Part 2 : The Calculator Image

Sources of images

The best place to obtain an image is of course to take one yourself of a calculator you own (or have access to). A digital camera is essential here as you are unlikely to obtain a decent usable image by using processed film and scanning it in. The instant feedback of digital camera technology allows one to have multiple tries to get the lighting right, keys legible etc.

Another source of good images is the internet. There are many collectors out there with images, and you may find a suitable one of the calculator you want to model. If you do find one assume that it is copyrighted material. Nearly all websites have a contact e-mail address, which you can write to and ask permission to use the image. I have a three-out-of-three success rate with this approach, so don't be afraid to ask---but don't steal copyrighted material.

A third way to obtain an image is from an image you already have. If you want to simulate a particular calculator model which is very close in appearance to a model you have an image of (maybe the make name is the only difference, or just one or two keys) then it could be possible to use a graphics package to manipulate the image to make a passable representation of the model you want. It depends on how much effort you wish to spend. For example, I have an image of an Anita 811 and wanted an Anita 831. The differences were mainly a squareroot symbol and a switch label. Compare the two images below---both from the same calculator source image.

Features of the image

To get good results, the image you are going to want needs to have legible labels, enough contrast to show its colouring and mouldings, and be of a suitable size for an online calculator. No outside dimension should be greater than 500 pixels in general. A picture in natural daylight is always better, and flash photography is no good. Handheld calculators tend to be easier because they are rectangular, and a bird's-eye-view image gives all the salient fetures. A desktop poses a bigger challenge. A compromise between the angle of the keyboard and the angle of the display must be reached so that all the keys show clearly, but the display isn't too narrowly angled so the numbers are distored.

If lighting the calculator well weakens out the display (say in the case of an LED calculator), then a separate image may be made for the display in darkened conditions. The final simulator image can then be a merging of the body and display from the two photos.

Display images

The image of the display must contain enough features to make all the possible display images. The numbers may be all present in the image, or if suitable, an image of an 8 may be used to construct the others in a graphics package if the display is nicely segmented. If a decimal place is too close to a number so that it merges in the digitised photo with its adjacent number, then you may need the numbers in both normal and 'dot' versions. A minus

is almost certainly needed, and any error modes. If there are any other display features, you'll need pictures of those as well.

If the calculator has a display that 'glows' (e.g. an LED calculator), then you may have to dim the lights to get a clear set of numbers in a second image (turn off the camera's flash as well). Try and make the image of the calculator nearly the same size as the master image.

Switch images

If your calculator has switches that move, you'll need images of them in on and off positions. For multiple switches that are all the same, just an example of an off and on switch should be enough (like in the Anita 811 image above). Switches that rock (and I don't mean they are cool guitarists) cast shadows, and adjacent switches 'interfere' with each others lighting. The best approach here is to take pictures of the calculator with all combinations of adjacent switches (two switches have 4 combinations: on-on, on-off, off-on, off-off).

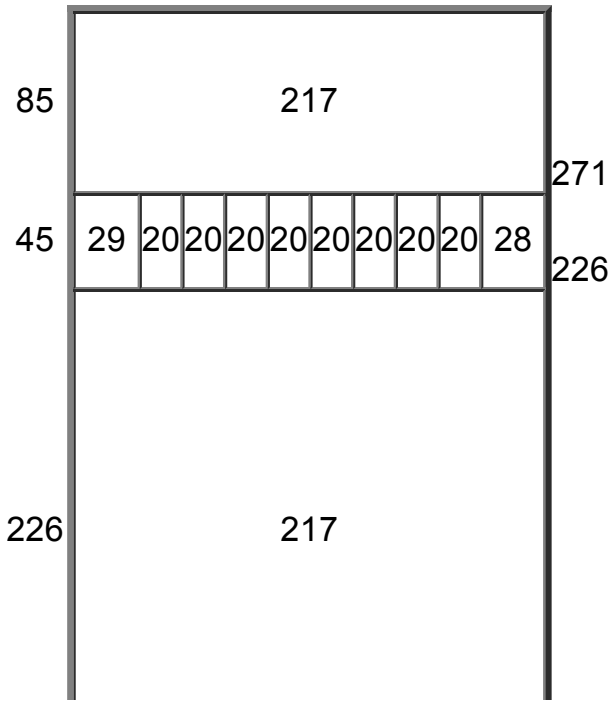
If you want to animate the keys, then good luck! Unless you're prepared to glue, stick, or jam a key in the activated state, you'll always have a picture with your finger in it. I have never simulated a calculator where the keys are activated, but do have a crack if you're feeling lucky.

Carving up the image

You now have one or more images with which to construct a composite image. Before breaking the image up, you need to decide where the cuts are going to be.

The image as a whole has two classes of area: areas that never change and areas that change. The latter type of areas are bits of the display where numbers and things alter, and switches that move (and any other bits you've chosen to animate). Get a pencil and paper and draw a rectangle approximately the shape of your calculator. Moving from top to bottom draw a line across the box at the start and end of any horizontal segment where anything will change. For instance, maybe a quarter way down is the display, so two lines for this. A row of switches is a third the way down, so two lines for these, and so on. None of this needs to be to an accurate scale. Just a representation.

Going back to each 'active' row, this needs to be carved up for each separate active area within the row. For a calculator with an 8 segment display with a combined number and decimal place, this row needs 10 areas---a left and right edge, and 8 number segments All the number segments will end up being the same size, but the left and right portions of the row don't need to be the same. Similarly for, say, three non-adjacent switches the row must be divided into 7---left, right, three switches and two areas between the switches. Unless you're intending to animate the keys, these will not need carving up in this way. In our example we have no switches, so only a display active area is needed, cutting up the image into three rows. A sketch of the example cut points is shown below (a bit neater than you'll need).



The table above shows the relative sizes of the images making up the calculator. The left-hand numbers show the heights of the three segments, whilst the right-hand numbers give the accumulative pixel number at the start of each segment. The numbers in the boxes show the widths of the segments. So you should now have a template for actually fragmenting the image. We don't know the numbers of the pixels where the breaks are going to be yet, and that's the next task.

There are two ways to determine how to break up the image at the right places: using a commercial software package, or using my `bmp` program available for download here (22K). The first method is good as far as it goes, but in the packages I have available it is difficult to get very accurate segments using the mouse and drag type of operation. You may have better success than me, so feel free to try. From now on I will refer to the `bmp` program (which is a DOS program driven from the command line). The discussion is applicable to both methods. However, using `bmp` implies a bitmap image is available, and you have to convert if it isn't. For more details on the bmp utility, refer to Appendix B.

The first step, then, is to break up the image horizontally as indicated by the lines on your drawing. With bmp this is a trial and error approach. You need to know the overall dimensions of your image. You can find this out with bmp as shown below

```
bmp -d -i calc.bmp
```

This displays bitmap header information, including width and height values (in hexadecimal).

Guessing where the first line occurs (say a quarter way down), then the top to the first cut is done as (assuming an image which is 217 wide by 356 high)

```
bmp -i calc.bmp -o top.bmp -C "0 217 271 356"
```

Check the image in top.bmp (or whatever you've called it) and see if it is the segment you want. The series of numbers mark the area to extract, with left, right, bottom and top boundaries specified. The right and top boundaries actually specify the pixel after the cut. Adjust the third parameter as necessary until the cut is made in the right place. At this point I mark on the drawing the numbers of the boundaries for reference, as well as the name of the segment used in the filename. You going to have many images at the end of this process, and it's hard to keep track. This process is repeated for all the other horizontal segments. Note that the end of one segment is the number to use as the start of the next (which is why bmp works in the way it does). So the top of the segment adjacent to 'top' in the bmp example would be 271. Our example image, after refining the cut positions, is now split horizonatlly into the three images shown below.

In general, after this process you'll have some files with no active areas and some with. The inactive files are done, and need to be put aside for later. In the example there are two inactive areas, the top image (placed in a file top.bmp) and the keys image (placed in a file keys.bmp). The middle image (display.bmp) is the sole active area. The active files now need to be split vertically. Exactly the same process is used as for horizontal splits, but the left and right parameters are changed. Don't forget that the image you're spliting vertically has a different height to the master image, which you should be able to work out from your drawing (but you can use bmp -d -i file.bmp to make sure if you want).

The display needs some special attention. You must work out the left and right segment sizes (which may be different from each other), with the remainder constituting the display proper---i.e. only the segments that contain numbers etc. Note the size of this and divide it into equal portions to match the number of display segments. For separate decimal point images, the segments are divided in two, though probably assymetrically. If you have a separate image for extracting the display figures, then now is the time to slice out a display row the same height as the master images display row. A trial and error approach to extracting numbers aligned correctly is the best I can recommend. But each segment must be the same width, and if you, say, left align the number images, they should all be pretty much identically positioned. You will also need an 'off' number segment, and an on and off decimal point segment if separate.

In the example, the display.bmp images is 217 pixels wide and 45 pixels high. The area from the left of the 8 figure (including its decimal point) to the left of the seven is about 20 pixels wide. Eight of those would be 160 pixels, so the left and right of the display is 217 - 160 = 51. If we make the left 29 pixels and the right 28 pixels wide, that will tally to 217. Spliting the display image in this way yields the following:

A similar approach is used for the switches (if there are any), getting fragments for each combination. In all of this it is best to note the filename and dimensions of the segments on your drawing (you'll thank me later).

In the image used in the example, only the 8 has a decimal point. It is very close to the 8, so we need a combined number-dot set of images, as well as dotless versions. We also need a minus, a 9 and a 0. These can all be constructed from what we have using a graphics tool---even one as simple as Microsoft paint. The dot may be copied from the 8, and pasted into the other numbers. The 9, 0 and minus can be constructed from the 8 by carefully erasing segments with colours from the LCD background. Similarly we need an 'off' version (no segments) and an error version (an 'E' constructed from the 8). In this example a naming convention is used for the filesname---lcd<val>[dot].bmp, where <val> is the value (0 to 9, minus, off, E) and the dot versions (where needed) have the 'dot' extension.

You now have all your image files. If you've used bmp, then they are all bitmaps. JPEG files tend to be smaller and thus better for the internet, and you will probably want to convert them at some stage, but bitmaps can be read by a browser and it is probably best to wait until you have it all working before doing this conversion. We'll assume from now on, though, that we have JPEG images (with a .jpg suffix).

"How to write a calculator simulator"

<- Prev Page Next Page->