Designer's guide to DPI

This guide is designed as a "get started" or introductory read for the starting to intermediate designer who wants to learn or get more knowledge about cross-DPI and cross-platform design from the very beginning.

As little math as possible and no un-parsable graph, just straight forward explanations ordered in short sections for you to understand and apply directly to your design process.

Written by Sebastien Gabriel.
Feel free to send any question of feedback to sgabriel.contact@gmail.com.
You can also find on me Twitter, Google+ or Facebook.

SHARE

TWITTER GOOGLE+ FACEBOOK

What is DPI and PPI

DPI or Dots Per Inch is a measure of spatial dot density initially used in print. It's the number of ink drops your printer can put in an inch. The more dots per inch, the sharper your image.

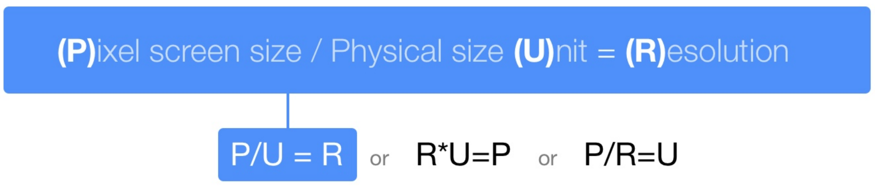
This concept is applies to computer screens under the name PPI for Pixels Per Inch. Same principle: It counts the number of pixels your screen displays per inch. The name DPI is also used in screens.

For a concrete and relatable example, Windows computers had an initial PPI of 96. Mac used 72. These values were determined because the screens they were building at the time turned out to display 72 “dots” or pixels per inches in the configuration they created. These values date back in the 80’s, now Windows, mac, or any device manufacturer creates an huge range of screens sporting a wide variety of PPI.

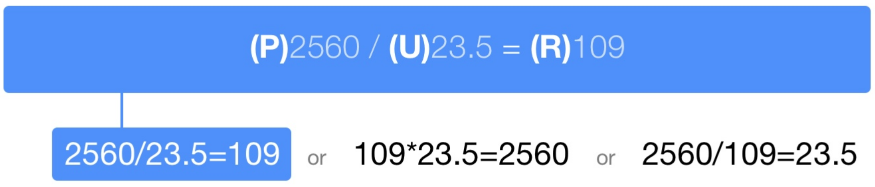
Resolution, pixel and physical size

Asking someone what the size of a pixel is is a good way to confuse him or her because it’s a trick question. A pixel has no size, no physical value or meaning outside of its mathematical representation. It is a part of a relationship between **the physical screen size**, expressed in inches, **the screen resolution**, expressed in pixel per inches and the **pixel screen size**, expressed in pixels. Laying it all out, it looks like this:

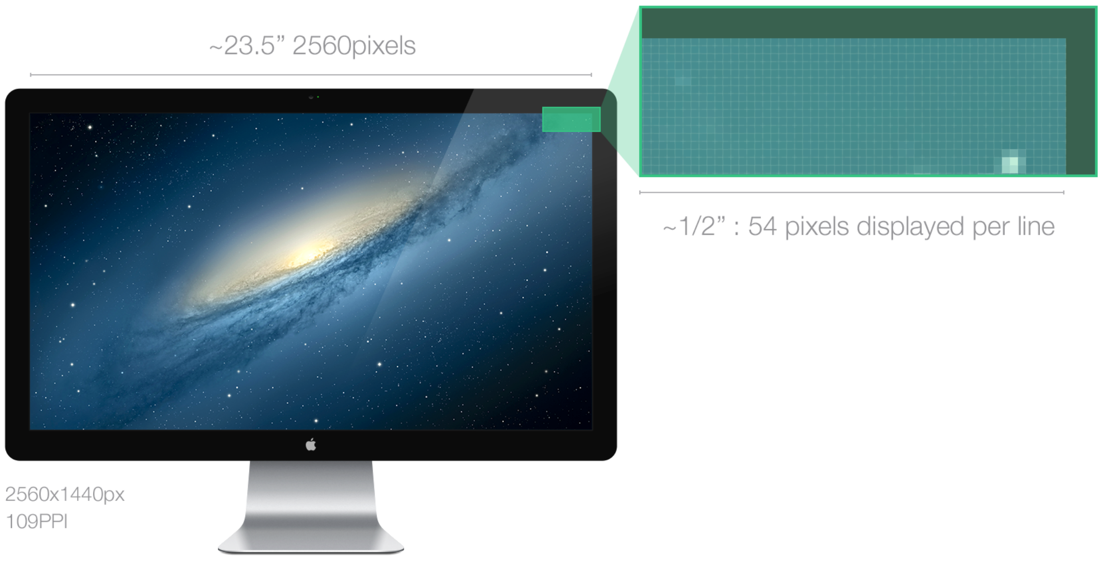
SHARE


Resolution density and screen size relationship

Here's an applied example: A Mac Cinema Display 27" has a PPI of 109, which means that it displays 109 pixels per inch of screen. The width with bevel is 25.7 inches. The width of the actual screen is approximately 23.5 inches so if we apply these values for the formula above, we can understand how they work together.


Density of an Apple 27" cinema display

SHARE



As you might have noticed in my explanations, "Resolution" stands for PPI, in this case "109" but not "2560x1440", like you might commonly see everywhere on the web.

"2560x1440" is the pixel count, as referred in the first paragraph as "Pixel screen size". The Pixel screen size in itself doesn't give you any information about how sharp the screen might be . You can very well have a 40" TV using 1920*1080px or a 5" phone. The only way of appreciating its resolution is by pairing the pixel count it with the device physical size, and that measurement is the PPI.

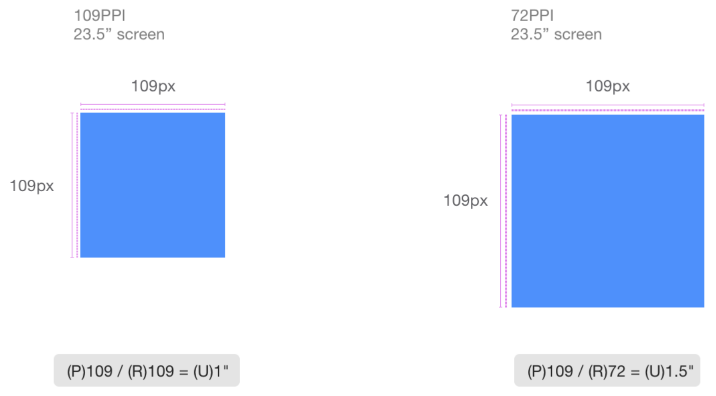
Takeaway:
A pixel in itself has no size or physical representation, it can only carry value through its relationship with the screen physical size, creating the resolution, or PPI. Understand this and screen density will have no secret for you.

Impact on your design

SHARE

width of 23.5". Because its size matches exactly the number of pixel per inch the monitor displays, the square will have a width of 1" and a height or 1".

Now take the same 23.5" screen with with a resolution 33% smaller, 72PPI. In this case, the same blue scare will be 33% bigger because displayed on the same physical size.



Takeaway
Leaving color and resolution differences aside, keep in mind that everybody will see your design slightly differently. You should aim for the best compromise and hit the largest percentage of users. Do not assume that the user has a screen similar and/or as good as yours.

Screen pixel size (and "Best" resolution)

Screen resolution can have a huge impact on how the user perceives your design.

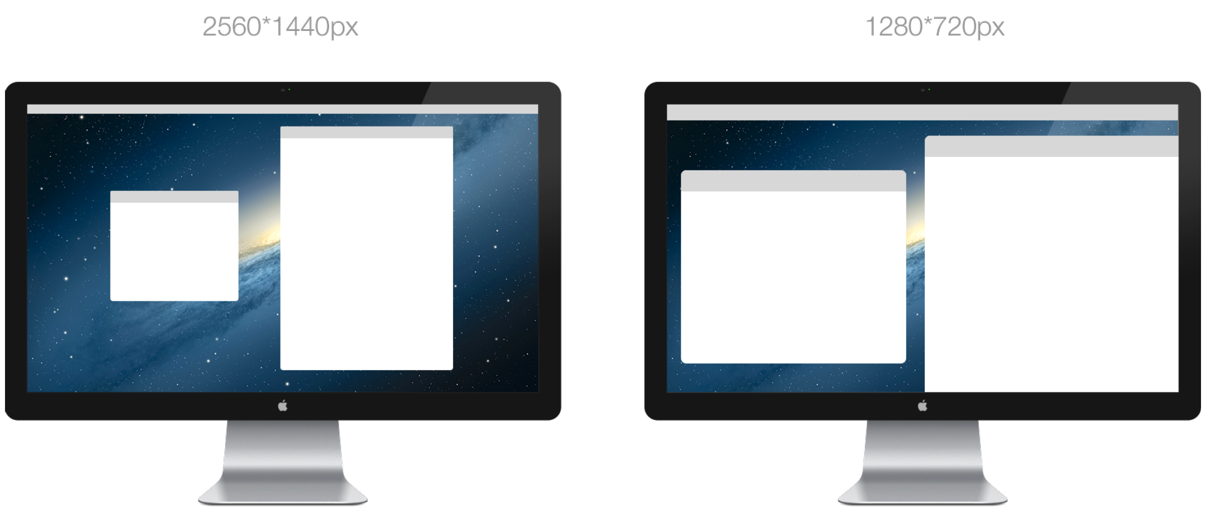
The "screen pixel size", also commonly referred as "resolution" (PPI being referred as pixel density) everywhere is the number of pixels displayed horizontally and vertically on a screen. For example 2560*1440px for the cinema display 27in consitutes its "screen

SHARE

number.

Today's LCD monitors have a pre-defined default or "best" resolution. **The "best" resolution is the state where the number of pixel physically display matches the number of pixel drawn by the software one to one**. It was slightly different with old CRT monitors, but since those can be considered dead, let's not enter into the details (and not reveal my only partial understanding of the good old TV).

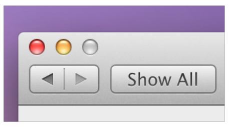
Let's take our 27" Cinema Display that can display 109 PPI at a best pixel screen size of 2560*1440px. If you reduce it, the elements will appear bigger. After all you'll have 23.5 horizontal inches to fill with virtually less pixels.



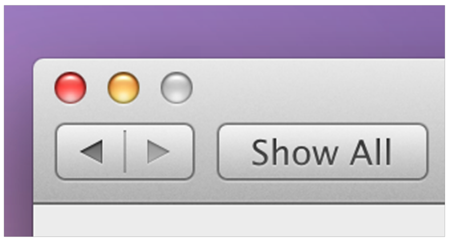
I said virtually because in this case it will be. The screen has a best pixel screen size of 2560*1440px. If it goes down, the pixels are still here, displayed in 109PPI. What your OS will do to fill the "gap" is stretch everything. Your GPU/CPU will take every pixels and calculate them with the new ratio.

SHARE

What will the result be ? Well, blur. While half the ratio will look fairly ok because of the simple divider, if you ask for 1/3 or 3/4 of the ratio then you'll end up with decimal numbers, and you CANNOT divide a pixel. See example below.



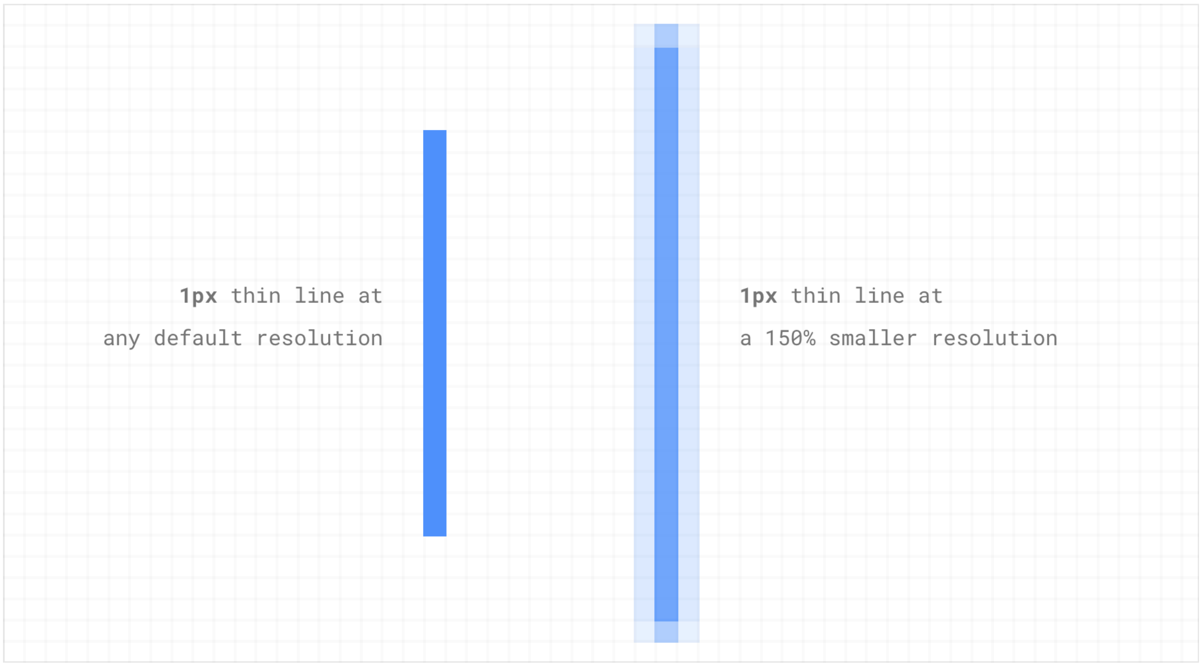
OSX Window rendering
in native resolution
(1440*900px retina)



OSX Window rendering
in simulated smaller resolution
(1024*640px retina)

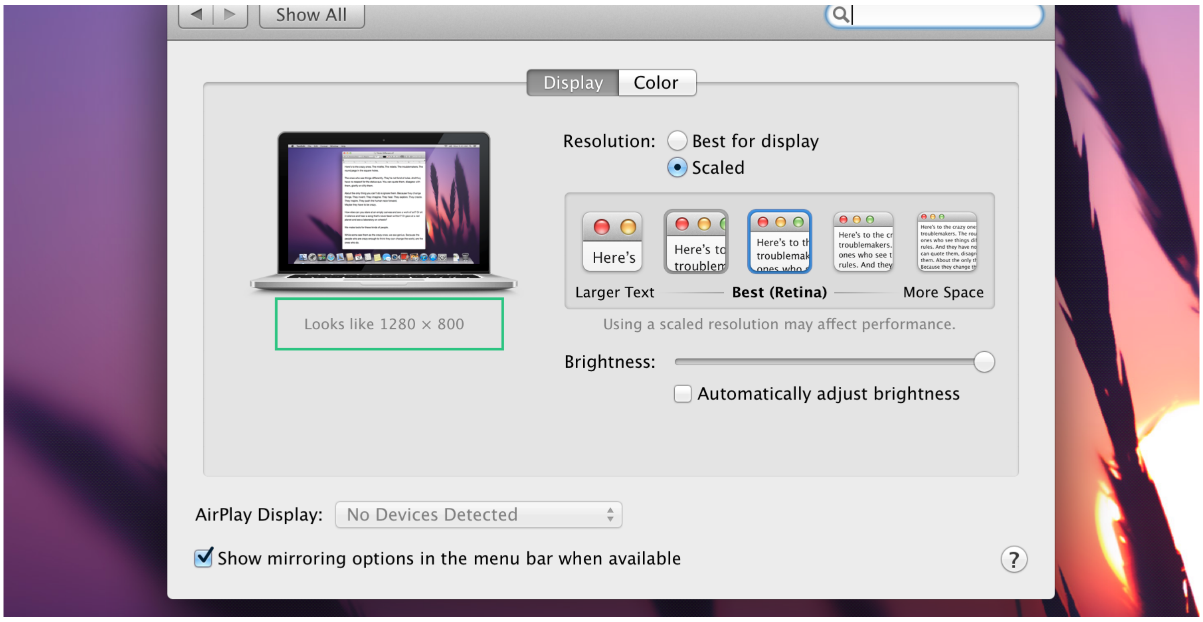
Consider the other example below. Take a 1px line on a best resolution screen. Now apply a pixel count 150% smaller. To fill the screen the CPU will have to generate the visuals at 150%, multiplying everything by 1.5. 1*1.5=1.5, but you can't have half pixels. What will happen is that it will fill the surrounding pixels by a fraction of the color, creating blur.

SHARE



That's why if you have a Retina Macbook Pro and want to change your resolution, it will display the window below, letting you know that (in the screenshot below) this resolution will "look like" 1280*800px. It uses the user's resolution experience to express a size ratio.

SHARE



Resolution: Best for display [✓ Scaled]

[✓ Best (Retina)]

Looks like 1280 × 800

Using a scaled resolution may affect performance.

☐ Automatically adjust brightness

AirPlay Display: No Devices Detected

☑ Show mirroring options in the menu bar when available

This is a highly subjective representation because it's using a pixel count as a unit of physical size, but it is not a lie, at least from their standpoint.

As you noticed, the word "resolution" is often employed to describe what I refer to here as "pixel screen size", PPI being sometimes referred as "Pixel density". While this is not wrong, it is unprecise. Make sure to understand the real meaning of "resolution" when you see or hear it. For example Apple uses the Phrasing: "2048-by-1536 resolution at 264 pixels per inch (ppi)" to describe a screen resolution. In this case they properly use it in context, along with the PPI.

Takeaway:
If you want to always see your design (or any design) pixel-perfect, never use a pixel count different from your native one. Yes you may be more comfortable with a smaller ratio but when it comes to pixels, you want to be as accurate as possible.

What is 4k

You may have heard the term 4K a lot lately (at least when I wrote this, beginning of

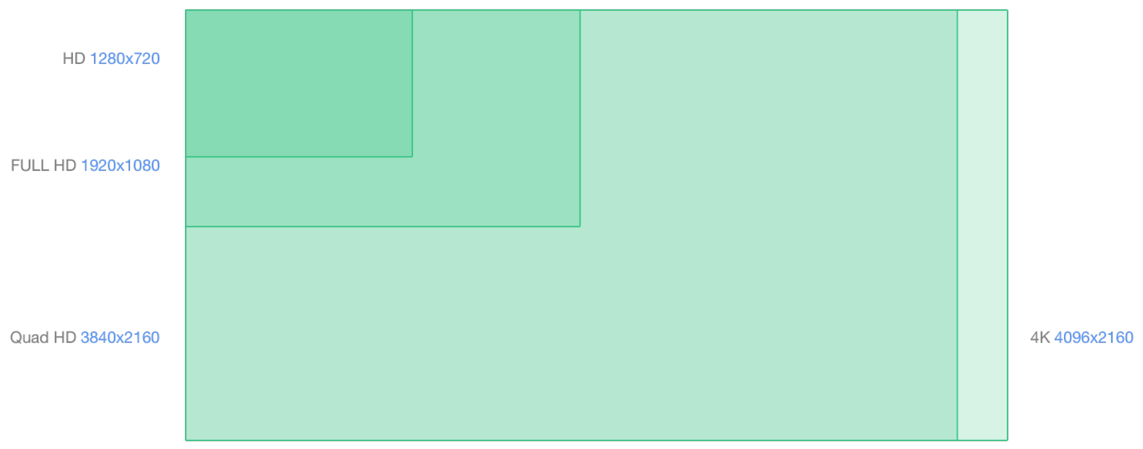
SHARE

Careful, this is an ultra-simplification. I'm only going to talk about the most common resolutions. There is different categories of HD.
The term HD is applicable to any pixel screen sizes starting from 1280x720px or 720p for 720 horizontal lines. Some may also call this resolution SD for standard definition

The term full HD applies to the 1920x1080px screens. Most TVs uses this resolution and more and more high end phones (Galaxy SIV, HTC one, Sony Xperia Z, Nexus5)

4K starts at 3840x2160 pixels. It was also call Quad HD and can be referred to as UHD for Ultra HD. Simply put, you can put 4 1080p in a 4K screen in term of number of pixels.

Another pixel count of 4K is 4096x2160. It's slightly larger and used for projectors and professional cameras.



What happens if I plug a 4K display to my computer

Current OS do not scale 4K, it means that if you plug a 4K display to a Chromebook or a macbook, it will use the highest DPI asset, in this case the 200% or @2x ones, and

SHARE

(2x), everything will appear twice as small.

Takeaway:
- 4k is 4 times Full HD.
- If current OS handles 4K but do not scale it, meaning no 4K specific assets yet.
- No phone or tablet uses 4K as of today.

Monitor Hertz

A little break from the PPI and pixels here for a quick note. You might have seen that close to the resolution settings for your screen there is the monitor Hz value. This has nothing to do with PPI, but just in case you're wondering, the monitor Hertz - or refresh rate - is the unit of speed at which your monitor will display a fixed image or frame, per second. A monitor with 60Hz will be able to display 60 frame per second. A monitor of 120Hz, 120fps etc...

In the context of an UI, monitor Hertz(Hz) will define how smooth and detailed your animation will look. Most screens are 60Hz. Note that the number of frames displayed per second is also dependent on the device's processing and graphical power. Adapting a 120Hz screen on an Atari 2600 would be quite useless.

To better understand, look at the example below. the T-rex goes to point A to point B at a fast and exactly equal pace on both a 60Hz and 120Hz screen. The 60fps screen is able to display 9 frames during the animation while the 120fps logically displays twice as more frame in the same fraction of time. The animation will appear much smoother on the 120Hz screen.

SHARE

T-rex walking fast at 60fps

T-rex walking fast at 120fps

Takeaway:
Some people might say that the human eye can't see above 60fps. This is wrong. Don't listen and walk away while laughing in the most obnoxious way possible.

What is a retina display

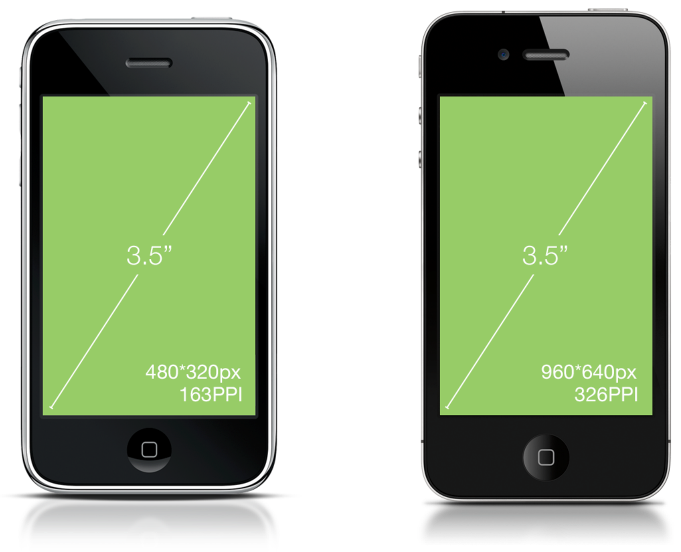
The naming of “Retina display” was introduced by Apple for the iPhone 4 release. It's called Retina because the PPI of the device was so high that the human's retina was supposedly not able to distinguish the pixels on the screens.

This statement is true for the range of devices' screen size it's used in, but as the screens are getting better and better, our eyes are now trained enough to perceive the pixels - especially for rounded UI elements.

Technically, What they did is display twice as many pixels in height and width in the exact same physical size.

The iPhone 3G/S was 3.5 inch diagonal with a pixel count of 480*320px which makes 163PPI.
The iPhone 4/S was 3.5 inch diagonal with a pixel of 960*640px which makes 326PPI.

SHARE



iPhone 3G/S
non Retina

iPhone 4/S
Retina

BOOM! Exactly twice. An easy multiplier. So instead of being smaller, the elements on the screen are twice as visually sharp because they have twice as many pixels and are exactly the same physical size. A single 1 normal pixel = 4 retina pixels, four times as many pixels.

Consider the example below for direct application in a complex design.

SHARE

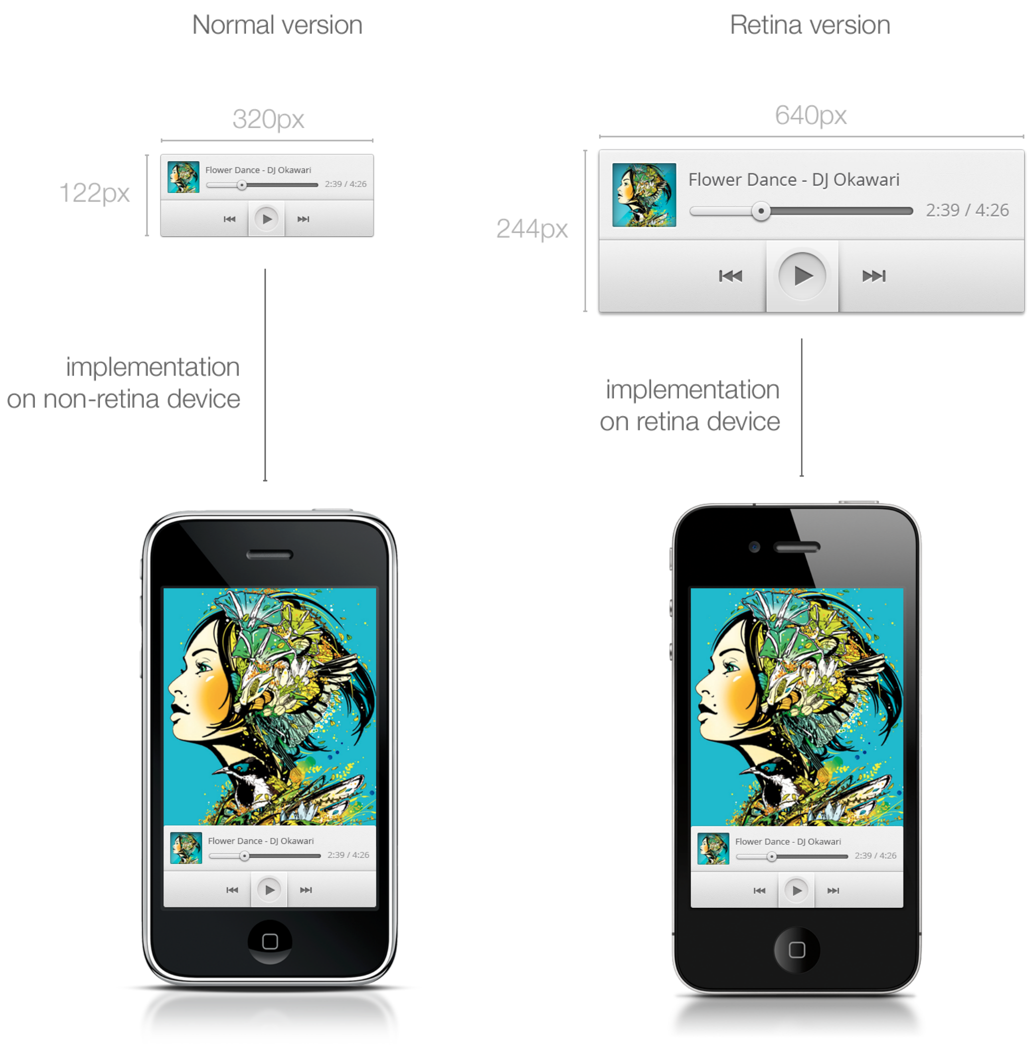

Normal version
Retina version
320px
122px
640px
244px
Flower Dance - DJ Okawari
2:39 / 4:26
implementation on non-retina device
implementation on retina device

Image note: It is hard to simulate different image quality from two devices on a third device, i.e. the one you are looking at right now. For the retina music player above, even while taking the exact same physical space, image quality will look twice as nice and sharp on the iPhone 4. If you want to check it out locally, I used one of my freebies for the purpose of this demonstration and you can download the source.

"Retina" display naming is owned by Apple so other companies will use "HI-DPI" or "Super power pixel maximum sp33d display" (I'm going to trademark the latter) or nothing at all. It'll up to you while reading device spec to figure out what's the DPI and screen size (how fun.)

SHARE

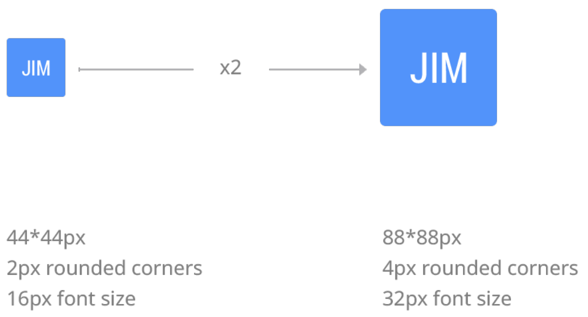
the differences between screen pixel size, PPI and physical size ratio because you'll only have to worry about 1 multiplier.

What is a multiplier

The multiplier is your mathematical savior when it comes to converting your design for all the different PPIs. When you know the multiplier, you don't have to care about the detailed specs of the device anymore.

Let's take our iPhone 3G and 4 example. You have 4 times as many pixels (2x width, 2x height) in the same physical size. Therefore your multiplier is 2. It means that to make your assets compatible with the 4G resolution, you just have to multiply your assets size by 2 and you'll be done.

Let's say you create a 44*44px button which is the recommended touch target by iOS (I'll talk about that later). Let's call him by a typical button name like "Jim."
To make Jim look good on the iPhone 4 you're going to have to create a twice-as-large version of him. That's what we're doing below.



Pretty simple. Now Jim has a Jim.png version for the normal PPI (iPhone 3) and a

SHARE

Now you may ask, "But wait! I'm pretty sure there are other multipliers beyond just two." There are, and that's where it becomes a nightmare. OK, maybe not a nightmare, but I'm pretty sure you'd prefer spending a day ironing your socks then handling a gazillion multipliers. Thankfully it is not as terrible as you think, we'll get to that later.

Let's talk units first because now you are going to need an unit other than pixel to spec your multi-DPI designs. That's where DP and PT comes in.

Takeaway:
The multiplier is what you need to know for every design you're working on. Multipliers are what hold together this world of chaos that is screen size and PPI and make it understandable to humans.

What are DP, PT and SP

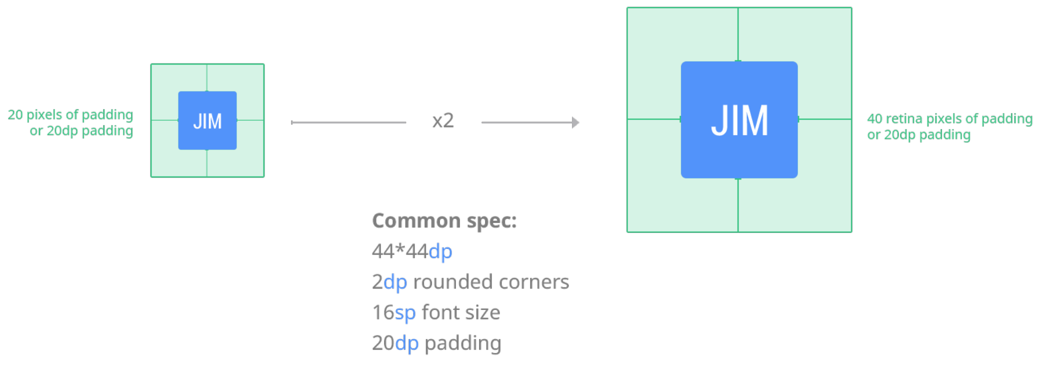
DP or PT is the unit of measure you can use to spec out your multi-device, multi-DPI app mock-ups.
DP or DiP stands for Device independent Pixel and PT for Point. PT is an Apple thing; DP is an Android thing, but they are essentially the same.

In short, it will define a size independently of the device multiplier. This helps a lot while discussing a spec between different actors like the designer and the engineer.

Let's take our previous button example, "Jim."
Jim has a 44px width on normal non-retina screens and a 88px width for retina screens. Let's get technical and add a padding around Jim of 20px because he likes having his space. The padding will then be 40px for retina. But it really doesn't make sense to count retina pixels when you're designing on a non-retina screen.

So what we're going to do is take a normal 100% non-retina ratio as the base of everything.

SHARE



In this case Jim will have a size of 44*44DP or PT and a padding of 20DP or PT. You can deliver your spec in any PPI, Jim will still be 44*44dp or pt.

Android and iOS will adapt this size to the screen and convert with the right multiplier. That's why I find it easier to always design using the default PPI of your screen.

SP is separated from DP and PT by it's usage but works the same way. SP stands for Scale-independent pixels and is used to define font sizes. The SP will be influenced by the user font settings on their Android devices. As a designer, defining the SP is like defining a DP for anything else. Base it on what's legible at 1x scale (16sp for example, is a great font size for readibility).

Takeaway:
Always use resolution/scale-independent values when specing. Always. The more varied the screen size/resolution are, the more it becomes essential.

The PPI setting

Now That you know what PPI, retina and a multiplier is, it is important to talk about something that have been asked to me quite a bit and is quite confusing: **"What happens if I change the PPI setting in my design tool ?"**

SHARE

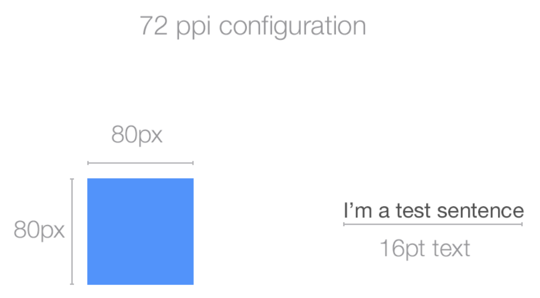
while:

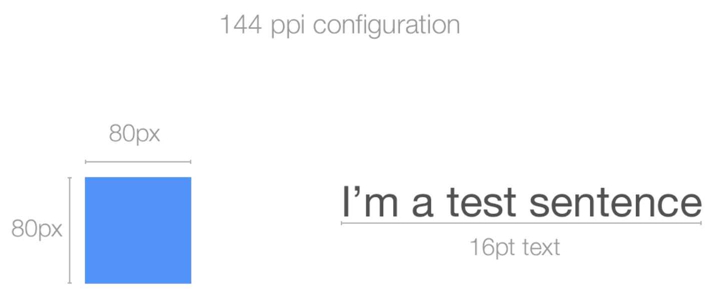
Anything non-print uses pixel sizes regardless of the initial PPI configuration.

PPI setting in software is a printing legacy. If you design only for the web, PPI won't have any influense on the size of your bitmap.

This is why we use multipliers and not direct PPI value. Your canvas and graphics will always be converted to pixel by the software using the corresponding multiplier.

Here's an example. You can try it yourself with a program that allows PPI configuration, like Photoshop. I drew a 80*80px square and a text with a size of 16pt in photoshop with a setting of 72PPI. The second one is the same thing with a 144PPI configuration.





As you can see, the text got quite bigger, twice bigger to be exact while the square

SHARE

rendering size of the text on the doubled PPI configuration. On the other hand, what has been defined using pixel, i.e, the blue square shape, remains the exact same size. A pixel is a pixel and will stay a pixel whatever the PPI configuration. What will make it different is the PPI of the screen that displays it.

What is important to remember is that when designing for digital, the PPI will only have an incidence on how you perceive your design and on your workflow and on pt sized graphics such as font. If you include in your workflow source files with various PPI configurations, the program will resize any transferred visual between the different files by the PPI ratio of the receiving file. It will be come a problem for you.

The solution ? Use a PPI (preferably in the 72-120 range for 1x design) and stick to it. I personnaly use 72PPI because it's the default setting in Photoshop and most of my co-workers do the same.

Takeaway:
- PPI setting doesn't have any influence when exporting for the web.
- PPI setting will only have influence on graphics generated based on PPI-independent measurement such as PT
- Pixel is the unit of measure for anything digital.
- Keep the multipliers in mind and what you are designing for, not the PPI.
- Use a realistic PPI setting when designing for digital, something that gives you a sense of what it's going to look like on the targeted device (72-120ppi for 1x web/desktop for example).
- Keep the same PPI setting thoughout your files.

Additional reading on that on this very interesting StackExchange post.

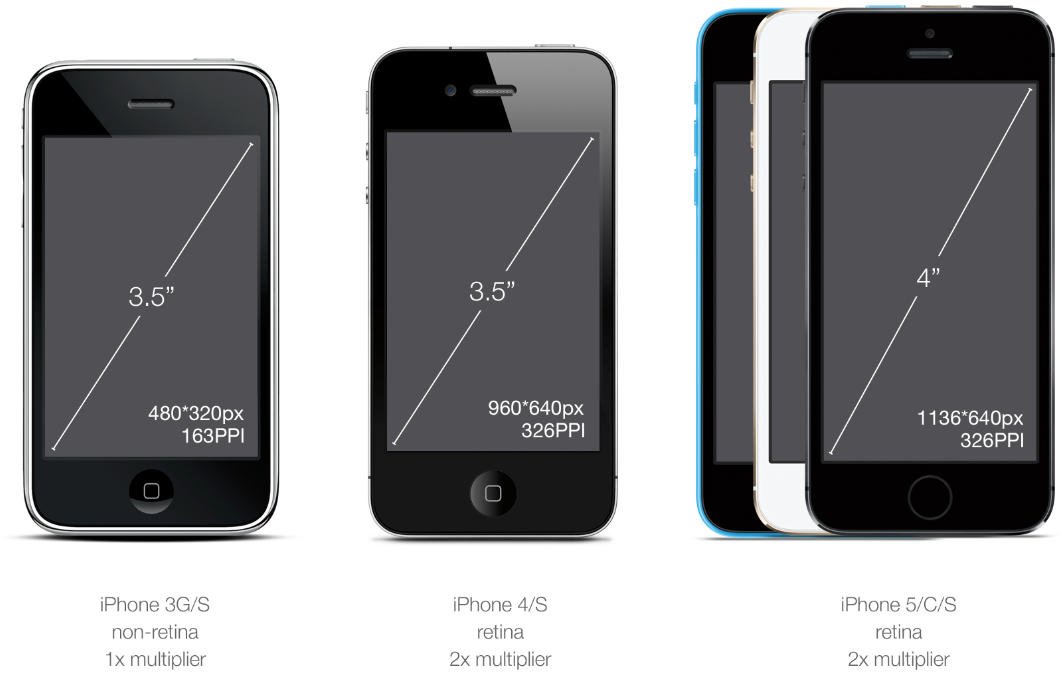
Handling PPI on iOS

Time to dive into platform-specific design.

SHARE

laptop/desktop screens.

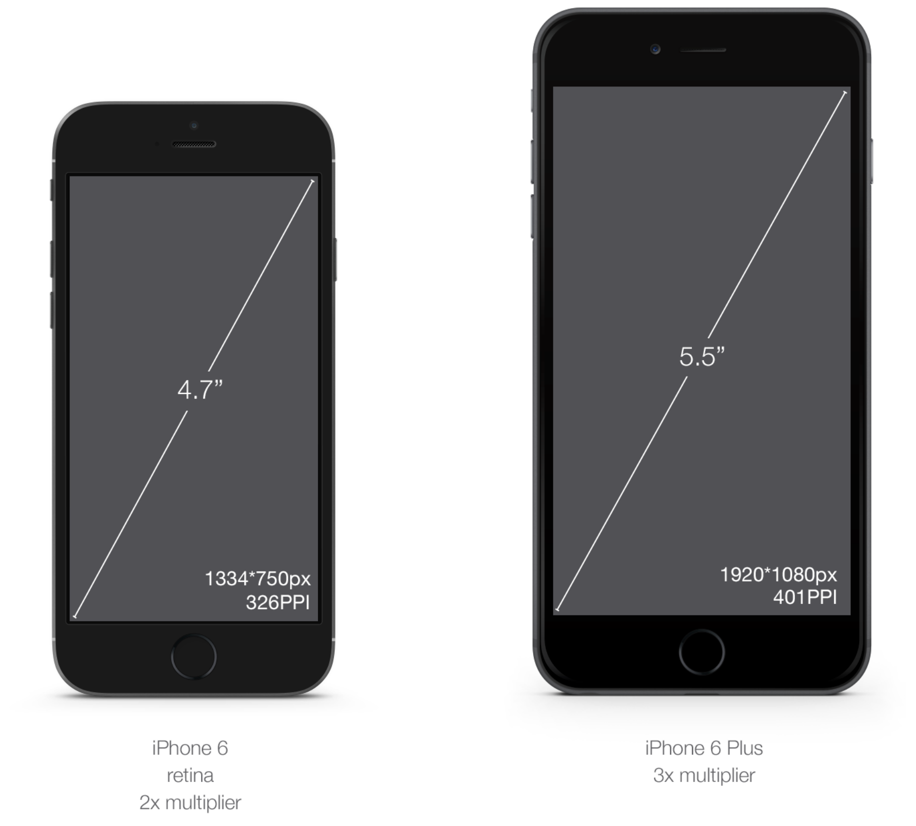
On mobile, they have the iPhone, of course, and the iPad.
In the phone category, you have the old 3GS (still supported on iOS6) and up. Only the iPhone 3GS is non-retina. iPhone 5 and up use a taller screen with the same DPI as the iPhone 4 and 4s. See cheatsheet below:



Announced at the september 2014 Apple Keynote, you now have 2 new categories of iPhone: The iPhone 6 and the iPhone 6 Plus.

The iPhone 6 is a bit bigger than the 5 (0.7" more) but carries the same PPI. The iPhone 6 Plus on the other hand introduces a whole new multiplier for iOS, @3x due to it's size, 5.5".

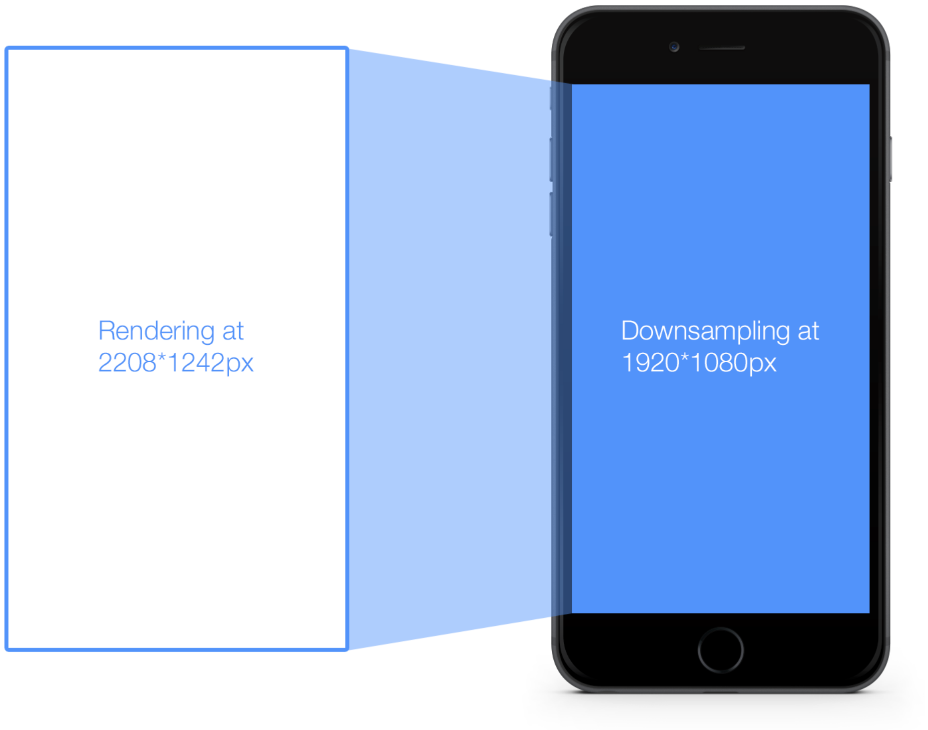
SHARE



There is something very special to know about how the iPhone 6 Plus handles its display compared to all the other iPhones:
It downsamples the visuals.

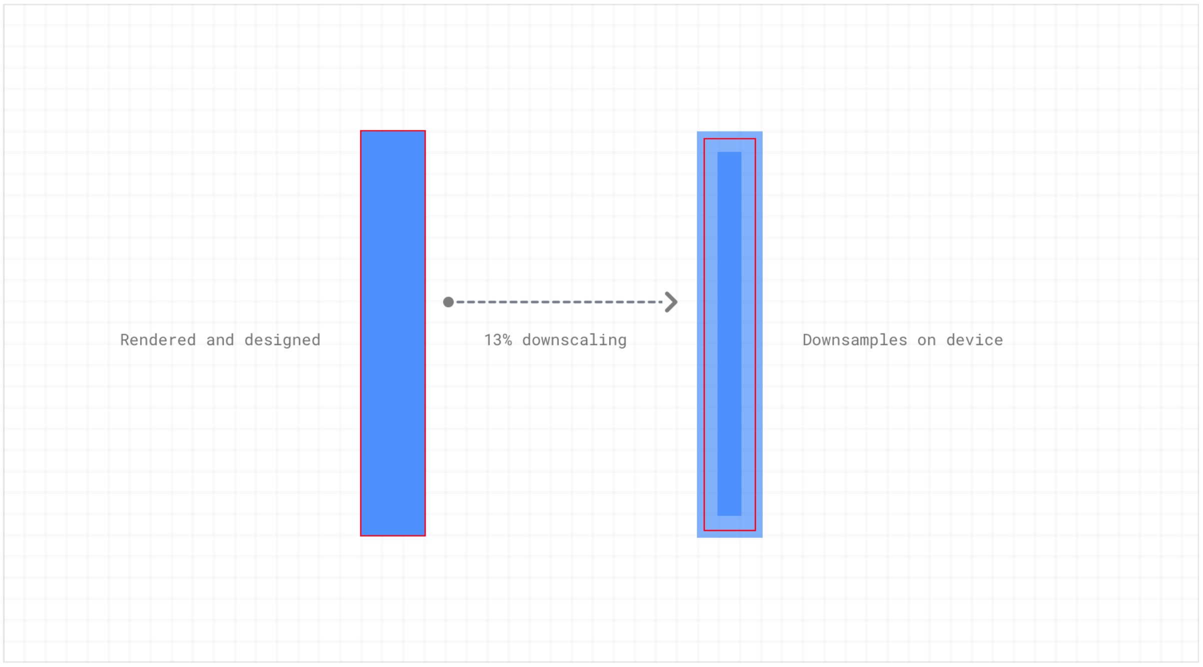
When you design for the iPhone 6 for example, you'll design on a canvas of 1334*750px and the phone will render 1334*750 physical pixel as well. In the case of the Iphone 6 Plus, the phone has a smaller pixel count than the rendered image so you will have to design for a pixel screen size of 2208*1242px and the Phone will downsample it to its pixel size of 1920*1080px. See illustration below:

SHARE



The device pixel count being 15% smaller than the rendered resolution, it will create a few glitches such as half-pixels making very fine detail a bit blurry. The resolution is so high though that it will be very subtle, unless you look really closely. So design on a 2208*1242px canvas and be aware of that for the really fine part of your design such as super thing separators. See a simulation of what is happening below:

SHARE

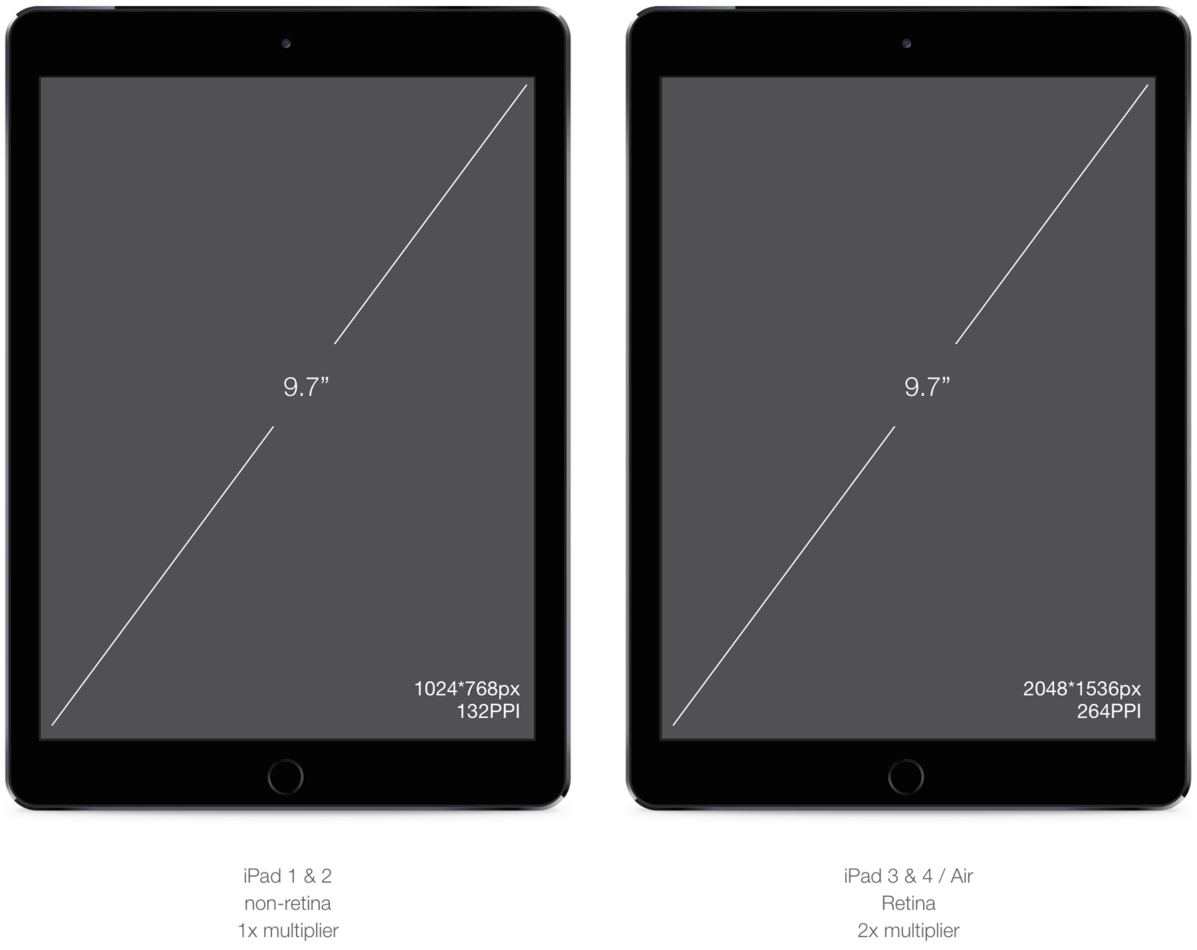


Thanks to Paintcode for coming up with an awesome explanation of this right after the keynote. Check out their dedicated page. This explanation is derived of their very nice schematics.

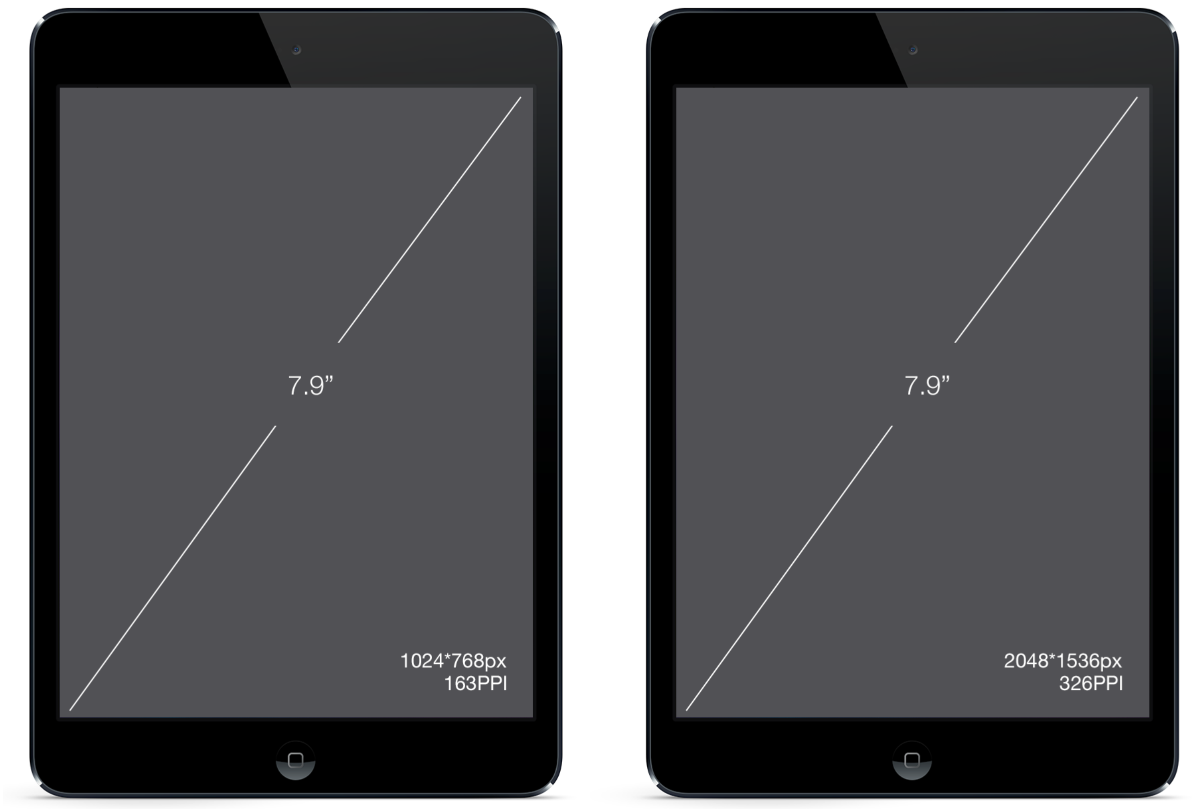
Then you have the iPod touch category. Consider them as iPhones when it comes to design. iPod 4th gen and down are using iOS6 and are non-retina. iPod 5th gen is using a retina(@2x) screen and is compatible with iOS7. Screen-wise iPod 5th gen is using an iPhone 5-sized screen.

Finally you have the iPads. With the exception of iPad 1st gen (out-of-date today,) they all use iOS7, and only the iPad2 and iPad mini 1st Gen uses a non-retina screen. If you're wondering what an iPad mini is from a design standpoint, it's a regular iPad (same PPI screen), but physically smaller. By this I mean that they took the same resolution and reduced it from 9.7in to 7.9in. Keeping the same ratio and therefore increasing the pixel density. Your visual assets will appear slightly smaller.

SHARE


9.7”
1024*768px
132PPI
9.7”
2048*1536px
264PPI
iPad 1 & 2
non-retina
1x multiplier
iPad 3 & 4 / Air
Retina
2x multiplier

SHARE



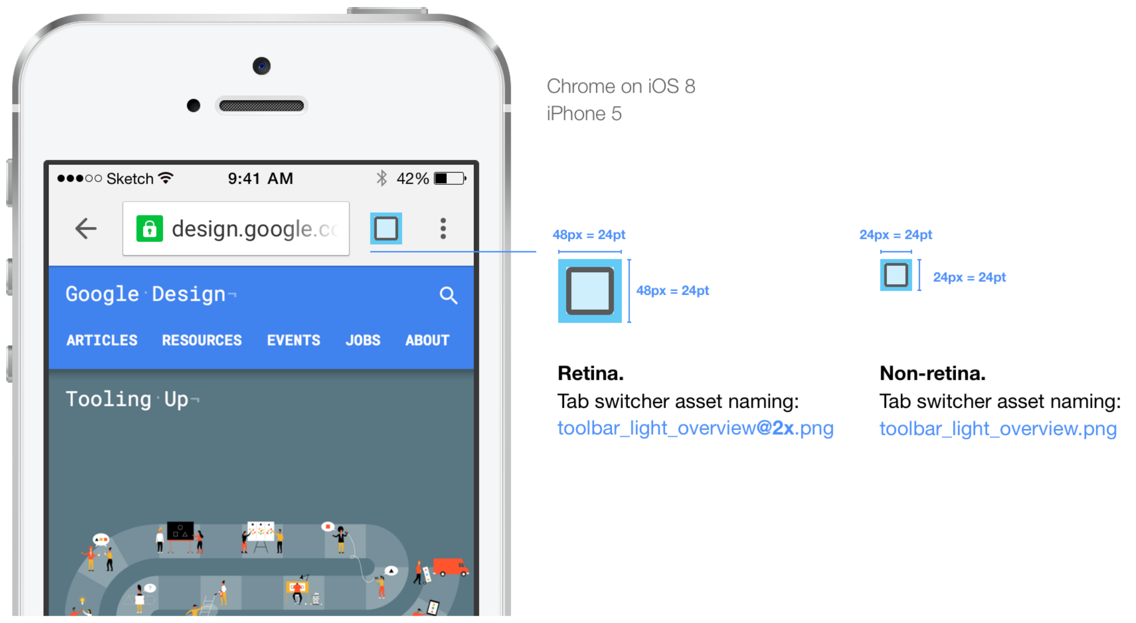
iPad Mini 1st Gen
non-retina
1x multiplier

iPad Mini 2nd Gen
Retina
2x multiplier

Regarding the desktop/laptop category, we're not going to be covering every single screen sizes Apple offers. As of today, Apple offers most of its screens in 1x multiplier (Macbook, Macbook Air, old Macbook Pros, desktop screens.) Retina exists in 13 and 15" for Macbook Pros only. The multiplier is 2x, exactly like iPads and iPhones. If designing for desktop is different than mobile, you'll produce assets the same way to cover the 2 different types of screens.

With only one multiplier, creating assets for iOS and OSX is pretty straightforward. I suggest to start designing for the base PPI (i.e 100%/1x) and multiply by 2 afterwards to proof your design on a @2x screen and generate the @2x assets. Once you are more comfortable with switching back and forth between 1x and 2x, you'll be able to design directly in @2x, scaling down your assets for lower resolution. This will be particularly helpful if you are designing on a retina screen (Macbook pro).

SHARE



As you can see we need to deliver two images per asset each time. Non-retina images are called name.png. For retina images we append @2x to that to have name@2x.png. This is an iOS convention and it should be followed.

If you create an image that is only going to be used on iPads, we use ~ipad after the .@2x. It is only a Chrome convention. Repeat this process for every asset you need. Never give only one version of an asset; cover every DPI.

Takeaway, iOS ruleset:
- @2x asset must be de double of 1x asset, always.
- Append @2x for retina assets and @3x for the 300% ones (iPhone 6Plus)
- Always create 100% and 200% images.
- Always have the same name for 1x and 2x assets.
- Start designing in 100%, then multiply.
- Deliver .png images.
- Create specs in pt not px.

SHARE

The Android platform has a wider range of devices than iOS. The reason is that any OEM can build a device and with few restrictions in term of scale and put their own version of Android on it. As a result, you end up with a virtually unlimited variety of screen sizes and DPI, from phones as big as tablets and tablets almost as small as phones. For this reason, your design will always have to adapt.

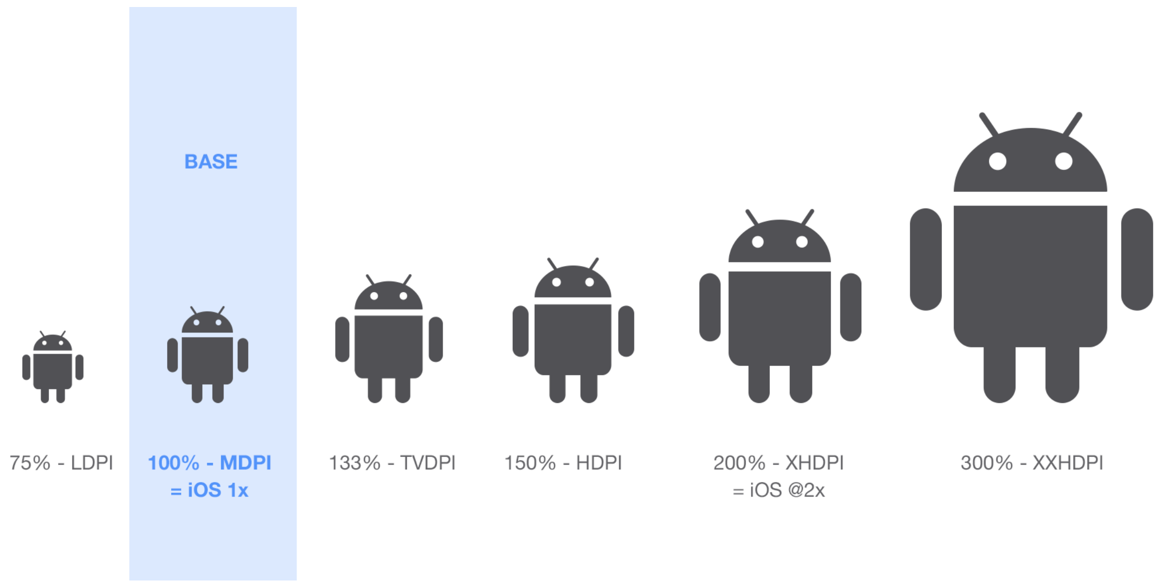
For this section we're going to take a different approach than for iOS. We'll talk about the multipliers and categories of DPI first.

Like for iOS, you have two categories of devices: phones and tablets. Both categories can be arranged in different DPI categories: ldpi, mdpi, tvdpi, hdpi, xhdpi, xxhdpi and xxxhdpi.

Fortunately, some are used more frequently than others, some are even deprecated.

The first thing we have to do is to find the base unit that is the equivalent of the 1x for iOS. On Android, this base is MDPI.

Let's take a look at the multipliers in the cheatsheet below.



Yes, it's a lot, and it's not over. There is one left.

SHARE

400% - XXXHDPI

Effectively there are five DPI in use: MDPI, HDPI, XHDPI, XXHDPI and XXXHDPI.
LDPI is an old DPI, not used anymore, TVDPI was a specific case for TV UI and was used briefly for the Nexus 7 2012 edition. It can be considered not necessary for phone and tablet use. A note though, the TVDPI's multiplier(1.33x) is used in some of Android Wear's devices such as the LG G watch but we'll talk about that later.

Let's put everything in perspective by associating Android phones and tablets with their respective DPI.

LDPI

HTC Tatoo

SHARE

MDPI

LG Optimus S

TVDPI

Asus Nexus 7 tablet 2012

SHARE

HDPI

Samsung Galaxy S II

Google Nexus S

XHDPI

Samsung Galaxy S III

Google Nexus 9 Tablet

SHARE

XXHDPI

Google Nexus 5X

XXXHDPI

Google Nexus 6P

Samsung Galaxy S6

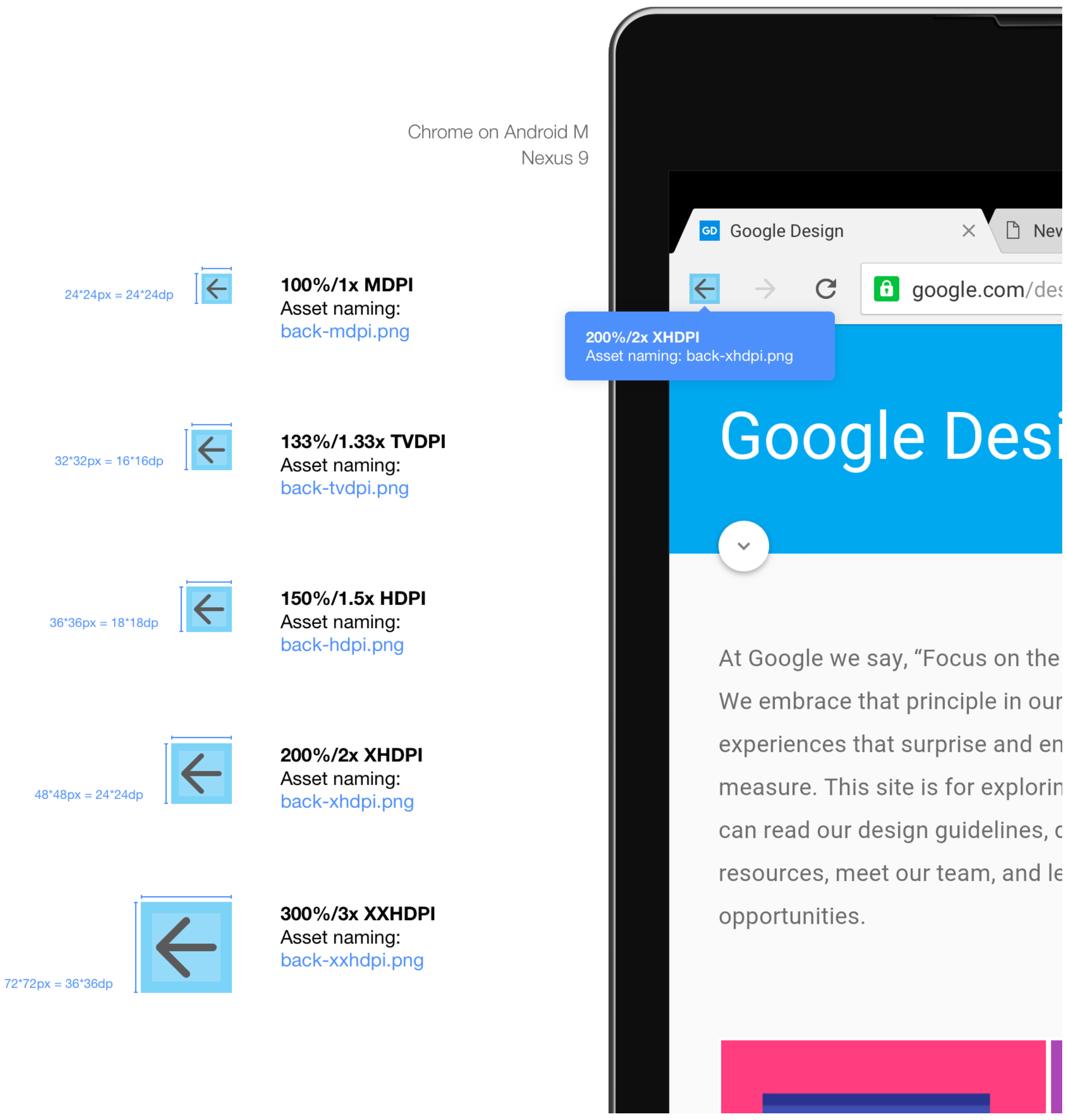
Required Assets, Chrome example.

You are going to have to deliver a set of 4 images per asset, from MDPI to XXHDPI. You can leave LDPI out of the set. Note that in the case of the version of Chrome showed below, TVDPI was exported as well that why the count is 5 images per asset in this specific case.

Just as for iOS, I suggest you take the 100% or 1x multiplier as a base for your design. This makes prepping the design for every other multiplier easier, expecially on Android

SHARE

See below for an example Chrome's back button on Android.



The naming suggested here with the appended DPI is not something that is mandatory nor presented in the Android official guidelines. It is the way that we have been naming our assets because of the limitation of the current design tools that make it hard to define a specific path per asset export.

Considering that an asset source can sometimes hold hundreds of assets, it is a way to make the export process less painful and to avoid duplicated names errors on the designer's side. The way that the asset buckets are structured in the source repository will be as followed:

- drawable-mdpi/asset.png

SHARE

As you can see, the asset is cut in a 32*32dp square. The issue with Android multipliers is that some of them use decimals. When you multiply a number by 1.33 or 1.5, chances are you're going to end up with a decimal number. In this case you'll want to round the number to what you think makes sense. In the case of the example, 32*1.33=42.56 so we rounded it up to 43px.

You'll need to be careful for pixel-sized elements such as stroke. you may want to make sure your stroke is either 1px wide or 2px wide and not blurry as described in the screen resolution section.

Takeaway, Android ruleset:
- Android has 7 different DPIs, you need to worry about 4: mdpi,hdpi,xhdpi,xxhdpi plus one if you want to future-proof your app, XXXHDPI
- MDPI is the base DPI or your 1x multiplier
- Android uses dp instead of pt for specs, but they are the same
- Use your best judgment for decimal multipliers.
- Deliver .png images.
- Make sure to validate your naming convention and export process with the person responsible of the implementation.

Mac and Chrome OS PPI

Mac (OSX) and Chrome OS behave quite the same way in term of PPI handling.
Both OS support regular PPI (100%) and hi-res / retina PPI (200%). Like for the iPhone and iPads, there is only a 2x multiplier.

Even if most of your users, both Mac and Chrome OS will be on low res devices(for now), I highly recommend future proofing your apps for these high-end screens. Future proofing for ChromeOS means creating hi-res assets for your Web-app or website, which will never be wasted time.
There are currently a total of 3 laptops handling this PPI, the Macbook pro 13", 15" and

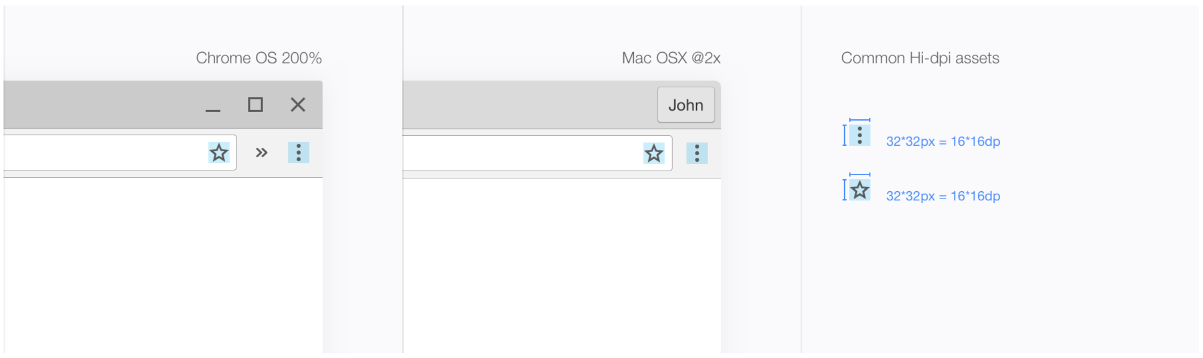
SHARE

Physical screen sizes: 13.3” and 15.4”
Resolutions: 2560x1600px and 2880x1800px
PPIs: 227 and 220

Physical screen size: 12.9”
Resolution: 2560x1700px
PPI: 239

Required Assets, Chrome example.

A perfect example of this similarity would be the Chrome toolbar assets button. We use the exact same ones across both platforms. If the code is different, the visuals are the same. See below the Chrome menu and bookmark buttons example.



Takeaway:

- Chrome OS and OSX use the same multiplier, 2.

- The only Chrome OS hi-resolution display also handles touch.

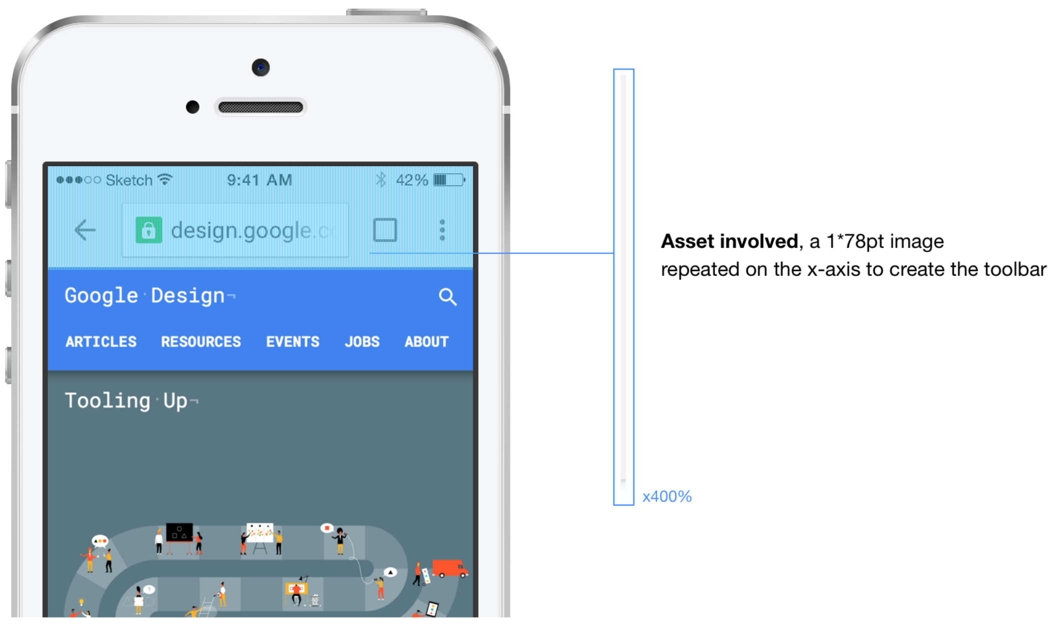
SHARE

Whether your app is on desktop or mobile. You'll almost always require stretchable assets.

A stretchable asset is set up so the code will be able to make it as big as it needs to be without degrading the rendering.

They are different from repeatable assets, which work differently even while sometimes displaying the same result.

See the Chrome example below. The toolbar on iOS is generated using only one super thin asset that is repeated on the X-axis across the entire screen.



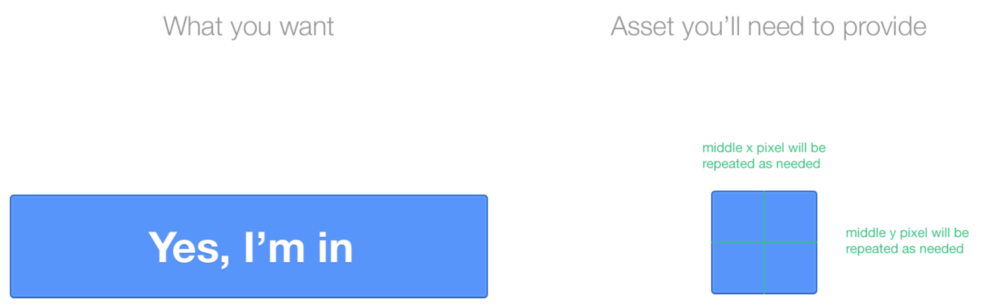
Now that this is out of the way, let's see how different platforms handle stretchable assets.

Stretchable assets on iOS

iOS makes it easy for the designer because the stretch is defined in the code rather that in the way you make your asset slices or markings. All you'll have to do is provide a base

SHARE

content button on iOS.


What you want
Asset you'll need to provide
Yes, I'm in
middle x pixel will be repeated as needed
middle y pixel will be repeated as needed

Stretchable assets on Android

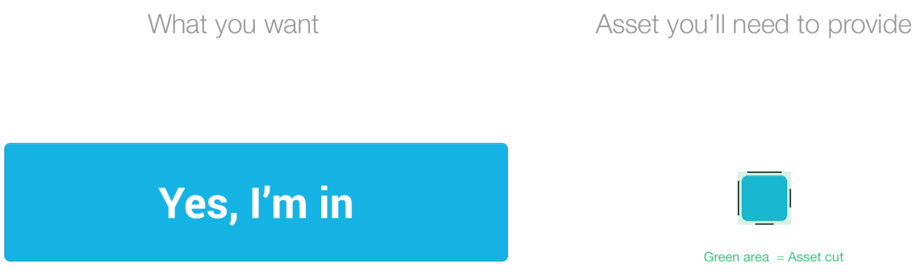
Android has a different way of doing stretchable assets than iOS. It relies a bit more on the designer.

For this platform, you'll be using 9-patch guides. These guides consist of 4 lines surrounding the asset itself. They have to be delivered in the slice/image like it is part of the visual itself, literally visually display its specs within it.

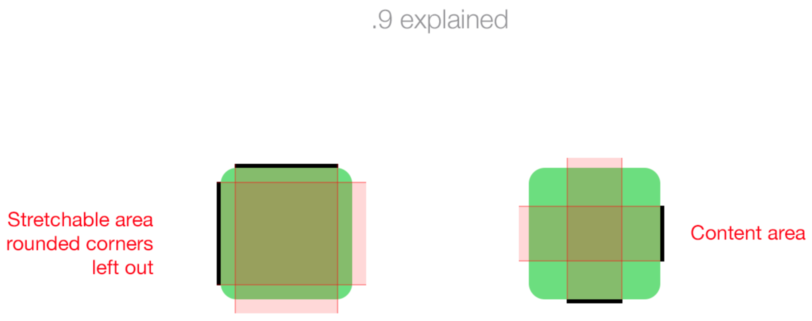
They define two things: the scalable area and the fill area. Once these are defined, the code will only be able to stretch what you defined and put content where you defined it to go.

See the example below, which is the Android version of the default Chrome button you saw earlier. I made it bigger on purpose for the demonstration.

SHARE

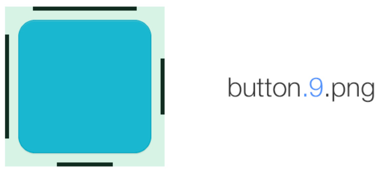


As you can see, the 9-patch is a set of 4 pure #000000 bars. They should have a width of 1px for any DPI; this is a code indication. The stretchable area does not include the rounded corners because it is not something that can be repeated (or it will look terrible.) In this case, we added a 10dp padding for the button. This is something you won't have to spec out. .9 indicators also need to lay and a 100% transparent part of the asset cut. If not, it won't work and require modification.

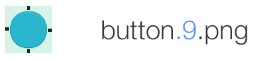


Using 9-patch requires you to append .9 to the name, the same way you add @2x for iOS assets. Retaking our button asset example below:

SHARE



Note that you should be careful of the size of your asset. If I made it quite big for demonstration, it is important that you optimize your asset weight by reducing it's size to a minimum, as show below. I kept the corners as they were but reduced the stretchable and content area to a minimum.



Be careful that the 9-patch markings do not overlap your design and that the cut of the asset is correct. The .9 should be as close to the asset as possible without overlapping it, try not to build-in padding. The example before has built-in padding because of shadowing.

The 9-patch doesn't replace exporting your asset in every DPI. It needs to be done for each version of the asset.

SHARE

Takeaway:
Always ask the person implementing your design what's the best solution to adopt, especially for desktop. The more images you'll have, the heavier the app will be, and it will become harder for you to track and update your assets. 9-patch should be used only with good naming and good organization of your sources.

Vector assets

As the range of screen using an even greater range of DPI configuration increases, switching to vector assets instead of bitmaps is an option absolutely worth considering.

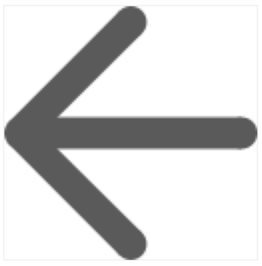
The most used and spread out form of vector drawable asset is the .svg format. It's and .xml based file, readable and editable in almost all design softwares as well as web browsers, as it was created for the web. Other format supports vectors such as .ai (Adobe illustrator), .eps or even .pdf.

The primary benefit out of using vector images is the scalability. No need to create bitmaps for all the various PPI buckets, the vector will scale automatically based on the screen multiplier.

SHARE

192dp **vector** asset scaled 300%

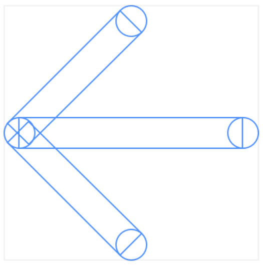
192dp **bitmap** asset scaled 300%



The .svg carries XML information on how to draw the visual. The software/OS/browser then interprets these command to render the asset in the chosen size.

.svg rendered in software

.svg drawing strokes



SHARE

```
<?xml version="1.0" encoding="utf-8"?>

<svg version="1.1" id="Layer_1" xmlns="http://www.w3.org/2000/svg" xmlns:xlink="http://www.w3.org/1999/xlink" x="0px" y="0px" width="72px"
height="72px" viewBox="0 0 72 72" style="enable-background:new 0 0 72 72;" xml:space="preserve">

<style type="text/css"> .st0{fill:#5A5A5A;} </style>

<g id="Page-1" sketch:type="MSPage">
    <g id="Artboard-9" transform="translate(-457.000000, -468.000000)" sketch:type="MSArtboardGroup">

        <path id="Fill-1" sketch:type="MSShapeGroup" class="st0"

d="M495.1,528.6L474.5,508h46.3v0c0.1,0,0.1,0,0.2,0c2.2,0,4-1.8,4-4s-1.8-4-4-4c-0.1,0-0.1,0-0.2,0v0h-463l20.6-20.6c1.2-0.7,1.9-2,1.9-3.4c0-2.
2-1.8-4-4-4c-1.1,0-2.2,0.5-2.9,1.3l-0.1-0.1l-28.4,28.4l0.1,0.1c-0.4,0.6-0.7,1.4-0.7,2.3s0.3,1.6,0.7,2.3l-0.1,0.1l28.4,28.4l0.1-0.1c0.7,0.8,1
.8,1.3,2.9,1.3 c2.2,0,4-1.8,4-4C497,530.5,496.2,529.3,495.1,528.6z"/>

    </g>
</g>
</svg>
```

There are tremendous benefits in using them:

- Reduction of the app install size
- Massive reduction in bitmap usage
- Programmatic color change is made easier
- Automatic and non-destructive scaling

However there are downsides to them as well:

- Less visual freedom, not good for complex graphics, espacially complex shadows, gradients or other type of effects
- May have an bad impact on app/site performance, because of the amount of processing power required
- No control over pixel perfection because of automated scaling

With the evolution of UI design towards a "flatter" style, using less shadows, etching and gradients, the .vector path is becoming more and more useful, and used. That being said, you need to be careful as to how you use it and where.

As mentionned in the downsides, .svg might impact performance tremendously and while they work perfectly well on the web, iOS and Android will prefer their own vector drawable solutions. iOS is using **.pdf**, Android is using **VectorDrawables**.
Android even created a tool in Android studio to convert .svg to Android-ready Vector Drawable code.

Takeaway:
Depending on your project, what you'll need to deliver to your implementation team or person will vary depending on the targeted platform. Make sure that the solution you opt

SHARE

Always check with your engineering team to understand what is the best solution for your project and what are the implication performance and design wise. Vector drawables might be your best allies but they might not fit every use cases so be careful.

Touch and touch targets

The first thing to understand here is that making something touch ready has nothing to do with DPI. But when it comes to creating a UI or generating assets, it is important to understand the relationship between touch and DPI.

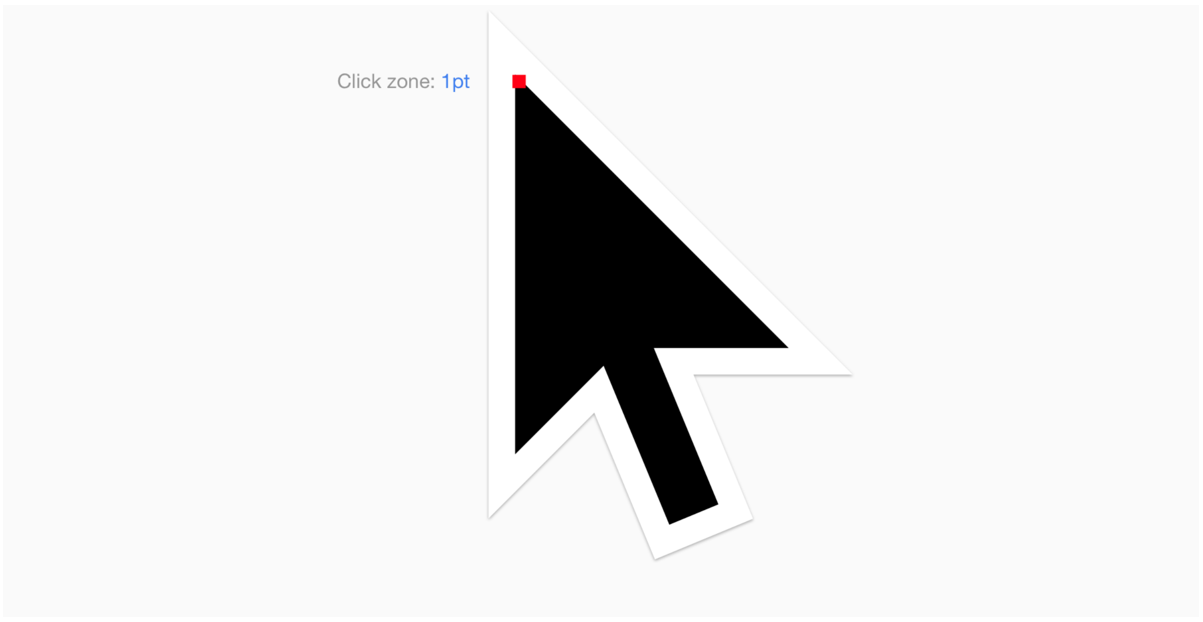
Making the choice between touch or non-touch will highly depend on the scope of your app, where it is going to be deployed and how you want the user experience to be.
Let's split it in simple categories, desktop non-touch and mobile.

Desktop, non-touch

Let's not give a history class here but unless you were born in 2005, you know that the computing technology wasn't created with touch in mind.
We use mouse and keyboard, that are extremely precise tools to navigate a UI. The precision of your mouse cursor is 1pt. You could in theory create a 1x1pt button that would be clickable by any super-human out there.

See the illustration below.

SHARE



This is a 20x version of the Chrome OS cursor.
The red zoning it the actual area that triggers an action on the UI. Pretty precise.
You know were I'm heading. What's not very precise ? Our fingers.

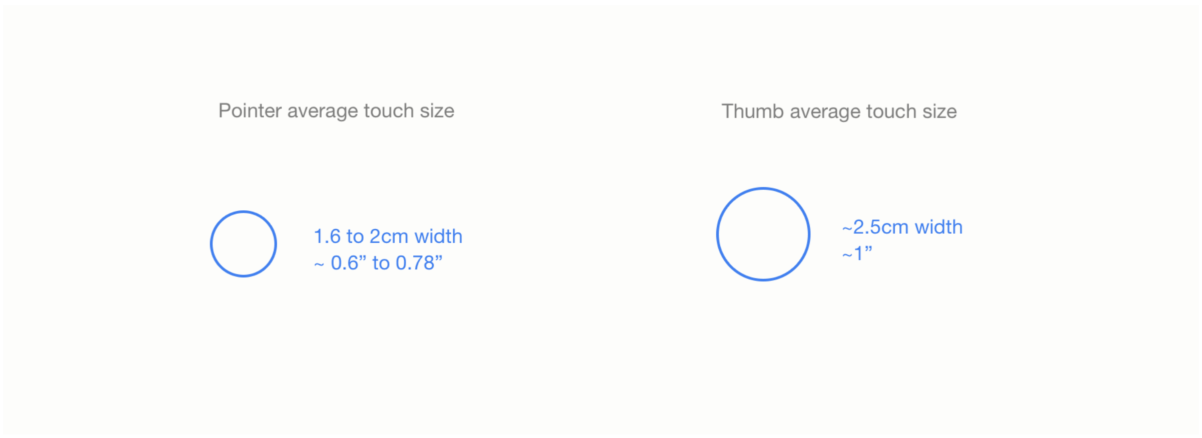
So how do you design for touch ? Well you make everything bigger.

Finger size

Here's the average size of the two most used fingers for UI interaction, the pointer and the thumb. It represents both the touch zone and the the area obstructed by the finger. The actual touch zone (i.e the part of your finger that is in contact with the screen) will of course be smaller and a bit more precise, unless you really smash your finger against the screen.

When designing for touch, it's safer to overestimate the size needed for touch targets than underestimate them.

SHARE



How to apply this to my design flow

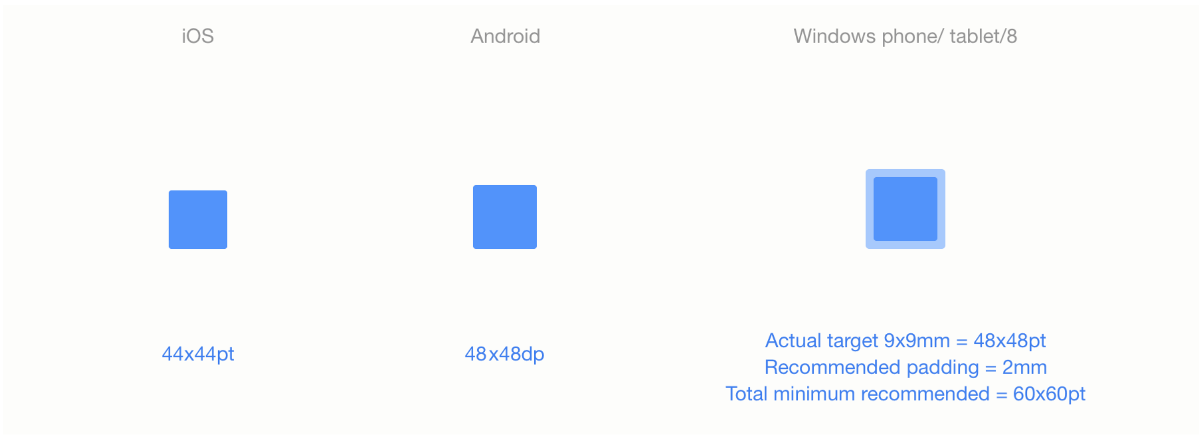
As we already saw, inches or cm are not the best way to count in a pixel world. Matter of fact, even pixel is not a really good way to count. So how do you make sure your design is touch ready ?

I'm going to state the obvious but you should always try your design on the targeted devices/platform.
But to avoid losing too much time, there is some base pixel-based sizes that are considered safe to use and that are recommended on an per OS basis.

Recommended touch targets per platform

Again, careful, these sizes are for convenience and are not a unit of real life size measurement whatsoever. They work because OEM and manufacturers are following guidelines to make screens consistent in term of size/dpi ratio.

SHARE


iOS
Android
Windows phone/ tablet/8
44x44pt
48x48dp
Actual target 9x9mm = 48x48pt
Recommended padding = 2mm
Total minimum recommended = 60x60pt

As you can see each OS has its own set of recommendations but they are all around the 48pt. Windows includes the padding in its specs, that's why I added it here.

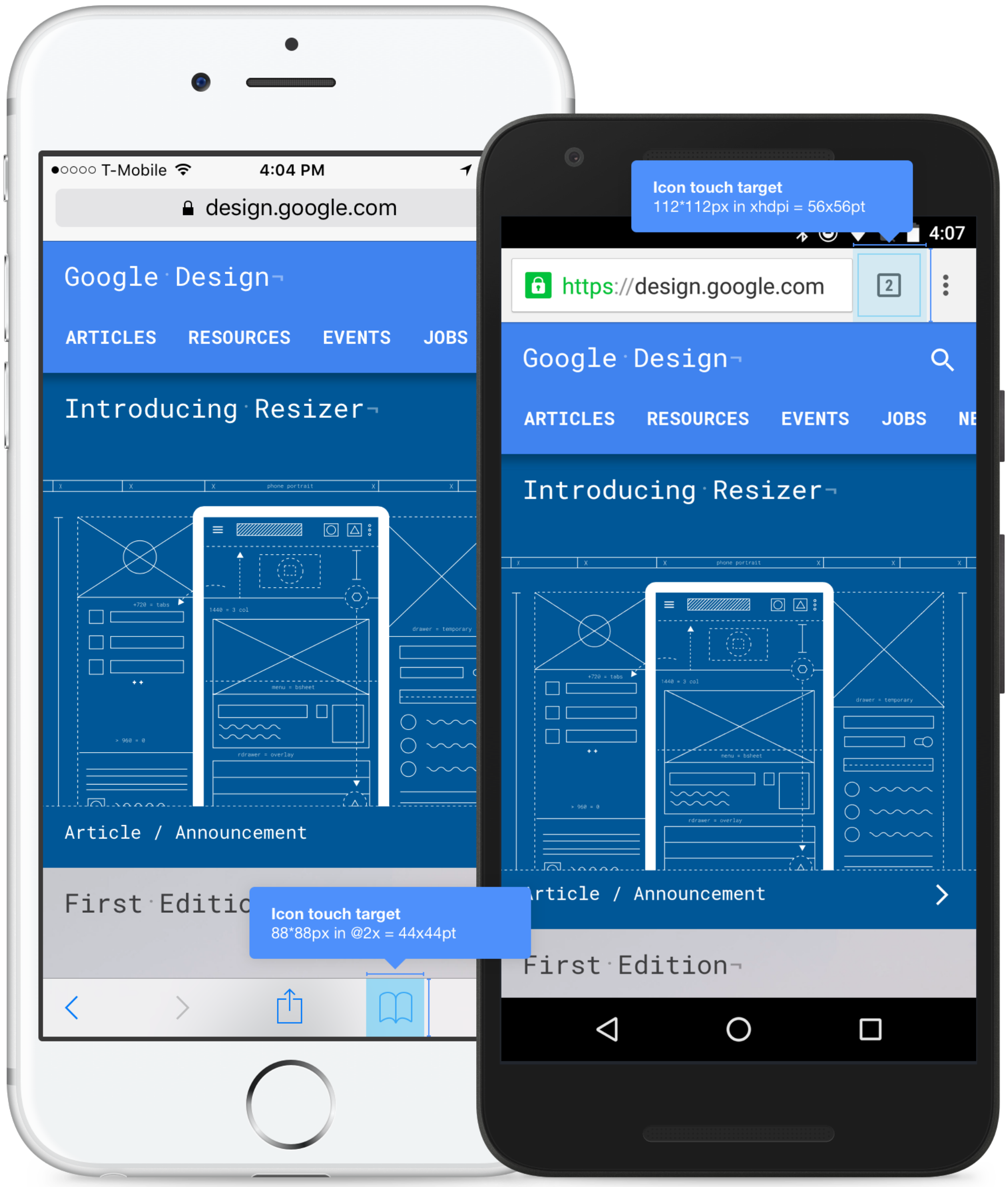
The difference in these sizes comes from different factors.
Apple controls its hardware so they know the quality of the touch screen and control the exact ratio. They can rely on a smaller touch target. Additionaly, their hardware tends to be physically smaller.

Android and Windows on the other hand have different OEMs, each building its own hardware, having bigger touch targets makes them "safer". Their UI is also more spaced out (especially windows) and their devices tends to be physically bigger.

Example

Here's an example of how Safari applies touch target on iOs and how Chrome applies it on Android.

SHARE



As you can see, both toolbars are the recommended touch target height for each platform. Also the area surrounding the visual is a 44x44pt and 48x48pt square for iOS and Android respectively. Note that since the release of Material design, the height of a toolbar has been bumped to 56dp, increasing icon effective touch targets to 56*56dp within the toolbar. Minimum touch target for the reste of the UI remains at 48dp. Not only this makes the UI consistent with the rest of the OS in term of sizing but it's a great way for you to have a minimum size for everything you want the user to interact with.

SHARE

for a Windows 8 app, I'd highly recommend following their guidelines for touch targets.

Chrome OS guidelines has yet to be released but and the Pixel usage it not big. However, since all Chrome OS apps are web based, I'd suggest designing to touch anyway. My recommendation would be to apply the Android touch targets guidelines.

The web, hybrid devices and the future.

If you're designing for mobile, it will be clear what decision to make, go touch. If you're designing for desktop, go non-touch. It sounds easy but it'd be ignoring a new tendency that arrises, hybrid devices.

An hybrid device is a device that supposedly does both touch and non-touch. The Chromebook Pixel, the Surface Pro and the Lenovo Yoga are a good example.
What to do in this case ? Well there is no easy answer but I'm going to go ahead and give one, go for touch. That's where the technology is going to evolve.
If you design for the web, or anything for that matter, think touch upfront.

Takeaway:
- Think mobile, think touch in almost everything you'll do in the future.
- Use recommended touch targets for each OS. This will help make your design better and help you reach consistency within the OS. - Touch targets are reference values, it doesn't mean you should follow them to the letter. Ultimately, you control the experience.

Design softwares

The software doesn't make the designer, but choosing the right software for the task at hand can improve your productivity and ease of work by quite a bit. Software "know-hows" shouldn't be your only skill but learning and mastering the right tool will be a great asset to make your ideas happen.

When it comes to handling DPI variation in interface design, different software work in different ways. Some are better at particular tasks than others. Here are the most

SHARE

Photoshop CC Illustrator CC Sketch Figma

Photoshop

The mother of interface design tools. Probably the most used tool out there today. There are an infinite number of resources, tutorials and articles for it. Photoshop has been around almost since the beginning of interface design.

As its name suggests, the first intention of the program wasn't interface design but photo or bitmap retouching. It evolved over the year and with the birth of interface design, designers appropriated it and re-purposed it. Part of this was because they were used to it and because it was the only program around that was able to do things as good as needed.

Photoshop is, to this day, the master of Bitmap editing and is still the most used program out there for UI design. Its decades long legacy makes it a hard program to approach and learn though. As a gigantic swiss army knife of a software, you'll be able to do anything, but not always in the most efficient way.

As it it bitmap based, it only offers pixel preview, meaning that the preview will match your monitor pixels and that contrarily to Illustrator, it will not re-render your vector drawings based on your zoom level. It will however handles scaling perfectly fine.

Adobe's official site: https://www.adobe.com/

Illustrator

Photoshop's vector based sibling. As its name indicates, it is aimed at Illustrators but it is

SHARE

Illustrator is suited for print design as well so its interface, color management, scale, rulers and units may throw you off at first and it requires a few tweaks to be easily usable for interface design only. Like Photoshop, it is an incredibly powerful tool with a steep learning curve.

What differs from Photoshop is that it is DPI independent due to its reliance on vector shapes. Contrary to bitmap or raster images, graphics made using vector shapes, relying on mathematical formulas, will be rescaled programmatically without any quality loss.

Understanding the difference between rasterized and vectorized image is key to build scalable visual design and assets.

If you want to get started with using Illustrator for web/interface design, I recommend reading "My vector workflow" by @janoskoos.

Adobe's official site: https://www.adobe.com/

Sketch 3.0

Sketch is new compared to Photoshop and Illustrator. With only 4 years of age, this program generated a lot of hype (in a good way) in the UI designer industry. The reason is that Sketch is aimed, from the start, to be used by interface and UX designers. Without the legacy of Photoshop or Illustrator, Sketch positions itself as the perfectly adapted tool for the niche audience that is interface designers.

Sketch is suited for fast wireframing as well as more complex visual design. It is entirely vector based, like Illustrator, with a minimal and well thought UI. The combination of artboards and the ease of use and flexibility of its asset generation system makes it the fastest tool for multi-DPI and multi-platform design. The recent release of its 3.0 version make it a very solid alternative to Photoshop.

On the downside, Sketch is supported by a smaller team and is still fairly recent. Its team is extremely reactive but doesn't have the scale of the Adobe (Photoshop and Illustrator) one. Sketch offers (by design) the bare minimum when it comes to bitmap edition. Photoshop will be more suited for this kind of job. Finally, due to its fairly still young life, the resources in term of source files, tutorials and overall community is orders of magnitude smaller than Photoshop. That being said, the community is very active and

SHARE

On a more personal note, I've been a Photoshop user since I started design 8 years ago but I recently switch to Sketch 3.0 for the most part of my design process. This is not a judgement of quality, Photoshop is still a hell of a good program, it just suits my needs better.

If you want to learn more on my particular experience I encourage you to read my "A month with Sketch 3.0" article or my "Sketch tutorial_01".

Want to get even deeper and understand how vectors work in sketch ? Head to @pnowelldesign's article "Harnessing vector awesomeness in Sketch"

Sketch's official site: https://www.sketchapp.com/

Figma

Newcomer of the end of the year 2015, Figma is a browser based (Chrome most specifically) and vector based design tool. Think of it a Sketch in the cloud with team collaboration capabilities and Slack integration. An impressive feat of engineering, paving the way of the future of design tools.

The biggest benefits are the cross-platform availability (anything that can run Chrome) and team collaboration/simultaneous editing. However, if you or your company are not comfortable with entirely web-based services, this might not be a good fit as no local version of the tool exists yet.

Figma's official site: https://www.figma.com/

Takeaway:
There is no perfect tool for the job but the one you are comfortable using. If you can afford the time an money, I recommend you try them all to make up your own opinion.

Doc and resources

This guide was only an introduction, time to start doing and learn more. Here are a few

SHARE

Platform documentation

Android UI guidelines

Android Auto guidelines

Android TV guidelines

Google Material guidelines

iOS7 UI guidelines

Windows UI guidelines

Google dev Principles of site design

Apple TV guidelines

Cheat-sheets and templates

Design resources index by @MengTo

Material design device metrics

Material design device resizer

Facebook design resources and templates iPhone 6 Screens Demystified

Ultimate guide to iphone resolutions by PaintCodeApp

Screen sizes, ratio and PPI

iOS7 designer cheat sheet

iOS7 design resource (requires Apple account)

App icons template, Android and iOS

Bjango blog (A design article gold mine)

iPhone GUI and iPad GUI(.psd) by @teehanlax

Tools

Sketch app rock, index of Sketch resources

Density converter by @brdrck

Android asset generation by @brdrck

Android design tips by @destroywerk and @BPScott

9patch creation in Android by @romannurik

Android asset studio by @romannurik. Lots of great tools for Android specific asset creation.

Great reads and more information

Designing at 1x

SHARE

[Best practices for using .svg on Android]
[Harnessing vector awesomeness in Sketch]
[Device independent pixel formula for Mobile devices]
[More information about 4K by Cnet.com]
[More informations about touch targets by Smashing Mag]
[The Android Screen Fragmentation Myth]

About

I'm Sebastien Gabriel aka @Kounterb. I'm a visual designer for at Google, working on Chrome and Chrome OS. I also write things on Medium.

One thing I wish I had when I started is a clear guide explaining to me what DPI is and what the challenges of multi-platform design were going to be. This is what I'm trying to do here. By designing Chrome for almost every platforms out there, I learned a lot about these subjects and this is my effort to try and deliver it in the simplest way possible. If you think I got anything wrong, if anything lacks detail or if you would like to learn more about something, send me a message at sgabriel.contact@gmail.com.

SHARE