Screen Coordinates

Last updated July 12, 2018

This tech-note describes the various coordinate systems in X-Plane.

Contents [hide]

Coordinates Overview

X-Plane draws in 3-d and 2-d. The “world” is drawn in 3-d; this includes the world, objects in the world, other planes, and the cockpit object for 3-d cockpits. The 2-d panel and all windows are drawn in 2-d. If a plugin makes windows, they are drawn in 2-d coordinates; if you use a drawing phase, the coordinate system varies with the drawing phase.

X-Plane requires a minimum of 1024×768 to run, but can run in a larger screen. The following rules apply to how X-Plane uses the extra screen space:

– For 3-d drawing, the field of view represents horizontal field of view; this degree of view goes from the left to right edge of your screen, at any resolution. Vertical field of view is chosen to maintain a consistent aspect ratio. (This does not meant that the ratio of FOVs is the same as the aspect ratio of pixels on the screen.)

Angles are not ratios. Imagine a screen that’s twice as high as it is wide with a horizontal FOV of 90 deg, and put yourself at the point of the viewing pyramid. Your horizontal FOV

is 90 deg, but your vertical FOV is 'only' about 126.87 deg. X-Plane does this: float fovratio = tandeg(fov/2); viewingpyramid.horizontal = fovratio; viewingpyramid.vertical = fovratio*screen.height/screen.width; — Jonathan

- For 2-d drawing of dialog boxes and the startup screen, the middle 1024 horizontal pixels and bottom 768 pixels are used; the rest of the area to the sides and above this area are filled with blue. This does not affect plugins since plugins cannot draw when a dialog box is up.

- For 2-d windows, the coordinate system runs from 0,0 in the lower left, to width, height in the upper right, no matter what the screen size. You can use the datarefs sim/graphics/view/window_width and sim/graphics/view/window_height to determine the full size of the x-plane window.

The panel is a bit different; the panel is expanded as much as possible such that the entire panel is still visible horizontally, and at least 768 pixels of panel are visible vertically. This can be a bit counterintuitive…on a case-by-case basis:

- If you run at 1024×768 or any other 4:3 aspect ratio, the panel takes the whole screen. Extra-tall panels scroll.

- If you run at a wider resolution than 4:3 extra space is added to the side of the screen to the left and right of the panel.

- If your run at a taller resolution thatn 4:3, extra space is added above the panel. If the panel is taller than 768 pixels tall, some of the panel that was out of view may come into view. If your aspect ratio is taller than 1:1, the panel will not scroll at all, because no panel is taller than it is wide.

In summary, the panel is zoomed to maximize space on the screen while guaranteeing that at least 1024×768 of the panel bitmap is visible, so the actual behavior is a function of aspect ratio.

3-D Coordinate System

X-Plane employs a local cartesian 3-d coordinate system for all 3-d drawing. This cartesian coordinate system is typically set up as follows:

- The unit for all axes is meters (e.g. a 1x1x1 box will appear to be 1 meter tall relative to other scenery).

– The origin 0,0,0 is on the surface of the earth at sea level at some “reference point”.

– The +X axis points east from the reference point.

– The +Z axis points south from the reference point.

– The +Y axis points straight up away from the center of the earth at the reference point.

A few pitfalls of this coordinate system: the Y axis is not synonymous with up; this divergence increases as you go away from the reference point. Most parts of x-plane compensate for this, but there are still some shortcuts, typically for performance reasons.

True north is only the same as the negative Z axis for the point 0,0,0. As you move east or west, true north’s heading (expressed as a rotation around the +Y axis) will change slightly.

The reference point is generally the center of the mapped scenery, which is __not__ the same as sim/flightmodel/position/lat_ref and sim/flightmodel/position/lon_ref. Only use the lat_ref and lon_ref data references to detect a coordinate system shift.

Do **not** program your own coordinate transformations! The only safe way to change coordinate systems is to use XPLMWorldToLocal and XPLMLocalToWorld.

Aircraft Coordinates

A given aircraft in X-Plane can be thought to have its own coordinate system with the same units (meters) as world coordinate systems, but positioned differently:

– The origin is at the default center of gravity for the aircraft.

– The X axis points to the right side of the aircraft.

– The Y axis points up.

– The Z axis points to the tail of the aircraft.

You can draw in aircraft coordinates using these OpenGL transformations:

```
glTranslatef(local_x, local_y, local_z);
glRotatef       (-heading, 0.0,1.0,0.0);
glRotatef       (-pitch,-1.0,0.0,0.0);
glRotatef       (-roll, 0.0,0.0,1.0);
```

where local_x, local_y, and local_z is the plane’s location in “local” (OpenGL) coordinates,

and pitch, heading, and roll are the Euler angles for the plane. Be sure to use glPushMtarix and glPopMatrix to restore the coordinate system.

You can manually transform a point from airplane to world coordinates using the following formula:

```
INPUTS: (x_plane,y_plane,z_plane) = source location in airplane coordi
        phi = roll, psi = heading, the = pitch.
        (local_x, local_y, local_z) = plane's location in the world
OUTPUTS:(x_wrl, y_wrl, z_wrl) = transformed location in world.
x_phi=x_plane*cos(phi) + y_plane*sin(phi)
y_phi=y_plane*cos(phi) - x_plane*sin(phi)
z_phi=z_plane
x_the=x_phi
y_the=y_phi*cos(the) - z_phi*sin(the)
z_the=z_phi*cos(the) + y_phi*sin(the)
x_wrl=x_the*cos(psi) - z_the*sin(psi) + local_x
y_wrl=y_the                           + local_y
z_wrl=z_the*cos(psi) + x_the*sin(psi) + local_z
```

This is in fact 3 2-d rotations plus an offset.

2-D Windows and Panels

For the purpose of user interface, you may assume that the region from 0,0 to 1024,768 is always visible on the screen, because this is the minimum X-Plane resolution. However you should not make assumptions about where wider screen real-estate has been added (e.g. in negative X area or on the right side). The less you put in code about the coordinate system the better.

Currently correct panel positioning is extremely tricky. The dataref sim/graphics /view/panel_scroll_pos gives an offset from the panel's uppermost position (zero) downward, to a max value of 256 for the tallest panel on the shortest screen (1024 panel on 768 tall screen).

For X-Plane through 8.06 and earlier, the coordinate 0,0 is at the left edge of the visible panel. For X-Plane 8.10 and later, 0,0 is at the lower left corner of the screen.

Note: currently it is extremely difficult to do precise alignment with the panel or handle

larger resolutions. A future set of datarefs will provide comprehensive insulated coordinate system information.

As a final note, if you need to have comprehensive access to the screen in a known manner and do not need to align with Austin's graphics or mouse clicks, you can simply change the projection matrix entirely, based on a physical window from 0,0 to sim/graphics/view/window_width and sim/graphics/view/window_height. However, this does not provide an easy way to get mapped mouse clicks.

Using the New Panel Datarefs

X-Plane 8.15 introduces a series of new datarefs that simplifies interaction between plugins and X-Plane's panel. Here's the concept:

- Coordinates are given in terms of rectangles (left, bottom, right and top) that describe the edges of a given useful rectangle on screen.
- The panel rectangle is the rectangle covered by the panel.png bitmap from the current plane. Please note that the height of this panel is variable up to 1024 pixels, so the aspect ratio of this rectangle is variable.
- The visible rectangle is the area on screen that the user can see. Usually this rectangle is smaller than the panel (for a scrolling panel, for example), but in the case of the user running at extra-high resolution, the visible rectangle may be larger than the panel rectangle.
- There are essentially two coordinate systems: __window__ coordinates are the coordinate system used by the XPLMDisplay APIs, drawing callbacks before and after windows, and XPLMGetMouseLocation. __panel__ cordinates are the coordinate system in effect when the panel and gauge drawing callbacks are called.

While at this instant (XP815) the two coordinates are the same, this is __not__ at all guaranteed. The useful rectangles are provided in two different coordinate systems; you should use the ones appropriate to the APIs/drawing callbacks you are using! Because the rectangles are provided in two coordinate systems it is also possible for you to convert coordinate systems using these datarefs.

Besides coordinate mapping and correct positioning of your graphics to correspond to the panel you can also determine the panel scroll position and what parts of the panel are visible at a given instant.

Here are the datarefs:

The total panel in panel coordinates.

sim/graphics/view/panel_total_pnl_l
sim/graphics/view/panel_total_pnl_b
sim/graphics/view/panel_total_pnl_r
sim/graphics/view/panel_total_pnl_t

The visible panel in panel coordinates.

sim/graphics/view/panel_visible_pnl_l
sim/graphics/view/panel_visible_pnl_b
sim/graphics/view/panel_visible_pnl_r
sim/graphics/view/panel_visible_pnl_t

The total panel in window coordinates.

sim/graphics/view/panel_total_win_l
sim/graphics/view/panel_total_win_b
sim/graphics/view/panel_total_win_r
sim/graphics/view/panel_total_win_t

The visible panel in window coordinates.

sim/graphics/view/panel_visible_win_l
sim/graphics/view/panel_visible_win_b
sim/graphics/view/panel_visible_win_r
sim/graphics/view/panel_visible_win_t

All datarefs are non-writable floating point. Note: the panel position may be in fractional pixels. Your plugin is responsible for reducing the positions to integers as needed to deal with texture linear or nearest filtering.

Here are a few important warnings about these datarefs:

– The panel-coordinate datarefs are only valid from a panel or gauge drawing callback! This is because the coordinate system used in drawing the panel may be different for the 2-d or 3-d object! So they really return the “panel coordinates now” and are invalid if no panel coordinate system is established.

– Similarly the window coordinate datarefs are only valid from XPLM windows or the window-related drawing phases, for similar reasons to above.

– Do not assume the difference between the left and right sides or top and bottom sides of the panels is 1024! Be prepared to *scale* your coordinates to make your drawing

larger or smaller in OpenGL units as needed. We provide the entire rectangle so that you can detect both a translation and a scaling in both the X and Y dimensions!

– Panel draw callbacks may be called more than once per flight loop! When the panel texture is larger than the screen, the panel draw callback will be called multiple times with different scroll positions to composite together the panel. So expect the panel datarefs to change even between different drawing callbacks!

Here is some sample code that draws boxes showing the various panel boundaries: DrawPanelBounds.

Contents

Developer resources for the X-Plane flight simulator

Never miss an update

Sign up for email updates below to get the latest X-Plane news in your inbox.

you@domain.com

SIGN UP

We will never sell or share your email address; we'll only use it to send you X-Plane-related emails. (More info in our privacy policy.)

You'll receive about one email a month, and you can unsubscribe at any time.