Sergei I. Vyatkin,
Alexander N. Romanyuk,
Oleksandr O. Dudnyk

TILE BASED RENDERING TECHNOLOGY

Abstract

The paper reviews the tile based rendering technology. We consider virtual approach the same as the visual system "Arius". "Arius" combines a good image quality and excellent 3D graphics performance.

Tile based rendering approach takes care of increased scene complexity.

1. Introduction

Any Tile Based Rendering System operates on the divide and conquers principle, breaking the screen up into n x n pixel chunks called tiles. STMicroelectronics announced its KYRO 3D and video accelerator based on Tile Technology [1]. For example, Real Time Visual System "Arius" from CSV Lab. (Novosibirsk) used tiles 8x8 pixels [2, 3].

Each tile contains a small portion of the total image that can be processed independently of, and hence in parallel with, every other portion. The Visual System "Arius" is divided into Geometry, Tile, and Pixel Processors.

Triangles are three dimensional entities, yet the screen on which they'll be displayed is two-dimensional. Getting rid of the third dimension is accomplished by projecting the triangles onto a view plane.

Triangles may be mapped to the whole virtual screen, or to just a small portion, they do allow multiple views of the database to be drawn on the same display.

Triangles that extend beyond the screen boundaries are clipped against the viewing pyramid before their vertices are projected. Thus, the geometry function project triangles onto a plane. The Tile Processor's job is to identify the tiles containing all or part of each triangle.

Pixel Processors work a tile at a time, colouring in the entire pixel in one tile before moving on to the next. The Tile Processor manages tiles for the Pixel Processors. The Tile Processor keeps a list of features for every tile in its Tile Buffer.

A triangle's per-pixel shading, texture, fogging, area coverage, and transparency values are combined in the Pixel Processor to yield the triangle's contribution to the colour of every pixel in the tile. Since more than one triangle may contribute to a pixel's colour, an accumulator at each pixel sums up the contributions from the triangles as they're processed.

Only when the sub pixel masks indicate that every pixel in the tile has been completely covered are the pixel colours ready to be sent to the second buffer-Frame Buffer.

2. Geometry Processor

The Geometry Processor performs the following major functions:

1. Data base primitives are sorted so that only that data which is potentially visible to the viewer is retained for further processing.

2. Primitives are transformed from a three-dimensional representation to a 2D view plane representation in proper perspective.

3. Triangles are projected and clipped to the window boundaries.

4. Triangle colours are modified to simulate sun angle illumination effects and to smoothly blend the transition between levels of detail.

5. The colour and intensity of light sources are computed.

6. Coefficients used to describe the texture for triangles are computed.

Since only a limited number of triangles can be displayed in a given field, it is advantageous to process only those triangles that contribute significantly to the final image. An important observation is that the farther away something is the less apparent detail it has.

The lowest level of detail (LOD) has very few triangles, while the highest LOD has many triangles.

Abrupt changes in the scene, caused by the sudden appearance or disappearance of object in the scene, must be avoided. Transparency blending is a process in which the transparency of a face is smoothly varied from its base value to invisible.

3. Tile Processor

The Tile Processor function identifies the tiles (pixel arrays) containing all or part of each triangle. In addition, the Tile Processor computes a set of edge coefficients for each triangle. A triangle's edge coefficients enabled the Pixel Processor to determine whether or not the triangle encloses a particular pixel.

The Tile Processor's function is to manage all the tiles for the visual system, and coordinate the activities of the Pixel Processor that render the images within the tiles.

The Tile Processor keeps a list of triangles or fragments for every tile. All of the lists are empty at the start of the field interval. Each time it receives a tile list from the fragment generator, the tile processor stores the tile list containing all of the triangles.

The information includes the coefficients of the equations of the triangle's edges, which the fragment generator function calculates from the coordinates of the vertices and the triangle number.

The edge coefficients are especially important; they are used by the Pixel Processor to decide how much of each pixel the triangle covers. The coefficients of the equations of the triangle's edges the best variant is, which the fragment generator calculates directly from edge equations.

In the channel architecture base of processing the polygonal figures prescribed idea of recursive procedure of doing a screen [4-6], localizing internal area of polygonal figures in the process of rasterization. Main distinctive devils of method are a polygon description by kits of direct, getting through ribs, and recursive fission of screen on hutches, which area decreases in 4 with each step of fission times.

Advantages of given approach:

1. Discriminating characteristic of given approach is concluded in that that

exists nearly single-line dependency of speedup factor Ssqt from the amount of processors, this is a feature of optimum system architecture.

2. Square organization a tile-buffer and a frame-buffer strategy reaches speedups for any orientation vectors, in that time, as a single-line organization capable to reach a parallelism for horizontal or vertical vectors only.

Once all of the visible fragments have been entered in the fragment list, the tile processor is ready to feed its Pixel Processors. Fragments in the tile are then sent one by one to the Pixel Processor.

If the fragment does not completely cover the tile, the Tile Processor sends the next fragment, and the next, until all of the pixels have been coloured.

The Pixel Processor is then given another tile to do. Note that it isn't always necessary for a Pixel Processor to process all of a tile's fragments. If a triangle in the foreground completely covers the tile, all of the triangles and fragments behind it are thrown away. Pixel Processor doesn’t work on fragments that can't be seen.

4. Pixel Processor

The Pixel Processor provides the following major functions:

1. Antialiasing
2. LOD blending
3. Shading
4. Texturing
5. Fogging

The Pixel Processor computes the contribution of each tile feature to the currently processed tile taking into account such factors as transparency blending, illumination, fading, antialiasing, texturing, and shading.

Then pixels are stashed in one half the Frame Buffer until a complete image is ready. In the other half of the Frame Buffer is the complete image computed during the previous field interval. The Frame Buffer works only Read/Write mode.

6. Two Variants of Arrangement of the Pixel Processors

Table 1 – analysis variants of arrangement of the Pixel Processors

	Distribution of fragments per processors	One fragment per one processor	One fragment per all M processors
1	Mode of processors	Asynchronous	Synchronous
2	Distribution of banks of Frame Buffer	It is possible Non-Assigned Unit	Assigned Unit
3	Addressing capabilities of Tile Buffer	Support M lists	Support one list
4	Memory of one subpixel processor	n*n*Nsub*L	n*n*Nsub*L/M
5	Effectiveness of load of processors	If M/(n*n) = ¼ about 100%	If M/(n*n) = ¼ about 70%
6	Modularity	Unlimited quantity of processors with linear speed-up if M/(n*n)<=1/4	Limited. Speed-up depends from M/(n*n)

It is possible two variants of arrangement of the Pixel Processors:

1)Each processor evaluates his tile.

2)All processors evaluate one tile.

The result of analysis is shown in the table 1.

Where a Tile is n x n, quantity of processors are M. Nsub - quantity of subpixels into pixel. L - length of word per subpixel.

6. Conclusion

When drawing an image, the average number of faces required filling up a tile is the scene's depth complexity. Depth complexity is an important consideration when designing databases. Tile based rendering approach takes care of increased scene complexity. Tile based rendering makes extremely efficient use of the availably memory bandwidth.

References

1. http://us.st.com/stonline/index.shtml, Tile Based Rendering, White Paper (down load 108 Kb pdf), discusses the techniques used when KYRO performs transform and lighting, hidden surface removal and texturing and shading using a tile based approach.

2. R.I. Velickohatny, S.I. Vyatkin, O.Yu. Ghimautdinov, B.S. Dolgovesov, N.R. Kaipov, A.V. Romanovsky, S.E. Chizhick, “ARIUS” - Family of real-time 3D

graphics systems for PC platforms, Graphicon’97 Proceedings, S. Klimenko et al. (Eds). 132-135, 1997.

3. S.I. Vyatkin, B.S. Dolgovesov, N.R. Kaipov, S.E. Chizhick, Tile Arhitecture Based on DSP//Autometry - N 1. 1999.

4. S.I. Vyatkin, B.S. Dolgovesov, B.S. Mazurok et al. An Effective Rasterization Method for Real-Time Computer System Visualization // Autometry - N 5. 1993.

5. S.I. Vyatkin, B.S. Dolgovesov, A.F. Rozhkov et al. Algorithms of Visualization for Real-Time Visual Systems // Graphicon’95 Proceedings, S. Klimenko et al. (Eds). St-Petersburg 1995.

6. A.E. Asmus, A.I. Bogomyakov, S.I. Vyatkin et al. Video-Processor of Computer System Visualization “Albatross” // Autometry - N 6. 1994.