

## PortalNode demo for Panda3D

Contributed by Tom SF Haines  
Thursday, 21 May 2009

I have been experimenting with the PortalNode object provided by Panda 3D, and have put together an example showing how to use it. Its purpose is to optimise when you have large levels with limited visibility from one area to the next - think of being inside a building: Portal Demo

Whilst the example is running you can switch between two kinds of portal geometry by pressing 'p' - the default allows you to see the culling effect, the alternate hides the culling effect so you can see that it would in fact look fine in a game.

There is a problem when using this with the current version of Panda - it displays hard coded debug information in the form of a square where the portal is, with lines to the screen corners, and it can't be switched off. This has already been fixed in cvs and will be picked up with the next release of Panda - thanks go to pro-rsoft for that. This demo should continue to work when that happens, just without the silly lines.

A more detailed explanation of what is going on can be found in the header of the linked to code, but I've copy and pasted it below for those who want to know if this is of value to them:

The first thing to realise is that these are not portals in the walk in at one location, walk out at another location sense - they only exist to optimise rendering speed by culling large amounts of geometry. Nothing changes location, or is perceived to have changed location. The idea is that you divide the level into cells, where each cell has limited visibility of the other cells. Each cell is represented by a PandaNode, with everything in that cell a child of the node. You then have portals within each cell - a portal is a single polygon with 4 vertices through which you can see into another cell. For instance two cells might be a room and a corridor, so you would have a portal in the doorway. Portals are one way, going from the cell they are in to another cell; this means they will often come in pairs, one for each way. What they do is when you can see a portal it checks and renders only stuff that can be seen from the cameras viewpoint, that is in the 'out' cell. This works recursively, so if you can see a portal through a portal then that portal will also go through its children and check visibility, and so on recursively. Whilst this is not much use in outdoor environments in indoor environments this method could easily save rendering the vast majority of a level. The below code demonstrates usage but a key issue is that all cells must be hidden except the one that the camera is in. The currentCellTask task achieves this below. Additionally, if an object were to move from one cell to the next its parent cell would need to be updated - the PortalNode only provides basic clipping functionality, the rest has to be done in code. There is also a potential visibility glitch in that a portal could be not visible but an object inside its 'out' cell could be poking through and visible, but because of the portal code not rendered - this can be ignored or has to be solved by the user. (Its usually rare that a user will see it, and only happens with very particular level geometry/object combos.) As a further point you need to enable portal culling in the settings file - this is done by the settings file provided with this example.