# A Multi-Processor Workstation with a Logic-Enhanced Distributed Frame Buffer

*Frederik W. Jansen*

*Department of Industrial Design*
*Delft University of Technology*
*The Netherlands*

A graphics workstation should offer both a wide variety of 2D and 3D real-time display functions as well as a programmable parallel-processing capacity for large processing tasks. A system concept is proposed that meets these requirements by offering a multi-processor configuration with general-purpose programmable processors, enhanced with specific logic that can perform for each node a large number of simple pixel operations in parallel.

## 1. Introduction.

Developments in graphics accelerators and logic-enhanced frame buffer systems have shown an increased performance in real-time display [Clark, 1982], [Fuchs et al., 1985]. However, this performance has only be achieved for a rather limited class of object representations and shading techniques. For interactive display and design of 3D objects, both a real-time display and a high-quality rendering mode are desired. Recently, several multi-processor architectures have been published that offer an improved functionality at nearly real-time [Niimi et al., 1984], [Torborg, 1987], [Denault et al., 1987] or a full-functionality and programmability at moderate speed [Sutherland, 1986], [Potmesil et al., 1987]. Most desirable, however, would be an hybrid architecture that combines the full flexibility of a multi-processor configuration, built with general-purpose processors, and the real-time performance of the logic-enhanced frame-buffer systems.

For the design of a graphics workstation, a good understanding of both the software and the hardware is important to achieve a good tuning between the software functionality and the hardware support. However, it is often difficult to obtain a good understanding of both the software and the hardware world, and to find an optimum for both. It is the intention of this contribution to provide a frame work for discussion and an outline for further developments. The paper starts with an investigation of the performance requirements for a 3D CAD workstation and a review of recent hardware developments. On basis of this analysis, an

outline of a hardware architecture is proposed. In section 2, an overview is given of 2D drawing and bitmap manipulation, 3D projective display, and 3D ray tracing and rendering functions. A number of functional modules are defined to perform these graphical functions. In section 3, the current developments in graphics hardware are reviewed. From this analysis a concept is derived that is described in section 4.

## 2. Requirements for a 3D-CAD workstation.

The performance of a CAD workstation is largely dependent on the hardware support for interactive graphics and visualisation of 3D objects. For our discussion we disgard the first subject (the functionality of a modern user-interface, as known from the present generation of workstations is assumed) and concentrate on the display functions.

Graphics drawing functions can be divided in 2D drawing and 3D display functions. The 2D drawing functions comprise character and vector generation, polygon-fill, and bitmap manipulations known as bitblt operations. These bitblt operations are used to perform image transformations and general bitmap manipulations which support the multi-window and mouse functions of a modern user-interface. These techniques are well-known and improvements on existing systems are only to be expected with respect to the information per pixel (24 bits and more), etc.

For 3D display, there are basically two modes: projective display and ray tracing. Projective methods are based on the drawing analogue: surface contours are projected on the image plane and the interior area is set to the calculated shading intensity of the object. Ray tracing is based on the camera analogue: for every point of the image plane the light is quantified that is reflected by the scene. Ray tracing effectively models optical effects, such as mirroring reflections, cast shadows and transparency with refraction.

Besides the differences in technique there is also a difference in the kind of processing required for these two display modes. Basic to the ray tracing activity is the intersection calculation. For solids described with higher-order or procedural-defined surfaces, the intersection is complex and involves a large number of floating-point calculations. Otherwise, if the model is defined with polygons or linear halfspaces, the intersection calculation is simple but the search for the right candidates, to avoid numerous intersection tests, is expensive. Further, the calculations needed for a realistic reflection model of the surfaces are of similar complexity. This is in particular true for reflection models that also take into account the indirect diffuse reflection (radiosity models). Finally, for realistic high-quality rendering, often additional features are desired such as the ability to map textures (graphics, text, surface texture) on the surfaces, requiring filter operations for scaling and anti-aliasing. For all these options, a large and programmable processing power capacity is needed.

Projective display at the other hand is a sequence of rather simple computations (transformation, clipping, scan-conversion) that can be performed to the object data in one pass. In particular, the depth-buffer (z-buffer) algorithm lends itself well for pipe-lining because the depth sort can be performed by simple depth-comparison tests at the end of the "viewing pipe-line". The depth-buffer algorithm is therefore used in most VLSI-based systems.

An important object representation in solid modeling is Constructive Solid Geometry (CSG). With CSG, objects are defined as Boolean combinations of elementary volumes, such as block, sphere, cylinder and cone. For several years it has been assumed that the CSG classification needed for the correct display of CSG-defined objects would require some kind of sorting. Recently it has been shown that only repeated depth-buffer tests are needed in combination with some simple logical operations; these operations can very well be performed in a pipe-lined processing approach [Thomas, 1983], [Goldfeather and Fuchs, 1986], [Jansen, 1986].

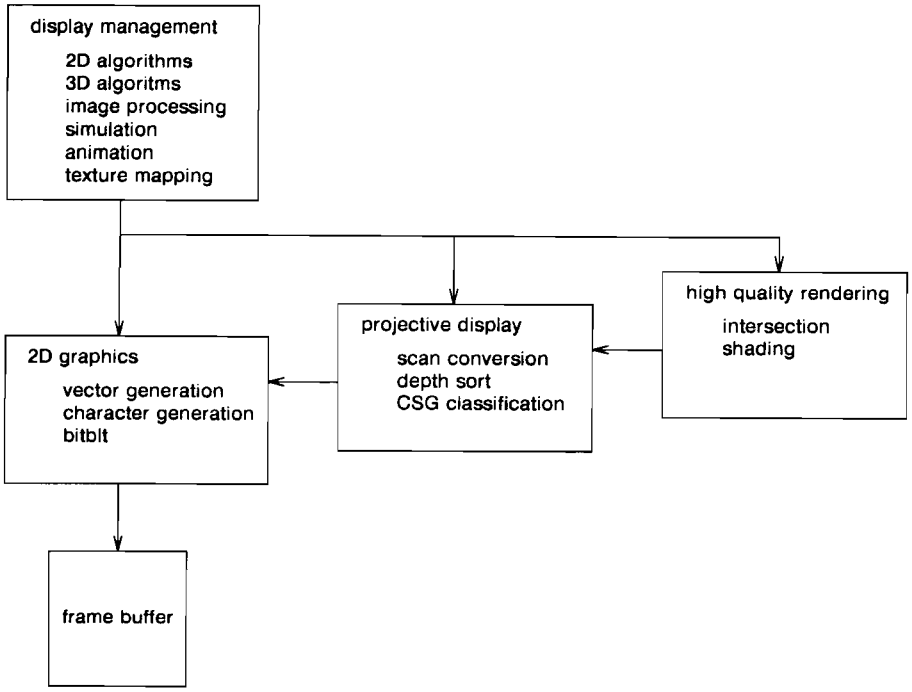


**Figure 1:** Conceptual scheme.

More optional are facilities for image processing. In current CAD systems image processing techniques are not so widely used. This may change, however, in the near future, as the integration of artificial intelligence, computer vision and CAD/CAM proceeds (sketch recognition, etc.). Another interesting area is simulation and animation. The real-time simulation of complex movements involves an

immense amount of computations to generate the flow of display commands. In some modern graphics micro's there are special animation chips to generate the instructions for the display processor fast enough.

These functions can be organised in four modules: display management, high-quality rendering, projective display and 2D graphics (figure 1).

The display management functions involve three kinds of tasks:

- the set-up and coordination of 2D and 3D graphics algorithms performed in the other modules,
- the preprocessing for animation and simulation sequences (scenario management),
- additional processing tasks, such as preprocessing for texture mapping.

The display management tasks (except for the additional processing) have to be performed in real-time but are relatively simple, i.e. these tasks are stored in programs that are straightforward to execute. The second module, the high-quality rendering, involves a large amount of computation. These functions do not have to be performed in real-time; general-purpose parallel-processing capabilities are sufficient here. The 2D graphics and 3D projective functions have to be performed in real-time. They can be decomposed in a general pre-processing part that can be done by the display manager, and a large number of relatively simple operations at the individual pixels (line-pixel intersections, depth calculations, depth comparisons). To obtain the desired performance, some speed-up is needed here in the form of parallellism or dedicated hardware. Some form of functional coherence (the use of functions of one module by the other modules) may be advantageous.

## 3. Hardware developments.

During the 1970s several special hardware implementations of the projective viewing pipe-line have been made for flight simulators, etc., but only with the VLSI-based implementations, these hardware solutions appeared to be cost effective [Clark, 1982]. Several thousands of polygons can be displayed per second with a viewing pipe-line of "geometric engines". Further performance increases are found by using several pipe-lines in parallel. However, to avoid memory contention the frame buffer has to be distributed over the different pipe-lines or processors. The extreme of this strategy, one processor per pixel, has been realised with the Pixel-planes system [Fuchs et al., 1985]. The performance of this system is impressive and fairly complex scenes can be rendered in real-time. Further, the system allows the display of CSG models, anti-aliasing and cast shadows. Although the current version is only for polygon modeling, rendering of quadratic surfaces and higher surfaces is also feasible [Goldfeather and Fuchs, 1986]. Nevertheless, the system is still limited with respect to ray tracing, more complex shading and general texture mapping. A smaller number of more flexible processors seems therefore to be a more desirable solution.

Recently, several multi-processor systems with a distributed frame buffer have been proposed. The different systems can be characterised by the method the frame buffer is distributed over the processor nodes. There are two main memory organisation modes: image subdivision and memory interleaving. The image subdivision mode assigns every processor to a segment of the screen (see figure 2).

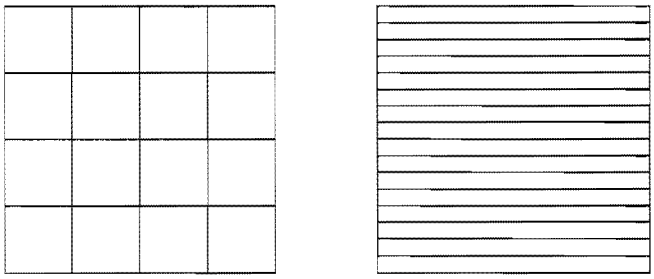


**Figure 2:** Image subdivision.

There are two types of subdivision: block mode (square image segments, figure 2a) and line mode (rectangular image segments, figure 2b). Block mode guarantees optimal area coherence; line mode guarantees optimal scan-line coherence. The latter is advantageous for the most efficient projective display algorithm in software for single processor systems, the scan-line algorithm. The scan-line algorithm displays objects of a few thousand polygons in a few minutes. With a multiprocessor system, the performance can be improved with a factor of 4-10, but this approach fails for a larger number of processors. To guarantee an even load distribution, a load sharing mechanism has to be introduced, resulting in additional overheads. With a growing number of processors, the communication overhead tends to outweigh the savings in processing time. In the end, the brute-force depth-buffer algorithm is as fast [Sato et al., 1985]. For the same reason, also the block mode is not attractive.

In [Fuchs and Johnson, 1979] and [Parke, 1981], an interleaved memory organisation is proposed. Here the image is subdivided into a number of small blocks and the pixels within a block are equally divided over the processors. The blocks can be processed sequentially or in parallel; however, the processing within each block is always done in parallel. The processing is thus evenly distributed over the processors, even when the processing load for each block differs. The interleaved (or interlaced) memory organisation can be realised in two modes, the dot-mode (figure 3a), where each processor stores one pixel of a N-by-N pixel square (for N*N processors), and the line mode (figure 3b), where each processor stores one scan-line out of N lines (for N processors). For a moderate number of processors (16-64), the interleaved-memory organisation guarantees that the load will be evenly distributed over the processors (for a larger number of processors this will be less

effective). It is difficult to estimate which scheme, the dot or line mode, is preferable. The line mode maintains the possibility to exploit line coherence; the dot mode guarantees a better load distribution.

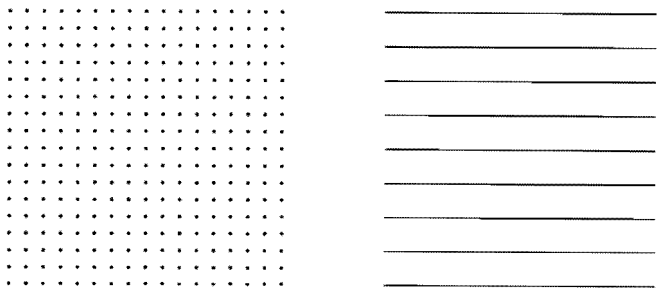


**Figure 3:** Interleaving memory organisation

In [Niimi et al., 1984], a two-level hierarchical multi-processor system based on the line mode is proposed. The system is composed of several scan-line processors (eg. 8), each of which in turn control several pixel processors (eg. 8 for each scan-line processor). The scan-line processors do the preprocessing for a scan-line and the pixel processors manipulate the data within their own pixel territory.

Another line-mode system architecture is proposed in [Torborg, 1987] and [Denault et al., 1987]. The system consists of custom-designed programmable scan-line processors that provide a combination of 2D vector and character generators and 3D triangle shading and depth-buffer routines. Four processors can be combined to render a R, G, B and a Z component for a depth-buffer display algorithm in parallel. The standard configuration consists of four of these sets in an interlaced pattern. The system is designed for the functionality of PHIGS+ but the same functionality could be programmed for other graphics systems.

Although these scan-line systems are programmable to a certain extent, the scan-line processors are particularly designed for the projective display of faceted models. For a more general functionality, a multi-processor system with general-purpose 32-bits processors is needed. In [Sutherland, 1986], a multi-processor architecture is proposed based on the interleaved-memory concept. A similar system has been recently announced [Potmesil et al., 1987]. These systems offer the combination of high-processing power and programming flexibility. In [Jansen and Sutherland, 1987], a CSG depth-buffer algorithm has been described for such a system. An experimental simulation of this algorithm shows that a large number of fairly simple depth comparisons have to be performed. Although the systems are very powerful, it turns out that the large number of relatively simple operations can not be performed in real-time. The sophisticated processors are too slow to achieve

real-time performances. Also the 2D vector drawing functions and bitmap manipulations are relatively slow. This indicates that there is a need for a combination of general-purpose processing power and logic enhancements to the frame-buffer to obtain real-time performances for projective display and 2D graphics.

## 4. A logic-enhanced distributed frame buffer.

The central idea of this scheme is to make a separation between the general "display management" functions and the billions of simple pixel processing manipulations that are needed for real-time performances (figure 4).

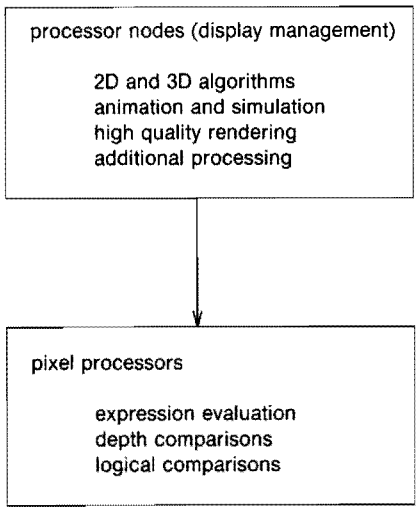


**Figure 4:** Separation between display management and pixel operations.

The 2D and 3D projective functions (vector generation, scan-conversion, etc.) can be decomposed in a general pre-processing part that can be done by the display manager, and a large number of relatively simple operations at the individual pixels (line-pixel intersections, depth calculations, depth comparisons). The same applies for the depth-buffer and CSG classification tests. The result is a large number (but of a relative limited set) of simple pixel operations. To obtain the desired performance some speed-up is needed here, which can be obtained by processing pixels in parallel. It is not necessary to do calculations for all pixels in parallel such as in Pixel-planes but some form of parallellism and hardware support is needed.

So far the scheme is in line with earlier developments [Niimi, 1984], [Torborg, 1987]. The approach described here differs from these in that the separation between higher and lower graphics commands is put on a lower level and more parallellism is achieved at the pixel level with a larger number of relatively simpler pixel processors. One reason for this approach is the large number of local depth comparisons needed for the CSG classification. For that part, this approach is comparable with the Pixel-planes concept.

Secondly, a halfspace modeling scheme in combination with a depth-buffer projective algorithm is chosen instead of a scan-line oriented polygon or triangle modeling scheme, because the halfspace approach offers a suitable basis for all types of surface representations, including quadratic halfspaces [Goldfeather and Fuchs, 1986] and bicubic patches.

Thirdly, a multi-processor built with standard processors is chosen to maintain the programming flexibility for new algorithms and additional processing tasks. If more processor power is desired, general purpose parallel processing power (larger array's, extra array's, extra pipe-lines) can be added without a change of functional concepts. New processor developments are also easier to incorporate without change of the system.

The proposed hardware concept is a multi-processor system with interleaved memory organisation. Each node of the processor array is a general purpose 32-bits processor. Added to each node is an array of 4*4 or 8*8 pixel processors. The pixel processors perform elementary operations for vector and character generation, evaluation of linear and quadratic expressions. Each pixel processor contains additional memory for pixel coordinates and offsets, depth buffers and intermediate results. The pixel processors can perform depth comparisons, logical computations and bitwise operations. However, the algorithms are executed by the display manager in the processor node and all the pixel processors perform the same instructions in parallel. It is not the intention of this paper to specify these operations in detail, but they are derived from the standard 2D raster algorithms [Foley and van Dam, 1982] and 3D projective (depth-buffer) algorithms [Thomas, 1983], [Goldfeather and Fuchs, 1986], [Jansen, 1987a], [Jansen, 1987b]. More sophisticated algorithms such as ray tracing are completely performed in the processor nodes.

The system configuration may vary depending on the number of processor nodes and the number of pixel processors for each processor node. A system could for instance include 16 processors with 64 pixel processors each. For a 1024*1024 display, each pixel processor then manages a number of 1024 pixels. The hardware configuration can be used in a flexible way. Computations can be done "global", i.e. for all pixels in the image or "local", i.e. only for an image segment. This allows the use of image subdivision which may result in a reduction of the rendering load by a pruning of the rendering data [Jansen and Sutherland, 1987]. With 16 processors and 64 pixel processors at each node, an image segment of 32*32 pixels can be rendered in parallel. For a 1024*1024 image, 1024 of these subsegments have to be rendered sequentially.

At this point, several hardware design decisions have to be made concerning the node architecture, i.e. the design and hardware implementation of the pixel processors and whether the display memory should be divided over the pixel processors or not. Only after such a detailed hardware design, the concept can be verified with respect to costs and performances.

## 5. Conclusions.

A functional specification is given for a CAD workstation including high performance 2D and 3D graphics facilities. Recent developments in graphics hardware are shortly reviewed and a direction is indicated for further improvements on both real-time performance and functional flexibility. The presented approach makes a separation between the general processing in the processor node and the execution of a limited set of simple pixel operations in parallel by additional logic (pixel processors).

## 6. References.


Clark, J. (1982), "The geometric engine: a VLSI geometry system for graphics," *Computer Graphics* **16**(3):127-133, Siggraph82.

Denault, D., Ryherd, E., Torborg, J., Tosi, R., Werner, R. (1987), "VLSI drawing processor utilizing multiple parallel scan-line processors," *Advances in Graphics Hardware II*, (this volume) Eurographic Seminars.

Foley, J.D., van Dam, A. (1982), "Fundamentals of Interactive Computer Graphics," Addison-Wesley Publishing Company.

Fuchs, H. and Johnson, B. (1979), "An expandable multi-processor architecture for video graphics," *Proc. 6th ACM-IEEE Symposium on Computer Architecture,* 58-67.

Fuchs, H., Goldfeather, J., Hultquist, J.P., Spach, S., Austin, J.D., Brooks Jr., F.P., Eyles, J.G., Poulton, J. (1985), "Fast spheres, shadows, textures, transparencies, and image enhancements in Pixel-planes," *Computer Graphics* **19**(3):111-120, Siggraph85.

Goldfeather, J., Hultquist, J.P.M, Fuchs, H. (1986), "Fast constructive solid geometry in the Pixel-powers graphics system," *Computer Graphics* **20**(4):107-116, Siggraph86.

Jansen, F.W. (1986), "A pixel-parallel hidden surface algorithm for constructive solid geometry," *Proceedings Eurographics'86*, Elseviers Science Publ., 29-40.

Jansen, F.W. (1987a), "CSG hidden surface algorithms for VLSI hardware systems," *Advances on Graphics Hardware I*, W. Strasser (ed.), Eurographics Seminars, Springer-Verlag.

Jansen, F.W. (1987b), "Solid modelling with faceted primitives," PhD thesis, Delft University of Technology.

Jansen, F.W., Sutherland, R.J. (1987), "Display of solid models with a multiprocessor system," *Proc. Eurographics'87*, Elsevier Science Publ., 377-387.

Niimi, H., Imai,Y., Murakai, M., Tomita, S., Hagiwara, H. (1984), "A parallel processor system for three-dimensional graphics," *Computer Graphics* **18**(3):67-76, Siggraph84.

Parke, F.I. (1980), "Simulation and expected performance analysis of multiple processor z-buffer systems," *Computer Graphics* **14**(4):48-56, Siggraph80.

Potmesil, M. et al. (1987), "Leaflet for AT&T Pixel machine." See also *Electronics*, July 23, p. 54-56.

Rossignac, J.R. and Requicha, A.A.G. (1986), "Depth-buffering display techniques for constructive solid geometry," *IEEE Computer Graphics and Applications* **6**(9):29-39.

Sato, H., Ishii, M., Sato, K., Ikesaka, M., Ishihata, H., Kakimoto, M., Hirota, K., Inoue, K. (1985), "Fast image generation of constructive solid geometry using a cellular array processor," *Computer Graphics* **19**(3):95-102, Siggraph85.

Sutherland, R.J. (1986), "A multi-processor architecture for high-quality interactive displays," *Proc. Eurographics'86*, Elseviers Science Publ., 265-277.

Torborg, J.G. (1987), "A parallel processor architecture for graphics arithmetic operations," *Computer Graphics* **21**(4):197-204, Siggraph87.

Thomas, A.L. (1983), "Geometric modeling and display primitives towards specialised hardware," *Computer Graphics* **17**(3):299-310, Siggraph83.