Autodesk Inventor OpenGL to DirectX Evolution

Reference the following Autodesk Inventor Discussion Group thread
http://discussion.autodesk.com/thread.jspa?messageID=5641245

Original Question: **Is open GL supported on my M-90 vista 32bit ult using AIP 2008? Nvideo fx2500m The /application/options/hardware/open GL option is missing. Are we all going to direct3D or do i have to hunt for the missing switch?**

Answer: **"Norbert" - Autodesk Inventor Graphics Team**

The switch is not missing, OpenGL use with Inventor is _not_ supported on Vista or on WinXP x64. You can use OpenGL SW, GDI Generic, but not the HW.

Pardon the length of this post, but I think we need to be clear to you and all our users about the situation regarding Direct3D and OpenGL.

When we use OpenGL, we have found over the past many years (and still today) that we need to invest in a large, significant amount of QA that simply verifies that the OpenGL graphics driver supports the OpenGL API on the level that we use (which is actually rather dated, to be consistent with OpenGL GDI Generic, from circa 1997). In spite of the fact that we do not use any new fancy OpenGL extensions and use OpenGL almost on the level of 1997 graphics HW technology, we routinely encounter OpenGL graphics drivers that do not work correctly and as a result, we have our extensive OpenGL graphics HW certification process which involves a serious amount of testing effort on our part that does not actually test Inventor, it merely tests the OpenGL graphics driver. In fact, we currently have 44 (and counting) OpenGL "Workarounds" that we can invoke in our OpenGL graphics layer to "workaround" various OpenGL graphics driver problems.

The opposite situation exists for Direct3D. If a Direct3D graphics driver is Microsoft WHQL (Windows Hardware Quality Lab) certified, then it works correctly as far as Direct3D itself is concerned. This is the purpose of the WHQL testing and certification suite at Microsoft, to enforce compliance with the Direct3D specification. No such process exists in the graphics community for OpenGL and the quality of OpenGL graphics drivers suffers greatly as a result. With Direct3D, our QA team can focus on testing _our_ code and finding defects in _our_ graphics code, instead of having to spend all their time just verifying that the graphics HW vendors have done their job correctly to produce an OpenGL graphics driver that actually works.

Also ... Direct3D works identically on x64 as on x86 so we do not face the problem that an x64 graphics driver is a complete unknown compared to its x86 counterpart ... which is the case with OpenGL.

Now, just because a Direct3D graphics driver is WHQL certified so it supports Direct3D correctly does not mean that _our_ Inventor graphics layer code _uses_ Direct3D correctly. In the case of essentially all defects that we see with the use of Direct3D in Inventor, the problem is in _our_ Inventor graphics layer code and the way we are using Direct3D in Inventor, not in the graphics HW vendor's Direct3D graphics driver. Remember, we have had 10 years to "solidify" our OpenGL graphics layer code on circa 1997 usage of graphics HW so the number of defects in our OpenGL graphics layer code is very small, by definition. The historical state of graphics HW in 1997 defines the way our graphics layer works in Inventor. So ... on occasion, a specific problem is caused by this fact that our Inventor graphics layer code is rather dated (circa 1997) in the way it uses graphics HW.

In addition, when we use OpenGL, we _never_ use the graphics HW for the rendering of any offscreen images, so any shaded views in Drawings, images of Parts and Assemblies, thumbnails associated with files, all printing of shaded 3D views, etc. are _all_ done using the OpenGL SW GDI Generic renderer. This means that we have a limit on the prospect of actually using newer graphics HW capabilites, because we cannot do anything on graphics HW that is not supported in the (circa 1997) GDI Generic renderer; our onscreen and offscreen generated images would not match.

When we use Direct3D in Inventor, we are using the Direct3D graphics HW for _all_ offscreen rendering. For example, any shaded views in Drawings are generated using the same Direct3D graphics HW calls and using the same graphics HW capabilities as the images that are rendered directly onscreen for Parts and Assemblies. This now means that we _can_ start to use current graphics HW capabilities, because any offscreen rendering will _identically_ match the onscreen rendering.

Where does this place us? We are now supporting several new OS configurations. We could expand our QA team to test OpenGL graphics driver support on _each_ of these OS configurations, and hire a signifcant number of people, just to verify that OpenGL graphics drivers work, without ever expanding the testing of our _own_ graphics layer code. Alternatively, we have decided to focus on supporting Direct3D where we can focus the QA efforts on testing our _own_ code and actually redirect some of the people currently spending all their time testing OpenGL graphics drivers to start testing our own graphics layer code, including possible new graphics features and functionality. It also means that in Inventor development, we can add new graphics features and functionality because we know that we will have the QA resources available to test the new features ... and we know that offscreen rendering will match onscreen rendering.

In addition, Direct3D is WHQL certified on a _wide_ range of graphics HW, meaning workstation, consumer, game, laptop, and integrated-on-the-motherboard graphics chipsets. Essentially, using Direct3D, the only differentiator is performance. This is not true with OpenGL, where OpenGL graphics drivers actually _disable_ some OpenGL functionality on consumer, game, laptop, and chipset HW so you are _forced_ to purchase the more expensive workstation graphics HW, just so your OpenGL graphics works correctly.

In this new release of Inventor, we are now accepting this reality and any Direct3D graphics driver that is WHQL certified is certified for use with Inventor. Will there be some situations that arise that WHQL does not test? Yes, we did find _one_ specific situation in R11 and the Microsoft WHQL team did investigate the situation and the WHQL testing and certification suite for _all_ graphics HW on Windows now includes an additional test for the situation that we uncovered with Inventor.

Ultimately, we want to provide an Inventor product that is of the highest quality in terms of graphics _and_ that is affordable for a _wide_ range of budgets. For example, if you are no longer _required_ to spend $1000 to $2000 to simply have graphics HW that works, but can instead purchase a $100 to $350 game card ... then you can take the $900 to $1650 you saved and spend it on other system aspects that might have a _much_ more significant impact on your user experience with Inventor, such as having 8 GB of RAM instead of 4 GB (a _real_ 64-bit system!), having high performance Ultra SCSI SAS drives instead of SATA, or having top-of-the-line multi-core CPU(s) instead of a single lower cost CPU version. Or ... alternatively, if you so desire, you can purchase a lower-cost system in order to save some money and Inventor will actually work on it from a graphics point of view. Any student budget should allow you to purchase a low cost laptop that will work with Inventor becasue if you can play a Direct3D game, you can run Inventor, so far as the graphics are concerned.

What I am asking from you and all of our users, is the investment of your time to report any problems you see when using Direct3D in Inventor. Please work with us as a team to improve Inventor graphics. The defects you find are problems that _we_ can fix in _our_ code, not problems in Direct3D graphics drivers. The long term benefit to you as our users is immense when you no longer have any restrictive dependence on OpenGL graphics drivers or any reliance on only using expensive workstation graphics HW with Inventor.

Thanks _very_ much for your time in reading this _long_ post; I look forward to working with you and our other users to enhance Inventor graphics in addition to finding and fixing any problems you see in our Direct3D graphics layer code.

Part of the difference you may notice is because the environment reflection texture used with Direct3D is different from the one we use with OpenGL.

OpenGL uses an older technology called sphere maps for reflections, which means you use the image of your environment taken from the perspective of a perfectly reflecting sphere.

Direct3D uses a newer technology called cube maps for reflections which means that you supply six textures as the faces of a cube surrounding your model. Direct3D does the reflection computations using the cube.

We have similar but different environments for each. Both in Inventor 2008 are from our parking lot outside our office; however, the OpenGL sphere map is many years old and the car featured in the image no longer exists. The Direct3D cube map was constructed from a series of six images taken in the parking lot in the past year.

You can change the Direct3D cube map to another one if you desire; any Direct3D cube map will work.

The OpenGL certified HW should run Direct3D fine, and frequently better than OpenGL.

We have not compiled a "list" because most graphics HW using Direct3D with a WHQL certified graphics driver (note that WHQL does _not_ certify OpenGL) should work fine for Inventor.

Essentially with Direct3D, performance is the differentiator. You may see better performance by using more expensive graphics HW and the relative performance should correspond roughly to the relative differences in the graphics HW specifications from the HW vendor.

Note that the "workstation" premium does not apply with Direct3D in the same way as with OpenGL. In the case of OpenGL, it is essentially binary; frequently consumer graphics HW will not work (often by design on the part of the graphics HW vendor) whereas workstation graphics HW will allow OpenGL to draw triangles from CAD applications. In the case of Direct3D, both consumer and workstation graphics HW should work (please use WHQL certified graphics drivers!) but there still might be a reason to use workstation graphics HW in terms of moderate differences such as performance or memory consumption benefits that may come from the additional time a graphics HW vendor may spend on a workstation graphics driver as opposed to a consumer graphics driver.

However, the main point is that using Direct3D, you can decide how you wish to apportion your system HW budget. If you want to spend a large portion on graphics HW, you can and it should work well. Alternatively, if you prefer to spend more proportionally on the other aspects of the system such as System RAM, hard disks, CPU choices, etc., you have much greater flexibility to do so.

The Dell M90 is indeed an excellent laptop for CAD applications and Inventor in particular. While the graphics is "marketed" as OpenGL, it works just as well (and sometimes better) with Direct3D. You must remember that we are dealing with graphics HW vendor "marketing" which attempts to direct you to OpenGL where they have the freedom to restrict your choices to only the most expensive graphics HW (i.e. "workstation" graphics HW).

For example, since you mention the Dell M90 which uses nVidia graphics HW, in terms of nVidia desktop graphics HW, we use "workstation" graphics HW (e.g. nVidia Quadro FX 4600, 5600) in Inventor Direct3D development and they are excellent choices for Direct3D. We also use "consumer" graphics HW (e.g. nVidia GeForce 8800 versions) and these also work very well. As an Inventor user, you have much greater flexibility to apportion your system HW budget to where you would prefer to spend it when you use Direct3D.

In terms of the main differences between Direct3D and OpenGL, this can best be summarized by saying that with Direct3D, ñperformance is the only differentiatorò whereas with OpenGL, ñeverything is a differentiatorò. Essentially, OpenGL is defined as a very limited least-common-denominator specification with extensive flexibility for every graphics HW vendor and every generation of graphics HW to ñdifferentiateòitself, including not just fancy new effects but also the basic graphics such as the specific pixels that are illuminated when lines are drawn to the screen and the mathematical computations involved in computing colors. We have over ten years of experience with our Inventor QA and the complete failure of OpenGL graphics HW to meet any strict CAD compliance or accuracy standards. We use OpenGL SW (GDI Generic) in all our QA because at least we get the same consistent result.

The fundamental difference between games and CAD is that in the CAD domain, accuracy is paramount. Performance can vary; however, the final image must be accurate because ultimately, the design will be manufactured. The teapot needs to not only appear to be a teapot (which would be sufficient for a game), in CAD applications, it must hold water.

A good discussion regarding the design of the current version of Direct3D (Direct3D 10) and the philosophy that Microsoft is pursuing with future Direct3D development can be found in the SIGGRAPH 2006 paper by Direct3D Architect David Blythe:

http://download.microsoft.com/download/f/2/d/f2d5ee2c-b7ba-4cd0-9686-b6508b5479a1/Direct3D10_web.pdf

I would like to highlight some specific items in this paper. First, in the Introduction:

ñLike previous versions of Direct3D, Direct3D 10 was designed in three-way collaboration between application developers, hardware designers, and the API/runtime architects.ò

Direct3D is not defined by Microsoft alone. It is a collaborative process (including Autodesk) to define each generation of the Direct3D graphics specification and pipeline.

Also in the introduction, the second point mentioned as a limitation for SW development is especially true for CAD:

ñExcessive variation in hardware accelerator capabilities. Applications respond by writing a small number of ñcode pathsòcustomized to a few families of implementations plus a single minimum-common-feature-set path to cover the remaining implementations. This problem encompasses variations in feature sets, resource limits, arithmetic precision, and storage formats. ñ

This is discussed more in section 4:

ñA less visible, but equally significant change to the processing core is that the data representations, arithmetic accuracy, and behavior are much more rigorously specified than in the past. Much of this comes from recognition that shading programs, constants, and other pipeline state are actually part of the art content in an application rather than part of the execution engine. In this respect, there is increasing pressure to make this content more portable between implementations as well as preserving compatibility between generations of content. This is a reflection of the success of programmable shading.

Where possible we have avoided inventing custom behavior and follow CPU norms. We transitioned to the IEEE-754 [IEEE 1985] single-precision floating-point representation with a goal to converge to the exact behavior within a few years. In this generation, the basic arithmetic operations (add, subtract, multiply) are accurate to 1 ulp (rather than .5 ulp required in IEEE-754). Divide and square root are accurate to 2 ulp. Denormalized numbers are flushed to zero (but are defined and required for float16 operations) and IEEE-754 specials (NaNs, infinities) are fully implemented. These tolerances are driven by cost versus benefit considerations, where our first priority is to create well-defined, consistent behavior between hardware implementations and second to improve accuracy as hardware costs permit.

One of the more controversial design decisions has been the explicit adoption of IEEE-754 special behavior. This new behavior was introduced in the previous generation (shader model 3.0) and has caused some problems with portability of shading programs that relied on NaNs being flushed to zero. While we considered adding a unique mode to allow suppression of specials, we ultimately decided this would ease shader development in the short term at the expense of a greater long term maintenance cost in supporting this mode in all future hardware.

This rigor in specification is not solely limited to the programmable units; it extends to the definitions of the rules for filtering, rasterization, subpixel precision, data conversion, blending operations, etc. Our goal is twofold: to achieve both behavioral consistency and predictability for application developers. Pursuit of these objectives necessitated detailed discussions about NaN-propagation, optimizations of arithmetic or memory operations involving coefficients of 0 and 1, etc. Generally, we tried to arrive at compromises that allow for important performance optimizations. ñ

At Autodesk, we could not agree more with this effort and we have been a major supporter of the position that Microsoft has taken for Direct3D as the Windows platform graphics HW API. We have tried for over 10 years to work with graphics HW vendors to achieve this sort of definition and specification with OpenGL but have failed, primarily because OpenGL by definition is designed to allow for ñdifferentiationò so the products from every graphics HW vendor and of each generation of graphics HW can arrive at a different answer in terms of what you see on your screen. While this may be ideal for games and encouraging you to purchase specific graphics HW to have the best experience with a game, this is completely opposite to the requirements for high precision CAD applications and it is highly ironic that OpenGL is considered a ñprofessionalò graphics API given the complete lack of any enforcement of rigorous conformance and accuracy specifications. In fact, every major CAD product at Autodesk relies on a SW graphics pipeline for QA of the product because OpenGL HW will not meet the rigor required for QA of professional CAD products.

Essentially with Direct3D 10, we finally have the ability to look forward to the same sort of conformance and accuracy for graphics HW products that we take for granted in the CPU world. The importance of this level of conformance and accuracy for CAD or digital prototyping (see: http://www.autodesk.com/digitalprototyping) cannot be underestimated. Getting the wrong answer or any one of a variety of answers really fast (as with OpenGL) is not at all consistent with the demands of professional CAD applications or the new world of digital prototyping. We have the hope that using Direct3D, we will be able to rely on graphics HW products to actually deliver CAD level conformance and accuracy. The "control" of our use of graphics HW is now much more in our hands as SW developers, instead of being largely up to the graphics HW vendor driver to "interpret" according to their implementation details.

I hope this helps provide more background into the Autodesk position regarding the focus on Direct3D for the future of our graphics HW pipeline.

**This information has to be “THE” best response I have ever read from an Autodesk employee.
A very big THANK YOU is rightfully deserved by Norbert!**