Windows, Viewports, and Rectangles

SUIT uses four different types of rectangles: SUIT_windows, SUIT_viewports, GP_rectangles, and rectangles. This section defines each of these data types and explains how they are related. Understanding the difference between these rectangles starts with understanding the way SUIT draws graphics.

If you already understand the process of window to viewport mapping, and you just need quick definitions of the terms, turn to page 20 for a summary.

The Case Against Pixel Based Drawing

Suppose you wanted to draw graphics on a bitmapped display. Probably the simplest way of doing this is to specify the pixel positions you want to light up. For example, you might easily envision a line drawing routine that looks like: "line(10, 10, 30 30)" which would draw a line from pixel (10, 10) to pixel (30, 30). There are at least two problems with this:

Not very portable: Graphics systems vary widely in resolution. If you specified lines that went from pixel (0, 0) to pixel (800, 800) such lines would run off the screen on a VGA equipped PC. Furthermore, a 600x400 image fills a Macintosh screen, but an image with the same pixel dimensions on a Sun Unix workstation approximately fills only the lower left hand corner of the screen.

Resize makes for blocky graphics: If a picture is displayed using pixel coordinates, and the pixel is scaled up to twice its size, the only real choice the software has is "pixel replication", whereby each pixel is replaced by a block two pixels wide and two pixels high. This leads to blocky, choppy pictures.

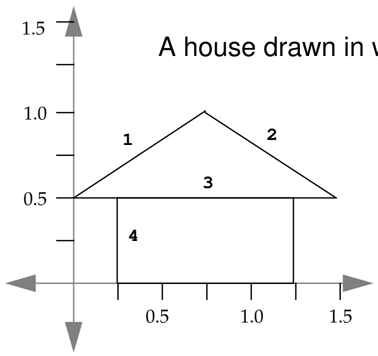
The Solution: World Coordinates

To avoid the problems of portability and image quality that a pixel based drawing system can have, we need a way to draw graphics that doesn't depend on the size or resolution of the screen. SUIT meets this need by avoiding drawing in pixel coordinates whenever possible; instead of an integer (pixel) based coordinate system, SUIT uses a *floating point* coordinate system called the *world coordinate system*.

The best way to picture the world coordinate system is to imagine that you are drawing the graphics on a large piece of graph paper, somewhere off screen. You can choose the relative sizes and locations of everything you draw using floating point numbers, without needing to be bothered about exactly which pixels need to be turned on in order to display your picture. The commands you use to create these pictures will come from SUIT's graphics library, a package called "GP." There are GP_lines, GP_ellipses, GP_rectangles and the like, all specified with floating point, rather than pixel coordinates for their parameters.

For example, here is the code for drawing a house:

```
GP_lineCoord (0.0, 0.5, 0.75, 1.0);
GP_lineCoord (0.75, 1.0, 1.5, 0.5);
GP_lineCoord (1.5, 0.5, 0.0, 0.5);
GP_rectangleCoord(0.25, 0.0, 1.25, 0.5);
```



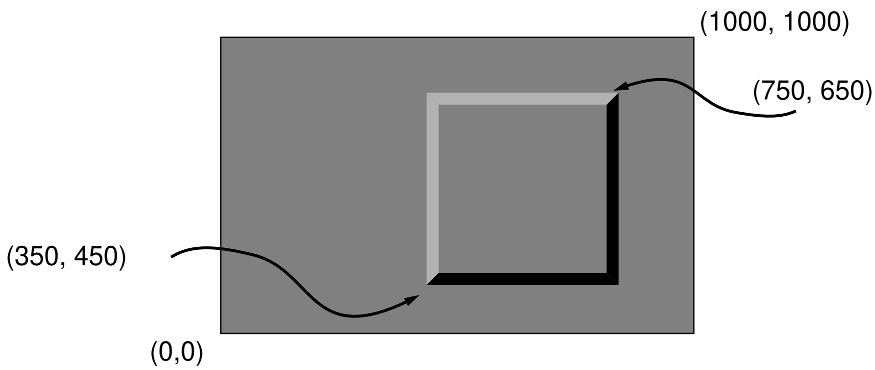
A house drawn in world coordinates.

The world coordinates are infinite in all directions.

The last entity drawn in this example is a *GP_rectangle,* which is just a rectangle defined by two pairs of floating point coordinates, one for the lower left hand corner, one for upper right. Notice that this coordinate system says nothing about where on the screen the house is to appear. Placing GP graphics on the screen is done by choosing a SUIT viewport.

SUIT_viewports

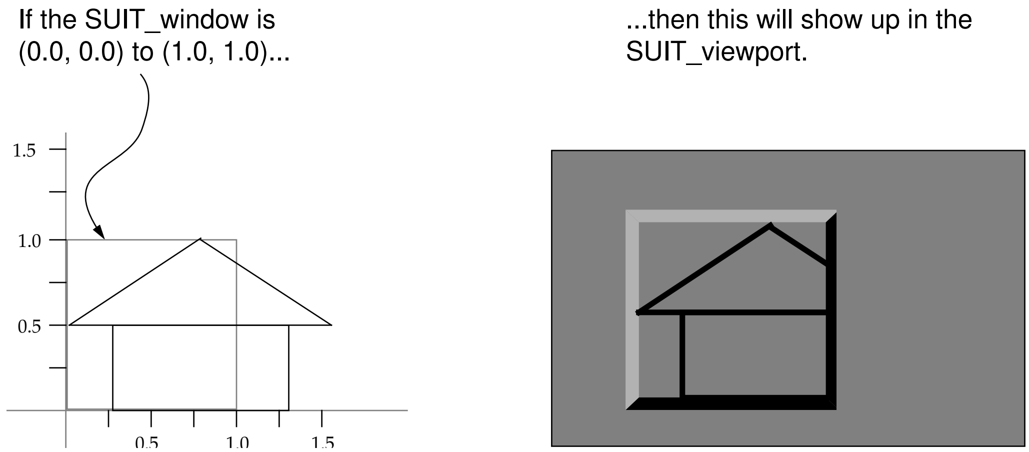
A SUIT_viewport is a *rectangular region on the screen*, measured in pixels. On a 1000x1000 pixel screen, a typical viewport might have its lower left hand corner at (350, 450) and its upper right hand corner at (750, 650). For nearly all widgets, the origin is at the lower left hand corner of the application window (or of the screen in a non-windowing environment). We say "nearly all" because things are slightly more complicated for SUIT widgets that are nested inside other widgets (so called "hierarchical widgets"). We will address this very minor point in the section on hierarchy on page 153. For now, just remember that a SUIT_viewport is nothing more than the rectangular piece of screen real estate that a widget occupies. Clearly, each widget in SUIT (buttons, sliders, labels, etc.) has its own viewport. This viewport information is something that each widget carries with it in the form of a SUIT property simply called VIEWPORT.



Typical Viewport Coordinates. Pixel Coordinates for viewports increase from the lower left to the upper right.

SUIT_windows

So now, we have a drawing that is specified in world coordinates and a region of the screen that is specified in pixels. The only remaining question is: “How do we convert the floating point drawing into pixels on the screen?” Clearly, we can’t transfer all of the world coordinate system to the screen; world coordinates are infinite in all directions, and we know that the viewport is finite. We must choose some finite part of the world coordinate system, a *SUIT_window*, that will get transferred to the viewport on the screen. The graphics that fall inside this world coordinate window will be the graphics that appear inside a widget’s viewport. We usually arrange the drawing so that all of the graphics we are interested in will fall inside the boundaries of the SUIT_window; everything outside the window is blank. All SUIT widgets carry with them a property of type SUIT_window called WINDOW. Using the example from the previous page:



By default, all widgets are created with their SUIT_window set with the lower left hand corner at the world coordinates origin (0.0, 0.0) and the upper right hand corner at (1.0, 1.0). This means that in order for your GP graphics to show up on the screen, you need to make sure that the coordinates that you use fall in the range of 0.0 - 1.0. You can change this default by setting the “window” property of a widget to some other value:

```
SUIT_setWindow(mywidget, WINDOW, GP_defWindow(x1, y1, x2, y2));
```

NOTE: If you are used to working with windowing systems (X windows, Macintosh, etc.), you may be accustomed to thinking that a window is a region on the screen. *This is not the case here; in SUIT, a window is a region of the world coordinate system.* Regions on the screen are viewports.

The Size of a Window

Choosing the size of a window is very much like choosing a scale at which to draw. Most of the time is doesn’t matter because there are no “real world dimensions” associated with the widget; a coordinate like “0.50” means 50% of the way across the widget, no more. For widgets that refer to real world objects (e.g. a widget that allows the user to pan across the surface of a piece of paper as in a paint program like idraw or MacPaint) it may turn out that the default window size (0.0, 0.0) to (1.0, 1.0) is inappropriate. To scale the window to be in proportion to an 8-1/2” x 11” piece of paper you would choose a window size of: (0.0, 0.0) to (8.5, 11.0). To change the minimum and maximum values of a SUIT window belonging to a widget, use the **`SUIT_setWindow()`** and **`SUIT_getWindow()`** functions described on page 62 in the *SUIT Reference Manual*.

Converting From Windows to Viewports and Back

On occasion, it may be necessary to convert viewport pixel coordinates to the corresponding floating point coordinates in the SUIT_window where the graphics are drawn. To do this conversion, SUIT provides a complete set of conversion routines that map X and Y floating point coordinates to their corresponding integer coordinates as well as routines that unmap integers to floating point. For more information, see page 86.

Viewports And Hierarchy

SUIT objects can be nested inside other SUIT objects for purposes of creating composite widgets. When one widget is contained by some other widget, the **contained** widget's viewport is measured from the lower left hand corner of the **container** widget, not of the screen. There is a complete discussion of viewports and hierarchy in the section covering hierarchy, starting on page 153.

Summary of Terms

SUIT_viewport: A rectangular region on the screen, measured in integer (pixel) coordinates. The origin of pixel coordinates is usually the lower left hand corner of the application window, or in the case of non-windowing environments, of the screen. Viewports for hierarchical widgets are measured relative to the lower left hand corner of the parent, or container widget, not of the screen.

SUIT_window: A rectangular region of the world coordinate system, measured in floating point coordinates. A SUIT_window defines how much and what part of the world is to be mapped to the screen.

GP_rectangle: A rectangular region of the world coordinate system, measured in floating point units. Usually meant to refer to a graphical primitive supported by the GP drawing package.

rectangle: A rectangular region of the screen measured in pixel coordinates. The origin is always in the lower left hand corner of the application window, or of the screen (in non-windowing environments). This is a data type that only needs to be used when calling SRGP directly. SRGP is described in the section called "Drawing In Pixels: SRGP" on page 159.