Towards a 3-D Graphics Workstation

Arie Kaufmann

Department of Computer Science, State University of New York at Stony Brook
Stony Brook, NY 11794-4400, USA*

Abstract

A voxel-map based architecture which lays the foundations for a 3-D graphics workstation, called the *CUBE Workstation*, is presented. The architecture is centered around a large cubic frame-buffer of voxels, operated on by three processors: a *3-D Geometry Processor* which scan-converts geometric objects into their voxel representation, a *3-D Frame-Buffer Processor* which manipulates the voxel-based images and controls interaction, and a *3-D Viewing Processor* which projects the images on a 2-D monitor. Two other supporting processors, the *2-D Frame-Buffer Processor* which manipulates the 2-D images and the *Color Transform Processor* which handles color transformations, are also introduced.

1. INTRODUCTION

The variety of techniques that attempt to cope with the complex 3-D graphics "pipeline" can be characterized in terms of the space they manipulate, namely, techniques that manipulate the ***continuous object (geometric) space***, the ***discrete pixel-image plane***, and the ***discrete voxel-image space***. The former two approaches have received considerable attention in the literature. We focus on an architecture that falls in the latter category, called the *CUBE (CUbic frame BuffEr) Architecture* (Kaufman and Bakalash 1985a), and present a way to incorporate it within a 3-D graphics workstation, the *CUBE Workstation*.

The theme of the CUBE architecture is based on the concept that the 3-D inherently continuous scene is discretized, sampled, and stored in a large cubic frame-buffer of unit cells called volume elements or *voxels*. Three-dimensional objects are digitized, and a regularly spaced array of values is obtained and stored in the cubic buffer. A cubic buffer with 256^3 resolution and 8-bit-deep voxels, for example, is a large memory buffer of size 16M bytes, and a 1024^3 buffer is of size 1G bytes. Since computer memories are significantly decreasing in price and increasing in their compactness, such huge memories are becoming feasible.

There are three additional voxel-image space architectures, GODPA (Goldwasser 1984), PARCUM (Jackel 1985), and $3DP^4$ (Ohashi, Uchiki and Tokoro 1985), that have been

* On leave from the Center of Computer Graphics, Ben-Gurion University, Beer-Sheva, Israel

recently reported in the literature. All the four voxel-based systems, including CUBE, utilize high performance multi-processing architectures to generate on a 2-D display shaded projections of the 3-D scene within the cubic buffer. The huge throughput that has to be handled and the arbitrary projection and rendering functions that have to be accommodated, require special architecture and processing attention. Following is a short introduction to the architectures of GODPA, PARCUM, and $3DP^4$. CUBE, which is the main theme of this chapter, is presented in Section 4.

GODPA (Goldwasser 1984) is a hierarchical hardware organization for viewing medical data, in which the 256-cube voxel memory is divided into 64 symmetric equal sub-cubes. Each sub-cube is processed by a separate concurrent Processing Element outputting a sub-image. These sub-images are processed by eight Intermediate Processors, each of which merges, based on a sequence control table, eight sub-images. The eight resulting buffers are merged by the Output Processor into the a 512×512 buffer, using the same control table. The post-processor performs gradient shading to enhance the realism of the image. GODPA, unlike its predecessor the Voxel processor (Goldwasser and Reynolds 1983), is capable of displaying many independently controllable objects. The Voxel architecture has been used as the underlying architecture for a physician workstation (Goldwasser et. al 1985).

Jackel (Jackel 1985) has described a graphics solid-modeling system, called *PARCUM*. The memory cube of PARCUM is divided into 64^3 macro voxels each of which is 4^3 voxels one bit each. The address of a voxel consists of the region address, and a voxel address. All voxels with the same voxel address are assigned to a specific memory module, each of which can be addressed separately. Therefore, a read access can fetch in parallel a macro voxel, which is then sorted into a 64-bit output data word. Jackel has proposed two hardware implementations of the projection process, a surface detection technique (based upon the binary rate multiplier mechanism), and an analytical approach (using matrix multiplication). The illumination process is based on a distance-only technique which generates the gradient matrix for the shading.

The *$3DP^4$* architecture (Ohashi, Uchiki and Tokoro 1985), realizes an interactive animation system using solid modeling. It employs a 3-D array of 256^3 voxels arranged like GODPA as a symmetric set of subcubes, each of which is 64^3 voxels. The architecture is a pipeline, the first stage of which employs a set of parallel processors, one for each subcube, to perform the projection using linear interpolation. The next stage is a merging processor that merges the projected images. Shading is done in a post processor using a depth map.

This chapter presents the underlying architecture and algorithms of a voxel-based architecture, the *CUBE Architecture*, and suggests an avenue for the implementation of the architecture as part of a *3-D CUBE Workstation*. A possible layout of the CUBE workstation is portrayed in Fig. 1. The workstation has two components, a 3-D component which is the CUBE architecture, and a 2-D component. The latter includes two new supporting processors, in addition to the contemporary 2-D raster frame-buffer, video processor, and 2-D screen. The supporting processors might be incorporated in any 2-D or 3-D graphics system, and in particular in the CUBE workstation. The first supporting processor, described

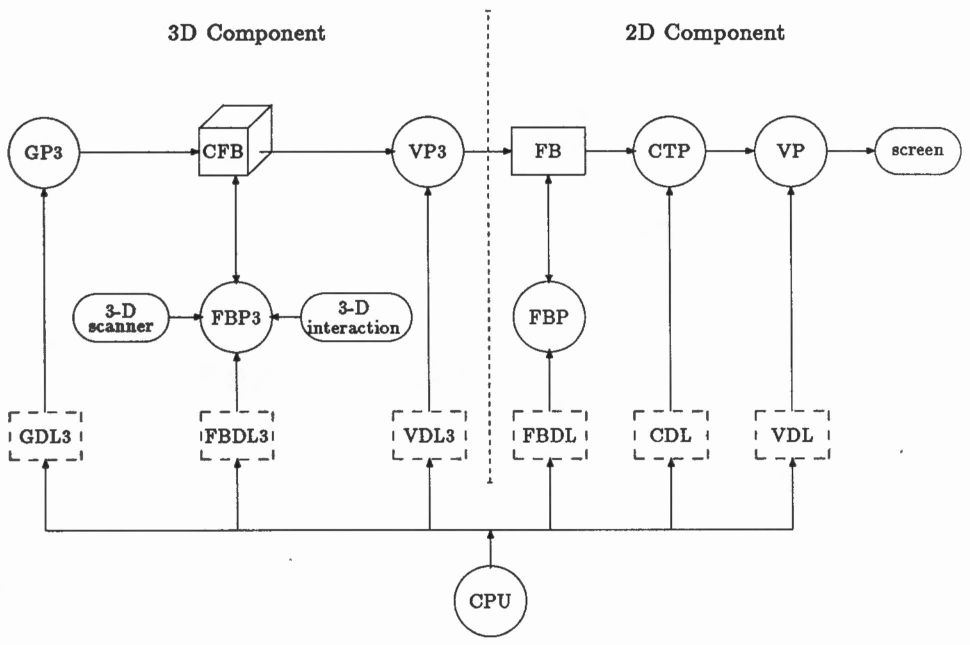

Figure 1: Architecture of the CUBE Workstation

in Section 2, is a *2-D Frame-Buffer Processor* which is an improved *bitblt*-like engine. The second supporting processor, described in Section 3, is a *Color Transform Processor* which transforms color values from an input color notation system to an output system. An overview of the 3-D component (the CUBE architecture), its memory organization and its three processors, is given in Section 4. Section 5 sums up with notes on the organization of the entire CUBE workstation.

Unlike the other systems, CUBE has no limits imposed on the memory size nor does it have to be a power of 2 cubed. Moreover, CUBE produces projections in real-time and supports rendering with full color, semi-transparency, and shading. The fundamental difference between CUBE and the other systems lies in the fact that these systems operate solely in the discrete voxel-image space and they all thus assume that a database of voxel-based objects exists for loading and image viewing, while CUBE further provides geometry space capabilities and ways to intermix the two, as well as a full spectrum of workstation functions. A detailed comparison of GODPA, PARCUM, $3DP^4$ and CUBE can be found in (Kaufman 1986b).

2. THE 2-D FRAME-BUFFER PROCESSOR

The 2-D *Frame-Buffer Processor (FBP)* is part of a proposed two-dimensional raster graphics architecture. The FBP, as a co-processor to the conventional graphics/geometry and video processors, works directly on the frame-buffer memory. It executes its commands from a display list, the FBDL. The introduction of a co-processor would increase efficiency and speed, and would support operations that have ordinarily been performed by software running on the CPU, like a window manager. The FBP is a variation of a *bitblt* engine (Guibas and Stolfi 1982) with a richer repertoire. It is a pixel-map processor which operates on rectangular sub-areas within the frame-buffer, and accepts events and data from an input locator and a 2-D scanner.

The FBP primitives are *windows*, *cursors*, and *icons*. A window is a rectangle defined by two corners and a priority value which is a unique value for each window facilitating the support of overlapping windows. Cursors provide a flexible feedback mechanism for man-machine interaction. Unlike contemporary graphics system where the cursor pattern is intermixed with the video signal, the FBP physically places the cursors within the FB and stores the data that have been covered by the cursors for later restoration when the cursors move. Icons are small application-oriented constant images, that are either anchored to a window or may float independently.

The instruction set of the FBP includes operations on 2-D primitives (windows, icons, cursors), and instructions for program control and internal register settings. The operations on primitives include: definition and deletion of primitives, writing, reading and erasing the primitive's content, swapping, translating, scaling, rotating, copying, filtering, and changing priorities of primitives. Internal registers can be set or reset, and all subsequent operations are affected. The registers that are alterable during the active mode of the FBP are: status, FB origin, depth plane mask, color mode, color component sizes, texture mask, pixel-ops, orientation, zooming, background, foreground, and filter registers. There are additional initialization, system, and input registers. The sets of operations and registers provide an extremely flexible environment for manipulating the FB images. The FBP provides hardware support for applications such as: window management, image processing, graphical interaction, animation, and supports two-and-a-half dimensional environment. It can be incorporated within either a 2-D or a 3-D system.

3. THE COLOR TRANSFORM PROCESSOR

The other supporting processor is the *Color Transform Processor (CTP)*, which transforms colors from one color system to another. The processor handles eight systems which gained wide acceptance: RGB (Red-Green-Blue), CMY (Cyan-Magenta-Yellow), HLS (Hue-Lightness-Saturation), HSV (Hue-Saturation-Value), XYZ (CIE primaries), IQY (NTSC), CNS (Color Naming System) (Berk, Brownston and Kaufman 1982), and ACNS (Artist's Color Naming System) (Kaufman 1986c). These color notation systems can be either the input or the output color system for the CTP.

The processor input, which is a list of color components represented in one of the color notation systems, may come from a variety of sources, e.g., a frame-buffer, a video processor, a user's program. The processor transforms the input colors to a different output representation. For this the processor employs one, two or three algorithms out of a set of 20 conversion algorithms. All the algorithms have been designed for possible hardware implementation or coding in microcode. The output, a new list of color components, is sent to the desired destination. The processor may operate in a LUT mode, where the input colors are mapped onto the destination colors using a color look-up table, followed by an optional color transformation. The processor also provides a mechanism for Gamma correction, if required.

The CTP may be incorporated in any 2-D or 3-D graphics system, where a change in the color notation system is necessary. The layout of the CUBE architecture in Fig. 1 suggests that the CTP is installed between the FB and the VP, so that the source color values in the FB are transformed into the color mode of the video signal. The CTP can alternatively be installed before the FB, or a modified CTP can be installed between the VP and the screen or possibly combined with the VP itself.

4. THE 3-D COMPONENT

The 3-D component of the CUBE Workstation is the CUBE architecture. The heart of this architecture is a *Cubic Frame Buffer (CFB)* which is a large three-dimensional cubic memory. The basic memory element is a voxel with a numerical value of density in the broad sense, representing the material, color, texture, and translucency ratio of a small unit cube of the real scene. To support the real-time storage and retrieval of images to and from the CFB, a unique memory organization, which enables the memory to handle beams of voxels simultaneously, has been designed. To support the real-time viewing of such a huge cube of memory, a special common bus has been developed. It processes a full beam of voxels simultaneously and selects the first opaque voxel in a time which is proportional to the *log* of the length of the beam. There are three processors, the *3-D Geometry Processor (GP3)*, the *3-D Frame-Buffer Processor (FBP3)*, and the *3-D Viewing Processor (VP3)*, which access the CFB, to input, to manipulate, and to view the CFB images, respectively.

4.1. The 3-D Geometry Processor

The *3-D Geometry Processor (GP3)* scan-converts 3-D geometric objects into their 3-D discrete representation within the CFB. The GP3 executes a GDL3 display list which consists of instructions to generate voxel-based versions of 3-D geometric objects or compositions thereof. These objects are 3-D lines, polygons (optionally filled), polyhedra (optionally filled), parametric curves (possibly bounded by planes), parametric surface patches (possibly bounded by planes), volumes bounded by surfaces and planes, circles (optionally filled), and quadratic objects, such as spheres, cylinders and cones (optionally filled).

The GP3 operates in the CFB coordinate system, thus, the 3-D objects have to be transformed and mapped onto the CFB domain by the CPU before being sent to the GP3, although the GP3 can clip objects to planes parallel to a major axes plane. The GP3 is basically a write-only algorithmic-oriented processor, where for each geometric object a 3-D scan-conversion algorithm is applied. For linear objects (lines, polygons, polyhedra) either a 3-D Bresenham-like algorithm, a first order DDA, or a modified scan-line algorithm, is utilized. For polynomial objects (curves, surfaces) a third order DDA is used. For quadratic objects adaptations of the 2-D circle and ellipse algorithms are used. Details of the algorithms can be found in (Kaufman 1986d).

4.2. The 3-D Frame-Buffer Processor

The *3-D Frame-Buffer Processor (FBP3)* is the 3-D extension of the 2-D Frame-Buffer Processor introduced in Section 2. It manipulates and processes 3-D discrete images stored in the CFB. It also acts as an interface for 3-D input devices and as a channel for 3-D scanned data. The FBP3 is an independent processor which executes commands stored in the main memory in a FBDL3 display list.

The primitives of the FBP3 are 3-D sub-arrays of the CFB of three types: rectangular *boxes* (cuboids, 3-D windows), *jacks* (3-D cursors), and *figurines* (3-D icons, vicons). A box is cuboid defined by two opposite corners and a priority value which is a unique value for each box facilitating the support of overlapping boxes. Jacks provide a flexible feedback mechanism and enable the user to tour the 3-D scene. Figurines are small application-oriented constant 3-D images, that are either anchored to a box or may float independently. FBP3 is actually a voxel-map engine performing such operations as 3-D box manipulation, cursor and figurines handling and inputing from 3-D scanners.

The instruction set of the FBP3 includes register operations (e.g., set, get), program control instructions (e.g., jump, jump on error, call, return, pop, push, activate, halt, reset, attach/disattach device), and box, jack and figurine operations (e.g., define, delete, write, read, erase, swap, translate, rotate, scale, spin, copy, filter, change priority). Special care has been taken to provide simple and fast operations especially for the large box operands. All transformations, including shifting, scaling (using non-integral factors), rotation (through any angle), are performed using an incremental transformation technique, in which each voxel is transformed, based on the new position assumed by its immediate previous neighbor, with a maximum of 3 additions.

4.3. The 3-D Viewing Processor

Two-dimensional projections of the CFB images onto the graphics display are accomplished by means of a *3-D Viewing Processor (VP3)*, which executes a VDL3 display list. The VP3 generates from the CFB parallel orthographic projections, arbitrary parallel projections, and perspective projections. In order to handle the huge throughput a special common bus, the *Voxel Multiple-Write Bus (VMWB)*, has been designed. It is based on the Multiple-Write Bus technique (Gemballa and Lindner 1982) developed at the Technical University of Darmstadt. The VMWB selects in a given beam the opaque voxel closest to the assumed observer, in a time which is proportional to the *log* of the length of the beam. The common bus concept is also appropriate for back and front clipping, for selecting isocolor surfaces, and for handling semi-transparent surfaces. The VP3 also renders the 2-D image during the projection process, implementing semi-transparency, and lighting and shading (using local gradient field).

4.4. Organization of the Cubic Frame Buffer

A unique memory organization, which fetches/stores a full beam of voxels simultaneously, enables the processors to handle beams rather than single voxels, and thus enabling the system to cope with real-time constraints. In order to fetch/store a full beam of voxels simultaneously, a special 3-D symmetric organization of the CFB has been designed (Kaufman 1986a). It consists of diagonal parallel sections having a 45 degree angle with the main axes planes, and are grouped in n distinct memory modules. A voxel with space coordinates (x, y, z) is being mapped onto the k-th module by:

$$k = (x + y + z) \ mod \ n \tag{1}$$

The internal mapping (i, j) within the memory module is given by:

$$i = x, \qquad j = y \tag{2}$$

Since two coordinates are always constant along any beam, regardless of the view direction, the third coordinate guarantees that one and only one voxel from the beam resides in any one of the modules.

5. THE CUBE WORKSTATION

Many components of the CUBE architecture are clearly ideal for hardware implementation, and some components have the potential for implementation with VLSI technologies. These possibilities has been examined, and they have been taken into consideration when designing each of the components and the processors of the system, enabling a compact implementation of the CUBE workstation.

In addition to the design and simulation of the memory layout and each of the processors as a separate entity on a VAX computer with a RAMTEK 9400 graphics system, the overall integrated system architecture has been examined. The entire set of simulations have been also integrated on two workstations: on an IBM/AT with a Galaxy graphics board and prototype IC boards for the memory/VP3 processing units, and on a SUN-3 MicroSystems 160C workstation with a color monitor. Each of these systems has been configured as a CUBE workstation according to Fig. 1. The pictures in Figures 2-5 are snapshots taken off the CUBE workstation screen. Figure 2, taken off the IBM screen, presents the interference between a cylinder and a sphere both scan-converted by the GP3. The first column shows three orthographic projections (CFB resolution of only 64^3), the second column is the intersection of the two objects, while the third and fourth columns show the projections of the two objects separately. Figure 3 shows three orthographic projections of objects scan-converted by the GP3 and projected and shaded by the VP3 on the RAMTEK from a 128^3 CFB. Figure 4 presents four different views of an object made out of three concentric spheres (blue, red, and green), enclosed by a red cylinder which shares a common base with a blue cone. The composed object has been generated by the GP3 and has been studied, by slicing and "peeling" off exterior layers, using the VP3 and a CFB of 128^3 on the RAMTEK monitor. Figure 5, taken off the SUN monitor using a 128^3 CFB, presents a model of Saturn with semi-transparent rings. The symmetric patterns on the objects in Figures 2-5 are a result of the low-resolution discrete shading.

The integrated graphics system works as a slave of a host CPU, forming together the CUBE workstation. All the different display lists (GDL3, FBDL3, VDL3, FBDL, CDL, VDL), residents of the main memory, are prepared (interactively) and processed by a 3-D graphics software running on the host. The FBP, the CTP, and their corresponding display lists (FBDL and CDL) are optional in the CUBE workstation. The user interacts with a versatile interface which enables a flexible control over all the processors and their operations. A user's command or a sequence of commands generates a display list, which is then submitted to the appropriate processor.

The CUBE workstation is suitable for many 3-D applications, such as: medical imaging (e.g., tomography and ultrasounding), CAD (e.g., solid modeling) (Kaufman and Bakalash 1985b), 3-D animation and simulation (e.g., instrumentation simulation), 3-D image processing and pattern recognition (e.g., time varying 2-D images), quantitative microscopy (e.g., biological/geological microstructures), and 3-D graphics interaction. Although the voxel representation is more effective for empirical imagery, it also has a significant utility in synthetic 3-D graphics or in applications merging empirical and geometric model (e.g., a synthetic injection needle superimposed on an ultrasound image.)

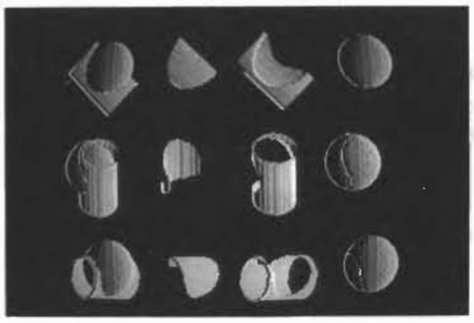

Figure 2

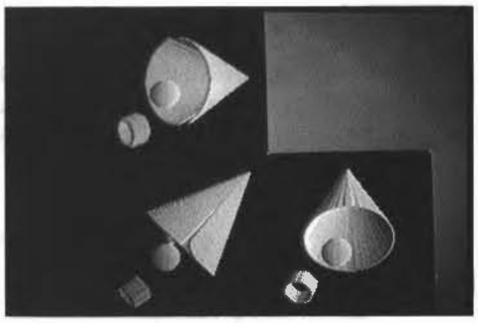

Figure 3

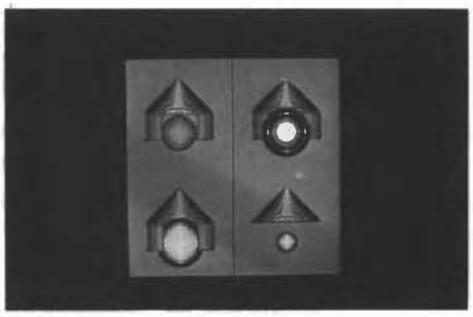

Figure 4

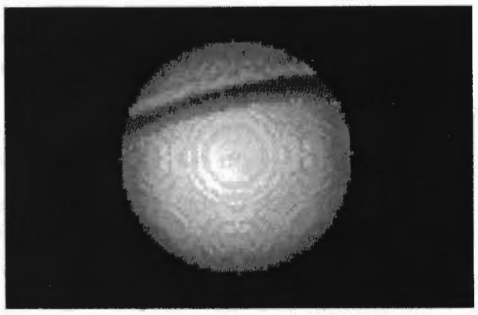

Figure 5

Acknowledgment

This work was supported by the National Science Foundation under grant DCR-86-03603. The initial steps of the this research were supported by the National Council for Research and Development, Israel, under grant 283-63.601. The assistance of Reuven Bakalash in this project is greatly appreciated.

REFERENCES

Berk T, Brownston L, Kaufman A (1982) A New Color-Naming System for Graphics Languages. *IEEE Computer Graphics and Applications* 2:37-44

Gemballa R, Lindner R (1982) The Multiple-Write Bus Technique. *IEEE Computer Graphics & Applications* 2:33-41

Goldwasser SM, Reynolds RA (1983) An Architecture for the Real-Time Display and Manipulation of Three-Dimensional Objects. *Proceedings International Conference on Parallel Processing* Bellaire, Michigan 269-274

Goldwasser SM (1984) A Generalized Object Display Processor Architecture. *IEEE Computer Graphics & Applications* 4:43-55

Goldwasser SM, Reynolds RA, Bapty T, Baraff D, Summers J, Talton DA, Walsh E (1985) Physician's Workstation with Real-Time Performance. *IEEE Computer Graphics & Applications* 5:44-57

Guibas LJ, Stolfi J (1982) A Language for Bitmap Manipulation. *ACM Transactions on Graphics* 1:191-214

Jackel D (1985) The Graphics PARCUM System: A 3D Memory Based Computer Architecture for Processing and Display of Solid Models. *Computer Graphics Forum* 4:21-32

Kaufman A, Bakalash R (1985a) A 3-D Cellular Frame Buffer. *Proceedings EUROGRAPHICS'85* Nice, France 215-220

Kaufman A, Bakalash R (1985b) A New Approach for a Real-Time 3-D CAD Workstation. *Proceedings Seventh Conference on CAD/CAM and Robotics* Tel-Aviv, Israel

Kaufman A (1986a) Memory Organization for a Cubic Frame Buffer. *Proceedings EUROGRAPHICS'86* Lisbon, Portugal 93-100

Kaufman A (1986b) Voxel-Based Architectures for Three-Dimensional Graphics. *Proceedings IFIP'86, 10th World Computer Congress* Dublin, Ireland 361-366

Kaufman A (1986c) Computer Artist's Color Naming System. *The Visual Computer* 2:361-366

Kaufman A (1986d) 3D Scan-Conversion Algorithms for Voxel-Based Graphics. *ACM/NSF Workshop on Interactive 3D Graphics* Chapel Hill, North Carolina

Ohashi T, Uchiki T, Tokoro M (1985) A Three-Dimensional Shaded Display Method for Voxel-Based Representation. *Proceedings EUROGRAPHICS'85* Nice, France 221-232