Defining and Using View Groups in Prepar3D® v3.3

Mark Hargrove
June, 2016

'View Groups' are a new feature in P3D v3.3 that allow for the definition of a set of "views" (which you can safely think of as "cameras"). The feature seems mostly intended to help us manage multi-monitor configurations and is also the basis for the re-introduction of multichannel (i.e., network PC) configurations. While the multichannel feature is available only in the Professional Plus version of P3D, View Groups can be created and used in all versions.

Views can be automatically generated by Prepar3D for you based upon a fairly simple definition of how many monitors you have and how they are physically arranged or you can take full, fine-grained control of each view.

From the little bit of hands-on experience I've had, I don't really see much value in the various automatic layout features, so this tutorial is a fairly deep-dive into creating a view group with fine-grained control of each view.

For the tutorial we'll be using a 3-monitor system – a fairly typical configuration for serious simmers. The three monitors are attached to an Nvidia GTX 980Ti graphics card that has individual output ports for up to four monitors. The card is configured to individually drive each monitor – Nvidia Surround is *not* in use.

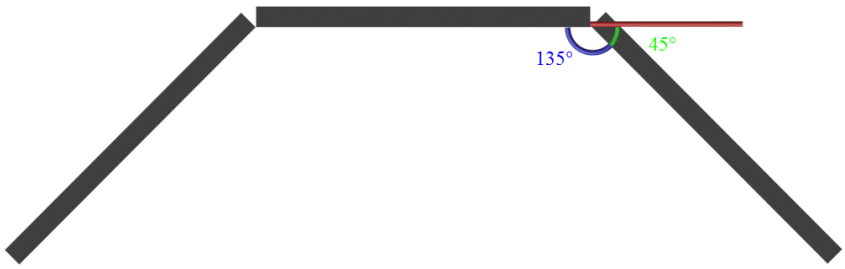
There are a handful of physical characteristics we need to know about the monitors in our setup:

- The physical size of the display area (the screen) – the actual measurements, not the advertised size.
- The width of the bezels (left and right bezels for sure; top and bottom if you're 'stacking' your monitors as well as aligning them horizontally).
- The aspect ratio of your screen (which can be computed from the native resolution if you're not sure what the aspect ratio is – and we'll cover how to compute it later in the tutorial).

In my case, I have 3 30" Dell monitors for my desktop system with a native resolution of 2560x1600 (a 16:10 aspect ratio). The physical display area is 25.3125" wide X 15.875" high. The left and right bezels are both exactly 1.00" wide.

You really only need to measure the width of your displays and the left and right bezels on each monitor because we don't have any vertically stacked monitors for the tutorial. I measured to the nearest 1/16" inch and then converted to decimal.

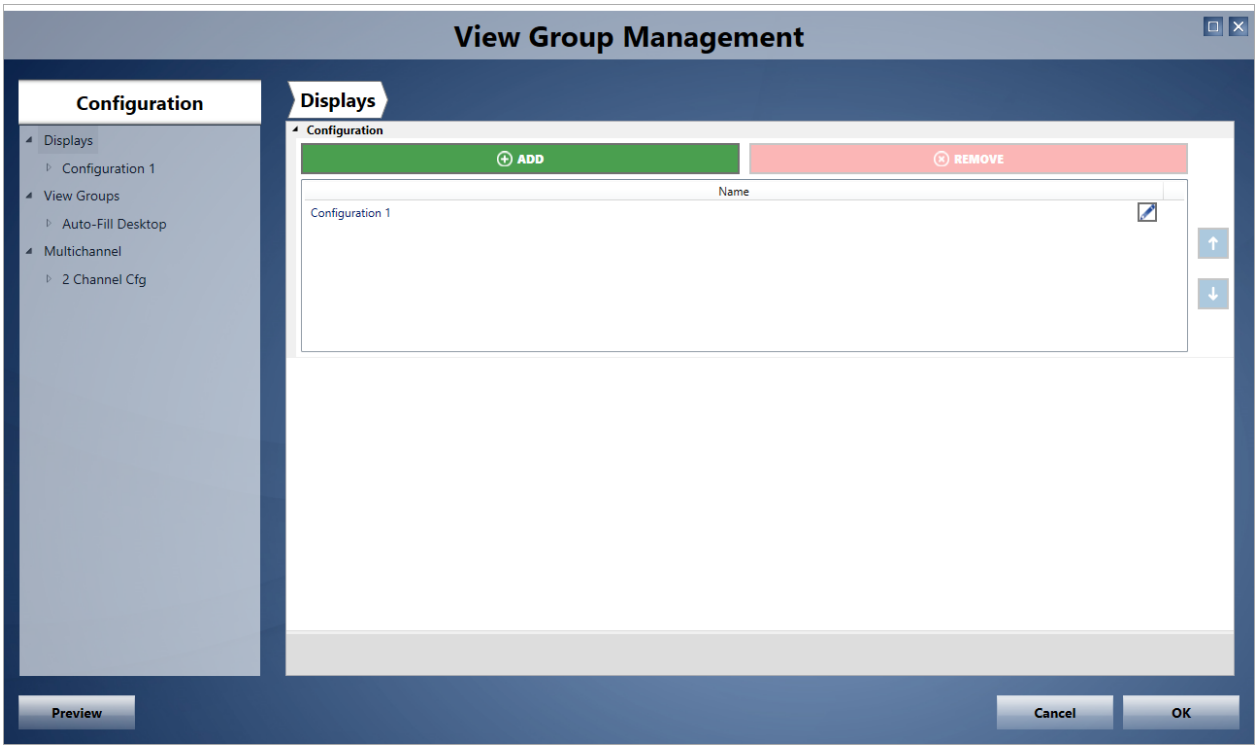
For proper view alignment you need to angle your left and right monitors inwards by 45° measured by the exterior angle from your center monitor (the green arc below) -- or you could say 135° as measured by the interior angle(the blue arc below).



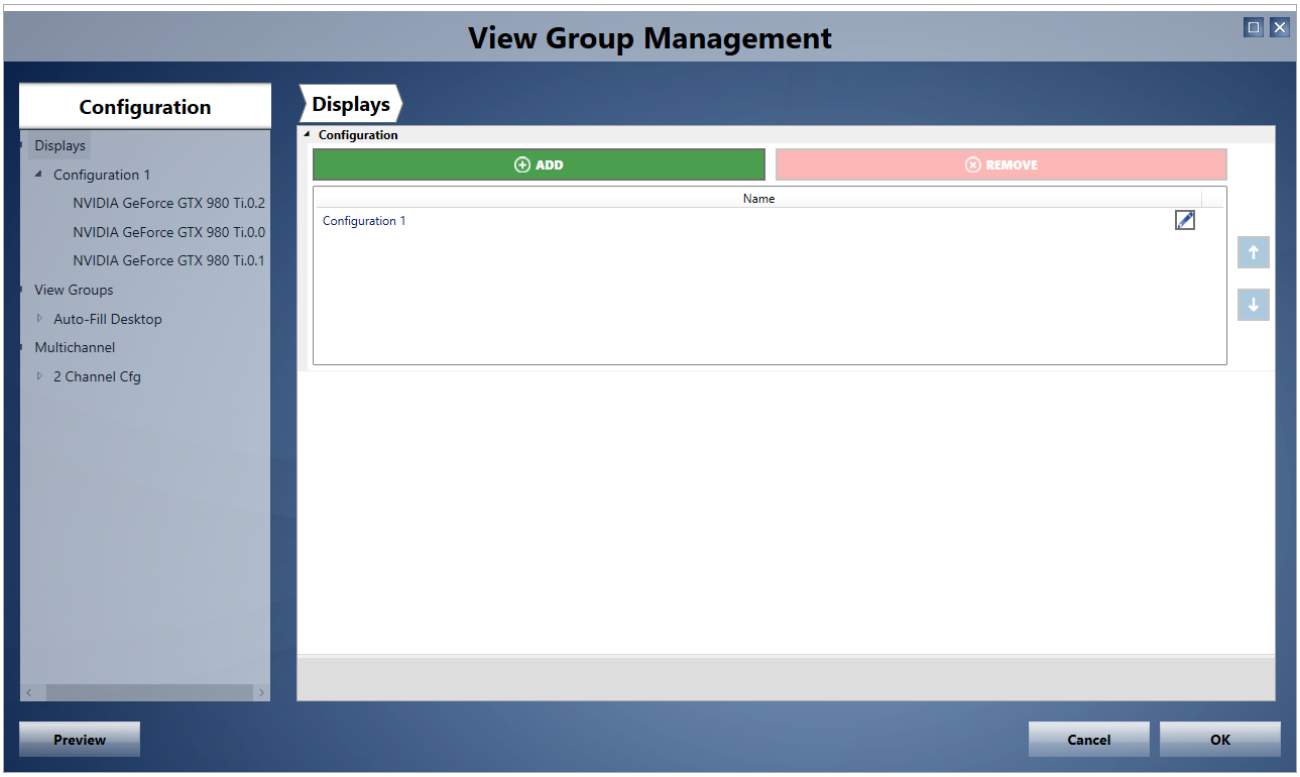
It doesn't matter what aircraft you have selected – View Groups are aircraft-independent (but I've got the stock Beech Bonanza loaded throughout the tutorial if you wonder what you're looking at when a cockpit is visible in an image).

To get started:

- From the ‘Views’ menu, select ‘View Group Management’
- You’ll get a screen that looks like this:

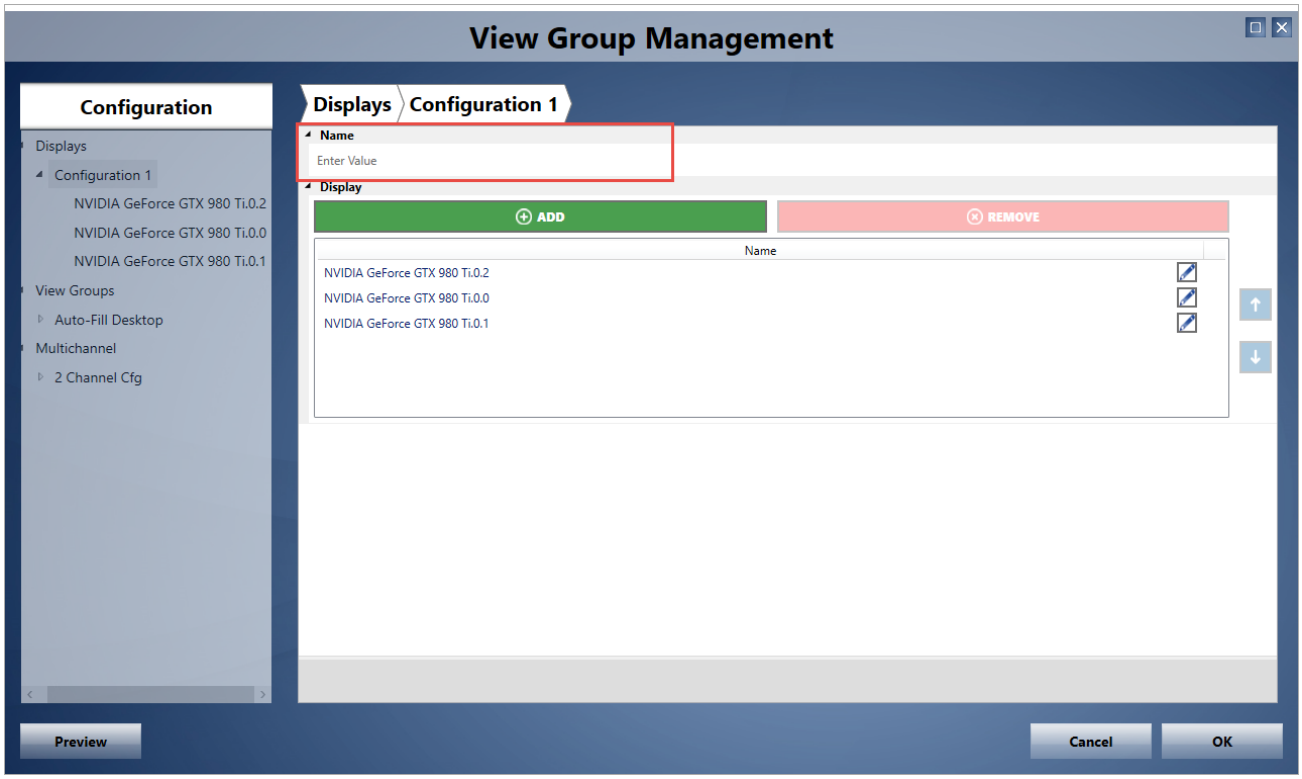


- If you expand Displays/Configuration 1 you’ll see:

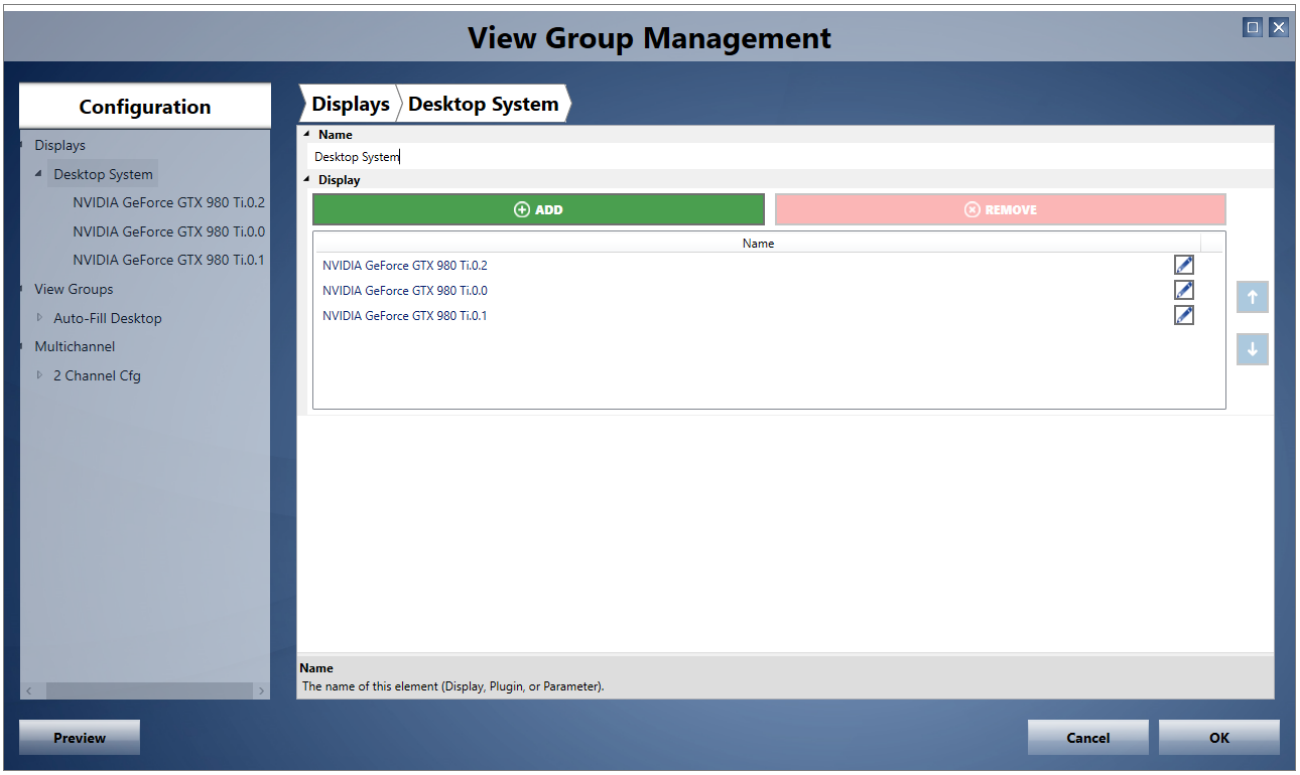


…though your graphics card adapter names and monitor assignments will match what you have on your system.

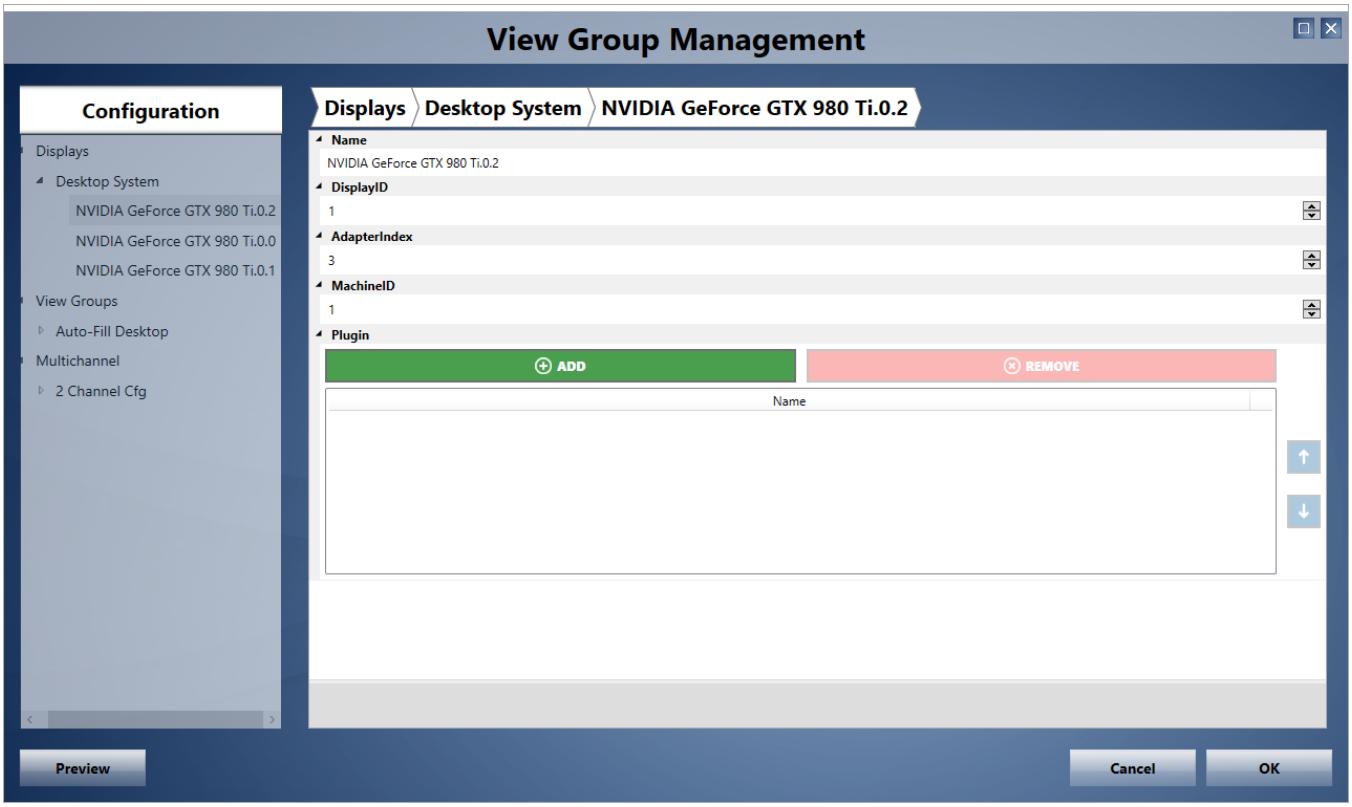
- Click the little pencil icon to the right of ‘Configuration 1’ to see (and edit) the properties for your displays. In particular, we’re going to rename ‘Configuration 1’ to something more meaningful.



- I changed the Displays name to ‘Desktop System’. Notice that the change is immediately reflected in the ‘Configuration’ panel on the left. This will be true anytime you edit the ‘Name’ of something.



- Although we're not going to change any of the individual display properties, click on the 'edit' icon to take a look at the first one. You'll see something like:



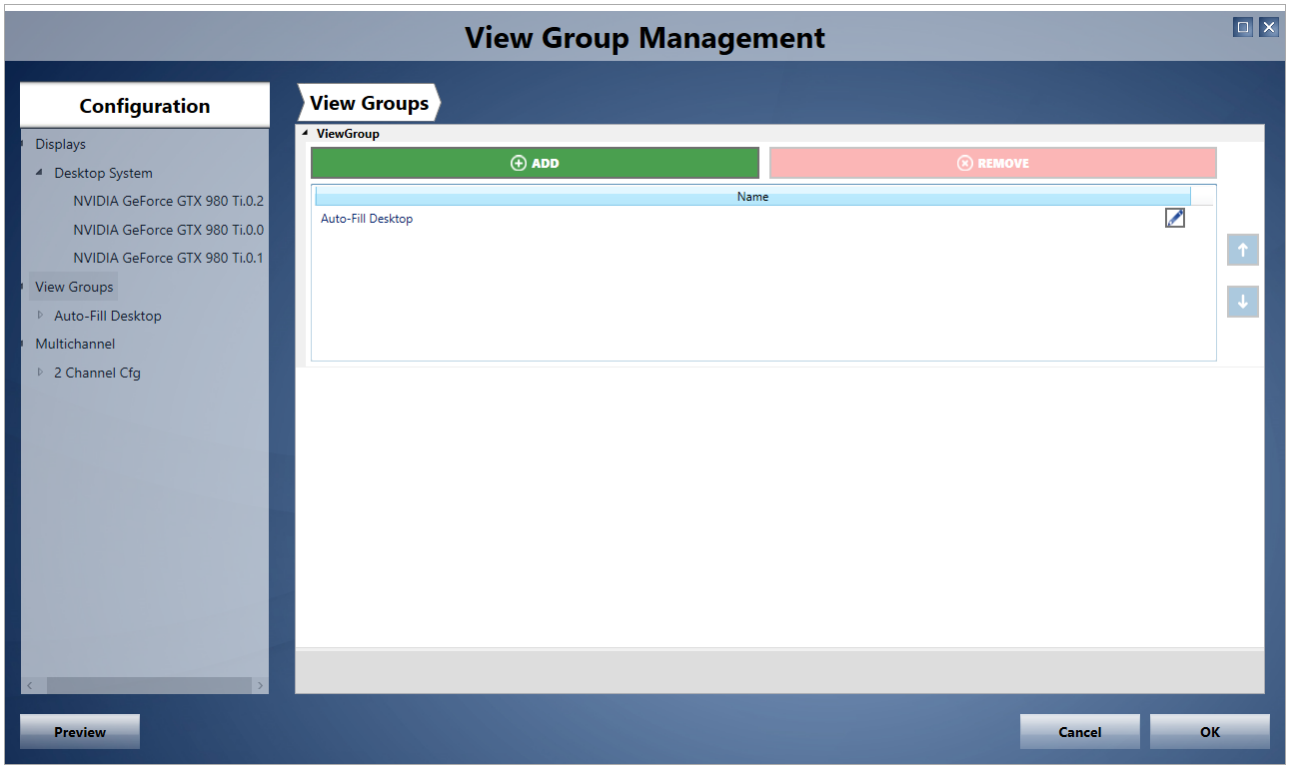
- Most of these don't need to be edited under normal circumstances, but a few of the values definitions are worth highlighting – and you *will* need to know the 'DisplayID' for each monitor to use later.

    - The **DisplayID** field identifies a specific monitor. It appears that the monitors are listed from left-to-right in the 'Configuration' section (at least, that was true of my monitors), so 'Nvidia GeForce GTX 980Ti.0.2' with DisplayID=1 is my left monitor, …Ti0.0 is my center monitor (DisplayID=2 if we looked at it), and …Ti0.1 is my right monitor (DisplayID=3).

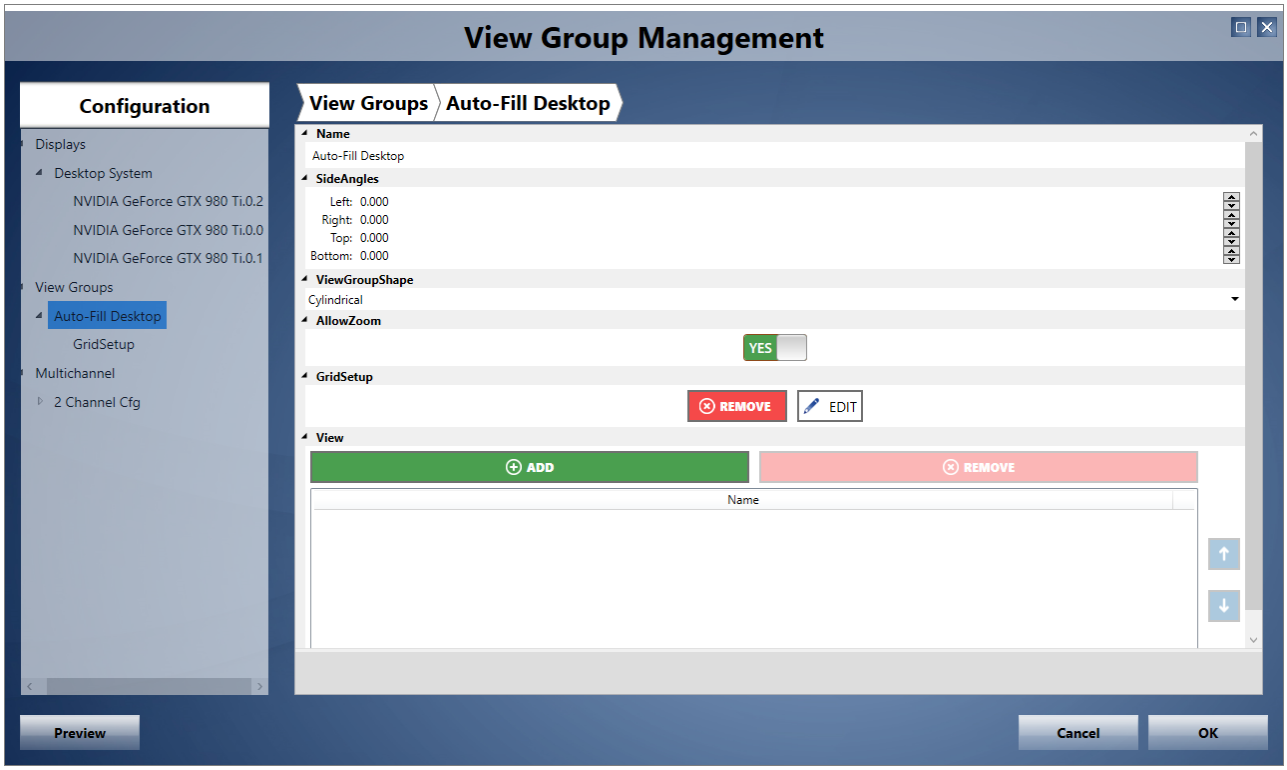
    - The '**MachineID**' value will be used (and have to be set) only for multichannel configuration. Each PC that participates in multichannel operations will need a different 'MachineID' value. The 'Host' PC in a multichannel config needs to have a MachineID of '1'. Multichannel ops is not part of this tutorial.

    - The **Plugin** section allow you to specify software plugins for post-processing the display results on all views defined for this device. You might need this for edge-blending the overlap area on projectors, for example. We're not using any plugins for this tutorial.

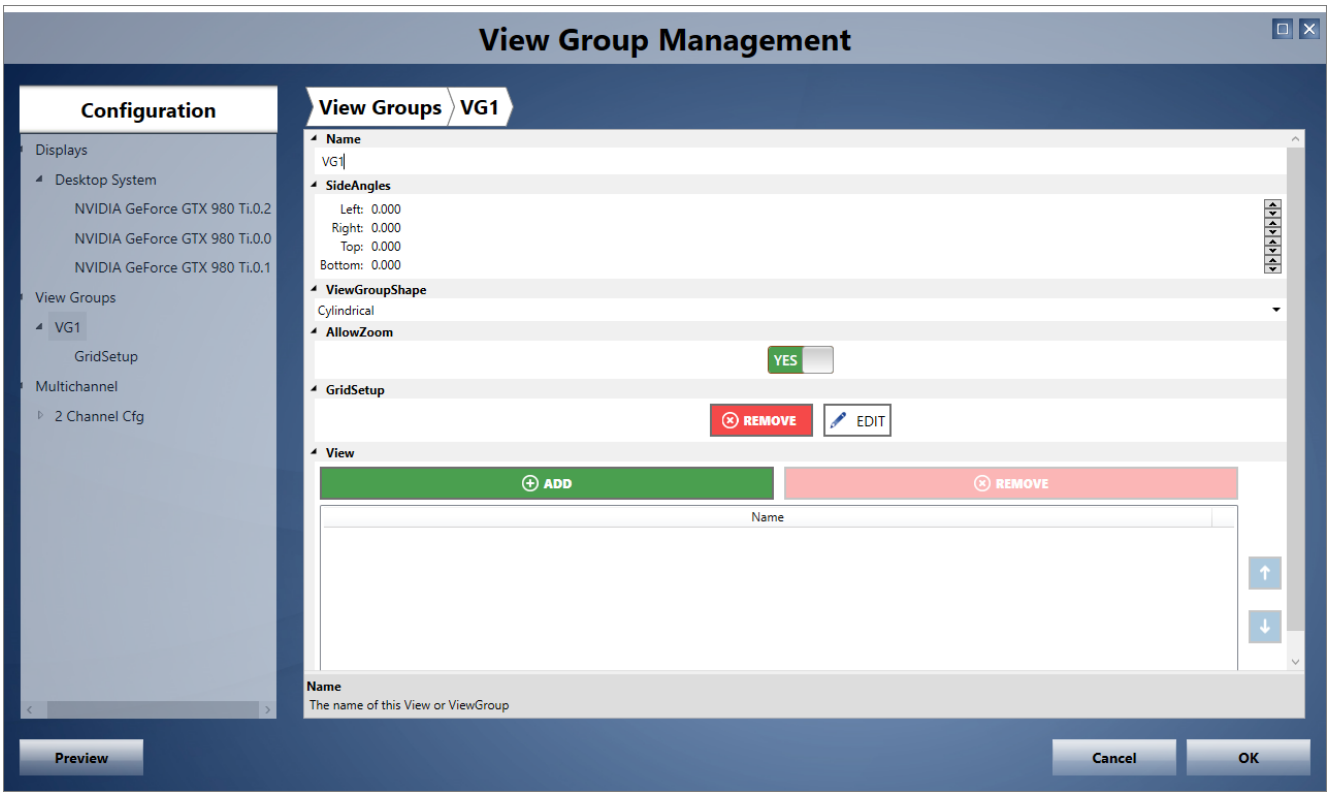
- Now click on ‘View Groups’ in the config panel on the left and you’ll see:



‘Auto-Fill Deskop’ is just the default name for the view group. If we expand it we see:



- Edit the ‘Name’ field to something more meaningful. Note that whatever value you choose here is going to show up in the ‘View Groups’ menu in the simulation later, so you might want to use a short-ish name. I’ve chosen ‘VG1’ (for View Group 1 – you can’t get more creative than that. No, no – don’t even try. You’ll just hurt yourself.)



- Now we start to get down to some of the nitty-gritty details. The *default* values for the View Group's SideAngles are defined here. These are only needed if we're going let P3D calculate the ViewFrustrum for us, but we're going to that ourselves. So:

    - The **SideAngles** values should be left at zero. We are going to have to set these eventually, but we'll do it later for each individual monitor (and I'll define exactly what 'SideAngles' are when we set them in just a couple of steps further on).

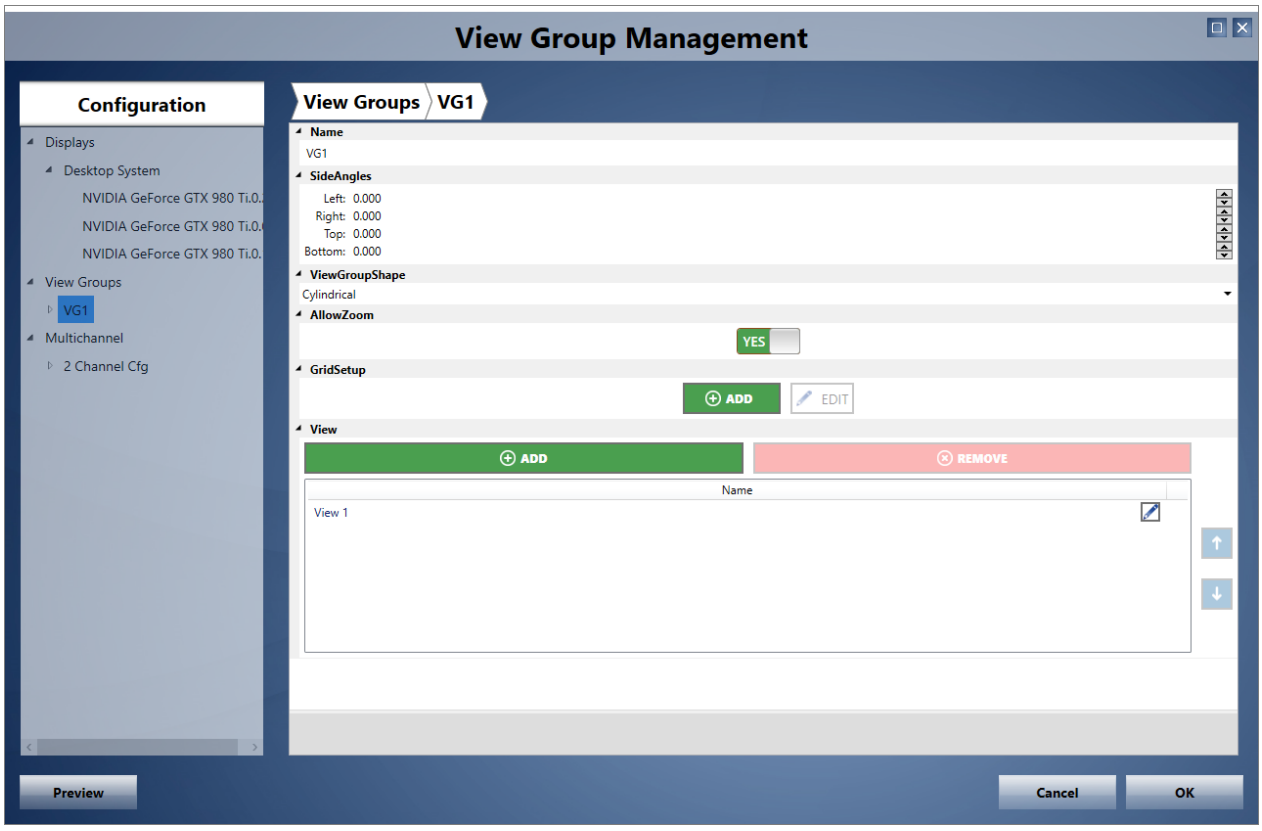
    - The **ViewGroupShape** has three possible values: Cylindrical, Flat, and Spherical. This is the perspective correction that will be applied to ALL of the views in the view group. This is mostly applicable to projectors, but even an LCD panel with a wide horizontal field of view will benefit from using the 'Cylindrical' shape to get rid of the edge distortion. This will (probably; I haven't played with it) help with systems using Nvidia Surround or Radeon Eyefinity setups.

    - **AllowZoom** is a simple "yes/no" flag indicating whether a manual zoom operation (using the +/- keys in the simulator) are allowed for this view group. Set it however you like, but I left it at 'Yes'.

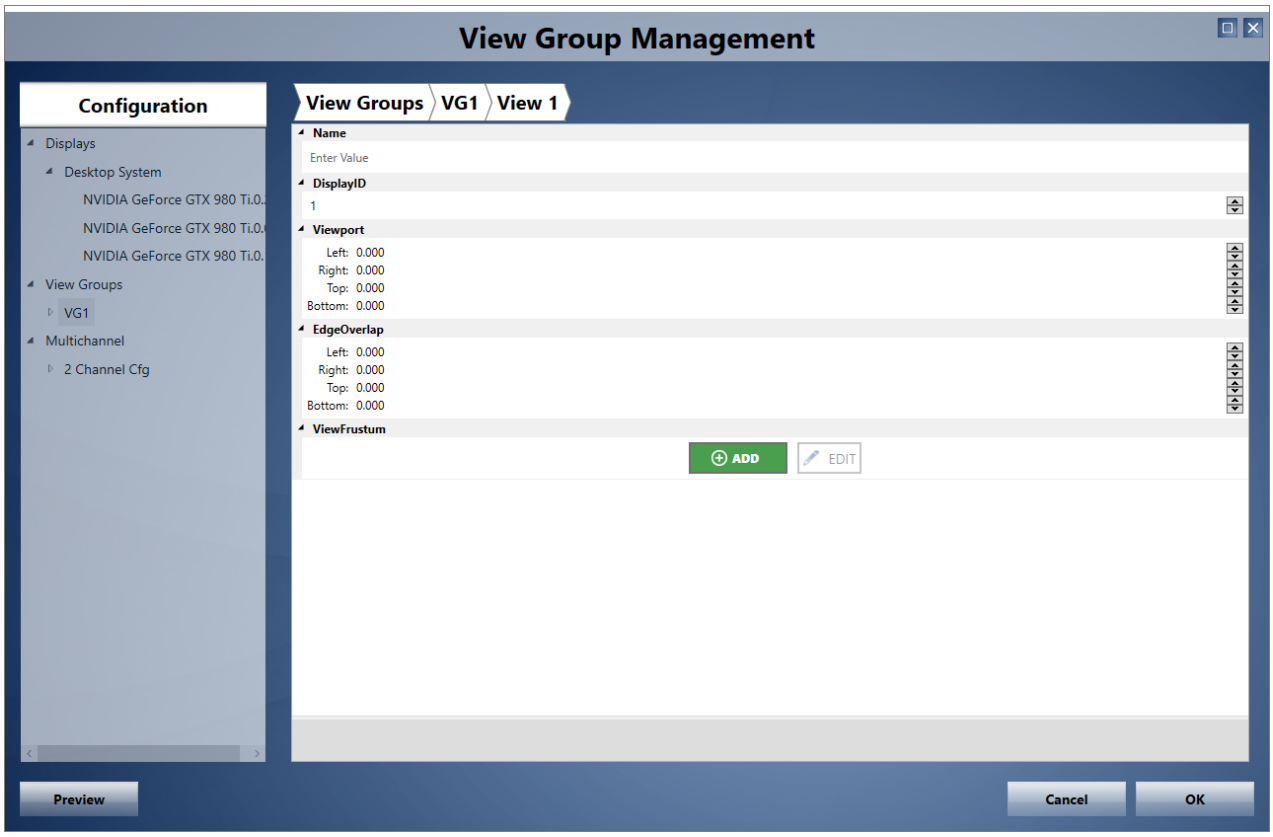
    - **GridSetup** allows you to define what your 'grid' of monitors looks like. If you have three side-by-side monitors, for example, you could specify a 1 x 3 'grid'. If you had a whole wall of monitors you could specify 6 x 24 (or whatever you had). The grid setup is *only* needed if you're going to allow P3D to automatically layout your views, which, again – we're not going to do. You can safely leave this setting alone.

    - **View** is where we define our actual views (cameras) that will make up our View Group.

- Click ‘Add’ in the “View” section (avoid accidentally clicking the ‘Add’ button for a GridSetup. If you do, just click ‘Remove’).



- Click the ‘Edit’ button for View 1:

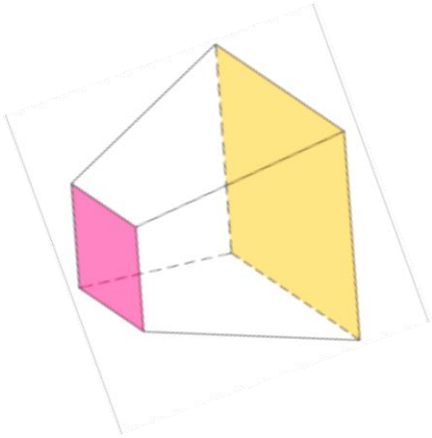


- Now we're finally getting to the actual camera definitions.

    - Start by choosing a name for this View. I'll be starting with my left monitor, so this will be my 'Left 45' view and I'll name it that way.

    - **DisplayID** is 1 (we discussed this earlier – this view goes on the left-most monitor)

    - **Viewport** is used to define the default view frustrum. A *frustrum* is simply a 3-dimensional, four-sided pyramid that defines what the camera sees.



Imagine the magenta shaded plane is the camera location, and the yellow-shaded area is what the camera "sees" at some distance away from the camera lens. The orientation of the resulting 3-dimensional pyramid shape is the 'view frustrum'. *Frustrum* is a term from geometry that you might have learned in 10th grade Geometry class, but have almost certainly suppressed along with all the other stuff from that class. Thank goodness for Wikipedia.

The 'Viewport' values are used only if we're NOT going to supply our own frustrum definition, which we ARE going to do – so leave all these Viewport values at zero.

- **EdgeOverlap** can be used two ways. If you're using multiple projectors, you want the edges of each view to actually overlap a little bit to provide a seamless view (but note: P3D does NOT provide edge "blending" – you need to use a plugin for that. If you want the views to overlap a little bit for projectors, you'll use POSITIVE values for the EdgeOverlap fields.

  The second way this section can be used is to apply *bezel correction* for flat LCD monitors. From reading posts from in the various simulation forums over time I've seen people get confused about what 'bezel correction' means. It's simply a way to account for the fact that the display area of your monitors going from side to side (and up and down if you have a 'video wall') don't actually butt directly against the adjacent monitor's display area – the monitors' bezels (the frame around the monitor) is in the way. If you don't correct for this, the side-to-side (and up and down) images will look wrong. Our eyes see the bezels and want to interpret them as something the displayed image should be "behind". That's what bezel correction does – it crops the displayed image on each monitor slightly so that (for example) the bezels of the left and center monitors appear to 'block' our view of what's 'behind' them, much as a door post in the plane (or in your car) would interrupt a little bit of our view of the scenery behind them. The 'EdgeOverlap' values in the View definition allow for very precise bezel correction (assuming you calculate the EdgeOverlap correctly) so that the side-to-side (and up-and-down, if you have that) views look perfect. For bezel correction the EdgeOverlap values will be NEGATIVE.

The EdgeOverlap values are expressed as a *percentage of the actual display size.* This is just the tiniest bit complicated, but if you follow the procedure below you'll get it exactly right the first time. I have just three monitors arranged horizontally so I'm just going to calculate the EdgeOverlap (really, bezel correction in this case) only for left and right – top and bottom don't matter to my setup. If they matter to yours, just do the same measurements and calculations for top and bottom as well.

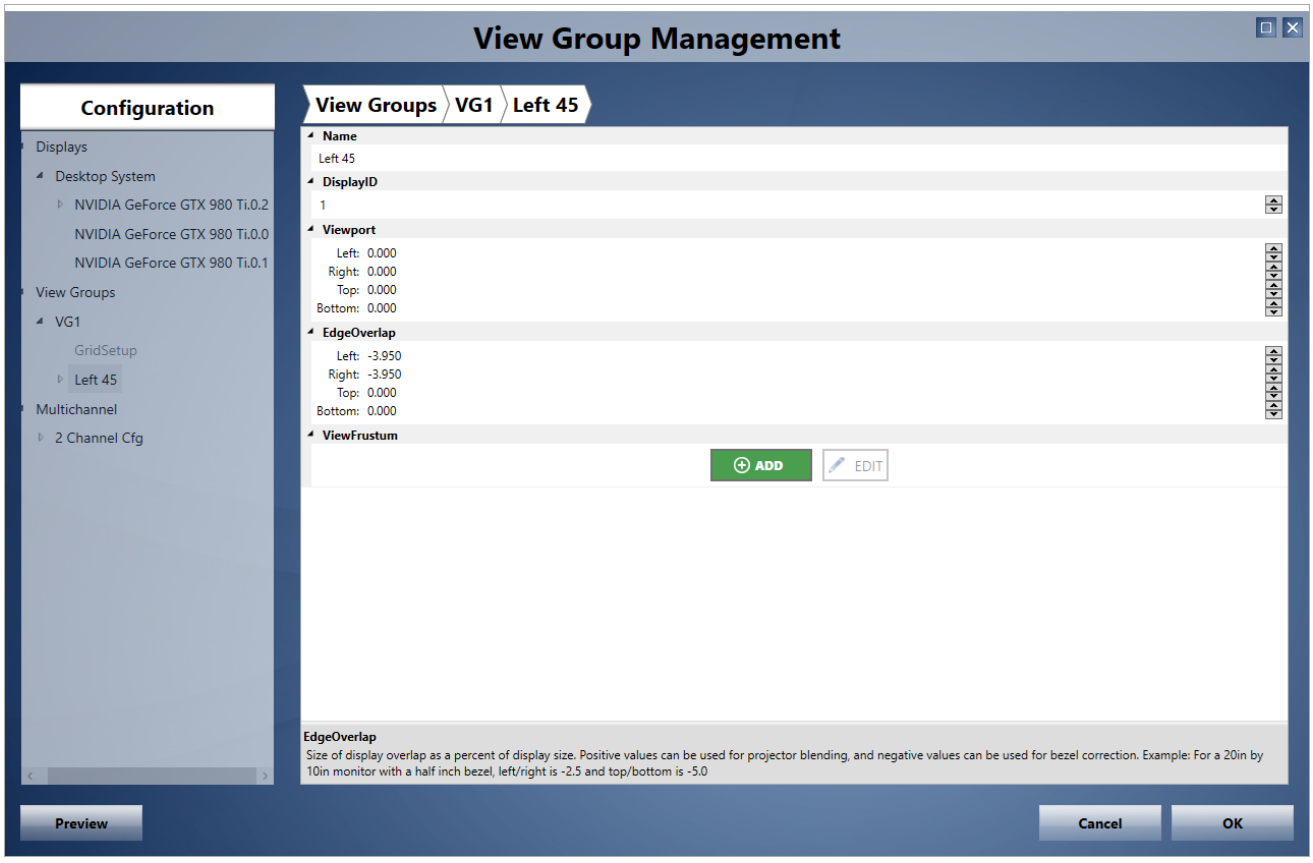
- Measure the exact width of the *display* area of your monitor – that is, everything *inside* the bezel. I measured to the nearest 1/16" inch, then converted that to a decimal value using a calculator. My 30" monitors have a horizontal display area of 25 5/16" (=25.3125 inches).

- Measure the exact width of your bezel on the left and right. It's almost certainly the same value for the left and right, but is very possibly *not* the same up-and-down (if you need that). My bezels are pretty much exactly 1 inch wide (to the nearest 16th of an inch).

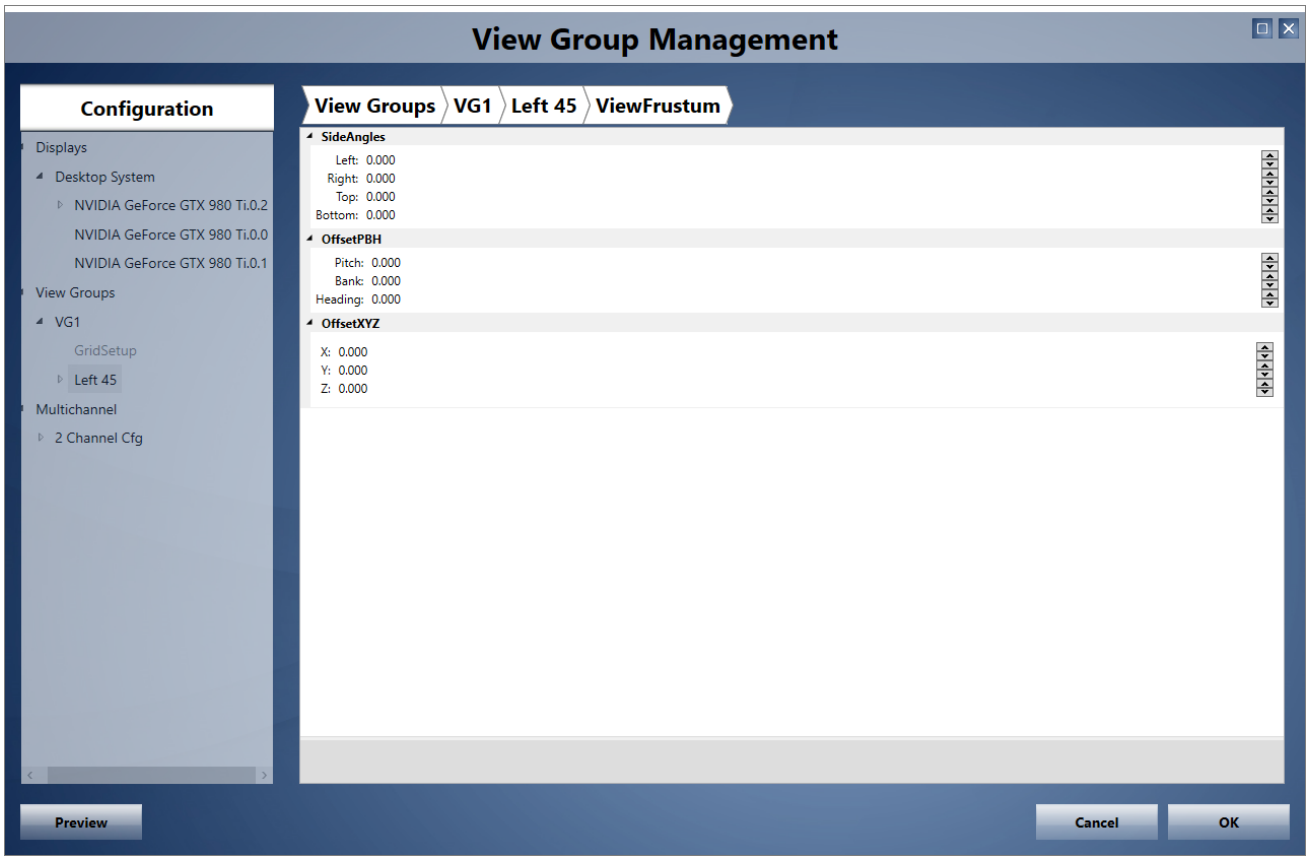
- Expressed as a *percentage* of my display size:

  1.00 ÷ 25.3125 = .03951 …which we'll just call 3.95%

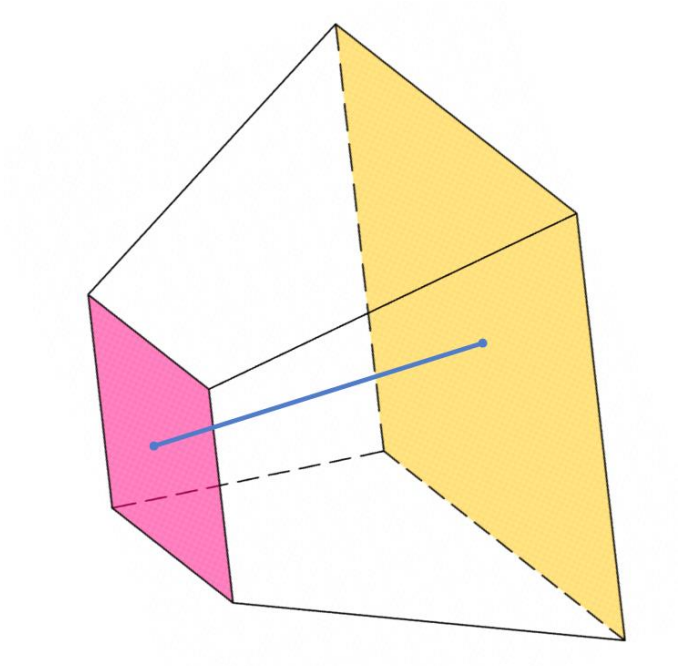
- This needs to be entered as a NEGATIVE value for the Left and Right EdgeOverlap fields:



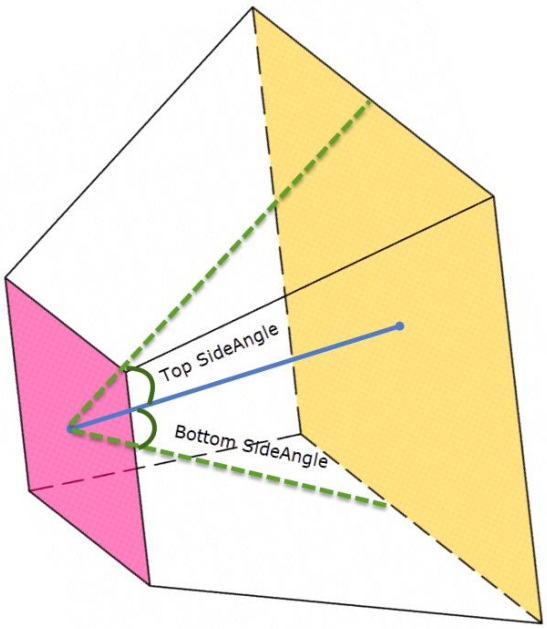
- Now we'll define the ViewFrustrum. Click on 'Add', and then 'Edit' in the ViewFrustrum section:



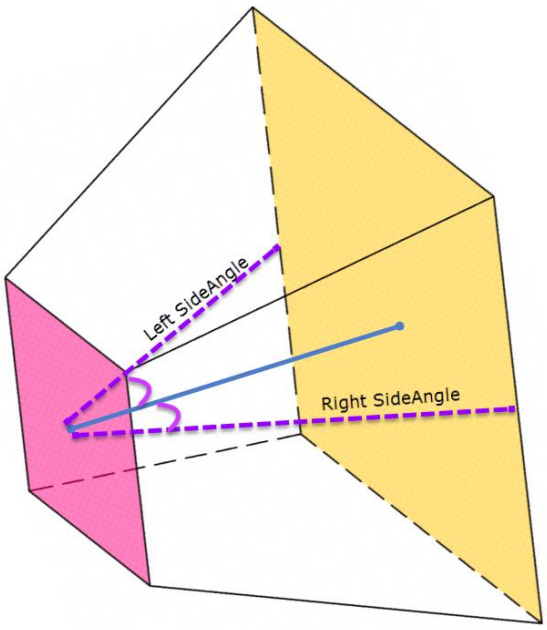
- **SideAngles** are a little confusing at first. Let's look at the 'frustrum' picture again to help understand what they are. The blue line below in the figure below is perpendicular to the magenta and yellow planes and intersects the planes exactly in their centers.



- The ‘top’ and ‘bottom’ SideAngles are the angles, in degrees, from that blue center line to the top and bottom edge of the viewport (the yellow plane). These angles, added together, represent the *vertical field of view*.



- The ‘left’ and ‘right’ SideAngles are the angles, in degrees from the blue center line to the left and right edges of the viewport. These angles, added together, represent the *horizontal field of view*.



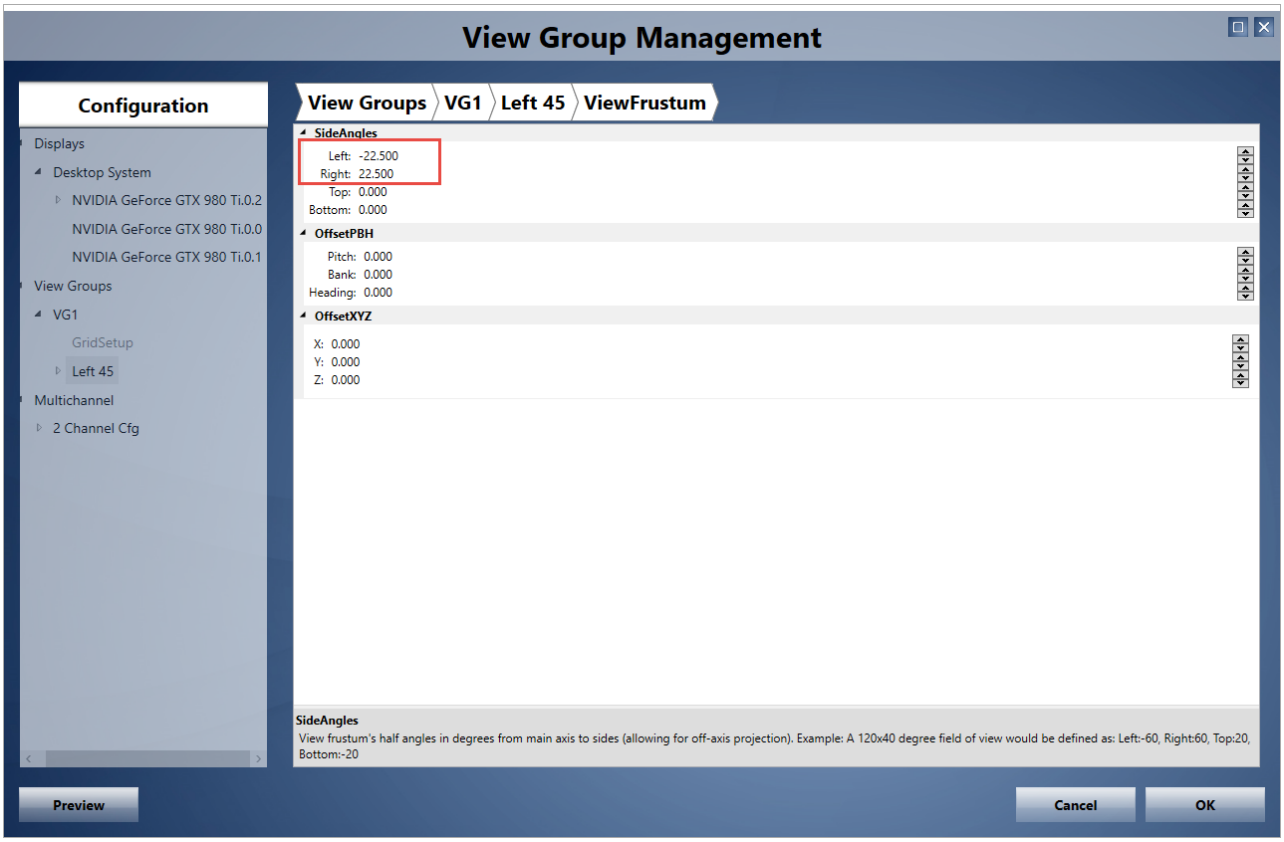
- For this tutorial we’re assuming that left/right and top/bottom have the same absolute value. **“Left” and “Down” will be expressed as negative-value angles, while “Right” and “Up” will be positive-value angles.**

- Now you've got to decide what horizontal field-of-view each of your monitors will represent. Said another way, looking at the frustrum picture directly above, what is the angle you want between the two purple lines? This is pretty much completely up to you. If you have three monitors and decide that each will represent a 45-degree field of view, your total field of view across all three monitors will be 135 degrees. This is quite a bit less than your eyes' real field of view.

  With both eyes working together, the normal human field-of-view is about 200 degrees horizontally (and about 130-135 degrees vertically). So you COULD make each monitor have, say, a 65-degree field of view for 195 degrees total to approximate what your eyes would really see sitting in the cockpit. With only three monitors, though, that would actually look a little weird because your eyes are ACTUALLY seeing roughly a 200-degree FOV but the three monitors aren't wrapping around you enough for that to work (in my real simulator setup I have 5 monitors, each with a 45-degree horizontal field-of-view for 225 degrees total, which *does* completely wrap around my physical field of view).

  I'm doing this tutorial on my desktop system, though, which only has three monitors. Just to keep things simple, we'll still start with a 45-degree horizontal FOV – that will produce a nice looking result – and then once you see how this works, you can play with alternatives to your hearts content. I'll give you some hints on making a wider FOV towards the end of the tutorial.

- For a 45-degree horizontal FOV we have to divide that in half to get the 'side angles' – so that's simply -22.5 degrees for the "left" SideAngle and 22.5 degrees for the "right" SideAngle:



- Now we get to the trickiest calculation of the whole thing – what are the appropriate values for the top and bottom side angles? --or said another way, what do we select as our *vertical field of view?*

  Well, you CAN choose anything you want, but if you want things to look good you're constrained by the *aspect ratio* of your monitor – that is, the ratio of the width to the height of your display area. The easiest way to figure this out is to just look at the native resolution of your monitor: An HD monitor will be 1920 x 1080 pixels, or an aspect ratio of 16:9. My 30" monitors are 2560 x 1600 pixels, an aspect ratio of 16:10. If

you have a UHD (“4K”) monitor at 3840 x 2160 your aspect ratio is also 16:9, same as an HD monitor. Converting that to a fraction, 16:9 is 1.78 (rounded) and 16:10 is 1.6. If you have a monitor with non-standard resolution, just divide the width (in pixels) by the height (in pixels) and you’ll have the value you need.

With this number, calculate the vertical field of view as:

vertical FOV = horizontal FOV ÷ aspect ratio

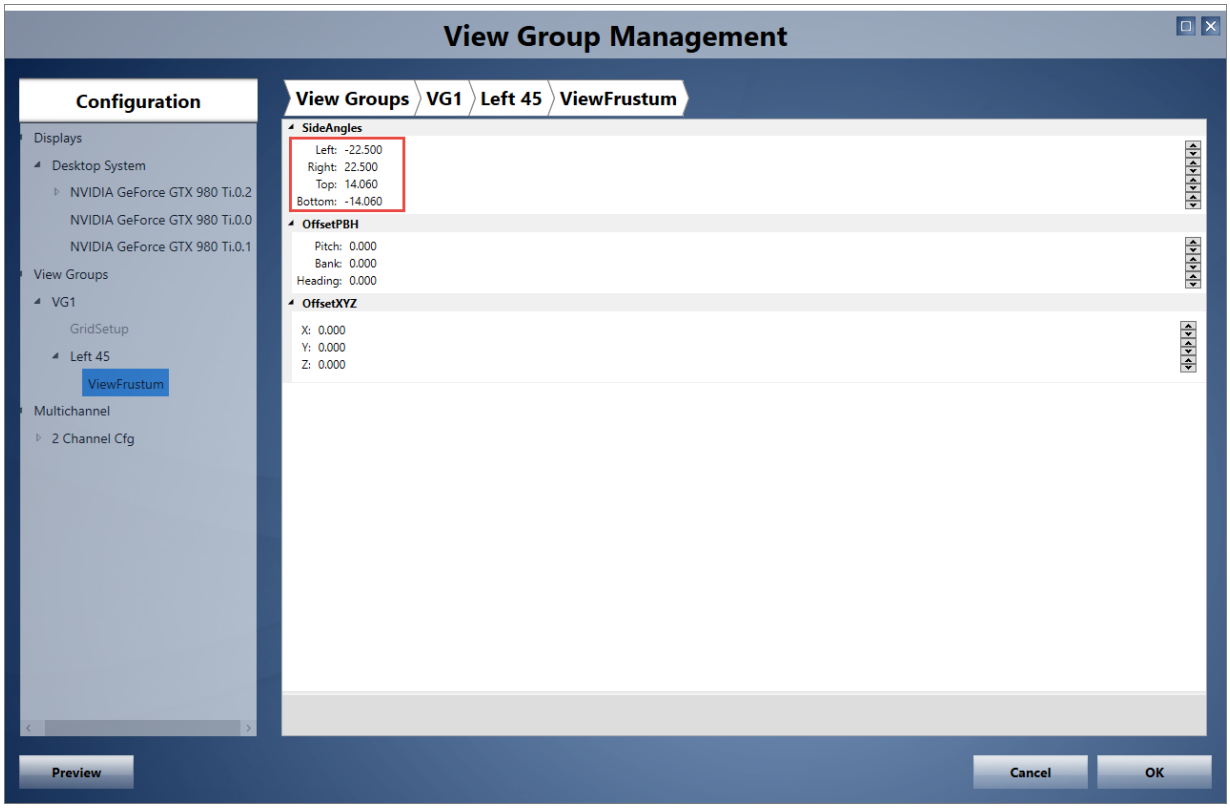
For example:

45 degrees ÷ 1.6 = **28.125 degrees** vertical FOV for my 16:10 monitor, and

45 degrees ÷ 1.78 = **25.28 degrees** vertical FOV for a 16:9 HD or UHD monitor.

Now divide that result by two (as we did with the horizontal FOV) to get the top and bottom “side angles”. For my monitors, that’s approximately 14.06 degrees. For a 16:9 monitor, it would be about 12.64 degrees.

For my monitors, the completed SideAngles section looks like this:

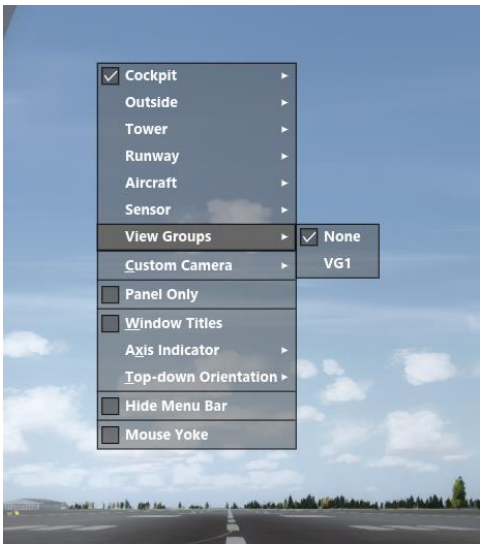


Remember that the ‘bottom’ value is negative.

- We’re not done with defining the ViewFrustrum yet, but we do have enough done to test how this is going to look.

- Click ‘OK’ -- the View Group Management window will just disappear (which is a little disconcerting after all the work we did, but trust me, our work was saved.

- In your ‘main’ view, right-click to obtain the view context menu and slide down to ‘View Groups’. You should see a sub-menu like this:



…assuming you named you View Group VG1. If you named it something else, you should see that in the context menu now.

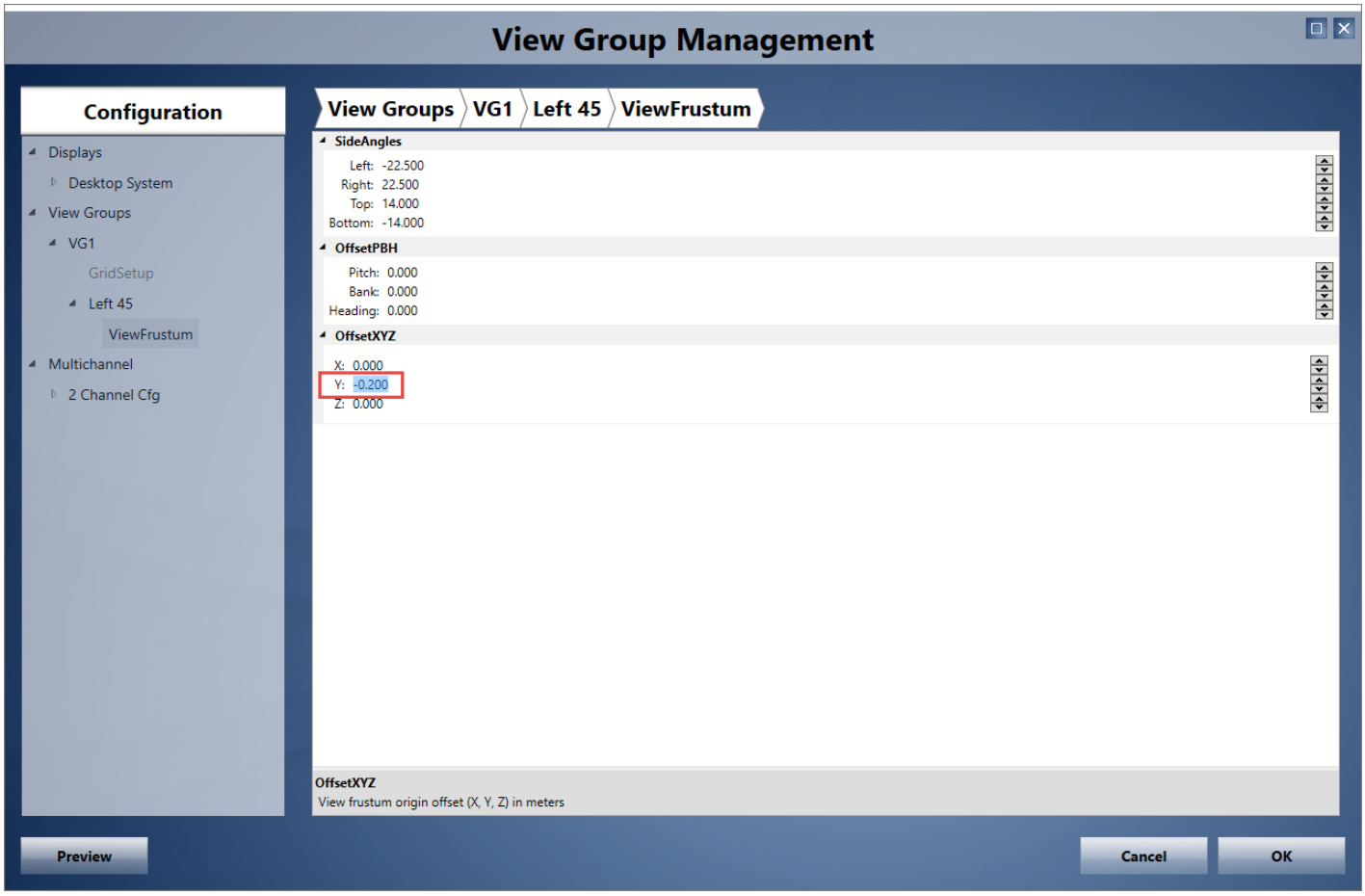
- Select VG1 (or whatever you named your View Group). Immediately you should see a new window open on your Display ‘1’ (probably your left-most monitor). For me, the resulting view looked like this:

A couple of things to check and note:

- The view looks pretty good, but we're really not seeing a couple of things that will tell us for sure – we'll check those things in a moment.
- We're looking straight down the runway, but this is our "Left 45" view – so we should be looking 45 degrees off to the left – we'll fix that in a bit too (I did say up above that we weren't done with the view definition yet, right?)
- To check that we got our vertical FOV right, let's move the eyepoint down so we can see instruments and knobs. You can do this two ways:
  - The usual 'eyepoint movement' keyboard commands for P3D work. Type 'Shift-Backspace' several times to move the eyepoint down.

    Alternatively:

  - Bring up the 'View Group Management' UI again, expand VG1 , expand 'Left 45', and then select "ViewFrustum". Change the OffsetXYZ 'Y' value to -0.2 (this is a temporary change, but also demonstrates what the OffsetXYZ values do. Setting the 'Y' value to a negative value is going to move the camera down by 0.2 meters – just enough to see the instrument panel):



Now when you click ‘OK’ your VG1 window will change to something like:

- And now we can verify that we got the vertical FOV computed correctly – notice that round things actually look round? That means we got it right. If we got it wrong, we’d see something like this:

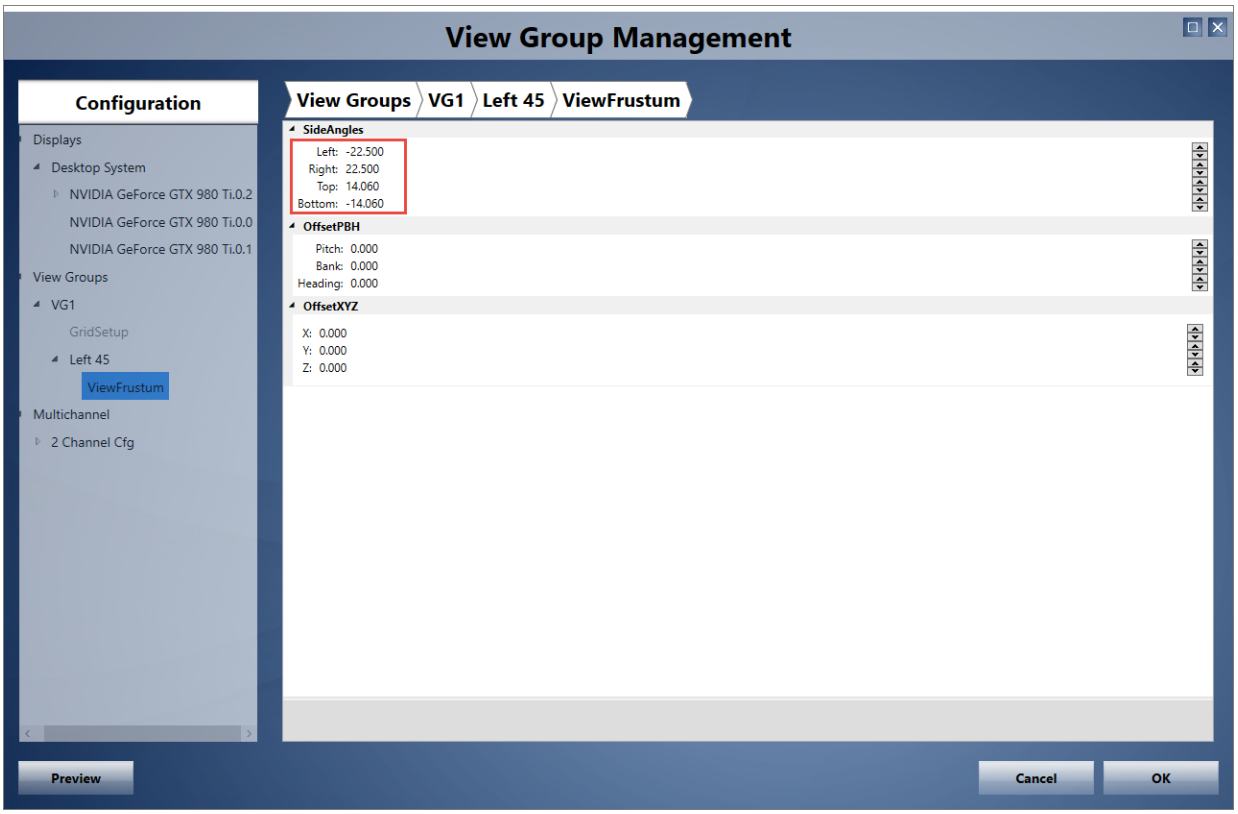
In this case I specified top and bottom SideAngles of 10.0 and -10.0 degrees (a vertical FOV of 20 degrees) which is observably much too small of a vertical FOV angle for things to look right.

Or this:

Here I specified top and bottom SideAngles of 18.0 and -18.0 degrees which causes the circular gauges to be distorted in the other direction.

The theoretical calculations we did above should produce a view that looks very good, but don't be afraid to "fine tune" the vertical FOV if you think your gauges still don't look perfectly round. If they look too "squat" (like in the image above) you need to make the top and bottom side angles smaller. Once you get close to what you want, make very small (like .1 degree) adjustments until it looks perfect to you. Remember to adjust both values (top and bottom).

- Now back to the frustrum definition (and with all of the values restored back to where we were before we started testing):



At this point the only thing we need to adjust is in the OffsetPBH section – specifically, the Heading value.

- Change the Heading value to -45.0 degrees
- Click ‘OK’

What we’re doing is rotating the camera 45 degrees to the left, and our new view looks like this:

- That completes the definition of the first 'View' in our VG1 View Group. The definition of the other two views is virtually identical; the ONLY differences are the OffsetPBH value for 'Heading' in the ViewFrustrum definition and the DisplayID (i.e., the monitor) we'll assign to each view.

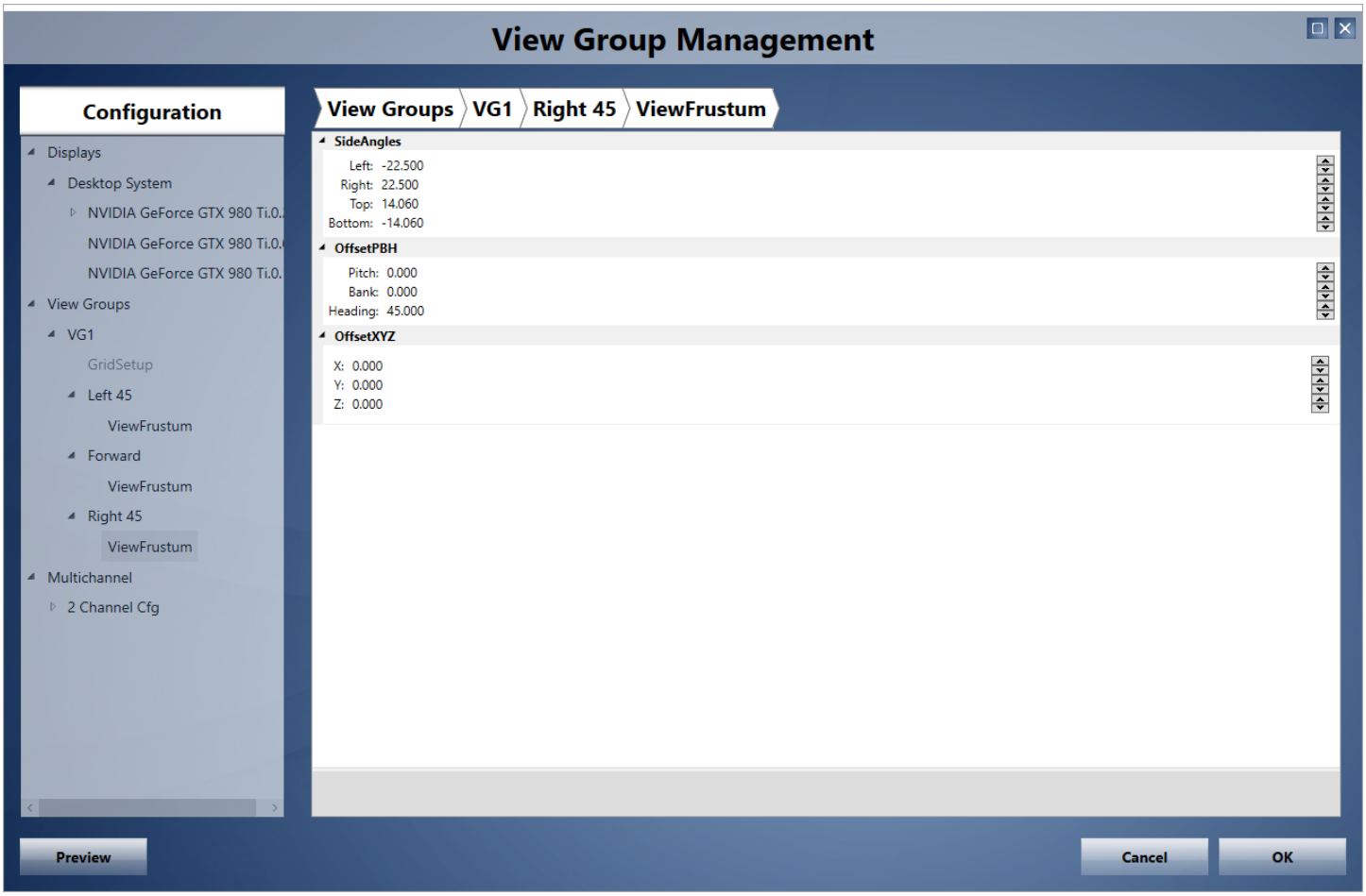
- Here's a quick guide to defining the second view (which will be the center monitor, looking straight ahead).

    - Bring up the View Group Management UI
    - Select VG1
    - In the 'View' section at the bottom, click 'Add' to generate another view (which will be named 'View 2' by default)
    - Click the 'Edit' icon
    - Name the view 'Forward' (or 'Front' or whatever you want)
    - Change the DisplayID to 2 -- this assigns the view to your middle monitor.
    - Leave the Viewport values at zero; we'll define those in the ViewFrustrum
    - Set the EdgeOverlap exactly as you did for the other monitor to apply a bezel correction for the view
        - Sidebar: I am assuming you have identical monitors for your setup. If you don't, use the *actual* bezel measurements for each monitor in the view definition.
    - In the ViewFrustrum section, click 'Add', then click 'Edit'
    - Set the left, right, top and bottom SideAngles exactly as you did for the first monitor (and don't forget: top and right are POSITIVE angle values, bottom and left are NEGATIVE angle values).
    - For this view, leave the OffsetPBH 'Heading' value at 0.0 – we want the camera looking straight ahead.
    - When you're all done, click 'OK'
    - If you have the VG1 view group active, you'll immediately see TWO windows now, one on your left monitor and one on the center (replacing the 'default' forward view that was there before).
    - Feel free to play with these two views – use the 'MouseLook' feature, for example, to pan around and notice that both views are perfectly coordinated (for 'MouseLook' hold down the spacebar and move the mouse around).
    - Notice also that the bezel correction is very good – as you pan left and right the scenery should appear to move 'behind' the bezel and then reappear on the other monitor. Also, straight lines (like the edge of the runway that are visible in both monitors should look more-or-less straight.
        - It's quite easy to get straight lines to look pretty darned good (they already should look quite good if you followed the procedure above and did your calculations correctly. Getting them *perfectly* straight, though, is actually pretty tedious. It can be done by fine-tuning values in the ViewFrustrum definitions. I used to obsess about getting the view alignment *perfect* when I first started working with multiple-monitor setups, but after a few years of fiddling and fiddling I finally realized that I really didn't even notice tiny misalignments when I was flying and quit worrying about getting it perfect.

- For the third monitor, follow exactly the procedure above, except:
    - Name the View 'Right 45'
    - Set the DisplayID to 3
    - Set the OffsetPBH heading value in the 'FrustrumView' definition to (positive) 45.0 degrees.
    - Click 'OK'

- This should give you three windows now, one on each of your three monitors – and we're done.

Here’s what my final configuration looks like with all sections expanded (and the final ViewFrustrum definition for my ‘Right 45’ view showing):



Final Notes

- Experiment the with 'Flat', 'Cylindrical' and 'Spherical', shapes for you view. I see pretty bad distortion of straight lines when panning the view group up/down and left/right if I don't use 'Flat'. Beau Hollis, the LM Rendering Team lead, recommends using the 'Cylindrical' view even for LCD panels and if I'm not going to be panning the view group I think I agree that it looks better.

- If you find that the 'zoom level' of the ViewFrustrum you've defined is too 'zoomed in', there is an easy (but temporary) way to fix it – just use the '-' (minus) key on the keyboard to zoom out. Note that there is a limit to how far your frustrum definition will allow you to zoom out. Even though you'll see the 'zoom factor' in the upper-right corner getting smaller, at some point the view will not zoom out any further.

  There is also a permanent way to change the 'zoom' factor: by changing the field of view for each camera. If you think about it, 'zooming' is just P3D recomputing the SideAngle definitions, keeping the view aspect ratio constant. You can achieve the same effect by using a wider or narrower horizontal FOV (and then computing an appropriate vertical FOV as we did above).

  Let's say we wanted to have the ViewGroup 'zoomed out' a little more. We could do this by setting the horizontal FOV for each view to, say, 60 degrees instead of 45 degrees. This involves going into the 'FrustrumView' definition on each view and:

    - Changing the left and right SideAngles to -30 and 30, respectively
    - Changing the top and bottom SideAngles to the appropriate value for your monitor's aspect ratio. For my 16:10 aspect ratio monitors the top and bottom side angles are 18.75 and -18.75, respectively.
    - Changing the OffsetPBH Heading values for the Left and Right views to -60 and 60, respectively.

  A nice way to experiment with this is to create a completely new ViewGroup (maybe call it VG-60) and name the three views under it 'Left 60', 'Forward 60' and 'Right 60'. If you do that you might also rename VG1 to VG-45 or something similar. By doing this you can switch back and forth between the two view groups from the view context menu.

  Don't forget to also change the physical angle between your monitors for proper view alignment of your new 'VG-60' view group. The interior 135° angle needs to widen to about 150°.

- I have noticed that going back and forth between the views and the "View Group Management" UI causes P3D to crash sometimes. Often, you'll lose whatever change you just made before you clicked 'OK'. Just restart P3D and you'll be back to where you were before the last change you made.

- The UI is managing three XML files that are stored in %AppData%\Lockheed Martin\Prepar3D v3 (the same place as your Prepar3D.cfg file). These files are:

  Displays.xml – which contains the definitions for your displays (duh).

  Multichannel.xml – which contains definitions for networked PC setups (but I don't know if this file is present or not on Academic and Professional versions because multichannel only works on the Professional Plus version). If you don't have it, don't worry about it.

  ViewGroups.xml – which contains the ViewGroups stuff (grid definitions if you use them, View definitions, FrustrumView definitions, etc.).

If you're familiar with XML files you can edit these directly (and that's a really good way to quickly make a new View Group with slightly different parameters).

Have fun with this new feature, and feel free to ask questions, make comments, or provide corrections to the thread in the P3D forum!