Architecture and Performance of Entertainment Systems

James Helman
Silicon Graphics Computer Systems

Designing Real-Time 3D Graphics for Entertainment
SIGGRAPH '95 Course

In the beginning there was the Drive In
And Walt said, "Let there be a Park."
And lo, there was Pirates of the Caribbean
Thus did Walt beget the Experience Industry
And Walt saw that it was Good...
- Michael Krantz Figure [Kranz94]

1 Introduction

The use of real-time 3D computer graphics in interactive entertainment has grown dramatically recently. These entertainment applications include better arcade games, 3D-capable home game consoles, more sophisticated multi-player games for location-based entertainment (LBE) centers, virtual actors on TV driven by puppeteers with motion capture devices, and even virtual interactive theatres where the "player" can assume the role of a character in a story and alter the course of the plot.

This chapter of the course notes tries to provide a general background into the elements that go into creating a real-time 3D graphics entertainment application and the basic performance levels required to meet human factors requirements. Subsequent chapters fill in the details of content generation, programming and graphics techniques that can be used to meet those performance requirements across platforms ranging from home $250 game consoles to image generators for multiplayer LBEs costing $100,000 and above.

2 What's New?

The initial wave of real-time, 3D graphics for is hitting the entertainment market at many different levels. At the high end, theme park based systems, such as Epcot Center's Aladdin VR experience which opened last summer, are running on high-end graphics workstations. Such systems can support a quality of content that approaches the production values of TV or film with complex scenes covered with hand-painted imagery and complex characters animated in real-time. The same set of underlying graphics capabilities that make this possible, most notably texture mapping, high polygonal complexity and 3D character animation can be seen moving into less expensive systems produced for location-based entertainment use such as Magic Edge, W Industries, Iwerks and Virtual World Entertainment. With the latest generation of arcade machines and home game consoles like the Sega Saturn, 3DO Multiplayer, Atari Jaguar and, of course, the Nintendo Ultra64, many of these same capabilities are beginning to appear in the home.

With most LBE sites, many arcade and home games taking on a "virtual" moniker, entertainment is often called the "killer-app" for virtual reality. Alternatively, one could say that the same improvements in technology that enable VR are also enabling new applications of computer graphics in entertainment, some which are immersive in the VR sense and some of which are not.

Whatever one's perspective, the technological forces behind this movement can be seen by looking back at 3D computer graphics over the last decade. Two developments are key: the evolution of graphics hardware and the creative skills to use 3D computer graphics effectively.

Hardware Evolution

One could divide relevant CG applications into areas with different performance and cost requirements.

- computer generated imagery (CGI) for film and broadcast.
  very high image quality → low frame rate @ high cost.
- modeling, animation production, MCAD and data visualization.
  medium frame rates → low image quality @ medium cost.
- 2D video games.
  low cost, high frame rates → low image quality
- visual simulators.
  high frame rate, medium image quality → high cost

The equation that has constrained high and medium quality rendering to the realm of frame-by-frame CGI is very roughly:

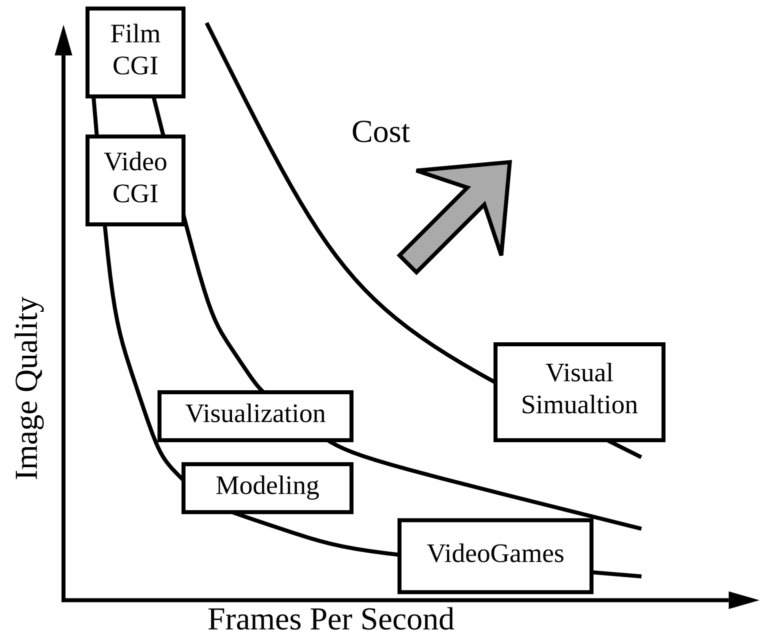

FIGURE 1. Frame Rate versus Image Quality versus Cost

$$cost \propto frame\ rate\ X\ image\ quality \qquad \textbf{(EQ 1)}$$

Based on their requirements, applications find different sets of tradeoffs between cost, image quality and frame rate attractive as shown in Figure 1

Until very recently, realistic 3D graphics has fallen into two camps. One could produce imagery with sufficiently high quality to meet Hollywood standards, but with rendering times measured in minutes per frame. Or using high-end visual simulation equipment, you could produce marginally realistic graphics at 12 to 60 frames per second, but at costs per display channel in the $200,000 to $1,000,000 range.

On the technical side, what's changing is the continuing improvement in the price performance of computers and graphics hardware. This decreases the proportionality constant every year and moves the cost curves back towards the origin for a particular combination of quality and frame rate. For example, a system suitable for visual simulation system that might have cost $200,000 per channel might now cost 1/10th or 1/20th of that. The result is that real-time 3D graphics becomes practical for more uses in entertainment systems all the way from 3D texture mapped video games to high-quality theme park attractions.

So that now it's becoming possible to have:

high frame rate *and* high image quality @ low cost

Frame Rate (frames/sec)	Application	Quality	Cost
frame-by-frame	Film CGI	very high	very high
.001 - 1 fps	Video CGI	high	high
interactive	modeling tools	low	medium
5-10 fps	motion capture	low	medium
	data visualization	low	medium
real-time	visual simulations	medium	medium
15-60fps	video games	low	low
	LBE	medium	medium
	broadcast	high	high

TABLE 1. Applications grouped by frame rate requirements

Creative Skills

"Movies did not flourish until the engineers lost control to artists"
-Paul Heckel [Heck91]

In addition to the technical developments, the second enabling element is the formation of creative talent with the knowledge and expertise to produce content for this new medium. The production of a top-notch entertainment experience requires a large set of skills. The team often consists of:

- story/game designers
- CGI animators and modelers
- visual simulator developers

The creation of compelling 3D scenery and characters draws heavily on experience that is found primarily among traditional animators and those working in the industry built around non-real-time computer generated imagery for film and video. These people have the ability to create compelling scenery and bring characters to life. In moving to the domain of real-time graphics, the main challenge is how to live within a limited budget of geometry and texture imagery without destroying the visual effect.

The integration of the many technical elements into a real-time system requires experience from the visual simulation industry which is familiar with the programming and integration of real-time processors, texture-mapped graphics hardware, sound systems, displays, motion platforms, input devices, etc.

The Result:

This merging of new technologies and new talent is facilitating the production of a new form of entertainment, perhaps even a new medium on a par with film or video. For lack of a better term, I'll refer to it using Krantz's: "realies". While overall realism and character quality are still somewhat limited, the interactivity and immersivity alone make it qualitatively different from the media which spawned it. It's characteristics are:

- more realistic than video games
- more story and character than video games
- more interactive than ride films
- more immersive and first-person than film or TV

	Realistic	**Interactive**	**Immersive**	**Detailed Character**
Video Games	No	Yes	No	No
Ride Films	Yes	No	Yes	Yes
Film & TV	Yes	No	No	Yes
3D "Realies"	Yes	Yes	Yes	Yes

TABLE 2. Characteristics of various entertainment media.

3 Platform Hardware and Software

Constructing a complete entertainment system requires many pieces including hardware, software and database. Usually these fall into two sets. The platform components are the low-level building blocks which are not highly specific to any particular game or experience. On top of this platform lie layers of increasing specificity. The next layer might be a software run-time manager that could control the system for a particular class of game experiences. The final layer then would be the content which is specific to a particular game: characters, 3D geometric models, scenery, game logic, behaviors, animations, AI for autonomous characters, in short the game application and its associated content.

Each of these functions requires a hardware component and a software component to provide a useful interface for driving it. Together they provide a common set of capabilities which can be used for running many different games, much as a movie projector provides a platform for the showing of films. Figure

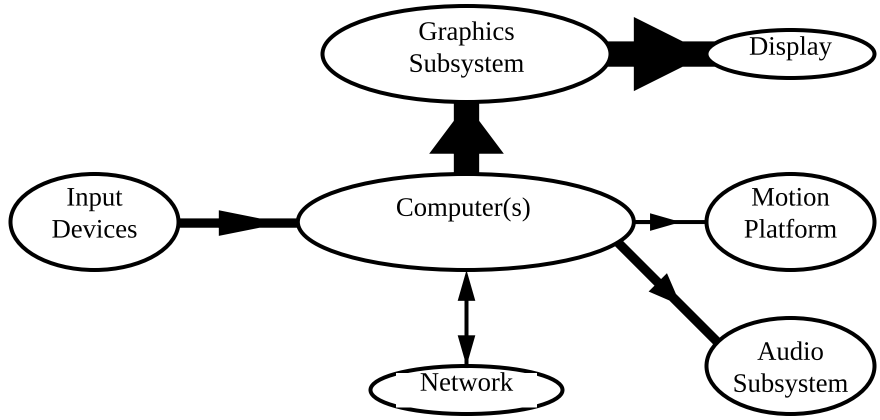

FIGURE 2. Common Hardware Components

2 shows a common set of hardware elements.At the lowest level, the platform can be divided into six areas:as shown in Table 3.

Function	Hardware Example	Software Example
general processing	workstation CPU(s)	Unix™
visual	graphics subsystem CRTs head-mounted displays	IRIS Performer™
audio	MIDI synthesizer	audio drivers
motion	motion platform	dynamics model motion drivers
input	joysticks trackers	device drivers
output	LED displays real bells & whistles	

TABLE 3. Low-level run-time functions and hardware/software examples

Hardware

In entertainment systems, image quality and frame rate are not goals in themselves. It's not about how many antialiased, textured, bump mapped, polygons per second you can draw, or about how many sound sources, diffusers and reflectors your 3D spatial sound system can render. Entertainment systems, whether in a home, arcade or theme park, need to achieve sufficient fun per dollar to pay for themselves either directly or as an attraction.

> *"Super Mario didn't sell millions of copies because the mushrooms were texture mapped"* *- Tom Garland, SGI*

But it takes certain capabilities to make something compelling and fun. And the standards go up every year. Pong was quite a hit in 1972. But if some neo-

Bushnell were to install a game with similar graphics quality in an arcade today, would he have problems with it shorting out from filling up too fast with quarters? Probably not. And theme park standards must be even higher to warrant the longer distances and greater expense. So there is an as yet unsated thirst for quality which is unlikely to be met until real-time rendering can approach the production quality of CGI for film and video. And judging by the visual quality which current graphics hardware can achieve at real-time rates of 20-30 fps, we've still got a ways to go before we have graphics power to spare.

The table below lists a number of factors which contribute to the quality of the experience and the hardware involved.

Capability	Enabling Hardware
visual complexity	GFX, displays
frame rate	GFX
character animation	CPU + GFX
motion realism	motion platform
audio fidelity	synthesizers, spatializers, speakers
player environment	pod, auxiliary displays, input devices
collision detection	CPU

TABLE 4. Capabilities and the associated hardware which enables (and limits) them

Often, the graphics subsystem is the single most expensive component.

While seeking quality to support richer content and distinguish themselves, developers of entertainment systems are also very sensitive to cost per player. Revenues for these systems typically range from the 25cents/play of arcade video games to the dollar/minute or more charged by many location-based entertainment installations. Total system costs can range roughly from $10,000 to perhaps as high as $100,000 per player. In many of these systems, the computing and graphics hardware are the largest factors in cost.

Unless many players share a single display, as in a large motion cab with a projection screen, these price constraints lie well below the cost per visual channel of traditional visual simulation image generators and are even a stretch for many workstation-based graphics subsystems.

Thus, developers are caught between their hunger for visual quality and a thin wallet, but are constrained by. Nothing except an infinite budget can eliminate the constraint imposed by (EQ 1). Every additional polygon costs something in the bottom line. But careful attention to the design of the visual database and implementation of the run-time system can be used to maximize the visual impact and frame rate at a particular cost, effectively reducing the proportionality constant in the equation and improving cost/performance.

Architectural Design

Once the performance and content requirements of the system have been determined. The architecture of the hardware system should be designed with three main things in mind:

- processing power - how much does a particular subsystem require?
- bandwidth - how much data needs to get in and out of each subsystem
- latency - what sort of delays are allowable in data generation and transfer

Graphics Hardware

Most graphics hardware capable of rendering reasonably high scene complexity at high frame rates scan convert polygons for rendering, rather than using ray tracing or volume rendering techniques. Architectures in this domain include like RealityEngine [Akeley93] or LEO [Deeri93].

While the fundamental tradeoff is between frame rate, visual complexity and cost, the details of a particular graphics architecture are often important to gaining maximum performance from it. Chapter 9 of these course notes, a reprint of [Akeley89], provides a good introduction to this class of graphics hardware. Depending on the graphics architecture used, one may be limited in the number or size of polygons that can be rendered, the number of pixels that can be drawn, the amount of data transferred between the host and graphics subsystem. Methods for optimizing rendering to these constraints are described in Chapter 4. Some techniques such as texture mapping (discussed in Chapter 2) can dramatically increase the perceived visual complexity without increasing the polygonal complexity of the database.

Host to Graphics Connection

Traditionally one of the largest consumers of bandwidth has been the connection between the visual database and the graphics subsystem. For this reason, many systems, such as the image generators used in visual simulation, have had the database reside in the graphics subsystem itself. This has advantages in that it allows hardware specific optimization of rendering and requires a much lower bandwidth between the host computer and the graphics subsystem (commands such as “move object #15 North 10 meters”). But such optimizations (e.g. binary space partitioning) often place restrictions on the dynamism of the data and because the rendering engine owns the database, any new features, such as character animation, must be coded directly in the rendering hardware rather than in the friendlier host development environment.

Most workstation and many PC graphics systems take a different approach and have the rendering traversal and sometimes the first stages of the rendering performed on the host CPU. This allows more flexibility in rendering, but requires a much larger bandwidth between the host CPU and the graphics subsystem, since every polygon vertex, normal and color must be transferred each frame.

Software

This platform layer consists of the core run-time functionality that could be used by many different games. This level of software should provide a high-performance layer which isolates the game developer from the details of the underlying hardware. Typically, it comes in the form of a toolkits or software libraries layered on top of an operating system.

Real-Time OS?

One approach to achieving real-time performance is to assemble a custom hardware dedicated and perfectly tuned to the task as with VIDEOPLACE which Myron Krueger has developed over the last 18 years. He meticulously "timed software modules with a logic analyzer" to be certain about the performance characteristics [Krueg90]. This is the level of certainty we desire, and predictability and high performance must be designed into the hardware and software platform. But developing a fully custom OS or even utilizing one of the standard "real-time" kernels available for embedded systems requires a very substantial amount of work. The development of hardware graphics drivers in itself would be prohibitive for many projects which are deployed in small numbers.

The fast CPUs, good development environment, flexibility and graphics performance of UNIX workstations makes it an attractive OS for high end game systems. One can prototype, develop, code, model, paint and debug all on the same or similar systems. But is it good for deployment? UNIX has a bad reputation for scheduling and interrupt latency.

In order to achieve consistent and predictable performance on a workstation, one needs to insure that the desired application processing is not interrupted by daemon or ancillary interrupt driven kernel activity. Multiprocessing workstations can provide this functionality by allowing processors to be restricted to certain tasks so that applications and real-time device drivers have guaranteed response times. The REACT extensions to Silicon Graphics' IRIX operating system supports these features.

Other issues in trying to achieve constant frame rates and constant latencies lie in the application domain. Frequently gaming systems are so complex that it may not be feasible to exhaustively simulate all contingencies. To prevent the system from failing under stress, the platform software should also provide mechanisms for monitoring performance and respond gracefully when overloaded preferably without dropping its frame rate. Most often this involves decoupling various processing tasks and being able to provide extrapolated values when actual information is delayed.

Synchronization and Communication

Sometimes the most cost effective solution requires having a single powerful computer driving the experiences of multiple players. For example, a high-end, multiprocessing workstation could drive six game pods playing the same or different games as shown in the left side of Figure 3. In this case, synchronizing the different players is a trivial matter of communication through shared memory.

But when multiple systems need to be networked to connect players into a common game or environment, as shown in the right side of Figure 3, synchronization is required to ensure that the

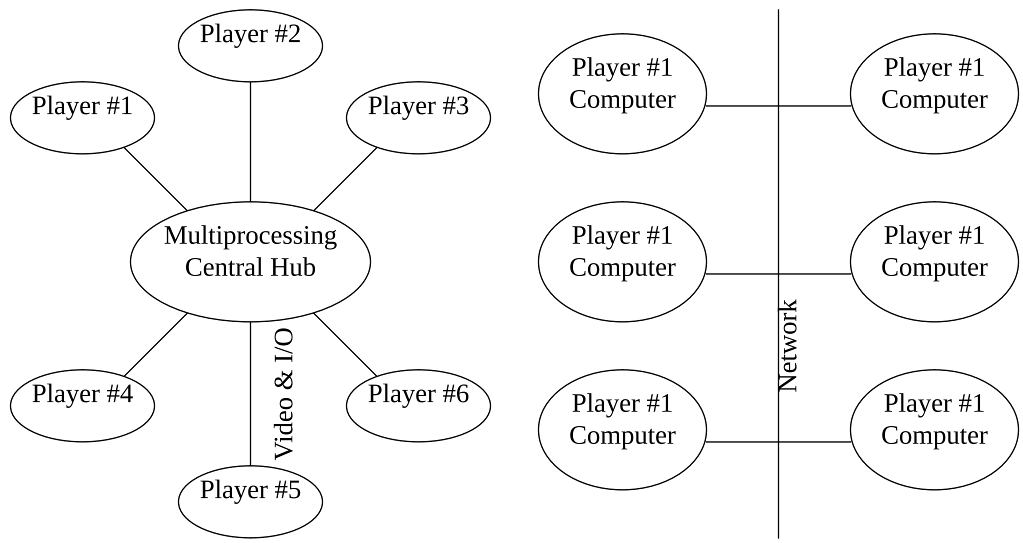

FIGURE 3. Centralized vs. decentralized systems

global state (e.g. locations and conditions of players) for the game is the same for everyone. This becomes complicated when the network connecting the systems has high latencies, and is particularly a problem for fast-paced games in which objects are moving rapidly. Maintaining accurate information about player location requires all systems to have an accurately synchronized notion of time and a method for extrapolating position information. and correcting for extrapolation issues such as the paper [Sing94] reprinted in Chapter 10.The accuracy of synchronization in multiplayer simulation is largely a function of network bandwidth and latency. Depending on requirements and budget, solutions range from relatively high-latency connections such as modems over telephone lines to low-latency reflective shared-memory systems such as SCRAMNet (as used in the CAVE [Cruz93]).

When multiple machines drive the same display (e.g. panoramic, multiprojector display), very fine synchronization is required. Typically this is done through synchronizing the video clock and buffer swap mechanisms of the different systems rather than through a standard network.

Domain Specific Software

For example, on the graphics side, one might use a toolkit such as IRIS Performer to provide a graphics and database processing, as well as a framework for multiprocessing. But a complete system requires comparable low-level software for handling input devices, networking, audio, dynamics, collision response, and so on. Some higher-level platform software covers more than one of these areas (e.g. Paradigm Simulation's Vega and Division's dVS), but currently no single solution covers the entire set of needs listed above. Consequently, a number of different software suppliers and a fair amount of custom coding are often used.

4 Artistic Content

The other major component is the content of a particular game, which in the projector analogy corresponds to what's on the film being projected. In many ways, producing a game is like producing a film. First and foremost comes the concept, plot, characters, scenarios and game activity. Jordan Weisman, the creator of Virtual Worlds Entertainment, designed role playing board games for many years before venturing into location-based entertainment. Game designers have much more experience to draw on today than when Chris Crawford developed Excalibur at Atari a decade ago, but his comments on the state of the art still sound true, especially as advances continue to change the design medium and range of possible content.

> *Computer games constitute a new and as yet poorly developed art form that holds great promise for both designers and players.*
>
> *-Chris Crawford [Craw84]*

For the most part, these issues lie outside the graphics and engineering focus of this course. Some aspects of the design process are covered in later chapters in this course. Crawford's book [Craw84] and section in Laurel's book [Laurel90] give many practical insights for designing computer games. [Laurel91] and [Thom81] also provide essential background material. In his review of PC, CD-ROM and video game technology[Morr94], Morrison also briefly describes the design considerations in the making of several current games. Moses Ma, the designer of Spectre VR, a 3D simulation game for PC and Macintosh, sums up his ideas on game design as follows:

Moses' Ten Commandments of Game Design

1. High Concept: A story that's hot, original, obvious, leading edge on technology, emotional
2. Barriers: Non-limiting, challenging — like flying through closing barn doors. "Difficult landings", try thread the needle, barriers *must* move and change
3. Evolving Enemies: Interesting characterizations, show inner conflict! The enemy keeps you from getting what you want. Interaction on 3 levels: world, personal, inner.
4. Dazzling Graphics: I mean really... Is it breathtaking? Is there a consistent cinematic image system? Does it have a look?
5. Emotion producing sound.
6. Action well-timed and rising/falling action and boredom detection - 20 minute episodes — sweat drenching — opening hook, conflict, crisis, resolution.
7. Tight registration of feedback: Both collision detection, joystick and visual meter feedback.
8. Positive Monitoring in Learning Curve: Use a flight log on data disk, easy to learn / a challenge to master.
9. Infinite Replayability: Create data disk format/ flightlog - levels of skill.
10. Make the gaming simulation world real, authentic (not cliche). Interesting characterizations! Know the world, details count.

Implementation

After moving from concept to detailed design, next comes the implementation. As with a film, sets must be constructed and populated with characters. This requires the modeling of geometry, the generation of imagery for texturing and the specification of the data necessary to animate the characters. Because this medium is new, there is a relative scarcity of production tools.

Component	Examples
visual database	geometry, texture, animated models
audio database	samples, data for synthesis
behavioral models	animation control, collision response, AI players

Component	Examples
visual database	geometry, texture, animated models
motion platform	motion models
user interface	interface models
game logic	scenario handling
application	overall control, story, coordination, continuity

Geometric Data

Typically the geometric database is constructed with a 3D modeling tool. The desired output is a collection of polygons suitable for real-time rendering. Commercial modeling tools currently tend to fall into two categories, modelers with origins in visual simulation and those with origins in non-real-time computer animation. Visual simulation modelers tend to focus on supporting the construction of objects polygon by polygon or through terrain height fields, combined with lofting, rotation and extrusion. These methods of construction are appropriate for real-time systems since polygon count must be minimized to achieve reasonable frame rates. Such modelers also provide real-time features such as level-of-detail switching and precomputed animation sequences as discussed in Chapter 2.

Modelers used for producing frame-by-frame animations usually have a richer set of surface construction tools, but often lack real-time concepts such as ranges for level of detail switching and prerendered animation sequences. Higher level primitives such as spheres, cylinders and NURBS are powerful but can be dangerous for real-time systems. While they make the construction of multiple levels of detail much easier, they also make it easy to generate far to many polygons for real-time rendering.

Image Data

Image data can be produced by many different means: photographs of objects or hand paintings, use of computer paint tools, procedural generation, etc. When game is based on an existing property such as a film or TV show, much of the image material may already exist. This image data is then mapped onto geometry, usually using the same tools that were used to generate the geometry.

Animation & Motion

Introducing dynamism is the great challenge in designing content for real-time systems. A deserted scene in which nothing moves or changes probably won’t make a good game. Defining paths for object to move through a scene is quite straightforward as is specifying jointed articulation of static elements such as the motion of a crane or the wheels on a car. But complex animations require more than this. Traditionally in visual simulation, such animations have been modeled as flip-card sequences of preanimated models. Flip-card animation sequences eliminate the run-time computational load of complex animations. The next step up is to use geometric morphing to interpolate between steps in the sequence. One is still limited to the prerendered sequencing, but at least the motion can be smooth and existing animation packages can generate the sequences of models.

Total animation of a character requires both articulation and geometric morphing as discussed briefly in Chapter 2. Depending on the type of animation, some combination of key frame animation, live motion capture, goal-directed motion [Badl91] and dynamics modeling can be used. Since deGraf and Wahrman showed “Mike the Talking Head” at SIGGRAPH ‘88, a growing amount of work has been done on generating character animations at interactive frame rates. But the computational and graphics requirements make this difficult, and commercial tools for translating animations into forms suitable for real-time rendering remain scarce.

Some types of geometry and imagery are well-suited for procedural generation and animation [Ebert92][Prus90]. Extensive procedural generation of geometry (e.g. fractals, plants) and texture images is rarely done at run time because the large computational burden required. With a few notable exceptions such as Pixel-Planes 5 [Fuchs89], most graphics hardware systems do not accelerate procedural texture generation.

So, most procedural animation used in real-time systems takes much simpler forms such as defining a mathematical model for the motion of waves. Such models usually end up being defined in application code rather than in the database partially for efficiency, but mainly due to lack of tools and database formats to support them.

Special Effects

> *“Nintendo informed Jaleco that the exploding hamster had to be deleted in future cartridges.” - David Sheff [Sheff93]*

The creation of special effects whether sparkling pixie dust or something more visceral, often falls outside the capabilities of standard modelers. Often some combination of modeling and procedural generation at run-time are used. The run-time capabilities available for use include flip-card animation of textures and geometric models, morphing, texture coordinate animation *and simple* particle systems.

5 The Director

The tar pit of software engineering will continue to be sticky for a long time to come. - Fred Brooks [Brook82]

We've covered two obvious elements: artistic content and the platform for showing it on. Using the film analogy, on one side we have the actors, the set and the script and on the other we have the projector, or at least a motor, bulb and lens. Unfortunately, the technology for making a "realie" is nowhere near as well developed as film or video. Each developer must put together their own system for bridging the gap between the artistic content and the platform. Several years or more may elapse before any standard solutions are settled on. Until then, one of the major challenges of assembling a system is writing this software director in a way that it can be reused for different experiences.

This "Director" software is the run-time engine that coordinates all activity. On one side the director is soaking up information from all of the input devices, and following the script, tells the camera operator how to shoot, the audio system what to say to the player, the characters how to move.

One problem is the script. It appears to have no simple representation. It covers a large range of activities: gaming logic, autonomous characters, complex animations, scenery shifts, scenario tuning based on player's responses, collision response, etc. How can one hope to embody something so ill-defined? Software, of course. It's the programmability of our machines that makes rapid progress possible even in the face of problems on the scale of trying to create virtual theatres and gaming worlds.

The topic of "Director" software and other elements of game construction are covered in Chapters 5-7.

6 Conclusions

Never mistake a clear view for a short distance. - Paul Saffo [Saff90]

The current situation as far as tools for creating content, hardware for producing the sound, motion and graphics, and the software for integrating the content to the hardware indicates that this industry is still in its infancy. Modeling and animation tools do not yet fully reflect the requirements for producing scenery and characters that can be rendered in real-time. With large gains every year, graphics hardware is now able to produce much more compelling scenes than were available to the first developers of computer games. But it still falls far short of being able to unable to render everything that our artists would like to at the real-time frame rates required for interactive gaming and theatre. And while software platforms are improving along with the hardware, there are as yet no standard or fully reusable run-time “projection” systems.

No doubt the available tools and software will improve as the market and our understanding of the problem improve. Until then, those wishing to create such games or experiences have a lot work to do ranging from building their own sets, cameras, and even projectors.

As with other entertainment media, the most important element lies in the design of compelling and feasible concept. Beyond that, the largest problem is trying to squeeze the execution of that concept into the 15 to 50 milliseconds we have to think about and render each frame.

Graphics is currently one of the most expensive elements in the run-time system, and nothing can save developers from the quality vs. frame rate vs. cost tradeoff embodied in (EQ 1). But if the entire game design process from modeling and animation to the run-time software proceeds knowing that every polygon, every character animation and every special effect costs something in the bottom cost/performance line, we can at least shift the balance in favor of more visual impact. This becomes more important as the producers of these new entertainment systems seek to distinguish themselves. Chapters 2-4 of these notes discuss various software and modeling methods for use in this effort.

7 Bibliography

[Akeley89] Kurt Akeley, "The Silicon Graphics 4D/240GTX Superworkstation", *Computer Graphics and Applications*, Vol. 9, No. 4, July 1989, pp. 71 - 83.

[Akeley93] Kurt Akeley, "RealityEngine Graphics", *Proceedings of SIGGRAPH '93*, July 1993, pp. 109-116.

[Appi92] P.A. Appino, J.B. Lewis, L.Koved, D.T. Ling, D.A. Rabenhorst, and C.F. Codella. An architecture for virtual worlds. *Presence*, 1(1), Winter 1992, pp. 1-17.

[Badl91] Norman Badler, Brian Barsky and David Zeltzer, eds. *Making Them* Move: Mechanics, Control and Animation of Articulated Figures. Morgan Kaufmann, San Mateo, California, 1991.

[Brook82] Frederick P. Brooks, *The Mythical Man-Month*. Addison-Wesley. 1982.

[Craw84] Chris Crawford. *The Art of Computer Game Design*. Undated manuscript. 1984(?).

[Cruz93] Carolina Cruz-Neira, Daniel J. Sandin, and Thomas A. DeFanti. "Surround-screen projection -based virtual reality: The design and implementation of the cave," *Proceedings of SIGGRAPH '93*, July 1993, pp. 135-142

[Deeri93] Michael F. Deering, "Leo: A System for Cost Effective 3D Shaded Graphics", *Proceedings of SIGGRAPH '93*, July 1993, pp. 101-108.

[Ebert92] David S. Ebert, F. Kenton Musgrave, Darwyn Peachey, Ken Perlin. *Pro*cedural Modeling and Rendering Techniques. Course Notes. SIGGRAPH 1992.

[Fuchs89] Henry Fuchs, et. al. Pixel-Planes 5: A Heterogeneous Multiprocessor Graphics System. Using Processor-Enhance Memories. *Proceedings SIG*GRAPH '89. Computer Graphics 23(3), July 1989, pp. 79-88.

[Heck91] Paul Heckel. *The Elements of Friendly Software Design*. Sybex, San Francisco, 1991.

[Kranz94] Michael Krantz. Dollar a Minute. *Wired* 2.05, May, 1994.

[Krueg90] Myron W. Krueger. Simulation versus artificial reality. In E.G. Monroe, editor, Proceedings of the 1990 Image V Conference (Phoenix, 19-22 June 1990), Image Society, 1990, pp. 147-155

[Laurel90] Brenda Laurel, ed. *The Art of Human Computer Interface Design.* Addison-Wesley, Reading, Massachusetts, 1990.

[Laurel91] Brenda Laurel. *Computers as Theatre*. Addison-Wesley, Reading, Massachusetts, 1990.

[Morr94] Mike Morrison. *The Magic of Interactive Entertainment*. Sams, Indianapolis, Indiana, 1994.

[Prus90] Przemyslaw Prusinkiewicz. *The Algorithmic Beauty of Plants*. Springer-Verlag, New York, 1990.

[Saff90] Paul Saffo, “The Zen of Information Technology”, EE380 Seminar, Stanford University, November 28, 1990.

[Sheff93] David Sheff. *Game Over.* Random House, New York, 1993.

[Sing94] Sandeep K. Singhal and David R. Cheriton, Technical Report: STAN-CS-TR-94-1505, Stanford University, Computer Science Dept., February 1994

[Thom81] Frank Thomas and Ollie Johnston. *Disney Animation: The Illusion of Life*. Abbeville, New York, 1981.

Appendix A. Performance Requirements and Human Factors

A 1 Visual Perception

Of all the sensory information we need to provide, the visual component is certainly the most important in producing the illusion of a virtual environment. The ideal would be to provide simulated input with fidelity which matched or exceeded the limits of human visual perception in all aspects. Fortunately, in practice, this is not a prerequisite for producing a usable virtual environment. But in designing a system, we need to always keep the capabilities of the human visual system in mind because the limitations of technology will force many tradeoffs to be made for the foreseeable future.

The human visual system is complex and even basic perceptual thresholds defy simple characterization, usually depending on a variety of factors. The following briefly discusses some of the perceptual limits and how they have been addressed in traditional simulators.

Visual Acuity

Spatial resolution limits can be measured in many ways and depend strongly on many factors including brightness, color, contrast, off-axis eccentricity, length of exposure and retinal illumination.

Visual acuity is commonly measured in terms of the angle subtended at the eye. For reasonably bright objects, on-axis, the limit is around 1 minute of arc. This corresponds roughly to a 20/20 result in a standard vision test, which indicates the ability to recognize letters which subtend 5 minutes of arc. Sensitivity to spatial frequencies is measured by the contrast required to perceive vertical bars and is largest at around 5 cycles/degree [Buse92a]. This indicates that resolutions of around 5 minutes of arc are required to get into the region of peak sensitivity, a limit important for creating sharp edges.

Acuity falls off rapidly as the object moves outside the central 2 degree region. At 10 degree of off-axis eccentricity, acuity drops to around 10 arcmin.

For comparison, a first generation LCD-based head-mounted display (HMD) with around 185 RGB triads across an 75 degrees field of view per eye [Robi91] yields a resolution of around 24 arcmins, or around 20/480 vision. To make resolution matters worse, typically optical blurring was often employed to help fuse the red, green and blue pixels [Teit90a].

Even a “high-resolution” display with 1280 pixels across a narrower 60 degrees field of view per eye achieves only around 3 arcmins.

Standards set for visual simulators are valuable guideposts because they embody years of experience. In this case, the standard for out-the window commercial flight simulators (Level C) set by the Federal Aviation Administration (FAA) requires 3 arcmin resolution [FAA89a].

Temporal Resolution

Peak sensitivity to temporally modulated illumination occurs around 10Hz to 25Hz, with the frequency increasing with luminance. The frequency at which modulation is no longer perceptible, the *critical flicker fusion frequency*, varies from 15Hz up to around 50Hz for high illumination levels [Wysz82a]. On CRT displays, which fill a large field of view and have non-sinusoidal temporal profiles, some individuals still perceive flicker at the common 60Hz refresh rate, which has lead workstation manufacturers to introduce video formats in the 66Hz to 76Hz range. Bright displays with large fields of view can require refresh rates of 85Hz or more [Padm92a].

Luminance

Including dark adaptation the eye has a dynamic range of around 7 orders of magnitude, far greater than any current display device. The eye is sensitive to ratios of intensities rather than absolute differences, and at high illuminations the eye can detect differences in luminance as small as 1% [Wysz82a]. Thus a CRT with a dynamic range of around 100 can display no more than log100/log1.01 = 463 perceptible levels.

For reference, Padmos indicates that contrast ratios of 10:1 to 25:1 are sufficient. The FAA standard for commercial flight simulators requires a 5:1 contrast ratio for scenery and 25:1 for light points [FAA89a].

Color

The human eye can perceive light in the range of 400nm to 700nm in wavelength. Fortunately for the creators of simulated environments, the human eye can't perceive the exact spectrum of light emanating from an object. According to the tristimulus theory, our perception of color starts in three different types of receptors on the retina. Each type of color receptor has a different spectral response with a single dominant peak. The resulting three-dimensional color space can be mapped in many ways.

The CIE chromaticity space factors out luminous energy to yield a two-dimensional color gamut which serves as a benchmark tristimulus based color generation. Most reproduction methods whether printing inks, film layers or phosphors used in CRTs, only cover a portion of the color gamut [Fole90a].

But the tristimulus-based color gamut is not the final word. Colors are perceived differently depending on their context, for example we perceive an apple to have the same color even under a variety of illuminations. This same adaptation that allows colors to "look right" under varying illumination also tends to cause limited color ranges to appear richer than we would expect from the tristimulus theory. Land in his retinex theory [Land83a] showed that stimulation by a combination of only two spectral sources can give the impression of a surprisingly wide range of colors. This could be relevant some displays such as the two-color version of the BOOM [McDo90a].

Stereopsis and Depth

The limit of stereo vision typically occurs for a binocular disparity of 12 arcsec which translates into perceiving the depth ordering of objects separated by 0.1cm at a distance of 1m, 9cm at 10m, and 56cm at 25m [Buse92a].

When looking at computer-generated imagery, the eyes' focus (accommodation) and vergence often do not match as they must focus at the screen or the image plane defined by the optics of an HMD but converge at an angle dictated by the rendered images. And without proper calibration or in a monoscopic system such as a dome, neither focus nor convergence may reflect the actual position of the virtual object relative to the viewer. This causes errors in accommodation (defocus) and vergence (fixation disparity) [Hung94]. Of even greater concern in systems intended for public use is the possibility that oculomotor problems can persist after participation in a virtual environment as a form of simulator sickness.

Out-the-window simulators often use collimated optics to place the images at infinity thereby making convergence and focus match closely for distant scenery.

Many users of stereoscopic systems have trouble fusing stereo images, perhaps because of these inconsistencies. Computer graphics applications which do not need to accurately depict the scale of depth can artificially adjust the parallax to allow more comfortable viewing [Hibb91a].

But for virtual environments requiring close-up manipulation of objects and especially for those requiring accurate registration of virtual objects with the real world, the false depth cues generated by this intentional decalibration are unacceptable. The best that can be done is to calibrate and consider all variables affecting stereo viewing [Deer92a] and carefully choose the size, overlap and image distance of the system to match the task and operating distance.

The blurring of objects due to depth-of-field effects poses another challenge. Depth of field can be rendered using multi-pass techniques [Haeb90a]. But even if methods for measuring or inferring eye accommodation existed, the 2X to 4X performance penalties for simulating depth of field are probably unacceptable. Real-time holography solves this particular problem, but brings a few too many of its own.

Field of View

Each eye has approximately a 150 degrees horizontal field of view (60 degrees towards the nose and 90 degrees to the side) and 120 degrees vertically (50 degrees up and 80 degrees down) [Buse92a]. The FAA standard for commercial out-the-window flight simulators (Level C) requires 75 degrees horizontal and 30 degrees vertical fields of view [FAA89a].

It's important to remember that visual acuity limited to only a few degrees around the axis of gaze direction. Whether in a head-mounted display or a projection system, vast amounts of rendering power are wasted drawing high resolution imagery where you can't see it, because you're looking at something two or more degrees away.

Binocular overlap when focused at infinity is approximately 120 degrees. Overlap varies substantially among different HMDs binocular with some supporting variable overlaps. With the relatively small fields of view (40 degrees to 60 degrees of HMDs used in simulation), large overlaps of more than 50% have been found useful. Because of other problems such as blending of brightness at the borders of overlap, one report found that 100% overlap produced best performance on a visual search task [Edga91a].

Motion

Motion of bright objects can be perceived down to 0.3 min arc/sec [Buse92a]. In out-the-window simulators, the maximum displacement of an object per update which gives an impression of continuous motion is 15 arcmin

[Padm92a]. If carried directly over to HMD usage, a head slew rate in excess of 180 degrees/sec translates into an unattainable 720 Hz. At such high motion rates, temporal antialiasing, i.e. motion blur, could help simulate the temporal averaging of the visual system, but the appropriateness of motion blur for an object also depends on whether the gaze is fixed on the moving object.

Motion of the visual field causes a sense of motion even without corresponding physical body motion. For example, in a simulator without a motion platform the participant gets an impression of self-motion from the motion of objects on the screen.

For rotations, this visually-induced motion varies depending on the axis and the physical orientation of the subject, presumably because of conditioning by gravity. The sense of rotation occurs for roll, pitch and yaw, with sense of yaw being the strongest and roll the weakest [Howa87a]. At very high rates of change in yaw, the sense of motion begins to saturate at 60 degrees/sec [Mooi87a].

This visually-induced sense of motion is tied to field of view becoming “effective” at around 60 degrees and most effective at 180 degrees[Mooi87a].

This sense of motion is accompanied by a perception of a change in orientation, even in the absence of a physical change in orientation. A sense of change in orientation occurs in the opposite direction of the rotation of the visual image and can range up to 45 degrees [Howa87a].

A 2   Temporal Artifacts In Simulated Displays

Field-Sequential Artifacts

Ideally, each pixel should be accurate for the moment it is scanned out on the display device including motion blur (temporal antialiasing) for the period between screen refreshes. In practice this is not feasible and would also unrealistically blur moving objects which the gaze is fixed on. For most applications, the performance penalties for doing full scene motion blur outweigh the benefit, and brute force per-pixel temporal alignment is several orders of magnitude more costly. The bottom line is that relatively few applications can even update the screen at the display refresh rate, let alone worry about these higher quality issues.

Repeating Frames

Most displays run at 60Hz or higher field refresh rates to prevent flicker, but many visual simulations run at lower update rates. This means that a single

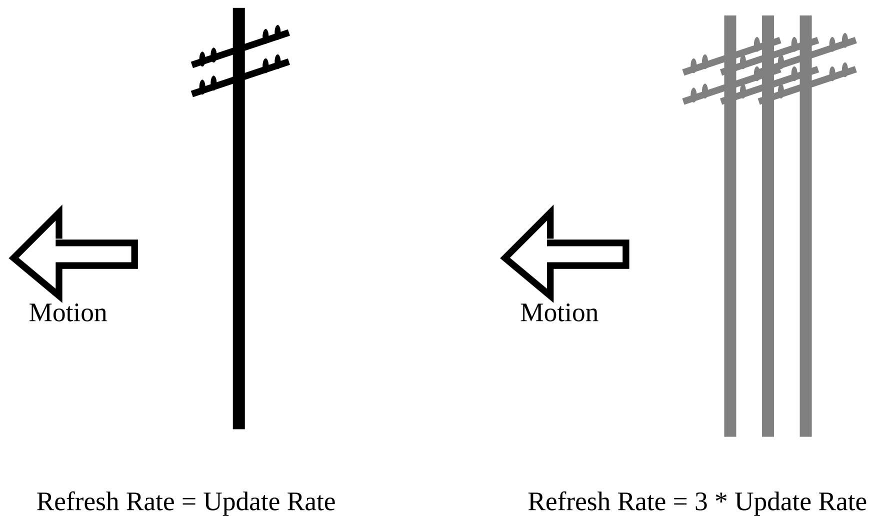

FIGURE 1. Multiple image artifact when fixating on moving object.

image may be scanned out several times before being changed. The sense of motion at the lower update rate is not as smooth, especially when imagery is moving rapidly. In addition, the repetition of a frame means that the image is temporally inaccurate for motion. Real moving objects do not stay in one place for a couple frame times and then move. The result is that when one fixates on a moving object, it appears to split into multiple copies along the direction of motion with the number of ghosts equaling the refresh rate divided by the update rate. So a simulation running at 20Hz update on a display refreshing at 60Hz, the object will appear tripled as shown in Figure 1. On large objects such as horizon silhouette, the effect manifests itself as multiple edges.

Motion blur would mitigate this effect, but poses other problems. In practice, smooth motion and the absence of ghosting are best achieved by an update rate which equals the display refresh rate.

Interlacing

One way to make the update rate equal the refresh rate is to interlace the video so that even and odd line fields are drawn at 60Hz. The graphics hardware can then render the scene with at 30Hz. This suffers from problems similar to those above. But in this case, the temporal inaccuracy for motion causes combed ghosting with edges breaking up into combs due to the interlacing.

Another option is to render with half-resolution at 60Hz (just the even or odd field lines). This has some advantages since rendering latency is decreased by 1/60th sec, motion is smoother and combed ghosting is reduced. As this requires rendering all the geometry in the scene twice, it is only possible for scenes which are strongly limited by pixel fill rates. But many highly-textured databases with simple geometry falls into this category.

Sequential Stereo

Most common CRT-based stereo viewing systems operate by sequentially presenting left and right eye images. For moving objects the common approach of rendering both the left and right eye with the same positions for moving objects is temporally incorrect and produces visual artifacts, including erroneous stereo depth cuing in low refresh rate systems. Lipton reports that the artifacts which are noticeable at 30 fields/sec/eye are not perceptible at updates of 60 fields/sec/eye[Lipt91a].

Sequential Color

Color monitors typically have lower contrast than monochrome monitors because of the shadow mask. Shadow masks also make the manufacture of small, high-resolution color CRTs extremely difficult. Thus a system which uses a monochrome monitor to sequentially displaying each color channel with an accompanying change in color by a shutter has advantages for use in head-mounted displays over low-contrast LCD displays. However, the sequential display of color also generates artifacts. For example, a monochrome monitor may be driven at a field rate of 180Hz thus creating a full RGB image at 60Hz. But if the frame buffer is only updated at a rate of 60Hz, the positions of moving objects will only be correct for one of the three color scans. This leads to an effect similar to poorly registered color printing plates, especially when the eye is tracking an object of high contrast. Updating the frame buffer at 180Hz might alleviate this somewhat, but few if any graphics platforms are configured to drive visuals at this rate, and a factor of three increase in the required rendering is likely to be unacceptable.

Frame Rate Variations

The aforementioned discussions assume a constant frame rate. Varying frame rates pose a number of additional perceptual and practical issues. Fixed frame rates are required in the design of most visual simulators.

A varying frame rate tends to distract the user from the task at hand, particularly tasks involving manipulation of the world or accurate perception of velocities. In particular when going from 60Hz update to 30Hz update, the factor of two is quite noticeable in the quality of motion. Frame rate changes can also false training, if for example a trainee relied on a change in frame rate as an indication of some otherwise hidden feature, such as a large installation hidden behind a hill.

Unanticipated frame rate variations also cause temporal inaccuracies, because the frame does not appear at the time for which it was planned and all visual latencies through the system change as well.

A 3 Latency

Latency is the time measured from the setting of an input until the corresponding output is manifested. Many factors contribute to latency: input devices, software architecture, rendering time, motion response. Different portions of a system may have different latencies, e.g. the response of the visuals to changes in eye point might differ from the collision response which might in turn differ from the response of the motion system. Any path from an input to an output in Figure 2 could take a different amount of time.

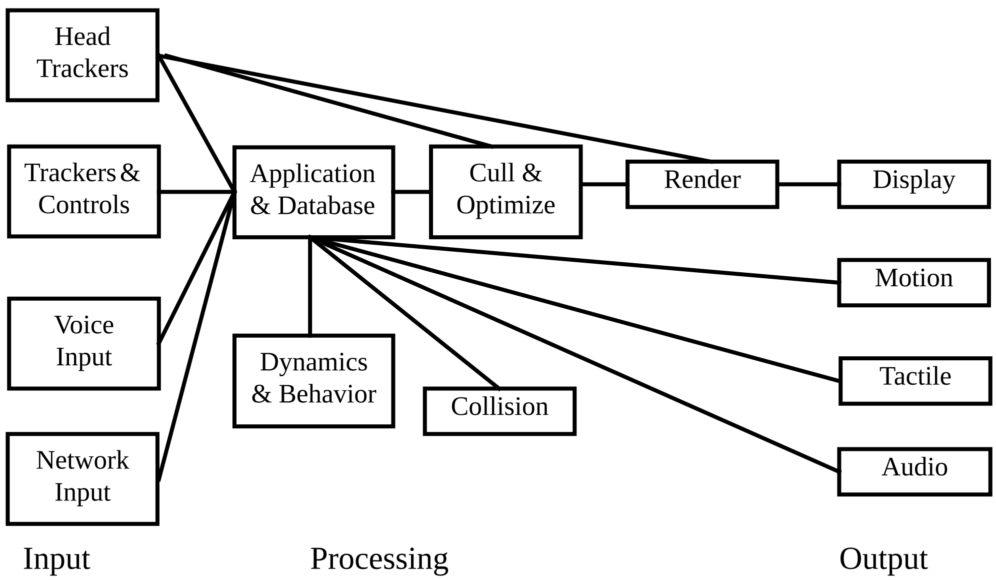

FIGURE 2. Sources of latency in a hypothetical virtual environment system.

For the rendering portion of the system, the latency is typically taken as the time after some value, such as the eyepoint, is set until the last pixel of the corresponding frame is scanned out by the display device.

Typical Simulator Latencies

Reports from a flight training system indicate that the user-perceived quality of the simulation degrades steadily for total latencies over 100ms [Beck92a]. Other studies have indicated that latencies below 100ms have little effect [Card90a]. According to Padmos, in out-the-window simulators, latencies typically range from 40-80ms for driving simulators to 100-150ms for low-maneuverability flight simulation [Padm92a]. The FAA latency requirement for commercial flight simulators (Level C) is 150ms [FAA89a].

Latencies in the 200-350ms range significantly decrease human response rates at measured by transfer functions in the frequency domain [Hess83a]. As is well known from aircraft control theory, this can lead to over-compensation and oscillations induced by the operator.

A 4 Simulator Sickness

The term *simulator sickness* refers to a variety of maladies that result from the use of simulators, both with and without real body motion. The prevalence of simulator sickness in out-the-window simulators indicates that it could seriously impact the usage of VEs. If the first virtual environments to enter main-stream use make users sick, it could damage their acceptance into real-world use.

Virtual Environments vs. Simulators

Virtual environments share many attributes with traditional simulators:

- A visually-induced sense of motion either without corresponding physical accelerations or imperfectly simulated by a motion platform.
- Latencies between events or actions and the manifestation of their results whether visual, auditory or physical.
- Displays with wide fields of view.
- Visual artifacts from inadequate temporal and spatial aliasing.
- Visual artifacts due to inadequate frame rates.
- Display resolutions which at best barely meet human resolution limits.

While most virtual environments may not involve the sort of gut-wrenching maneuvers which fighter pilots go through, VEs offer other challenges not common in traditional visual simulation:

- The slew rates for HMD use are substantially larger than those encountered on most out-the-window visual simulators.
- In HMDs, fixating while turning the head poses latency and update challenges greater than in out-the-window simulation.
- Head-tracked displays are subject to errors in 6 degree-of-freedom trackers.

Many virtual environments involve viewing objects up close. In the real world, such viewing is accompanied by focus distance and depth-of-field effects which are difficult to simulate. Out-the-window applications can often closely simulate the optical configuration of physical world by using collimated optics to place the imagery at infinity.

Proposed uses of virtual environments span many diverse population groups. Unlike pilots these groups are unlikely to have become accommodated through extensive simulator usage and have not gone through a winnowing process which strongly selected for resistance to motion sickness.

Simulator sickness is surprisingly common with 20% to 40% of fighter pilots using simulators suffering ill effects. Some simulators have incidence rates of up to 87% [Mone91b].

This despite the facts that:

- Fighter pilots are a population group highly selected for resistance to motion sickness, and are better accommodated to simulator use than the general public.
- The simulators have stringent human factors requirements
- Most simulators are out-the-window, i.e. CRT, projection or dome based, so the visuals need only track the vehicle motion, and not rapid head motion.

Symptoms

While nausea is a significant and perhaps most obvious feature of motion and simulator sickness, it is not the only or even the principal symptom [Kenn90a]. Symptoms include visuomotor dysfunctions (e.g. eyestrain, blurred vision, difficulty focusing), mental disorientation (e.g. difficulty concentrating, confusion, apathy) and nausea (e.g. stomach awareness, nausea, vomiting). Drowsiness, fatigue, eyestrain and headache are among the more common symptoms [Kenn87a]. In pilot training, one of the main concerns is the persistence of some of these effects for many hours, requiring a recovery period before the pilot is ready for actual flight.

Causes

The lack of much formal research in HMD visually induced motion sickness means we must mostly extrapolate from military and commercial simulators both with and without real body motion.

It's generally agreed that simulator sickness has two prerequisites, a functioning vestibular system and a sense of motion [Hett92a]. The vestibular system is the set of canals, tubes and sacs in the inner ear give us our sense of orientation and acceleration. Individuals without functioning vestibular systems do not exhibit simulator sickness either with or without real body motion [Eben92a].

One hypothesis is that simulator sickness arises from a mismatch between visual motion cues and the vestibular system. This would commonly occur when visual motion does not match physical motion either because no motion platform is used or because the motion platform lags the visuals and cannot match all the accelerations and orientations. In virtual environments, simulator sickness can be expected both in motion based systems (e.g. game pods) and physically static ones (e.g. HMD user seated in a chair).

Wired for Cookie Tossing

Why did this unfortunate response to simulated experience develop? Evolutionarily, it's postulated that in nature the disruption and inconsistency of perceptions which trigger simulator sickness would likely be the side effect of ingesting poisons, and so vomiting is a useful response [Mone91a].

Contributing Factors

Since a sense of visual motion is required for simulator sickness, it's reasonable to expect that many features which make a virtual environment convincing will contribute to simulator sickness. This is born out by studies which have found that bright imagery is more likely to induce sickness than night time scenes, and that wide fields of view cause more problems than narrow ones. Also domes projection systems seem to elicit fewer symptoms than CRT based systems. In motion systems, roll in the 0.2Hz region is particularly nauseogenic.

One author proposes a set of workarounds for pilots using simulators [Mone91b]. Some of these bits of wisdom are paraphrased here for application to virtual environments:

- Don't suggest to users that they will get sick or let them see someone else vomiting. It's contagious.
- Don't go into a VE when your are hung over or have an upset stomach.
- Adaptation is the best fix. Do the VE every day.
- Don't do the real thing the same day you do it in a VE.
- Get set before turning the VE on.
- Set the VE up for night flying.
- Don't roll or pitch too much.
- Don't move your head too much.
- Turn off the VE before getting out.

Until further research is done, we won't know how much additional problems are caused by inadequate resolution, frame rate and latency in HMDs or to what extent BOOM mounted displays [McDo90a] are affected. The initial response to the CAVE [Cruz93a] do not indicate that surround scene projection systems with standing observers are particularly prone to inducing simulator sickness.

But since problems occur even in visual simulators with good visual quality and no head tracking issues, e.g. dome projection systems, we will always be confronted with the prospect of simulator sickness in some types of virtual environments, particularly those with large visually induced motions.

A 5 Conclusions

The human factors requirements of head-mounted displays are in many ways harder to meet than those for traditional out-the-window visual simulation. Many virtual environments today are far from meeting even the standards for traditional out-the-window simulators, which themselves still suffer from many human factors problems. So as virtual environments become more convincing to our senses, it's reasonable to expect that the problems arising from lags and inconsistencies in the sensory input provided by virtual environments will cause significant problems in their use in the real world, particularly in fast-paced applications such as games and entertainment.

While we have some rough guidelines to work with, much more research and actual experience are needed before we will know how to design systems that will not potentially render a significant fraction of their users sick or disoriented. Until that time, careful attention to the design and extensive testing are a prerequisite to the fielding of any virtual environment.

A 6 Bibliography

[Beck92] P.Beckett. Effective cueing during approach and touchdown - comparison with flight. In *Piloted Simulation Effectiveness AGARD Conference Proceedings 513* (Brussels, 14-17 Oct 1991), pp. 30.1-30.11,

[Buse92] P.Buser and M.Imbert. Vision. MIT Press, Cambridge, Massachusetts, 1992.

[Card90] F.M. Cardullo. Virtual system lags: The problem, the cause, the cure. In E.G. Monroe, editor, *Proceedings of the 1990 Image V Conference (Phoenix, 19-22 June 1990)*, Image Society, 1990, pp. 31-42.

[Cruz93] C.Cruz-Neira, D.J. Sandin, and T.A. DeFanti. Surround-screen projection-based virtual reality: The design and implementation of the cave. *Computer Graphics Annual Conference Series '93 (Proceedings of SIGGRAPH '93)*, August 1993, pp. 135-142.

[Deer92] M.Deering. High resolution virtual reality. *Computer Graphics (Proceedings SIGGRAPH '92)*, 26(2), August 1992, pp. 195-202.

[Eben92] S.M. Ebenholtz. Motion sickness and oculomotor systems in virtual environments. *Presence*, 1(3), Summer 1992, pp. 302-305.

[Edga91] G.K. Edgar, K.T. Carr, M.Williams, J.Page, and A.L. Clarke. The effects upon visual performance of varying binocular overlap. In *Helmet Mounted Displays and Night Vision Goggles, AGARD Conference Proceedings 517* (Pensacola, Florida, 2 May 1991), pp. 8.1-8.15.

[FAA89] Airline Simulator Qualification. Advisory Circular 120-40B, Federal Aviation Administration, May 1989.

[Fole90] J.D. Foley, A.van Dam, S.K. Feiner, and J.F. Hughes. *Computer Graphics: Principles and Practice*. Addison-Wesley, Reading, Massachusetts, 1990.

[Haeb90] P.Haeberli and K.Akeley. The accumulation buffer: Hardware support for high-quality rendering. *Computer Graphics (Proceedings SIGGRAPH '90)*, 24(4), August 1990, pp. 309-318.

[Hess83] R.A. Hess. Effects of time delays on systems subject to manual control. *Journal of Guidance, Control and Dynamics*, 7(4):416-421, 1983.

[Hett92] L.J. Hettinger and G.E. Riccio. Visually induced motion sickness in virtual environments. *Presence*, 1(3), Summer 1992, pp. 306-310.

[Hibb91] E.A. Hibbard, M.E. Bauer, M.S. Bradshaw, D.G. Deardorff, K.C. Hu, and D.J. Whitney. On the theory and application of stereographics in scientific visualization. *In Proceedings of Eurographics '91 (Vienna, Austria, 2-6 Sept 1991)*, pp. 1-21.

[Howa87] I.P. Howard and B.Cheung. Influence of vection axis and body posture on visually- induced self-rotation and tilt. In *Motion Cues in Flight Simulation and Simulator Induced Sickness, AGARD Conference Proceedings 433 (Brussels, 29 Sept/1 Oct 1987)*, pp. 15.1-15.8.

[Hung94] George K. Hung and Kenneth J. Ciuffreda. Sensitivity Analysis of Relative Accommodation and Vergence, *IEEE Transactions on Biomedical Engineering. 41(3), March 1994. pp 241-248.*

[Kenn87] R.S. Kennedy, K.S. Berbaum, G.O. Allgood, N.E. Lane, M.G. Lilienthal, and D.R. Baltzley. Etiological significance of equipment features and pilot history in simulator sickness. In *Motion Cues in Flight Simulation and Simulator Induced Sickness, AGARD Conference Proceedings 433 (Brussels, 29 Sept/1 Oct 1987)*, pp. 2.1-2.15.

[Kenn90] R.S. Kennedy and J.E. Fowlkes. What does it mean when we say that simulator sickness is polygenic and polysymptomatic. In E.G. Monroe, editor, *Proceedings of the 1990 Image V Conference (Phoenix, 19-22 June 1990)*, Image Society, pp. 45-44.

[Land83] E.Land. Recent advances in retinex theory and some implications for cortical computations: Color vision and the natural image. *Proceedings of the National Academy of Sciences*, Vol. 80, 1983, pp. 5163-5169.

[Lipt91] L.Lipton. Temporal artifacts in field-sequential stereoscopic displays. In *Proceedings of SID ‘91 (Anaheim, CA, 6-10 May 1991)*, pp. 834-835.

[McDo90] I.E. McDowall, M.Bolas, S.Pieper, S.S. Fisher, and J.Humphries. Implementation and integration of a counterbalanced crt-based stereoscopic display for interactive viewpoint control in virtual environment applications. In *Stereoscopic Displays and Applications*, *SPIE Proceedings*, Vol. 1256, February 1990, pp. 136-146.

[Mone91a] K.E. Money. Signs and symptoms of motion sickness and its basic nature. *Motion Sickness: Significance in Aerospace Operations and Prophylaxis (Toronoto, 7-8 Oct 1991) AGARD Lecture Series 175.*

[Mone91b] K.E. Money. Simulator sickness. *Motion Sickness: Significance in Aerospace Operations and Prophylaxis (Toronoto, 7-8 Oct 1991) AGARD Lecture Series 175*.

[Mooi87] H.A. Mooij. Technology involved in the simulation of motion cues: The current trend. In *Motion Cues in Flight Simulation and Simulator Induced Sickness, AGARD Conference Proceedings 433 (Brussels, 29 Sept/1 Oct 1987)*, pp. 2.1-2.15.

[Padm92] P.Padmos and M.Milders. Checklist for outside-world images of simulators. In *Proceedings of ITEC ‘92, International Training Equipment Conference and Exhibition (Luxembourg, 7-9 April 1992)*, pp. 2-14.

[Robi91] W.Robinett and J.P. Rolland. A computational model for the stereoscopic optics of a head mounted display. In *Stereoscopic Displays and Applications II, pp. 140-160. SPIE Proceedings* Vol. 1457, May 1991.

[Teit90] M.A. Teitel. The eyephone, a head mounted stereo display. In *Stereoscopic Displays and Applications, SPIE Proceedings* Vol. 1256, February 1990, *pp. 168-171*.

[Wysz82] G.Wyszecki and W.S. Stiles. *Color Science.* John Wiley and Sons, New York, 1982.