Using Images with MIT App Inventor

Avoid some common pitfalls!

Synopsis: App Inventor works best if you use images whose size matches the size you want them to appear on your screen. If you import larger images into your app, your app may run out of system memory.

1. Out of memory errors

People building apps in App Inventor might find that their app crashes when they try to run it, with an error message like:

> *Failed to allocate a 25165836 byte allocation with 3395432 free bytes and 3MB until OOM*

What's happening here is that the app is is running out of memory (OOM). In the case of the message above, the phone is trying to allocate 25 megabytes (25165836 bytes) but there are only 3 megabytes (3395432 bytes) free.   The most common reason for attempting to allocate such a large amount of memory is that the phone is trying to display a huge image, generally an image much larger than the app requires.

Every image has a "size" based on its width and height in pixels.   You can multiply these two numbers to get a sense of the amount of memory required for displaying the image.

App Inventor lets you get images (for example, from the Web) and import these into your app as Media. When you use the images, and App Inventor will rescale them to fit the designated screen area in your app.   Sometimes the image to be displayed will be larger than the designed phone area.   Even so, the large image needs to be held in memory in order for rescaling to occur, even if the result of the rescaling will be a small image.

2. An example of misusing large images

Suppose you are going to display a grid of the letters of the alphabet, where each letter is in a 30×30 pixel square. You'd expect that to take 30×30 = 900 pixels per letter, so the entire grid (assuming 4 bytes per pixel, which is typical for Android) would require 900×26×4 = 93600 or about 100 kilobytes, which will easily fit in the phone memory.

On the other hand, suppose that each letter is loaded from its own image file into the app's Media from a high-resolution image where the size of each image is 1000 pixels by 1000 pixels. Each of those images requires 1000×1000 = 1 million pixels or 1000×1000×4 = 4 Megabytes and the entire grid would require 26×4 = 104 Megabytes. Typical amounts of memory available to a running app range anywhere from 20 to 50 Megabytes (not to mention that MIT App Inventor requires the app source file, including images, to be less than 5 Megabytes). So the app will simply not work, and the project will get an OutOfMemory exception when it loads.

Even if the app could work, the result on the phone would hardly look different than if you'd used 30×30 images for the letters. At the end of the day, each image on the screen would be 30×30 pixels, even if it required 4 Megabytes to generate it: a huge amount of wasted space.
Keep in mind that if the original image is larger and is scaled down to 30×30 for display, the image will still consume memory on the Android deviced based on its actual large size.
There's no reason to load a giant image file into your app if you will use it displayed only as a small image on the phone.

3. Use images that are "Just the right size"

The general point of the alphabet example above is that you should try to use images whose size is close to the actual size they will appear on the device. There's no reason to use larger images: it just wastes space. And if you use smaller images, App Inventor will scale them up to fit the screen, which may lower the image quality.

> Rule: Resize images to make them not too large and not too small, but rather, "just the right size".

For example, it's common to see apps that violate this principle with background images, where a person downloads a high-resolution image (e.g. 3000×3000 pixels) and sets that as the screen background. That just wastes space. It's better to resize the image to better match the device's screen size before uploading it to the app.

There are many tools that you can use if you need to resize images to be "just the right size". Tools include Photoshop, Preview (on MacOS), GIMP (on MacOS and GNU/Linux and Windows) and Paint (on Windows PCs). There are many more. Some tools are commercial software other tools are free or provided with the computer's operating system. You do not need a particularly fancy program (like Photoshop) just to do image scaling. There may even be websites that you can upload your images to that will rescale them or compress them for you. A Google search for “programs for scaling images” or “image compression” will likely turn up more than a few candidates. However, compression is not what is needed; you want to *resize* to “just the right size.”

You can also use services like Google image search to find images for your app that are already the right size, so you don’t need to rescale them.

Keep in mind that image size refers to the size (height and width) in pixels.   You can find this using one of the tools above.   It’s not good to rely on the file size of the image as a guide because most image file formats have some amount of compression. Depending on the type of image and the actual image contents, a large image may compress to a small file. The compression factor can be particularly significant with graphics (as opposed to photographs).

4. Preferred Image Format

What is the best image format to use for images images?   Although many formats are supported, most users will get the best results using JPG (JPEG) or PNG images. The PNG images will generally look sharper than JPG on Android devices, due to the difference in compression algorithms although the file sizes tend to be larger that JPG. (PNG uses lossless compression; JPG does not.) Avoid BMP images; BMP images are huge compared to other formats because they are not compressed at all.

5. Picking “just the right size”

When you resize an image for use in an app, you don’t need to exactly match the size it will appear on the screen.   It most cases, it will be adequate to get sizes within an approximate range and let App Inventor’s automatic scaling do the final adjustment.   One exception to relying only on automatic scaling is that if the image size is much smaller than the screen area size, then the image on the screen might appear low quality.

You might want to try to avoid automatic rescaling altogether and match image sizes to the screen area size exactly. That could work if you are designing your app for a single device. But different devices have different screen sizes and different screen densities (pixels per inch), so there’s no single image size that can exactly match all devices.

In principle, having your app display the best quality images on all devices, could require you to provide copies of each image at several sizes match to different devices, and then have the app pick the appropriate image when it runs.   This is the approach taken by Eclipse and Android Studio. In designing App Inventor, we did not want to require our developers to deal with this complexity.   So we suggest that you pick “just the right size” depending on the device you care most about and rely on App Inventor’s automatic scaling do the best it can. We might add the multiple image capability to App Inventor if there is enough demand for it.

6. Determining the size of the image on the screen

Multiple devices aside, the size for the displayed image on the screen will depend on how the app is designed.

In some contexts, when you use an image, the size of the image will be used to determine the size of the object on the screen. For example in the HelloPurr tutorial, you set a button to have an image (the cat in this case) as its on-screen representation and set the Height and Width of the button to *Automatic*. In this case, the size of the image determines the size of the button.

In other contexts the image is scaled to fit a specified size. For example if you place an Image component on the screen and set its Width and Height properties each 100 pixels, then the image displayed will take up 100 pixels of the screen in each direction, even if the actual image you load is a different size. Android will automatically scale the image to fit the size of its container when it is displayed. Keep in mind that if the original image is larger and is scaled down to 100×100 for display, the image will still consume memory on the Android deviced based on its actual large size. To repeat the main rule: It's just wasting space to load a huge image if you will use it displayed only at small size your app.

7. Details on fixed vs. responsive sizing

This section provides some background on App Inventor image scaling and how it has changed over different App Inventor releases.

App Inventor introduced Responsive sizing in August 2015. Prior to that release, App Inventor created Apps that were marked for Android API 3 (Android version 1.45). That meant that MIT App Inventor Apps prior to version 1.45 of App Inventor 2, created apps could run on phones as old as Cupcake (Android 1.5). However there was a catch. With API 3 Android supported only a single device size (320 pixels wide by 480 pixels high) and a single screen density.

Newer devices usually are both larger than this and support "denser" pixels (there are more pixels to a inch of screen space). Unfortunately, when a newer device loads a program designed for API 3, the Android automatically scales the screen to make it "look" like the older device. This compatibility mode makes tablets look like giant phones and simply magnified the 320 x 480 pixel screen, which could make images (and fonts) appear to be low quality.

As Android evolved and MIT App Inventor matured, AI2 developers expressed a desire to design apps that looked good on tablets. Additionally newer Android devices are being shipped without the API 3 compatibility mode. This made MIT App Inventor apps use just a small section of the screen and not look right.

To support those devices the MIT development team needed to abandon support for Android 1.5 and require at least to Android 1.6 (Donut, API 4). Starting with API 4, Android is aware of different pixel densities and screen sizes. The improved screen handling makes Android application development more complicated than before because now the application developer has to design their application to deal with different screen sizes and densities. The approach taken by Google is to design several UI versions for your application, complete with each version's image files, and provide all of the images in your APK file. The device then chooses the correct set at runtime. This is the system used in the Eclipse and Android Studio compilers.

We decided that creating multiple UI designs on the Google Eclipse/Android Studio model, is more complicated than we want to subject the App Inventor programmers to. We also did not want to make a change to App Inventor requiring every App Inventor programmer to deal with different size devices and multiple duplicate images. We did want, to maintain the compatibility mode that we had with API 3. However to do so requires us to provide our own compatibility code, which we have. The result is Fixed sizing (which is an attempt to emulate the original API 3 compatibility mode as much as possible) and Responsive sizing.

In current releases of MIT App Inventor you can set the "Sizing" property on Screen1 to either "Responsive" or "Fixed" (with "Fixed" being the default for new projects and older projects that automatically get upgraded). Fixed mode attempts to emulate the old API 3 compatibility mode behavior. The emulation does not duplicate the old mode exactly, but it is close enough in our opinion. Responsive mode does not invoke this compatibility feature.   Programmers using Responsive sizing may have to pay attention to the differences between devices. To make this opportunity to code for multiple screen sizes as easy as possible for AI2 developers we introduced a way to specify the width and height of screen elements based on a percentage of the device's screen, instead of having to give absolute pixels.

Another change that comes with this upgrade is the notion of "Density Independent Pixels" or "dips." Density independent pixels are a measure of size that takes into account the screen density of a device. On some devices a dip may map to one pixel while on others it may be nine pixels. The idea is that most screens will have the same number of "dips" to an inch of screen real estate.

Developers can still give UI elements sizes in "pixels" in App Inventor, but realize those pixels are in fact dips.

Images are still based on "real" pixels. So if you load HelloPurr onto a denser device, the cat picture will be smaller than on a 320×480 pixel device. To deal with this difference App Inventor automatically scales images when it loads them based on a device's screen density. This automatic response can cause issues with memory consumption.   Returning to the Alphabet example above, if you attempt to load the Project into a device with a screen density of 3 (not uncommon) the Project will try to use 9 times as much memory as the original example with density 1, or 2.3 Gigabytes of memory. So even if the original example fit in memory on a device with a screen density of 1, it will certainly not fit on a device with a screen density of 3!

8. In summary: What MIT App Inventor Programmers Should Do

The simple solution is to make sure that the images you load are the “correct size” for the device you care about most and let App Inventor scaling handle the rest.   To return to the Alphabet example, the images of the individual letters should be scaled to be 30 pixels by 30 pixels. If you resize the images to 30×30, each image will require only about 1000 bytes of memory (instead of 10 Megabytes!). The entire alphabet will fit in 26,000 bytes of memory. Even if you load this onto a device with a screen density of 3, it will consume only about 200,000 bytes of memory.