Tutorial: Gauge graphics, Part 3

Text Tool

The rule for text is: Simpler is better. In flight, that can be translated as safer. Plain fonts narrow the choices down considerably. What fonts can you use for gauge graphics? That will depend upon the types installed on your machine. But some suggestions can be made as to the most desirable characteristics of a font that will produce readability down to small sizes.

A True Type font is the only font that XaraX will load. These fonts can be resized at will without losing readability. Since XaraX is a vector program, it will use only fonts suitable for vector drawing.

Avoid fancy fonts such as scripts and specialized alphabets with cartoon-like letters. Avoid designs with serifs such as Times New Roman. Stick with simpler fonts like Arial or Verdana. There are others also such as HANA, Glass Gauge, Tahoma, and Maximo. Plain lettering contributes to the clarity of a gauge, especially when used at smaller font sizes. At left, Arial Bold style is shown at 100% zoom from sizes 1-pt to 8-pts. Even though these are readable in this illustration down to 3 points, this may not be satisfactory once a gauge is installed on a panel. That is the guiding criteria regarding font size. No one enjoys using a gauge containing information that is unreadable.

Some of the standard text-handling features found in XaraX occupy the text toolbar shown above. These include all the True Type fonts available on your computer, font size, aspect ratio of letters, styles, and justification. Some features apply more to web pages than FS gauges but they are available if the need should arise.

On the right side of the above toolbar are other controls that give you a much wider range of specialized options. These are tracking, line spacing, baseline shift, and kerning. With these, text is first written, then tailored to suit almost any formatting of letters needed on a gauge.

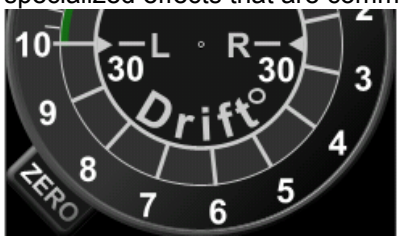
XaraX considers text as an object; therefore, object editing is available. Text can receive all editing effects including those that have already been introduced in this tutorial; beveling, transparency, shadows, re-sizing, stretching, rotation, and borders. By having both text and object editing tools available, the possibilities for getting just the right textual effect for a gauge is almost certain.

Above, a few of the editing possibilities are demonstrated. The red area has the letters stretched horizontally but shrunk vertically. In the green zone, each individual letter was selected with the Text Tool, then changed in color. A border was placed around the center letter. The gray illustrates various changes in font size, font type, and font style applied to the individual letters. Blue has increased Tracking (spacing between letter). Gold shows the Tracking applied in the opposite way. Yellow has shadows turned to blue and the white uses Linear transparency. These effects can be applied in various combinations increasing the choice for the final effect.

The only text on the Pattern Gauge is that atop each of the Heading Buttons and the letters that identifies the Power Button. The word "PWR" can be typed two ways. It can be created as a single object containing the three letters. Or each letter can be produced as a single object, and then aligned to appear as one word. The advantage to the last method is flexibility. Since each letter stands alone, they could be edited so that certain effects could be applied to only one letter such as different transparency values.

For example, the PWR could be altered so that it would wrap around the circular Power Button. That would be achieved by distorting each letter as seen here. This was done with the Mould Tool. While this effect may seem rather cool, it adds not a whit to the readability of the text.

Because of the limited amount of text on the demo gauge, no further time will be spent discussing it other than saying it was typed, colored, and sized. No special text effects were applied. Instead, focus will now shift to other specialized effects that are common to gauges. And the Mould Tool will be described further.

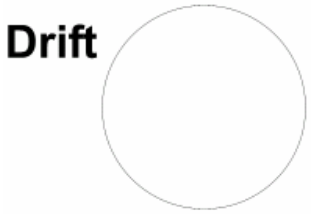


One such effect is seen in the graphic at left. The word "Drift" has be given a curve to parallel the lines around the gauge. But there is another text effect on this gauge. Can you spot it? (Answer below)

Curving text has little purpose other than that shown here. It is not Mt. Everest, you don't climb it just because it is there. You don't curve text just because you know how.

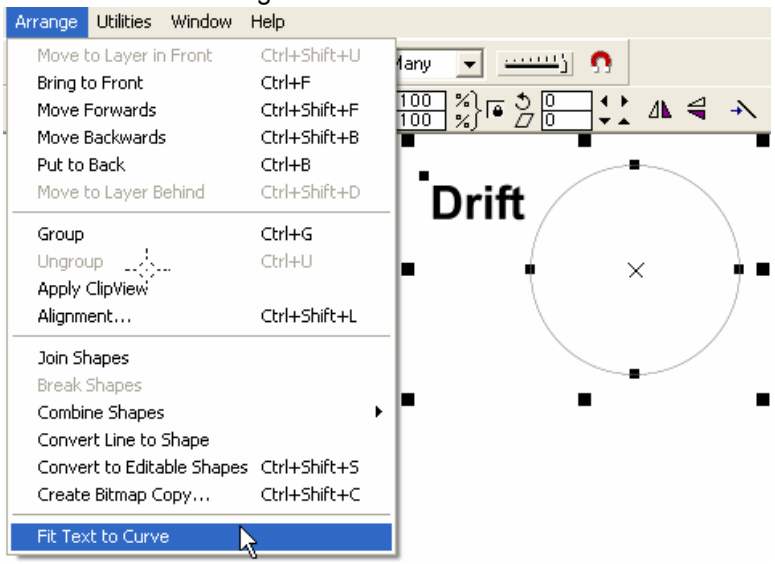
It could even be argued that the curved text at left would be just as effective if left in a straight line. This is oftentimes the case in circular gauges inside FS. It is a design decision. I will demonstrate the curving technique and you can apply judgment on its appropriateness.

The second text effect mentioned above is the Linear transparency on the word "ZERO". That produced a word partially in the shadow of the rim of the gauge.

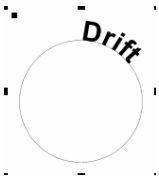


To curve text, you must first give XaraX the elements with which to work. At left, those elements are the text and the nearby controlling curve. It is a transparent circle with a thin, black border. In fact, it is a clone of the circle from the gauge illustrated above.

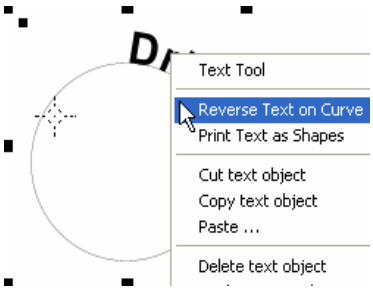
XaraX can now be instructed to work its magic on these two elements. For that, both objects must be selected. The tool chosen is found inside the Arrange/Fit Text to Curve menu shown below.



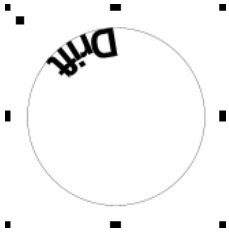
When the Fit Text tool is applied, the text will wrap to the circle. By default, the location of the text will be outside the circle (see below). This will curve the text so that center letters are higher than those on the end of the word. This is the opposite effect of what is needed in this gauge.



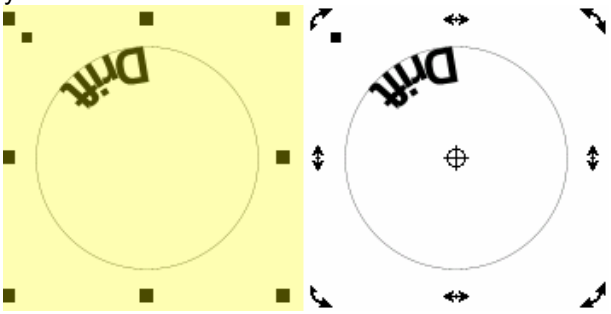
XaraX foresaw this possibility and will now make available a second tool.



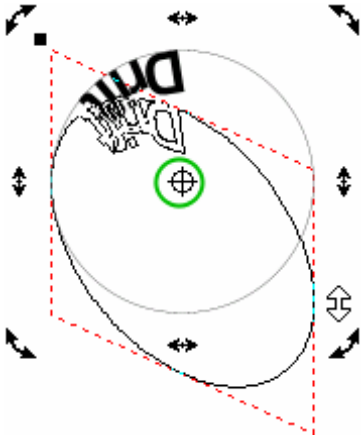
That tool is found inside a pop-up box activated by right-clicking one of the selected objects. The Reverse Text on Curve, at left, will snap the text object to the inside of the curve producing the result shown at right. Okay, the text is curved properly but it is now hanging upsidedown. Unless we want a pilot with a sore neck we must rotate the word until it is located at the bottom of the circle.



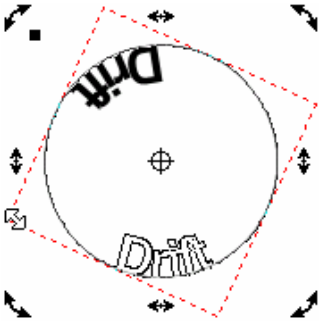
XaraX also makes this rotation easy. If you have been working along with this tutorial you will have already made the discovery on how this is done. If you have only been reading, then you don't have a clue.



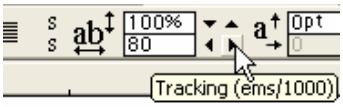
Objects go through two selection phases. The first click with the Selection Arrow activates the square handles seen in the yellow area. With these, an object can be changed in size. A second click with the Selection tool will change the handles into double-pointed arrow seen on the right object. These are rotation handles.



If a center handle is moved with the mouse, a distortion will result. That effect is see at right. The circle and text is being pulled downward. The circle is taking on an oval shape. This is not what is wanted. To rotate the text around the rotation point (inside green circle at right), a corner handle must be used.



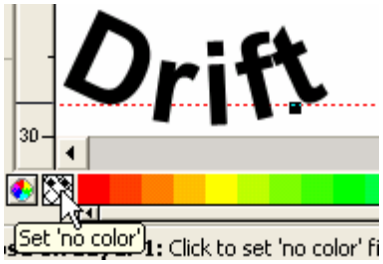
At left, the outline of the text indicates the present location of object during the rotation. That lets you monitor the progress of the edit. When the text is in the approximate position, a click of the mouse will end the edit.



When words are located on the inside of a curve, the letters have a tendency to become too crowded and overlap at the top. Such is the case here. By increasing the spacing between each letter, that condition can be corrected. The tool to use is the Tracking tool described earlier.

This tool is activated by choosing the Text Tool. At left, the Text Tool was swiped across the letters to select the entire word. By clicking the Tracking button at the top of the XaraX screen, the tracking value can be increased. As that value grows, so does the spacing between letters. The value of 80 gave a satisfactory result in the example seen at right.

At this point, chances are good that the two end letters will not be positioned evenly at the bottom of the circle. If so, a red horizontal Guide can be used as shown at right. One of the corner rotation handles can then be further adjusted until the two letters are equal in height.



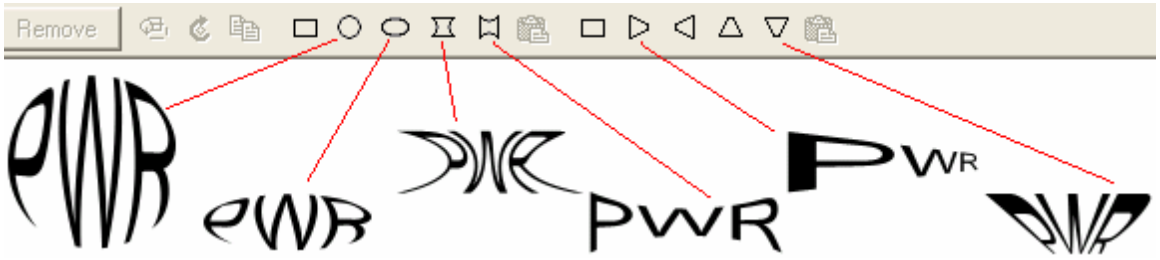
What about the circle used to produce the curved text? The element that was used was the border of that circle. The interior of that object was transparent. If the border is also made transparent, it will disappear from view. This is illustrated at left. The circle is still there, but now it is invisible.

Perhaps not as useful as the previous technique, text can also be made to follow a line as shown at right. The red line was constructed first. Then the Text cursor was clicked at the beginning of the line. XaraX then placed each letter along the line as they were typed.

Another effect that was mentioned earlier was the Mould Tool. While it is a tool you will not need often, if ever, it will be described just in case. In my version of XaraX, this tool uses two spellings for the name. The Help sections uses Mold, the toolbar shows Mould. Whatever the preferred spelling, the job of this tool is to alter object shapes.

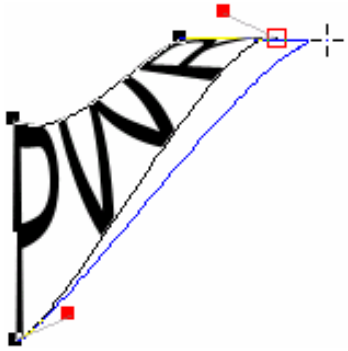
The molding process employs pre-shaped objects that control the final result. If you can imagine pressing clay (the object to be transformed) into a casting (the pre-shape object), you will get the idea. XaraX has several default shapes but custom shapes can also be created although their parameters are limited to four-sided figures.

Applying a mould is very simple. The object, text or otherwise, to be transformed is selected with the Selection Arrow, then the Mould Tool is activated in the toolbar at the left side of the XaraX screen. At the top of the screen, the Molding pre-shaped choices will then be displayed. Upon selecting one of these, the object will snap immediately into the new shape.



Above, the PWR text object has been moulded using some of the default shapes available. The result produced by these is demonstrated by the altered text. But this step does not necessarily end the process. After applying the initial mould, the objects can continue to be altered manually.

To do this, the object is first selected using the Selection Arrow. The Shape Editor Tool, shown at left, is chosen. This tool will move the molding handles of an object in limitless ways. As an example, the text at right has received additional modification. Distorting text in this manner has no place on a gauge. But, objects other than text can also be moulded. That may offer a unique way to construct unusual shapes for a gauge.



As you can see at left, the Molding cursor (arrow) can attach to the shape at any point around the perimeter and move it. The object that is inside the mould will flow to fill the new form.

Exporting XaraX graphics

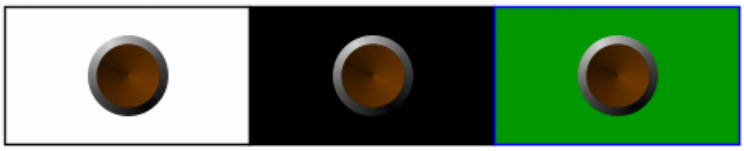
This is the last step for all graphics produced by XaraX. Once it has generated the graphics, the remainder of the gauge work will be handed over to other software more suited to final editing. In this tutorial, that software will be Paint Shop Pro. It was chosen because it contains a large amount of top-notch image editing functions. Many of these are highly adaptive to the requirements of FS. And the fact that PSP continues to have a relatively low cost and learning curve doesn't hurt. PhotoShop has tools comparable to those inside PSP and can be used to work on XaraX graphics in a similar manner. The software that is used is really unimportant as long as the final results are satisfactory.

What graphics get exported? The OFF background graphic and the ON illuminated graphic. Why and how these two images will be finalized inside PSP will be made clear as the discussion continues.

Before exporting, one last element is needed for both images. This is a 100-percent black rectangular background positioned behind each exported image.

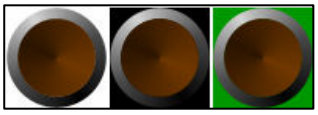
These last-minute rectangles are not needed for gauges that have square corners. But they are required for gauges that are circular or use rounded corners. The demo gauge uses the latter so a rectangle will be required. Why? It is due to an exporting characteristic of XaraX.

Tip: During an export, XaraX will construct a background behind round images that will match any color behind the image even if that color is the white of the workspace.



Let me illustrate this aspect of exporting. At left, the round Power Button is placed atop three background colors; the white of the workspace, a black rectangle, and a green rectangle. The first button is selected (the button only) and exported into PSP. The same is done with the other two buttons.

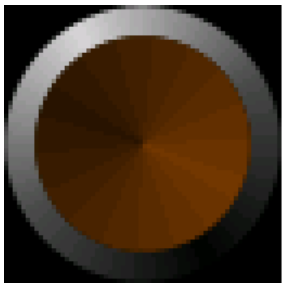
They arrive in PSP looking like the illustration at right. You can see how the corner area around each round button has taken on the background color that was behind each object during export. This is a highly desirable feature of XaraX. It allows rounded images to retain the smooth curves produced by vector graphics.



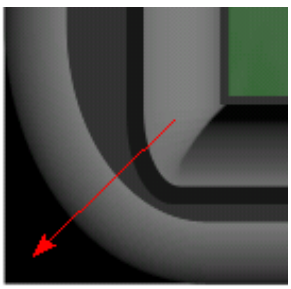
All bitmaps in FS are required to be either square or rectangular. Even the round buttons shown here must meet this requirement. But, to do so, they must be "fleshed" out from the round shape to a square shape. That requires filling in all the non-button areas with a color. In FS that color should be transparent black (0, 0, 0). When displaying this color in an aircraft panel, it will be invisible. This has the visual effect of allowing the square graphic to appear round since you can only see colors that are non-transparent.

Why not export images using a white background and apply the transparent black using tools inside PSP? This is the technique that was used in the illustration at right. The white corners were painted black using the Flood Fill Tool within PSP. The tool was set to different tolerances before each corner received its color. The yellow numbers show the tolerance value used per corner.

All white pixels were turned black. The higher tolerances allowed more of the gray pixels around the edge of the button to be colored black. But many gray pixels still remain. Forcing light colored pixels to become black when they should not or allowing light colored pixels to remain when they should be darkened are both unacceptable. Compare the PSP edited image with the image below it. The lower graphic used a transparent black background during the export into PSP. This gave the smooth edging you want on a gauge.



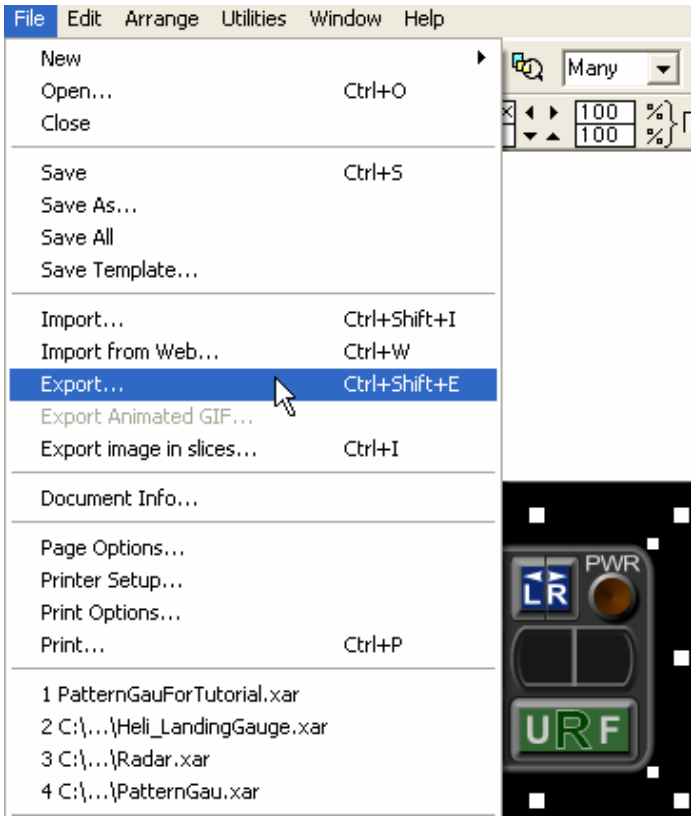
Most gauges will be located in areas of a panel that have been prepared in advance for the installation. These areas, like the one at left, are tailored to look like a cutout in the face of the panel. They are generally shaped to match the outline of a gauge, are a little larger than the gauge, and are colored dark black, but not transparent black. These dark cutouts will contrast sharply against any light colored pixel around the perimeter of a gauge causing them to stick out like a sore thumb.



At right, one corner of the demo gauge is shown. The red arrow points to the transparent black background that fills that corner. The curved rim makes the turn in a flowing fashion and has shading that looks realistic because of its smoothness. This gauge will sit inside a black cutout without lighter specks around the rim screaming for attention. Consequently, you can get on with the business of flying without listening to hoards of shrieking pixels. They sound similar to a cicada just before you step on it.

The two graphics at right are ready to be exported. They sit atop a 100% black background. Each gauge will be selected (NOT the black background) using the Selection Arrow and exported separately. That will produce two images equal in every way except for the illumination effect.

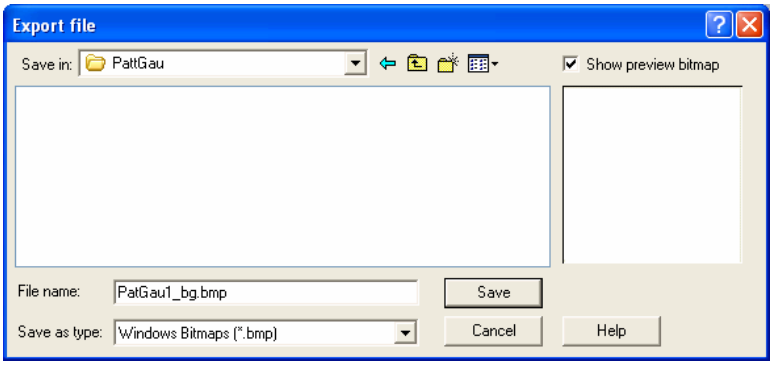
The OFF graphic will be given a file name of PatGau1_bg.bmp. FS will learn this name through the XML code. The ON graphic will be named simply On.bmp. While the OFF image will be a permanent fixture inside FS, the ON image is temporarily. It will soon undergo surgery to have the individual lighted controls removed. Each of those smaller graphics will then be finalized into a permanent fixture inside FS. The OFF graphic will furnish the background for the gauge, the ON graphics will simulate the controls as they become active. XML code will oversee these events and issue instruction to Flight Simulator when it is appropriate to "turn on" a button. It does the same when a button is turned "off".



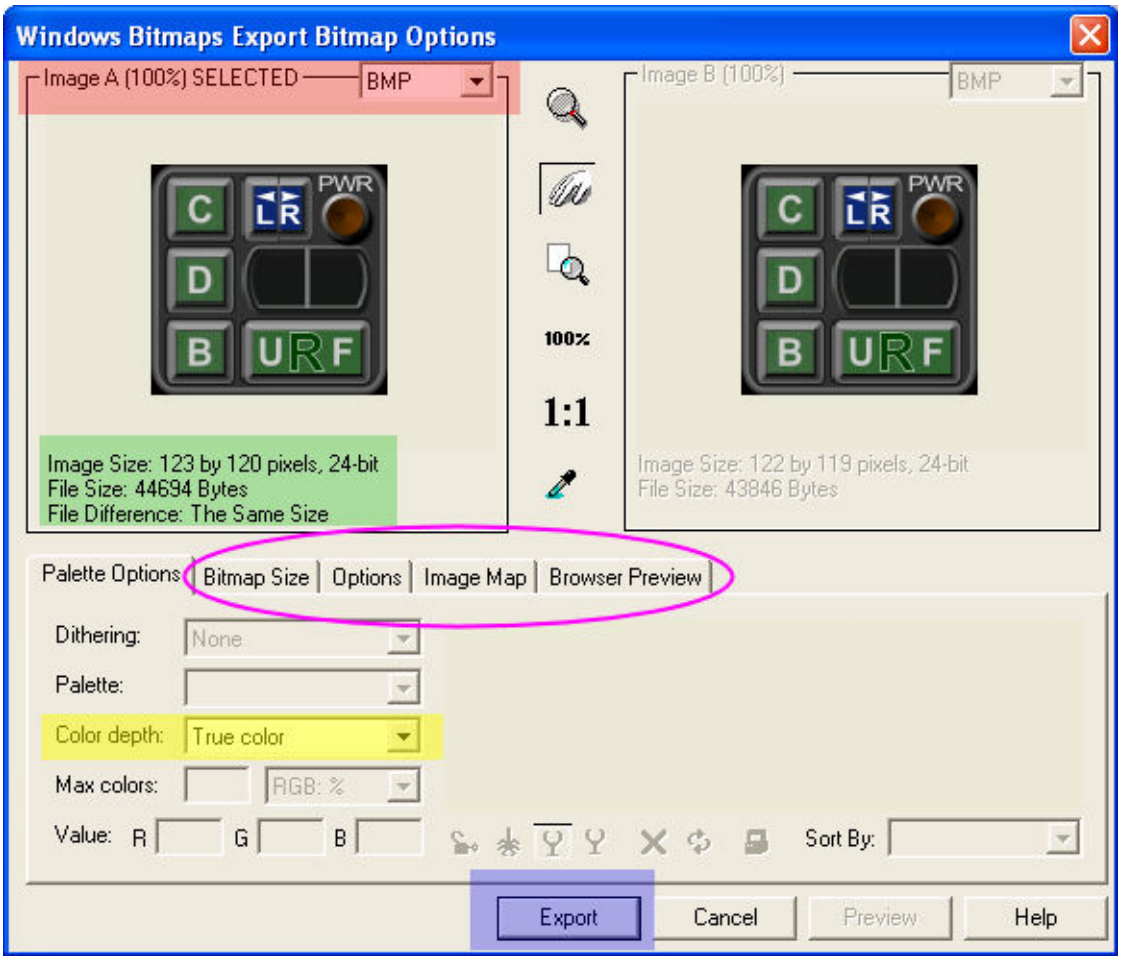
Let's export the OFF graphic to the hard drive. This is a command that will convert the vector image into a raster bitmap and save it at the same time. It is stored in the BMP format. Paint Shop Pro can open that format, and then continue to finalize the graphic.

The graphic (not the black background) is selected, and then exported using the File/Export menu shown at left.

This in turn will activate the dialog box at right. Files must be named and saved into a working folder before continuing with the export.



After saving, the window below will be activated.



The parameters for the exported graphic will be set here. Many of the options available are designed for graphics used on web pages and will not be needed for gauge graphics. This makes our job easier.

In the red area at the top of the window, the type of image file is chosen. For FS files, that will always be BMP. This file format requires fewer options to be chosen.

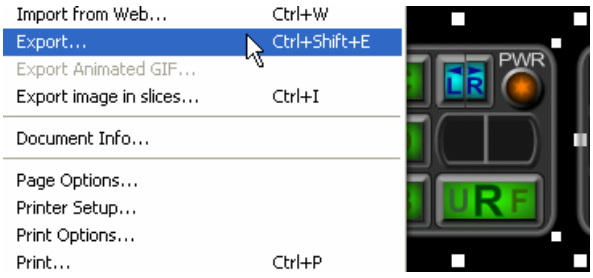
In the center of the window, the image to be exported will be shown. Below that, in the green area, the pixel size and the memory size is displayed. If you will recall, the graphic was made to a size of 123.3 x 120 pixels. In the export, the size is rounded to 123 x 120 because bitmaps can work with only whole pixels, not fractions of a pixel.

Inside the magenta oval, there are several tabs for other options. The only one that may be needed is the Bitmap Size. The others are either empty of choices when exporting a BMP, or they relate only to images for the web. The width and height of the exported graphic could be changed inside this tab, if desired. It does this a proportional manner.

The yellow area sets the Color depth choice. While FS can handle both True color and 256-colors, gauge graphics should always be placed within FS as the latter. That will help the simulator refresh rate to be as high as possible. One gauge should not hog the stage at the expense of other graphics.

However, the export from XaraX to PSP should be done using True color. The change to the lower color value will be made in PSP after all graphics are finalized. Doing it this way allows all effects and tools to remain available in PSP as the graphics are edited. This is not the case when working on a 256-color graphic in Paint Shop Pro.

And lastly, the blue Export button saves the graphic into the working folder and gives it the file name that was just chosen.

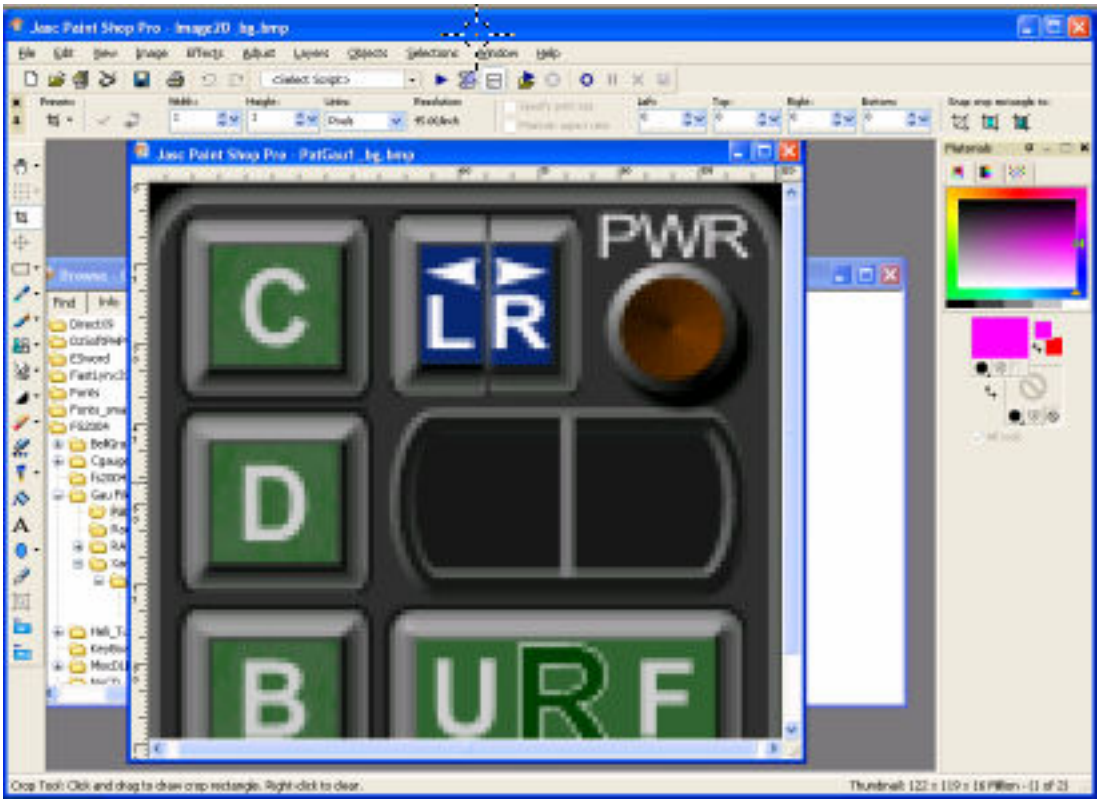


The same procedure will be repeated to export the illuminated gauge. It will also be saved in the BMP format using True color. The exported pixel size should be identical to that of the first image. But check it to be sure. If a discrepancy were present, it would indicate some element had shifted in size while working in XaraX. This could happen when you try to select an element, but select two instead. If you were not aware of this before editing the desired graphic, you could edit both graphics. That action may not show up until the export.

Preventing such a possibility is the reason this tutorial stresses the importance of grouping elements together when they are finished. These include not only controls, but also grouping the entire gauge as both the ON and OFF versions are finished.

In this case, both exported images measured the same in width and height, so let's continue. The continuation will mean a shift in software. It is now time to switch to Paint Shop Pro.

Final editing in Paint Shop Pro

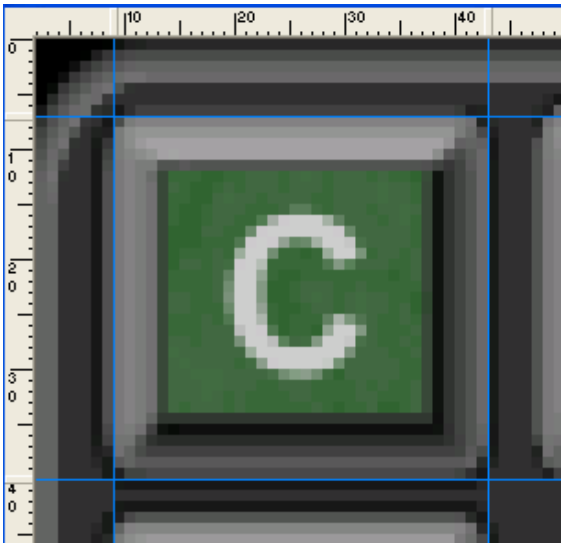


We will begin by opening the first BMP onto the workspace. Zoom in so the image can be seen clearly. PSP contains alignment guides similar to those in XaraX. This step will require that guides be positioned accurately on the background graphic. They serve three purposes.

First, these guides will define active mouse areas for this gauge. The gauge will be able to communicate with FS when the mouse is placed within one of those areas. XML code will instruct FS what the coordinates of these areas are. FS constantly monitors the mouse position. When it detects the cursor inside a mouse area, the cursor usually changes shape to give a visual indication that an input to FS is possible. For the demo gauge, the mouse areas will be each button and both input windows. The cursor will turn to a little hand in those areas.

Second, guides will locate the anchoring set of coordinates for each of the illuminated graphics. These images will require positioning directly over their non-illuminated cousin. FS positions graphics by using the top, left corner of an image. XML is the entity that will feed the coordinates of that corner to FS so it can draw the ON button in the proper spot.

Third, the guides will mark the perimeter of each button atop the OFF graphic. Guides will also be placed in identical locations atop the ON graphic. This insures each ON button equals the size of the corresponding OFF button.

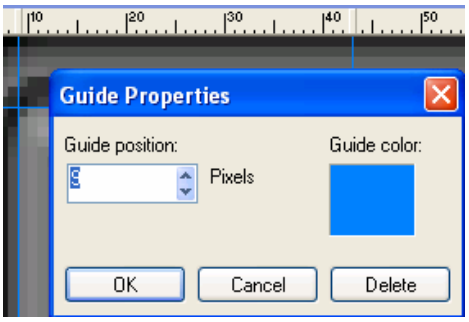


The objective during guide placement is to define the outer pixels that form a control. This requires two horizontal guides and two vertical guides.

At left, the blue guides show this placement around the Crosswind Button. As you can see, they touch the last row of pixels around all four sides of the button. This is not supremely critical. A pixel or two would not be catastrophic. For example, the left guide could be moved one pixel to the left to include the last row of gray pixels. But it does not matter because the matching illuminated button is defined by a left guide in an identical position.

One important point must be followed, however. Guides must be located outside those areas that will be illuminated. In this illustration, that would be the green pixels. It is not always clear what pixels will "light up" when placing guides around an OFF button so it is best to keep outside that area.

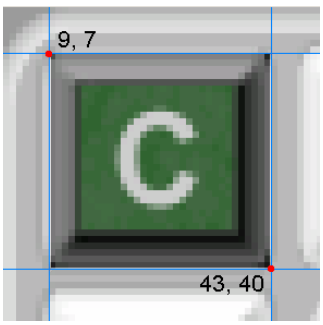
The blue guides define the mouse area. On the operational gauge, any pixel within that area will activate a button when the mouse is clicked. The guides can furnish the mouse coordinates for an area because areas have an X and Y value. These can be read on the rulers along the top and left side of the PSP screen. Even better, right-clicking a guide handle will activate a window that displays these values. Guide handles are the thicker, raised areas showing on the rulers. One can be seen inside the green oval at right.



In the illustration at left, the guide on the left side of the button was clicked. The Guide Properties box shows that this guide is located at pixel number 9. This is an X-value because we are reading from a vertical guide. Zero value for both X and Y is located at the top, left corner of any graphic.

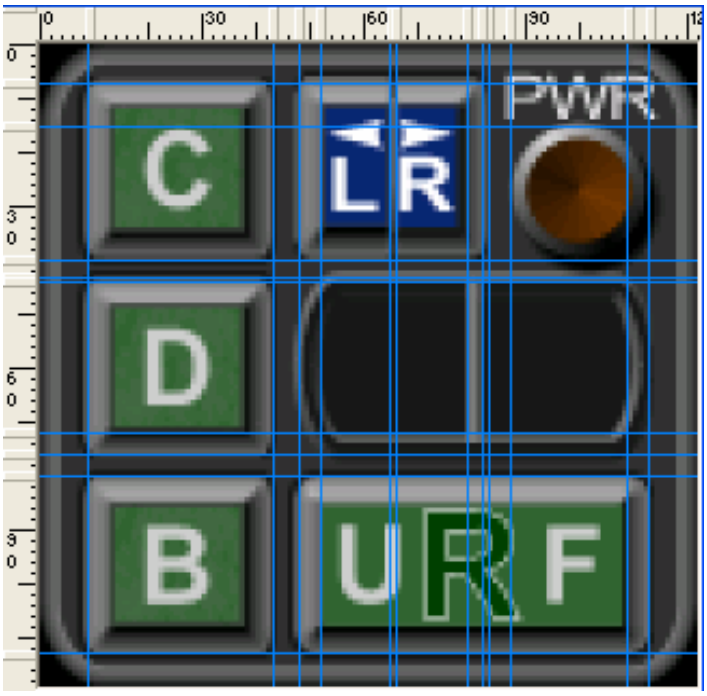
Also, note the Guide position has the value in focus by default. That allows a number to be typed into the box and Entered. Doing so will relocate the guide to the new value. This allows guides to be positioned easily when the need arises.

At right, the coordinate for two intersections is shown. The red dots define those points. The top left corner has a coordinate value of X= 9, Y=7. The opposite corner reads 43, 40. From the value of these two corners, the pixel width and height of the enclosed button can be calculated. The size of the enclosed area measures 34 pixels wide by 33 pixels high.

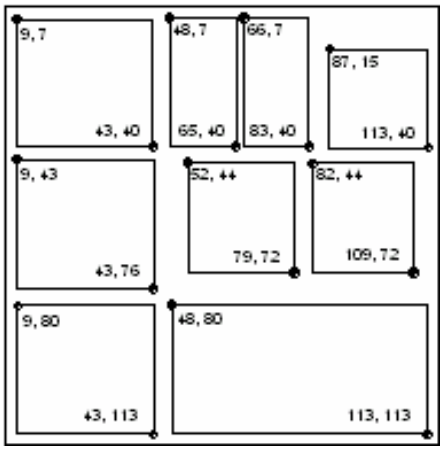


From these two sets of coordinates, we have determined two of the parameters necessary for communications between XML and FS: one, the coordinates for the mouse area; two, the anchor position for the illuminated version of the Crosswind Button. That anchor is located at the 9, 7 corner.

All buttons plus the two windows require these two sets of X-Y coordinates. The entire setup for every area is shown at right. But, with so many lines running across the graphic, it would be too easy to get a pair of coordinates wrong. It is best to work one button or one row at a time. How you arrive at the numbers is not important. What is important is that the value of every necessary intersection be recorded for future use. Guide placements for the OFF image will determine guide placement for the ON graphic.

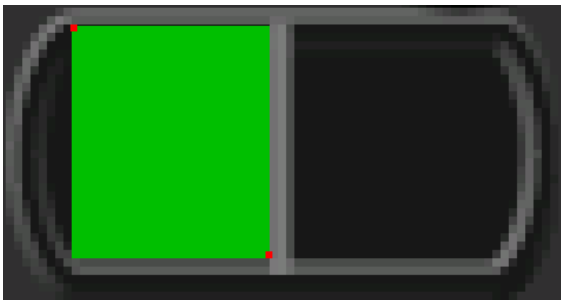


A diagram such as the one below is a handy way to record the intersection of guides. Plus, it is easy to identify the element to which each set of coordinates belongs.



The coordinates needed for the two windows are determined a little differently. For those, values are needed for the interior portion instead of the outer perimeter. These elements do not use graphics like the lighted buttons, but rely on XML's ability to print digits. But, we must inform XML where to print those digits, and that requires coordinates.

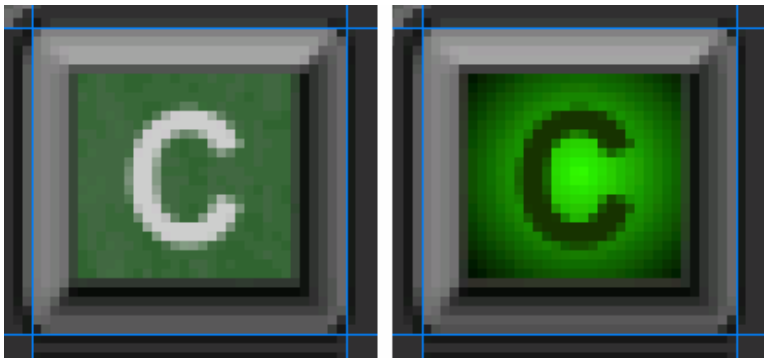
In the picture at right, the green area would define the limits for a digit inside the left window. Coordinates are measured at the two corners dotted in red. The right window would be done in a similar way. The printable area does not extend all the way to the curved ends of the windows because you do not want digits printing on the rim. They should be centered within the coordinates of the green region. The window coordinates are used by the XML code to print each digit in the correct location.



After locating guides atop the OFF background image and recording all necessary coordinates for each button, attention can shift to finalizing graphics for the lighted controls.

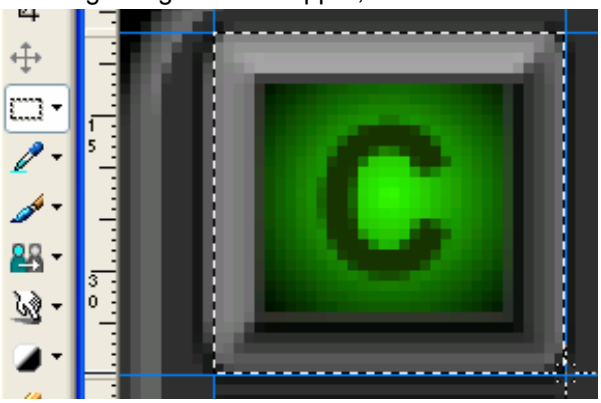
We have two gauges, the ON and OFF version. Each button on the two graphics occupies the same location because the two gauges are cloned copies. Therefore, the illuminated buttons will have coordinates that are identical to those of the non-illuminated buttons.

Let's begin this operation by locating guides atop the ON graphic using the previously determined set of coordinates.

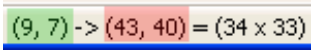


At right, four blue guides are shown positioned around both buttons. A comparison can be made that will confirm both areas inside the guides are identical except for the lighted portion.

To fashion the lighted button, draw a selection rectangle using the Selection Tool, as shown below. The rectangle begins at the upper, left intersection and stops at the lower, right intersection.

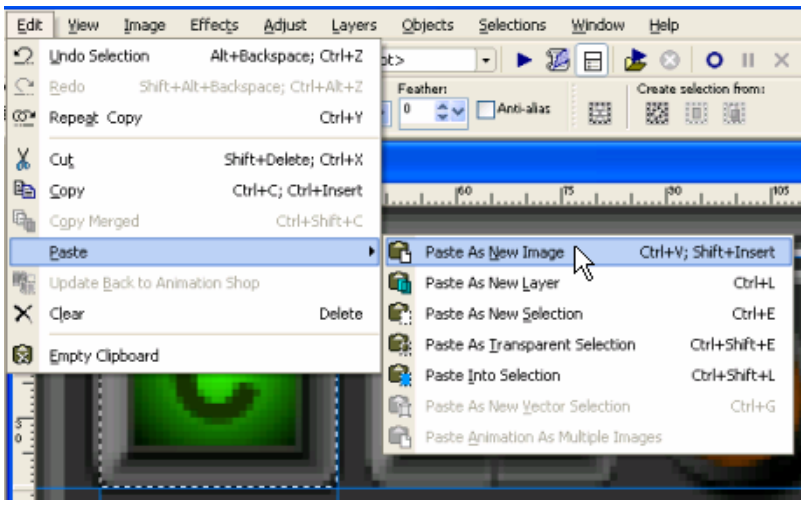


In PSP, the position of the cursor can be seen in the bottom, right corner of the screen. These values can be used to double check that the guide coordinates are being followed.

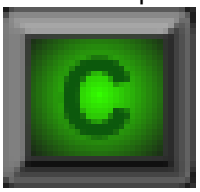


Above, the green area shows the beginning X, Y. The red shows the ending. Those cursor values match the set of coordinates for the Crosswind Button.

The selection is copied, then "Paste As New Image". In the illustration below, the choices for pasting are shown inside the drop-down menu.



The graphic that is pasted now becomes the illuminated version of the Crosswind Button. That image is shown at the bottom of the page. It is given a file name and saved inside the working folder using 16-million colors. The color depth for all illuminated controls will be lowered to 256 colors in a later step.

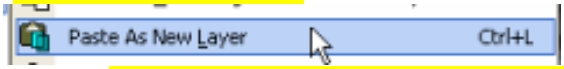


Guides are placed around all remaining buttons, one at a time, and the copy/Paste step is repeated until all illuminated bitmaps have been saved.

It is suggested that the ON buttons be checked to make sure no errors were made in setting up guides or when selecting the illuminated buttons. There is a fast and easy way to do that. This check is performed by PSP without having a test gauge or running FS.

Tip: Use the following method to check the size and alignment of an ON button when it is placed above its OFF counterpart. This technique will simulate the button turning on and off quickly. The sudden switch from one graphic to another allows the eye to detect whether the transition is smooth.

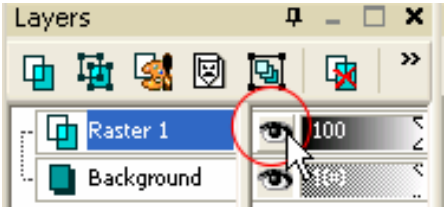
In previous steps, the illuminated button was selected, copied, and then pasted to form a new image. This image became the new version of the Crosswind Button.



While the image is still on the Clipboard, it can be Pasted As New Layer onto the OFF background bitmap. The illustration at left shows the Crosswind button pasted and centered between the guides. The ON button occupies the top layer; the OFF button occupies the bottom layer. To see the OFF button, the top layer must be turned off. To check the alignment between the two buttons, the layer must be toggled on/off quickly. That will simulate the same on/off effect that will eventually take place when the gauge is installed and operational inside an aircraft.

To control the visibility of the top layer, the Layer palette is activated. This is done inside the View/Palettes/Layer menu.

At right, this palette shows the eye icon inside the red circle. A click on that tool will toggle the selected layer on and off. In this case, the selected layer is the top layer containing the ON button.

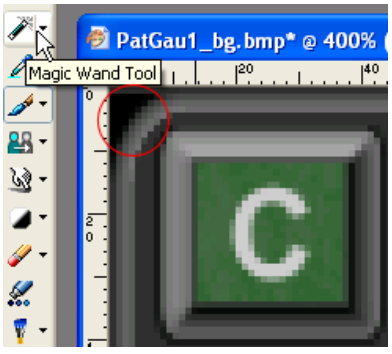


Zoom the button to a large size to see this effect better. Toggle the top layer on and off rapidly. This will allow your eye to detect any discrepancies in alignment. You are looking for a jerky movement as the illuminated graphic first appears. If there is misalignment, guide positions and copy/paste techniques need to be double-checked. Trying to adjust a bad alignment by altering the ON graphic in size or position usually does not produce satisfactory results.

After all illuminated graphics have been made and saved; it is time to check all graphics for the use of transparent black; 0, 0, 0. The black value of 1, 1, 1 will be discussed later. Pixels found using transparent black **in the interior** of the gauge will be edited to another value of black. Transparent pixels should be expected **in the exterior** of this gauge (around the curved corners) because they were placed there intentionally.

To edit the reserved black, I like to use 10, 10, 10. That value is sufficiently dark to appear as black and will not change in value when graphics are reduced in depth to 256 colors.

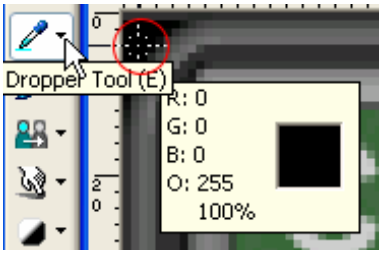
How does the reserve black appear in a gauge if none was used when creating graphics in XaraX? It may happen as a vector graphic is transformed into a BMP. That process uses an algorithm that causes the software to make color decisions as it tries to reproduce a vector drawing into a raster drawing. The choices it makes in darker areas could result in some pixels being assigned the reserved black colors. That's my guess, anyway.



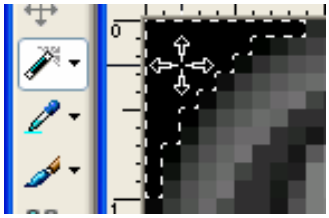
Each graphic that will become a part of your flight simulator gauge must be checked and edited, if needed. Paint Shop Pro has a number of tools that will ease the task of finding the rogue pixels and changing them into productive citizens.

Let's start this process with the background image. The first task is to find the reserved color on the graphic. Those pixels are then selected with the Magic Wand. This tool is shown at left.

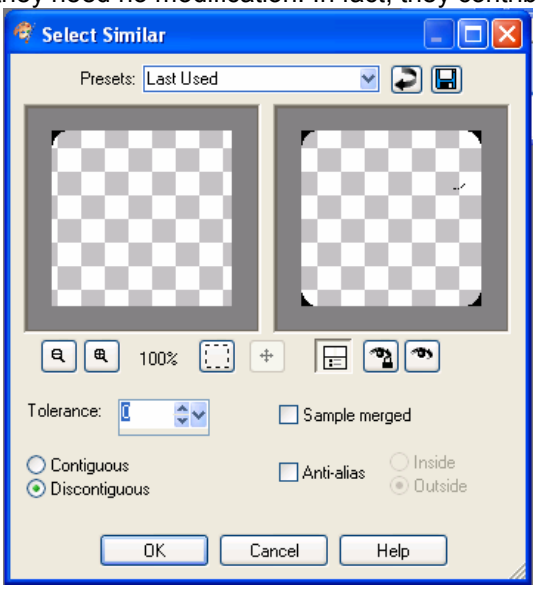
If you remember, the area (red circle) just outside the curved corners of this gauge was colored with transparent black in XaraX. Let's check the value of that area in PSP.



The color or those pixels can be confirmed with the Dropper Tool at left. The cursor (inside red circle) is placed in the black region outside the body of the gauge. The info box will show the color values. As you can see, the black is indeed transparent. That area will be selected using the Magic Wand. The tolerance for this tool is set to 0 to eliminate any shade of black other than transparent from being selected.



At right, the transparent black can be seen with the irregular selection (white dashed line) surrounding the area. If you look closely, you will see a few black pixels directly against the rim of the gauge that are not selected. Those are not pure black. Setting the tolerance of the Magic Wand to zero prevented these pixels from being selected. Since they are not reserved blacks, they need no modification. In fact, they contribute visually to maintaining the smooth corners.

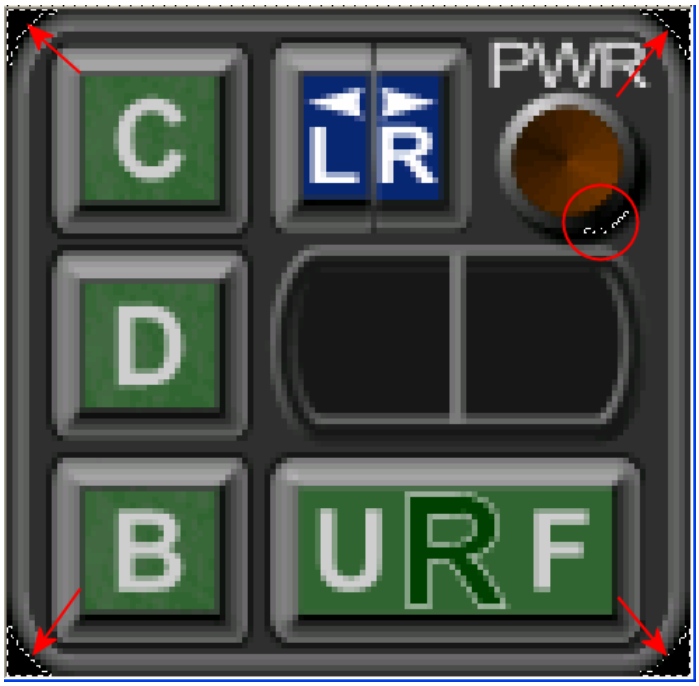


With pure black pixels selected, you can instruct PSP to find all other pixels that have the same value. It will search the entire graphic and add any qualifying pixels to the selection. The tool for this is found in Selections/Modify/Select Similar. That will activate the window at left.

Set Tolerance to 0 and choose the Discontiguous radio button. That will allow all pockets of pixels anywhere on the graphic to be added to the selection if they meet the 100% black criteria.

The right window of this dialog box displays the pixels that were found during the search. Click OK to display the result on the background bitmap.

Those results are shown below. The red arrows show all corners were found. This was expected. Inside the red circle is a shadow area under the Power Button. That also contains transparent black. This was not expected. To locate unknown areas such as this is why all graphics need to be checked.

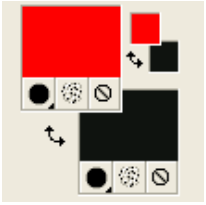


If this area were left transparent, what would happen? The clear pixels would allow any color lying below them to be seen. That could be visually distracting, but not necessarily.

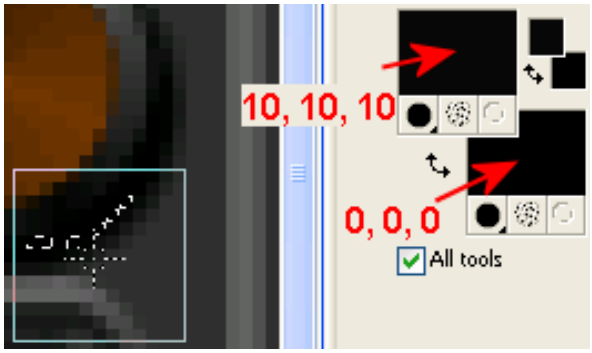
For example, if the gauge was located inside a panel cutout which itself was non-transparent black; the eye would see the cutout through the transparent pixels. If these were located in a dark shadow, the eye would not be aware of the bleed-through.

But there is a potential for a problem and for this reason I suggest removing all transparent areas within the body of a gauge. An easy way to accomplish this task will be the next topic.

Another great tool in PSP is the Color Replacer, shown at left. Its job is to exchange one color for another, just the task needed to remove transparent black. In my version of PSP this tool is located with the Eyedropper Tool. Other versions have it located differently.



The Color Replacer requires two colors to be set before employing the tool. Those colors are the Foreground and the Background. In the example at right, the Foreground is the top color patch (red) and the Background (black) is below it. When set as shown, the Color Replacer will turn all black pixels (within a set tolerance) into red pixels. In other words, it replaces the Background color for the Foreground color. The color change is accomplished by painting areas with the tool in a way that is similar to using the Paint Brush. To replace color on an entire graphic, double-click anywhere on the image. That will perform the color exchange in an instant.

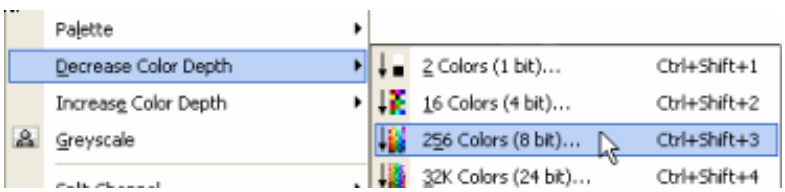


To remove unwanted transparency, the Background color is set to 100% black. The Foreground color is set to any shade of black that is not reserved. As mentioned previously, I use 10, 10, 10. This setup is shown at right. The Color Replacer is set to cover a size of about 25 pixels square. The rectangular shape seen at right represents the covered area. A double-click of the mouse is not used here. That would change all selected black areas, including those at each rounded corner. That is not desired. Instead, the Color Replace is brushed over the group of selected pixels below the Power Button. Only pixels inside the selection will have their colors altered.

The Paint Brush Tool could also be used for this job. It too would be set to the 10, 10, 10 color and to a size similar to the Color Replacer. Because only selected pixels would be affected by the brush, the other colors surrounding that area would remain unaffected.

What about the second reserved black (1, 1, 1)? Should a check be made for that color also? This black is reserved for use by XML to produce masks. These are black graphics that partially cover those elements of a gauge that should not be visible. Such would be the case at left. The needle (circled in red) rotates around the face of this gauge. It uses a pivot point located inside the green circle. A major part of this needle simulates being covered by the face of the gauge. Only the tip of the needle is visible. To create that illusion in FS a round mask was created. That mask blocked out (made invisible) the portion of the needle that should not be seen. Other than that, the mask affected no other part of this gauge.

In order for XML to use black (1, 1, 1), it must have coded instructions that it can follow when informing FS how to draw the mask. Without such instructions, mask black is unknown to XML. Therefore, it will not cause visual problems in a gauge. No check is needed for this black.



One final task still remains after all graphics, including the illuminated buttons, are checked for transparent black. When you are satisfied that all graphics are "clean", each can be saved using 256-colors. This feature is found inside the Image menu. The decrease in color depth will reduce the video burden for flight simulator without affecting the visual quality of the gauges to an unacceptable degree.

This completes the design and preparation for all graphics used by the Pattern Gauge. The tutorial will now focus on specialized graphics that may be needed for certain effects.

Graphic techniques for special effects

The first effect will demonstrate how to apply a texture to the body of a gauge. This texture could also be given to other parts such as buttons. But these elements are often too small to benefit. The only purpose of a texture is to add an additional level of visual enhancement. If overdone, it will quickly detract from the gauge.

At left the body of the Pattern Gauge was given a "Sandstone" pattern using Paint Shop Pro. This is a texture found within an Effects tool called Texturizer. It is a standard feature in PhotoShop but can also be placed into PSP. Paint Shop Pro is highly compatible with those file types.

Most readers will not have the Texturizer tool. For those, a Sandstone pattern is included with this archive. That file is named "Sandstone.bmp".

The Sandstone texture is about the only useful pattern I have found for gauges. Most look out of place. For example, who would consider having an instrument with tiny bricks covering the surface. Think how much weight that would add and increase takeoff rolls.

The technique to produce the textured effect will be called Vector-Bitmap-Vector (VBV). It begins with a vector drawing inside XaraX. The part of the gauge you wish to texture is exported to PSP as a bitmap drawing. There the texture is applied. Then, the bitmap is imported back into XaraX where it is fitted onto the body of the gauge. The final gauge body will be a graphic composed of both a vector image and a bitmap image.

Before demonstrating this technique, let's briefly discuss texture size versus graphic size.

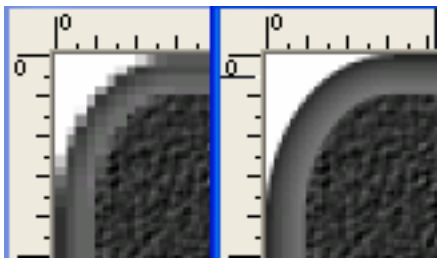
The size of any given image may or may not be suitable for texturing. An image that is too small may cause an applied texture to look too large. The correct image size can be determined only by trial and error. The texture is applied, then a judgment is made on its appropriateness. The process is repeated until satisfaction sets in.

At left, the background graphic for the Pattern Gauge was exported in its original size (123 x 120 pixels). The Sandstone was then applied at its smallest scale. You can see that the size of the wrinkled surface is not in proportion to the gauge. It resembles bumps, not subtle texture. What to do?

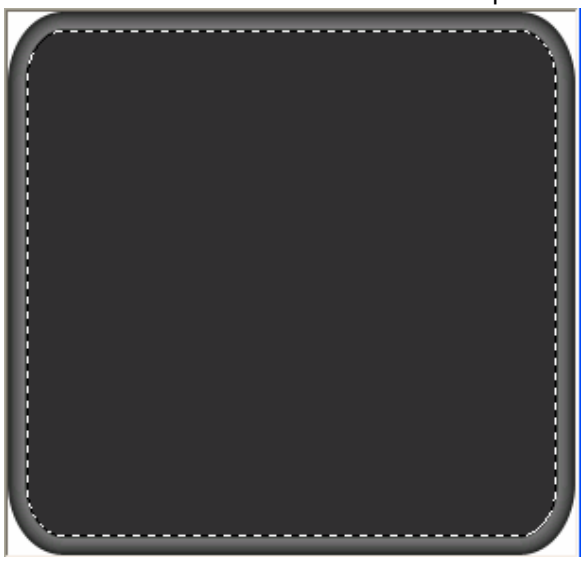
Inside XaraX, increase the size of the vector drawing. Export again and re-apply the texture. When it looks like the illustration at the top of this page, then you have a keeper. The size of that graphic is 303 x 294.

Why go through all the exporting rigmarole? Why not re-size the exported graphic to larger dimensions using PSP? After all, it has the Image/Resize function for exactly that purpose.

That is the method used in the next illustration. The image on the left was enlarged using PSP. The right side shows the XaraX version of enlargement. Compare the curved corners and you will understand the advantage of resizing using a vector drawing program instead of bitmap software. There are suitable tools and there are make-do tools. Sometimes XaraX is the tool of choice, sometimes it is PSP. There is often an overlap in abilities between these applications. If given a choice, choose vectors.



To illustrate the texturing technique, the body of the Pattern Gauge will be exported into Paint Shop Pro. For this demonstration, the larger image known to produce textures at the correct scale will be used and the trial and error phase will be summarily skipped.

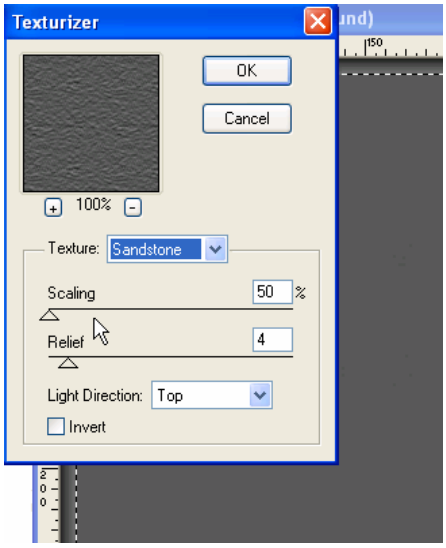


At left, the image sits on the PSP screen. As you can see, there was no black background exported with this image. The corners contain the white of the XaraX workspace. These corners will be edited after returning to XaraX, so that does not matter.

The Magic Wand was used to select the interior portion of the body. The rim will receive no texture. The tolerance of the Magic Wand may have to be adjusted in order to select all desired pixels near the edge of the rim. The Sandstone texture can now be applied to the selected area.

The first example will show the texture applied using the Texturizer inside PSP. The second example will show how to use XaraX and the Sandstone.bmp file to do essentially the same thing.

The Texturizer is a tool found inside the Effects/Plug-in menu. This will activate the window shown at the right. The options are Texture, Scaling, Relief, and Light Direction. The Scaling and Relief are adjusted until the texture inside the small window looks correct. Clicking OK will apply the Sandstone to the selected area of the bitmap.



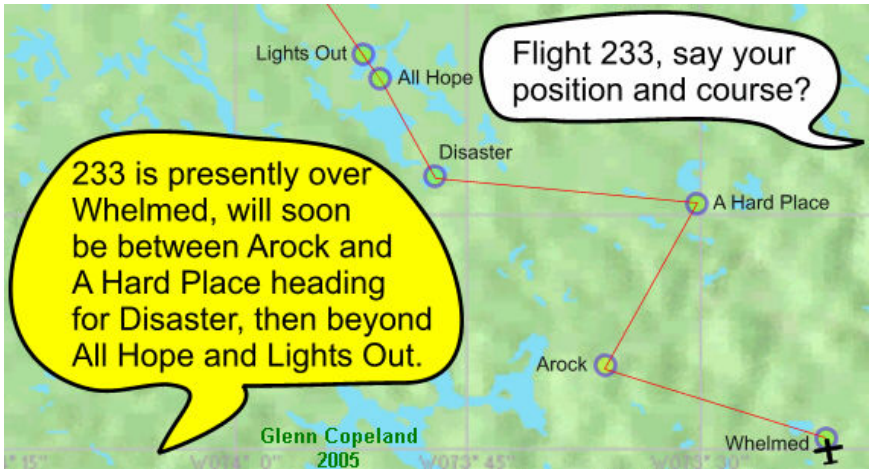
Once applied the selection is copied to the Clipboard. Back inside XaraX, paste the contents of the Clipboard into the workspace as a device-independent bitmap. You will be allowed that choice from a popup box in XaraX as soon as you activate the Edit/Paste menu.

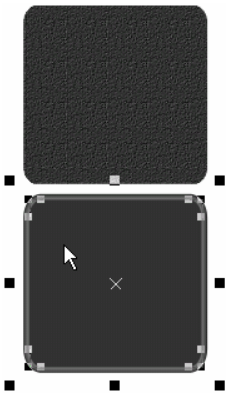
At lower right, the pasted image is seen inside XaraX. The perimeter of the bitmap is outlined in gray to illustrate that square corners now occupy the graphic, complements of PSP. The corners need to be trimmed back to their original curved shape so the image will fit properly within the rim of the gauge. Otherwise, the white corners would overlap.

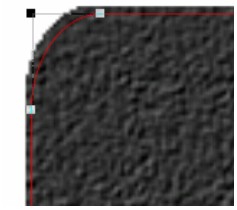
At this stage, we are working with a raster image that will be fitted over a vector image. In effect, the combination would become the body of a gauge. Once they are grouped together, who will be the wiser?

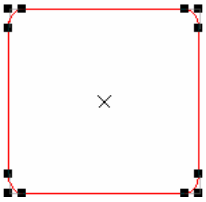
Trimming the corners may bring back memories of a technique introduced on previous pages. This was subtraction of one object from another. Perhaps you are now wishing that more attention had been paid to that discussion instead of nodding off.

Not to worry, this time we won't be subtracting objects, but slicing them. The two are similar processes that XaraX makes available for altering multiple objects to form a new, single object. Refer back to Part 1, page 7 for a quick refresher comparing subtraction, intersecting, and slicing objects.







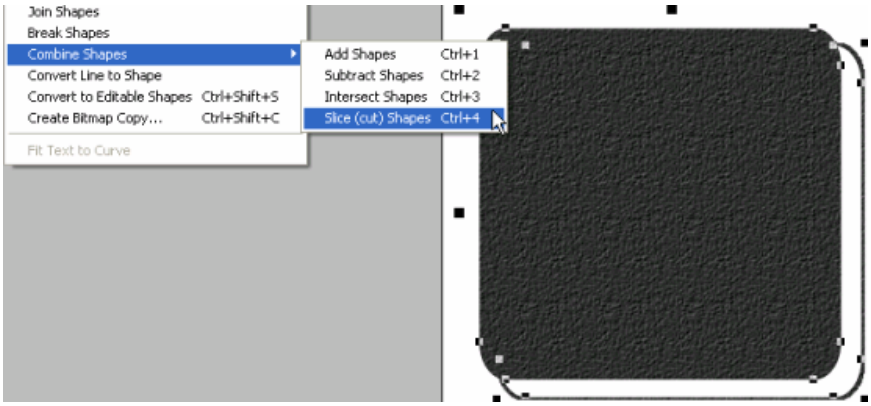


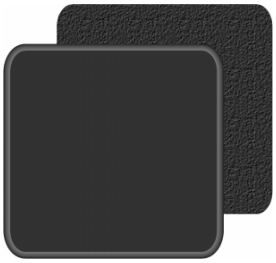
At left, the top image is the textured bitmap. The bottom image is a clone of the body of the gauge, a vector drawing. The clone will be selected and have the beveled edge (the rim) removed. The interior color will be turned transparent. The image will be given a border having a width of 0.1 pixels. The border will be red so that it can be easily distinguished from the dark bitmap. The clone will be positioned atop the bitmap and be sized (proportionately) so that it slices a very narrow edge around that object, including the square corners. This trimming will produce a final bitmap that is sized and shaped (curved corners) to fit perfectly within the body of the gauge. Why? Because a clone of the gauge is used to do the slicing.

At right, the clone (red) is shown stripped of the rim and interior color. Note the curved corners. Those were inherited from the body of the gauge. It is the cloning technique that makes this slicing process so accurate.

In the enlarged graphic at left, the red clone is positioned over the textured bitmap. It is resized until it fits just inside the edges of the bitmap. When resizing takes place, ONLY use proportional adjustments. The vector image is not allowed to change in its width-to-height ratio.

Both images are selected to begin the trimming process. The tool is located inside the Arrange/Combine Shapes/Slice Shapes menu. In the illustration at right, the slicing operation has produced two separate parts. They have been separated a short distance so they can be seen. The solid image is the new bitmap trimmed to shape by the red clone. The outline image is the second part of the slice. Though not visible against the white page, it contains the white corners and can be discarded.

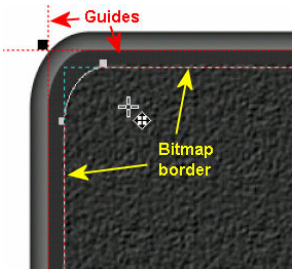




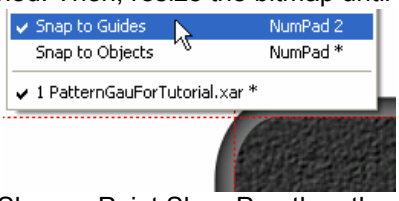
In the drawing at left, the body of the gauge and the trimmed bitmap are shown. You will notice that one is above the other. XaraX controls the drawing order of bitmaps the same way it does vector objects. We need the bitmap to be above the body of the gauge. It is selected, and then moved forward using the "Bring to Front" tool inside the Arrange menu.

The last step is to adjust the bitmap in size until it rests just inside the rim around the entire perimeter of the gauge. The picture at right shows that the colors of both objects blend making this alignment task difficult. If you look closely, the edge of the textured bitmap can barely be seen.

But, this is XaraX and we have aids to make our job easier. For the final alignment, we will use Guides. They are positioned around the interior of the rim around all four sides of the gauge.



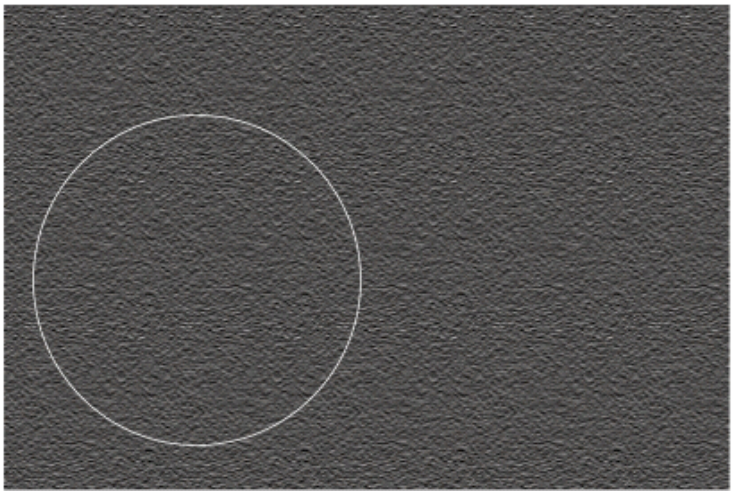
At left, the bitmap is being moved into position. As it moves, XaraX outlines the graphic in a dashed line. This makes it clear when the bitmap is directly over the Guides. More assistance is available in the Window/Snap to Guides menu. That tool will cause the bitmap to snap to the Guide. First, one corner is aligned. Then, resize the bitmap until the opposite corner is in position.



The last step in the texturing technique is to group the vector and bitmap together. Once they are aligned, you want no accidental change in the position of either object.

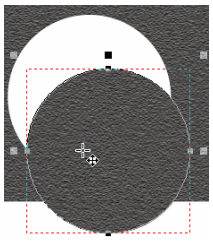
If you do not have access to the Texturizer inside PhotoShop or Paint Shop Pro, then the following technique can be used to impart bitmaps with a Sandstone texture. This method requires importing the Sandstone.bmp (included in this zip archive) into XaraX, trimming it to the required shape of the gauge, and then fitting the bitmap over the body of the gauge. As you can see, this texturing method is very similar to the last.

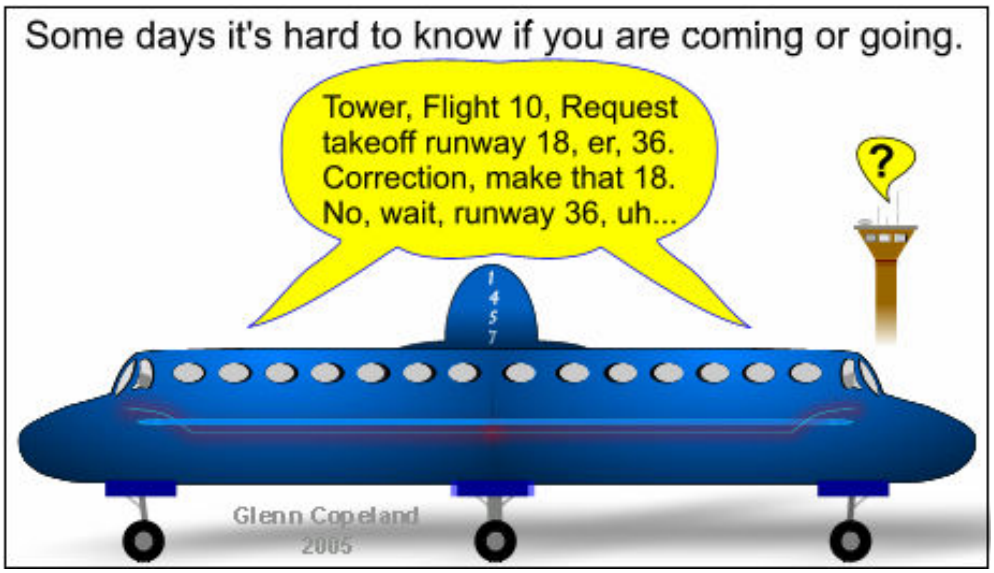
First, the Sandstone.bmp is brought into XaraX using the File/Import menu. A clone of the gauge body is also needed. It will become the slicing shape after removing any effects such as beveled edges, etc. All that is needed is the thin border of the gauge. For this illustration, I will make the gauge body round.



At left, the imported bitmap and the clone of a round gauge are shown. Either can be adjusted proportionally until the texture size looks appropriate to the gauge.

To slice out a circular shape requires both objects to be selected. Then, Arrange/Combine Shapes/Slice will perform the cut leaving the two objects shown below.





The remaining portion of the textured bitmap is not longer needed and can be discarded. The second object is the keeper. This is the round, textured element that would be fitted to the body of a fictitious, round gauge. This fitting operation would follow the same steps as outlined in the first texturing method.

Texturing is a interesting process. It may have application on larger gauges, but it is doubtful that the technique should be used on smaller gauges that are already struggling with clarity problems.

Part 4 will complete this tutorial. It will begin with a discussion of radial ticks. Also covered are panel cutouts, screws and bolts, gauge needles, plus other helpful information.