Parallel Architecture and Algorithms for Real-Time Synthesis of High-Quality Images Using Voxel-Based Surfaces

Sergei I. Vyatkin, Boris. S. Dolgovesov, Alexander V. Yesin, Alexander A.Zigach, Sergei E. Chizhik
Institute of Automation and Electrometry, Siberian Division of Russian Academy of Sciences
Computer-based Visualization Systems Laboratory
Novosibirsk, Russia

Abstract

We present a new project on volume modeling that combines volume representations by voxel data and by continuous real functions. The design principles and architectural features of the first version of the real-time visualization system were developed at the beginning of 90s. A novel graphics engine, being designed in Computer-based Visualization Systems Laboratory (Novosibirsk, Russia) is based on application specific integrated circuits (ASICs) and characterized by high parallelism and homogeneity of vector computations which result in superior performance along with comparatively low price.
The project focuses on applications in such areas as:

- scientific visualization;
- volumetric CAD, MCAD, DCC (Digital Content Creation);
- flight simulators;
- rapid development of visual scenes for games and entertainment;
- medicine (virtual endoscopy, surgery support and training).

The goal of the project is to design a new modeling and visualization system (integrated software/hardware computing system). In the proposed system the following main problems have to be solved: combination of implicit freeform surfaces, height and volume data; atmospheric effects (fog, haze, rain, snow, layered atmospheric effects, round earth, sky illumination); animation, deformation and 3D metamorphosis. The possibility to render both implicit surfaces and volumetric data is defined by a 3D grid with parameters such as opacity, color, etc. is a principal feature of the proposed system.

Keywords: Free-Forms, Perturbation Implicit Functions, Perturbation Scalar Functions, Voxels.

1. INTRODUCTION

Users interested in simulation, for example, will particularly appreciate the translucency mapping, fog and haze effects, and anti-aliased points (for star fields and landing lights). Instead of flying a simulated airplane over flat-shaded polygons that attempt to represent a terrain, you could fly that airplane through a cloud bank and then pop on out to fly over a sunlit, textured terrain that includes trees, roads, and billboards captured from real-word pictures.
Computational chemists and molecular biologists can work interactively with larger molecules than ever before; the quadric primitive will help make the generation of those models faster [1]. Industrial designers and mechanical engineers, for whom the arbitrary clipping planes, constructive solid geometry have the most application, can design far more interactively and can more closely simulate the final look of their products.
Volume visualization techniques are becoming available and popular. This is due to the increasing availability of scientific data generated by a variety of computer simulations, medical data obtained by scanners, and geological, oceanographic, and meteorological data collected from various sensors, and also due to the decreasing costs of high-performance computing required for volume rendering [2].

Image generators traditionally use polygons as database primitives. It is difficult to use polygons to make convincing models of clouds, smoke, trees and anything else for which no simple surface representation exists. The analysis of possible directions of evolution of a real-time visualization systems shows that the easiest way to improve picture quality, i.e. to increase number of polygons rendered per frame, is not the most effective one. Going this way, the qualitative changes can be hardly achieved. In part, the high performance ratings can be attributed to a change in the basis for measurement: Early machines were rated on the numbers of independent polygons or triangles generated per second. The next machines use polygon or triangle meshes for their benchmarks, and these meshes can be generated more quickly than can individual polygons or triangles. Even so, the performance numbers are significantly higher. And the move to meshes is a reasonable transition in light of the fact that this generation’s modeling primitive is changing. As modeling moves away from user-defined polygons and toward NURBS (non-uniform rational B-splines), we’re beginning to see user-defined parametric surfaces that are automatically diced into polygonal or triangular meshes when they’re rendered.
Real-time computer graphics oriented to 3-D scene visualization has attained appreciable success nowadays. Though a sufficiently high realism of real-time scene imaging has been attained, some problems are still present (e.g., imaging of large terrain regions), where it is necessary to store and visualize scenes containing a greater number of polygons than it is implemented in the present-day systems. For instance, the problem of imaging of mountains requires for initial description hundreds of thousands of polygons. On the other hand, exact modeling of shapes of the car, airplane, submarine frames requires thousands of spline-surfaces (curvilinear areas defined by polynomial functions) whereas the case of definition by polygons will require tens and sometimes hundreds of thousands of polygons. Moreover, polygonal 3-D graphics with scanning of polygons in the image plane is not three-dimensional in the full sense of the word. Information presented to the user in such an approach is incomplete. The main point is the absence of information on object depth. This case implies not the absence of the Z-coordinate of the surface point but the absence of information about the beam passing through the object.
We suggest expanding the notion of primitives and making it possible to process them by easy and effective method without approximation by polygons. For MCAD applications (and Digital Content Creation as well), we would have to seamlessly integrate

voxels and freeform surfaces (embed an opaque surface object inside the volume), which means reading and writing a Z-depth buffer. For flight simulation applications, we would have to generate voxel-based terrain [3]. While traditional machine's sophistication is most notable at the "back-end" (or the image-space end of the graphics pipeline), Voxel-Volumes (VxVl) graphics are highly sophisticated at the "fronted" (or the object-space end of the pipeline), where the data is manipulated. That's where the VxVl is most programmable; and that has allowed VxVl to offer some unique features for analysis and design.

We developed new generation visualization system for volume-oriented rendering. The system provides high level of versatility and high image quality. It is based on:

a) New surface representation by free-form surfaces without using polynomials of high degree or polygonal/patch approximation and representation volume areas by arrays;

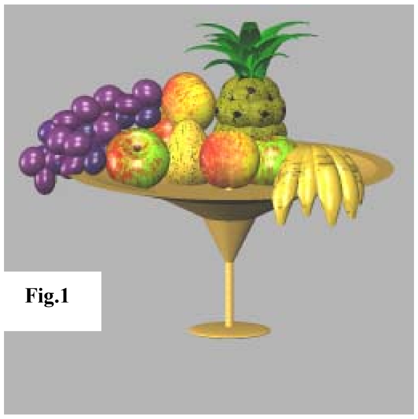


Fig.1

b) Efficient algorithms of rasterization.

2. FREE-FORM REPRESENTATION

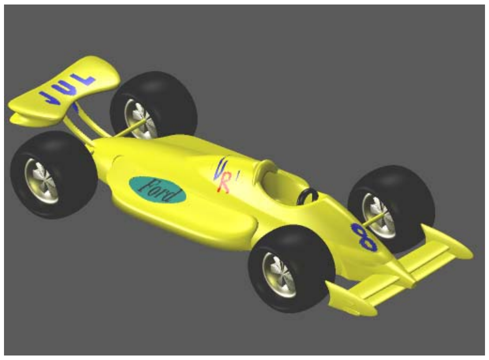
Proposed surface representation uses quadrics-surfaces of second order as basic primitives. Quadrics are foundation for free-form surfaces. The function of quadrics with perturbation functions defines a free-form object by a real continuous function of several variables as $F(X) \geq 0$ where X is a position vector of a point in Euclidean space [4-8]. As an alternative representation for surfaces we have been exploring volumes. A volume has some major advantages: they can represent the interior of objects and not only the outer shell, like surfaces do. Rendering and processing do not depend of the object's complexity, they depend only on volume resolution.

2.1 Supremacy of proposed technology

1) For visualizations of high-realistic expressing is realized transition from the scan of two-dimensional space to three-dimensional.
2) Task four types of the free forms of objects alongside with polygonal: 1. Spline-surfaces. 2. Implicit perturbation functions. 3. Scalar perturbation functions. 4. 3D-data of volume (voxel's array).
3) Rasterization of enumerating primitives without partitioning them on polygons**.**

2.2 Perturbation Implicit Functions

It is possible to describe complex geometry forms by specifying surface deviation function (of second order) in addition to surface base function of second order. Generally a function F(x,y,z)



specifies surface of second order – quadric. Free-form surface is a composition of surface base function and surface deviation (deformation) function [9].
Problem of object construction is reduced to finding appropriate deviation function for quadric – surface base function. There is no need to approximate surface by polygons, patches (B-splines etc). Surfaces produced by composition of base and deviation function are smooth (Fig. 1). Often only few of such compositions are needed to construct complex surfaces. Defining volumes by free-form surfaces allows one to compress data base size 500 times as compared with polygonal representation (Fig. 2).

2.3 Perturbation Scalar Functions

Rendering photo-realistic, complex terrain features at interactive rates requires innovative techniques. A polygonal model and geometric pipeline can be used but this introduces massive storage requirements and, ideally, a parallel implementation of the algorithm. Using a voxel-based model, however, can achieve the same results at a much lower hardware requirement. As a software solution, the method is portable so it can be integrated into any flight simulation system regardless of hardware architecture. An objects with Perturbation Scalar Functions like sculpture the triangle models (Fig. 3) or terrain (Fig. 4) are coded as differential height map, i.e. the carrier surface is defined by algebraic means and only deviation from this basic surface is stored in the each node. Such a modeling method simplifies creation of smooth detail levels and shading. The data of height grid is not subject of geometry transformations as vertices do. The geometry transformations are only required for

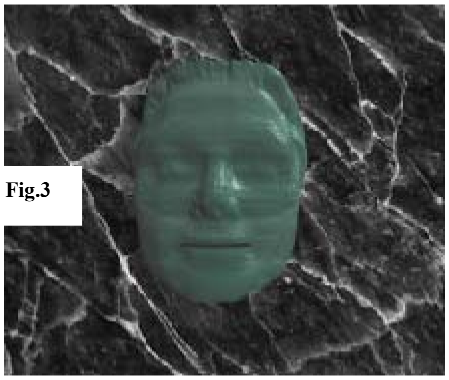


Fig.3

the carrier surface. For switching between levels of detail the same procedure is used as for the ordinary texture [3,9]. The effect is that this approach allows construct objects with highly complex topology. Time of landscape processing and rendering does not depend on elevation map resolution.

Fig.2

3. DEFINITION OF VOLUME AREAS BY 3D SCALAR ARRAYS

In addition, while solving the descriptive function in the form of inequality $F(X) \geq 0$, we can visualize not only the surface but also the internal structure of the object. To visualize semi-transparent structures with internal density distribution, 3D textures, boundary and internal voxels are used during rendering. It is possible to render such structures when they are bounded by non-trivial surfaces. Light penetration through materials and semi-transparency is modeled using color blending. In general 3D texture in this case can be considered as separate object returning in all space positive result when requested about intersection. Mapping texture to object can be seen as intersection of the object with texture object, color and transparencies are taken not from object attributes but from texture map. Texture is generated at visualization time using three-dimensional density map. Transparency of elements is computed depending on density value and density gradient, which used as normal to iso-surface.

4. RASTERIZATION TECHNIQUE

In this paper we used the multilevel ray casting algorithm [3, 9], which performs efficient search for volume elements - voxels which participate in image generation. At the first step of recursion, the initial viewing pyramid is divided into four smaller subpyramids in the screen plane. Using results of test, we perform division of subpyramids that fall within the quadric completely or, probably, partially, and the knowingly external subpyramids are eliminated from processing. If the intersection is determined, then the subpyramid is subject to the next recursion level. Subpyramids that do not intersect with the object are not subject to further immersion to recursion, this corresponds to elimination from consideration of square screen spaces which the subpyramid (and, consequently, the object surface) is not mapped to. The viewing pyramid is subdivided until reaching the maximal set level of recursion. The technique has an advantage that allows discard of large parts of empty space at an early stage. While searching for voxels containing the imaging object surface areas, the pyramidal space is traversed along the quaternary tree whose leaves are roots of binary trees. Masking is used during traverse of the tree in the case of opaque objects. The multilevel ray casting technique allows us to determine effectively and quickly belonging of rays of different levels (pyramids) to surfaces, and discard space regions outside the objects.

While visualizing the surfaces a test is verified for belonging of only intersected voxels (unit volume elements), external and internal voxels are discarded. To improve imaging realism and extend the class of objects imaged (translucent structures with internal density distribution, 3-D textures), it is necessary to image the internal translucent object structure. For this purpose, not only voxels lying on the surface but also voxels inside the object must participate in imaging. Therefore, while dividing the volume the internal object parts are not discarded, for them algorithm recursion is performed further. Scanning of the scene along the Z-coordinate, which corresponds to scanning of the volume through the depth, is not interrupted upon meeting the surface but is continued until the volume is scanned completely or a certain value of transparency higher than a threshold value is stored. To reduce the computation time the algorithm is adapted to quick passage of homogeneous spaces of objects, for which it is unnecessary to scan the volume completely reaching the last recursion level. It is necessary to “skip” empty or homogeneous spaces along the Z-coordinate and immediately calculate the color and the total transparency. Since ray passage through empty space makes no contribution to the final image, then the skip of the empty space is able to make the processing substantially fast and does not affect the image quality

5. ARCHITECTURE

Method of multisequencing is insufficient, if canonical scheme is multiplied by means of repetitions of all components [10]. Unfortunately, the replicated pipelines cannot work independently of each other. Two possible interconnects are used. The first interconnect is used to route primitives to the appropriate rasterizers (Silicon Graphics’ Reality Engine). The other possibility is, to route fragments after rasterization to the appropriate memory location (Evans and Sutherland’ Freedom System). Fortunately, the replicated pipelines can work independently of each other in our approach. The main merit of our approach is the reduction of the load on the geometry processor and decrease of data flow from it to the video processor.

5.1 Chip-Set Voxel-Volumes

The chip-set Voxel-Volumes consists of four ASIC’s - Application Specific Integrated Circuits (ASICs— often referred to as “microchips”) (if have geometry processor-GP) or from three ASIC’s in the absence in maximum configuration. Minimum configuration is one ASIC of pixel processor. Let’s enumerate the order all chips (Fig. 5):

- Geometry Set-up Engine.
- Terr_Pipeline.
- Quad_Pipeline.
- Pixel Processor.

The first stage of the VxVl is a processor of geometric transformations. The second stage is a pyramid traversal processor, which consists of pipelines (terrain_pipeline, quadric_pipeline, polygon_pipeline) of uniform cellular processors. Rendering, the process of coloring in the surfaces supplied by the traversal subsystem to create the final image, takes place in the Pixel Processor (PP).

On entry Geometry Set-up Engine enter data from Host_bus or AGP_bus, which reception capacity must be not less 1.2 Gbyte/sec. Functions of given chip - geometric transformations of primitives. From leaving a microchip data enter on the bus Par_bus , on entering a Channel of processing of terrain (Terr_Pipeline), Channel of processing of quadrics (Quad_Pipeline) and on entering an Pixel Processor, if primitives are triangles. From leaving the microchips of Channel of processing of terrain (Terr_Pipeline) and Channel of processing of quadrics (Quad_Pipeline) on entering to Pixel Processor enter data (pixels), which sending rate must be not less 3.2Gbyte/sec. The Pixel Processor does not only implement surface boundaries smoothing, but it also handles such effects as smooth shading (Phong interpolation), fading, texturing, color blending, and translucency.

Alongside with data chips a charge of accelerator has microchips of memory for keeping data height maps (Hight_data Memory) and memory of frame, coordinates Z, texture memory (Frame/Z/Texture_Memory). There is possible a variant of accomodation of frame memory and coordinate Z inside the chip of Pixel Processor. From leaving of Pixel Processor video-signal in formats VGA, TV or DVI enters Display Device.

5.1.1 Channel of processing of terrain (Terr_Pipeline)

Given microchip is intended for processing the objects of the free-forms with scalar perturbation functions and voxel-based terrain, which is a private event of the free forms (Fig. 6).

On entering a Channel of processing of terrain (Terr_Pipeline) data enters from Par_bus. From leaving a microchip data (pixels) enter the bus Pix_channel, on entering to Pixel Processor, which sending rate must be not less 3.2Gbyte/sec. Except this the given microchip is connected with the memory for keeping data height maps (High_data Memory). Let's enumerate the order of all main blocks of microchip:

- Input Data Interface (Par_bus Interface).
- Preprocessor (Ray bunch Set-up Engine).
- Processor of processing the rays of binary tree (Ray_bunch Procesor).
- Cache data maps of heights 128 Kbyte (Cache Lev1).
- Data Height Maps Interface (Hight_data Memory Interface).
- Output Data Interface (Pix_channel Interface).

To send 2-bytes map data of heights, for instance 40x40 kilometers with the permit in 1 metre it is necessary to have reception capacity of bus Hightdata _bus not less than 300 Gbyte/sec. that in principle for a present-day is not real. So it is used contiguity to the account of presence Cache Lev1 that allows to decrease a given value in 3 times. However it is insufficient, so it is used for compression information by the method of certain sample of values of heights and exact standard of judgement on crossing the rays with terrain, this must reduce reception capacity of bus Hightdata _bus before 16 Gbyte/sec that already is a real value.

Integration of given chip - an order 20 million transistors.

5.1.2 Channel of processing of quadrics (Quad_Pipeline)

Given microchip is intended for processing the objects of the free forms with analytical perturbation functions and quadrics, which are a private event of the free forms and algebraic pathes.

On entering a Channel of processing of quadrics (Quad_Pipeline) data enters from Par_bus. From leaving a microchip data (pixels) enter the bus Pix_channel , on entering to Pixel Processor, which sending rate must be not less 3.2Gbyte/sec. Let's enumerate the order of all main blocks of microchip:

- Input Data Interface (Par_bus Interface).
- Processor of quaternary tree (Quad_tree Processor).
- Processor of binary tree (Binary_tree Processor).
- Output Data Interface (Pix_channel Interface).

Processor of quaternary searching tree (Quad_tree Processor)

On the first step of recursion source pyramid of visibility is divided into four smaller subpyramids in screen planes. Test is executed for each new subpyramid on intersection with the object. On the grounds of results of this test is realized fission of subpyramids, which lie inwardly of surface wholly or, possible, partly, but undoubtedly external of subpyramid are excluded from processing. Fission of pyramid visibility leads until it greatly stated recursion level is not reached.

Processor of binary tree (Binary_tree Procesor)

To raise realism of expressing and increase a class of displayed objects (translucent structures with internal sharing density, 3D textures) it is necessary to display an internal translucent structure of object. For this in imaging must participate not only voxels, which rest upon surfaces, but also those, which are inside. Consequently, when doing a volume internal parts of the object are not rejected, for they are conducted most further recursion of algorithm. At the scan a scene on Z coordinate that corresponds the scan a volume in the depth, scan is not broken when meeting with the surface, but works hereinafter, while completely is not scanned a volume or will not be accumulated to definite sign of transparency of greater certain threshold value. For reducing a time of calculations an algorithm is adapted to the quick passing of uniform areas of objects. For which at all unnecessary completely scan a volume, getting to the last recursion level, but follows race through started or uniform area on Z coordinate, and immediately calculate a colour and general transparency.

Integration of given chip - an order 12 million transistors.

5.1.3 Pixel Processor

Given microchip is intended for processing the triangles, removing the invisible surfaces, texturing and shading, with provision for the mist and without it (Fig. 7).

On entering to Pixel Processor, data enters on Par_bus bus (triangles) and on the bus Pix_channel 0 from the Channel of processing of terrain (Terr_Pipeline), but on the bus Pix_channel 1 from the Channel of processing of quadrics (Quad_Pipeline) - a pixel. Let's enumerate the order of all main blocks of microchip:

- Channel of processing the triangles (Polygon_Pipeline).
- Texture Memory Interface (Texture Memory Interface).
- Block of calculation of colour (Shader).
- Block of calculation of texture (Texture calculation).
- Block of removing the invisible surfaces on the algorithm Z-buffer.
- Frame Buffer Interface.

Polygon_Pipeline

In the channel architecture base of processing the polygons prescribed idea of recursive procedure of doing a screen [11,12], localizing internal area of polygons in the process of rasterization. Main distinctive features of method are a polygon description by kits of direct, getting through ribs, and recursive division of screen on cells, which area decreases in 4 with each step of division times (Quaternary Searching Tree). For eliminating the defects, in accordance with discrete television raster, determination an accesories of element of expressing to the polygon is realized on the increased permit of raster: each pixel is divided into 16 subpixels. As a result is formed subpixel code of fragment, which is compared to the code of mask of cell, formed from earlier processing polygons.

Several types of parallelism are used in the channel of processing the polygons:

· Pixel level and primitive level realize word aligned.
· To the account asynchronous functioning the pixel channels occurs pixel aligned, which carries out the most effect for 2D array and ensures plural-primitive parallelism level (rasterization of several primitives or several fragments of one big primitive parallel).
· Technology of rarefying raster with the aperture is used.
Advantages of given approach:
1. Discriminating characteristic of given approach is concluded in the fact that there is nearly single-line dependency of speedup factor from the amount of processors, this is a feature of optimum system architecture.
2. Square organization 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.
Integration of given chip - an order 15 million transistors.

6. CONCLUSION

Absolutely new technology is a visualization of the free forms as

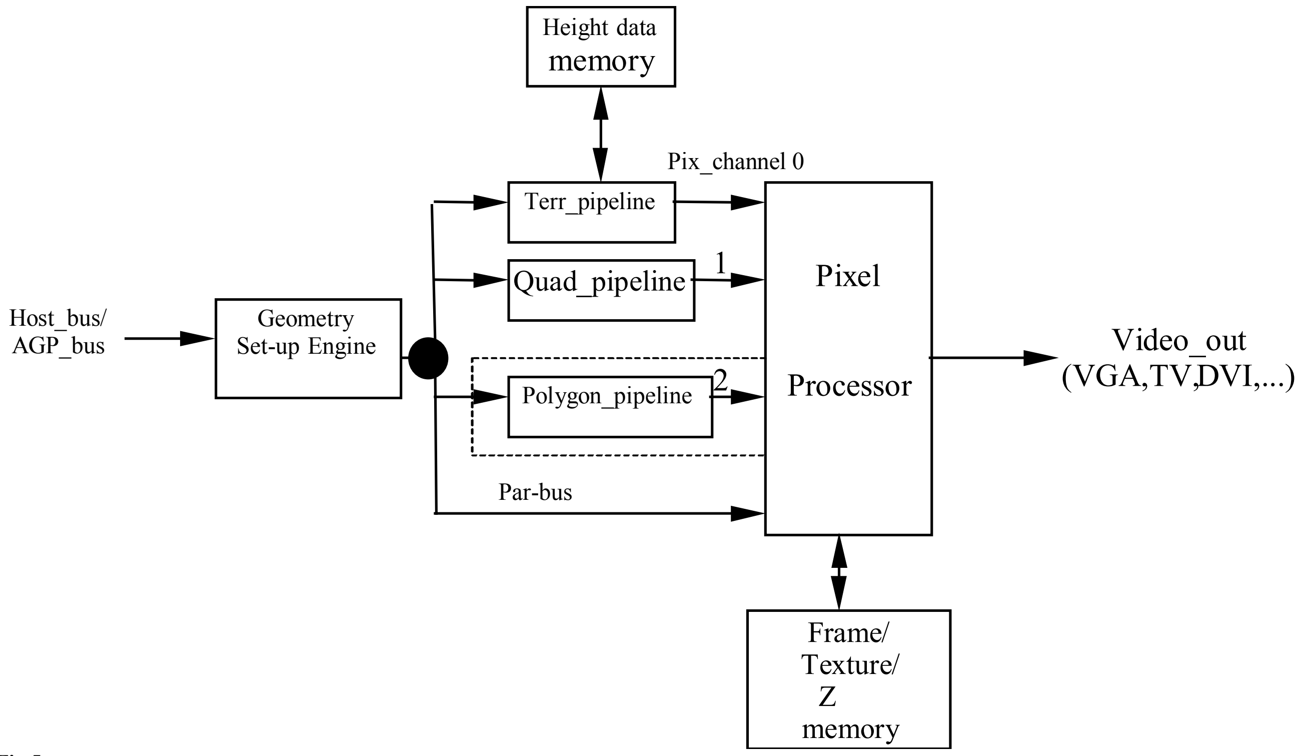


Fig.5

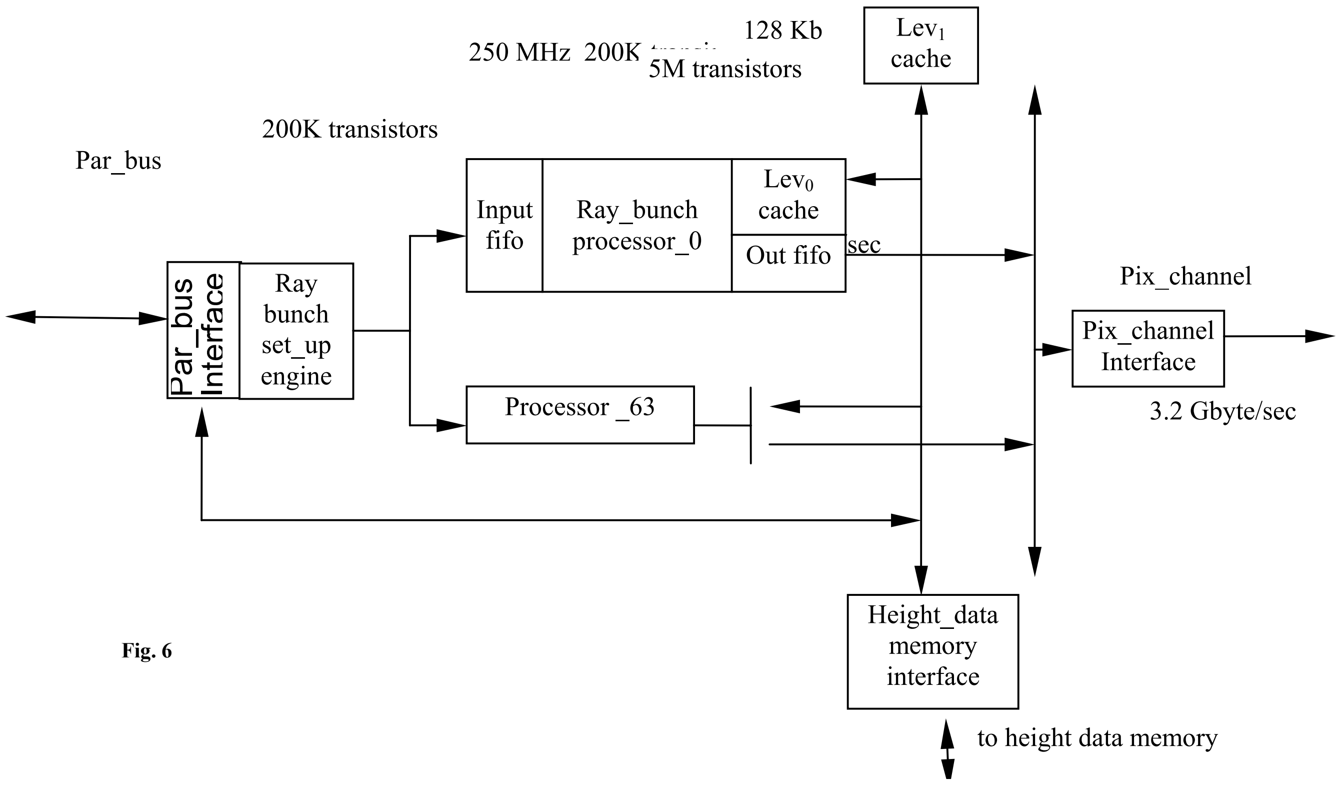

250 MHz 200K
128 Kb
5M transistors
Lev$_1$ cache
200K transistors
Par_bus
Input fifo
Ray_bunch processor_0
Lev$_0$ cache
Out fifo
sec
Pix_channel
Par_bus Interface
Ray bunch set_up engine
Pix_channel Interface
Processor _63
3.2 Gbyte/sec
Height_data memory interface
to height data memory

Fig. 6

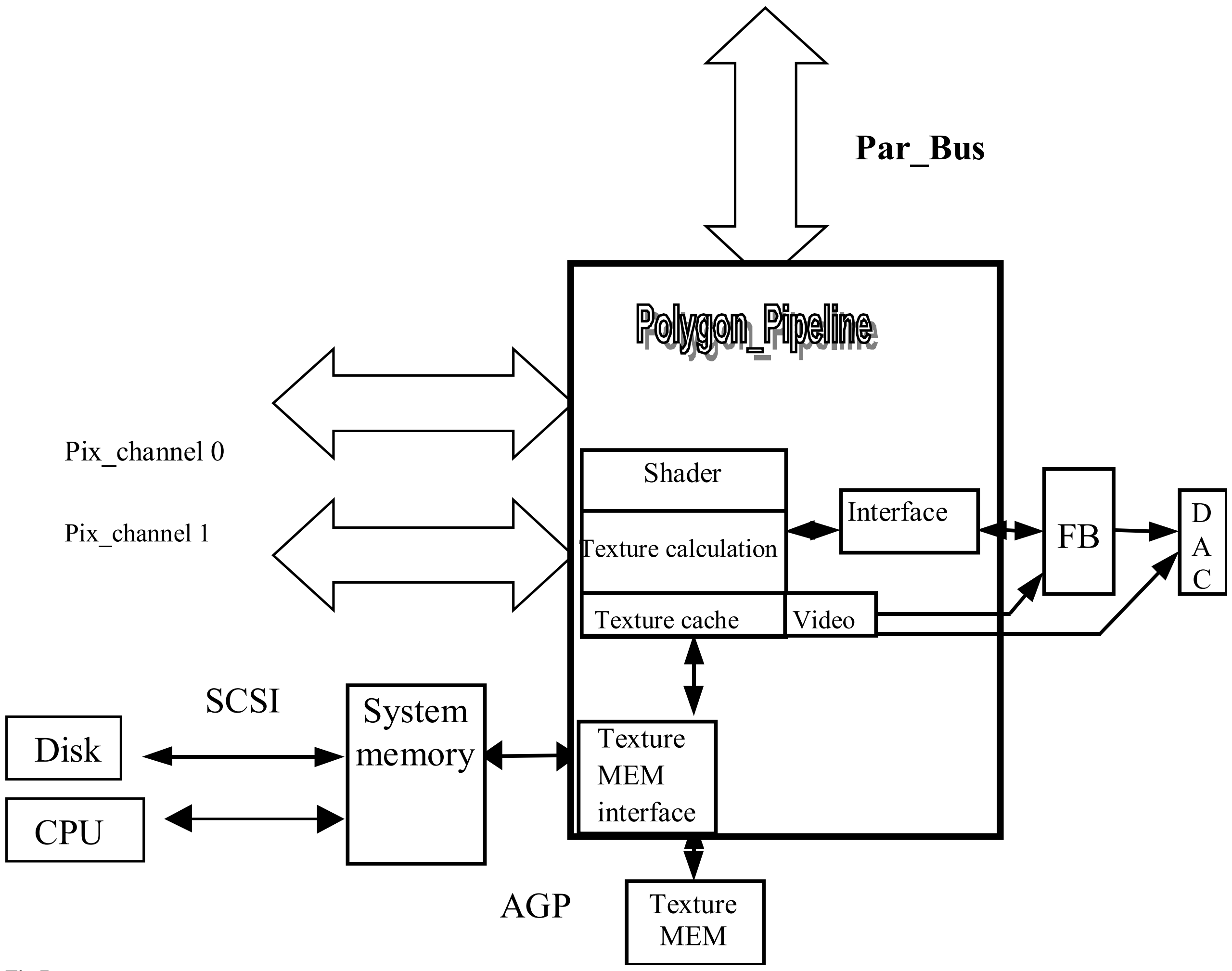

Par_Bus
Polygon_Pipeline
Pix_channel 0
Pix_channel 1
Shader
Texture calculation
Interface
FB
D A C
Texture cache
Video
SCSI
System memory
Disk
CPU
Texture MEM interface
AGP
Texture MEM

Fig.7

7. References

1. **Fuchs H., Poulton J., Eyles J. et al.** Pixel-Planes 5: A heterogeneous multiprocessor graphics system using processor-enhanced memories // Computer Graphics 1989. N. 3, P. 79.
2. **Pfister H. et al.** A real-time volume rendering architecture using an adaptive resampling scheme for parallel and perspective projections //Proc. of 1998 IEEE Symposium on Volume Visualization. Research Triangle Park, North Carolina, 1998. P. 31.
3. **Vyatkin S.I., Dolgovesov B.S., Ovechkin V.V. et al.** Photorealistic imaging of digital terrains, freeforms and thematic textures in realtime visualization system Voxel-Volumes // GraphiCon'97. Moscow.
4. **Pasko A., Adzhiev V., Sourin, A.,** et al. Function representation in geometric modeling: concepts, implementation and applications //The Visual Computer, 11,6 (429-446), 1995.
5. **V. Savchenko, A. Pasko, O. Okunev and T. Kunii**, "Function representation of solids reconstructed from scattered surface points and contours", Computer Graphics Forum, 14,4 (181-188), 1995.
6. **A.A. Pasko and V.V. Savchenko**, "Blending operations for the functionally based constructive geometry", Set-theoretic Solid Modelling: Techniques and Applications, CSG 94 Conference Proceedings, Information Geometers, Winchester, UK, (151-161), 1994.
7. **V.V. Savchenko, A.A. Pasko, T.L. Kunii, A.V. Savchenko,** "Feature based sculpting of functionally defined 3D geometric objects", Multimedia Modeling. Towards Information Superhighway, T.S. Chua, H.K. Pung and T.L. Kunii (Eds.), World Scientific, Singapore, (341-348), 1995.
8. **Savchenko V.V., Pasko A.A. Vyatkin S.I. et al.** New approach in geometric modeling: distributed and hardware implementation perspectives.// International Conference on Computers and Devices for Communication CODEC-98. Calcutta, India, 1998. P. 285.
9. **S.I. Vyatkin, B.S. Dolgovesov, A.V. Yesin et al.** Voxel Volumes Volume-Oriented Visualization System // Computer Graphics and Geometry, Internet Journal http://cg-g1.newmail.ru/smi99_1/VYATKIN.htm
10. **W. Straber, A. Schilling, G. Knittel**," High Performance Graphics Architectures", // Graphicon'95 Proceedings, S. Klimenko et al. (Eds). St-Petersburg 1995.
11. **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
12. **A.E. Asmus, A.I. Bogomyakov, S.I. Vyatkin et al.** Video-Processor of Computer System Visualization "Albatross" // Autometry - N 6. 1994

Параллельная архитектура и алгоритмы для визуализации высокореалистичных изображений с использованием воксельно-базируемых поверхностей

При сканировании двумерного пространства нельзя получить полноценного трехмерного изображения. Полигональная трехмерная графика со сканированием полигонов в плоскости изображения не является трехмерной в полном смысле этого слова. Информация, которая предоставляется пользователю в такой технологии – неполная. Главное – это отсутствие информации о глубине объекта, имеется ввиду не отсутствие Z – координаты точки поверхности, а отсутствие информации о луче, проходящем сквозь объект. В данной работе рассматриваются способы задания и алгоритмы визуализации свободных форм нескольких видов. Показано преимущество задания примитивов описывающей функцией в виде неравенства $F(X) \geq 0$, с помощью которой можно визуализировать не только поверхность, но и внутреннюю структуру объекта. Также приведена архитектура на базе заказных микросхем системы Voxel-Volumes.

Сведения об авторах:

Лаборатория Компьютерных Систем Визуализации, ИАиЭ СО РАН.
Россия 630090, Новосибирск, пр. Коптюга, 1.

телефон: (+7-3832) 33-36-30

н.с. Вяткин С.И., sivser@mail.ru,
зав. лаб. Долговесов Б.С., bsd@iae.nsk.su,
инж. Есин А.В., esin@iae.nsk.su
инж. Жигач А.А., raven@woland.afti.nsu.ru
вед. инж. Чижик С.Е., cse@iae.nsk.su.