POWERVR MBX

Technology Overview

Filename	:	POWERVR MBX.Technology Overview.1.7f.External.doc
Version	:	1.7f External Issue (Package: POWERVR SDK 2.05.25.0804)
Issue Date	:	07 Jul 2009
Author	:	POWERVR

Contents

List of Figures

1. Introduction

Most 3D graphic hardware uses the brute force approach to render polygons on screen. Tile-based rendering is an alternative 3D rendering technique that is used in POWERVR graphic chips. This document explains why tile-based rendering and its particular POWERVR implementation allow for great performance, image quality and low power consumption and why tile-based rendering is very well suited for present and future real-time 3D graphic applications in embedded systems.

1.1. What is POWERVR?

POWERVR is a 3D technology developed by Imagination Technologies. It is a display list renderer using a tile-based approach. This 3D technology is radically different from existing solutions as it uses a more subtle approach to rendering than traditional renderers. The main differences are in the way pixels are rendered on the screen: while a traditional renderer (or “brute force” renderer) renders everything on screen and relies on a Z-Buffer to sort the end results, POWERVR determines up-front what is visible or not and only renders what is necessary. This ingenious approach makes great savings in memory bandwidth and thus enables modern games and other graphics rich applications to run at optimal performance in memory and power limited environments.

2. POWERVR MBX Architecture

2.1. Structure

A chip based on POWERVR architecture is mainly composed of three modules. These modules will interact with on-board memory to convert 3D data sent by the CPU to on-screen pixels. These three modules are the Tile Accelerator (TA), the Image Synthesis Processor (ISP) and the Texture and Shading Processor (TSP).

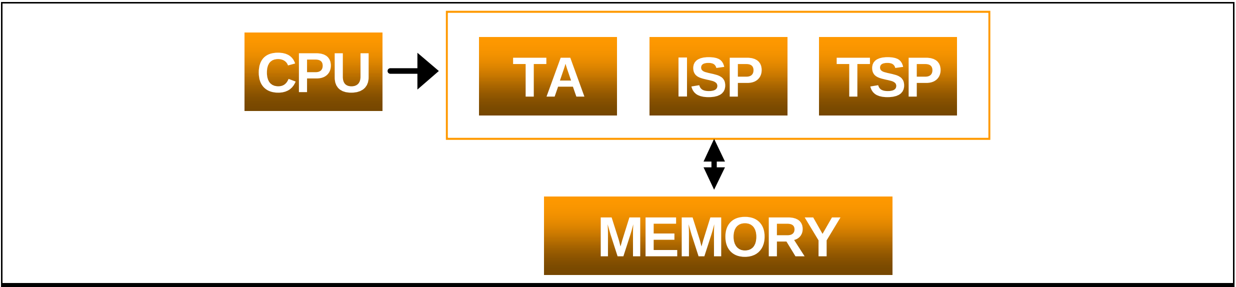


Figure 1: POWERVR chip structure

TA

The Tile Accelerator has the responsibility of storing the scene data and dividing the screen into tiles. This process consists of distributing the 3D data among the screen tiles.

ISP

The Image Synthesis Processor is the module that performs Hidden Surface Removal (HSR). It determines which pixels are visible or not.

TSP

The Texture and Shading Processor has the responsibility of shading and texturing pixels before they are drawn onto the frame buffer.

2.2. Tile Processing

Instead of rendering each polygon on the screen until the full image is rendered, POWERVR will render each screen tile one after the other until the full image is rendered. The TA will create 3D parameters for each tile of the screen until the scene has been completely sent to the 3D hardware. The actual 3D rendering starts once all tile data is in memory. Each tile is then processed in turn: first the ISP will determine which are the visible pixels in the tile and then only these visible pixels will be rendered by the TSP. After the current tile has finished rendering the ISP starts working on the next tile until the scene is completely rendered.

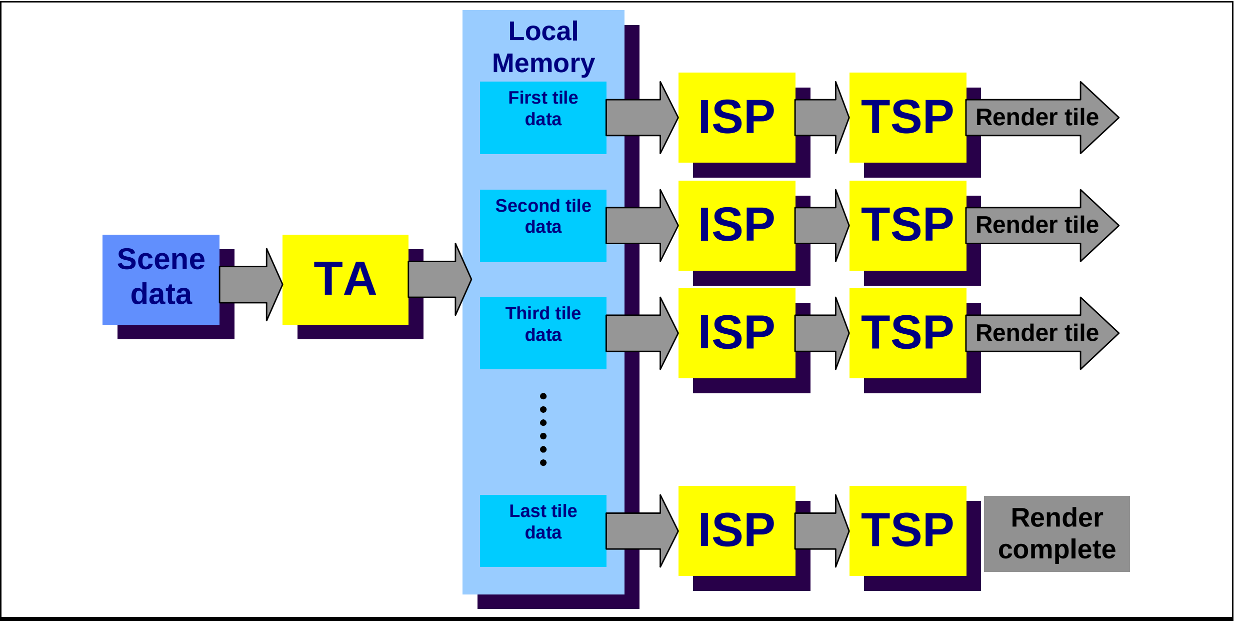


Figure 2: Tile Processing

2.3. TA – Tile Accelerator

Tiling

The main advantage of tiling a scene is the reduced external memory bandwidth requirement. Other advantages such as great cache efficiency and parallel processing of localized data are also important factors. The data for each tile is stored in board memory and contains all the polygons which affect this tile

The choice of a tile size corresponds to the best compromise considering current application technology, hardware design and overall cost.

Storing parameters

The TA creates a "display list" for each tile on the screen. Each tile's display list will eventually include references to all 3D data contained within the tile's boundaries. This process is performed on the fly, as each new primitive call will cause the TA to update the display lists affected by this call.
The display lists contain all relevant scene data, including triangles, texture handles, multitexturing arguments, render states, etc. The scene is effectively "captured" to be rendered later on, hence the term "scene-capture renderer".

Vertex stripping

In order to maximize memory usage efficiency, the TA automatically converts all primitives to strips. Strips are a primitive type requiring less memory than triangle lists. Using strips, n triangles are defined by n+2 vertices. The following figure shows a 6-triangle strip defined by 8 vertices:

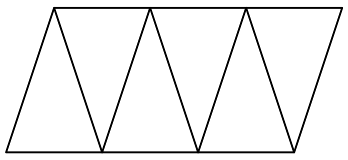


Figure 3: A typical strip

Tile Buffer

This term refers to the portion of memory that contains pointers to the triangles present in the tile. The tile buffer consists in a linked list of pointers pointing to strip data in local memory. The same strip can cross a tile boundary; in this case a pointer to the strip data will be added to the display list of both tiles.

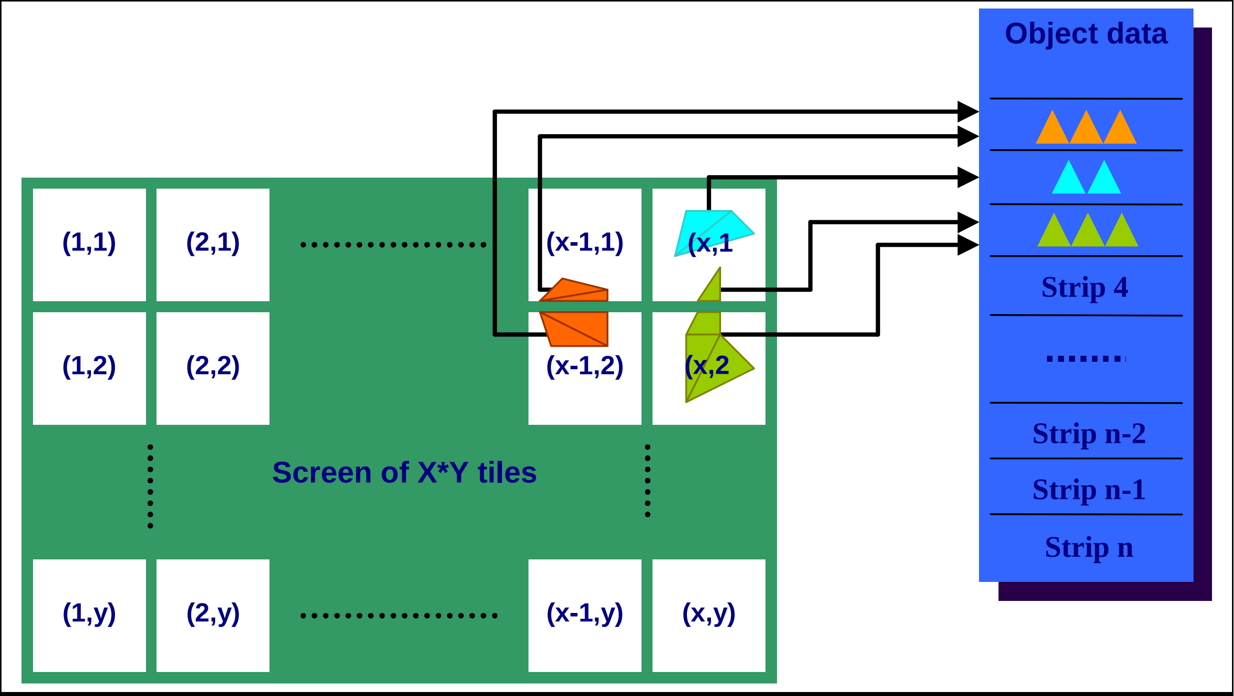


Figure 4: Tile buffer

2.4. ISP - Image Synthesis Processor

Visible Pixel Determination

The ISP has the responsibility of determining which pixels in a tile are visible. Hidden Surface Removal is performed on a tile per tile basis, with each tile's HSR results sent to the TSP for rasterization of visible pixels. Only the triangles affecting a tile will be processed. If a strip is shared between two tiles, say, then only the triangles contained within the current tile will be processed.

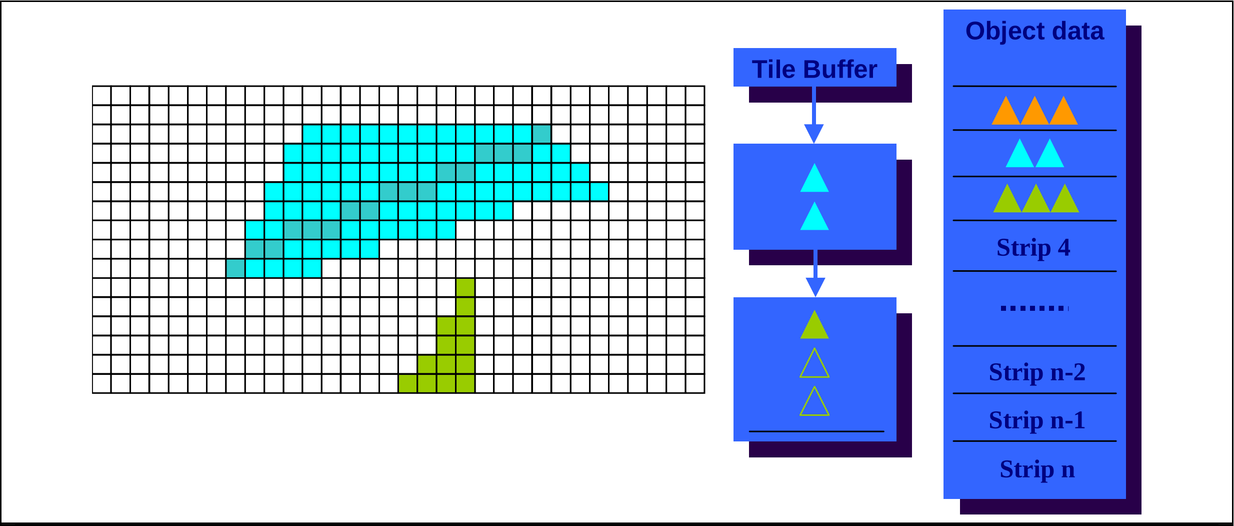


Figure 5: Tile data

The ISP processes all triangles affecting a tile one by one. Calculating the triangle equation and projecting a ray at each position in the triangle return accurate depth information for all pixels. This depth information is then compared with the values in the tile's depth buffer to determine whether these pixels are visible or not.

HSR is done at very high speed, as each line in a tile is processed in one clock (32 pixels). This is an obvious advantage compared to immediate mode renderers, which have to perform per-pixel depth interpolation for each polygon. Another advantage of this technique is that vertical and horizontal clipping (XY clipping) is free, as HSR is only performed on the pixels we're interested in (i.e. on-screen pixels).

ISP Output and Texture Grouping

After the ISP has finished processing all triangles in a tile, each pixel has a "tag" attached to it whose role is to identify the polygon properties that should be used to texture this pixel. However, texturing is always better done when dealing with whole polygons rather than individual pixels. Indeed dealing with whole polygons sharing the same texture properties greatly improves cache memory efficiency.

Because the ISP outputs spans of HSR result on a per-pixel basis, a "texture grouping" module has the responsibility of re-ordering this data into whole polygons. All pixels of the same texture handle are grouped together to maximize cache effectiveness. This module also acts as an input FIFO to the TSP so that both ISP and TSP are kept busy at all times.

The following diagram illustrates this process on an arbitrary 8x4 tile:

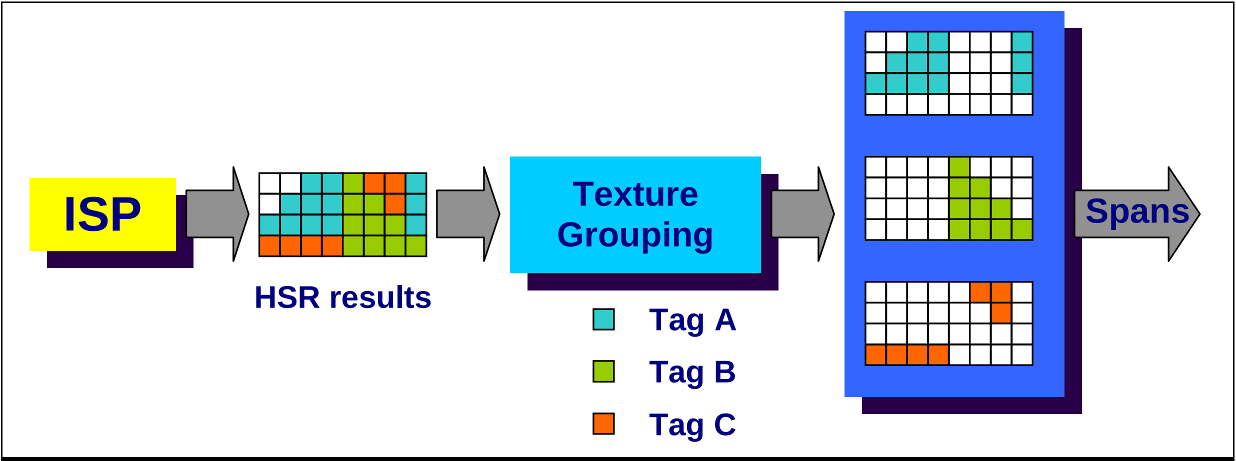


Figure 6: ISP and Texture Grouping module

2.5. TSP - Texture and Shading Processor

Shading and Texturing

The shading and texturing module in the POWERVR pipeline behaves much like a traditional shading and texturing engine. The output spans from the Texture Grouping module tell the TSP what are the lighting and texturing parameters, and what pixels they affect in the tile.

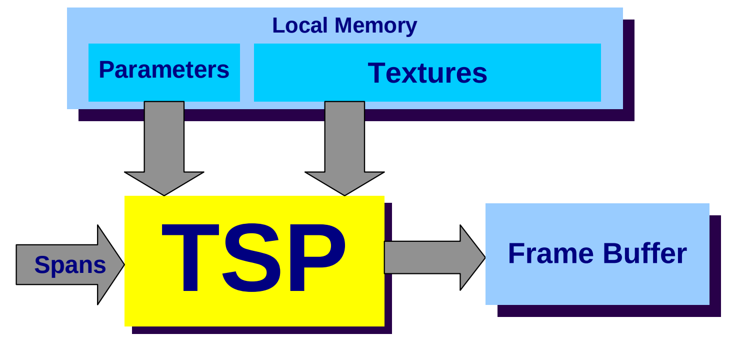


Figure 7: Texture and Shading Processor

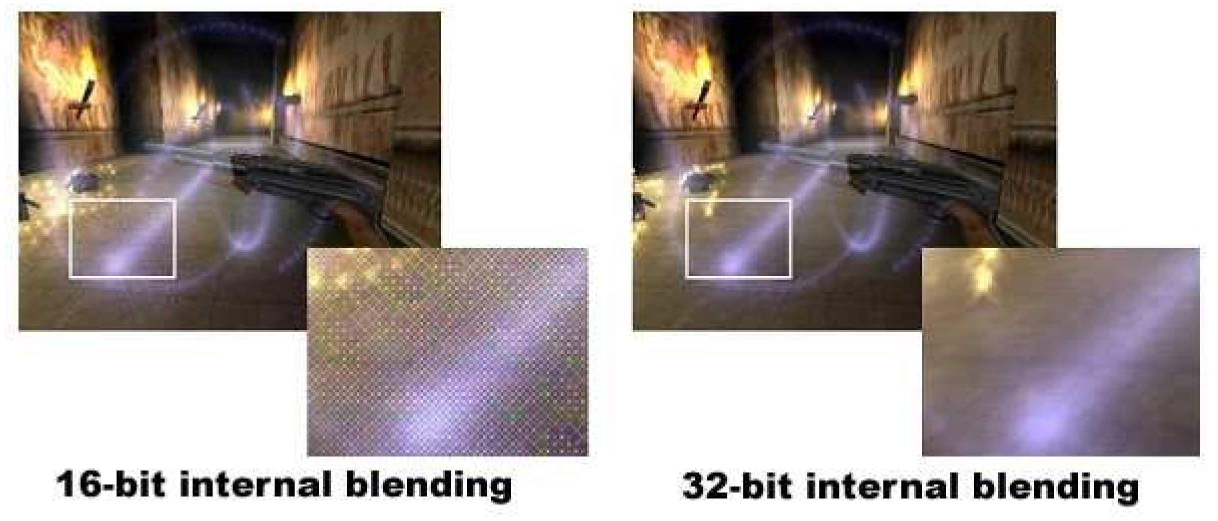
A great feature of the TSP is that all shading and texturing takes place with 32 bits of colour precision, independent of the frame buffer colour depth. This feature is called Internal True Colour™ (more on this later in this document).

3. Internal True Colour™

Internal True Colour™ is a term that refers to the internal processing of all blending operations in 32 bits colour precision. This feature is only possible on tile-based architectures where all colour processing is done in the 32 bits tile buffer.

On immediate mode renderers, image quality is a direct result of the frame buffer colour depth. On a 16 bits colour depth frame buffer, multiple reads and writes will result in a loss of precision for the final colour. This problem usually creates a “banding” effect on the screen. While dithering is often used to try to alleviate this problem, it can potentially create worse result by giving the image a “grainy” look. On POWERVR, each pixel is only written *once* to the frame buffer, hence the final render looks much better.

To illustrate this feature, here are two Serious Sam screenshots taken on a 16-bit and a 32-bit internal colour cards.



Internal True Colour comparison

4. Full Scene Anti-Aliasing

Super-sampling is the POWERVR method for Full Scene Anti-Aliasing; it is one of the best anti-aliasing methods available. It produces crisp results and greatly enhances the overall look of the final render. Super-sampling is the process of rendering a frame at a higher resolution and filtering it down to the frame buffer size. POWERVR supports 2x horizontal, 2x vertical and 4x anti-aliasing.

Because POWERVR is a tile-based renderer, no extra memory is required for the frame and depth buffers. i.e. they will keep their original size during the whole process. What happens in the hardware is that more tiles are rendered to match the super-sampled size but the scaling-down process is transparent. This is a clear advantage compared to immediate more renderers, which have to allocate precious video memory to cater for the super-sampled frame and depth buffers.

The following are two portions of screenshot taken from LucasArts' Escape from Monkey Island. The left image is normal while the right one is 4x super-sampled. Both images were scaled up 2x to allow for better comparison:

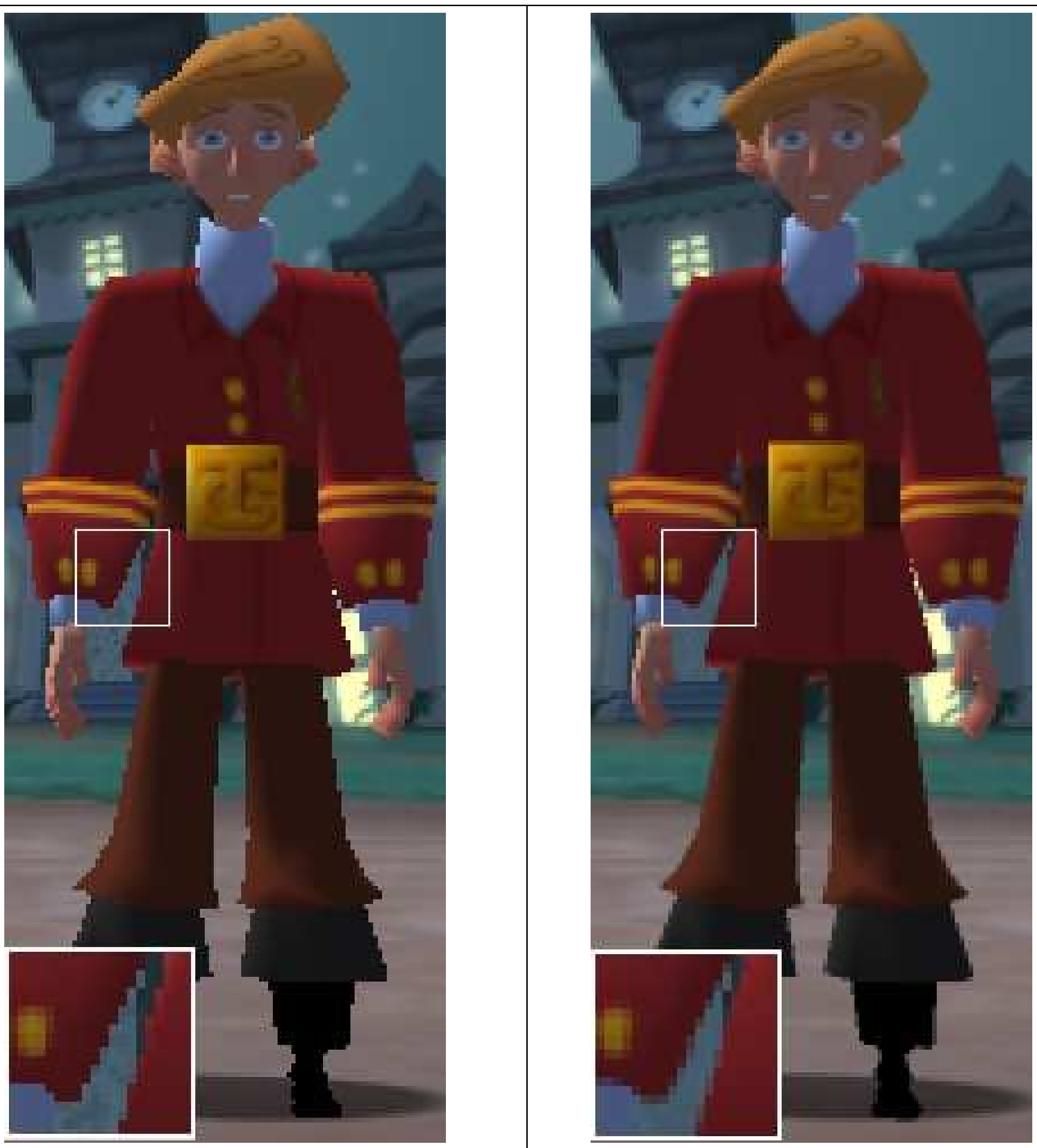


Figure 8: Escape from Monkey Island Normal and 4x FSAA'ed

5. Depth Complexity

What is Depth Complexity?

Depth complexity (also sometimes called "overdraw") is the average number of times pixels overwrite each other in a scene. Depth complexity varies from one game to another, depending on the scene and rendering algorithms used.

Fill rate

Nowadays, fill rate is very often used to compare capabilities of different 3D graphics processors. Most of the time, fill rate is the limiting factor that prevents a scene to be rendered at full speed. In these cases the CPU is idle, waiting for the render to finish before it can send the parameters for the new scene to the 3D processor. A game suffering from this behaviour is called "fill-rate limited."

POWERVR "super" fill rate

Because POWERVR performs HSR up-front in the pipeline, it will only render what is visible on screen. No hidden pixels will be rendered, thus enabling an effective and intelligent use of fill rate. This architecture enables POWERVR to beat the traditional bandwidth barrier and reach "super" fill rates.

Real game example

The following are two screenshots from Quake 3. The first screenshot is a normal screenshot from a scene in Quake 3 while the second renders everything translucently to get an idea of the overdraw of the scene:

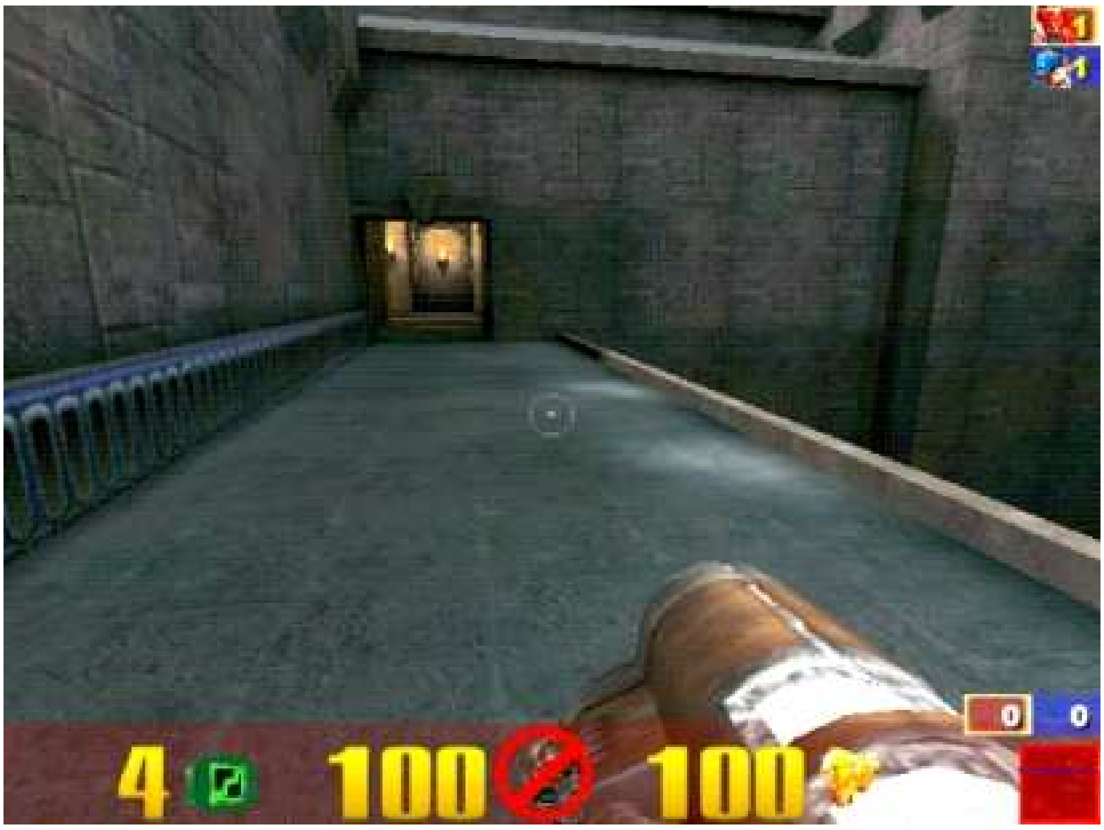


Figure 9: Quake 3 Arena screenshot

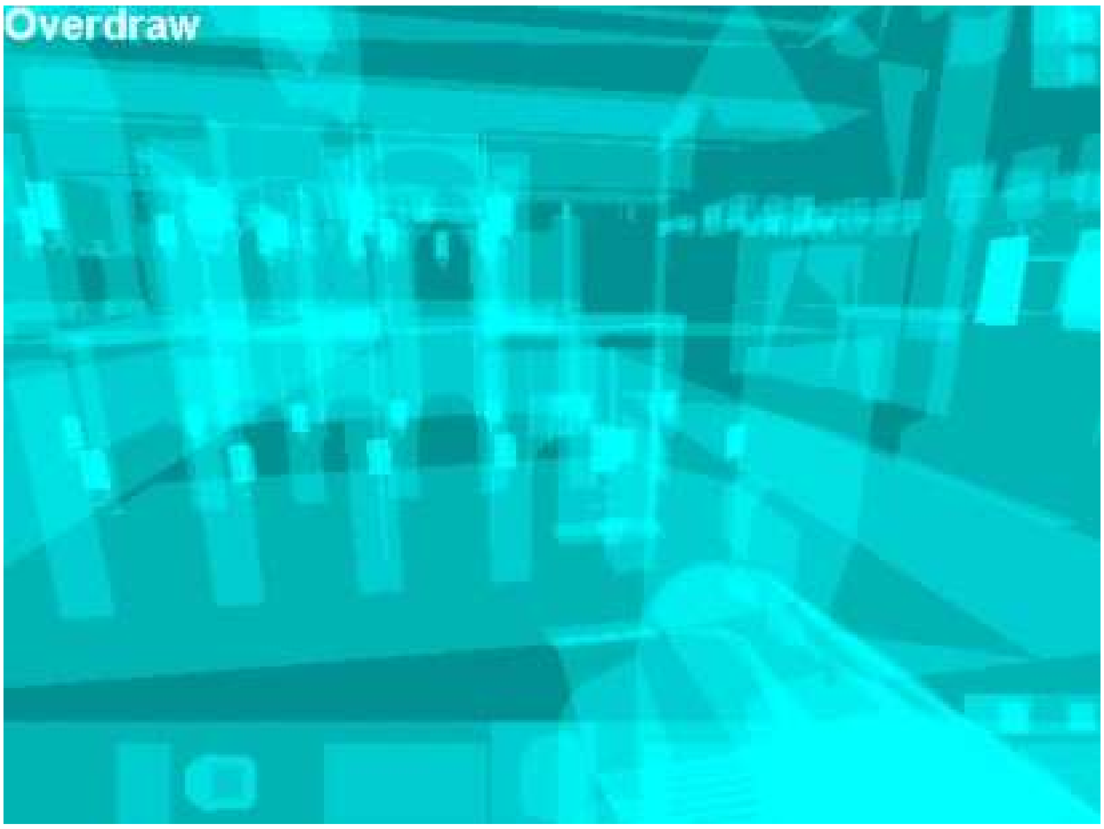


Figure 10: Same Quake 3 Arena screenshot rendered translucently

As you can see there is much more on the screen than one would think there is!

Although Quake 3 uses BSP and portal techniques to attempt to reduce overdraw, the scene complexity is such that overdraw is very often quite high. As an example we measured overdraw of the Quake 3 demo001 at 3.39 (i.e. each pixel on the screen is drawn an average of 3.39 times).

Some immediate mode renderers are implementing some techniques to try to reduce overdraw. Early Z test (the action of testing and rejecting a pixel very early in the pixel pipeline) is one of these techniques but because one cannot predict the rendering order of polygons on the screen, this method is still quite ineffective. The overdraw value for the same scene using early Z test is 3.14, i.e. only a small improvement compared to the normal value.

What about the future?

As game complexity increases, overdraw will too. Far clipping planes will become larger and larger and as such more objects will be rendered onto the screen. Real-world type environments like cities or outdoor terrains cannot benefit from efficient occlusion detection algorithms, and as such will generate a great amount of overdraw which will impact the fill-rate of immediate mode renderers.

6. Features only for Tile Based Render

There are features that naturally scale themselves very well to a tile-based architecture. These optional implementations can be incorporated depending on product design, platform and cost.

True Scalability

POWERVR is a very scalable architecture. For instance MBX is POWERVR implementation for hand-held devices. Sega's Naomi 2 uses two POWERVR chips in parallel and these are used to render alternate tiles in the frame buffer.

Super Fast front-end Super-Sampling FSAA

A tile-based architecture is ideal to perform very fast FSAA. The fetching of samples would be done in the HSR module for optimal image quality and performance. Because the tile buffer memory is very fast, samples can be fetch quicker than on a traditional architecture.

There are other features that scale themselves very well to a tile-based architecture but this document does not cover these ☺.

7. Comparison with brute force

7.1. Traditional renderer

Let's examine the pixel pipeline of a traditional, immediate mode renderer:

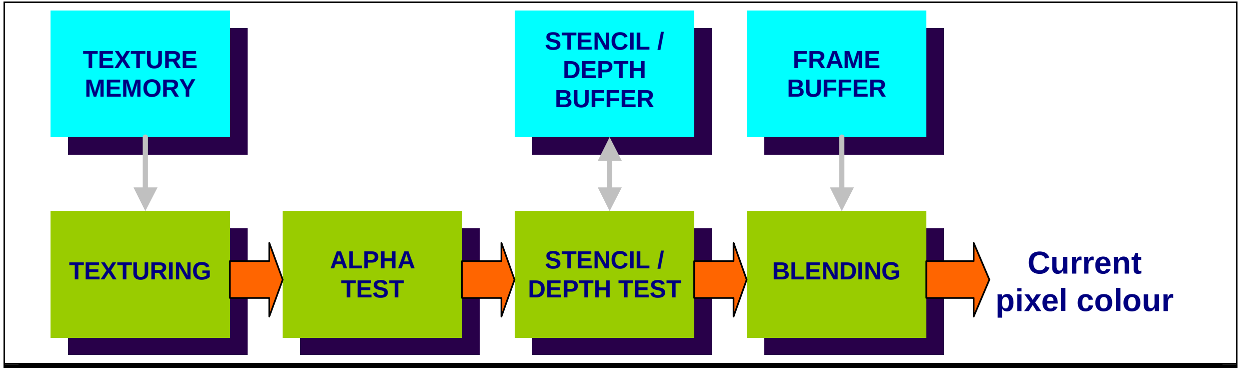


Figure 11: Traditional renderer's pixel pipeline

The most important thing to notice in this diagram is that Texturing is performed *before* the depth test, which means that all pixels will be textured even if they turn out to be masked in the depth buffer (i.e. they fail the depth test). Memory accesses will be performed during the depth test and blending with the frame buffer, even if the current pixel doesn't turn out to be the *final* pixel at this screen location. All these memory accesses are the reason why immediate mode renderers are hitting the bandwidth barrier.

7.2. POWERVR renderer

The following is a simplified pixel pipeline for POWERVR:

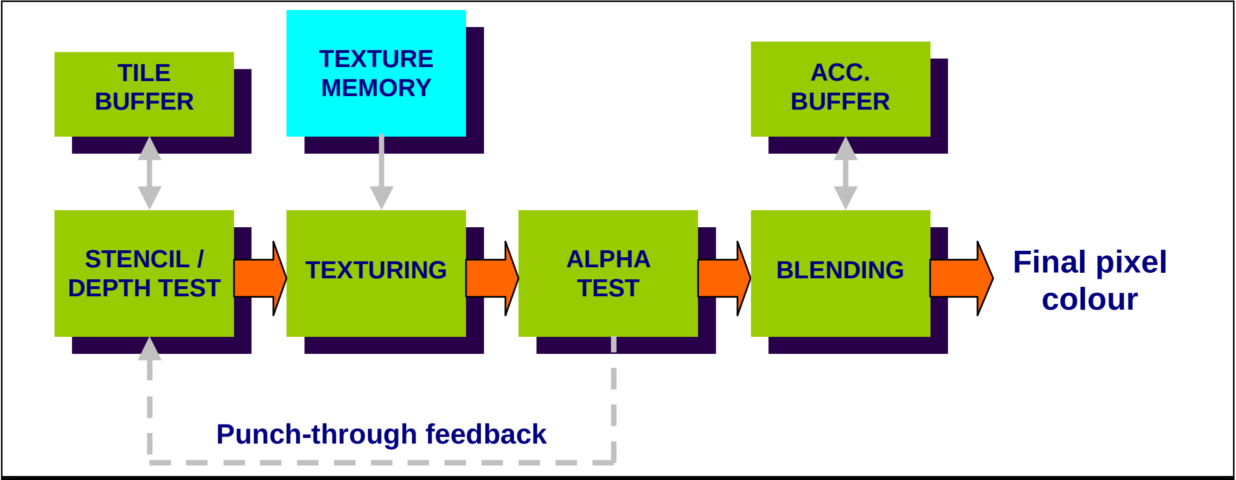


Figure 12: POWERVR pixel pipeline

The main difference resides in the fact that HSR (depth test) is performed up-front. All HSR results are kept in chip (tile) memory. Once visible pixels have been determined, texturing can take place, then alpha testing. The feedback loop to the depth test is there so that pixels succeeding the alpha test are flagged as transparent by the depth test module. Then blending with the current contents of the tile buffer is performed. Finally the pixel colour at the current screen location will be written (once) to the frame buffer.

Localising data by tiling the screen enables the use of ultra-fast on-chip memory and thus to break the traditional memory bandwidth barrier.

7.3. Comparison

The following is a table comparing traditional and tile-based architectures:

Immediate mode rendering	POWERVR Tile-based rendering
"For each polygon process all the pixels"	"For each pixel, process all polygons in a tile"
"Texel fetch for every pixel in a polygon"	"Paint by numbers – fetch only the visible texels"
Multiple frame buffer accesses for blending	Single frame buffer access to output final colour
Expensive design (embedded memory, number of pins…)	Cost-effective architecture

Table 1: Immediate mode rendering vs. POWERVR Tile-based rendering

8. Answering public comments

In this section we'll be trying to answer common questions that sometimes get raised about tile-based rendering or POWERVR products.

8.1. Tile-based render parameters

"As complexity increases, tile-based renderers will run out of space to store scene parameters"

This assumption is usually raised when dealing with tile-based rendering. Because scene-capture renders effectively "capture" the whole scene parameters, some people fear that they will run out of memory to contain all the data.

A key aspect of POWERVR technology is parameter management. Minimising parameter data is an important part of the technology since less memory transfers equates to more performance! Possible techniques to permit efficient storage of parameter data are culling, stripping and indexing. There are different techniques to trade overdraw for parameter space while still retaining some degree of architectural advantage.

9. Conclusion

POWERVR's tile-based rendering allow for great performance and image quality on any platform. Its intelligent architecture minimises external memory accesses and thus manages to break the traditional memory bandwidth barrier. Great features like multitexturing, internal true colour™, bump mapping and texture compression enable current games to reach an unrivalled image quality, even in 16 bits colour depth. This intelligent architecture is affordable thanks to the correct choice of features and a cost-effective design. POWERVR has already solved the memory bandwidth problem when immediate mode renderers are still struggling with it. Future hardware will have to go tile-based if they want to stay competitive, as adding more pipes, memory and even chips will only succeed in dramatically increasing the cost. POWERVR is already proving today that tile-based rendering is a solution for 3D graphics; in the future it will be the only one…