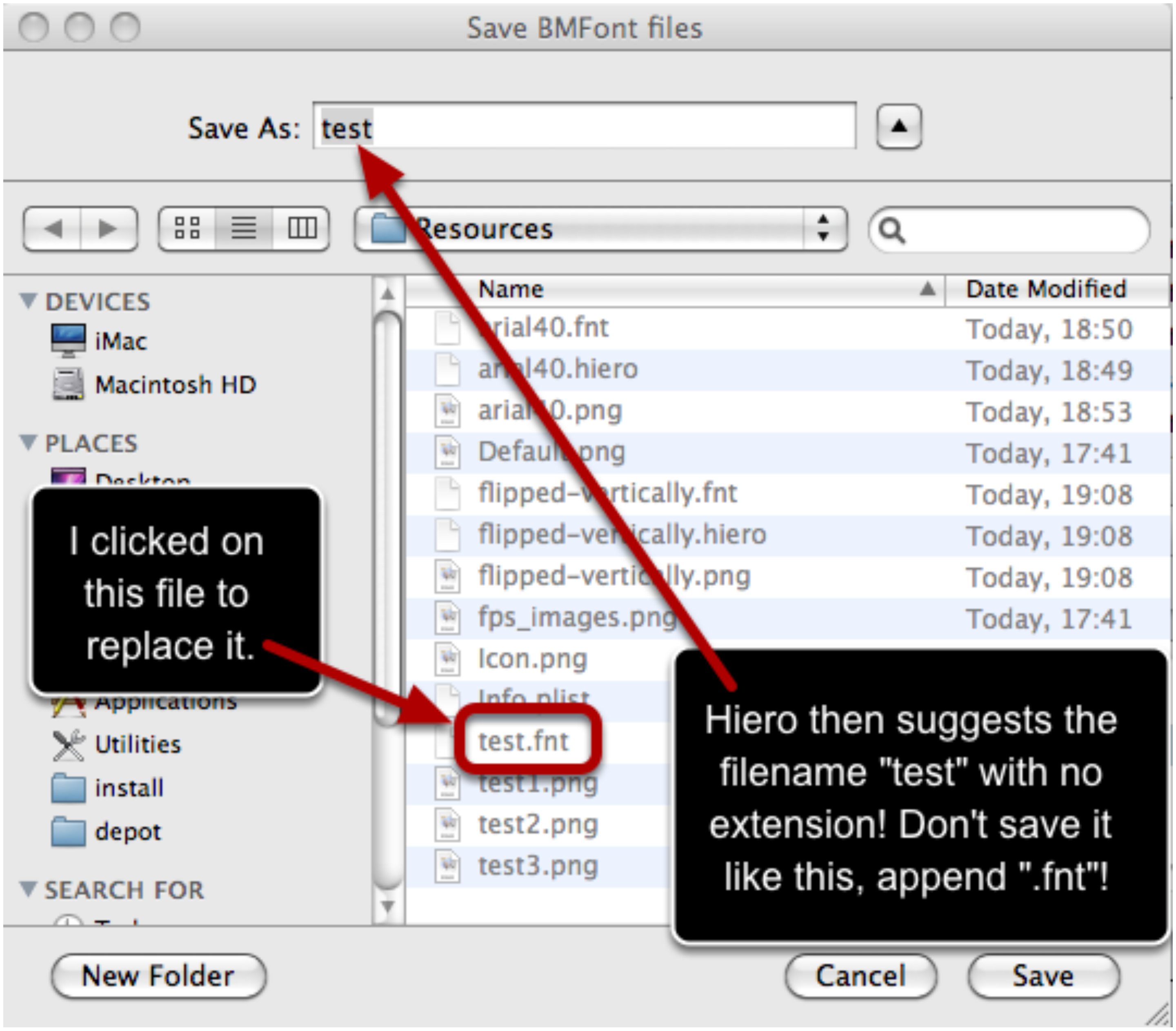
Bitmap Font has weird pixel artifacts!



If your bitmap font shows slight artifacts like in the above magnified, brightness-enhanced screenshot, you have to increase Padding in Hiero and re-save the .fnt file.

In this case it should be sufficient to add padding of 1 pixel at the top and bottom.

Saved files are not overwritten / have no extension



Name	Date Modified
arial40.fnt	Today, 18:50
arial40.hiero	Today, 18:49
arial40.png	Today, 18:53
Default.png	Today, 17:41
flipped-vertically.fnt	Today, 19:08
flipped-vertically.hiero	Today, 19:08
flipped-vertically.png	Today, 19:08
fps_images.png	Today, 17:41
Icon.png	
Info.plist	
test.fnt	
test1.png	
test2.png	
test3.png	

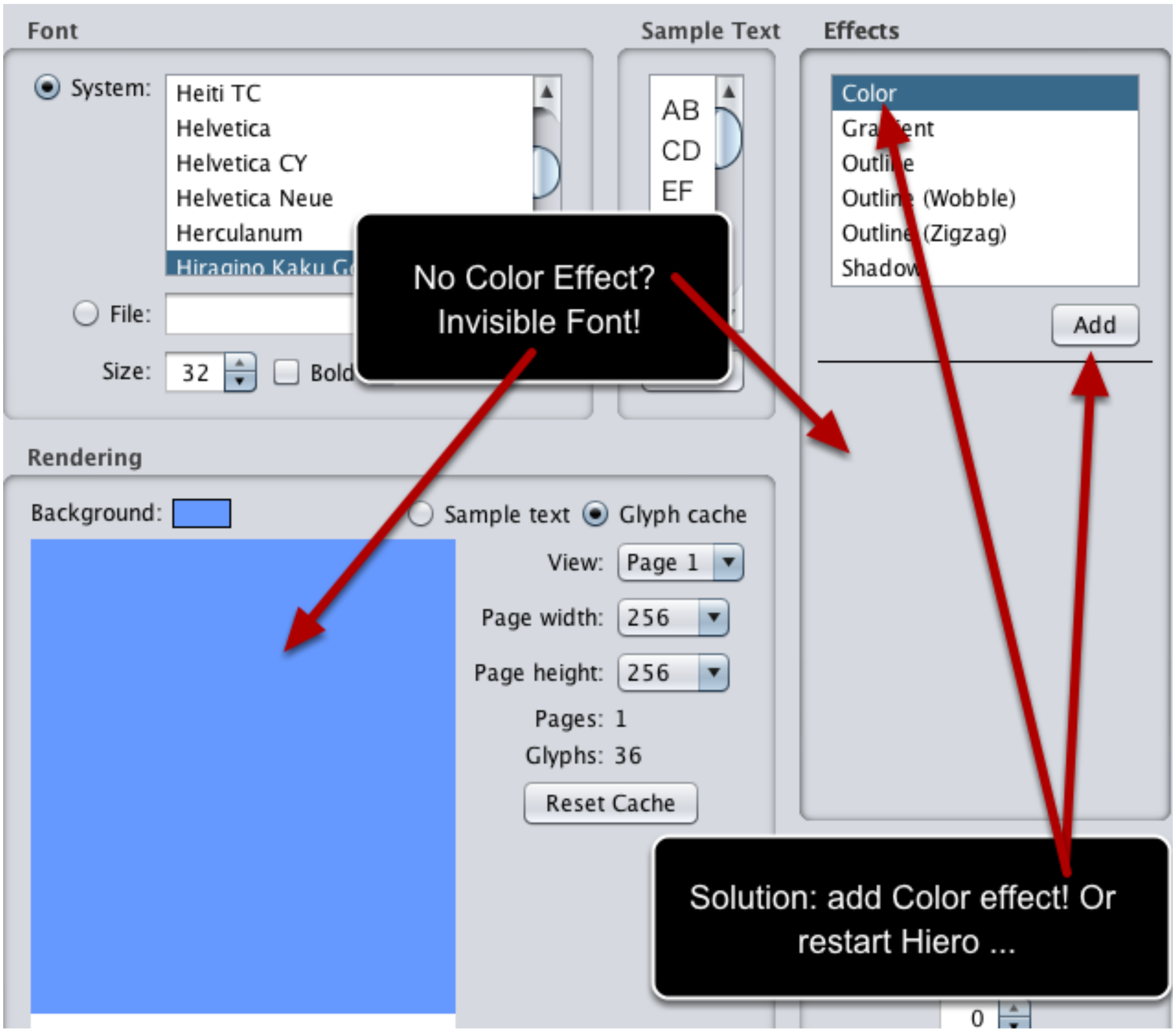
I clicked on this file to replace it.

Hiero then suggests the filename "test" with no extension! Don't save it like this, append ".fnt"!

Hiero has an ugly bug when trying to overwrite existing files. See the screenshot. Normally you'll just click on the existing file and trust the application to replace it. Not so with Hiero, which simply removes the file's extension. This is true for both .fnt and .hiero file formats.

To really replace the existing file you'll have to manually add the correct file extension .fnt or .hiero before saving. You've done it right when you're asked whether to replace the existing file.

Font is not rendered (invisible) in Hiero



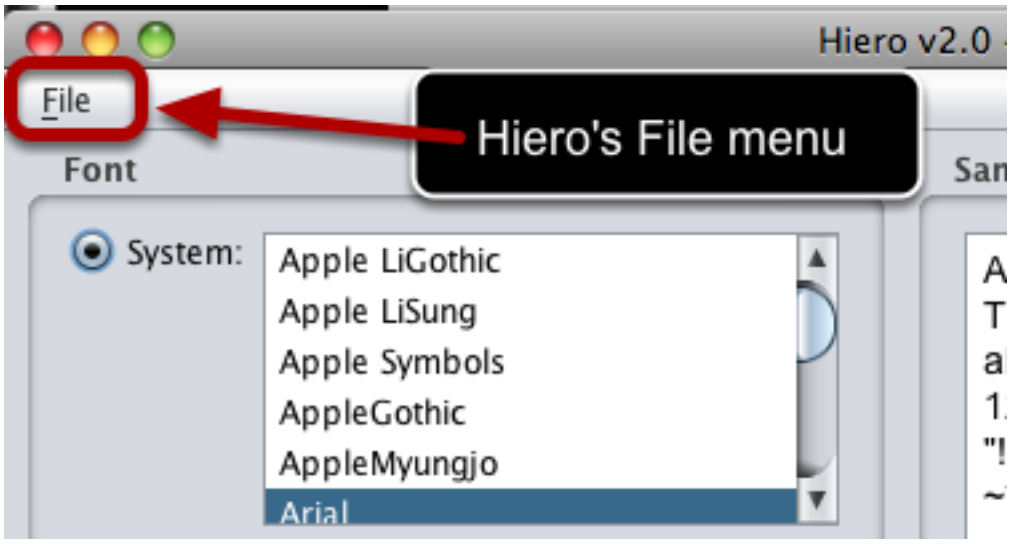
If the Rendering view just contains a flat-colored space with no font in it, please check that the "Color" effect is in use. If it's not, add the Color effect.

In some cases it may also be a display bug. This can happen from time to time. Save your settings to a .hiero file and restart Hiero.

I can't get Hiero to open when double-clicking .hiero files

Yup, nothing you can do about it. It's not a native Mac OS X app, it just doesn't support this.

Hiero doesn't have a File menu!



Yes it does, just not where you expect it. It behaves like a Windows application and has its menu on the top-left corner of the application window.

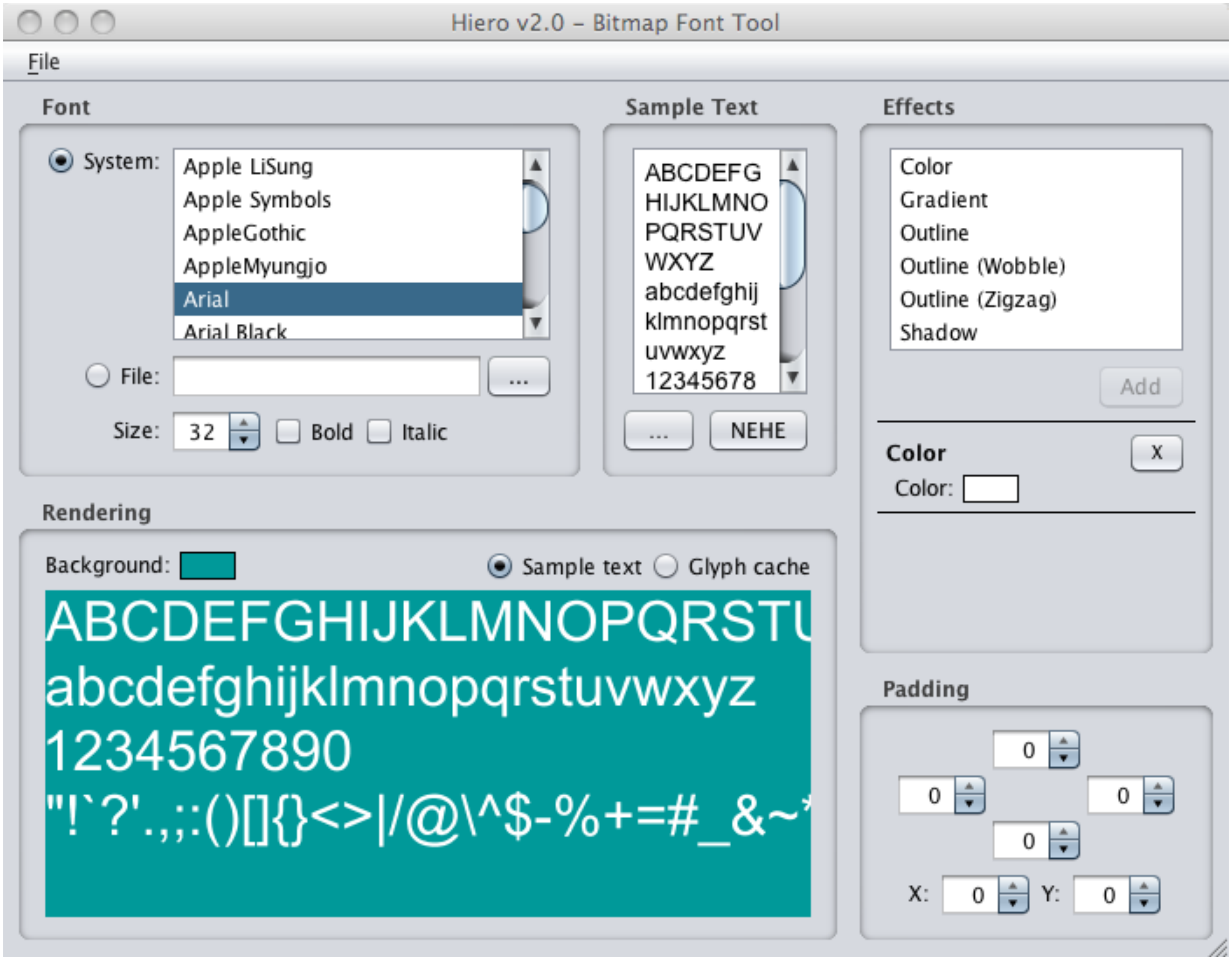
Hiero won't quit/close!

Click the red x icon in the upper left corner of the application window. It behaves like a Windows application. Regular Quit commands are ignored.

Hiero's User Interface

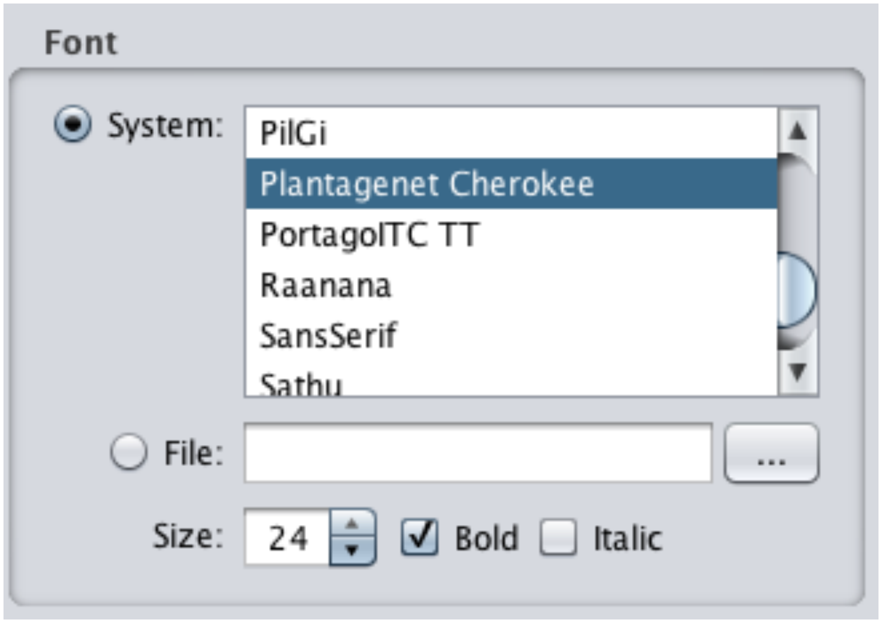
I'll talk about each view seperately since Hiero's User Interface uses the "all-in-one" approach which can be a little overwhelming at first. But once you get past that you can be very productive with it.

Hiero's User Interface



This is the whole Hiero window after you start it. I made it even smaller to not have to scale the screenshot down too much. In general the very first thing I do after starting Hiero is to increase its window size, highly recommended!

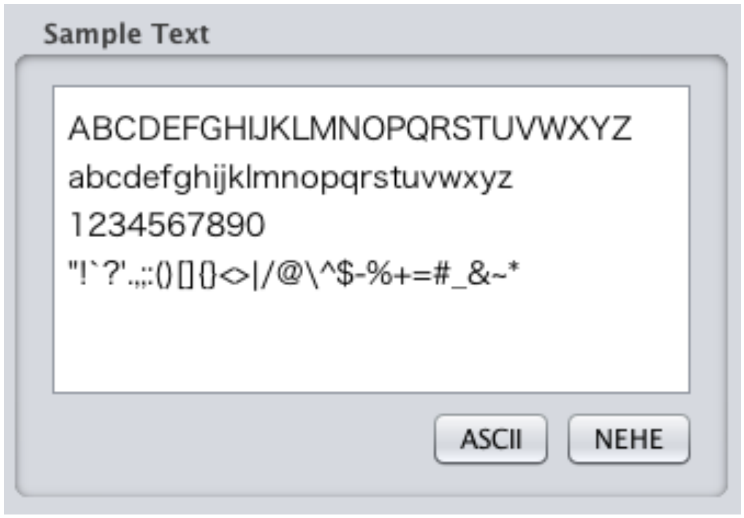
The Font view



Here you select which TrueType font to use as basis for your bitmap font. You can also choose to render the font bold or italic and adjust its size. You may have to experiment with the size to make your font fit your needs.

Using the File menu you are able to load additional TrueType fonts which are not installed in your system. It may come in handy when you've downloaded a .ttf file from the web, you can load it right there.

The Sample Text view

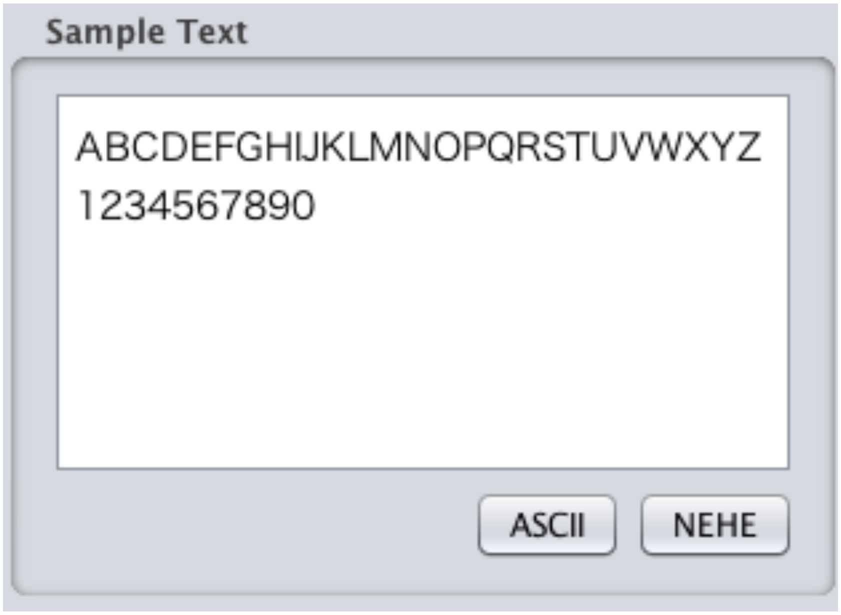


ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz
1234567890
"!`?'.,;:()[]{}<>|/@\^$-%+=#_&~*

This is where you can see and edit the characters used by the bitmap font you're creating. By clicking the ASCII and NEHE buttons you can load presets of characters. Normally you don't want the full ASCII character set, so NEHE is fine and the default setting. NEHE refers to the 95 printable

ASCII characters as shown in the Wikipedia page about ASCII.

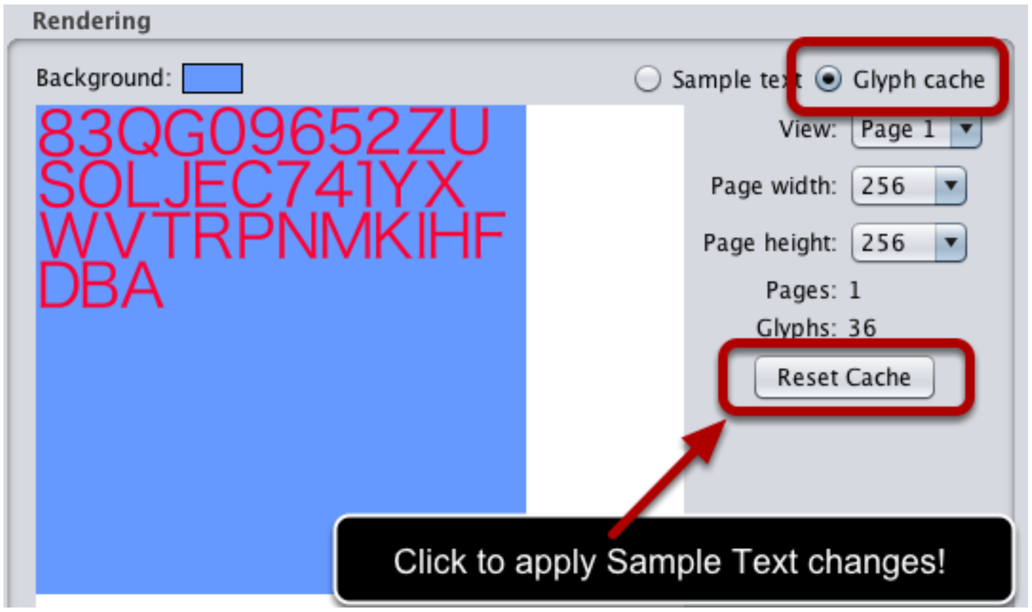
The Sample Text view: enter a character subset



ABCDEFGHIJKLMNOPQRSTUVWXYZ
1234567890

You can also enter a subset of characters into the Sample Text box. If you know you only need certain characters you can reduce the memory consumption of the bitmap font. In this case I only selected uppercase characters and digits.

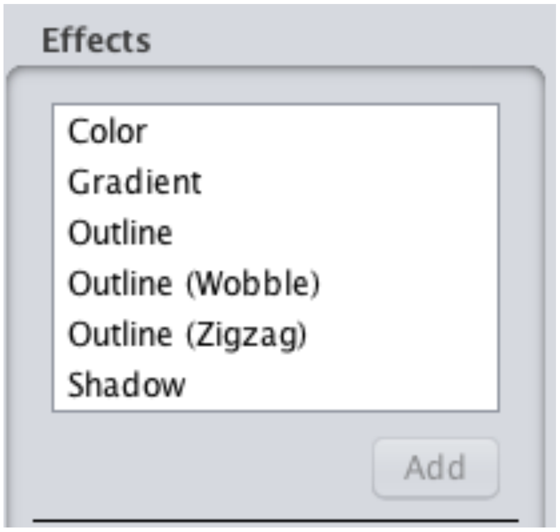
The Sample Text view: reset cache to use character subset



( ) Sample text [✓ Glyph cache]
View: Page 1
Page width: 256
Page height: 256
Pages: 1
Glyphs: 36

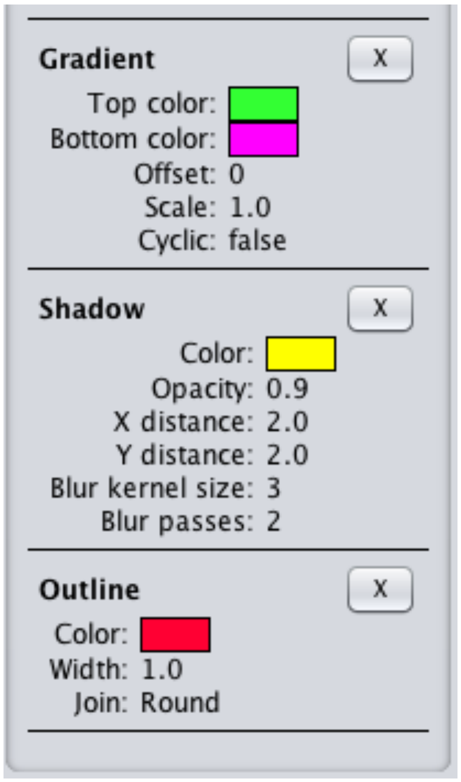
Note: if you change the characters in the Sample Text box you'll have to change the Rendering display to Glyph cache and click the Reset Cache button. Otherwise the bitmap font will still have the previous Sample Text characters in the font.

The Effects view



You can choose to apply several effects to the bitmap font. Note that you can only add each effect once, and Color and Gradient effects are mutually exclusive. You can add both but only the one added later will have any effect.

The Effects view: active effects list



Gradient
Top color: 🟩
Bottom color: 🟪
Offset: 0
Scale: 1.0
Cyclic: false

Shadow
Color: 🟨
Opacity: 0.9
X distance: 2.0
Y distance: 2.0
Blur kernel size: 3
Blur passes: 2

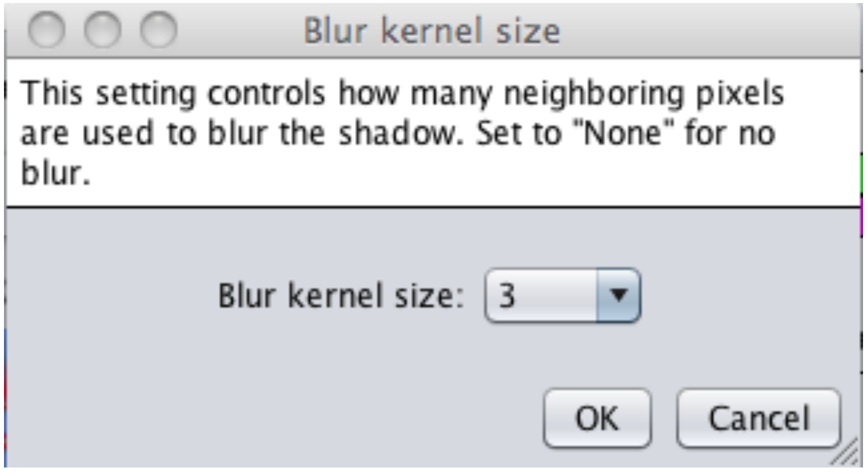
Outline
Color: 🟥
Width: 1.0
Join: Round

Below the available Effects are the effects currently in use. Effects are run in the order they appear in this list. It makes a difference whether you have Color, then Shadow or Shadow first, then Color. Unfortunately there is no way to re-order effects other than removing and re-adding them in a new order.

You can change each effect's parameters by clicking on the numbers or colors. Some dialogs will give you useful hints, such as to increase padding appropriately.

Click the X button to remove an effect from the active effects list.

The Effects view: changing effect properties

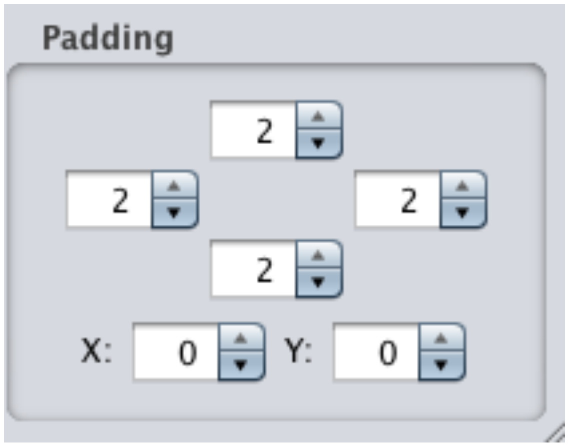


This setting controls how many neighboring pixels are used to blur the shadow. Set to "None" for no blur.

Blur kernel size: 3

This is one of the dialogs which appears when editing effect parameters. Read the help text, they contain important tips.

The Padding view



2
2 2
2
X: 0 Y: 0

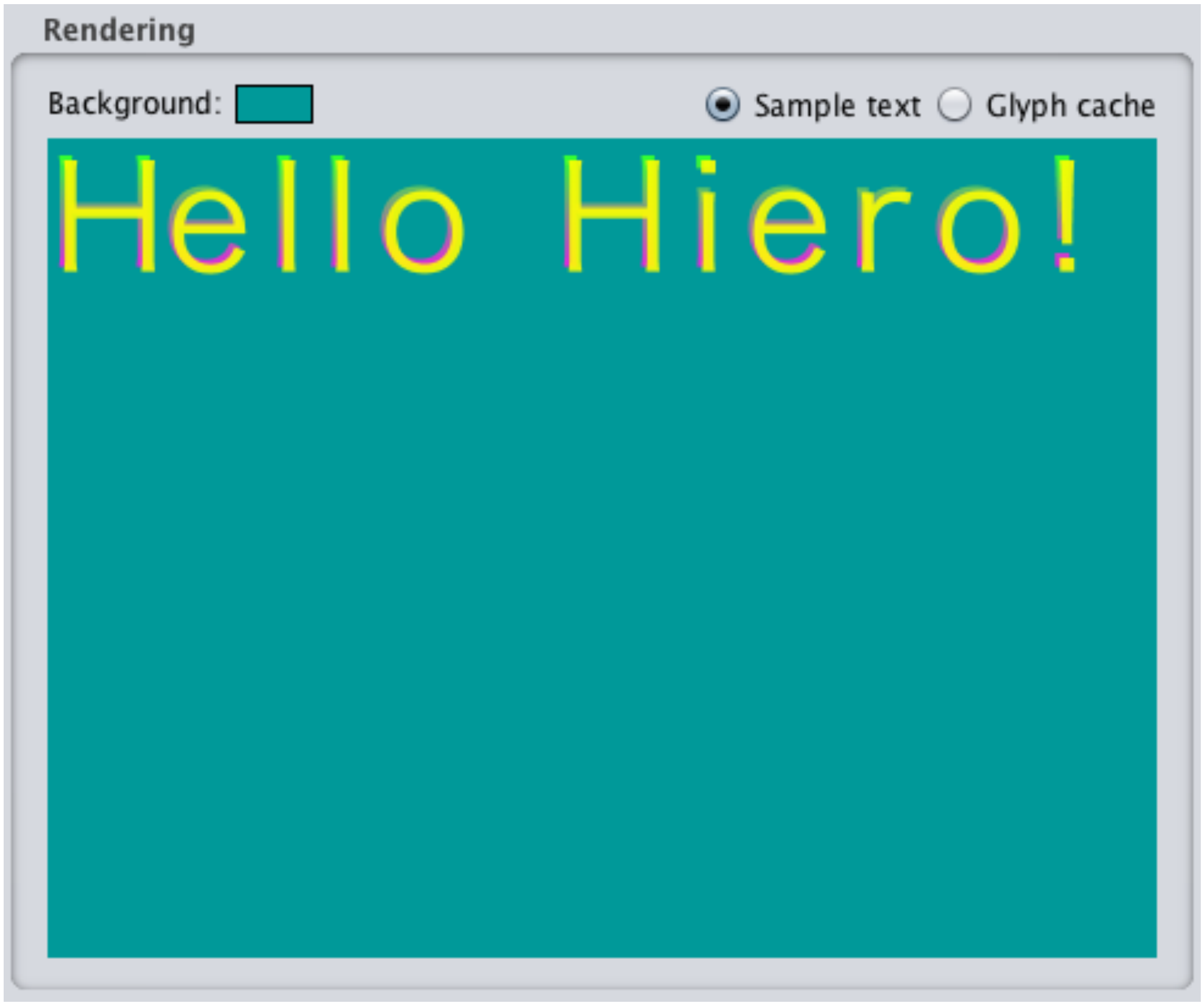
Allows you to change the spacing (or padding) between individual characters in the image file. Change only when using certain effects, or when artifacts appear (characters overlap). Note that there is a bug causing bitmap fonts to be mostly artifacts, please check the troubleshooting section if your characters are a total mess.

Some effects add shadows or blur and make each character bigger than it is. To avoid these shadows or other artifacts to bleed into other characters you may have to adjust padding. This is a trial and error process. If you see artifacts in your bitmap font in your game, adjust the padding until all the artifacts disappear. Note that increasing padding will also increase the space each character needs.

The upper 4 number boxes allow you to adjust the space each character takes up. I recommend to adjust the two vertical and horizontal pairs equally, like in the image above, instead of using 0-4.

The lower 2 boxes labelled X and Y seem to have no effect.

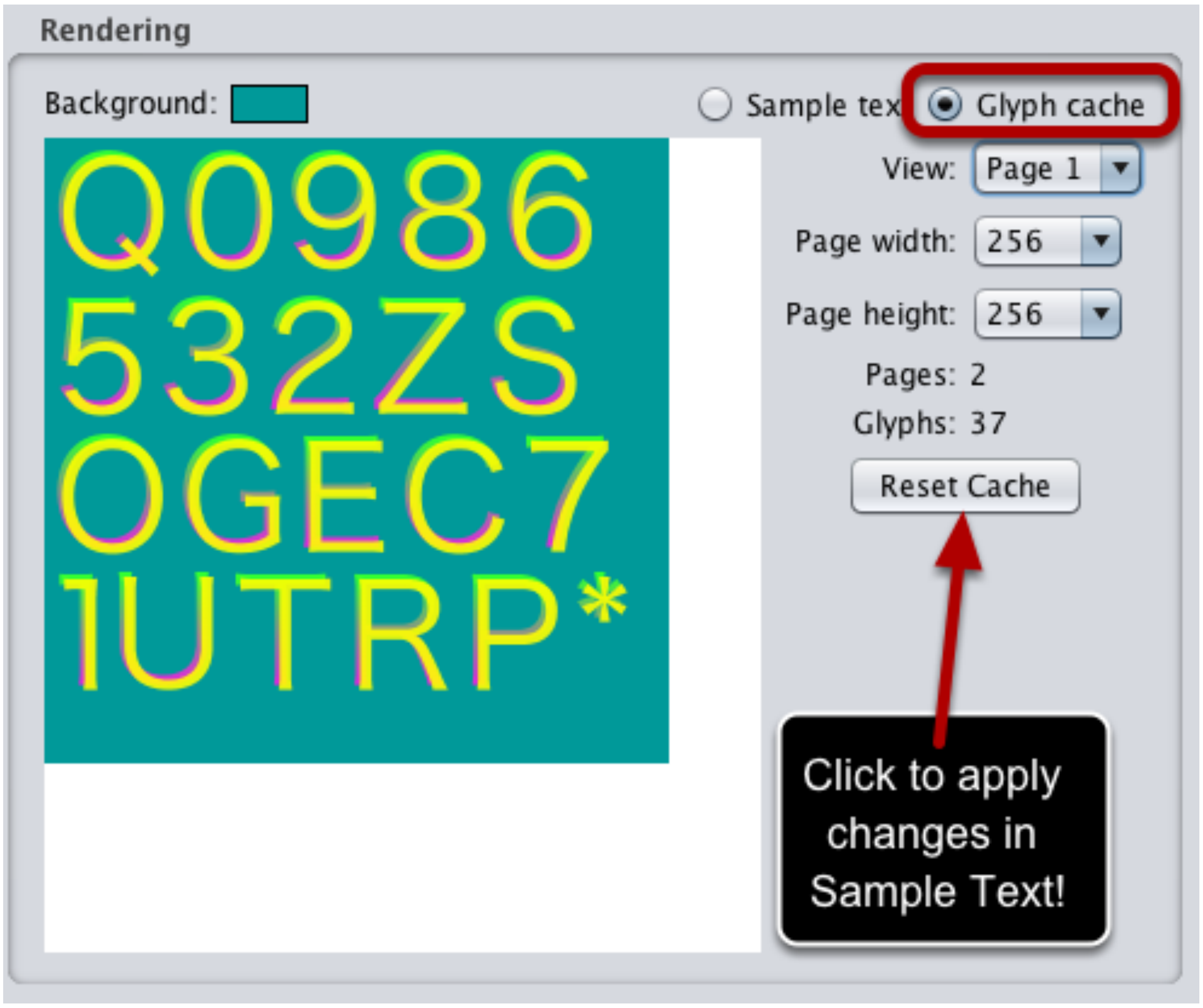
The Rendering view



This is where Hiero displays the Bitmap font's characters with all effects applied. You can also change the background color. This is purely cosmetic, so that you can see any font properly.

The Sample Text view will display whatever text you enter into the Sample Text window. You can try out specific strings you are going to display in your game to see what it might look like.

The Rendering view: Glyph cache


Rendering

Background:

Sample tex [✓ Glyph cache]

View: Page 1

Page width: 256

Page height: 256

Pages: 2

Glyphs: 37

By switching to the Glyph Cache view you can actually see which characters are used in your bitmap. Bitmap fonts use one or more Texture Atlas textures to store the characters (glyphs).

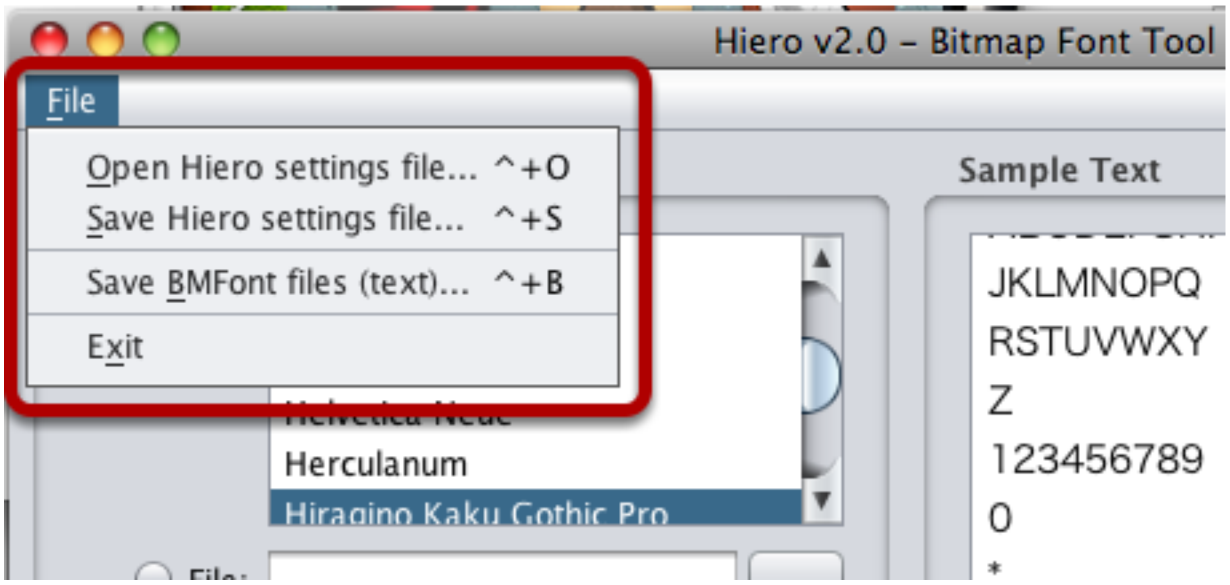
You can adjust the Page width and height settings to optimize the memory used by your bitmap font. Ideally you want all characters to fit just into the smallest possible Texture Atlas. Usually it's most effective to try to get all characters into just one Texture Atlas. In Hiero a Texture Atlas texture is called "Page". In the above image the number of Pages is 2. It may be possible to squeeze all characters into a 512x256 texture. If you choose a smaller font size try reducing the Page width and height one by one until the page size increases to 2. Then back up one step and use that image. Otherwise you might be wasting precious memory.

Remember that 1st and 2nd Generation iOS devices can only load textures which are no larger than 1024x1024 pixels. I hope you're never going to waste this much space with a single bitmap font anyway but I wanted to have mentioned it.

Finally, if you're in Glyph cache mode and change the Sample Text, then click Reset Cache to use the characters entered in Sample Text as the letters you want to use in your bitmap font.

Note that the characters may be randomly ordered. That is normal and nothing to be concerned about. Hiero is just trying to squeeze as many characters into the smallest space, and re-orders the letters in the process. They don't have to be in sequential order, the bitmap font will work perfectly fine this way.

File Menu


Hiero v2.0 – Bitmap Font Tool
File
Open Hiero settings file... ^+O
Save Hiero settings file... ^+S
Save BMFont files (text)... ^+B
Exit
Sample Text
Herculanum
Hiragino Kaku Gothic Pro
File:
JKLMNOPQ
RSTUVWXY
Z
123456789
0

Notice that Hiero's File menu is in an unusual location. It's like a Windows application.

The **Hiero settings file** stores and loads all of the settings and parameters you can adjust in Hiero. You should save this to be able to later tweak your bitmap font without having to recreate it. The file extension is .hiero

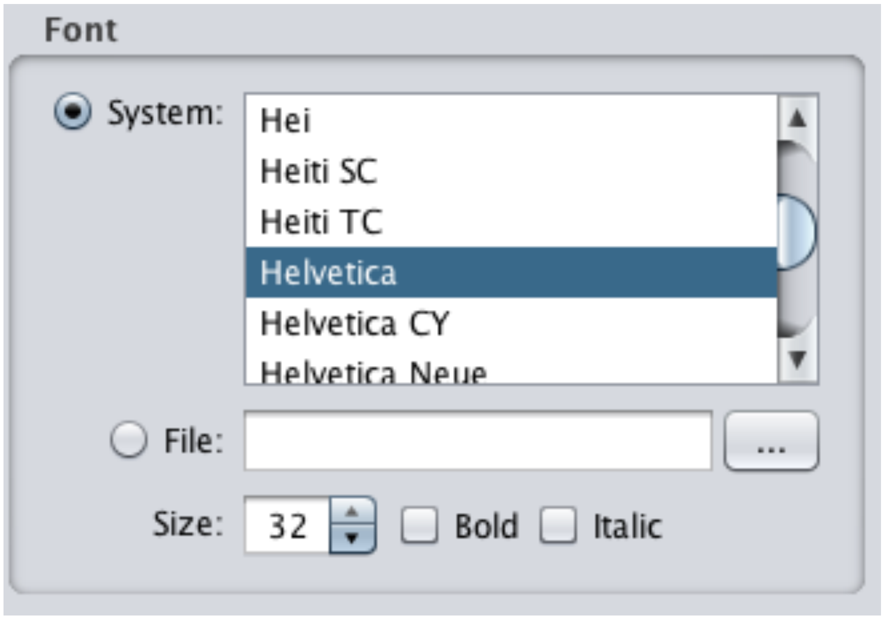
Save BMFont files (text) will save your current settings as a bitmap font that cocos2d's CCBitmapFontAtlas class can load. The file extension is .fnt and accompanied by sequentially numbered .png files.

Exit is the only way to close Hiero, next to clicking on the red X icon. Usual Mac OS X Quit commands are ignored by Hiero.

Creating a Bitmap Font Step-by-Step

I'll walk you through the steps necessary to create a bitmap font with Hiero.

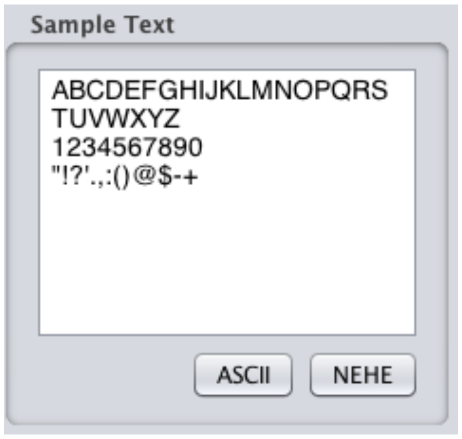
Pick a TrueType Font you like



Pick a font from the list, change its size and possibly make it bold or italic.

Note: if you suddenly don't see any characters in the Rendering view anymore, save your Hiero settings and restart Hiero. This problem occurs most often when cycling through a lot of fonts in this list.

Create a character set

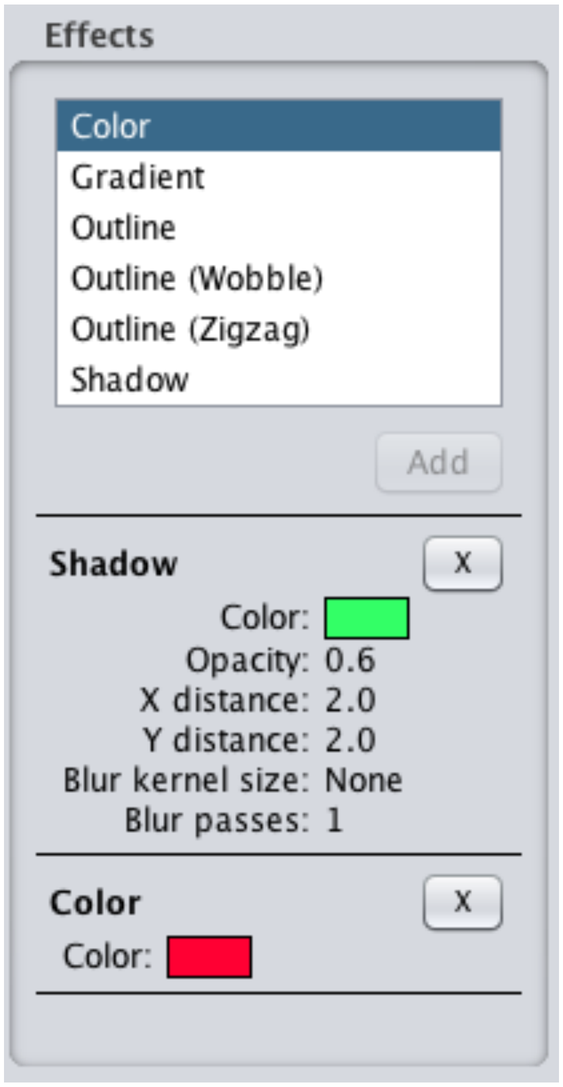


```
ABCDEFGHIJKLMNOPQRS
TUVWXYZ
1234567890
"!?'.,:()@$-+
```

In the Sample Text view, enter only the characters you really need to have in your bitmap font. In this case I chose to throw away all lowercase characters because I intend to use only uppercase characters in all of my texts. I also reduced the set of punctuation characters.

If in doubt, use the NEHE set by clicking on the NEHE button. This includes all of the most important characters.

Apply effects



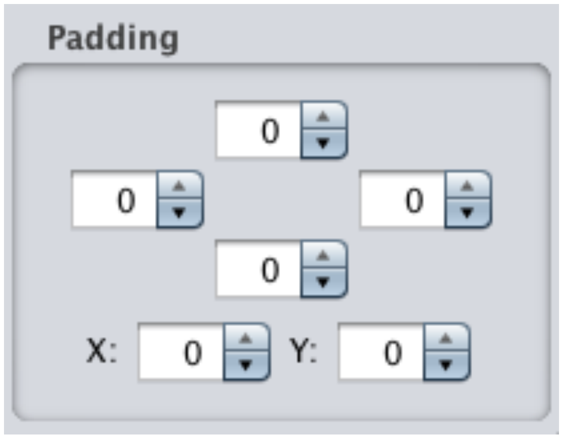
Shadow
Color: 🟩
Opacity: 0.6
X distance: 2.0
Y distance: 2.0
Blur kernel size: None
Blur passes: 1

Color
Color: 🟥

Apply any effects to the font as needed. Here I've added the Shadow and Color effects.

Note: I had to remove the existing Color effect, then add Shadow and Color again in order to actually add the Shadow to the background. Effects are drawn in order. Drawing the Shadow first, then Color (which will draw the font), creates the desired effect with a green shadow. The other way around may not look so good. Unfortunately you can't re-order effects without removing and re-adding them.

Adjust Padding (optional)

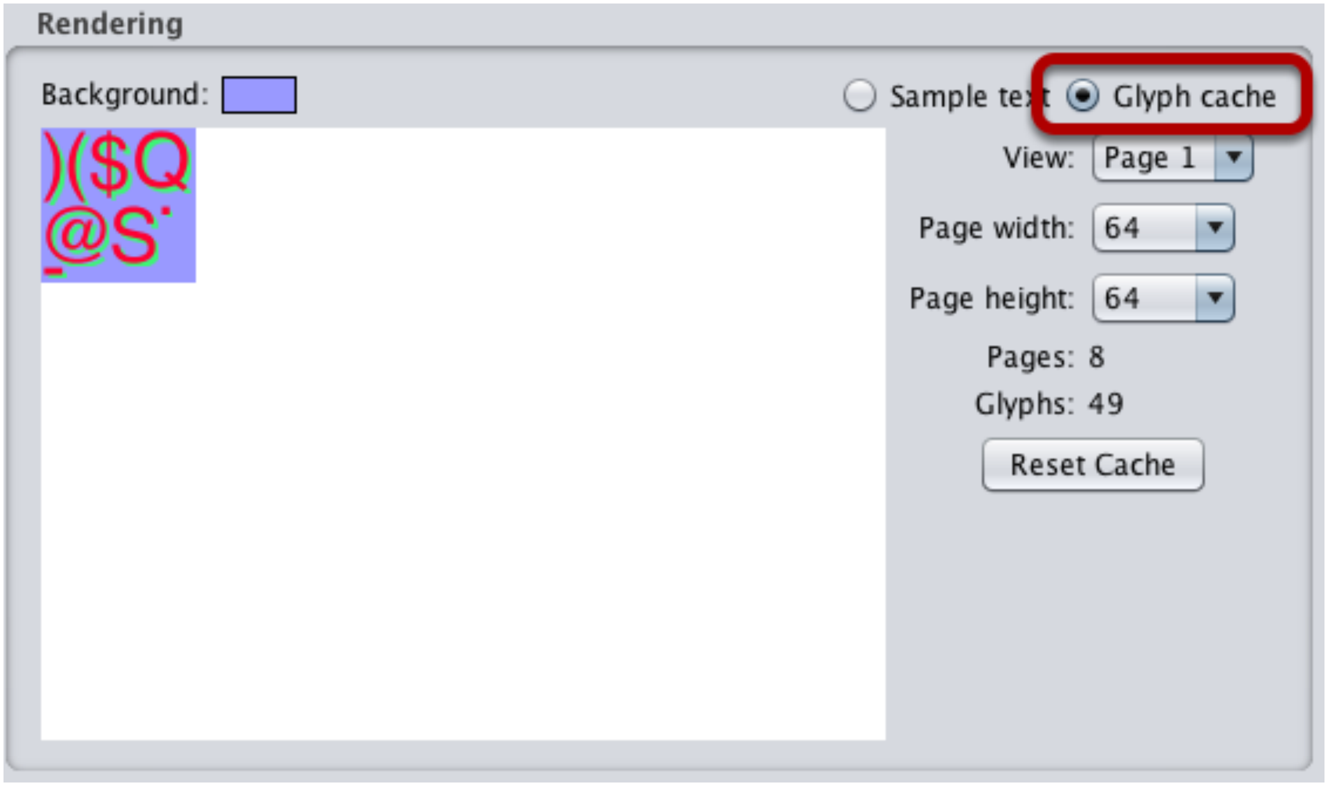


Padding: top 0, left 0, right 0, bottom 0, X: 0, Y: 0

If you added any effect which increases the font's size you may have to pad each character so that they have more space in between them. Otherwise you'll see artifacts in your bitmap font when rendering it ingame. Tweaking the padding is a trial and error process.

In this case I go without padding, in most cases this will be fine.

View the Glyph Cache

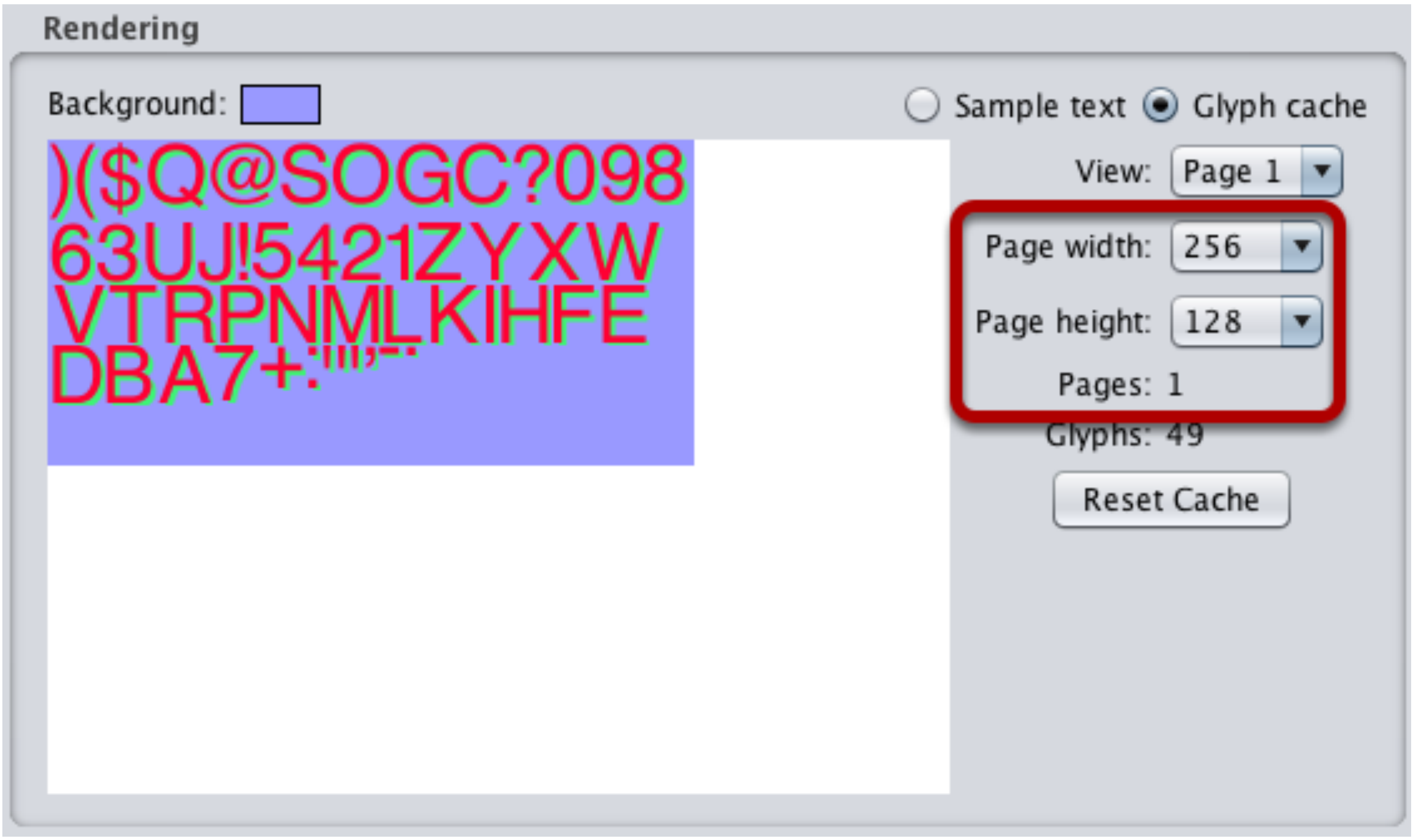


Sample text [✓ Glyph cache]
View: Page 1
Page width: 64
Page height: 64
Pages: 8
Glyphs: 49

Switch the Rendering view to display the Glyph cache. Then click **Reset Cache** since you changed the Sample Text characters. This makes sure that only the characters currently entered in the Sample Text field are used in the font.

Notice how Hiero wants to create 14 pages (textures)? This is pretty inefficient, ideally we want to get down to just one page (texture).

Adjust Page settings

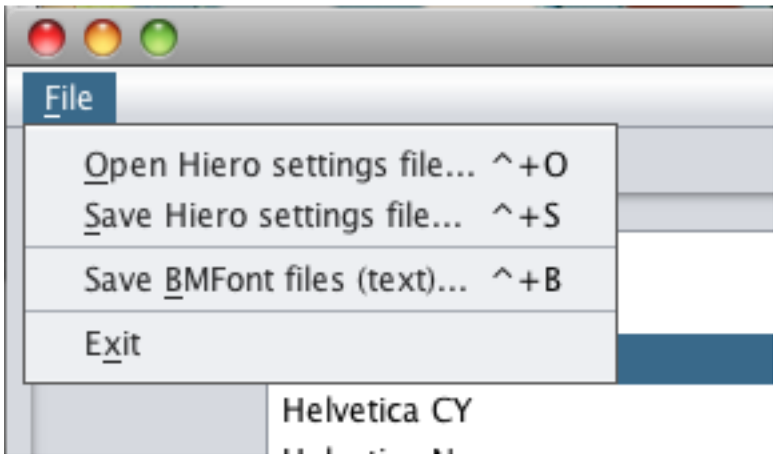


Glyph cache [✓]
View: Page 1
Page width: 256
Page height: 128
Pages: 1
Glyphs: 49

Change the Page width and height settings until you get down to just one Page (texture). This looks much better now.

Note that it doesn't really matter whether you use 256x128 or 512x64 in this case. The font fits both settings but the texture will use the same amount of memory in both cases, since the number of pixels equals 32,768 in both cases. Do the math. ;)

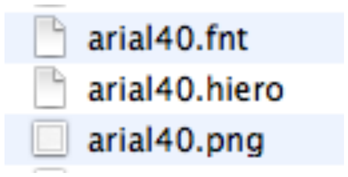
Save the file


File
Open Hiero settings file... ^+O
Save Hiero settings file... ^+S
Save BMFont files (text)... ^+B
Exit
Helvetica CY

From Hiero's File menu save both the Hiero settings file (extension: .hiero) and BMFont files (extension: .fnt with accompanying .png).

If you overwrite existing files make sure the file extensions are appended correctly. See the Troubleshooting section for more info.

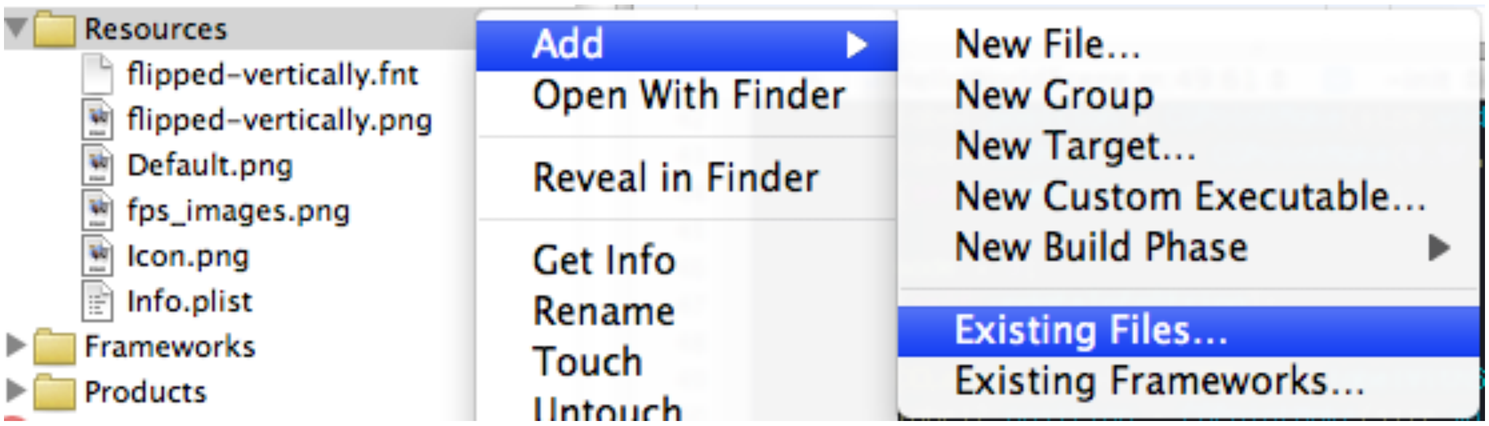
Files saved!



Success! Your bitmap font has been created. If you ever need to tweak it again, load the .hiero file to restore the same settings with which you've created the .fnt file. It's a good idea to name both files identical.

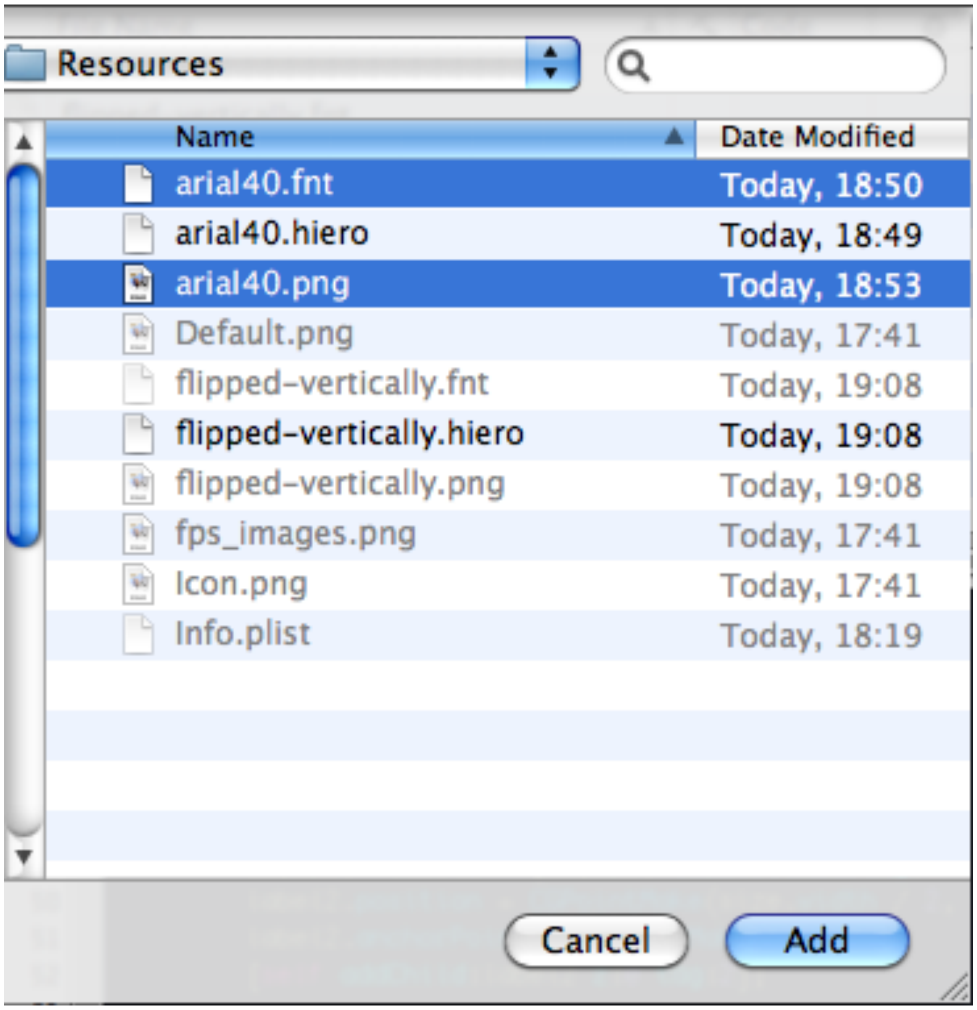
If you did not tweak the number of Pages you'll see two or more sequentially numbered .png files. You may want to go back and reduce the number of textures to one, and delete the superfluous .png files so that you don't accidentally add them to your Xcode project where they would only be wasting space.

Add the files to your Xcode project



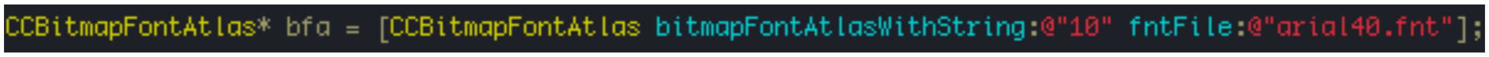
Right-Click the group where you want to add the bitmap font files and select Add -> Existing Files ...

Select only .fnt and .png files



Only add the .fnt file and all corresponding .png files. Do not add the .hiero file, you don't need that in Xcode!

Create a CCBitmapFontAtlas



```
CCBitmapFontAtlas* bfa = [CCBitmapFontAtlas bitmapFontAtlasWithString:@"10" fntFile:@"arial40.fnt"];
```

You can use a CCBitmapFontAtlas exactly like you would use a CCLabel class object. You only initialize it differently, as follows:

CCBitmapFontAtlas* bfa = [CCBitmapFontAtlas bitmapFontAtlasWithString:@"10" fntFile:@"arial40.fnt"];

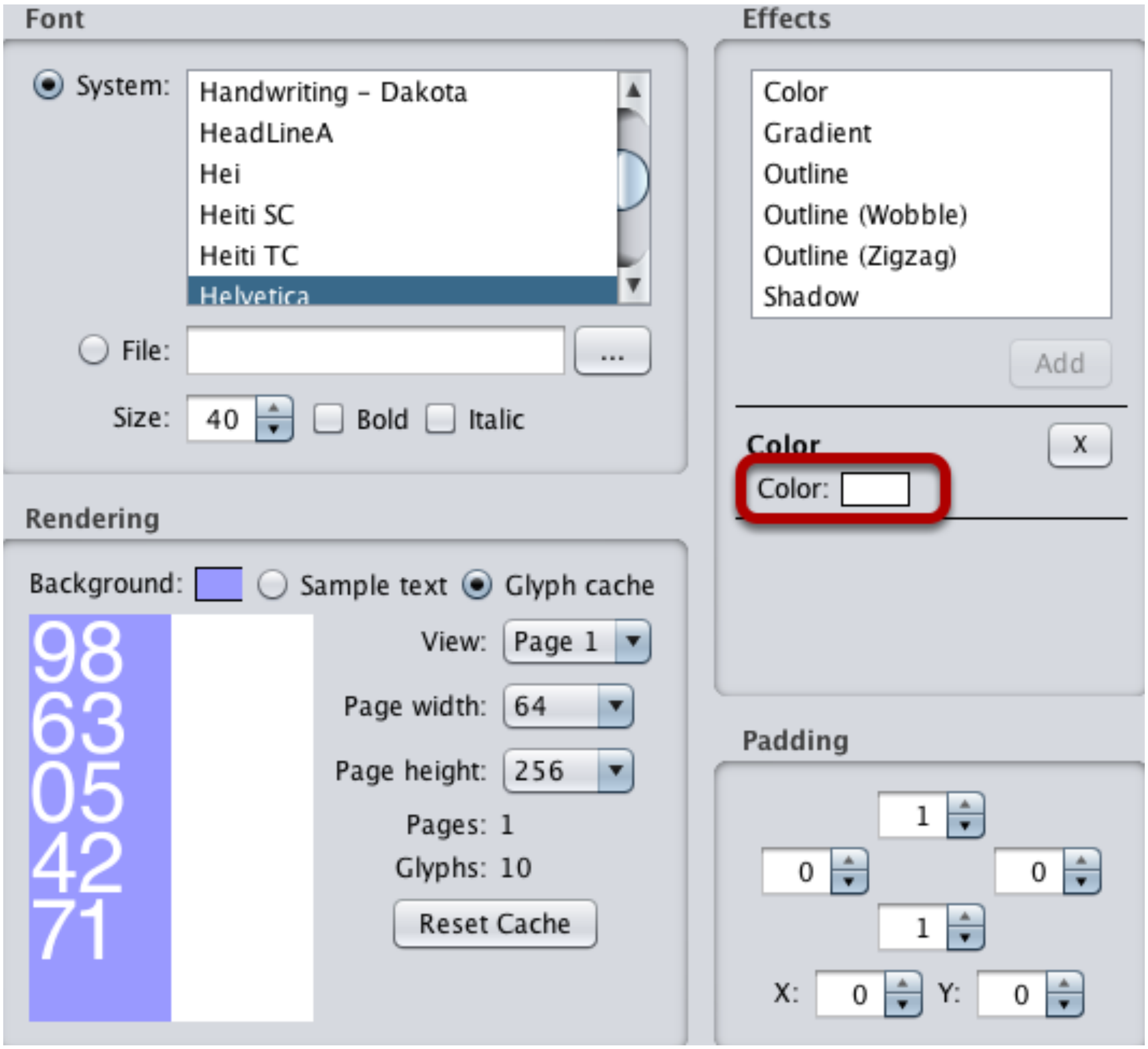
That's it!

You can now create your own bitmap fonts and use super-fast CCBitmapFontAtlas labels!

Tips for using CCBitmapFontAtlas

Here are a few handy tips which may not be immediately obvious when using a CCBitmapFontAtlas.

Colorizing (tinting) bitmap fonts



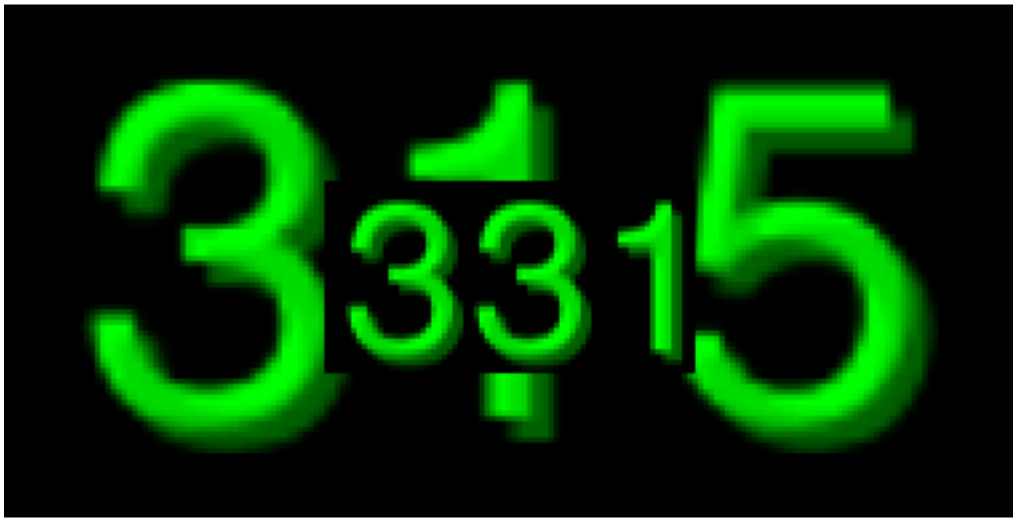
[✓ System:] Helvetica
File:
Size: 40 Bold Italic
Color
Color:
Background: Sample text [✓ Glyph cache]
View: Page 1
Page width: 64
Page height: 256
Pages: 1
Glyphs: 10
Padding: 1, 0, 0, 1
X: 0 Y: 0

You can use the label's color property to change the font color of a bitmap font, like so:

label.color = ccRED;

This works best with a font which has white characters and no other colors. See the screenshot above, simply choose white as color. Of course you can tint bitmap fonts of any color, even with gradients, but the best results are achieved when using a white font color.

Scaling a bitmap font label



You can also scale a CCBitmapFontAtlas label by using the scale property as follows:

label.scale = 2.25f;

The problem with scaling bitmap fonts is that they tend to get blurry when you scale them up, losing details. See the screenshot above, both fonts are magnified to better see the effect. The larger font in the background was scaled up 2.2 times whereas the smaller font is not scaled at all. Notice how the font in the original size looks sharper, crisper while the scaled up version looks a bit blurry.

If you scale a bitmap font label down it can create some aliasing effects and it depends on your font whether this is noticeable or not. For example, bitmap fonts using a gradient or outlines won't look too good when scaled down either. But scaling down will generally work better than scaling up. However, scaling in any direction is a costly operation (although moreso when scaling up) and should be avoided whenever possible.

Once again this decision boils down to conserving memory or conserving runtime performance. If you create several variations of your bitmap font in the various sizes that you'll need it, you'll waste memory since each size variation of the bitmap font uses its own texture atlas. On the other hand if you just use one bitmap font size and use the scale property, you'll lose image quality of the letters and performance due to the scaling operations.

Visit www.learn-cocos2d.com for more!

Get the latest version of this document and more!

This document was created by Steffen Itterheim exclusively for www.learn-cocos2d.com --> Visit the Website for cocos2d for iPhone Starter Kits, Game Components, Tutorials, Frequently Asked Questions, Free Source Code and more!

Follow this link to view or download the latest version of this document in the www.learn-cocos2d.com Knowledge Base.

Join my Newsletter

Join my Newsletter to stay informed on any updates and additions to the www.learn-cocos2d.com website!

Copyright Notice & Permissions

This document is (C) Copyright 2010 by Steffen Itterheim. All Rights Reserved.
This document was created exclusively for www.learn-cocos2d.com.

You may **NOT** upload and host this document on a public website, forum, or any other kind of publicly accessible file repository. Instead link to my Knowledge Base or the desired Subpage at: http://www.learn-cocos2d.com/knowledge-base

You may **NOT** reprint the contents of this document in whole or parts for print magazines, newspapers, etc. without my prior written permission.

You are encouraged to share this document among your peers given that access to the document remains free of charge (excluding user's fees for Internet access). You are allowed to quote excerpts of this document on your own website or in comments under the Fair Use doctrine.