Talisman Redefines 3D Rendering

New Microsoft Architecture Trades Precision for Speed

by Peter N. Glaskowsky

In a bold step, Microsoft has entered the business of hardware design and licensing with a new 3D rendering architecture optimized for speed and realism at the expense of pure geometric accuracy. Talisman, introduced at the recent Siggraph, does for 3D rendering what MPEG did for digital video: by applying spatial and temporal compression to the rendering process, Talisman reduces local storage requirements by a factor of 10 and bandwidth requirements by a factor of 50.

These savings come with a price: Talisman renderers will not compete in the CAD market. Instead, the Talisman architecture is aimed squarely at the entertainment market and may eventually dominate it once low-cost Talisman display cards become available, probably in 1998.

Reference Design Uses MSP or TriMedia

Figure 1 shows the major components of the Talisman reference design, expected to be available in late 1997. The Media DSP can in theory be any programmable device, but Microsoft is currently supporting only Samsung's MSP *(see* **101101.PDF***)* and Philips' TriMedia *(see* **091506.PDF***)*.

The Polygon Object Processor (POP), to be manufactured by Cirrus Logic, handles the actual 3D rendering. The Image Layer Compositor (also from Cirrus) and Compositing Buffer/DAC (from Fujitsu) combine multiple image layers into the final screen bitmap and drive the monitor. The Media I/O device (also from Fujitsu) and an AMD 1843 audio chip provide standard 1394, USB, and audio interfaces.

Microsoft did the architectural definition of the POP and ILC, providing a bit- and cycle-accurate simulation of both parts written in C. Synthesizable Verilog code was then generated by Silicon Engineering and provided to Cirrus. Microsoft also developed the real-time scheduling kernel running on the Media DSP.

Microsoft estimates the materials cost of this reference board at $200 to $250. This is quite high compared with the sub-$100 materials cost of typical graphics cards, but it is acceptable for a software-development platform. As the elements of the reference design are integrated together, ultimately into one or two VLSI components plus local DRAM storage, Talisman will become a viable mass-market product.

Bandwidth Limits 3D Performance

The most significant obstacle to high-end 3D graphics on personal computers is the extremely high bandwidth requirements of high-end 3D renderers. For example, a conventional 3D rendering pipeline requires a bandwidth of more than

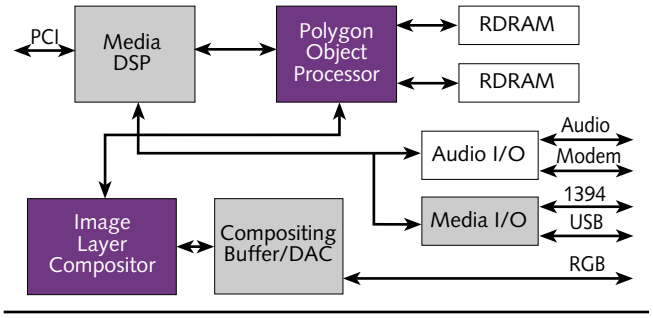

Figure 1. The Talisman reference design consists of a Media DSP, four ASICs, two Rambus DRAMs, and a commodity audio chip. Microsoft-designed components are shown in purple, partner-designed components in gray.

4 Gbytes/s to its frame buffer and texture store to drive a 1,024 × 768 24-bit display at 75 Hz with a 24-bit Z-buffer and 32-bit trilinear MIP-mapped textures. This bandwidth is typical of today's workstation 3D rendering subsystems.

On PCs, this kind of bandwidth is simply unavailable. A single bank of 64-bit 100-MHz SDRAMs, a reasonable maximum configuration for low-cost PC graphics cards, provides about 500 Mbytes/s of net throughput.

During the past few years, Microsoft has hired a number of leading 3D researchers from the academic and commercial worlds. James Kajiya, Jim Blinn, Andrew Glassner, Alvy Ray Smith, and others have been working on the bandwidth problem, and Talisman is the result.

Talisman Reduces Bandwidth Needs

Talisman attacks the problem on a number of fronts. First, unlike a traditional 3D graphics system, the entire display is not updated at the same time. The 3D scene is described by the programmer in terms of *image layers*. These layers will generally consist of a single 3D object that can be separately rendered onto the layer from the current camera position.

Next, each image layer is subdivided into 32 × 32-pixel *chunks*. The polygons from the object to be rendered on each layer are sorted into *bins* based on which chunk (or chunks) the polygon might be rendered into.

Rendering then proceeds on all polygons assigned to a chunk before going on to the next chunk. The 32 × 32-pixel chunk buffer and depth buffer are small enough to be built into the rendering chip itself, allowing faster rendering and greatly reducing off-chip memory accesses.

A sophisticated anti-aliasing algorithm is built into the POP, taking advantage of the small size of the chunk buffer. As rendering progresses, a separate fragment buffer keeps track of pixels within the chunk that have partial coverage or translucency. This technique allows Talisman to produce

high-quality translucency effects that would be prohibitively complex in conventional rendering architectures.

Once a chunk is fully rendered, the POP compresses the chunk using a JPEG-like algorithm called TREC—typically achieving compression ratios of 10:1 or better with reasonable quality—before storing the chunk into local memory.

Textures are also compressed with TREC and may be stored in local memory or host memory. Indeed, rendered image layers may be treated as textures and reused by the graphics pipeline, enabling special effects like shadows and reflective objects. The Talisman texturing engine is based on anisotropic filtering, an improvement over standard trilinear MIP mapping that allows for more accurate texturing of objects that are oblique to the camera position.

Compositor Eliminates Frame Buffer

Once all image layers have been rendered, the compositor combines them into a displayable bitmap. There is no conventional frame buffer, however, in the Talisman reference design. Instead, the compositing buffer provides two 32-scan-line buffers, and compositing takes place in real time. One buffer drives the DAC while the other is used for compositing, in ping-pong fashion.

This architecture makes the rendering process independent of the display process, allowing image layers to be rendered at different resolutions and scaled to final size on the screen: for example, higher resolution for the main characters, lower resolution for less prominent objects, and very low resolution for backgrounds like sky or sea.This strategy provides another level of compression, if the application developer wishes to take advantage of it.

The compositor assembles the raster image from the image layers using an affine transform engine, a mechanism that scales, rotates, and skews the source image as necessary to fit into its position on the screen. This allows the image layers to be minimal rectangular bounding boxes around the rendered 3D objects, which helps to reduce memory requirements. It also enables the next major compression technique in Talisman, one based on temporal coherency.

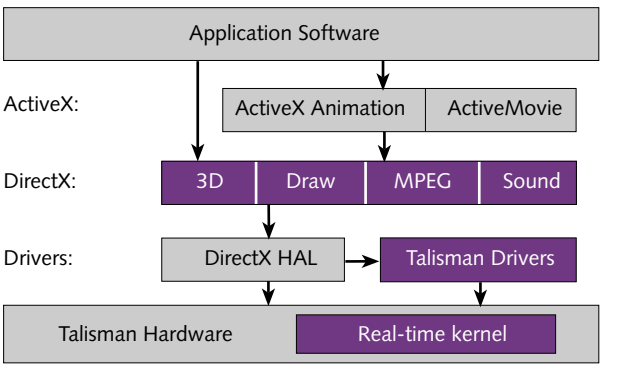

Figure 2. Talisman will initially accelerate the DirectX layers of Microsoft's multimedia APIs, with ActiveX support expected later. Talisman software components are shown in purple.

In 3D animation, objects rarely move much from one frame to the next. Talisman makes it possible to quantify how much each object changes. The application can use this information, along with its own knowledge of how important the object is, to determine how often the object needs to be rendered. If the object is in the background, or not moving rapidly, Talisman can defer rendering it from one frame to the next, substituting the previously rendered image with a new affine transform. This preserves the natural motion of the object at the full frame rate of the display while reducing the rendering requirements of a typical frame by about 3:1.

The net effect of these techniques is to reduce the bandwidth requirement for local memory to about 280 Mbytes/s, compared with more than 14 Gbytes/s for the equivalent features in a conventional rendering pipeline.

Demand for local storage is also reduced by the spatial compression and absence of a frame buffer. While a comparable 3D subsystem might require 10M to 15M for a frame buffer and texture memory, the complete Talisman subsystem requires only 3M for image layers (including textures), plus 0.5M for audio buffers and 0.5M for the Media DSP code.

The maximum screen resolution for the reference design is 1,344 × 1,024 pixels in 24-bit color at a 75-Hz refresh rate. The maximum scene complexity is 20,000 to 30,000 polygons, comparable to a conventional 3D rendering engine rated at 1.5–2 million polygons per second.

Talisman Also Supports 2D, Video, Audio

Conventional 2D graphics can be accommodated within the Talisman architecture. The Windows desktop, together with any application windows visible on it, can be stored as a single image layer. Talisman provides a lossless compression method for this type of image data and, because of the low level of detail in Windows screen objects, relatively high compression ratios can be achieved. The initial Talisman reference design includes a VGA register interface but not a VGA core; VGA operations are managed in software by the Media DSP.

Talisman also provides excellent support for digital video operations. 1394 and USB interfaces provide digital video I/O. The Media DSP supports MPEG-2 decoding and videoconferencing algorithms. Video (from any source) becomes another image layer in local memory. It can be displayed in a window on the screen, or it can be treated as a texture and mapped onto objects in a 3D scene.

Audio is also handled by the Media DSP. Wavetable audio, 3D spatialization, Dolby AC-3, and other modes are supported. Since the Media DSP is programmable, adding support for other audio and video standards is just a matter of writing software. The reference implementation also includes a SoundBlaster-compatible register set, allowing the Media DSP to perform software SoundBlaster emulation for compatibility with older DOS-based games.

Integration with Existing APIs

The Talisman architecture is closely tied to Microsoft's existing DirectX API, as Figure 2 shows. Talisman was developed in concert with DirectX, and only minor extensions to the current release of DirectX will be required to fully utilize the advances of the Talisman architecture.

Existing software written to Direct3D is expected to work normally on Talisman. New Talisman applications will normally use Direct3D's immediate mode. Direct3D's retained mode is also supported, but some of Talisman's performance enhancements are not yet available in this mode.

Taking advantage of Talisman's enhanced performance will require developers to manage image layers on DirectDraw surfaces. This is commonly done with sprites and backgrounds in 2D games, but there has previously been no reason to use image layers this way in 3D applications.

Extensions will be added to Direct3D in early 1997 to support the affine transforms, anisotropic filtering, and other new features in Talisman. Also required will be some mechanism for the application to determine the rendering cost of each object and the available rendering capacity. This will allow applications to create an appropriate balance among scene complexity, rendering precision, and update rates for each object. A feedback mechanism for the compositor is also needed, and Microsoft plans to provide information on how busy the compositor is on each 32-line strip.

Support for Talisman's other capabilities will also be added to the appropriate APIs. DirectSound and DirectMPEG will be provided by code running on the Media DSP, allowing the Talisman subsystem to provide audio and MPEG decoding as if it were a fixed-function device.

With enough code in the Media DSP, a system could provide high-performance audio, video, and 3D support even with a slow host processor, or possibly, in a non-PC application like a DVD player, no host processor at all. This is not part of Microsoft's immediate plan, but the inclusion of the Media DSP clearly makes it a viable long-term strategy.

Talisman to Be Widely Licensed

The essential Talisman intellectual property is available from Microsoft for a nominal administration fee. Vendors can also license the Talisman simulator, Verilog models of the POP and compositor, and the real-time kernel for the Media DSP.

Microsoft expects vendors to create differentiated products by integrating the initial Talisman elements with new features. For example, the texture-mapping engine in the POP is very similar to the compositing engine. Microsoft expects to see these two devices combined into a single chip, allowing rendering performance to be traded for depth complexity on a scene-by-scene basis; this tradeoff is not possible with the current architecture.

Future systems may include Talisman hardware on the motherboard. In this configuration, a P6-class host processor could eliminate the need for the Media DSP, and an AGP interface could provide enough bandwidth to main memory to allow the designer to eliminate all local memory. Talisman can tolerate relatively long-latency accesses to memory to accommodate this configuration.

For More Information

The SIGgraph presentation and some additional Talisman information are available on Microsoft's Web site at *www.microsoft.com/hwdev/devdes/talisman.htm*.

A description and some still images from *Chicken Crossing*, a 3D animated film produced by Microsoft on a Talisman simulator, are at *www.research.microsoft.com/research/graphics/glassner/work/films/chicken.htm*.

New Programming Techniques Required

As previously noted, Talisman is not an appropriate architecture for applications like high-end CAD, where geometric precision and absolute accuracy of the displayed image are required. Talisman can be operated in a mode where none of the spatial or temporal compression mechanisms are used, but this would result in significantly reduced performance, below most of today's 3D accelerators.

The image compositor also imposes special requirements on developers. A 3D scene may include many dozens of image layers, but the compositor cannot load them all each time it generates a strip of 32 scan lines. The average depth complexity per strip is about five layers for the 1024 × 768 24-bit configuration. To solve this problem, the application developer can combine multiple layers into one or reduce the display resolution.

Some types of 3D scenes are difficult for Talisman to render. If the lighting or camera position in the scene changes dramatically from one frame to the next, Talisman may not be able to reuse any of the previously rendered image layers. It will not, however, have time to re-render everything. The application must manage these transitions carefully, possibly by presenting a static or low-resolution image while allowing two or three frame times for Talisman to catch up.

The absence of a frame buffer also prevents the use of other classic display techniques. Software that depends on being able to read the frame buffer (like remote-control programs) will not work correctly. Adding support for a frame buffer in addition to the compositing circuitry will solve this problem and provide another opportunity for product differentiation.

Finally, if a Talisman-aware application is run on a non-Talisman system, it will need to reduce its display resolution and scene complexity. This will add some overhead to the code and the development process.

For games, where scene complexity and fast update rates are paramount, Talisman provides a compelling improvement over conventional 3D rendering architectures and is almost certain to become a popular platform. M