Walkthrough – A Dynamic Graphics System for Simulating Virtual Buildings

Frederick P. Brooks, Jr.

Department of Computer Science
University of North Carolina at Chapel Hill

0.1 Abstract

As part of our graphics research into virtual worlds, we are building a tool for an architect and his client to use for rapid prototyping of buildings by visually "walking through" them in order to refine specifications.

Our first prototype simulated the new UNC Computer Science building with some 8000 polygons. BSP-tree software on the Adage Ikonas gave a colored, shaded perspective view every 3-5 seconds while the user moved a cursor in real-time over floorplans shown on the Vector-General 3300.

The current (third) version uses Pixel-Planes to generate 9 updates/second, view images shown 4' × 6' by projector.

Active short- and long-term research questions include speed-up, stereo, a 6-DoF interface with eye-level defaults, and an interactive model-building, model-changing system.

0.2 Application Concept

At the 1965 IFIP Congress, Ivan Sutherland challenged computer graphicists to look beyond "the picture in the window" toward creating virtual environments in which the viewer is immersed, and where he sees, feels, and manipulates the virtual objects as if they were real. Much of the research at Chapel Hill has been aimed at this objective.

The Walkthrough Project aims at providing a tool in which virtual buildings, designed but not yet constructed, can be explored by "walking through" them in the same way that simulated airplanes "fly" over virtual terrain. The object is not training, as it is with flight simulators, but visualization to permit the architect to

"prototype" a building and to iterate with his client on the detailed desiderata for it.

Today's CAD-CAE systems for building architecture aim primarily at assisting the designer in recording his decisions in a database and in producing standard blueprints and specifications. We envision another, complementary, computer graphics tool to help the designer visualize and explore the spaces he is creating, as part of the design process.

Such a tool could be especially useful for designer-client dialogues. Whereas the designer is trained and experienced in imagining spaces, the client is not. He needs visualizations to aid his thinking and his communication with the designer. So the objective is a system that allows the user to visually experience the spaces in a virtual building, a vision first articulated by Donald Greenberg:

> For architects the ability to simulate motion is highly useful. One of the principal concerns of architectural design is space: the internal spaces of a building and the external space of the building and its setting. One does not react to space from a static position, as one might view a painting. To obtain a deeper understanding of architectural space it is necessary to move through the space, experiencing new views and discovering the sequence of complex spatial relations. [Greenberg, 1974]

Given this objective, what can be done? The fielded state of the art offers two extremes: moving views and user-steered pre-computed high-quality views. Color wire-frame models of great intricacy can be moved and rotated in real time by vector displays. Skidmore-Owings-Merrill's vector model of the Chicago Loop is a most impressive example. (We define real-time to be twelve updates/second or better, matching silent movies).

Alternatively, one can select a fixed view of a virtual building or other model and produce a precisely rendered, color-shaded, illuminated, textured, anti-aliased image of high quality. Each such frame takes at least 20 minutes on a 1-MIPS computer. One can make video of a succession of such frames along a preplanned cruise path, at 30 frames, or 10 hours of computing, per second of video.

Today the technology puts an attractive intermediate within striking distance: real-time user-directed cruising through a virtual building, with continuous real-time production of TV-cartoon-quality shaded, colored, stereo views.

0.3 Ikonas Walkthrough

Last year we built a prototype of such a system for our new Computer Science building. The user gets a floorplan view on the vector display, with eye position and direction marked by a cursor, and he gets a rendered view on the Ikonas display. Thus the user can steer himself down corridors, peer into offices, even see the several floors as he rides in the elevator – all before the building has been constructed.

This work was initially motivated by our own needs for our interactions with our architects, the firm of O'Brien and Atkins. This prototype has been useful to us in decision-making, especially about the lobby layout and the balcony width. The user sees the spaces in 3-D stereo perspective, properly solid, colored, and lighted.

The program was still too slow for real-time viewing, by a factor of about 35. One did get real-time updating of the floorplan cursor, so navigation was not difficult. The eye view got updated about every 3 seconds, at best, if the computer had no other load. By stopping for three seconds at a desired viewpoint, one could get any arbitrary view.

This architectural tool was a lash-up of some in-hand advanced technology, namely

- The Fuchs, Naylor, Kedem binary-space-partitioning algorithm, and Gregory Abram's software for it,
- an Adage Ikonas 3000 video frame buffer and a VG 3303 vector display,
- a very fast bit-sliced microprocessor in the Adage Ikonas,
- new stereo viewing devices.

This prototype is demonstrated on the videotape, made with an earlier 5000-polygon database. The building database today consists of some 8000 polygons.

In fact, we had not expected to get so near to real-time viewing speed in this first try. The success of this prototype attracted the attention of Turner Whitted, Henry Fuchs, and me to the possibility of trying to make it into a useful tool. We are approaching our dream system in steps, following a sort of Bresenham's algorithm in which each successive version makes a step improvement in whichever aspect seems to have the widest remaining discrepancy between where we are and the ideal system.

0.4 Pixel-Planes Walkthrough

In this first prototype, Ikonas Walkthrough, the slow update rate was the shortcoming that most impaired the illusion we were seeking.

Our second system was therefore designed to take advantage of the high polygon rendering speed of Fuchs's Pixel-Planes display engine. A team under John Poulton was building a 512 × 512 pixel prototype aimed at demonstration at SIGGRAPH '86.

Building a Pixel-Planes-based Walkthrough necessitated two steps:

Interface Conversion. The user interface was converted

> *from* a Vax-driven system using the Vector-General 3303 vector display, an HP 2621 character display, a data tablet, and a box of analog knobs and sliders *to* a Masscomp workstation system using only a mouse and the Masscomp screen for floor plans, menus, and analog controls.

Oliver Steele, Michael Sayko and John Singleton did this work as a spring-term class project, using the Blox interface management system sold by Rubel Software.

Database Conversion. The polygon database was converted from the form suitable for the binary-space-partitioning tree algorithm to the form suitable for driving Pixel-planes. Douglass Turner did this work and integrated and tested the whole system.

0.5 Big-Screen Walkthrough

Wide Angle. In Pixel-Planes Walkthrough, the small image size, and especially the narrow angle subtended at the eye, seemed to be the major impediments to the realism. The use of a commercial high-quality Barcodata video projector promised to address both of these problems. With projection, the viewer's field of view can be quite wide, although the resolution per steradian drops proportionately. We do not yet know where the best point in this tradeoff lies.

Stereo. In June a team under Michael Pique established that rear projection through our plastic screen preserves circular polarization, so that the Tektronix liquid-crystal stereo shutter works with our projector. This offered the option of having big-screen stereo, at the cost of cutting the effective update rate in half.

Meanwhile, Douglass Turner had been dividing the database into zones of guaranteed mutual invisibility, which is always possible with buildings. This is aimed at getting to real-time update rates.

Interface. The mouse-driven interface in our Pixel-Planes system proved awkward for easy-to-understand reasons. User head-motion in walking through a real building has six degrees of freedom, all of which can be changing dynamically at once. Pixel-Planes Walkthrough's mouse system, which allows at most two degrees of freedom to change dynamically, was noticeably worse than the Ikonas Walkthrough's interface, where three could be changed dynamically.

John Hughes therefore rigged our Polhemus sensor into an appropriate carrier, a sphere with the bottom flattened so that the head stays level as its default position. This gives six degrees of freedom and seems more satisfactory. Turner made the extensive software modifications necessary to accept these inputs.

0.6 System Structure

The figure shows the structure of the Walkthrough System as planned. It consists of six major parts

- the View Specifier, the user interface with which one drives the system. The Specifier reads mouse, Polhemus, data-tablet, joystick, etc., devices.
- Builder, a subsystem with which the user enters and modifies the building model.
- the Master Model, an Ingres-based relational database carrying the building's components in easy-to-manipulate form.
- the Working Model, a specialized database constructed for fast view generation.
- the View Instancer, which encapsulates the Working Model and makes views from it. The Instancer drives the display devices.
- the Toolbox, which contains algorithms and data manipulation routines needed by the other modules.

This decoupled modular structure is not the way the prototype is built today. Builder and View-Specifier are decoupled modules today; the rest is fairly monolithic.

Data Acquisition

A major part of the work associated with using a building simulator is getting the building into it. The database must be constructed. Then one wants to modify it interactively as one explores design concepts. During building design, one modifies a building model many times, and most modifications are minor, affecting only a few polygons out of thousands.

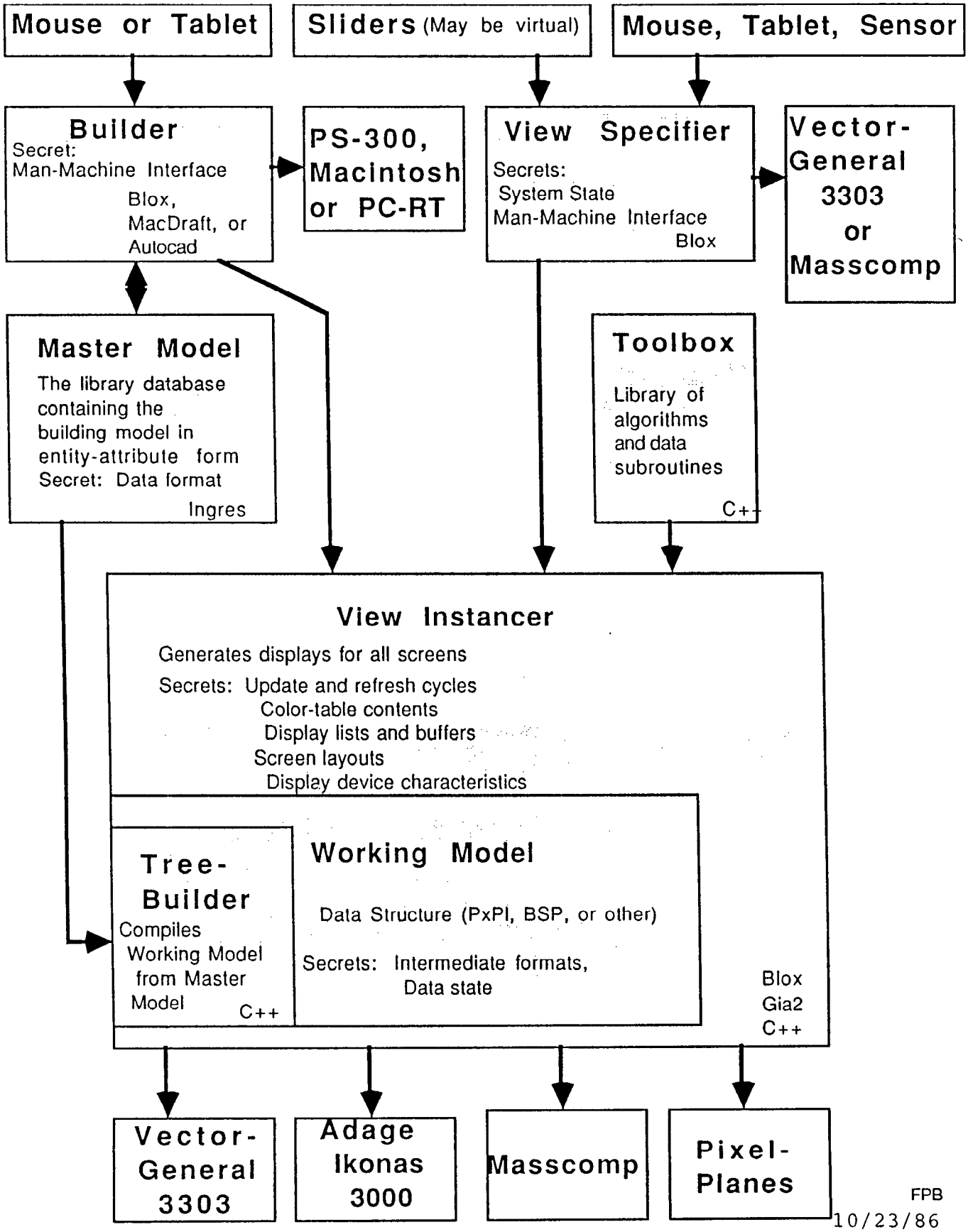
Constructing a mathematical model of a building polygon-by-polygon is tedious work. The work is moreover unnatural, in that designers think in more hierarchical fashion. Much of it is redundant; a rectangular solid requires 6 parameters to be specified, but doing it polygon-by-polygon could require 72. For our running prototype, Dana Smith adapted Ousterhout's Caesar microchip design tool for the input of building parameters. [Ousterhout, 1981] The Caesar tool is especially designed for manipulating, sizing, connecting, and labeling scaled rectangles, so it was an excellent choice for our prototype.

It is crucial, if we are to make a usable system, to devise data representations and interactive data specification techniques so that specifying and maintaining a building model is made as easy and natural as the information content will inherently allow, and so that visual feedback is promptly available.

Central to our concept is the use of two distinct databases, one containing a subset of the other's data:

- the Master Model, an entity-attribute relational database, operated with the Ingres relational database software, designed for cleanliness of structure and ease of modification. Of this, more below.
- the Working Model, a specialized data structure containing a subset of the data – visibility and rendering parameters only, for polygons as the only entities. This is designed for speed of instancing views.

Planned System Structure



FPB
10/23/86

Working Model. The Working Model is now, and will be, entirely encapsulated by the View Instancer, so that new visibility algorithms with their own data structures can be substituted bodily for it without affecting the rest of the system.

The Working Model contains a Model Compiler that upon invocation accesses the Master Model and rebuilds the Working Model. Ultimately we would hope to use journaled data of changes to the Master Model to allow rapid and partial modification of the Working Model.

Builder Subsystem. The system component Builder is envisioned as an interactive program that gets model parameters from the user and transforms them to canonical form and places them in the Master Model. In the process it is to be doubly interactive – by interacting with the Master Model it determines inconsistencies and incompletenesses and generates prompts for the user. By invoking the View Instancer to make dynamic views on the vector scope, it gives visual feedback to the user. The user will work with a mouse or datatablet, selecting from pop-up or pull-down menus and scale views of the component he is specifying.

0.7 The Usefulness Question

We have now the third version of Walkthrough. So what? Is this any more than an exercise in expensive toy-building? Can its usefulness to architects and their clients be shown?

We believe this to be an inescapable, central question for all advanced application research. So UNC teams have in the past undertaken to establish, with various degrees of formality, the usefulness of graphics systems for:

- molecule modeling
- object definition by reconstruction from medical images
- force display
- true 3-D display
- multi-parameter study of inventory control policies
- teaching numerical methods

As to Walkthrough, we don't know about usefulness yet. Conclusive validation of a system such as Walkthrough is very difficult. Flight simulators can be measured by testing the operational behavior of pilots trained in them. No easy operational

test is available to us.

We plan first to collect and document anecdotal and testimonial evidence from real users. Indeed, I am reluctant to do much more technical development of the system until we get such feedback.

Our fifteen years of research on graphics for molecular structure studies has hammered home one lesson again and again:

Frequent, close, iterative interaction with real users doing real work is an essential discipline for the system researcher.

Direct feedback from real work helps

- **aim** research at concrete objectives,
- **focus** research on a few objectives at a time,
- **order** research goals,
- **measure** accomplishment against extrinsic objective criteria.

We have found capturing user sessions, by computer logging and on videotape, to yield many insights and surprises.

There is, of course, the danger of getting idiosyncratic, not generally valid, input when one works intently with a few users. Our experience encourages us to take this risk. Each time we have learned how to solve some user's particular problem well, we have found generalization to solving a larger set of problems to come easily.

Hence our next activity is to find one or more architectural enterprises whose working designers are willing to invest hours of their time in collaborating by using our system. We clearly must undertake to make the effort worthwhile to them on a "today's job" basis. We suspect that usefulness will first be demonstrated for designer-client dialogues.

0.8 Research Questions

The Walkthrough System encounters several research questions of wider interest and applicability in 3-D graphics research.

1. How fast an update rate is fast enough to give the illusion of continuous motion in a raster color-graphics context?
2. Which components of visual fidelity and world-model accuracy make the most difference to real designers and their clients:

- stereopsis – (Will a cheap post-perspective shear transformation be satisfactory?)
- lighting models,
- model detail (number of polygons, e.g., doorknobs, moldings),
- texturing,
- curved-surface rendering,
- wide angle of view?

3. Which of the viewing and model parameters need to be smoothly controllable, and how?
   - Can analog sliders with little loss be replaced by pop-up slider scales overlaid on views?
   - Can some parameters that today are analog-specifiable be fixed without seriously affecting function?
   - How is the best way to specify a head turn involving no position shift?
   - Which of our dozens of switchable display options (e.g., walls, trees, furniture) are worth the trouble?
4. How can we take advantage of special properties of the class building models to simplify or speed up our computations? Note that flight simulators take extensive advantage of the special properties of their world models [Schachter, 1983]. Among these properties for buildings are:
   - Most zones of a building are permanently invisible to each other.
   - Most polygons are rectangular.
   - Most polygons are axial, i.e., parallel to two of the axes. Within each of the x,y; x,z; y,z sets of axial rectangular polygons, visibility calculation is simplified.
   - Most potentially visible polygons are usually hidden by a very few that are close to the eye.
   - Most polygons are members of coplanar sets.
   - Many coplanar polygons are also nested.
   - Rooms, doors, windows, furniture, and other elements are replicated.
   - Furniture is compact – building polygons do not intersect the convex hull of a furniture piece.
   - Walk-paths are continuous and can be extrapolated [Shelley and Greenberg, 1982]. In particular, most steps change polygon visibility only slightly.
   - Outdoor scenery visible through windows can perhaps be satisfactorily treated as flat painted backdrops.
   - Many polygons are partly defined by intersections with others:
     - floors extend to exterior walls
     - interior walls extend floor-to-ceiling or floor-to-adjacent-floor.

- A limited set of standard textures: wood, carpet, bookshelves, acoustic tile – account for most surfaces.
- Most texturing is rectangular textures on axial planes:
  - bookshelves
  - multi-pane windows
  - acoustic-ceiling
  - paneling
  - bricks

  Perhaps some special congruence operator can be devised for such.

A System Vision

It is worthwhile to follow the precept and example of Edwin Land and envision the system we would like to have, and then iterate stepwise towards it.

Our ideal system includes

- a world-model database at least 10-fold more detailed than at present – e.g., 100,000 polygons for a structure.
- a model-building system that uses defaults and procedural modeling to give a complete, but undetailed and perhaps inaccurate view early on, with progressive refinement in detailing and truthfulness, as proposed by Turner Whitted. [Amburn, et. al., 1986]
- automatic model building from standard architectural drawings and standard architectural CAD-CAE systems.
- automatic color input from samples as well as interactive hue intensity, saturation specification.
- a head-mounted display with 2π steradian solid angle and high resolution.
- a rendering system capable of handling hundreds of textured polygons per frame-time.
- A real-time rendering system capable of handling local light sources, point and distributed.
- a three-degree-of-freedom head orientation sensor.
- a two-degree-of-freedom treadmill able to sense changes of direction.

References

Abram, Gregory D., "Real-Time Image Generation with Anti-Aliasing and Texture Mapping," PhD thesis, expected 1986.

Amburn, Phil, E. Grant, and T. Whitted, "Managing Geometric Complexities with Enhanced Procedural Models", *SIGGRAPH*, vol. 20, no. 4, August, 1986, pp. 189-195.

Bishop, Thomas Gary, "SELF TRACKER: A Smart Optical Sensor on Silicon," PhD. thesis, University of North Carolina at Chapel Hill, 1984.

Eastman, Charles M., *Spatial Synthesis in Computer-Aided Building Design*, John Wiley and Sons, New York, 1975.

Eastman, Charles M., "General Purpose Building Description Systems," *Computer-Aided Design*, vol. 8, no. 1, January 1976, pp. 17-26.

Fuchs, Henry, Z. Kedem, and B. Naylor, "Predetermining Visibility Priority in 3-D Scenes," *SIGGRAPH*, vol. 13, no. 2, 1979, pp. 175-182.

Fuchs, Henry, Gregory D. Abram, and Eric Grant, "Near Real-Time Shaded Display of Rigid Objects," *SIGGRAPH*, vol. 17, no. 3, 1983, pp. 65-72.

Greenberg, Donald P., "Computer Graphics in Architecture," *Scientific American*, vol. 230, no. 5, May 1974, pp. 98-106.

Ousterhout, John, "Caesar: An Interactive Editor for VLSI Layouts," *VLSI Design*, 4th Quarter, 1981, pp. 34-41.

Schachter, Bruce J., ed., *Computer Image Generation*, John Wiley and Sons, New York, 1983.

Shelley, K.L. and D.P. Greenberg, "Path Specification and Path Coherence," *Computer Graphics*, vol. 16, no. 3, July 1982, pp. 157-166.

Weingarten, N.H., "Computer Graphics Input Methods for Interactive Design," M.S. thesis, Cornell, 1977.