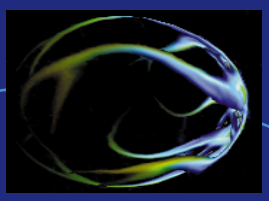


Informal. Surprising. Adventurous.
Risky. Fascinating. Popular.

Sketches & Applications

Animation

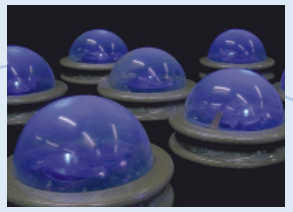
Richard Kidd
Cinesite Visual Effects

Art, Design, and Multimedia

208 A 3D Natural Emulation Design Approach to Virtual Communities
209 The Application of Non-Periodic Tiling Patterns in the Creation of Artistic Images
210 Computational Expressionism: A Model for Drawing with Computation
211 Explorations of New Visual Systems
212 Hyper-3D Paintings in QuickTime VR: Wunderkammer and Hyperaesthesia
213 MSA's Attractors: Navigational Aids for Virtual Environments
214 The Mutable Cursor: Using the Cursor as a Descriptive and Directive Device in Digital Interactive Stories
215 Nami
216 Nicholson NY/Pequot Interactives
217 Passion Spaces Based on the Synesthesia Phenomenon
218 Phene-: Creating a Digital Chimera
219 SaltoArte: Explorations in Spatial Interactive Multimedia
220 Setup of the Konsum Art.Server
221 Virtual Music Reproduction
222 VisiPhone

Technical

223 3D Facial Reconstruction and Visualization of Ancient Egyptian Mummies Using Spiral CT Data
224 3D Gait Reconstruction Using Two-Camera Markerless Video
225 3D Imaging for Rapid Response on Remote Sites
226 3D Physics-Based Brush Model for Painting
227 3D Rendering Effects for 2D Animation
228 A Level-Set Approach for the Metamorphosis of Solid Models
229 An Interface for Transcribing American Sign Language
230 Animating Bird Flight Using Aerodynamics
231 Animating Expressivity Through Effort Elements
232 Application of Computer Graphics for Design and Delivery of Conformal Radiation Therapy
233 Asynchronous, Adaptive, Rigid Body Simulation
234 Automatic Recognition and Mapping of Constraints for Motion Retargetting
235 Capturing the Motions of Actors in Movies
236 Color Super-Histograms for Video Representation: Preliminary Research and Findings
237 Declarative Behaviors for Virtual Creatures
238 Decremental Delaunay Triangulation
239 Dynamic Texture: Physically Based 2D Animation
240 Enhancing the Efficiency and Versatility of Directly Manipulated Free-Form Deformation
241 Fast Polygon Mesh Querying by Example
242 Filtered Noise and The Fourth Dimension
243 Freeform Curve Generation by Recursive Subdivision of Polygonal Strip Complexes
244 Handheld Interactions: Tailoring Interfaces for Single-Purpose Devices
245 Head-Mounted Projector for Projection of Virtual Environments on Ubiquitous Object-Oriented Retroreflective Screens in Real Environment
246 The Holodeck Interactive Ray Cache
247 Image Moment-Based Stroke Placement
248 Image Re-Composer
249 Image-Based Modeling, Rendering, and Lighting in Fiat Lux
250 Image-Based Techniques for Object Removal
251 Interactive CSG
252 Interactive Haptic Modeling of Tensegrities and Network Structures
253 Interactive Rendering with Arbitrary BRDFs Using Separable Approximations
254 LiveWeb: Visualizing Live User Activities on the Web
255 Methods for Preventing Cloth Self-Intersection
256 Modeling HIV
257 The Morphological Cross-Dissolve
258 Multi-Dimensional Quaternion Interpolation
259 Multifluid Finite Volume Navier-Stokes Solutions for Realistic Fluid Animation
260 Oblique Projector Rendering on Planar Surfaces for a Tracked User
261 Occlusion Culling with Optimized Hierarchical Buffering
262 OpenGL Texture-Mapping With Very Large Datasets and Multi-Resolution Tiles
263 Phong Shading at Gouraud Speed
264 Physically Based Anatomic Modeling for Construction of Musculoskeletal Systems
265 Postprocess 2D Motion Blur for Cel Animation
266 Projecting Computer Graphics on Moving Surfaces: A Simple Calibration and Tracking Method
267 Prototype System of Mutual Telexistence
268 Quasi-Linear Z Buffer
269 Real-Time Shadows, Reflections, and Transparency Using a Z Buffer/Ray Tracer Hybrid
270 Real-Time and Physically Realistic Simulation of Global Deformation
271 Real-Time Principal Direction Line Drawings of Arbitrary 3D Surfaces
272 Real-Time Translation of Human Motion from Video to Animation
273 Rendering 3D Objects into Photographs Taken by Uncalibrated Perspective Cameras
274 Representation of the Tactile Surface Texture of an Object Using a Force Feedback System.
275 Shading and Shadow Casting in Image-Based Rendering Without Geometric Models
276 Shape Extraction for a Polygon Mesh
277 Speedlines: Depicting Motion in Motionless Pictures
278 Stereo Analyst: Visualizing Large Stereoscopic Imagery in Real-Time
279 Tangible Modeling System
280 Tracking and Modifying Human Motion with Dynamic Simulation
281 Video Embodiment - MovieSpiral: Towards Intuitive/ Comprehensive Interfaces for Digital Video Interaction
282 Virtual Car
283 Volumetric Modeling of Artistic Techniques in Colored Pencil Drawing
284 Wet and Messy Fur
285 Which Way Is the Flow?
286 WorldBoard: Enabling a Global Augmented Reality Infrastructure



Informal. Surprising. Adventurous. Risky. Fascinating. Popular.

Those are the words that SIGGRAPH conference attendees most often use to describe Sketches & Applications, the forum for works in progress, preliminary drafts of promising inquiries, case studies of proven solutions to tenacious problems, and "the making of" presentations on how today's visual effects technologies are becoming tomorrow's routine tools.

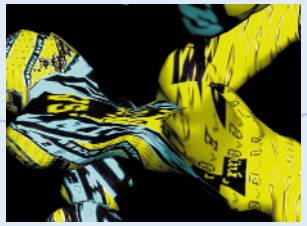
This is where all the disciplines within and related to computer graphics and interactive techniques converge to exchange insight, inspiration, and just plain facts. Every year since it began (as skeptically received Technical Sketches at SIGGRAPH 94), this program has generated more and more submissions and attracted larger and larger crowds.

The jury deliberated, reviewed, and debated for hundreds of hours before selecting 93 presentations in three categories:

Animation
Again this year, animation technology advanced dramatically, and those who use it discovered some surprising and very imaginative ways to apply the technology in feature films and advertising. Fourteen animation Sketches & Applications include sessions on ghosts, creature wrangling, digital corpses, skin movement, animated paintings, cloth modeling and animation, and digital cars. The turn-of-the-century trend in this category indicates that the SIGGRAPH community is moving rapidly toward seamless CG realities.

Art, Design, and Multimedia
These SIGGRAPH 99 Sketches & Applications demonstrate aggressive extension of the principles of computer graphics to new visual systems. The jury selected 15 outstanding examples of how artists, designers, and engineers are always looking beyond the horizon. Topics include: digital drawing and painting, virtual communities, digital choreography, and synesthesia.

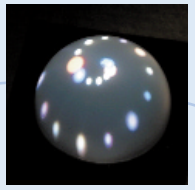
Technical
This year's Technical Sketches range from possible solutions for practical production problems to amazing flights of intellectual exploration. The jury selected 64 presentations that illustrate the very healthy worldwide spirit of adventure in the computer graphics community. Topics include: expressive animation, asynchronous simulation, virtual behaviors, dynamic textures, interactive rendering, mutual telexistence, recursive subdivision, new methods of quaternion interpolation and texture mapping, haptic surfaces, and a 3D physical model of HIV.



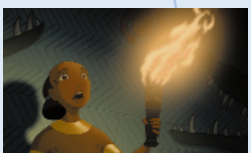
Acknowledgements

On behalf of the Sketches & Applications Committee, I thank everyone who submitted their work for consideration. Your contributions to the international SIGGRAPH community are deeply appreciated. In fact, without your enthusiasm and contributions, the community would not exist.

Many thanks also, to the members of the Sketches & Applications Committee and the SIGGRAPH 99 Conference Committee for their generous support, encouragement, and sense of humor. For their energy, commitment, and competence, I am indebted to Vicki Schaefer and Carrie Ewert, who managed and coordinated the submission, selection, and publication process. And I extend enthusiastic thanks to the other special people who made this program possible: Warren Waggenspack, Nancy Reynolds, Jill Smolin, Robin Myran, Carolyn Roemheld, and my colleagues at Cinesite Visual Effects.



Sketches & Applications Committee



David Horsley
Julija Leary
Industrial Light & Magic

A Ghostly Figure Rising Out of an Evil, Dark Bog: The Making of "The Wraith" from "The Mummy"

In the remake of the classic Universal Picture, "The Mummy," director Steve Sommers requested a spectral apparition of a living person rising out of a bog and floating across an Egyptian burial chamber to rest on Imhotep's lover to bring her back to life. This character became known as "The Wraith" during the post-production period at Industrial Light & Magic.

The animation of "The Wraith" was a collaboration of work completed by a team of character animators and technical directors (TDs) in a rather unusual way, due to ILM'S structure. Typically, the work completed by a TD and a character animator is done consecutively, where the character animator hands off the animation to the TD, who then places it in the scene. For "The Wraith," the TD and the character animator worked cooperatively in creating the final animated look.

The principal animation was developed by creating a series of blended shapes using ILM's in-house character animation software, called Cari (short for caricature). On "The Mummy," the character animator developed the main animation and story for the entire scene, developing the motion for communication of the spirit-like ghost drifting across the set. Along with Cari, Softimage was used by the character animator for spline path animation, articulated chains, and animation by expressions.

The animation was then imported into Alias|Wavefront's Maya software package, where the TD added cloth-like softbody animation on top of the shape animation that had been created by the character animator.

A softbody is a geometry in which the control vertices (CVs) are replaced by particles, which are dynamic objects influenced by forces like gravity, wind, and inertia. The original geometry and its CVs are the target geometry of the softbody and its particles. Each particle is attracted to the corresponding CV of the original object. The challenge with softbodies, and dynamics in general, is to keep a high level of control over them. Very often the dynamic elements in the scene are totally at the mercy of the forces that act on that scene. Particles seem to have a life of their own, which is partly what we wanted, because they can create natural-looking motion. But how does one keep the softbody particles under control, while keeping their natural motion intact?

The motion of the cloth-like material was controlled during the course of the scene using goal weights. At certain critical points, we needed to maintain the integrity of the character animator's intent, so our goal weights needed to be keyframe animated.

When the elements were exported to Pixar's RenderMan for rendering, two procedural turbulence displacement shaders were applied, adding an additional watery feeling. Opacity maps were animated by turbulence to create moving fingers on the edges. Environment maps were also applied to enhance the reflective quality of the surface. The secular component was enhanced by applying an iridescent shader.

The animation of the character animator, after it was imported into Maya, enabled us to use the velocity of the vertices to propel the particles, ultimately giving "The Wraith" an additional directional smoky quality. These particles were rendered separately and later composited in.

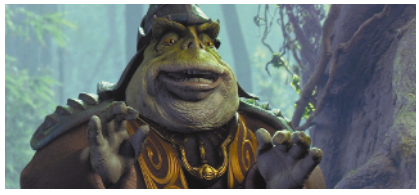
For this sequence, the director gave us a great deal of creative freedom to come up with an interesting effect: "Just make it look cool." Every day, we would create a new character animation and a new simulation, which were reviewed in dailies by the effects supervisor, character animators, and TDs. We went through many iterations to achieve a look that worked.

Cloth Animation for Star Wars: Episode I "The Phantom Menace"

Tim McLaughlin
John Anderson
Industrial Light & Magic

In Star Wars: Episode I "The Phantom Menace," George Lucas populated his alien worlds with fantastic creatures, many of whom wore clothing. This provided a new challenge for ILM. We had created clothing for other shows, such as the Martian robes in "Mars Attacks," but nothing that required the realism and on-screen scrutiny that the clothing for the creatures in Episode I would face.

To meet the challenge, the research and development team went to work in early 1997 and identified the key needs of the characters in the movie and the scope of the problems to be solved. The main focus of the effort was on Jar Jar. As a fully digital co-star of the film, Jar Jar was required to perform a wide range of actions in over 400 shots. His clothing consists of a short leather vest resting on his shoulders and a leather "sweater" wrapped around his waist. Additionally, Jar Jar's loose, floppy ears, while not considered clothing per se, were treated as if they were cloth.



There were two key challenges in designing the procedural animation software for the clothing: development of appropriate representations for the physical properties of the large range of materials that were needed for the film, and development of a set of controls that were to be used to define the performance aspects of the clothing. These controls allowed the artists to intermix traditional keyframe animation and procedural animation techniques in the same shot, and offered a wide range of creative control.

Success with Jar Jar's cloth applications encouraged the crew to use the same technology on over 40 other models featured in nearly 800 different shots. For each shot, the motion of the cloth was treated as an aspect of the character animation. Initial parameters for the cloth, such as stiffness and damping, were developed by a creature developer for each character.

Once the appropriate "look" of the cloth was identified, the files defining it were accessed from the model database whenever an artist was importing the model itself. The animators were able to run the simulation as a stand-alone process or as part of a batch render. For shots requiring special actions, the artists were able to dial between keyframed or fully simulated actions, as needed. Objects in the environment were added as collision bodies, wind-blown or gravity altered, as required by the performance of the shot.

Application of the cloth technology ranged from full clothing, such as that worn by Darth Maul and Boss Nass, to hats, jackets, vests, and even to reins, antennae, and tassles. The clothing software was so easy to use, and so versatile, that the most difficult task was creative: deciding what the cloth should do and how it should look.

Tim McLaughlin
Cary Phillips
Industrial Light & Magic

Creature Wrangling and Enveloping for Star Wars: Episode I "The Phantom Menace"

Meeting the challenges presented by the large number of complex digital creatures needed for Star Wars: Episode I "The Phantom Menace" required the full attention of a handful of creature developers equipped with specialized tools. With over 225 digital models, including nearly 100 photorealistic creatures, we needed to find new, efficient ways to both define the enveloping, or skin movement, of our models, and make sure that the models and their related data files were easily accessible to all artists on the show.

Beginning in the fall of 1996 and working through the spring of 1999, a crew of up to eight creature developers was busy full time managing the zoo of creatures for Episode I. The primary responsibility for this group was to ensure that the models retained the appearance of being organic as they were moved through their actions by the animators. This process, which we call "enveloping" at ILM, involves weighting the modeled surface to its animation controls. When done correctly, the skin moves in a natural manner according to the creature's motion and conveys the impression that the creature has an internal structure of bone, muscle, and fat under a supple skin.

Our new proprietary enveloping software, called Carienv, provided ways of defining the envelopes that were scalable to the performance needs of the creature, from very broad basic envelopes for background creatures to the very fine level of control needed for the hero models. Using Carienv, we were able to quickly scrub through animation on high-resolution geometry and build accurately detailed and robust envelopes. With some creatures, we had ample preproduction time to research relevant real-world creature movement and develop envelopes with proper skin and muscle motion. However, as the blocking of shots often changed, our enveloping system had to provide efficient methods for adjustment. Many times, we had to add complexity to existing envelopes, refit envelopes for creatures with changeable geometry resolutions, and in some cases refit one model type's envelope for use on another.

The secondary responsibility of the creature developer team on Episode I was to make the menagerie of digital creatures accessible to the team of around 40 animators and 70 technical directors putting together shots on the show. Caricature, the parent program of Carienv, was originally developed as a facial animation tool for "Dragonheart." It has since evolved into the tool for assembling all of the geometric information that goes into a shot. This includes animation data, mostly in the form of various Softimage scene files, high-resolution geometry data, envelope data for skinning, facial-animation data, cloth-simulation data, and texture data.

For Episode I, we added features to Caricature that enabled any artist to independently bootstrap a shot and setup and deliver fully textured and roughly comped thumbnail renders, without the assistance of technically skilled support staff. This autonomy enabled animators to work independently of the technical directors, who performed accurate checks of geometry elements before starting a full render.

Because we were often obligated to start animation on shots before final creature development was complete, the creature developers worked to ensure that the various pieces of data being used by each artist were compatible. Changes to models, whether animation, model, envelope, or texture related, were first tested on appropriate scenes. Once they were proven adequate for a wider range of shots, those changes were made available through a central database. For other artists, incorporating the new data was most often only a matter of using graphical file updating features built into the Caricature interface.

For Episode I, we were faced with the task of skinning and managing a larger and more disparate group of models than ILM had ever worked on before. These models had to be malleable to meet a wide range of performance requirements, and they had to be accessible to a crew of well over 100 artists. Through software developments and new techniques, we were able to scale the work required to develop each creature to fit its role in the show while at the same time coordinate quick alterations to models over a wide range of shots.

Creating a Digital World from Scratch: The Launch of the First Union Bank Advertising Campaign

Mary Beth Haggerty
Tim Stevenson
Industrial Light & Magic

Combining several different techniques for a desired effect is a staple of computer graphics production. In Industrial Light & Magic's work for First Union Bank, we employed several different cutting-edge and traditional techniques to achieve our final image.

We were confronted with the problem of creating a financial world consisting of familiar economic phrases and objects. Buildings would be shaped in the form of the words "Sell Now." The textures on buildings would be currency. Sculptural detail would consist of coins, dollar signs, and other elements. It was to become a visually robust world of financial information that, while connected with our current reality, also embodied the unpredictable world of finance.

Here is a breakdown of the opening shot, where we are introduced to this financial world. The shot combines a plate filmed on location, bluescreen people, pyrotechnic smoke, computer graphic people, and cars controlled with a particle system, digital matte backgrounds, computer graphic buildings, and painted buildings on cards. These techniques are combined into a seamless reality.

We filmed the plate and bluescreen people with a motion-controlled camera for consistency. We then imported this camera into the computer. A mock-up scene was created with computer-graphic mock-up models of the buildings to be represented in the shot. We then began constructing the network of roads and walkways that would connect the various buildings and regions in the world we had created. We delivered this city with no textures for the digital matte painters to use as reference. Each foreground building was painted separately. The reference frame for each structure was based on when the building's face was closest to perpendicular to the camera's direction of view. The background digital matte buildings were grouped and painted in a similar fashion. The paintings were then projected onto cards lined up at the reference frame and rendered in Softimage. The 3D computer-graphic buildings were modeled, and projection textures were applied for computational efficiency. Having 3D models in the scene was necessary to cue lighting changes. For the cars and people, we took advantage of the ability of Pixar's RenderMan (version 3.7) to load in different resolutions of data based on screen space. The path animation of the cars and people was done in Alias|Wavefront's Dynamation. The humans had animated cycles applied to these paths for realistic motion.

Intelligent use of different techniques can enhance the look of a production. Through the application of stage, computer graphics, and digital paintings, we were able to achieve images with a visual depth not often achieved with a single approach.

Catherine Craig
James Doherty
Rick Grandy
Industrial Light & Magic

Creating Digital Corpses for "The Mummy"

"The Mummy", Universal's modern remake of the classic horror film tells the story of Imhotep, a cursed Egyptian High Priest.

ILM's Effects Supervisor John Berton, was given the task of bringing the title character to life in various stages of reincarnation, from walking skeleton to full human form and all the steps along the way. This key challenge required digital recreation of an actor, inside and out, that had the look and performance of real human bone, tissue, skin, and body mechanics.

Model Supervisor Jim Doherty, Viewpaint Lead Catherine Craig, and Computer Graphics Supervisor Mike Bauer began the process of creating the digital monster. Reference materials created under the guidance of Art Director Alex Laurant became the bible for the look of the mummy in his various stages of decay. The geometry, built in three main stages, was with six viewpoint texture sets ranging from minimal rotted flesh over bone to a complete construction of Arnold Vosloo, the actor playing the undead monster.

A complete human model was constructed in five interdependent layers: bone, flesh, decayed skin, human skin, and clothing. Stage one had the most variance with rotted skin and withered muscles. Stage two was the next generation and took the creature closer to its final human state. The third major model was sculpted to match the actor and allowed creation of the final stages of Imhotep's transformation.

While the sculpts of the three models differ greatly, they are identical in surface name, orientation, resolution, center, and group association. Keeping these elements matched allowed transformations from one model into another. It also was a time-saving issue. The texture set-up was shared between the models. The surface enveloping for successive models started from the previous model. And the set-up of procedural elements was identical.

Because of the complexity of the creature, the geometry was too dense to get the final sculpt with the model alone. Large level displacement mapping provided the "secondary modeling" used in conjunction with color, opacity, bump, and specular maps.

Animation Supervisor Dennis Turner and Lead Creature Developer Rick Grandy created the animation control system for "The Mummy". Because of the realism needed in the creature's movements, motion capture was used to sample Vosloo's performance. A single animation model incorporated both keyframe control and motion-capture input. It allowed the animation team to refine the motion capture while adding enhancements to it. That same animation model was used for every CG mummy in the film, from Imhotep to the soldiers.

The mummy is the first creature at ILM to simulate the relationship between the skin and its underlying muscle structure. The system was developed out of the need to visualize animated tissue under the surface, which was to be made evident through opacity maps. The skin needed to behave as real skin, and it was decided that the traditional skin technique, a process referred to as "enveloping," would only be the beginning of what was needed. This process puts the animation directly onto the hires surface geometry and becomes the basis for the flesh simulations. Enveloping is a process of assigning a value to each surface control vertice for the amount of effect it receives from each control in the animation model.

The dynamic simulations on the creature were divided into two categories, cloth and flesh. The cloth engine simulated pieces of hanging flesh and cloth that needed to show the effects of inertia and motion through space, such as bandages and cobwebs. The flesh simulation engine was developed to deform the outer skin surface to the underlying bone and muscle geometry. This involved creation of a secondary animation model that controlled individual muscles and allowed for independent and realistic motion.

Creature Modeling and Facial Animation on Star Wars: Episode I "The Phantom Menace"

Geoff Campbell
Cary Phillips
Industrial Light & Magic

The making of Star Wars: Episode I "The Phantom Menace" involved an unprecedented amount of digital creature modeling. A team of 10 creature modelers created six major talking creatures, including Jar Jar Binks, who is one of the central characters in the story, together with 65 other life-like creature models. The duties of the modelers at Industrial Light & Magic involved building the static models, either from scanned maquettes or by sculpting from scratch, building libraries of facial shapes and expressions for the talking creatures, and working with the envelopers responsible for the flexible skin models to sculpt corrective adjustments at the last minute when something didn't look right. The models created for Episode I presented a variety of challenges, from experimenting with early concept designs to quickly putting together new creatures added on the fly as the show progressed.

ILM has a variety of proprietary in-house modeling tools, including software to build surface models directly from scanned data for creatures that were originally designed and approved in clay. A surface sculpting tool, called Isculpt, provides a very direct way of manipulating surfaces by pushing and pulling in ways analogous to modeling with clay. Isculpt makes it easy for modelers to make creative changes to a model after it has been built, whether it originated as a scanned model or was built from scratch in the computer.

ILM's facial animation system, called Cari, is based on shape animation, and the process of creating a talking creature begins with the modeler sculpting shapes that define facial movements and expressions. Cari is designed to handle creatures of very great geometric complexity and visual realism, and to give direct creative control to the artists, so that the process of describing the movements of the face and skin is largely a creative one, not a technical one.

Jar Jar Binks was the first creature created, and he received the most attention from the beginning. There were no maquettes built for him. Instead, the computer model was based only on early-concept artwork, and the director wanted to experiment with different looks before agreeing to a design. In fact, the original design evolved dramatically, due to difficulties in structuring the mouth so that the creature could talk convincingly.

A very different example was the two-headed Pod Race Announcer, which was originally planned as a practical effect. Only late in the process was it decided that the heads would be animated digitally, so the head models had to be constructed very quickly by altering other available models of similar structure.

The digital Yoda that appears at the end of the film represented a far extreme of the modeling experience, as it wasn't until the shot was in production that the director decided to include Yoda. At that point, it was too late to arrange to film the practical Yoda, so we built a digital model and augmented it with some simple facial expressions in a matter of several days.

Another important role of the modelers on Episode I was to correct imperfections in the movement of the skin, both generically and shot-by-shot. ILM uses a variety of techniques, including soft body dynamics, for animating skin, and each presents ample opportunity for the skin to move in ways that are not aesthetically appropriate. A major advantage of our skin animation system is that it allows for after-the-fact improvements that can be made by the artist at the last minute before a shot is finalled.

Eric Daniels
Walt Disney Feature Animation
eric.daniels@disney.com

Deep Canvas in Disney's Tarzan

What if you could paint a painting, then have the brushstrokes themselves come alive and move around?

This is what Deep Canvas is: an animating painting. Rather than texture-mapping, where the end result of the painting process is wrapped onto a surface, this process instead animates the events that make up a painting: the brushstrokes themselves. The process requires an entirely different type of renderer, one that actually repaints the entire painting for every frame.

Created in response to an unusually difficult problem (animating 10 minutes of painterly looking, densely lush jungle for the Disney movie "Tarzan," with a relatively small crew), the process leans heavily on the skills of the traditionally trained artists who work at the Disney Feature Animation Studio.

Early on, it was determined that an "automatic" approach would likely fail to capture the qualities that a good painter, with a lifetime of practice, can intuitively apply to the painting process. Since Disney has on hand dozens of the best painters, it seemed only natural to figure out a way that those artists could best contribute to these 3D scenes.

About a year before, Barb Meier had developed a painterly rendering technique that showed that brushstrokes could be rendered individually to look like a successful "painting in motion." Designed as an automatic process, it was "hard-wired" to create a Monet style of impressionism. Arguably, it could have been tweaked or rewritten to mimic any painting style. But we felt that an even more interesting challenge was to create an approach whereby the style was entirely in the hands of the painter, not the tool. Having seen what good painters can do with nothing more than some colorful goo and a stick with a bunch of hairs on the end, we couldn't wait to see what they could do with a stylus and some intelligent software.

First, we had to create painter-friendly software. Fortunately, we didn't need all the gizmos that a full-featured commercial program would need; we simplified the toolset down to the bare minimum necessary to satisfy the artists. As the artists paint, their brushstrokes are recorded for reuse.

When the time comes to render the 3D scene, the finished painting is disregarded entirely. Instead, each brushstroke is given information as to where it was intended to be in 3D space. Then we begin the "repainting" process, where the renderer, in effect, steps through each "painting event" (stroke) and determines where it should be in camera-space for that particular frame. It then paints that stroke into a buffer.

The cumulative result of all these painting events is called, of course, a painting, for the same reasons that the original is called a painting. The only difference is that in this painting, the brushstrokes might be in a different place than they were when the artist originally painted them. The completed sequence of paintings is an animation; just as a sequence of drawings is an animation.

Because of the unusual nature of the Deep Canvas process, there had to be an extremely tight relationship between the artistically adept people and the technically adept people. In fact, we found that the process went more smoothly the more the skillsets overlapped.

Digital Cars

Bob Hoffman
Digital Domain
bhoffman@d2.com

The use of CG cars at Digital Domain was first employed as a technique in the Plymouth "Neon" 1996 automotive advertising campaign, which presented a number of playful Neons bouncing off of an unseen trampoline. Conventional wisdom had the cars being shot practically, and then manipulated digitally. But while working for Director Terry Windell of A Band Apart, Visual Effects Supervisor Fred Raimondi came to the conclusion that the most artistic and cost-effective way to achieve this "effect" was to build and animate the cars digitally.

Raimondi's thought process went something like this: "What does computer graphics do best? The answer: shiny metallic things. Ding! A car is a big shiny metallic thing, so it should be natural!" With this in mind, he brought CG Supervisor Eric Barba and Digital Domain's Lightwave team onboard to recreate the photo-real look that was crucial to making the spot work. Relying on extremely high-resolution models, the team created all the textures, down to the safety-glass decal on the windows, that allow the viewer to distinguish reality from fiction, and, in this case, practical from CG. According to Raimondi, "lighting the cars became the key element in the process. When real cars are lit, they use a Fisher box, which actually isn't as much lighting the car as it is reflecting a big white card onto the car. So what we had the artists do was, instead of putting light sources over the car, we had them put a big white card over the car. We then lit that card and, like magic, we had a great CG car!"

The basic concept of CG cars took its next step into the mainstream with Dodge "Time" and "Time 2." Using a technique similar to that employed in the "Neon" spot, Director Terry Windell, this time paired with Visual Effects Supervisor Ray Giarratana, had Eric Barba and his Lightwave team create a Dodge Viper and then, for "Time 2," two additional digital vehicles.

General Motors' Phil Guarascio, General Manager, group VP-marketing and his advertising managers saw the success and cost benefits of the "Neon" and "Time" campaigns and in 1997, convened a seminar at Digital Domain for GM executives to determine how GM could take advantage of the latest advances in digital imaging. Of particular interest to GM were situations where practical photography was unavoidable.

This was the case in two spots for General Motors: Chevy Blazer "Roads" and "Obstacles," both of which were created in 1997 and then digitally updated in 1998 to reflect model-year modifications. GM, with Director Eric Saarinen and Plum Productions, and Visual Effects Supervisor Fred Raimondi, was able to realize the cost benefits while receiving original and exciting spots. The spots incorporated original tracking data from existing practical photography shot on location. This allowed new "updated" features to be "added" to the pre-existing campaign, which had been highly regarded the earlier year.

With the 1999 Pontiac campaign for GM and creative agency DMB & B ("Metal City" and "Metal Desert"), Director Ray Giarratana sought to create a highly stylized and entirely CG environment with a photo-real CG car. In particular, the goal of this campaign was to create digital camera moves intended to "mime" the characteristics of practical photography. In so doing, Giarratana, with Digital Director of Photography Eric Barba and CG Supervisor Wayne England, succeeded in pushing the animation to the next level – tricking the viewer into believing that a practical vehicle had been shot and composited into a digital universe.

This latest series of commercials is, in many respects, the culmination of the efforts of Digital Domain to bring CG cars to life, and, perhaps most importantly, they represent the increased flexibility that digital imaging provides in creation of advertising images.

Mark Voelpel
R/GA Digital Studios
mv2000@aol.com

Directing 3D Animated Characters for Advertising: Turning Marketing Strategy Into Storytelling Strategy

A spot is a 30-second movie. While the constraints of a commercial can seem overwhelming, remember how restrictive many classical forms are. Sonnets and Haiku, for instance. A tremendous amount of great work has been created by treating constraints as liberating, so why not take a similarly optimistic and enabling approach to making spots?

To successfully direct a commercial, you must turn marketing strategy into storytelling strategy. The availability of 3D animated characters provides a powerful and creative solution to this challenge. The first step is to reduce the marketing strategy or the message of the campaign to a singular emotion. Then write the story to evoke that emotion from the audience.

The Orkin Man strikes a manly pose while starring in *WAR ON PESTS.*

Don't be fooled when a corporate marketing strategy seems to be more of an idea than an emotion. A story with an idea but without emotion always fails. The story must convey the emotion and then carry the idea along for the ride.

A good story must draw the audience. Typically with a sympathetic protagonist. Well-designed 3D characters can provide wonderful and stylized characters for your stories. Design can build empathic protagonists, heroes, and villains, according to the needs of your story.

Use the language of cinema to persuade your audience to identify with your characters. Hitchcock persistently demonstrated that the most effective way to build strong psychological audience identification with a character is to use the POV sequence. Take advantage of it!

The Honey Bear, played by Dom Deluise, yells for help in *SPYGUY.*

Visual storytelling is, simply, editing. To edit is to speak film language. If you don't speak it yet, hire someone who does or take the time to learn it.

To create the most dynamic and powerful shot sequences and to do so while working for (and hopefully with) clients who are often untrained and unwise in the crafts of editing, filmmaking, and animation, I suggest the following approach:

1. From the first day of the project to the last day, look at sequences edited to time (30 seconds, on occasion). Many a gorgeous storyboard simply won't cut, so on the first day of the production, scan in the agency concept boards and cut them to time. This cut will immediately demonstrate the weaknesses (and strengths) of those agency boards.

2. Show this cut to the agency team, so they can see with their own eyes what does work and what doesn't work in their board.

3. Now rewrite the story as necessary. Generate a new storyboard, scan the frames, and edit them to time. A scratch track should be used at this stage, if possible. Study this "board-o-matic," analyze it, and then modify the story or storyboard as needed to make the story stronger (to more strongly evoke the emotion of choice).

4. Once this "board-o-matic" is working and approved, direct a small animation team to create a 3D animatic with appropriate lenses and camera moves. This 3D animatic will finally let you know if the shot sequences will work in what will be their final form: 3D computer animation. Change timings, lenses, framings, or anything else as needed.

5. Is the story told? Is the emotion conveyed? If so, now (and only now) are you ready to enter into full production with your production team. You have a story, and you have the shot sequences to tell that story.

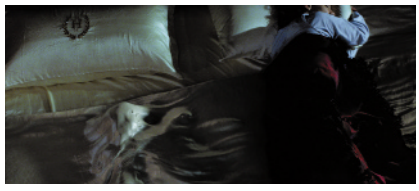
This method can be used for live-action, especially effects sequences, to great advantage and for great fiscal savings. So, while you exploit the mobility and weightlessness of your virtual camera, don't forget to take full advantage of the power of the 100-year-old language of cinema. After all, for better or worse, it's become the universal language of our time.

The Haunting

Craig Hayes
Tippett Studio
craig@tippett.com

This sketch analyzes two scenes from "The Haunting." In one scene, the heroine combs her hair in front of a mirror. She watches in horror as her hair takes on a life of its own, culminating in an elaborate braid before she shakes free of the ghostly encounter. The second scene involves the sleeping heroine. As she drifts into sleep in a giant antique bed, a ghost child enters through the window, causing a CG silk curtain to billow and rise. The ghost child continues her journey by traveling under the satin sheet of the bed and up to the pillow where her silhouetted form dissipates.

The CG hair had to be completely photorealistic, even when shown in close-up and inter-cut directly with the actor's real hair. A strictly procedural-based method of growing hair was not feasible, since the hairs had to grow in an orderly yet chaotic manner (uniform but unique at the same time). For maximum control over the modeling, the process started with a foam head with armature wires inserted into the scalp. These wires were digitized and represented by splines in the computer. Since modeling and animating each hair would have been overly laborious, software was applied to a reduced number of hair splines as guides for growing additional hairs to densely populate the head. The hair had to conform to the physics of the reality. Not only did it have to respond to gravity and inertial forces, but it also had to be combed. The hair had to transcend the realities of action and reaction, fall under supernatural control, gather itself together, and twist itself into a bun.



The silk sheet shot involved creation of CG fabric that takes on the shape of a child as it moves from the bottom to the top of the bed. The bed sheet utilized pseudo-ray traced renders to create displacement for geometric objects. These renders, together with standard key-frame animation and manipulation of a digital puppet, produced dynamic ripples along the object, creating the effect of a child moving through the satiny sheet, leaving folds and gathers in its wake.

Unlike other attempts to deform geometry, Tippett Studio's dynamic particle simulation (using a custom-spring algorithm) does not produce stretching, so the sheet doesn't look rubbery. It has all the natural movement one would expect from a sheer material like satin. The goal was to show a child in a sheet, not a child under a sheet.

Karen Ansel
Mobility, Inc.
karen@mobus.com

The Making of the Painted World: "What Dreams May Come"

The painted world of "What Dreams May Come," is transformed over 8.5 minutes of live-action photography into moving painted imagery in the style of 19th century painters, such as Casper David Friedrich and Claude Monet. "What Dreams May Come" required unique analysis of how artists and computers perceive and interpret the world.

Vision
In order to present the style of painting that the director Vincent Ward felt most appropriate to the emotional pitch of a shot, our digital artists learned to view the live-action footage using an eye similar to that of a fine artist working in oils. Additionally, the shots reflect the emotional state of the character, who was starting to realize his ability to transfer his environment to the painted world.

Process
The live-action plates were analyzed for motion to retain the feel of the natural world, such as wind blowing through flowers and trees or water ripples over a lake. The plates were then graded as a source of color palette for the paint particles, based on a style of painting influenced by 19th-century artists. Areas of the plate were recognized and isolated as layers to be applied with particular paintstroke styles, much as a painter will fill and layer details on the canvas.

The live-action plates were re-constructed in 3D space. We could then "grow" plants, flowers, and grass, and replace trees as elements to be composited back into the painted plate. We also added 3D effects such as light beams, cloud shadows, and waterfalls to either the live-action plate or the matte painting to enhance the otherworldly light and atmosphere inherent in the nature of the painted world.

Technical Challenges
The painted world sequences required untreated actors who freely move about and interact with the environment. The painted world also required the spatial relationship of an immersive 3D environment, with the effect of a 2D application of Paint. The viewer would have to register this as a painting yet also believe it to be a 3D world. We had to retain the motion of the live-action plates and the natural world. Motion-control rigs could not be used during filming in the wilds of Montana. The end result had to include character interaction with the environment, yet their rendering was not to be "painted." To meet this challenge, software called Motion Paint was developed in-house during the production. This toolset allowed us to match motion and spatial changes within the scene and attach 3D objects (such as wet paintstrokes) to the motion data. We were able to develop fast interaction between the artistic demands and technological solutions throughout the production.

As the artistic ideas were developing, we developed the software to meet these challenges. For example, one shot of grey static sky shouted out for a transformative Van Gogh swirling sky effect at the point in the story sequence where the character is starting to realize his influence on his environment. We were able to add additional software features that were subsequently used throughout the entire sequence.

New Vision
"What Dreams May Come" presented the opportunity to utilize 3D animation and reconstruction in a realm far from the "creature feature" of recent years. The technologies developed are of the lineage of visual effects that films like "2001" realized, where subjective immersive environments are key to the cinematic experience.

Multiple Creatures Choreography on Star Wars: Episode I "The Phantom Menace"

Marjolaine Tremblay
Hiromi Ono
Industrial Light & Magic

Boom! A battle droid has just been destroyed, but other droids keep moving forward. That is how our battle with the particle begins.

Creating the Naboo ground battle on Star Wars: Episode 1 "The Phantom Menace" was not an easy task. Organization and efficiency were our best allies. Working in Softimage, a group of character animators created a series of animation cycles needed for the battle while a group of technical directors (TDs) created and rendered particle choreographies in Alias|Wavefront's Maya.

The complexity of each of the cycles varied. Character animators often had to combine motion-capture data with keyframe animation, run a series of cloth simulations, and create shape animation for each of the creatures or droids.

Each cycle needed to have a high level of detail. A cycled creature might be placed right next to a hero character or far from the camera. We could not play our animation to a certain camera angle like a normal shot because the same cycles were used in various shots with different camera angles. This meant that each and every cycle had to be working from the back, the sides, or the front. It elevated the amount of work and detail necessary in a cycle, pushing the animators to perfect their animation techniques. Combining two or more characters was also a challenge, since they needed to be treated as a single unit.

Using a proprietary motion database worked to the character animator's advantage. The tool helped us manage and store the ever-growing number of motion cycles. It allowed us to follow an easy naming and numbering convention for each of the files needed for each individual creature cycle.

When an animation cycle was finalized and checked in the motion database, the TD took over. The animation cycle data was converted into renderable hires, midres, and lores rib files, as well as a more simplified polygon mesh called cube data, which was used to represent the creatures in Maya.

Based on the creature's distance from camera, the latest RenderMan version let us render the creatures using the different levels of rib files.

During "Return of the Jedi: Special Edition," a new technique allowing thousands of human figures to be choreographed with particles was in its infancy. As it evolved, we wrote a plug-in for Maya that displayed cube data, allowing us to manage more creatures in a growing number of sequences on a show like Episode 1. The plug-in directly draws the geometry using Open GL display lists, which allows us to check the massive number of creatures that were animated in a shot. Several plug-ins and expression scripts were also written to enhance control.

In Maya, each loaded cube was assigned to a particle. For each of the particles, an animation cycle and a frame range was set using a particle rule. We created several generic rules for the particles that let the TD procedurally choreograph the particles. We positioned the particles on the ground and animated them forward, sideways, or turning. The velocity was determined by the rules and by Maya fields and collisions, but each cycle had a distance curve that restricted the magnitude of the displacement, keeping the creatures from sliding on the ground.

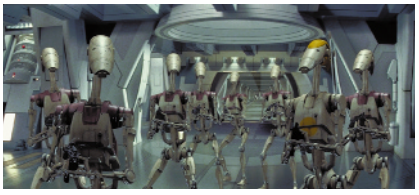
When a choreography was approved, we saved all the necessary information for rendering, such as materials, colors, and cycle data. To win the battle and keep total control of the crowd behavior, we used a procedurally choreographed-animation-cycles method instead of behavioral simulation techniques.

Finally, all the technologies began to work together, allowing us to animate and render an unprecedented number of detailed creatures with elaborate and varied motions. At last, the scene with all its creatures came together, and the grandest digital battle ever filmed took shape before our eyes.

Star Wars

James Tooley
Jim Hourihan
Industrial Light & Magic

Technical Animation Issues for the Battle Droids of Star Wars: Episode I "The Phantom Menace"



The battle droids in Star Wars: Episode I "The Phantom Menace" play a large role throughout the film. They are seen invading the city of Theed, engaging in action-packed combat sequences with Jedi Knights armed with light sabers, and battling an army of Gungans on the grassy plains of Naboo. As a result, two distinct factors were evident in the decision to use motion-capture technology in the early preproduction phase:

1. They were, by design, very humanoid in nature and, as such, required a hyper-realistic range of motions.
2. They appear in large numbers making the duplication of basic actions and motions a necessary production need in order to complete the shot.

Motion capture technology was utilized for the motions of the Battle Droids. There were over 200 selected samples of motion-capture data of the Battle Droids. The majority of the captured motions were designed to be generic movements such as run cycles, walk cycles, marches, shooting weapons, dodging movements, retreating movements, and dying. There were, however, numerous shots requiring a specific action such as walking down a staircase on a practical set or pushing through the Gungan battle shields.

For practical reasons, we needed the resulting motion on the same controls that the animators would regularly use. In this way the motion can be altered, blended, or discarded for any of the shots in the film. To meet this need, software was developed that converted the data onto any constrained animation hierarchy. This allowed for the conversion of motion capture data directly into the animation control systems. The converted motion information appeared as if it had been animated by an artist but had the controls available to manipulate the captured data.

With this technology, we were also able to use motion capture for various Gungans and distant pod race attendees even though the majority of the Gungan animation, including the Jar Jar Binks character, were keyframed.

In the film, the Battle Droids are no match for the powerful Jedi Knights. Wielding light sabers, the Jedi Knights are able to slice through the droids like a hot knife through butter. Ultimately, there were over 25 shots in the film that depicted 49 sliced droids. To achieve this effect, several versions of the sliced droids were created.

Each Battle Droid consisted of at least 204 dynamic elements, including 50 rigid bodies and 154 dynamic constraints. The actual slicing of the droids was achieved by using a rigid-body system created specifically for Episode 1.

The action of a droid was converted and read into the interactive dynamic system. The basic skeletal "bones" were converted into rigid bodies. At that point, dynamic pin was used to represent a ball and socket connection (three degrees of freedom) and two pins created a hinge-like connection (one degree of freedom). The geometry of the droid became the collision model that would then hit the ground or help to keep parts of the droid's body from penetrating other body parts. Other dynamic elements, like springs, helped to create an illusion that the droid's joints were stiff and had some resistance to external forces, thus altering their poses.

The Battle Droid sequences presented many technical and creative challenges for the motion-capture and physically based technology and specialists at Industrial Light & Magic. As with many of the techniques created for production needs, the images created using motion capture and dynamic simulations for Episode 1 have laid the groundwork for basic expectations that will become perfected as the ability to integrate these techniques into production grows.

Viewpainting Models for Star Wars: Episode I "The Phantom Menace"

Jean Bolte
David Benson
Industrial Light & Magic

One of the most significant aspects of successfully combining computer-generated models with live action has to do with the quality of the paint on the model. Computer model painting involves not only color, but also detailed sculpting of wrinkles and aging as well as any additional texture, such as fur. At Industrial Light & Magic, this process was first used on the dinosaurs in "Jurassic Park." With the development of an in-house paint software called Viewpaint, not only wrinkles and fur, but also feathers, rust, metal patina, decay, and corrosion have been incorporated into the models. The stage has been set for just about any look, from photorealistic to stylized, to become a possibility.

With Star Wars: Episode I "The Phantom Menace," facing a number of new and different challenges became the rule, not the exception. With over 200 different characters, creatures, props, and hard-surface models (some aircraft had as many as 5,000 individual pieces), the number of shots completed was unprecedented in any effects film to date. Clearly, handling the sheer size of the project, in a relatively short time, became the biggest challenge for ILM's Viewpaint team. Working closely with the artists, who in turn made suggestions based on their experiences with the tool, the software team came up with a number of upgrades that provided a dramatic increase in our efficiency and productivity.

One of the challenges we faced was the introduction of CG characters wearing clothing, a first for ILM. As painters, we needed to pay close attention to the subtleties of material: how wrinkles fold and bend, how shadows of folded material are impacted and so on. Our software team wrote features into the package to facilitate our need. For example, wrinkle maps were created for Jar Jar's shirt and trousers, which were painted in sequence then applied as displacement maps, somewhat like replacement animation. In order to see how the paint would stretch, where the point of impact occurred in crash sequences, etc., we wanted to see our paint jobs in motion. Again, working closely with the Viewpaint artists, the software team added the features that allowed us to accurately assess the painterly movement of fabric on the CG characters.

The character Jar Jar Binks is a good example of one of our creature paint jobs. Because he is a major character with an enormous amount of screen time, he came under much scrutiny during the development phase in pre-production. He was painted with color, bump, and specularity maps. Additionally, to make sure that he would hold up well during extreme close-ups, we used the smallest paintbrush possible. During this presentation, Jar Jar is examined in stages, from the first concept art to the finished paint job, showing all the layers of paint that were necessary in the creation of realistic skin.

After we demonstrate the painting progression on one lead character, we show a brief montage of all the creatures and hard-surface models, including interiors, that ILM's Viewpaint team was responsible for in the completion of Star Wars: Episode I "The Phantom Menace."

Steve DiPaola
David Collins
Communities.com
steve@dipaola.org

A 3D Natural Emulation Design Approach to Virtual Communities

The design goal for OnLive's Internet-based Virtual Community System was to develop avatars and virtual communities where the participants sense a tele-presence, so they are really there in the virtual space with other people. This collective sense of "being there" does not happen over the phone or with teleconferencing; it is a new and emerging phenomenon, unique to 3D virtual communities. While this group presence paradigm is a simple idea, the design and technical issues needed to begin to achieve this on Internet-based, consumer-PC platforms are complex. This design approach relies heavily on the following immersion-based techniques:

- 3D distanced, attenuated voice and sound with stereo "hearing."
- A 3D navigation scheme that strives to be as comfortable as walking around.
- An immersive first-person user interface with a human vision camera angle.
- Individualized 3D head avatars that breathe, have emotions, and lip sync.
- 3D space design that is geared toward human social interaction.

These techniques, which borrow from disciplines such as group dynamics, facial animation, architectural design, virtual reality, and cognitive sciences, allow the system to draw from the natural social neural programming inherent in all of us rather than creating artificial, social-enabling user interface mechanisms. The main goal of all of these techniques is to support multi-participant communication and socialization.

Community Comes from Communication: 3D Voice with 3D Navigation
The structural process of a community, whether real or virtual, is communication, and the most natural human form of communication is verbal. Verbal communication has both the explicit and the implicit message encoded in it.

We designed 3D spatial, multi-participant voice-with-distance attenuation and stereo positioning. Avatars closest to you are heard the loudest; those to your right, louder from your right speaker, and so on. Using this approach, the user interface becomes as simple as navigating towards the avatars you want to talk to and away from those you no longer want to talk to, just like you would at a real cocktail party. By using spatial sound with 3D navigation, natural group dynamics situations occur. Several small circular conversational groups of three to six avatars form and dynamically reform depending on individual and group social preferences.

Avatar Design: Binding the Pair – You Are Your Avatar.
Given the finite CPU/polygon/bandwidth resources, we invested them first in face-based avatars. The body, with its hand gestures and body language, is secondary for human communication and can be added as our resource limitations improve. The goal for us is what we call "binding the pair," binding the real person at the computer with a virtual avatar in cyberspace so the user experiences this feeling of tele-presence, of really being there. You cannot believably bind a person with an inanimate object or a texture-mapped photograph that does not emote. We tried to achieve "life" and believability with avatars that have autonomous blinking and facial movements (for example, "breathing"), that lip sync to their voices, and that can display (at user control) a range of emotions.

We now have some early positive indications that this technique is working, because it has been noticed that users make "eye contact" with each other. They turn toward the speaking avatar and can feel uncomfortable when another avatar comes too close and "invades their personal space." This last point was very encouraging considering our goal of "binding the pair." If someone in real life comes too close to you, you feel an uncomfortableness along with a physical tightening of your stomach muscles. This same sensation happens in the OnLive worlds, which shows that users perceive at some level that they are really there with other people, and that avatars are perceived as beings, not as objects being manipulated by other users on their home computers.

Evolving Issues: Getting Beyond Environment and Into Community Creation.
Even if we achieve all these goals (which we are still far from doing), we would end up with users feeling they are their avatar, and that they are really in a place with other avatar beings, collectively existing in the same virtual place. However, this merely provides infrastructure (the air, streets and buildings of a community), not a community. As we tackle the infrastructure issues and welcome thousands of people at our sites every day, we are just beginning to grapple with the social community issues. This requires the involvement of social engineers, cognitive psychologists, event producers, reporters, and sociologists, to begin to understand the nature and requirements of a virtual community.

Acknowledgements: Rod MacGregor, Henry Nash, Dave Owens, Ali Ebtekar, Stasia McGehee, James Grunke

The Application of Non-Periodic Tiling Patterns in the Creation of Artistic Images

Kenneth A. Huff
ken@itgoesboing.com

The physical world contains infinite examples of patterns of repetition, from the honeycomb of a beehive to the radial and spiral patterns found in a sunflower. While these patterns contain recognizable structure, they also have imperfections or a level of randomness. Computer-generated imagery often contains similar patterns, but frequently without the subtle imperfections found in nature, thus alerting our minds to the synthetic aspects of the image. The use of a non-periodic tiling pattern allows for an appealing structure along with natural randomness.

In 1704, a Dominican priest, Sebastien Truchet, painstakingly analyzed patterns by systematically placing multiple tiles, each divided by a diagonal line, into two different colored sections. Truchet's purpose was... "to be able to form pleasing designs and patterns by the arrangement of ... tiles." While Truchet was primarily concerned with symmetry and periodic repetition, these simple tiles may also be used as the basis for non-periodic patterns.

A pattern of continuous closures can be formed by using a grid of tiles, each consisting of a square divided into three sections by two quarter circles placed on diagonally opposite corners, and randomly orienting each tile constrained to 90-degree increments. The pattern is formed by the quarter circles. As is often the case with patterns in nature, these patterns contain a recognizable structure along with random qualities.

Additional levels of complexity can be produced by combining, or layering, a number of these patterns. The interference between layers can produce many unique patterns, each maintaining structure while remaining non-periodic.

Using a wide combination of commercially available software, the author has used this non-periodic pattern in various works, sometimes as a primary element, often to add an additional level of subtle detail.

Figure 1
"99.1," 1999. The background pattern of this image is made up of four layers of a duplicated continuous curve. While similar in appearance, the pattern is not based on a non-periodic tiling pattern, but rather was created freeform.

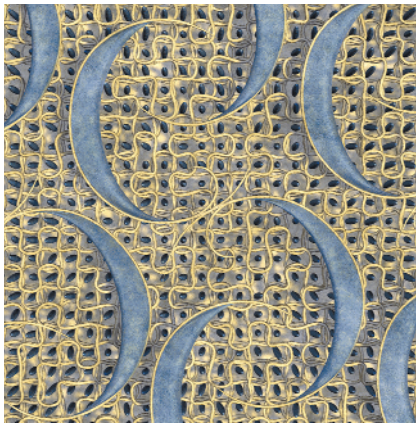


Figure 2
"99.4," 1999. The geometry in this image is based on a non-periodic tiling pattern. Two grids of tiles were layered to form the basis for the profile curves used to create the geometric forms.

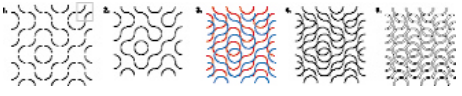
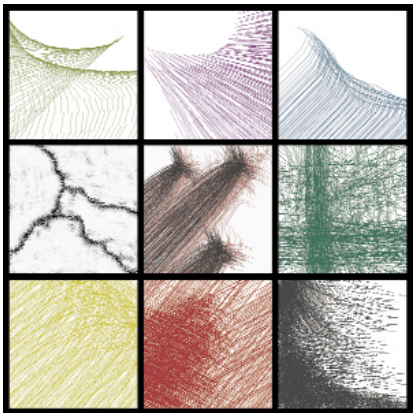


Figure 3
The development of the pattern used in "99.4". 1) A custom typeface was developed for two characters, one for each tile. A single character is highlighted in the illustration. A list of random characters was generated to form a grid. 2) Spacing of the characters was adjusted to remove gaps. 3) The grid was duplicated and offset from the original. 4) and 5) Additional manipulation produced the final 2D pattern.

Joanna Maria Berzowska
Massachusetts Institute of Technology
joey@media.mit.edu
www.media.mit.edu/~joey/x/

Computational Expressionism: A Model for Drawing with Computation



Computers have introduced new concepts into the artist's vocabulary (interactivity, dynamism, and emergence) that shape the development of new forms of visual expression. In particular, the computational medium suggests a new approach to the act of drawing. Instead of duplicating the methods and materials we know from traditional media, we seek to develop a different perspective on visual thinking, one that involves a more active participation in the higher-level design of drawing tools.

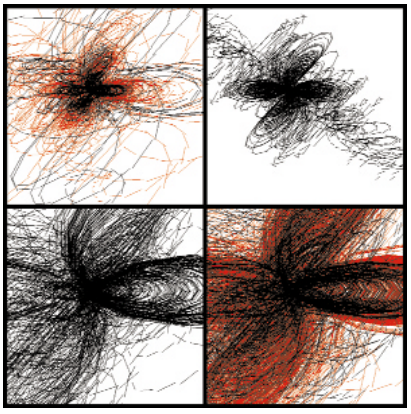
The work consists of 24 Java-based drawing programs, each constructed and used by the author to create specific series of drawings. We define the "computational line" as a sophisticated drawing element controlled by various parameters of a gesture, such as speed, direction, position, or order. Complex shapes can be created with very simple gestures. A computational line has three attributes:

1. Physical appearance (a shape, a pattern, a representation of a mathematical algorithm, or a color).

2. Individual behavior or dynamism, some transformation over time (a shift in color, position, shape, or other attribute).

3. Behavior in its interaction with the other lines on the canvas. The line can sense the presence, proximity, and topology of others and respond in pre-determined but unpredictable ways.

Computational expressionism presents a model for drawing that combines higher-level conceptual design with real-time gestural input. It is a two-fold process, at two distinct levels of interaction with the computer. The artist programs the appearance and behavior of computational lines and then draws with these by dragging a mouse or controlling another input device. The process of computational sketching consists of writing the code, compiling the program, running it, and making some sketches on the digital canvas. The drawings thus produce the next iteration of the code, and the algorithms are refined and varied. The iterative bipartite process of programming and drawing generates an individualized algorithmic style for a particular drawing.



The computer differs from other media in that it is both a constructive and a projective medium. Computation invites us to represent, but also to interpret the world around us, through experimentation and abstraction. In this work, the artist develops individual interpretations of the meaning of gestures and constructs tools to represent them in the final composition. This conceptual interpretation of gestures is a very personal part of the process and a key element of the model of drawing with computation.

Because computational lines are no longer literal representations of a gesture, the aesthetic qualities of an algorithm or of a dynamic interaction generally outweigh representational potential. Thus our work in computational drawing has been almost exclusively non-representational. The quality of lines is a strong aesthetic theme, together with the expressive potential of gesture and the textures and tones achieved through a layering of lines. The compositions emerge from the interaction between the algorithms and behaviors of the computational lines and the gestures used in drawing. The resulting aesthetic takes advantage of the medium's unique characteristics (interactivity, dynamism, and emergence) and highlights the tension between repetition and variation, regularity and irregularity, mathematical order, and gestural disorder.

Expressionism

Explorations of New Visual Systems

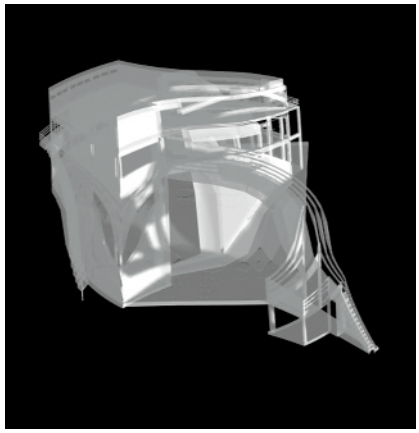
Kostas Terzidis
University of California, Los Angeles
kostas@ucla.edu

Perspective systems are designed to construct pictures that, when viewed, produce in the trained viewer the experience of depicted objects that match perceivable objects. Our capacities to see are constrained by the perspective system that we use (that is, by our way of depicting what we see). Mathematicians showed that geometries could be based on axioms other than Euclidean. This altered the notion of geometry by revealing its abstract postulational character and removing it from any connection with the structure of empirical intuition. Spaces were constructed on the basis of non-Euclidean axioms, revealing behaviors closer to our sensations than our perception.

This sketch investigates how computers and new media may extend the designer's perception and imagination. It presents a series of experimental mathematical functions that demonstrate some of these models or mappings for a variety of values for the parameters. The functions address geometric mappings as well as numerical models of projection, and their interest lies in the dynamic nature of continuous computer processing (real-time movement).

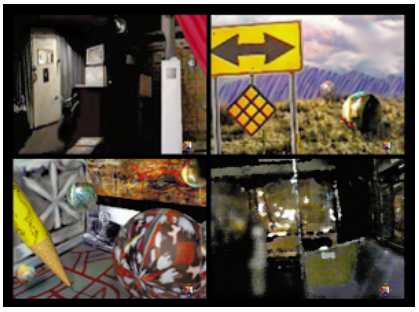
Some of these experiments investigate the implementation of art or design theories. For example, what would it look like to move inside a cubist world? Other experiments explore the effect of non-Euclidean theories in the exploration of visual systems. For example, how would an inverted perspective representation behave in a hyperbolic world? Most of the experiments can be combined together in rather unusual ways, such as, for example, a cubist world with supremacist shapes seen through a pointillist filter in real-time animation.

The computer system that was developed for this sketch is called "zhapes." It is a Java-based 3D-visualization system that resides at cda.ucla.edu/caad/java/x/formProj2/formB.html, where it can be downloaded for explorations.



Philip Sanders
The College of New Jersey
ps@thing.net

Hyper-3D Paintings in QuickTime VR: Wunderkammer and Hyperaesthesia



Wunderkammer series: Initial view of four linked QuickTime VR panoramas.

The Wunderkammer and Hyperaesthesia interactive panorama series explore culture and consciousness. These are photographic/painted constructions based on the idea that the computer, a "universal solvent," relates to early studio/museum/expositions that integrated art and technology. Composed of interchangeable virtual worlds, these series present a dialogue between the internal psychology of each individual viewer and external physical or social realities, juxtaposing actual-seeming places with archaeological, painterly images. Although the panoramas seem to present a photographic realism, everything has been constructed or revised with digital techniques. Meanings are developed through viewers' responses. The works may be characterized as the extension of painting into four dimensions.

The idea is to create an environmental metaphor that is not real, a resonant framework, world: artist: artwork: viewer.

Antecedents include:

- Early design of user/viewer experience passed through multigenerational digital technology evolving into user interface design.
- Early 3D installation and 3D paintings transformed by the computer into virtual worlds existing as data displayed as patterns of light.
- Early photography and painting conflated by digital processes into digital imaging.
- Historical exhibits – early, mid, late 1980s in the East Village, an outpost for artists experimenting with art and technologies-migrating from physical to WWW space.

Traditional paintings are built up layer by layer, each successive layer superseding and covering what came before. There is a regret for what is lost, a nagging interest in the dynamic permutations of the process, the archaeology of a painting. With digital image creation, an artist can save each stage of the work, and display it as an animation or a series of stills. Extending this temporal process along a spatial axis lets the artist give a viewer the ability to explore and recreate this entire process of creation via a temporal/spatial record of the work in three dimensions using technology such as QuickTime VR.

The ability to view any digital image all the way down to individual pixels lets artists create images on a microscopic level, as well as the photographic hyperrealism available at the conventional, "zoomed out" view. QuickTime VR allows zooming into alternate, microscopic realities, enabling works that can simultaneously contain abstraction, realism, and surrealism.

New technologies tend toward work utilizing collage, montage, and vision in motion. Just as word processing altered writing and editing, collapsing it into one process, and desktop publishing gives every computer user the freedom of the press, digital imaging and authoring transform possibilities for artists. As the technologies of photography and film collided with and influenced modern art at the beginning of the century, digital processes are affecting current artists as they create work that will give shape to the next.

These paintings are presented as QuickTime VR panoramas within Macromedia Director movies that incorporate interactivity and sound. The presentation looks at finished "Hyper-3D" paintings and then demonstrates the imaging processes and the authoring techniques used to create 3D panoramas from 2D imagery.

MSA's Attractors: Navigational Aids for Virtual Environments

Agueda Simó
Juan Francisco López
University of Southern California
simo@usc.edu
felix.usc.edu/~simo

Yu Uny Cao
University of California,
Los Angeles

An interesting and difficult challenge in designing a virtual environment is to attract the user's attention to an object or area with rich interactivity. If they are not guided, users could easily miss these special objects and areas. They would not be fully engaged by the virtual world we designed for them. Disoriented and frustrated, they would miss the magic of the virtual world.

We solve this problem in our environment, called Microworlds, Sirens, and Argonauts (MSA). In MSA, we use the mathematical concept of "attractors," which act like the Sirens in Jason's famous Greek voyage. The user, like an Argonaut, is drawn to special locations by the magnetism of the attractor's "song." The attractor leads the Argonauts to a perfect position, where they can view an interesting perspective of an object or reach the specific part of an object we want users to touch or pass through.

Attractors Characteristics
MSA uses spatialized sound objects to allow the user to find attractor regions. When users enter an area of attraction, they are pulled to a specific position called the final position. To achieve this, the path between the user's entry position and the final position is built on the fly. At this point, users lose navigational control until they reach the final position. Because the path is built on the fly, and the entry point of the attractor region varies between flights, there will never be two identical paths to the final position. Every "trip" is a unique experience.

Interocular distance and sensitivity of the 3D mouse are carefully controlled for each virtual object that uses attractors. Both parameters are modified at the final position. This greatly improves navigation. Otherwise, if the sensitivity is not adjusted, the user's movements in the virtual environment may be either too fast or too slow to move around and explore some objects. If the interocular distance is not controlled, it may be so big with respect to small objects that when users get closer to them, they suffer from diplopia. Attractors solve both problems.

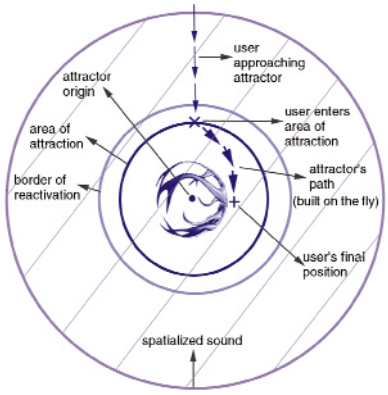
When the user reaches the final position, the attractor is de-activated and will be re-activated when the user passes through a given boundary. In addition, the attractor's magnetic area decreases each time the user visits it until it reaches a minimum size. This avoids the problem of the user being pulled to the same attractor each time.

Attractor Variables
For each attractor we can define the following variables:

- Origin
- Area of attraction and its minimum size
- Border of re-activation
- User's final position
- Final interocular distance
- Final 3D mouse sensitivity
- Spatialized sound area

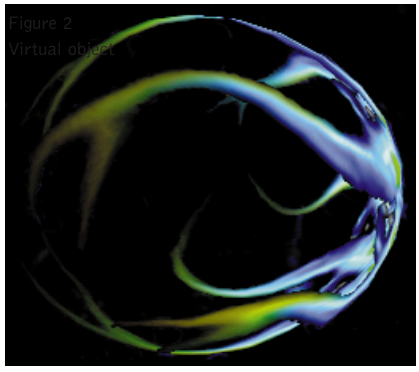
Attractors let Argonauts navigate freely, explore the virtual 3D space, and discover new emotions and thoughts. They facilitate the user's navigation and interaction within the environment. They help users orient in virtual space by providing navigational sound clues. Users can follow the spatialized sounds and make virtual "musical maps."

Acknowledgements
This project has been partially funded by The Labyrinth Project, a three-year research initiative funded by the Annenberg Center for Communication. The authors wish to acknowledge Vibeke Sorensen for her support and inspiration, Iguana Robotics Inc. for its valuable input to this research, FIGD for the donation of SAS (Spatial Audio Server), Robotiker for the use of their computers and support, WorldToolkit and Vrex for their support.



Attractor scheme

Figure 2
Virtual object



Virtual object

Ella Tallyn
Hewlett Packard Laboratories Bristol/
University of West England

Alan Chalmers
Scott Pitkethly
University of Bristol

The Mutable Cursor: Using the Cursor as a Descriptive and Directive Device in Digital Interactive Stories

Background

The game industry is ever striving for a more convincing illusion of reality. In Sony's 2 March 1999 announcement of the new PlayStation they ask you to: "Imagine walking into the screen and experiencing a movie in real-time." However, creating highly realistic environments is only a small part of producing this type of experience.

The problem of real-time, first-person, interactive stories has been the subject of much research. Most productions have felt "more like wandering around a movie set than watching a movie.",The problem is that story telling requires an exactness of timing to guide the reactions of the audience. In interactive situations, timing is often dictated by the actions of the player. What previous research has clearly established is that in order for an effective and meaningful story plot to take place, the player needs some subtle guidance that is either invisible or that takes place in the context of the story.

In addition to the problem of guidance, there is also the problem of the protagonist's character. In most adventure-story computer games, the player usually takes the part of the protagonist, whose character is not richly developed, as it would normally be in other media.

The Technique

The cursor is often no more than a clumsy necessity in adventure games. The graphical representation tends to be both out of context visually and inadequate for conveying story information. So in this technique, the cursor is put to use to both communicate information about a story character and direct the player's attention to appropriate places in the environment. It is hoped that this technique of guidance will both enrich the character and provide a form of guidance that will reduce the time a player has to search for important items, clues or information; the task of searching adds little or nothing to an experience.

Firstly, it was decided that the cursor should no longer be a graphical representation, but blend with the scene. So it becomes a way of changing the appearance of what it alights on. The way the cursor changes an object reflects the character's relationship to it. For example, if an object is especially desirable to a character, it might be seen as brightly lit, perhaps glowing slightly. If it is over a large unimportant object such as a wall, the cursor forms a circular area that is brighter than the surrounding environment, so the cursor is always visible. As the cursor moves onto an object, it gradually mutates to the shape of the object and changes its appearance, and as it is moved away, it returns to a circle of brightness.

Secondly, to direct attention with the cursor and describe the character's relationship to the objects in a scene, it was decided to give the cursor a level of attraction or repulsion to the objects. For example, it slips over some and lingers on others. The level of slippage or stickiness can be defined and attached to an object. This can be altered as a character's situation or attitude changes.

The Scene

This technique is presented in a scene entitled "A Change of Image," in which the player inhabits a character out on an important shopping trip. During this experience, it is hoped that without any other information except the bias of the cursor, the player will come to understand an important aspect of the character's personality, as it is revealed during the course of the trip. The models for this scene were created in 3D Studio Max ,and the code for the cursor was written in Visual C++.

Future Directions

In 1998, a method of placing the player in the first-person point of view of a particular story character was developed.[3] This technique reveals to the player the thoughts in a character's mind as the player examines a scene. This creates game protagonists that have a history and thought processes of their own. The next step will be to combine this with the mutable cursor. A further step will be creation of a scene containing multiple characters that each have their own agendas and inhabit the same space.

References

1. Sony Computer Entertainment Announces the Design of the Next Generation PlayStation System, 2 March 1999.
2. Platt, C. (1995). Interactive entertainment: Who writes it? Who reads it? Who needs it? www.wired.com/wired/archive/3.09/interactive_pr.html
3. Tallyn E., Meech J. Developing the first person point of view: using characters as a sensory lens. Proceedings of SIGGRAPH 98.

Nami

Kelly B. Heaton
Robert D. Poor
Andrew J. Wheeler
Massachusetts Institute of Technology
kelly@media.mit.edu

Nami is a decentralized community of identical "orbs," each of which can display a spectrum of colored light, respond to touch, and wirelessly communicate with its neighbors. When a user activates the spread of color within Nami by touching a single orb, the selected orb responds with expression of a new color. The new state is broadcast to neighboring orbs, prompting them to assume the color and forward the message. In this way, waves of colored light move throughout the distributed network and create visible patterns of behavior. There is no "right" or "wrong" way to arrange Nami, as messages are relayed through whatever pathways are made possible by the placement of the individual orbs.[1] The resulting combination of distributed networking, touch interface, and mutable colors of light provides a novel system for graphical animation in physical objects.

Computer Graphics in Sculpture

Computer-generated animation becomes possible when an artist can exercise dynamic control over a large quantity of individual pixels. Although a single changing pixel is uninteresting, increasing numbers of pixels will manifest recognizable patterns. Certain recursive algorithms can cause pixels to express life-like characteristics known as emergent behaviors, reminiscent of the spread of gossip within a community or the ripple of waves across a pond. Despite the powerful implications of graphic art on the computer, screen-based work is limited to what is expressible with a monitor or projection.

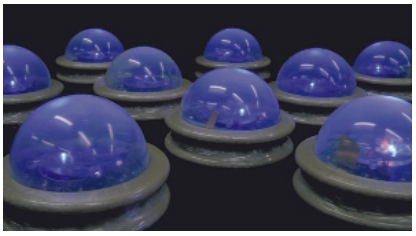
Nami is a preliminary attempt to broaden the scope of computer graphics through integration of color animation into a community of physical pixels, or "phyxels." Phyxels can be imagined as game pieces, building blocks, or other sculptural objects that can express a variety of colors in response to user input and network structure. While the ubiquitous "beige box" format of a computer discourages the union of physical and virtual form, phyxels enable the artist to design both. Like pixels, a phyxel is most interesting when treated as a unit from which an overall structure can be built; but whereas pixels can exhibit only virtual qualities, phyxels have both sculptural form and animated features.

First Implementation of Nami

The introduction of graphical behaviors into a decentralized network of physical objects presented several challenges to the Nami project. Each orb must be able to respond to the user, exhibit the desired behavior, and communicate with other orbs. Capacitive touch-sensing[3] seemed to be the most intuitive interface for the Nami orbs, because it enables the user to control an orb's state by simply cupping the domed top. Each orb has a microcontroller to manage user interactions and communication. A separate microcontroller modulates three LEDs (one each of red, green, and blue) to create different mixtures of colored light. Considerable effort was devoted to creating a gentle and soothing blending of color, so that the orbs would have an organic aesthetic suitable to the wavelike behaviors from whence Nami derives its name. The physical appearance of the orbs is similarly designed to have an undulating form, reflective of the color waves that they carry.

Future Research

In the first iteration of Nami, orbs were designed for wireless communication over a limited distance within a plane. Future research will investigate alternate designs, including connectivity in three dimensions. The current Nami relies on fixed-format messaging for network communication; the next generation will employ "mobile agents" to provide messages that are executable software objects. This architecture will enhance design flexibility and make Nami a useful test bed for visualization of distributed networking algorithms. Several improvements are being made to the infrared communication and touch sensors, and an inductive charging scheme is under development. Forthcoming versions of Nami will also include an electronic artist's palette, which is intended to enhance user control by providing a method for specific color and pattern selection.



A touch event initiates the spread of colored light throughout a community of orbs. (photo by Robert D. Poor)

Nami at the Tokyo ToyFair, March 1999. (photo by Robert D. Poor)

References

1. Robert Poor. Hyphos - A self-organizing wireless network, Master's Thesis, MIT Media Laboratory. 1997, www.media..mit.edu/~r/projects/hyphos/
2. Harold Abelson, Thomas F. Knight, Gerald Jay Sussman. Amorphous computing, 1996, www-swiss.ai.mit.edu/~switz/amorphous/index.html
3. Rehmi Post. Personal communications.
4. Nelson Minar, Kwindla Hultman Kramer, Pattie Maes. Cooperating mobile agents for dynamic network routing, Chapter 12, Software Agents for Future Communications Systems, Springer-Verlag, 1999, ISBN 3-540-65578-6.

Wells Packard
Nicholson NY
wells@nny.com

Nicholson NY/Pequot Interactives

In 1993, Nicholson NY was selected by the Mashantucket Pequot Tribal Nation to design and produce a series of interactive programs for their museum's permanent exhibits. There are six programs, accessible at 23 stations, on topics ranging from the geological history and natural resources of Southern New England to the social and political history of the Mashantucket Pequots. Touchscreens allow users to easily navigate an enormous amount of information in the form of 3D animations, documentary footage, traditional media, and even hand-painted cell animations, presented in broadcast-quality digital video. The interactives are accurate down to the native tree bark depicted in the 3D models, based on archeological data and US Geological Survey maps – realism that might be lost on some users, but not to the scholars who use the museum for research.

The Pequots, driven to the edge of extinction, were without a pictorial reference, so all designs had to be approved by historians as well as the tribe. Each of the programs contains from one to 2.5 hours of content, but they are also designed to provide casual users with informative nuggets of information that will enhance their understanding of the exhibit.

"Explore the Fort" uses 3D modeling to recreate a 17th-century fort discovered on the reservation in 1991. Users follow pre-rendered animation paths to navigate through the fort. Artifacts found at the site can be found in corresponding virtual locations in the model. Analysis and interpretation is provided in documentary video segments.

"Algonquin Languages"
No native speakers of Pequot survive, so this interactive was built around six stories told by native speakers in the surviving languages closest to Pequot. Each story is accompanied by animation and is subtitled in both English and its corresponding native language. The program also includes a vocabulary section, where users can compare cross-referenced words and phrases in four languages.

Passion Spaces Based on the Synesthesia Phenomenon

Tsutomu Miyasato
ATR Media Integration & Communications Research Laboratories
miyasato@mic.atr.co.jp

Human recognition and human communications are classified as being either logical or emotional. Passion spaces here are compared with conceptual spaces in the analogy of logos vs. pathos.

Synesthesia Phenomenon

The painter Wassily Kandinsky was said to possess synesthesia. He said that when he viewed some colors, the colors would also become audible. Moreover, the reverse was true. He could see some sounds as colors. In addition, the French poet Arthur Rimbaud created a poem relating images to colors.

Synesthesia (a fusion of stimuli) involves reactions to a combination of senses. One example of this is the phenomenon of hearing a sound and feeling a color (colored hearing), which is a fusion of sound stimuli and visual stimuli. Yellow is usually assigned to the sound of a child's high-toned voice.

Generation of Passion Spaces by Media Exchange

What kind of world do people with synesthesia perceive? We investigated the generation of spaces by imagining their world. Since relationships are thought to be strong among colors, sounds, and feelings, we tested color image creation from the Japanese poems of Ogura Hyakunin Isshu, whose poems are taught in Japanese schools to encourage understanding of emotions or passions, both from the overt contents and between-the-lines meanings. We then focused on an anthology of 100 poems by 100 different poets. However, this study was not about the correspondence of simple sounds and colors, and it is not like the flashback of concrete scenes imagined from the meanings of the words in the poems. Our goal is to generate spaces to reflect the passions we experience.

We generated passion spaces from Japanese poems in two steps:

1. A table was created of the corresponding relationships between phonemes and colors. For example, a vowel was made to correspond to red, another vowel was made to correspond to black, etc. In addition, correspondences were established between differences in brightness for differences in the sounds of voiced consonants. Here, the relationships between phonemes and colors were determined by the senses of the authors themselves.

2. The phonemes that compose 31-syllable Japanese poems were made to correspond to image elements, and a multidimensional image element array was constructed. We used a 2D array so that the image became rectangular and formed a square matrix. The image elements corresponding to the phonemes according to the time series of the reading were colored from the upper left to the right in the 2D array. After that, various types of image processing (low-pass filtering, etc.) were carried out and displayed.

This same method of connecting words to emotional spaces can be applied to music. For example, if a quarter note is the shortest musical note in a musical score, it can be made to correspond to image elements as the shortest part of the score. Color images can easily be made to correspond to musical notes, and even people who cannot read complex musical scores are able to feel the similarities. In this way, it is possible to instantly judge similarities between pieces of music without the need for a performance.

九（#9）
小野小町
花の色はうつりにけりないたづらにわが身世にふるながめせしまに

六一（#61）
伊勢大輔
いにしへの奈良の都の八重桜今日九重に匂ひぬるかな

No. 9 poem by Lady Ono no Komachi

Color of the flower
Has already faded away,
While in idle thoughts
My life passes vainly by,
As I watch the long rains fall.

No. 61 poem by Lady Ise no Osuke

Eight-fold cherry flowers
That at Nara—ancient seat
Of our state—have bloomed,
In our nine-fold palace court
Shed their sweet perfume today.

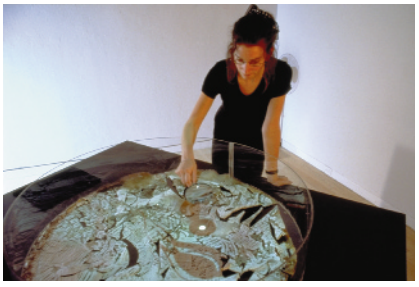
* English translations used by permission of the University of Virginia Library's Japanese Text Initiative: etext.lib.virginia.edu/japanese/hyakunin

Figure 1. Example of color image creation from poems of Ogura Hyakunin Isshu.

Tiffany Holmes
University of Michigan
tgholmes@umich.edu

Phene-: Creating a Digital Chimera

My research and creative practice explore the intersection among digital, biomedical, and linguistic modes of bodily representation. In my most recent installations, I place my viewers in situations that force them to take on unfamiliar roles to gain new perspectives on the act "performed." In Phene-, I invite the viewer to impersonate a scientist and use tools associated with the laboratory to interact with the art.



In Phene-, the metaphor of the laboratory "gone amuck" pervades the space. Approaching the installation, the viewer hears garbled sounds coming from a microscope. Signs direct the viewer to don gloves and inspect a group of slides. In attempting to make sense of the samples, the viewer turns the knob to focus the microscope. In this process of twisting, the viewer focuses not only the image but also the sound. The slide is "named" by the computer voice as it comes into view.

Behind the microscope, a wall displays items that allude to the process of capturing, cataloguing, and identifying specimens: dirty scalpels, unwashed flasks, dissection trays, etc. Notes and drawings pinned to the wall portray a bizarre array of hybrid organisms. Also, many of the containers reveal surprising contents. For example, several petri dishes appear to contain routine cultures. However, on closer inspection, colonies of bacteria reveal "intelligent" behavior as they mass into simple word forms such as "CAT."

The focal point of the installation lies behind the exterior wall. A sign directs viewers to examine the "chimera" on the opposite side. Rounding the corner and entering a darkened space, the viewer immediately senses moisture and a dank odor. Here lies a large petri dish illuminated by the light from a projector mounted in the ceiling. Moving closer, the viewer notes that the specimen is composed of a rapidly changing animation layered atop a mass of biotic material. The podium that houses the dish is littered with all sorts of tools: eyedroppers, misters, magnifying glasses, etc. A fan and a shelf with two vaporizer units occupy the far corners of the space.

Viewers interact with the chimerical specimen via processes of feeding, watering, magnifying, etc. In crouching to pick up the magnifying glass, the interactor trips a switch that creates sound. The auditory information consists of amplified breathing noises. As the interactor positions the lens atop the specimen, small dots of light trigger an array of animated creatures to immediately materialize beneath the lens. "Watering" the chimerical specimen alters the look of the dish as well: a light misting of the surface causes the digital organisms to multiply so that the entire space darkens as a result.

Phene- lures the visitor into a fecund space resplendent with the mysteries of the cycling of life. Respiration is a key feature of "aliveness." Thus, the installation uses the literal sounds of human breathing as well as mechanical "breath" in the form of the random turning on and off of the fan. The microbial growth in the dish itself was originally cultured from the artist's breath. In Phene-, the biological process of procreation is juxtaposed with the process of artistic creation. The dish represents the heart of the installation, a dark, womb-like arena for the growth of both virtual and microbial life forms.

SaltoArte: Explorations in Spatial Interactive Multimedia

George Liaropoulos-Legendre
Larry Burks
S. Kirsten Gay
Harvard University
lburks@gsd.harvard.edu

SaltoArte is the fruit of an ongoing collaboration between the Harvard Design School and the Busch-Reisinger Museum. It was part of the museum's exhibition titled "Multiple Configurations, Presenting the Contemporary Portfolio", 13 May - 1 August 1999. This exhibition focused on four possible ways to display a contemporary art portfolio in its entirety within the constraints of the museum. Each display necessitated a different type of engagement on the part of the visitor.

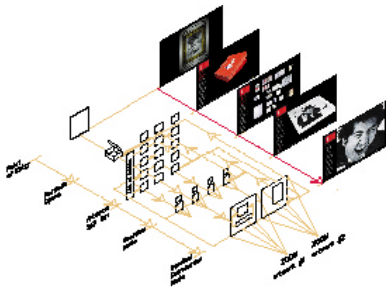
As part of this effort, the authors collaborated with the curator to develop an interactive display of the portfolio SaltoArte, a multiple edition of visual works by various European and American artists affiliated with Fluxus, a conceptual art movement of the 1960s.

The objective of this collaboration is to enable visitors to virtually handle art, thereby avoiding any harm to the objects. Visitors can select any number of objects, view them individually with details, as well as turn pages, unfold posters, and open envelopes. Using technology such as Java and VRML, we hope to empower museum audiences to activate Fluxus works as their makers' intended.

The concept of the project is based on two related notions:

1. We want to extend the tangible qualities of the artifacts as they exist in time and space. Whereas the traditional language of 2D multimedia helps us convey information about the artifacts, it typically fails to convey a sense of their physical presence and material qualities. To address this issue, we are building a seamless environment where the descriptive and informative aspects at which 2D multimedia environments excel are augmented by the kind of spatial experience afforded by virtual environments.

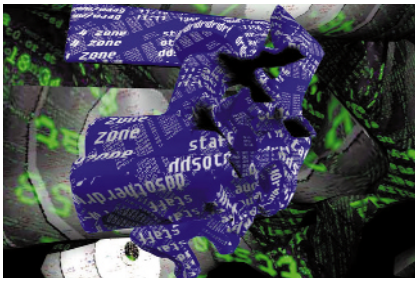
2. We need to communicate a variety of visual and auditory information that documents the artifacts in their historic and artistic context. This involves the use of textual and aural descriptions linked to related images and references. To seamlessly merge the textual, figurative, and spatial dimensions of the project, we propose a strategy that blurs the distinction between 2D and 3D by intertwining conventions taken from both realms. The artifacts are scattered in 3D space, but they are also laid out as a seemingly 2D image array. They can be accessed either by means of hypertext media conventions (by passing the cursor over keyword links) or by direct manipulation (by clicking on the artifacts themselves).

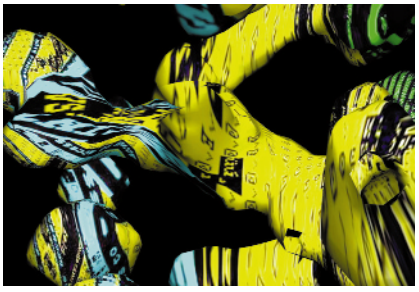


Subtle manipulations of lighting and viewpoint position are then used to reveal the 3D nature of the scene. 3D conventions include the ability to directly manipulate the orientation of the object and/or the distance between eye and artifact. Maintaining a level of fluidity between conventions involves development of fine-grained control systems, which enable museum visitors to handle the artifacts easily and intuitively with a virtual pointer and without extraneous visual control structures, such as dialog boxes and buttons.

Margarete Jahrmann
Max Moswitzer
Konsum.Arts
superfem@konsum.net
max@konsum.net

Setup of the Konsum Art.Server





The Konsum Art.Server is a conceptual Web3D artwork. It evaluates the aesthetics of different operating systems as cultural code and syntax. Certain visual and acoustic codes have already trickled into the cultural consciousness, where they have become socially accepted. They can be read and decoded. These systems include, for example, the codes of Joystick Nations, video games, and computer-game culture that we learned as kids in the arcades.

This online world is also a cyber-psychological tool. Different machine aesthetics offer individual users the possibility to identify themselves as (or with) a certain operating system or brand appearance, for example as a blue Unix shell. So the postmodern identity, defracted, fluid with an unsecure kernel, can be brought back to certain kernels and associated with these kernels of the machine code at the moment of login. This is not an attempt to restore the identity as one proofable unit but to give it an appearance for the moment in a playful environment.

The Konsum Art.Server is a browser experiment to connect tech and arts communities. Users can consider abstract VRML objects as their personal, non-humanoid VRML online avatars. The display always shows an actual freeze frame of your login shell: simple, but symbolically effective. These objects really display information in the 3D Web – information that is not displayed under ordinary conditions, but is in the net beyond the Web. Trace route commands, ns-lookups, and scripts imbedded in the VRML can be extended, but basically all commands that you also could make as user of a Linux system can be made available. This may be the linking point between the arts community and the emerging community of X3D, and between developer communities.

Web3D is providing tools for representing 3D data online. In relation to the 3D environment, the question of dynamic self representation becomes urgent. Aesthetic avatars based on concrete online information are another option. By displaying multiple identities (aka a variety of "nixes shells"), from IRIX, Sun OS, Linux, Free/Net BSDees, etc. directly into the 3D environment as aesthetic experiences, the underlying layer of the net is brought back into Web3D, and at the same time a cultural syntax is expressed through the surface of a specific login shell.

SUperFEM Datavatars - Datasituationism as Interface
In the SUperFEM performances, VRML Datavatars create situations as a social metadesign to enable users to experience conditions and netparameters in addition to the use of the machine prosthesis.

The structural aspect of the Internet is displayed. All navigation through objects is generated in real-time. By entering 3D objects, different textures of your actual movements on the net (time spent online), or log files of your chosen destinations, are displayed. Time is also displayed by sound samples that express the volume and speed/slowness of data flow.

Virtual Music Reproduction

Hartmut Chodura
Fraunhofer-IGD
chodura@igd.fhg.de

Arnold Kaup
Fachhochschule

Stereo sound has been the state of the art for musical recordings for more than 30 years. Consumer media like tapes, records, and CDs attempt sound fidelity by using two channels for audio playback. However, the illusion of "being there" is limited via only two channels.

This proposed application takes advantage of 3D graphical and acoustical output. It follows the idea of a new interactive, immersive, and semi-artificial way of experiencing music. All parameters, like location and orientation of the listener and sources as well as the visual and acoustical appearance of the virtual environment, can be modified using a 3D application (see Figure 1).

Required data files like scene descriptions, images, or live video of the musicians, as well as their audio tracks, are stored on a CD-ROM or a DVD. The prototype implementation consists of graphical and acoustical subsystems. The network SAS (Spatial Audio Server) has been developed for acoustical rendering of multiple audio sources in 3D environments. Thus, events can be invoked, like changing the acoustical environment (room acoustics), playing back sources, or updating their positions (see Figure 2).

In order to place sound sources in virtual environments, it is necessary to record non-reverberant signals of the sound material. Another important requirement is the acoustical separation of the different sources. Pop and rock music is usually produced by creating multitrack recordings. For acquisition of music material for a classical interactive production, a different approach has to be followed: In the prototype "VirtueString," the four instruments of a string quartet (Haydn, Op. 20 No. 5) were recorded individually in an anechoic chamber to get total separation without any room acoustics. In order to get synchronized material, first a stereo prerecording of the whole quartet was produced in the conventional way. This was later used as a pilot track for the single-instrument recordings in the anechoic chamber.

For the consumer market, a CD or DVD implementation of the software provides low-cost access to a virtual interactive music environment. Another marketing option is a variable "music-minus-one" playback system. "Minus-one" means half-playback recording of a certain music piece where one voice is muted. This allows musicians or music students to play with a complete virtual symphony orchestra.

The application is currently enhanced by an audio-visual environment with three different rooms of various sizes, geometry, and acoustical properties. Further investigations focus on how to achieve an adequate spatial mapping between graphics and sound under various geometric and technical constraints.

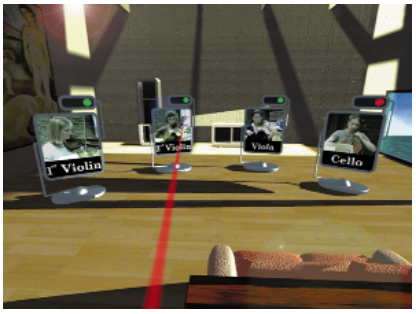


Figure 1
The audio-visual presentation is rendered in real-time.

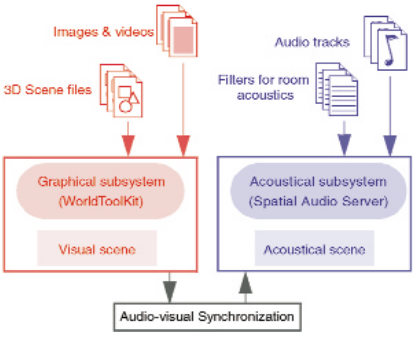
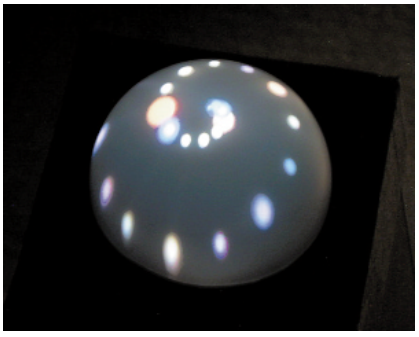


Figure 2
Graphics and sound are processed by two linked subsytems.

Karrie Karahalios
Massachusetts Institute of Technology
kkarahal@media.mit.edu

VisiPhone

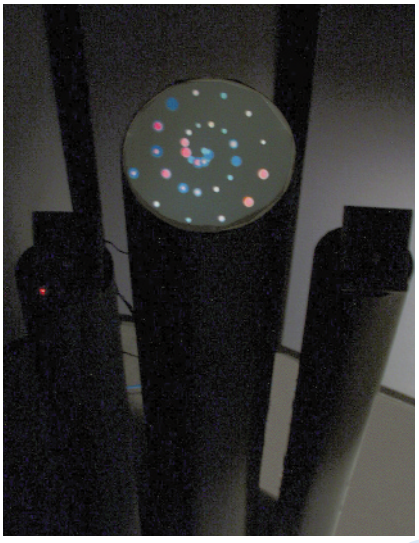
VisiPhone is a communication object that opens a graphical as well as an audio portal between two spaces. It is designed to provide a continuous, ubiquitous connection between these two different locations. The goal is to create an aesthetically intriguing object that enables users to perceive conversational patterns that are present but not obvious in traditional communication interfaces. Through this experimental medium, we are exploring the social and aesthetic aspects of sound visualization.



Function

The VisiPhone project began with the notion of building a virtual portal between two domestic spaces, one that would allow the inhabitants of two separate homes to communicate easily and to be intuitively aware of each other's presence. VisiPhone's graphics express the dynamics of the conversations originating at both locations by providing visual feedback that one's voice has carried sufficiently and indicating the presence of those on the other end. It portrays the existence of the connection even in moments of silence, thus removing the surveillance-like aura of the audio-only system.

Today, most communication devices such as the telephone or the computer are fairly utilitarian in design. Yet as networked communication and permanent connectivity become the norm, we believe that their capabilities will spread to a variety of interfaces and become integrated into a variety of objects and environments. VisiPhone is an exploration of the aesthetic and social possibilities we can achieve by melding computer graphics and communications.



Form

For a piece such as VisiPhone, form is function. It must be attractive and intriguing enough to claim a central place in a kitchen or a living room.

We are currently experimenting with two versions of the display unit: a dome and a podium pedestal. The dome shape breaks away from the flatness of current digital displays; its placement on the pedestal puts it into the category of sculptural object (and conceals the projector). The dome version is a smaller, more intimate interface whereas the podium is a larger unit meant to be used by a group of people. On both displays, the graphics are designed to convey a sense of rhythm and activity.

The abstract graphics rendered on the display surface portray the audio originating at each end of the connection. The source of the audio is revealed through the use of color (blue tones for incoming audio and orange tones for outgoing audio). Volume is mapped onto the size and color of the graphics: the louder the volume the bigger and more saturated the circle.

Whenever speakers at different ends of the connection talk at the same time, the graphics for both audio signals are rendered as overlapping colored circles. The different colors that represent each speaker blend, portraying the mixing of the audio signal.

VisiPhone displays both the real-time audio and a brief history of the conversation. The real-time rendering is shown in the center of the display and then spirals out toward the edge of the display. The circles fade as they approach the end of the spiral, enhancing the notion of evolving time.

Technical Description

A two-way VisiPhone interaction requires two VisiPhone interfaces. Although the dome and the podium differ in their physical appearances, they share the same computing and projecting hardware. Each VisiPhone consists of a set of speakers with embedded microphone, a PC with full-duplex audio card and GL compatible video card, a scan-converter, a projector, and a projection display.

Audio input comes from microphones; the input signal is transformed into a video and audio composite. The output is a graphical display and a set of attached speakers. The video from each corresponding PC is rear-projected onto the display surface, thereby transforming the VisiPhone apparatus into a dynamic interactive object.

VisiPhone

3D Facial Reconstruction and Visualization of Ancient Egyptian Mummies Using Spiral CT Data

Maurizio Forte
Consiglio Nazionale delle Ricerche
forte@nserv.icmat.mlib.cnr.it

Giuseppe Attardi
Marilina Betru
Silvano Imboden
Francesco Mallegni
Università di Pisa

Roberto Gori
Antonella Guidazzoli
CINECA-VISIT

The problem of rebuilding a face from human remains has been, until now, especially relevant in the ambit of forensic sciences, where it is obviously oriented toward identification of otherwise unrecognizable corpses, but its potential interest to archaeologists and anthropologists is not negligible. We present here the preliminary results of joint research by the Università di Pisa, the Visualization Laboratory of CINECA, and the CNR-ITABC (Institute of Technologies Applied to Cultural Heritage, National Research Council, Rome) whose aim is reconstructing, through spiral computed tomography data and virtual modeling techniques (in our case with VTK software), 3D models of the possible physiognomy of ancient Egyptian mummies. This work is carried out through a multidisciplinary approach, involving several specialties: image processing, anthropology, Egyptology, and computing archaeology.

Main project tasks are:

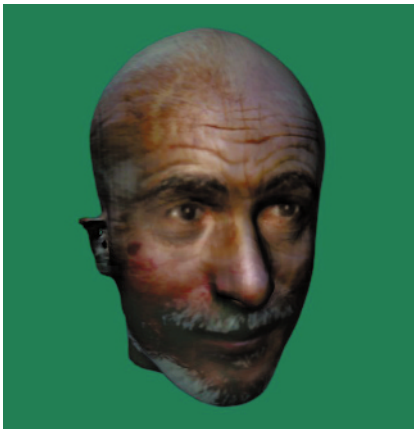
1. Anthropological and Egyptological analysis of the head.

2. Spiral CT of the head.

3. Reconstruction of a 3D model of the skull generated from CT data processing.

4. Reconstruction of soft tissues.

5. Application of textures fitting the somatic features.



3D facial reconstruction and visualization of ancient Egyptian mummies

Visualization

Suba Varadarajan
Xiaoning Fu
Rick Parent
Patrick J. Flynn
The Ohio State University
www.cis.ohio-state.edu/~parent/gait-project

Kathy Johnson
University of Texas, Houston

3D Gait Reconstruction Using Two-Camera Markerless Video

Leg contours and axis, skeleton, and NURB surface

Gait analysis is a valuable tool in evaluation of neuromuscular disorders, athletic injuries, and prosthetic joint replacements. Our goal is to reduce deployment of technology in acquisition of video used for gait analysis without sacrificing accuracy. We envision the following application context for this research: A patient with an abnormal gait has his or her leg measured and a typical gait sequence digitized at a central clinic. The patient then undergoes surgical correction and begins post-operative therapy with a local physician. After some time, that practitioner sends a video of the patient's walk to the central clinic for processing. Before-and-after sequences may reveal improvement in the gait and help determine further treatment.

Unsynchronized markerless color video sequences were taken at the OSU Gait Lab from the front and side of a person walking, and divided into frames. The subject wore form-fitting leg tights of different colors.

Leg Silhouette Contour Extraction
The side view images were deinterlaced and then expanded, yielding two field images per frame image. Edge detection was performed on a gray-scale version of each image using a combined Sobel operator. Strong edge pixels on each scan line of the images were used as starting points for a directed search for leg segments in the color images. Segment extraction used a simple color predicate, since different colored leg coverings were worn. Pixels satisfying the color predicate for each leg were grouped into a run, and the endpoints of the run were labeled as silhouette points. The resulting well-connected set of points was smoothed using a B-spline fit (see Figure). Points on the leg axis were assumed to lie on the midpoints of the scan lines belonging to the segments.

Hidden Surface Reconstruction
In the side view, parts of one leg that were occluded by the other leg were recovered in two ways. If, on a given scan line, one of the contours of the leg was found, the other contour point was estimated from a template that contained the width of the legs in any scan line. If both contours for a leg were missing, the positions of the missing contour points were estimated using the closest known contour points above and below them.

Knee and Ankle Location
Each leg axis in the side views was searched for knee and ankle "points," which divided the axis into three contiguous parts such that the independent linear fits applied to each part produced minimum squared error. Knee positions with larger angles between the upper-leg and lower-leg segments were used to predict the knee positions in less-reliable frames. Knee and ankle positions in front views were estimated using the corresponding side view measurements.

3D Skeleton and NURB Surface Reconstruction
Using the camera configuration geometry, the 3D positions of the knee, ankle, and leg-top were obtained as the intersection, or point of nearest approach, of the projection rays defined by the camera centers and the estimated anatomical position in the front and side views. The resulting pair of 3D line axes for each leg was used to reconstruct skeleton and NURB surfaces (see Figure).

Conclusion
The knee angles in each frame of the reconstructed motion, and in the actual 3D motion obtained from a six-view simultaneous synchronized marker video were compared. The reconstructed motion followed the actual 3D motion over time but produced a much reduced bend magnitude. Future work will include improving the accuracy and robustness of leg contour extraction and using anatomical data to determine the leg axes.

Acknowledgments
The OSU Interdisciplinary Seed Grant Program provided initial funding. Thanks to the OSU Gait Lab for their facilities and personnel, and the OSU Department of Computer and Information Science for infrastructure support.

References
1. D. DeCarlo et al, Deformable Model-Based Shape and Motion Analysis from images Using Motion Residual Error, Proceedings ICCV98, pp. 113-119, 1998.
2. S. Kim et al, Two-dimensional Analysis and Prediction of Human Knee Joint, Biomedical Sciences Instrumentation, 29, pp. 33-46, 1993.

3D Imaging for Rapid Response on Remote Sites

J.A. Beraldin
F. Blais
L. Cournoyer
M. Rioux
S.H. El-Hakim
National Research Council Canada
angelo.beraldin@iit.nrc.ca

Non-contact measurement techniques like those based on structured light have found widespread use in industrial metrology and reverse engineering. Laser-based technologies attempt to structure the environment, through artificial light projection. The results are dense range maps extracted from visible surfaces that are rather featureless to the naked eye.

In the field of heritage recording, onsite and complete 3D documentation is necessary in situations where objects and/or environments can't be moved or their access is restricted because of natural causes (for example, earthquake or tourism-related degradation). In this situation, 3D acquisition must be performed in a rapid-response timeframe, from several hours to maybe a day. In addition, many applications do not allow any alterations to the object and surroundings to suit the vision system (for example, by placing markings or changing the reflectivity of the surface with powder or abrasion).

We have put together a self-contained 3D imaging system that is capable of satisfying many demanding onsite 3D documentation tasks. Special attention is placed on accuracy, as it is critical for obtaining high-quality reconstruction of 3D models from range imagery.

Compact 3D Range Camera
The range camera (Biris, Figure 1) was developed to work in difficult environments where reliability, robustness, and ease of maintenance are as important as accuracy (see URLs). The main components of Biris are a solid-state laser diode, a CCD or CMOS camera, a camera lens, and a mask with two small apertures. The mask replaces the iris of the camera lens (Bi-iris). This imaging technique, when combined with advanced signal processing algorithms, allows Biris to become very tolerant to ambient light illumination (sunlight, for example). The Biris camera is mounted on a pan unit and tripod, and is connected to a PCI card installed in a portable PC. Reconstruction of a 3D object is achieved by acquiring a sufficient number of range images to cover the surface of that object. The views must have enough overlap between them to find their relative spatial position and orientation.

Results from Remote Sites in Italy, 1997-1998
Figures 1 and 2 show a number of 3D models acquired in Italy over a two-year period. The models include a marble sculpture of the Madonna col Bambino by G. Pisano, a bronze bas-relief from Donatello, and some architectural elements from the old Abbey of Pomposa. The range images for these projects were acquired in three separate one-day sessions. The models were generated using Polyworks on a PC. A number of sites have been documented using either Biris or National Research Council Canada's other 3D technologies. The sites, in Israel, Florence, NASA facilities, Boston, and Canada, are described at these URLs:

www.vit.iit.nrc.ca/blais/bi_home.html
www.vit.iit.nrc.ca/beraldin/Web_italia/Biris_in_ITALY.html
www.vit.iit.nrc.ca/Pages_Html/English/Research.html
www.vit.iit.nrc.ca/Pages_Html/English/Applications.html
www.innovmetric.com/

Figure 1
Top and center: portable 3D imaging system includes a Biris camera (standoff: 30 cm, range: 200 cm, data rate: 15,360 3D samples per sec, weight: 900 gr.), a pan unit and a tripod. Bottom: outdoor application at the Abbey of Pomposa (circa 850) in Italy 1998.

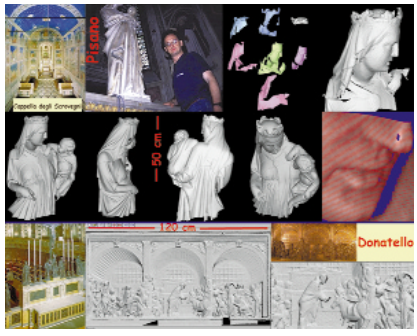


Figure 2
Top and center: Madonna col Bambino, G. Pisano (circa 1305) showing the seven distinct color-coded images used for the face and model of the top portion using 62 views for a total of 1,340,000 polygons. Bottom: bronze bas-relief of Donatello (circa 1446-1450), model with about 580,000 polygons, Italy 1997.

Suguru Saito
Masayuki Nakajima
Tokyo Institute of Technology
suguru@cs.titech.ac.jp

3D Physics-Based Brush Model for Painting

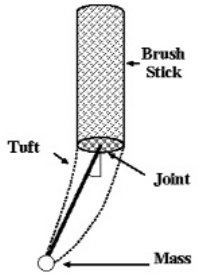


Figure 1 (a)

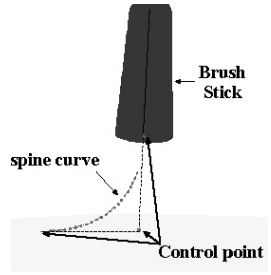


Figure 1 (b)

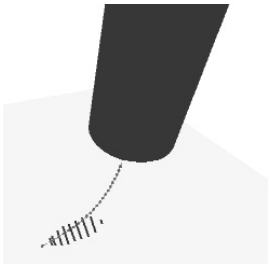


Figure 1 (c)

Figure 2 (a)

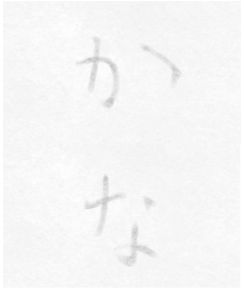


Figure 2 (b)

Figure 2 (c)

Though a great deal of free and commercial painting software already exists, we feel that the current software does not yet allow users to draw free and varied brush strokes. In the real world, a painter can express various strokes without "mode changes," using only one brush.

One major problem is that brushes in current major painting tools are based on the trace of a disc,[1] which limits the painter's ability to control the stroke shape. Other approaches for producing more natural-looking strokes exist. However, techniques that widen the center curve[2,3] have limitations similar to those mentioned above for popular paint programs. Morphing with a painted texture can express rich strokes, but the method is not intuitively controllable.[4]

Modeling

As an artist paints, the hair of the paint brush bends and changes shape, and so does its footprint. These changes allow various rich and expressive strokes. To represent natural strokes on a computer, we designed a 3D brush hair model.

The physical model is shown in Figure 1(a). When the position and posture of the brush handle are given, the joint angle is derived by calculating the minimum energy of the tuft model. This energy is a summation of the bend energy of the joint, the potential and kinetic energy of the tuft mass, and the frictional energy between the tuft mass and the paper.

The curve of the tuft spine is defined as a Bézier curve with three control points in Figure 1(b). Because the locations of the control points are determined from direct input of the brush stick posture and simulation results, the shape of the spine curve is reasonably natural and can be controlled easily.

Finally, the tuft itself is defined as a set of discs whose centers are on the spine curve. The footprint of the tuft is calculated by detection of the intersection of the discs and the xy plane in Figure 1(c). The drip shape of the footprint changes size and also bends according to stick motion. In order to apply paint to paper, the model also contains a simple liquid movement model.

Painting Results

The images in Figure 2 were painted with our brush model. A pen-type input device was used for stroke input. The paint model used is based on Kubelka-Munk theory.[5] Rendering is completed in real-time on a 450MHz Pentium-based Linux system and is fully interactive.

Figure 2(a) consists of only four strokes. However, each stroke is very expressive, and together they succeed in creating a meaningful artistic picture. In Figure 2(b), the sharp curve at the bottom right of the second character shows off the features of our brush model well. When the motion of the brush handle curves, the hair tip traces a different route than the handle. This effect is quite difficult to create with general disc tracing methods. Figure 2(c) is an oriental style flower painting. It shows various strokes and novel expression as a computer graphics image.

Conclusion

The 3D physics-based brush model widens the possible expression of stroke inputs. Because the hair tuft of the brush changes dynamically, stroke starting point, ending point, and curves are very expressive. Most importantly, the brush allows natural input in a manner very similar to real painting with a pen-type input device.

References

1. Painter, www.metacreations.com/products
2. Steve Strassmann. Hairy Brushes, Computer Graphics, 20(4), pp 225-232. 1986.
3. Oinglian Guo. Generating Realistic Calligraphy Words, IEICE, E78A(11), pp 1556-1558, November 1996.
4. Siu Chi Hsu and Irene H.H.Lee. Drawing and Animation Using Skeletal Strokes, SIGGRAPH 94 Proceedings, pp 109-118, 1994.
5. Von Paul Kubelka and Franz Munka, Ein Beitrag Zur Optik der Farbanstriche, Zeitschrift Fur Technische Physik, pp 593-601. 1931.

3D Rendering Effects for 2D Animation

Lance Williams
Galen Gornowicz
DreamWorks SKG
lwilliams@dreamworks.com

Absence of shading and texture is a problem of traditionally animated film. It inspired soul-searching and strenuous effort well before shaded, textured 3D computer animation arose. An interesting approach to this problem is to use the 2D shapes drawn by the animator to define 3D surfaces, which can be rendered by conventional 3D graphics algorithms. This method was outlined in a previous paper,[1] which describes an "inflation" algorithm for creating smooth 3D surfaces from 2D shapes.

These "inflations" are a necessary component of the automatic airbrushing, specular ornaments, and gobo shadow effects to be described. It is worth noting, though outside the scope of this presentation, that such surfaces arise in many contexts. Implicitization of a 2D matte can be performed using the signed distance function, but smoother implicitizations are possible in 2D and 3D.[2] Interiors of such functions are smooth "inflations" of the shapes. These surfaces, derived from drawings, are very useful for achieving a wide range of continuous-tone animated effects. They are not, however, accurate models of the characters the drawings depict.

To achieve the fidelity to the props and characters required for such effects as texture mapping, accurate 3D models are necessary. Previous efforts in this direction[3] require animators both to draw a sequence and animate a corresponding sequence in 3D. By tracking drawings and conforming models to them, we eliminate the 3D animation step. Instead, the 3D animation defined by the model and the drawings can be further edited in 3D after the tracking is performed.

Specular reflections on the ornaments are a reflection mapping of the torch flame (from "Prince of Egypt").

3D dress model automatically tracks and conforms to the drawing of the dress, permitting texture-mapping.

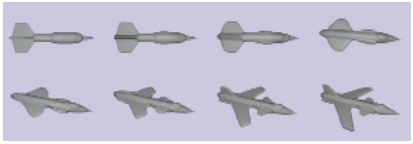
References

1. Lance Williams. Shading in Two Dimensions, Graphics Interface '91, pp.143-151, Morgan-Kaufman Publishers, June 3-7, 1991.
2. Luiz Velho and Jonas Gomes. Approximate Conversion of Parametric to Implicit Surfaces, Computer Graphics Forum, vol. 15, #5, pp. 327-388, Elsevier Science Publishers, 1996.
3. W.T. Correa, R.J. Jensen, C.E. Thayer, and A. Finkelstein. Texture Mapping for Cel Animation, Computer Graphics Proceedings, Annual Conference Series, 1998, ACM SIGGRAPH, pp.435-446.

David E. Breen
California Institute of Technology

Ross T. Whitaker
University of Tennessee

A Level-Set Approach for the Metamorphosis of Solid Models



We present a new approach to 3D model morphing that employs an active deformable surface that starts at the source shape and smoothly changes into the target shape. We represent the deformable surface as a level set (iso-surface) of a discretely sampled scalar function of three dimensions.

The sampling produces a regularly spaced rectilinear volume dataset. Such "level-set" models have been shown to mimic conventional parametric deformable surface models by encoding surface movements as changes in the grayscale values of the volume. A well-founded mathematical structure leads to a set of procedures that describe how voxel values can be manipulated to create deformations in the level sets. The result is a voxel-based modeling technology that offers several advantages over previous methods, including support for a wide range of user input, no need for reparameterization, flexible topology changes, and results with sub-voxel accuracy.

As with previous volumetric model morphing algorithms, our approach consists of two central steps: a "warping" step that utilizes a user-defined coordinate transformation that deforms the source model into approximately the same shape as the target model, followed by a "blending" step that completes the morph by automatically moving the level-set model towards the target model's surface. Since the warping stage of volumetric morphing has been satisfactorily studied by others, our work primarily focuses on developing a superior method for the blending stage, one which does not require significant user input in order to produce an acceptable morphing result. Our level-set approach will produce a reasonable morph with no user input. Users may then apply an increasing amount of input until the desired morphing sequence is achieved.

In general, an animator controls the morph by defining how the models overlap. The level-set model only moves in those areas where the target and source surfaces are separated from each other. The portion of the level-set model that is outside the destination model will shrink, while the portion inside the destination model will expand to become the final shape. As long as the models initially overlap, the source model is guaranteed to smoothly deform into the target model.

The level-set approach to 3D model morphing provides several advantages over other 3D morphing algorithms. Level-set models are active; their underlying motion is defined algorithmically and not defined interactively. The amount of user input required to produce a morph is directly proportional to the amount of control the animator wishes to impose on the process. The animator may allow the system to automatically generate the morph or employ a full 3D warping to describe the morph, with the level sets simply providing a small fine-tuning of the surface. Any intermediate amount of user input will produce a reasonable morph, as long as the two models overlap. Since level-set models do not rely on any kind of parameterization, they do not suffer the problems of parametric surfaces (for example, limits on their shape and the need for reparameterization after large shape or topological changes). This lack of parameterization, along with no direct representation of topological structure, allow level-set models to easily change topology while morphing.

The model can "split" into pieces to form multiple objects. Conversely, several disjoint objects may come together to make a single object. Finally, both the scan conversion and deformation stages of our morphing approach produce models with user-defined sub-voxel accuracy. This allows the user to specify the time/quality tradeoff, and generate superior results as compared to previous work.

An Interface for Transcribing American Sign Language

Rosalee Wolfe
DePaul University
wolfe@cs.depaul.edu

Karen Alkoby
Jamita Barnett
Pattaraporn Chomwong
Jacob Furst
Gary Honda
Glenn Lancaster
Frank "Tony" Lavoie
Steven Lytinen
John McDonald
Lopa Roychoudhuri
Carolyn Taylor
Noriko Tomuro
Jorge Toro

Deafness is not only a barrier of sound but also a barrier of language. American Sign Language (ASL) is a natural language used by members of the North American deaf community and is the third or fourth most widely used language in the United States. While ASL shares vocabulary with English, there is no simple word-for-word translation. Because of the differences between the two languages, most native ASL signers read English at the third- or fourth-grade level. This is why closed captioning on television is a good first effort at making spoken English more accessible to the deaf population but does not represent a completely satisfactory solution.

At present, deaf people rely on sign language interpreters for access to English, which is an awkward solution at best. A digital sign language interpreter, which translates spoken English into ASL, would help bridge the gulf between deaf and hearing worlds. We are currently developing a database of ASL that will be used as the lexical database for machine translation. It includes position, orientation, and shape of the hands as well as motions that comprise a sign. Just as important as the raw geometric information is the linguistic information. Certain geometric aspects of a sign may be changed based on its usage, and without linguistic information it is impossible to synthesize grammatically correct sentences. Thus, the geometric information is somewhat similar to a keyframe, but certain aspects must be treated as parameters during synthesis.

Problem

The largest task is gathering the data once the database scheme has been created. Gathering data involves transcribing ASL, and the people who know it best are ASL signers. While motion capture initially presents an appealing option, it cannot record the linguistic aspects of a sign. It is analogous to scanning in a printed sentence and attempting to use the raw bitmap instead of extracting the characters. Further, the motion and position of a sign may be modified depending on its context. However, using a general animation package to transcribe a sign also poses problems. Learning such a package requires a significant amount of time. Few members of the deaf community are willing to invest a large amount of time in training before beginning the transcription process.

Our Approach

To reduce the large time investment, we have customized an animation system (3DS Max) to accommodate ASL. The controls are couched in terms that are more familiar to ASL signers than such conventional animation operations as rotations and translations. The hierarchical system allows a user to specify a sign in terms of hand shape, location, and orientation. Figure 1 is a screen capture of the subsystem that specifies hand shape. Data encoded by users are stored and recalled from a Microsoft Access database. A Visual Basic ActiveX control communicates between the graphics package and the database.

Initial usability studies have been quite promising. On average, it takes 10 minutes for native ASL signers to learn enough about the system to input hand shapes, and transcribing a handshape takes an average of 82 seconds.

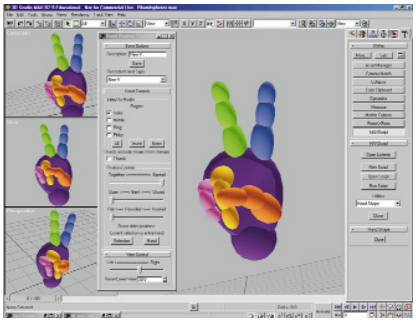


Figure 1

Transcribing

Balajee Ramakrishnananda
Wong Kok Cheong
Nanyang Technological University Singapore
aprama@ntu.edu.sg

Animating Bird Flight Using Aerodynamics

The interesting work by Haumann & Hodgins[1] addresses hovering flight and concentrates on the control aspect of flight. In our aerodynamic model for the forward flight of a bird with two wing segments, the emphasis is more on simulating the bird's undulating trajectory and the pitching motions of the bird's body due to the wing motions rather than simulating the wing motion itself.

We generate the wing shape and motions such as beating and gliding by judiciously varying the sweep, dihedral, and twist as functions of time.[2] The aerodynamic model can be applied to any other scheme that generates these as functions of time.

Based on wing shape, orientation, velocity, and motions, we calculate the aerodynamic force acting on the wing. The force distribution on the wings is predicted by an aerodynamic model that employs a combination of analytical and numerical techniques. The trajectory of the bird is then determined using principles of flight mechanics.[3]

The aerodynamic model is applicable to many flight phases and can be employed to animate imaginary characters as well. Interactive computational speeds are achieved due to assumptions of negligible induced velocity and quasi-steadiness.

We can also control the trajectory of the bird to pass through specified targets by direct or indirect control of the pitching moment. Direct control of pitching moment involves specification of the pitching moment of the bird through a p-d controller, which alters the body axis angle. The body axis angle alters the average angle of attack on the wing, thereby controlling the trajectory of the bird. The indirect specification of the pitching moment involves driving a control variable such as the wing sweep of the secondaries by means of a p-d controller. Changes in the control parameter alter the pitching moment in such a way as to propel the bird toward the target. While the direct approach is less costly, the pitching motions of the bird are not correlated to any observable wing motions. The indirect approach rectifies this deficiency, but it is about thrice as costly as the direct approach.

Our Web site contains a technical report[4] that presents the details of the methodologies and three animation sequences that illustrate the methodologies. A video sequence of a flying character utilizing the indirect approach to generate the flight trajectory was presented in the SIBGRAPI 98 Computer Animation Festival.[5] An image from this sequence is shown in the figure.

The main advantage of our approach lies in the interactive speed of the models, with which the effects of changes in parameters can be assessed.

Acknowledgements
The authors thank S.K. Lim, L.L. Goh, and B.H. Low for animating the flight sequences.

www2.cgit.ntu.edu.sg:8000/~iagwww/BirdFlight/index.html

Figure 1

References

1. D. Haumann and J.K. Hodgins. The Control of Hovering Flight for Computer Animation, Nadia Magnenat Thalmann and D. Thalmann, editors, Creating and Animating the Virtual World, Computer Animation Series, pages 3-19, Springer-Verlag, 1992.
2. A. Azuma. The Biokinetics of Flying and Swimming, Springer-Verlag, Tokyo, 1992.
3. C.D. Perkins and R.E. Hage. Airplane Performance, Stability and Control,Wiley, New York, 1949.
4. B. Ramakrishnananda and K.C. Wong. Animating Bird Flight Using Aerodynamics, Technical Report CGIT-IAG-TR98-4, Centre for Graphics and Imaging Technology. Nanyang Technological University, Singapore, Aug 1998.
5. K.C. Wong, S.K. Lim, L.L. Goh, B.H. Low and B. Ramakrishnananda. "Litter Bug," In SIBGRAPI '98 Video Festival, 1998 International Symposium on Computer Graphics, Image Processing and Vision, Rio de Janeiro, Brazil, Oct 1998.

Animating Expressivity Through Effort Elements

Diane M. Chi
Norman I. Badler
University of Pennsylvania
chi@graphics.cis.upenn.edu

Janis Pforsich
Laban/Bartenieff Institute of Movement Studies

Current methods for figure and character animation involve a tradeoff between the level of realism captured in the movements and the ease of generating the animations. At one end of the spectrum, computer-animated features and movie special effects display extremely realistic individuals with personalized movement characteristics. However, they require teams of animators using labor-intensive systems where every movement must be explicitly specified. At the other end of the spectrum, behavioral animation systems can automatically generate multiple characters that interact with each other and their environments by merely specifying a set of initial conditions, but the various characters often move in a fairly mechanical manner and share very similar motions. We introduce a motion-control paradigm that combines the advantages of both: the ability to specify a wide range of natural-looking movements with minimal user overhead.

Our approach involves simplifying generation of expressive movements by proceduralizing the qualitative aspects of movement to hide its non-intuitive, quantitative aspects and provide the user with easy-to-use, interactive control through textual and spatial descriptors.[1,2] To do this, we needed a language for describing expressive movements and a translation of that language into quantitative movement parameters.

We found that the Effort component of Laban Movement Analysis[2,3,4] met our requirements as a means for describing movements, has a solid foundation based on observation and analysis of humans performing a wide range of movements, and is being used in a growing number of fields. Effort is the part of Rudolf Laban's theories on movement that describes the qualitative aspects of movement using textual descriptors along four motion factors: space, weight, time, and flow.

The extremes of each motion factor give the eight Effort Elements: indirect, direct, light, strong, sustained, sudden, free, and bound.

We built an empirical model of Effort using kinematic movement parameters. The movement parameters fall into three general categories: those that affect the arm trajectory, those that affect timing, and flourishes. The arm trajectory is defined by two parameters: path curvature and interpolation space. The path curvature determines the straightness or roundness of path segments between keypoints. The interpolation is performed in end-effector position, joint angle, or elbow position space. The timing of movements is controlled by the number of frames between keypoints and parameters to a timing function (inflection time, time exponent, start, and end velocities), which define velocity curves, anticipation, and overshoot. Flourishes are miscellaneous parameters that add to the expressiveness of the movements and include: squash and stretch, breath, wrist bend, arm twist, displacement magnitude, and limb volume.

We describe EMOTE (Expressive MOTion Engine), a 3D animation control module that demonstrates this approach. EMOTE uses inverse kinematics with spatial movement requirements specified through end-effector positions. It lets a user express the essence of the movement by setting the four Effort motion factors. EMOTE uses these Effort settings to determine values for the kinematic movement parameters, specifies an animation that follows the defined position sequence, and displays the selected Effort qualities.

Since our translation process is computationally inexpensive, EMOTE provides interactive editing and real-time motion generation. EMOTE produces a wide range of expressive movements with an easy-to-use interface that is more intuitive than joint angle interpolation curves or physical parameters. Preliminary evaluations have shown that both experienced and first-time EMOTE users can create simple expressive character animations in a shorter amount of time at a quality that is the same or better than animations built using a popular commercial animation package. Embedded in a real-time simulation system, EMOTE provides procedural control for editing basic movements based on a character's motivation and intent.

References

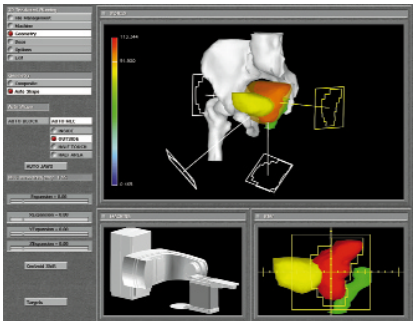
1. D. Chi. A Motion Control Scheme for Animating Expressive Arm Movements, PhD thesis, University of Pennsylvania, 1999.
2. I. Bartenieff and D. Lewis. Body Movement: Coping With the Environment, Gordon and Breach Science Publishers, New York, 1980.
3. C. Dell. A Primer for Movement Description: Using Effort-Shape and Supplementary Concepts, Dance Notation Bureau, Inc., New York, 1970.
4. C. L. Moore and K. Yamamoto. Beyond Words: Movement Observation and Analysis, Gordon and Breach Science Publishers, New York, 1988.

Marc L Kessler
Daniel L McShan
Benedick A Fraass
University of Michigan
mkessler@umich.edu

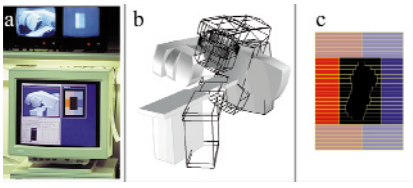
Application of Computer Graphics for Design and Delivery of Conformal Radiation Therapy

Radiation therapy, one of the most widely used forms of cancer treatment, affects the lives of over 600,000 patients annually. The challenge of planning radiotherapy treatments is to tailor delivery of radiation to conform to the tumor shape and avoid as much healthy tissue as possible. Visualization techniques exploiting 3D computer graphics and medical image data are now used to help accomplish this. By manipulating patient-specific 3D anatomic data and models, the clinician can better appreciate the complex spatial relationships among tumor, healthy tissues, and beam trajectory.

Visualization techniques are also used to help guide the delivery of the planned treatments. Together, such techniques permit more precise planning and delivery of beam arrangements, shapes, and strengths, making it possible to safely escalate radiation doses in many tumor sites well beyond previously defined levels of "tolerance" and achieve improved local control and better cure rates.



Visualization for radiation therapy planning. a) Beam trajectories and dose isosurface (transparent orange) relative to surface models of patient anatomy. b) Beam's-eye view of the target (red) and normal tissues (bladder in yellow and rectum in green). c) Machine model to help the planner detect impractical beam orientations.



Visualization for treatment delivery verification. a) A workstation at the treatment unit displays a realistic model of the machine and its components. A solid-surface representation of the machine b) and the beam-shaping collimator c) show actual machine positions, and the wireframe models indicate the intended move.

Therapy

Asynchronous, Adaptive, Rigid Body Simulation

Stephen Chenney
University of California, Berkeley
schenney@cs.berkeley.edu

Impulse-based algorithms for rigid body simulation consist of an outer loop that repeatedly steps forward in time, integrates trajectories, and identifies and processes collisions.[1] Algorithms use various methods to choose time-steps and detect and resolve collisions. Current algorithms update all objects at every step, so that:

- Individual objects are updated more frequently than necessary. An object that is far from any other will be updated at the same frequency as a set of closely packed objects.
- State integration and bounding box computation must be performed for every object at every step. Global data structures (for example, records of object overlaps) must be updated at every step.

We have developed an event-driven simulation algorithm that addresses these inefficiencies. Our algorithm considers different objects at different times (it is asynchronous) and adaptively adjusts the time at which a given object is considered based on its recent history and expected future behavior (unlike the method developed by Kim, Guibas, and Shin,[2] which cannot adapt and requires exact collision detection).

The algorithm stores state (position, velocity etc., and time) and an associated bounding volume for each object. Global data structures are: a priority queue for events, sorted lists of bounding box extents for each dimension, and a set of hash tables storing bounding box overlaps in 3D and each dimension.

There are three types of events: start, update, and pairwise. The simulation begins by inserting a start event for each moving object. An outer loop then repeatedly deletes the earliest event in the queue and processes it. The core operation is the Predict procedure, which predicts what may happen to a given object within a time interval. A spatio-temporal bounding volume that contains the object throughout the time interval is computed, and its correct position in the sorted lists is located (starting at its former position). During the re-sort, overlapping volumes and hence potential collisions are detected, and pairwise events are scheduled. Events are processed as follows:

Start: The Predict procedure is called with the object and a small initial time interval. The object's first update event is then scheduled for the end of the interval.
Update: The object's state is integrated up to the event time, and the Predict procedure is called with an interval of twice the time since the last event. Another update event is scheduled for the end of the interval.
Pairwise: The two objects involved are integrated to the current time then checked for collision. If not colliding, another pairwise event is scheduled based on a new predicted collision time. If colliding, the collision is resolved, and the update event process above is applied to each of the two objects.

Running time is determined by the total number of events and the cost of each event. The speedup we obtain depends on the simulation, but general observations indicate that the maximum possible speedup is proportional to n, the number of objects in the simulation, and experimental evidence demonstrates that this speedup is attainable. Both our algorithm and synchronous algorithms are driven by the need to perform collision tests, so the number of iterations of the event processing loop will be similar. Our algorithm processes additional update events, but adapts to keep the number low.

The advantage of the algorithm comes from a reduced cost for each event:
- Collision processing cost: will be the same for each algorithm, because both process the same sequence of collisions.
- Object integration cost: our asynchronous algorithm only integrates one or two objects per step, but the integration step-lengths are generally longer. The maximum available speedup is proportional to n.
- Overlap processing cost: Our asynchronous algorithm updates the bounding volume for at most two objects, rather than all n, and the re-sort process takes time proportional to the number of overlap changes for those objects, which is likely to be a small constant because the size of the bound used to find overlaps is adapted. A speedup of n results.

We compared the performance of our algorithm and a generic synchronous algorithm on a traditional example: simulating a number of cubes in a fixed-size box with each cube started in a random state. Each simulation ran for 500 seconds of simulated time with objects under the influence of gravity but frictionless, perfectly elastic collisions. Three different sets of initial conditions were simulated for each number of objects. The averaged results are presented in Table 1 and Figure 1. Our algorithm processes 5-20 percent more events (due to update events). However, the cost of each event is linear in n for the traditional algorithm, and sub-linear for our algorithm. Overall, for this particular scenario, the asynchronous algorithm demonstrates a speedup approximately linear in the number of objects.

	Synchronous			Asynchronous			
n	t (s)	#events	t/event (ms)	t (s)	#events	t/event (ms)	speedup
8	58.80	403255	0.1458	16.45	450857	0.0365	3.57
15	200.45	809232	0.2477	36.78	937317	0.0392	5.45
27	748.80	1680996	0.4455	81.20	1987819	0.0408	9.22
42	2123.54	3148716	0.6744	160.69	3680741	0.0437	13.21
64	6367.87	6083977	1.047	317.13	6928818	0.0458	20.08
91	15629.0	10758042	1.453	572.43	11923404	0.0480	27.30
125	37493.2	19098906	1.963	999.69	20122471	0.0497	37.50

Table 1: Results from timing tests comparing our asynchronous algorithm with a generic synchronous algorithm. n is the number of moving objects in the simulation and t is the total execution time. Times were obtained on a 200MHz Intel Pentium Pro.

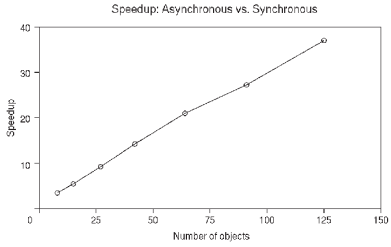


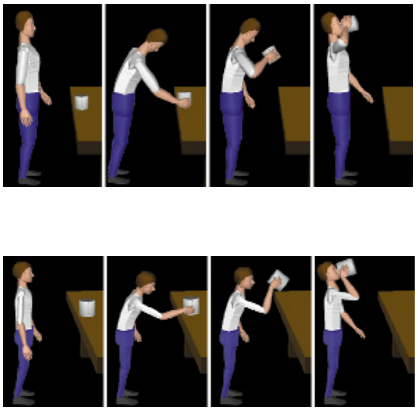
Figure 1: A plot of the speedups from table 1. Our new algorithm demonstrates a linear improvement for this scenario.

References

1. Brian Mirtich and John Canny. Impulse-Based Simulation of Rigid Bodies, Proceedings of the 1995 ACM Symposium on Interactive 3D Graphics, 181-188, 1995.
2. Dong-Jin Kim, Leonidas J. Guibas and Sung-Yon Shin. Fast Collision Detection Among Multiple Moving Spheres, IEEE Transactions on Visualization and Computer Graphics, 230-242, 4(3), 1998.

Rama Bindiganavale
Norman I. Badler
University of Pennsylvania
rama@graphics.cis.upenn.edu

Automatic Recognition and Mapping of Constraints for Motion Retargetting



Drink-from-mug action mapped from a six-foot-tall adult model (top) to a 4.5-foot-tall child model (bottom).

In this sketch, we address the problem of automatically controlling[1] virtual human models of various sizes and degrees of freedom that imitate the action of a motion-captured subject. We abstract information about significant events and spatial constraints from 3D motion-captured data. A subject, called the primary agent, is motion captured executing a specific action. We are mainly interested in movements involving interactions with other objects (for example, drinking from a cup, digging with a shovel, etc.). The imitators, referred to as secondary agents, try to execute the same action by interacting with the objects in a similar fashion. As the primary and secondary agents may be of different sizes, the motion imitation will be similar but not exact. We assume that maintaining spatial constraints for hands and eyes is more important than maintaining a similar trajectory or motion style between the constraints.

Gleicher[2] has addressed the problem of mapping constrained motions to different-sized agents using space-time constraints. But that technique cannot be solved in real-time, does not consider constraints on the figure's gaze direction, and is unable to automatically recognize the occurrence of the start and end times of constraints.

Zero crossings of the second derivative indicate significant changes in the motion and correspond to the local maxima of velocity. The start and end times of a spatial/visual constraint can be automatically established by the co-occurrence of the zero crossings of the raw data and the proximity of the end effector with the object. To reduce computation time, we tag only some specific sensors (monitoring sensors) for zero-crossing tracking and some specific objects (tag objects) for computing spatial proximity.

We begin by generating motions for the primary agent from motion-captured data. Then, we automatically recognize and map the constraints to a second agent in three stages. In the first stage, Recognition, a constraint is automatically set or deleted based on the proximity of a monitoring sensor and a tag object at a zero-crossing frame of the sensor. In the second stage of Location Determination, the new constraint location for the secondary agent is determined based on the type of the tag object. Finally, in the third stage, Motion Generation, inverse kinematics is used to solve for the joint angles at the relevant zero-crossing frames, and joint-angle interpolation techniques are used for the in-between frames.

To maintain a similar action style (frame-wise variations in the angular velocity), we compute the normalized angular distance, moved in joint space along a trajectory by the primary agent, at each frame. This style factor used as the interpolation factor causes the velocity profiles in the actions of the two agents to be the same and also retains the zero crossings of the original.

Capturing and maintaining visual attention is very important for movement realism in the secondary agent. Without it, actions appear unnatural even if all the other constraints are correctly satisfied. For visual constraints, we use a head-top sensor to track the focal points of attention. At zero crossings of the acceleration of the head sensor, we check for intersections of the line of sight with the tagged objects. The type of the tag object determines the exact location of the visual constraint for the secondary agent.

This new motion-capture tool could work with and augment other motion-editing software. It was successfully tested to automatically recognize and map actions of an adult to various virtual models including children.

References

1. Rama Bindiganavale and Norman Badler. Motion Abstraction and Mapping with Spatial Constraints, In Modeling and Motion Capture Techniques for Virtual Environments, International Workshop, CAPTECH '98, pp 70-82, Geneva, Switzerland, November 1998. Springer-Verlag.
2. Michael Gleicher. Retargetting Motion to New Characters, In Computer Graphics Proceedings, pp. 33-42.SIGGRAPH, ACM, 1998.

Capturing the Motions of Actors in Movies

Masanobu Yamamoto
Katsutoshi Yagishita
Hajime Yamanaka
Naoto Ohkubo
Niigata University
yamamoto@ie.niigata-u.ac.jp
www.vision.ie.niigata-u.ac.jp/~yamamoto/

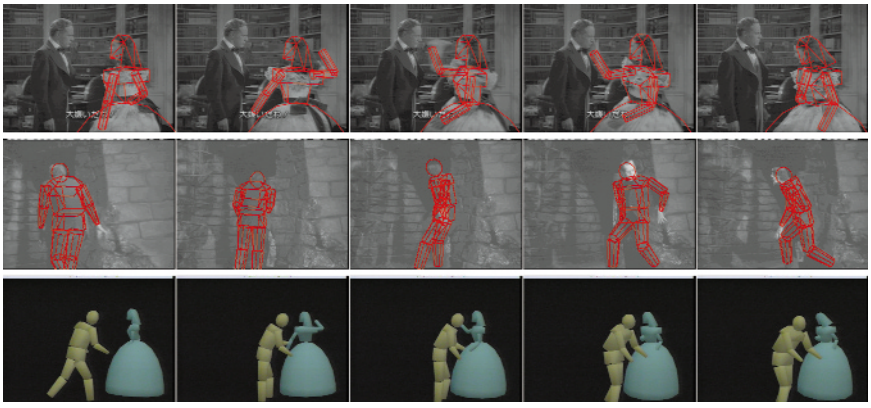
Image-based tracking of the human body in 3D motion has become an important technology for generating the motions of characters in computer animations and TV games. This sketch proposes an image-based method for capturing the motion of actors from movies. Assuming that an articulated model of an actor and the 3D pose at a few keyframes (for example, a starting frame, intermediate frames, and final frame), are given, this method can calculate a pose sequence of the actor from an image sequence. The pose sequence is obtained from iterative minimizing of an error function that is composed of some constraints to be satisfied by the human motion.

Using our proposed method, we show examples of capturing actors' motions from old movies. The first row in the figure depicts a tracking result of Scarlett (Vivien Leigh) slapping in "Gone with the Wind," 1939, obtained by overlapping a wire-frame representation of her 3D model onto the images. The second row in the figure depicts a tracking result of the monster (Boris Karloff) in "Frankenstein," 1931.

This tracking captures 3D motions of the actors. Therefore, we can faithfully reproduce their actions in a new movie. We produced a new short animation, "Scarlett is Slapping the Monster," based on the tracking results. The third row in the figure shows several images sampled from the new animation.

Most of the tracking methods proposed so far fit the model to the human body on a frame-by-frame basis. A disadvantage of this approach is that enormous computation resources are required in the process of analyzing the human body. In contrast, our method is a direct approach without the search process. It requires model fitting at least once (usually at the starting frame) to save computational resources and continues estimating and accumulating pose displacements for tracking.[1,2,3]

The pose displacements (motion parameters) can be directly estimated from the image sequence by gradient-based techniques. However, the estimation errors of the pose displacements are also cumulative. Over the long term, tracking may result in a great discrepancy between the model and the human body. Our method can eliminate this discrepancy by propagating the correct pose given at a few keyframes. Even if a part of the human body is occluded at some frames, this propagation enables the motion in these frames to be estimated.



References

1. M. Yamamoto and K. Koshikawa. Human Motion Analysis Based On a Robot Arm Model, Proc. IEEE Conf. Computer Vision and Pattern Recognition, pp.664-665, 1991.
2. M. Yamamoto, A. Sato, S. Kawada, T. Kondo and Y.Osaki. Incremental Tracking of Human Actions From Multiple Views, Proc. IEEE Conf. Computer Vision and Pattern Recognition, pp.2-7, 1998.
3. C. Bregler and J. Malik. Tracking People With Twists and Exponential Maps, Proc. IEEE Conf. Computer Vision and Pattern Recognition, pp.8-15, 1998.

Jacquelyn Martino
Nevenka Dimitrova
Lalitha Agnihotri
Herman Elenbaas
Philips Research, USA
iam@philabs.research.philips.com

Color Super-Histograms for Video Representation: Preliminary Research and Findings

Figure 1
Extraction method for creating super-histograms. A super-histogram is the ordered set of family histograms. The system orders the sets with respect to the temporal length of the video segment they represent.

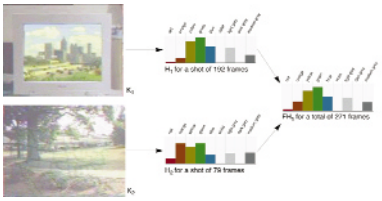


Figure 2
An example of an extracted family histogram.

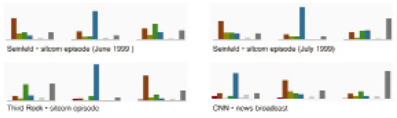


Figure 3
Comparison of results: the three largest family histograms for four different TV broadcasts.

References

1. M. Abdel-Mottaleb, N. Dimitrova, R. Desai and J. Martino. CONIVAS: Content-Based Image and Video Access System, Proc. of ACM Multimedia, pp 427-428, Boston 1996.
2. S. Krishnamachari and M. Abdel-Mottaleb. A Scalable Algorithm for Image Retrieval By Color, The Fifth International Conference on Theoretical, Experimental and Applied Image Processing, ICIP-98, Chicago, October 1998.
3. S-F. Chang, W. Chen, H. E. Meng, H. Sundaram and D. Zong. VideoQ: An Automated Content-based Video Search System Using Visual Cues, Proc. ACM Multimedia, pp. 313-324, Seattle, 1994.
4. M. Christel, T. Kanade, M. Mauldin, R. Reddy, M. Sirbu, S. Stevens and H. Wactlar. Informedia Digital Video Library, Comm. of the ACM, Vol. 38, No. 4, pp. 57-58, 1995.
5. T. McGee and N. Dimitrova. Parsing TV Programs For Identification and Removal of Non-Story Segments, SPIE Conference on Storage and Retrieval in Image and Video Databases, January, San Jose, 1999.
6. W. Niblack, X. Zhu, J.L. Hafner, T. Bruel, D. B. Ponceleon, D. Petkovic, M. Flickner, E. Upfal, S.I. Nin, S. Sull, B.E. Dom. Updates to the QBIC System, Proc. IS&T SPIE, Storage and Retrieval for Image and Video Databases VI, Volume 3312, pp. 150-161, San Jose, 1998.
7. S. Pfeiffer, R. Lienhart, S. Fischer, and W. Effelsberg. Abstracting Digital Movies Automatically, Proc. Journal on Visual Communications and Image Representation, Vol. 7, No. 4, pp. 345-353, 1996.

While working in the areas of keyframe extraction and commercial detection, we noticed that television programs tend to have strong visual consistencies within a specific program series. For example, we can easily identify a program as "Seinfeld" based on a very few frames of video. Obviously, this is due in part to our human ability to recognize key personalities who appear episode after episode. After watching very few episodes, we can also quite easily recognize the locations and set designs, or backgrounds, that recur during episodic programming. Since television content creators carefully craft these sets and locations, television programs develop an inherent visual consistency. With the current state of the art, some of these consistencies tend to be easier for computers to recognize than, say, the actor Jerry Seinfeld. Color is one such important aspect that content creators maintain through set and location design. This is also an aspect that computers are good at recognizing and classifying.

In our current research in progress, we hypothesize that classification of thematic visual elements is useful in overall recognition and classification of video content when we combine such classification with other methods. Specifically, we are developing a method that uses color information to represent video segments. We apply this method after applying video indexing and classification[1,3,4,5,6] methods.

In order to classify video content based on color, we have devised a method for computing super-histograms, which we outline in Figure 1. Briefly, the method computes color histograms for individual shots and then merges the histograms into a single cumulative histogram called a family histogram based on a comparison measure (for example, Euclidean distance). As in Figure 2, given the keyframes K_1, for a shot of 192 frames, and K_2, representing 79 frames, we extract the two histograms H_1 and H_2. We then merge H_1 and H_2 into family histogram FH_1. We use a total of 271 frames to create the family histogram. This family histogram represents the color union of the two shots. However, if the histogram of a new frame is different from the previously constructed family histograms, we form a new family. In the end, we have a few temporally interleaved families of histograms to represent the entire television program. We order this set of families with respect to the length of the temporal segment of video that they represent. The ordered set of family histograms is called a super-histogram.

In the examples in Figure 3, we calculated the histogram difference, D, using L2 distance measure. In the formula below, H_c is the current histogram, H_p is the previous histogram and N is the number of color bins (9 in our example). The values obtained using this formula range from 0 to twice the maximum number of pixels per image. To obtain percentage of similarity, we normalize the values of D by dividing with the total number of pixels. The normalized values are between 0 and 1. Values close to 0 mean that images are similar while values close to 1 mean images are dissimilar.

$$D = \sqrt{\sum_{i=1}^{N} (H_c(i) - H_p(i))^2}$$

The super-histograms representing episodes of the same sitcom look strikingly similar. The histograms extracted from TV news are not similar to the histograms extracted from sitcoms. This method can be used in a variety of applications that need video classification and retrieval methods: video editing, studio archiving, digital library search and retrieval, consumer products, and Web crawling.

Color histogram indexing has traditionally been applied to image retrieval[2] and video segmentation.[7] However, our preliminary results show that the histogram extraction and indexing method can be applied to video. We maintain that systematic analysis of thematic visual elements such as color can assist in defining richer characterization and description of video content. To this end, we are pursuing a method for representing video segments using cumulatively averaged, or super-histograms. In the immediate future, this extremely compact representation can be used for a variety of applications including program boundary detection and classification and search in large video archives. In the more distant future, we envision systems that use color as one of many visual elements to understand viewing habits and to reveal information about content at a thematic level.

Declarative Behaviors for Virtual Creatures

Philippe Codognet
University of Paris
philippe.codognet@lip6.fr
pauillac.inria.fr/~codognet

We present a high-level language (VRCC) for describing behaviors of reactive autonomous creatures in 3D virtual worlds. VRCC is a concurrent programming language connected to the VRML environment and based on the timed concurrent constraint (TCC) framework of V. Saraswat et al.

Declarative Behaviors Based on Constraints
The basic idea of TCC is that of synchronous programming exemplified by languages such as Esterel by G. Berry et al, programs run and respond to signals instantaneously. Time is decomposed in discrete time points generated by clocks outside the system, and programs are executed between time points. Thus, at each time point, concurrent agents are running simultaneously until quiescence, and then the program moves to the next time-point. This achieves the classic sense-think-act loop of autonomous agent systems. Basic actions performed by the agents are either posting a constraint to a shared store (Tell operation), suspending until some constraint is entailed by the store (Ask operation), performing some method (Call operation), or posting an action to be performed at the next time-point (Next operation). The Next operation allows actions to be delayed or executed repeatedly over time. Combined with the use of constraints and the event-driven behavior of the Ask operations, it provides a very simple and powerful mechanism to declaratively define behaviors. Constraints are used to state relations that have to be satisfied by the agent: minimal distance with regard to some other object (collision avoidance), maximal distance (following behavior), equidistance (flock or boid-like behavior), etc. It is, moreover, easy to combine such simple behaviors to build more complex ones.

Biologically Inspired Creatures
In order to design autonomous, life-like creatures that can autonomously navigate in the 3D world, we propose some simple behaviors derived from biologically inspired models of navigation. There is currently a growing interest in such models in both the artificial life community and the robotics community, and such models could obviously be applied to virtual agents as well. The creatures will have to react to a changing environment and avoid collision with moving obstacles.

Here, we only consider an agent that uses the taxon system for route navigation, with a very limited capacity for intelligence and no cognitive map. In our experiments, the creature traces a simple route toward a goal by avoiding obstacles. Therefore, the basic constraint is non-collision (enforcement of some minimal distance with regard to obstacles). But we have also considered some non-trivial navigation, as the creature does not know the location of the goal in advance but rather has to explore its environment, guided by a stimulus (for example, light or smell) toward the goal (food, for example).

In particular, we have investigated exploration guided by a stimulus using either temporal difference or spatial difference methods. Temporal differences consist of considering a single sensor (for example, the nose) and checking the intensity of the stimulus at every time point. If the stimulus is increasing, the agent continues in the same direction. Otherwise, the direction changes randomly. This behavior is exemplified, for instance, by the chemotaxis (reaction to a chemical stimulus) of Caenorabditis Elegans, a small soil nematode.

A more efficient strategy uses the spatial differences method, which requires two identical sensing organs placed at slightly different positions on the agent (for example, the two ears). The basic idea is simply to favor, at any time point, motion in the direction of the sensor that receives the stronger stimulus. This behavior gives very good results, and the creature moves directly toward the goal. When the goal is moved away, the agent reacts instantly to the new location.

Richard Hammersley
Hong-Qian (Karen) Lu
Schlumberger Limited

Decremental Delaunay Triangulation

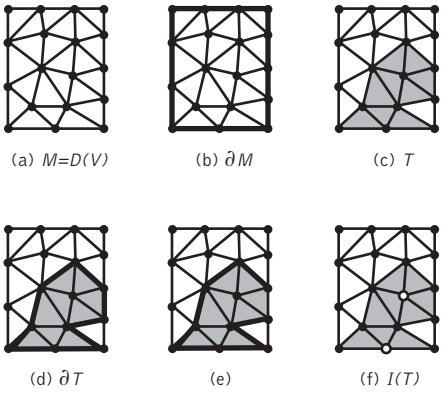


Figure 1 Definitions.

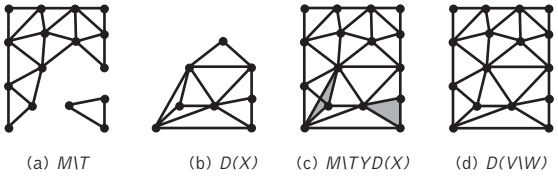


Figure 2 Decremental Delaunay triangulation algorithm.

Incremental Delaunay triangulation is a well-studied problem. Removing points from a Delaunay triangulation or decremental Delaunay triangulation is less studied.[1,2] This sketch presents theoretical results and an algorithm for decremental Delaunay triangulation.

Often, many points are removed, and it can be more efficient to remove many points at once than to remove them singly.
Let $\{a,b,c\} \in R^2$. Let $[a,b]$ and $[a,b,c]$ denote a line segment and triangle respectively. Let F denote the face operator, $F([a,b])=\{a,b\}$, $F([a,b,c])=\{[a,b],[b,c],[c,a]\}$, extended to collections of edges and triangles. Let M be a triangulation. The *boundary* of M, ∂M, is all those edges which are the edge of an unique triangle. If $T \subset M$, then the *sub-boundary* of T in M, δT is given by $\partial T \setminus \partial M$ (\ is subtraction). The *sup-interior* of T in M are the vertices given by $I(T)=FF(T)\setminus F(\delta T)$. Figure 1 gives examples of boundaries, a sub-boundary and a sup-interior. Let $V=\{v_1,\ldots,v_n\} \subset R^2$ be such that no four points lie on a common circle. Let $D(V)$ be the Delaunay triangulation of V. The following lemmas and corollary can be proven.

Lemma 1. Let $\{a,b\} \subset X \subset V$ with $[a,b] \in D(V)$. Then $[a,b] \in D(X)$.

Lemma 2. Let $T \subset D(V)$. Then for all $W \subset I(T)$, $D(V) \setminus T \subset D(V \setminus W)$.

Lemmas 1 and 2 show decremental Delaunay triangulation is a local operation. Furthermore, when removing multiple points many edges and hence triangles remain. The unshaded triangles in Figure 1-(f) will be in the Delaunay triangulation, $D(V\setminus W)$. This is similar to results for incremental Delaunay triangulation.[2]

Lemma 3. Let V, T and W be as in Lemma 2. Let $X \subset V$ such that $X \cap W = \emptyset$ and $FF(T) \setminus W \subset X$. Let $S \subset D(X)$ be all those triangles outside of δT. Then $D(V\setminus W)=D(V)\setminus T \cup D(X)\setminus S$.

Definition. Let T be a triangulation. For $[W \subset FF(T)$ let $Star(W)=Y\{t \in T \mid FF(t) \cap W \neq \emptyset\}.]$

Corollary 4. Let $W \subset V$. Let $T = Star(W)$, let $X=FF(T)\setminus W$ and let $S \subset D(X)$ be all those triangles outside of δT.
Then $D(V\setminus W)=D(V)\setminus T \cup D(X)\setminus S$.

Corollary 4 gives a method to compute a decremental Delaunay triangulation. We have shown that the Delaunay triangulation of the star is sufficient even when the star is not convex. Figure 2 illustrates Corollary 4 applied to the mesh in Figure 1.

There are similar results for constrained decremental Delaunay triangulation. To remove a constraining edge it is necessary to use the star of the edge, that is the star of the two end vertices of the constraining edge.

Decremental and incremental Delaunay triangulation is used in our work and is an efficient means to maintain a mesh during adaptive visualization of large parameterized surfaces.

Acknowledgement
We would like to thank Andreas Hubeli for our first decremental Delaunay triangulation implementation.

References
1. M. Heller. Triangulation Algorithms For Adaptive Terrain Modeling, Proceedings of the 4th International Symposium on Spatial Data Handling, pages 163-174, July 1990.
2. T. Midtbo. Removing points from a Delaunay Triangulation, Proc. 6th Int. Symposium on Spatial Data Handling, pages 739-750, vol. 2, Advances in GIS Research, 1994.

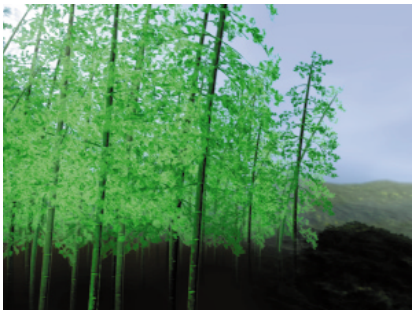
Dynamic Texture: Physically Based 2D Animation

Mikio Shinya
Masakatsu Aoki
Ken Tsutsuguchi
Naoya Kotani
NTT Cyber Space Laboratories
shinya@nttcvg.hil.ntt.co.jp

The physically based approach is successful in creating realistic motion for 3D objects. Modeling complex 3D shapes is, however, a labor-intensive job, and their rendering is computationally expensive. This limits the application of physically based animation to top-end computer animation.

Coupling physically based techniques and image morphing techniques has the potential to yield new directions in computer animation. In this approach, complex 3D models are not required, so it can be applied even to 2D animation. This idea was first proposed by Overveld.[1] Unfortunately, the paper was rather conceptual, without strong experimental support, and little attention has been paid to this approach.

This sketch demonstrates successful examples of this coupling approach. We successfully synthesized the stochastic motion of plants under the influence of wind, in which 2D textures are realistically animated based on dynamic simulation.



Dynamics
Modal analysis was adopted for dynamic simulation because it has been successfully applied to the motion synthesis of swaying trees.[2] In modal analysis, deformation of branches is described by a second-order differential equation, which can be solved, for example, by Euler's method.

The applied force is calculated from a stochastic wind field synthesized by the Fourier method.[2] Since the applied force is periodic, the equation has a periodic solution. Periodic motion is convenient because it can be repeatedly used in long animation sequences.

To obtain the required periodic solution, we adopt the superposition method.[3]

Procedure
Input images can be created by using conventional painting tools. To separate the images into components (for example, trunks and branches), the image parts are created in different layers.

Skeletons
Skeletons are specified for each component. In the current implementation, we manually draw skeletons as line segments on the image via a GUI. However, since our images are already segmented, it is not difficult to automatically extract skeletons from images by using image-processing techniques. We also specify kinematic characteristics, such as natural frequencies, amplitudes, and damping time, which are then converted to mechanical properties.

Dynamic simulation
The dynamic simulation is performed by the modal analysis, and it yields the periodic motion of skeletons.

Morphing
Following the deformed skeletons, the corresponding textures are deformed by using piece-wisely linear mapping functions.

Result
We successfully applied the method to create the texture-based animation, "Gift for Nature." Although the plants were all modeled using 2D textures, the motion appears very realistic. Images to the right are example shots from the sequence.

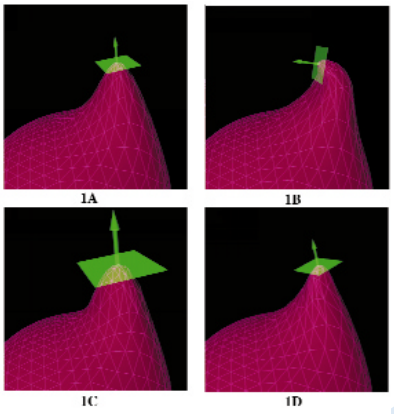
References
1. C. van Overveld. A Technique for Motion Specification in Computer Animation, The Visual Computer, vol. 6, pp. 106-116, 1990.
2. M. Shinya, A. Fournier. Stochastic Motion - Motion Under the Influence of Wind, Computer Graphics Forum (Proc. of Eurographics '92), vol 11, No.3, pp. C-119-128, 1992.
3. M. Shinya, T. Mori, N. Osumi. Periodic Motion Synthesis and Fourier Compression, The Journal of Visualization and Computer Animation, vol. 9, pp. 95-107, 1998.

James Gain
Neil Dodgson
University of Cambridge
james.gain@cl.cam.ac.uk
www.cl.cam.ac.uk/~jeg24

Enhancing the Efficiency and Versatility of Directly Manipulated Free-Form Deformation

Directly manipulated free-form deformation (DMFFD)[3] is a constraint-based space deformation technique that addresses the task of free-form solid modeling. The user is able to drag surface points directly and have the surrounding solid conform smoothly. This sketch focuses on improving the efficiency and versatility of DMFFD.

Free-form deformation: (FFD) warps the space surrounding an object and thereby transforms the object indirectly. The object is contained within a parallelepiped lattice of control points. Object points (*x, y, z*) are assigned local co-ordinates (*u, v, w*) within the lattice and their deformation is expressed as a weighted sum of control point displacements. For efficiency and local control reasons, we employ trivariate cubic B-spline tensor product basis functions and a world co-ordinate aligned lattice. FFD can be expressed in matrix notation as follows: $\Delta Q = B\Delta P$ (1)



Each row of ΔQ represents a change in the (*x, y, z*) co-ordinates of an object point. ΔP stores (by row) the changes in those control points that influence the object points; B holds the basis functions. A particular (*i,j*) entry of B is zero if the control point in row *j* of ΔP does not affect (because of local control) the object point in row *i* of ΔQ.

Direct Manipulation
Controlling deformations by moving lattice vertices, while producing sculpted results, tends to be cumbersome and counter-intuitive.[3] Direct manipulation uses the motion of a collection of constraint points to dictate alterations in the lattice. This new lattice is then applied to the original object. With DMFFD a user specifies the motion of some constraint points (ΔQ) and the system of linear equations (eqn. 1) is solved to find the corresponding alteration in control points (ΔP). In our context, eqn. 1 is generally underdetermined, and we use a pseudo-inverse solution:

$$\Delta P = B^T(BB^T)^{-1}\,\Delta Q \qquad (2)$$

Efficiency (A Compact Basis Matrix)
We considerably improve the space cost and efficiency of the pseudo-inverse calculation by exploiting the sparsity of B.[2] Each row of B holds 64 non-zero coefficients (weights for a contiguous 4 by 4 by 4 block of control points). The remaining row entries are zero filled. Instead of standard matrix multiplication to form $A = BB^T$ in eqn. 2, we calculate each $A_{(i,j)}$ entry by intersecting the lattice blocks of row *i* and *j* of B and summing the multiplied weights of this overlapping region. The remainder of eqn. 2 proceeds by modified Choleski Factorization. This technique represents an order of magnitude speedup (for example,15.4 times speedup with 60 constraint points across 10 adjacent blocks) over the previously best approach.[3]

Versatility (Direct Manipulation of First Derivatives)
DMFFD provides direct manipulation of the position of constraint points. This can be extended (from work by Fowler[1]) to encompass the first derivative frame. For first derivative manipulation at a single point (q), eqn. 1 takes the form:

$$\begin{bmatrix} b & \emptyset \\ b_u & \Delta q_u \\ b_v & \Delta q_v \\ b_w & \Delta q_w \end{bmatrix} \Delta P \qquad (3)$$

b is a row of basis function coefficients evaluated at q. The subscripts denote partial derivatives with respect to u,v,w as appropriate. The first row ($b\Delta P=\emptyset$) constrains q to a stationary position. The partial derivatives can be extracted in a geometrically intuitive fashion from a rotation matrix (J) applied to the first derivative frame at a point, since: $[\,\Delta q_u^T \quad \Delta q_v^T \quad \Delta q_w^T\,] = J - I$ (4)
This result follows from the contravariant transformation rule[4] (since J is also the Jacobian matrix of FFD) and the simplicity of the mapping is due to the world co-ordinate alignment of the lattice.

Results
The figures above illustrate an initial object (1A) and the results achievable by tilting (1B), scaling (1C), and twisting (1D) the first derivative frame at a constraint point. Each of these deformations was executed in real-time on an SGI Indigo[2] 175MHz R10000. Previously, as many as 20 positional direct manipulations would be required to achieve the same results.

References
1. Fowler, B. Geometric Manipulation of Tensor Product Surfaces, Computer Graphics (1992 Symposium on Interactive 3D Graphics), pp. 101-108.
2. Gain, J. Virtual sculpting: An Investigation of Directly Manipulated Free-Form Deformation in a Virtual Environment, MSc. Thesis, Rhodes University.
3. Hsu, W., Hughes, J., and Kaufman, H. Direct Manipulation of Free-Form Deformations, Computer Graphics (SIGGRAPH 92 Proceedings), pp. 177-182.
4. Millman, R., and Parker, G. Elements of Differential Geometry, Prentice-Hall, Englewood Cliffs, NJ.

Fast Polygon Mesh Querying by Example

James Gain
James Scott
University of Cambridge
james.gain@cl.cam.ac.uk
www.cl.cam.ac.uk/~jeg24

The reusability of 3D models has led to a proliferation in the number and size of 3D object repositories. These typically store objects in some polygon mesh file format and allow selection by means of categories and keywords. While keyword searches are useful in many circumstances, they suffer from the subjectivity, ambiguity, and resolution difficulties inherent in natural language. These problems become more acute as database sizes increase and may force users to sift through large numbers of returned thumbnail images.

One solution is to query the object database by example. The user constructs a rough low-resolution query model with the shape characteristics of the target object. This is then compared against entries in the database, and the closest matches are returned in order of similarity. The core of this method is the matching algorithm.

We advocate the use of wavelet decomposition[1] because this technique:
- Accepts objects at widely differing resolutions.
- Allows an overall reduction in fine detail by truncating wavelet coefficients, thereby mimicking the human focus on global shape characteristics.
- Has, as will be shown, fast search times.

Our implementation is an extension of wavelet-based image querying[2] to 3D polygon mesh objects. As shown in Figure 1, wavelets can be used to derive a "signature" from the original object in four steps: normalization, voxelization, wavelet decomposition, and signature extraction.

Polygon Mesh Objects
Our implementation imposes two restrictions on the input objects.
1. Only shape is considered, and surface details are discarded. In future work, this limitation could be overcome by a secondary image query on any texture maps.
2. A standard "natural" orientation and position are assumed. For instance, the positive y- and z-axes represent upwards and forwards respectively, and the object is centered on the origin.

Normalization
Given these considerations, normalization proceeds by uniformly scaling the object to fit within a unit cube, thereby preserving the aspect ratio and orientation.

Voxelization
This is accomplished using a 3D generalization of the standard polygon scan conversion algorithm.

Wavelet Decomposition
The non-standard Haar wavelet basis[2] is applied in each dimension to decompose the voxel array into a set of wavelet coefficients. The Haar basis was selected because of its simplicity and consequent efficiency. Inclusion and comparison of other wavelet basis functions are left as areas for future research.

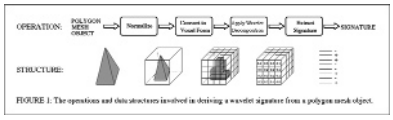
Signature Extraction
In keeping with Jacobs et al.,[1] a compact signature is then extracted by truncating and quantizing the coefficient array. Only the sign of the larger coefficients is retained. One exception is the average intensity, which is stored without quantization.

The Signature File
Two operations are permitted on a signature: incorporation into and comparison against the signature file. Since adding signatures is not time critical and can be executed offline, the signature file is optimized for comparison operations by using Jacobs et al.[1] inverted structure.

Performance
The signature extraction process executes in less than a second for an object with 8000 polygons. Since query models are likely to have low polygon counts, this is not a time-critical component of the system. The table lists the execution times of a signature query for different database sizes (numbers of models). Timings were taken on a Pentium II 300 MHz with 96Mb RAM.

Database Size	10000	20000	30000	50000
Query Times	0.001s	0.01s	1.0s	2.5s

Figure 1



References:
1. Stollnitz, E., De Rose, T., and Salesin, D. Wavelets For Computer Graphics: Theory and Applications, Morgan Kaufmann, San Francisco (1996).
2. Jacobs, C., Finkelstein, A., and Salesin, D. Fast Multiresolution Image Querying, Computer Graphics (SIGGRAPH 95 Proceedings), pp. 277-286.

Geoff Wyvill and Kevin Novins
University of Otago
geoff@otago.ac.nz
novins@cs.otago.ac.nz

Filtered Noise and The Fourth Dimension

Filtered noise[1] has become an indispensable tool for creating texture and animation effects. It is best generated by convolving a filter function with a sparse grid of pseudo-random numbers (PRNs).[2] If the grid of PRNs is "poorly packed," the number of grid points falling within the filter radius can vary wildly depending on the location of the sample point. This causes undesirable axis-aligned features.

In this sense, a cubic grid in space is poorly packed. In a 3D grid sampled with a filter of radius 2 grid spacings, the number of grid points used varies from 28 to 56.

We have improved noise quality by using a grid that is more densely and evenly packed than a simple, axis-aligned cubical arrangement. This works particularly well in four dimensions.

In two dimensions, the densest and "most even" packing of points is the triangular grid (Figure 1). This can be indexed using a skewed grid suggested by the blue lines.

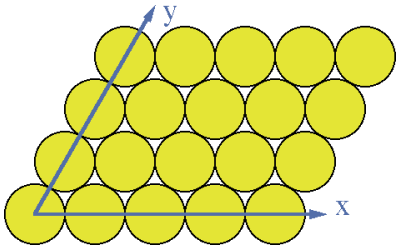


Figure 1
Ideal 2D packing

Spheres do not pack well in 3D, but in 4D there is a very neat packing related to the 2D case. Figure 2 shows a 3D cubical grid of red spheres with an additional blue sphere at the centre of each cube. The blue spheres constitute a second, identical, interleaved grid. A grid in 3D space can be seen as a 3D slice of a corresponding grid in 4D. The blue spheres represent an adjacent slice, shifted along the red/blue axis. In Figure 2, the spheres are drawn undersize to show the structure. If the spheres have a unit diameter and the grid consists of unit cubes, then the (red) spheres touch their neighbors.

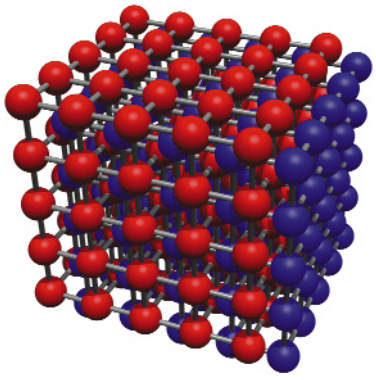


Figure2
Tight packing in 4D

The spheres are really 3D sections of 4D hyperspheres, and the blue hyperspheres in the next layer each touch eight of the adjacent red hyperspheres. Each hypersphere touches a total of 24 adjacent hyperspheres: eight in its own layer and eight at the corners of a cube in the 3D layer "before" and "after." Like the 2D packed grid, this grid can be indexed neatly by means of skewed axes.

Using this structure and a filter radius of 1.3, between 25 and 40 grid points affect any one noise value. We achieve a cheaper and better-quality 3D noise by taking a slice from a 4D structure than by working directly in 3D.

Is this structure the best possible? Surprisingly little has been proved about densest packing of hyperspheres. Conway and Sloane[3] have published a useful set of plausible hypotheses. Our structure fits with these.

An interesting application of 4D noise is the generation of a simulated underwater lighting effect. A 3D slice of the noise is used to apply lighting to every surface in the scene. The slice is then moved through the fourth dimension to produce the appearance of realistic caustics changing with time. Figure 3 shows a frame from "Dragon Gate."[4] The waterfall, mist, splashes, pebbles, and underwater lighting are modeled and animated using 4D noise.

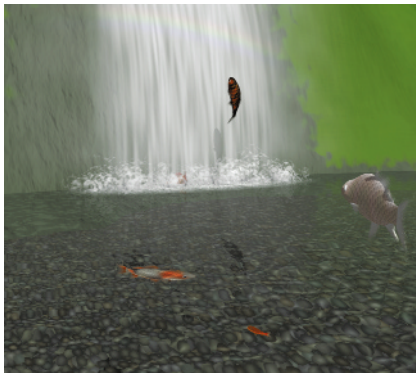


Figure 3
"Dragon Gate:" Leaping Fish

Implementation details are available at:
atlas.otago.ac.nz:800/~geoff/graphics/Noise.html

References:

1. Perlin, K. An Image Synthesizer, Computer Graphics (SIGGRAPH 85 Conference Proceedings), Vol. 19, No. 3, July 1985, pp. 287-296
2. Lewis, J.P. Algorithms For Solid Noise Synthesis, Computer Graphics (SIGGRAPH 89 Conference Proceedings), Vol. 23, No. 3, July 1989, 263-270
3. J.H. Conway and N.J.A. Sloane. Why Are All The Best Sphere Packings In Low Dimensions? Discrete and Computational Geometry, 13, 1995, 383-403
4. "Dragon Gate," Directed by Geoff Wyvill, University of Otago, 1999

Freeform Curve Generation By Recursive Subdivision of Polygonal Strip Complexes

Ahmad Nasri
American University of Beirut
anasri@aub.edu.lb

Recursive subdivision is an active area of research in computer graphics, modeling, character animation, and multiresolution.

A subdivision surface can simply be defined by a couple (P,R) where P is a mesh of arbitrary topology and R is a set of rules. The idea is to apply R to P to generate another mesh to which R is applied again and so on. At the limit, the mesh converges to a smooth surface.[1,2]

A polygonal strip complex is a sequence of n-sided polygons (or panels) such that every two adjacent panels share one edge only. The set of midpoints of the shared edges form the vertices of a control polygon called mid-polygon of the strip.

A free-form curve can be defined by a couple (Q,R) where Q is a polygonal strip complex and R is a set of rules applied to Q. At the limit, the strip will converge to a curve whose control polygon is difficult to determine. However, if the panels of the strip are symmetric, the limit curve will be the piecewise B-spline curve of the mid-polygon of the strip Q. A panel is symmetric if its vertices are symmetric about the Chebychev points of the edge joining the midpoints of its shared edges. A straightforward application is that if a symmetric strip Q is included in the mesh defining a subdivision surface S, the limit curve of Q is interpolated by the limit surface S with G1 continuity across.[3]

If two strips share one panel f (intersecting strips), their limit curves will intersect at the centroid of f, and the curves are tangent to it. If the shared edges of f are opposite (adjacent), the curves meet with C1 (C0) continuity (Figure 1).

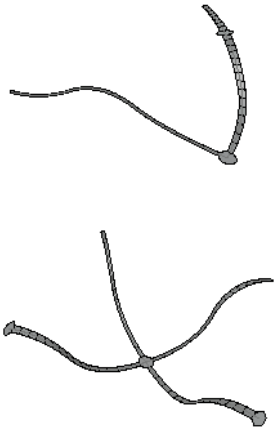


Figure 1

As applications, given an arbitrary set of intersecting strips, a subdivision surface can be generated through their intersecting limit curves,[4] (Figure 2). Alternatively, given a mesh P with tagged control polygons, our method constructs corresponding strips on P or its subsequently refined mesh, and the limit surface will interpolate the curves of the tagged polygons. Another application is that the shape of a subdivision surface can be controlled across its interpolated curves by repositioning the vertices of the panels around its Chebychev points.

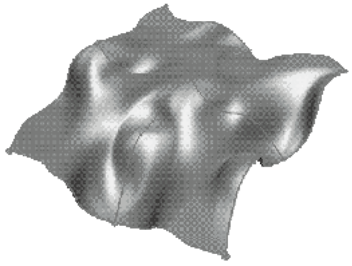


Figure 2

Finally, a surface can be trimmed along an interpolated curve by cutting along the mid-polygon of the polygonal strip.

Our future work incorporates extending the quadratic approach to higher-order surfaces.[5]

References

1. E. Catmull and J. Clark. Recursively Generated B-Spline Surfaces On Arbitrary Topological Meshes, CAD, 10(6):350-355, 1978.
2. Doo, H. and Sabin, M. Behavior of Recursive Division Surfaces, CAD, 10(6):356-349, 1978
3. A. Nasri. Curve Interpolation In Recursively Generated B-Spline Surfaces Over Arbitrary Topology, CAGD, 14(1), 1997.
4. A. Nasri. Interpolating Meshes of Curves By Subdivision Surfaces, TR-10/97, American University of Beriut, 1997.
5. A. Nasri. Recursive Subdivision of Polygonal Complexes and Its Applications, in CAGD, submitted for publication.

W. Bradford Paley
Digital Image Design Incorporated
brad@didi.com

Handheld Interactions: Tailoring Interfaces for Single-Purpose Devices

One of the world's largest financial firms, and the clear technology leader on the floor of the New York Stock Exchange currently has a simple application running on a handheld computer (the Casio Cassiopeia). It allows brokers to send digital-ink "looks" (descriptions of market conditions for specific stocks) directly to traders and salespeople. We are completing design and implementation of a huge step beyond this: actual posting and execution of orders.

Orders are directions from traders and salespeople to buy or sell stocks and other financial instruments. The handheld device will receive orders via a wireless link and allow brokers to execute one or more trades that fill the order. This extremely time-critical and human relationship-centered process is currently carried out with hasty scribbles on scraps of paper. Replacing the paper in this scenario is a very difficult task, since paper is concrete, easy to use, carries lots of information in both explicit and implicit forms, and rarely needs rebooting. We have developed innovative new interface widgets combining the benefits of text and object interactions, and synthesized them into a consistent overall design.

This project combines traditional UI design, applied interaction research, and physical interface design similar to industrial design. This sketch outlines our physical/graphic user interface design process, including the following steps: needs analysis, onsite immersive observation, designing with the user, paper and interactive prototypes, pilot release, user testing and observation, and production release. Each step is illustrated with examples from specific projects and physical artifacts.

Head-Mounted Projector for Projection of Virtual Environments on Ubiquitous Object-Oriented Retroreflective Screens in Real Environment

Masahiko Inami
Naoki Kawakami
Dairoku Sekiguchi
Yasuyuki Yanagida
Taro Maeda
Susumu Tachi
Kunihiko Mabuchi
The University of Tokyo
minami@star.t.u-tokyo.ac.jp
www.star.t.u-tokyo.ac.jp/

The head-mounted display (HMD),[1] or head-mounted projector (HMP), and other projection-based virtual reality systems such as the CAVE[2] are typical examples of virtual reality visual displays. Although quite useful, these two kinds of displays have some demerits. The disadvantage of the former is the trade-off of high-resolution and wide field of view. On the other hand, the latter systems impose a shadow of the user's body on the virtual environment and the interaction of the user's virtual body with the user's real body.

Design of an HMP with X'tal Vision

HMP is one of the traditional methods in the field of stereoscopic displays.[3] However, the conventional implementation presents obstacles. The projector's small depth of focus is a serious problem, because it is the depth of focus that determines the limit of the movable area in which the user can observe a focused image on the screen. An additional problem is the uneven brightness of the projected image, because the normal screen diffuses incident rays. Hence, the shape of the screen is restricted to a plane, cylindrical, or spherical surface for easy compensation, or the brightness of the image should be calibrated, which requires complex image computation.

X'tal Vision (Crystal Vision), which uses a projector with a small iris and retroreflective material as its screen, was originally developed for an object-oriented display.[4] With this system, the user observes a projected stereoscopic virtual environment on ubiquitous object-oriented retroreflective screens in the real environment. Because X'tal Vision is a simple but advantageous method, we speculated that applying X'tal Vision in the HMP method would be suitable for virtual/augmented-reality systems.

Figure 1 illustrates the principle of the display. A projector is arranged at the conjugate position of a user's eye, and an image is projected on a screen made of or painted or covered with retroreflective material. A small iris is placed in front of the projector to secure adequate depth of focus.

The retroreflector screen together with the small iris ensures that the user always sees images with accurate occlusion relationships. This means that if a real object has retroreflective material on it, it becomes a part of the virtual environment and disappears, replaced by a virtual object. On the other hand, if it does not have retroreflective material on it, it will occlude the virtual environment without any troublesome shadows on the virtual environment.

In the configuration of X'tal Vision, screen shapes are arbitrary, any shape is possible. This is due to the characteristics of the retroreflector and the small iris in the conjugate optical system. By using the same characteristics of X'tal Vision, binocular stereo vision becomes possible using only one screen with an arbitrary shape. This system should be mounted on the head of the user as an HMP. Two liquid crystal projectors mounted on a helmet project full-color NTSC-resolution images.

Applications

In the first application, we observed the virtual images of a skeleton on the retroreflective screen as if the patient's body had become transparent (Figure 2). The second application involves ubiquitous displays. Any material painted or covered with retroreflective materials can work as a display. We demonstrated a paper-type display (Figure 3). The third application is optical camouflage. We can make any object transparent, which interferes with images of a virtual object (Figure 4). These applications verify and demonstrate the effectiveness of an HMP with X'tal Vision.

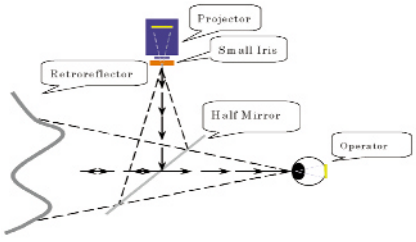


Figure 1
Principle of an X'tal Vision system.

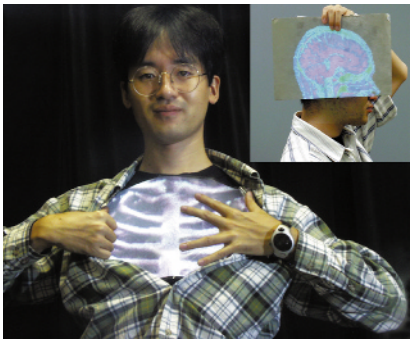


Figure 2
Diagnosis-assisted system.

Figure 3
Prototype of paper-type display.

Figure 4
Example of Optical Camouflage for a real scene.

References

1. Sutherland I. A Head-Mounted Three Dimensional Display, Proc. Fal Joint Computer Conference, AFIPS Conf. Proc. , vol. 33, pp. 757-764, 1968.
2. Carolina Cruz-Neira, Daniel J. Sandin, and Thomas A. DeFanti. Surround-Screen Projection-Based Virtual Reality: The Design and Implementation of the CAVE, Proc. of Computer Graphics, vol. 27, pp. 135-142, August 1993.
3. Ryugo Kijima and Takeo Ojika. Transition Between Environment and Workstation Environment With Projective Head-Mounted Display, Proc. Virtual Reality Annual International Symposium, pp. 130-137, March 1997.
4. Naoki Kawakami, Masahiko Inami, Yasuyuki Yanagida and Susumu Tachi. Object-Oriented Displays, Conference Abstracts and Applications (SIGGRAPH 98), p. 112, July 1998.

The Holodeck Interactive Ray Cache

Gregory Ward Larson
SGI
gwlarson@positron.cs.berkeley.edu

Maryann Simmons
University of California, Berkeley

Rendering with full global illumination enables designers to make intelligent choices regarding the materials, geometry, and lighting of complex objects and spaces. However, typical designers cannot wait all day for a final rendering of a candidate building or car design. But they want to look at their design, and walk around to see it from every angle. Conventional ray tracing doesn't facilitate this, because its view-dependent calculation must start over at each new eye point. A radiosity solution allows free movement, but only in diffuse environments, where there is much less to see. What we need is accurate visualization of realistic environments that is both interactive and progressive, employing parallel processing and graphics hardware to provide the fastest feedback possible to the designer.

Image 1a
An air traffic control tower cab displayed from a precalculated holodeck file (1a, left), then after 30 seconds of progressive ray tracing on 21 processors.

In this sketch, we present a technique for interactive rendering of complex environments with global illumination and arbitrary reflectance functions as provided by the Radiance lighting simulation system[1] and hardware acceleration. Rendering is interactive in the sense that the user is allowed to move freely in and around designated regions while reasonable camera animation is presented, and the image is refined when the user is still. In the context of our system, Radiance serves as the sample generator.

We introduce a 4D holographic ray-caching data structure, called the holodeck due to its similarity in form and function to the Star Trek invention. This structure serves as the rendering target that allows us to quickly load all precomputed ray samples related to a specific view. No ray computations are ever wasted or lost, and dozens of ray tracing processes may be managed in parallel. The process that manages ray computation and holodeck caching is called the server.

We take advantage of graphics acceleration hardware with custom-tailored display drivers, which render a given set of ray samples into a coherent view. The screen representation is generally "2.5D" in the sense that it encodes some depth information for local motion and stereopsis, but maintains unary depth complexity over most of the image. The driver also manages user interaction and drives the progressive rendering.

The sample generator, server, and display driver work together to compute, cache, and display ray samples interactively as the user moves freely about a space.

Image 1b
The Holodeck Interactive Ray Cache.

The air traffic control tower shown in Image 1a was rendered on a 24-processor SGI Onyx2 with IR graphics. In a precalculation that achieved 96-percent linear speedup running 24 ray trace processes over two hours, we produced a 300-Mbyte holodeck file that contains as much detail as shown in Image 1a at every eye point within the interior section, covering most of the control room. Each new view takes three seconds to retrieve and display 100,000 precalculated samples, and given another 30 seconds, improves to the resolution shown in Image 1b. The most important benefit for design applications is accurate prediction of lighting and visibility as provided by our global illumination and tone mapping methods. These features enable the user to make important design decisions regarding tower equipment and window shades, which in this example, could ultimately affect air traffic safety.

Reference

1. Ward, Greg. The RADIANCE Lighting Simulation and Rendering System, Computer Graphics (Proceedings SIGGRAPH 94), ACM, July 1994, pp. 459-472.

Image Moment-Based Stroke Placement

Michio Shiraishi
Yasushi Yamaguchi
University of Tokyo
shira@graco.c.u-tokyo.ac.jp
www.graco.c.u-tokyo.ac.jp/~shira/index.html

Painterly rendering algorithms synthesize images with a hand-crafted touch from a source image of a real scene. The current major scheme is based on Haeberli's method [1,2,3] which paints brush strokes on a canvas successively. In our approach, each rectangular stroke is controlled by several attributes: color $\mathbf{C}$, location (x_c, y_c), orientation θ, width w, and length l.

We propose an alternative method to determine these attributes, so that the stroke nicely approximates a local region of the source image.

All stroke attributes are computed with the following steps.

1. Preparation

Firstly, a tentative location is given for a stroke: Figure1(a). The stroke color $\mathbf{C}$ is sampled from the source image at the location: Figure 1(b). The source image is cropped with a square at the point: Figure1(c). This cropped image $\mathbf{C}_w$ is to be approximated by the stroke.

2. Color Difference Image Generation

A **color difference image** is a gray-scale image, obtained from $\mathbf{C}_w$ and $\mathbf{C}$, Figure 1 (d). Its pixel value $I(x,y)$ is larger if the corresponding $\mathbf{C}_w(x,y)$ is closer to the stroke color $\mathbf{C}$. $I(x,y)$ is given by:

$$I(x,y) = f(d(\mathbf{C}, \mathbf{C}_w(x,y))).$$

Here, $d(\mathbf{C}_1, \mathbf{C}_2) \in R$ is the Cartesian distance of two colors, $\mathbf{C}_1, \mathbf{C}_2$, in the CIE-$L^*u^*v^*$ color space. $f(d) \in [0,1]$ is the function that maps the color distance d to the pixel intensity of the color difference image. In our current implementation, $f(d)$ is:

$$f(d) = \begin{cases} (1 - (d/d_0)^2)^2 & \text{if } d \in [0, d_0] \\ 0 & \text{if } d \in [d_0, \infty) \end{cases},$$

where d_0 is a constant value which indicates the bound of `similar color.'

3. Equivalent Rectangle Calculation

The **image moment** is defined on the gray-scale image, $I(x,y) \in [0,1]$. The image moment of lth degree about the x-axis and mth degree about the y-axis is defined as:[4]

$$M_{lm} = \sum_x \sum_y x^l y^m I(x,y).$$

For $n = l+m$, M_{lm} is called the image moment of nth degree. The binary image of the equivalent rectangle has the same zeroth, first[3] and second moments as the original gray-scale image: Figure 1(e). The stroke attributes, (x_c, y_c), θ, w and l, are determined by the parameters of the equivalent rectangle: Figure 1(f)). The color difference image is approximated by this rectangle, which determines the location, orientation, and size of the stroke.

Results

One of the results is shown in Figure 2. Note that the stroke size varies according to the local complexity of the source image. It means the flat area is painted by larger strokes, while the details are painted with smaller ones.

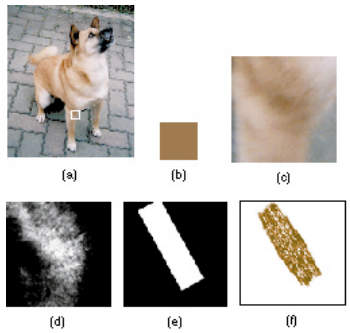


Fig. 1 Steps of algorithm. (a) Source image. The while rectangle indicates the tentative location. (b) Stroke color. (c) Local region of the source image. (d) Color difference image. (e) Equivalent rectangle. (f) Rendered brush stroke.

Fig. 2 Result.

References

1. Paul Haeberli. Paint By Numbers: Abstract Image Representations, SIGGRAPH 90 Conference Proceedings, 1990.
2. Barbara J. Meier. Painterly Rendering for Animation, SIGGRAPH 96 Conference Proceedings, 1996.
3. Peter Litwinowicz. Processing Images and Video for an Impressionist Effect, SIGGRAPH 97 Conference Proceedings, 1997.
4. William T. Freeman, et al. Computer Vision for Interactive Computer Graphics, IEEE Computer Graphics and Applications, Vol. 18, No. 3, 1998.

Shoji Tanaka
Jun Kurumizawa
ATR Media Integration & Communications
Research Labs
gon@mic.atr.co.jp

Seiji Inokuchi
Osaka University

Image Re-Composer

In order to draw or take a good picture, it is important to carefully consider how to compose the picture. If a picture is drawn or taken without an understanding of the composition, the result will be a boring or mediocre picture. There are no absolute rules for such composition, but there is a great deal of understanding of what will occur in terms of visual literacy organization and orchestration. Unfortunately, novice painters or photographers who do not have enough such knowledge usually find it difficult to determine the appropriate composition of a picture currently being drawn or about to be shot.

Re-Composition Galley

Image Re-Composer is a post-production tool that decomposes a picture and regenerates it as a new improved picture. Since the tool allows the user to generate different pictures depending on specified compositions, the user can experiment with a variety of good compositions of the original picture. As a result, the user can learn how masters of the past maintained the visual balance of pictures.

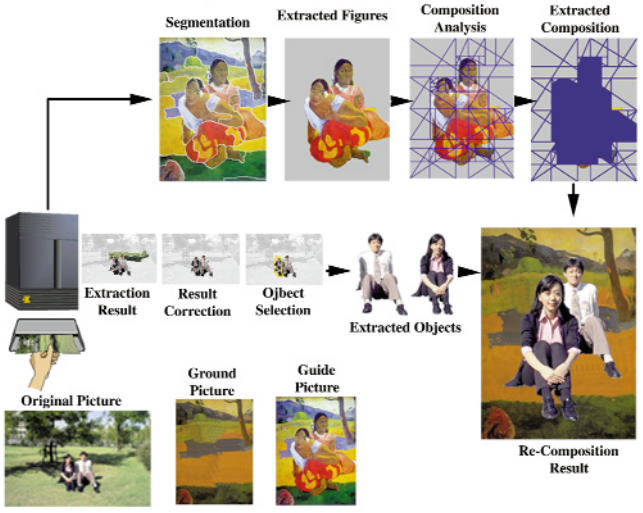


Image Re-Composer

Image Re-Composer consists of three modules:

1. Figure extractor. The figure extraction method extracts figures based on the characteristics of the V4 cortex in the human visual system, which plays an important role in figure-ground segmentation. The figure extractor first segments an input picture into several regions. It then discriminates the segmented regions into portions of the figures and a part of the ground. Finally, it shows the user the extraction results, which include figures in color and the ground in gray. If the user is not satisfied with the results, they can be corrected by selecting or de-selecting regions that are mis-discriminated by the extractor. Then, the user is asked to discriminate each object by grouping the extracted regions.

2. Composition analyzer, which extracts composition information, such as the location, size, and shape of a figure, based on the idea of the golden section. The golden section has been used throughout the history of art to make the most beautiful and ideal proportions of architectures or art work. The Dynamic Symmetry principle has been used to analyze the composition of paintings according to the golden section. This principle can find golden section points in a rectangle whose aspect ratio is the golden proportion. Actually, the aspect ratio of canvas for paintings and printing paper for photographs is the golden proportion. Therefore, applying the Dynamic Symmetry principle to paintings or photos makes it possible to determine how objects in a picture are composed to maintain a good visual balance of the picture. The composition analyzer first applies the Dynamic Symmetry principle to the figures extracted by the extractor. It then composes abstracted figures that are constructed by base lines of the golden section. Finally, it extracts the composition information from the abstracted figures.

3. Composer, which recomposes a picture according to the composition information extracted by the composition analyzer. For the image recomposition, the composer asks the user to input three pictures: an original picture, a ground picture, and a guide picture. The ground picture is a picture onto which the system recomposes figures of the original picture. The guide picture is a picture with composition information, which provides recomposition guidelines for the system. The composer recomposes the user-selected objects in the original picture by adjusting the sizes and locations of the objects according to the guide picture.

Re-Composer

Image-Based Modeling, Rendering, and Lighting in Fiat Lux

Paul Debevec
University of California, Berkeley
debevec@cs.berkeley.edu
www.cs.berkeley.edu/~debevec/FiatLux

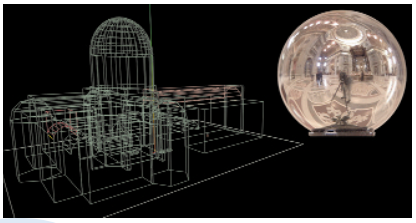
This animation sketch presents how image-based modeling, rendering, and lighting were used to create the animation "Fiat Lux" (SIGGRAPH 99 Electronic Theater). The film features a variety of dynamic objects realistically rendered into real environments, including St. Peter's Basilica in Rome. The geometry, appearance, and illumination of the environments were acquired through digital photography and augmented with the synthetic objects to create the animation.

The Imagery and Story

"Fiat Lux" draws its imagery from the life of Galileo Galilei (1564-1642) and his conflict with the church. When he was 20, Galileo discovered the principle of the pendulum by observing a swinging chandelier while attending mass. This useful timing device quickly set into motion a series of other important scientific discoveries. As the first to observe the sky with a telescope, Galileo made a number of discoveries supporting the Copernican theory of the solar system. As this conflicted with Church doctrine, an elderly Galielo was summoned to Rome, where he was tried, convicted, forced to recant, and sentenced to house arrest for life. Though honorably buried in Florence, Galileo was not formally exonerated by the church until 1992. "Fiat Lux" presents an abstract interpretation of this story using artifacts and environments from science and religion.

The Technology

The objects in "Fiat Lux" are synthetic, but the environments and the lighting are real. The renderings are a computed simulation of what the scenes would actually look like with the synthetic objects added to the real environments. The techniques represent an alternative to traditional compositing methods, in which the lighting on the objects is specified manually.



The environments were acquired in Florence and Rome. The images in St. Peter's were taken within the span of an hour in accordance with our permissions. To record the full range of illumination, we used high-dynamic-range photography, in which a series of exposures with varying shutter speeds is combined into a single linear-response radiance image. Several scenes exhibited a dynamic range of over 100,000:1.

The appearance and illumination of each environment was recorded with a set of panoramic images and light-probe measurements. Each light-probe measurement was made by taking one or two telephoto radiance images of a two-inch mirrored ball placed on a tripod; each provided an omnidirectional illumination measurement at a particular point in space. Several radiance images were retouched using a special high-dynamic-range editing procedure and specially processed to diminish glare.

We constructed a basic 3D model of each environment using the Façade photogrammetric modeling system. The models allowed us to create virtual 3D camera moves using projective texture mapping and fix the origin of the captured illumination. The light probe images were used to create light sources of the correct intensity and location, thus replicating the illumination for each environment. The illumination was used to "un-light" the ground in each scene, allowing the synthetic objects to cast shadows and appear in reflections. The dynamic objects were animated either procedurally or by using the dynamic simulator in Maya 1.0. Renderings were created on a cluster of workstations using Greg Larson's RADIANCE global illumination system to simulate the photometric interaction of the objects and the environments. The final look of the film was achieved using a combination of blur, flare, and vignetting filters applied to the high-dynamic-range renderings.

Supported by Interval Research Corporation, the Digital Media Innovation program, and the Berkeley Millennium project.

Contributors: Christine Cheng, H.P. Duiker, Tal Garfinkel, Tim Hawkins, Jenny Huang, and Westley Sarokin

References

1. Paul E. Debevec, Camillo J. Taylor, and Jitendra Malik. Modeling and Rendering Architecture from Photographs, SIGGRAPH 96, August 1996.
2. Paul Debevec and Jitendra Malik. Recovering High Dynamic Range Radiance Maps from Photographs, SIGGRAPH 97, August 1997.
3. Paul Debevec. Rendering Synthetic Objects Into Real Scenes: Bridging Traditional and Image-Based Graphics with Global Illumination and Dynamic Range Photography, SIGGRAPH 98, July 1998.

Rod G. Bogart
Industrial Light & Magic

Image-Based Techniques for Object Removal

In feature film effects, it is often necessary to remove an object from the background plate. The obvious solution is to remove the object from the actual set, and to shoot a second camera pass, called a clean plate. However, this is often not done or not possible. The camera move may not be repeatable, or the set may have been destroyed with the first take. This sketch describes a simple image-based rendering technique for producing a clean plate from the original background.

The algorithm consists of four steps: object identification, image depth extraction, image projection, and incremental rendering. In a full system, each of these portions can be rather complicated, but they all can be approximated through simpler methods to get the job done. Simply put, the algorithm makes a hole where the bad pixels are, then uses neighboring frames to incrementally fill the hole with good pixels.

The object identification phase can be as easy as a hand rotoscoped matte or as complicated as an optical flow system. The result needs to be a matte for the object to be removed. With these mattes, a hole is produced in each frame in the sequence, so that only background pixels remain.

In a complete 3D solution, the image depth would be determined for each pixel so that it can be projected into a different camera space in the next step. In the simple example shown, however, the depth can be approximated by assuming the ground is a flat plane. This also allows the following projection step to be performed as a 2D warp. In either case, the requirement is to create enough information to allow a future or past frame to be warped to the view of the current frame.

Image projection and incremental filling are done at the same time. For every frame to be cleaned, a frame is chosen that is incrementally far away in time. For this implementation, it was simply frame-1, frame+1, frame-2, frame+2, etc. Each of these frames is warped into the camera view of the current frame. In the full 3D solution, this involves taking the screen coordinates and known depths from the other camera, projecting into world space, then projecting back through the current camera to produce each pixel to render. In the simple 2D case, this is just a 2D-perspective matrix that can be determined from corresponding points in the image. Both methods will create a warped frame with a hole in it. But this hole will not correspond exactly to the hole that is being filled. This means valid pixels can be filled in, by comping the warped frame under the current frame. Future and past frames are accumulated in this manner until the hole is filled, or until some temporal threshold is reached.

In many cases, the object being removed was serving as a stand-in for a CG character, so the hole does not need to be entirely filled in. Some objects need to be removed completely, such as cables and rigs, and these narrow pieces usually fill in quickly. Note that there are antialiasing and lighting issues with this method, but this technique provides a simple way to obscure unwanted foreground material with very similar pixels in the desired background.

Interactive CSG

Chris Butcher
University of Otago
cbutcher@cs.otago.ac.nz

Constructive Solid Geometry has many features that make it attractive for conceptual design (for example, hierarchical modeling and intuitive operations). Existing techniques for interactive CSG[1] update the geometry of the entire model for every object manipulation. For larger models, such techniques cannot achieve the responsiveness required for interactive work. We use a space-subdivision technique to reduce the amount of computation by recomputing only those parts of the model that have changed since the last update.

Every primitive is polygonized in order to take advantage of graphics display hardware. This is done by dividing each primitive's surface up into a number of rectangular areas of space. We trace rays in a 3D grid inside these areas and find all intersection points of the primitive's algebraic surface with the grid. These points are polygonized by an implicit surface tiler[2] to generate a number of triangles inside every grid cell.

The modeling workspace is subdivided into an octree, typically three to four levels deep. Each node in the CSG graph contains an octree data structure that stores empty/full information. At the bottom level of the octree, if the object's surface is present, a "leaf node" stores a reference to every grid that intersects that area of space. Octrees that represent a CSG combination will reference grids that belong to a number of different primitives.

When a primitive object is created or moved, its octree is found by testing each section of the octree against the algebraic function. The octree is subdivided down to the bottom level, where each leaf node tests to see which grids overlap its area. References to these grids are stored, and if there are any changes from the previous octree then a "changed" flag is set and propagated up the octree structure, so that higher-level nodes are changed if any of their children are.

To perform a CSG operation between two objects, we traverse the changed sections of the octrees, looking for grids that overlap each other. For each cell in each grid, we find which cells it overlaps in each of the opposing grids that overlap its section of the octree (see Figure 1).

We use a triangle-splitting algorithm based on Laidlaw et al[1] to split the triangles in a cell against the opposing triangles, determining for each split triangle whether it lies inside or outside the opposing object. The CSG operation determines which triangle subset is inserted into the combined octree.

The split operation is O(n*m) in the number of triangles, but the octree division of the workspace reduces the number of grids that must be tested, and the uniform subdivision of each grid reduces the number of cells whose triangles must be split against. This allows us to keep n and m down to 2-10 triangles in almost all cases, although a few cells that lie on corners of the model may have up to 40 triangles.

Complex cutting operations may be performed on models such as Figure 2 with approximately five updates per second (Pentium II - 450 MHz). This lets us build CSG objects interactively, using a direct-manipulation interface.

This system meets many of the requirements of conceptual design; it allows rapid prototyping of objects and gives immediate feedback on changes, while retaining the features that make CSG attractive for solid modeling.

www.cs.otago.ac.nz/postgrads/cbutcher/s99.

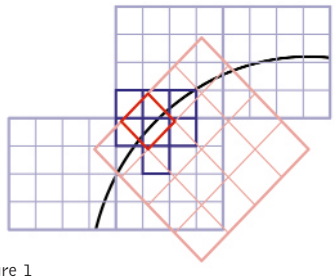


Figure 1
Each grid cell tests every overlapping grid to find cells that it overlaps. Triangles in the red grid cell are split against triangles from all the highlighted blue grid cells.

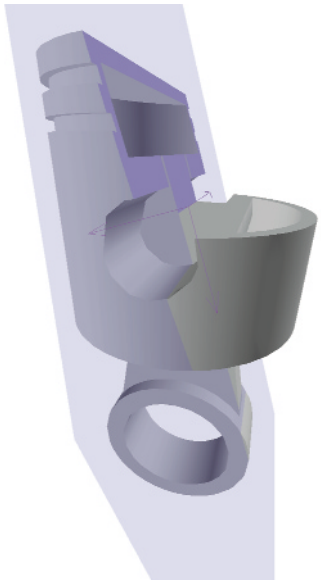


Figure 2
The cutting plane may be moved through this model in real-time, using the direct manipulation device. Approximately five frames per second are achieved during the manipulation.

References

1. D. Laidlaw, W. Trumbore and J. Hughes. Constructive Solid Geometry for Polyhedral Objects, Computer Graphics, Proceedings of SIGGRAPH 86, pages 161-170, August 1986.
2. P. Ning and J. Bloomenthal. An Evaluation of Implicit Surface Tilers, IEEE Computer Graphics and Applications, 13(6):33-41, Nov 1993.
3. A. Rappoport and S. Spitz. Interactive Boolean Operations for Conceptual Design of 3D Solids, Computer Graphics Proceedings, SIGGRAPH 97, pp. 269-278, August 1997.

M. Brady
K. Jung
H.T. Nguyen
Intel Corporation
brady@allover.com

R. Mullick
National Institutes of Health

W. Lawton
T. Poston
S. R. Ranjan
K. Schulz
S. Venkataraman
R. Viswanathan
Y. Yu
G. Zhu
National University of Singapore

R. Raghavan
John Hopkins University

Interactive Haptic Modeling of Tensegrities and Network Structures

We address handling deformable objects in the context of a reach-in environment[1] for dextrous work with virtual objects and tools. The user's interaction with the object is two-handed: the "workpiece" hand (commonly the left) moves the object rigidly in space with a tool (a stylus whose position and orientation are continuously monitored). The "detail" hand's stylus is part of a haptic device used to probe the structure and observe the response (visually and haptically).

Although our methods generalize readily, we limit ourselves here to tensegrity structures. They have many uses, in areas such as architecture, civil engineering, and biology, where they have been considered as simplified models of cell structure.[2] In addition, they are useful benchmarks for strategies for fast computation and haptic feedback. We have applied these techniques in the larger context of models of cell mechanics. The methods generalize directly to models of tissues, organs, or other deformable structures, and we anticipate a wide range of uses in the near future in medicine and education. The combination of working with full hand-eye coordination and dexterity in a 3D environment, and visualization and haptics, is very compelling for humans as tool users.

A tensegrity is a 3D framework of rigid rods and "rubber bands" attached at nodes. Consider a node, m, picked and displaced with the detail hand. On its initial selection, we equilibrate with the constraints that this node and one base rod are fixed. After this, we calculate a 3x3 linear-response matrix A where $f = A\Delta$, with f being the incremental force in response to the displacement Δ of node m. From this, we can calculate the change in displacement and orientation of all of the rods in response to the applied incremental displacement Δ. Displacing a rod (in both position, x, and orientation, ø is handled similarly, except that the input is now a 6D vector, so linear-response matrices are 6x6.

For a large structure, the above computation exceeds the one-millisecond delay allowed for smooth haptic interaction. We therefore apply two approximation techniques:

1. For small deformations, we compute and display the actual movement of the structure (so the result is visually correct) but compute the force felt with a constant linear response matrix A. This process may be refined by updating the linear-response matrices at reasonable intervals as the stylus is being displaced.

2. For larger deformations, we use a 3x3x3 grid of points in space for each node to represent a discretization of the displacements of the node in space. The effective linear-response function is computed for the 27 values for each node. From the actual displacement of the node, an interpolated force is provided. This large precomputation provides a fairly accurate response.

A different technique gives a feel for the static structure of the object. We implement a force field, rather than computing surface contacts (which our linear elements lack).[3] If the stylus tip x is within a 3D rectangular bounding box around the center line of a rod or string, we provide an attractive force toward the nearest point x' on the centerline. Its form is arbitrary, but we use $f = -a(d^2 - |x-x'|^2)(|x-x'|)$, where $a > 0$ is an adjustable constant, and $2d$ is the width of the bounding box. Where two rods or strings join, we interpolate between their centerlines.

In conclusion, we have computed the response of structures as accurately as compatible with haptic response (action and reaction are "instantaneous"), which is the fastest refresh rate required (every millisecond, compared with the visual, which needs refresh only every 30 milliseconds or so) for preserving the feeling of continuity that we have about the real world. We have done so in a fully integrated 3D virtual world for dextrous tool use.

References

1. T. Poston. The Virtual Workbench: A Path to Use of VR, CieMed Technical Report, Sept. 1993. Reprinted with added references: The Industrial Electronics Handbook, J. D. Irwin ed., CRC Press and IEEE Press, pp. 1390-3, 1997.
2. D. Ingber. The Architecture of Life, Sci. Amer., Jan. 1998.
3. R. W. Lawton, R. Raghavan, S. R. Ranjan, and R. R. Viswanathan. Tubes In Tubes: Catheter Navigation in Blood Vessels and Its Applications, Intl. J. of Solids and Structures, to appear.

Interactive Rendering with Arbitrary BRDFs Using Separable Approximations

Jan Kautz
Michael D. McCool
University of Waterloo
jnkautz@cgl.uwaterloo.ca
www.cgl.uwaterloo.ca/Projects/rendering/Papers/

Our aim is to render surfaces lit with point-light sources at interactive rates using arbitrary BRDFs, evaluated at per-pixel resolution on contemporary hardware. Neglecting surface position and wavelength, BRDFs are parameterized by at least four degrees of freedom.

Separable decompositions[1,4] approximate (to arbitrary precision) a high-dimensional function f using a sum of products of lower-dimensional functions g_k and h_k:

$$f(x,y,z,w) \approx \sum_{k=1}^{n} g_k(x,y)h_k(z,w).$$

Under certain changes of variables many BRDFs are highly separable[4] and even N=1 has proven to be visually adequate for many interesting BRDFs (see Figure 1). Because of the simplicity of the reconstruction process, we can perform the reconstruction at interactive rates using existing hardware support for texturing, compositing, and diffuse lighting.

Parameterization

The $\hat{\omega}_i \times \hat{\omega}_o$ parameterization is suitable for some kinds of BRDFs, but for common "glossy" BRDFs, we reparameterize them in terms of the halfvector $\hat{h}$ and a vector $\hat{d}$, which is $\hat{\omega}_i$ represented relative to a new basis $\{\hat{t}', \hat{b}', \hat{h}\}$ that is created by applying Gram-Schmidt orthonormalization to the tangent $\hat{t}$, the binormal $\hat{b}$, and $\hat{h}$. This parameterization aligns features of the BRDF to make it more separable.

Decomposition

A BRDF can be decomposed using the Singular Value Decomposition[1,3] or using the Normalized Decomposition (ND) algorithm, which computes only a single-term approximation

$$f_P(\hat{h}, \hat{d}) \approx \tilde{f}_P(\hat{h},\hat{d}) = g_1(\hat{h})h_1(\hat{d}).$$

The ND algorithm finds the average normalized profile along $\hat{d}$ and stores it in $h_1(d)$ and then stores the normalization factors in $g_1(\hat{h})$ (using the p-norm):

$$g_1(\hat{h}) = \left(\int_D |f_P|^p(\hat{h},\hat{d})\,d\hat{d}\right)^{\frac{1}{p}},$$

$$h_1(\hat{d}) = \frac{1}{\int_H \frac{f_P(\hat{h},\hat{d})}{g_1(\hat{h})}\,d\hat{h}}.$$

Both functions $g_1(\hat{h})$ and $h_1(\hat{d})$ are two-dimensional and can be put into hemispherical or parabolic[2] texture maps for proper interpolation of unit vectors $\hat{h}$ and $\hat{d}$.

Rendering

First, render the scene diffusely illuminated using $g_1(\hat{h})$ as a multiplicative texture map. Appropriate texture coordinates must be calculated at the vertices according to the parameterization of the BRDF. Without clearing the colour or depth buffers, render the scene again with no illumination and with $h_1(\hat{d})$ as a decalled texture map. For the second pass, set the depth test to "equality" and set the compositing operation to "multiply." This evaluates the following rendering equation (for one point light source) for all $\underline{x}$:

$$L_o(\hat{\omega}_o, \underline{x}) = \tilde{f}(\hat{\omega}_o, \hat{\omega}_i)\cos\Theta_i \frac{I}{r_i^2}$$

If multitexturing support is available, this can be done in a single pass.

Figure 1: Left to right: HTSG copper, Ward's anisotropic model, and measured grey vinyl. Top: Models with BRDFs evaluated at every pixel in a raytracer. Bottom: Hardware-accelerated single-term approximations using OpenGL.

References

1. A. Fournier. Separating Reflection Functions For Linear Radiosity, in Eurographics Rendering Workshop, pages 383-392, June 1995.
2. W. Heidrich and H. P. Seidel. Realistic, Hardware-Accelerated Shading and Lighting, In Proc. SIGGRAPH, August 1999.
3. J. Kautz. Hardware Rendering With Bidirectional Reflectances, Technical Report TR-99-02, Dept. Comp. Sci., U. of Waterloo, 1999.
4. S. Rusinkiewicz. A New Change of Variables For Efficient BRDF Representation, in Eurographics Workshop on Rendering, pages 11-23, June 1998.

Rebecca Xiong
Eric Brittain
Massachusetts Institute of Technology
becca@media.mit.edu
graphics.lcs.mit.edu/liveweb/

LiveWeb: Visualizing Live User Activities on the Web

Currently, Web users have little knowledge about the activities of fellow users. They cannot see the flow of online crowds or identify centers of online activity. LiveWeb seeks to enrich Web users' experience by visualizing the real-time activities of other users. The visualizations can help answer user questions about overall patterns and specifics such as: What are other people looking at? What is hot? Who is interested in what I am interested in?

LiveWeb visualizes two kinds of data about a Web site: its underlying structure and real-time user accesses. The visualizations first lay out the site structure, then overlay access data on top. The visual structure of a Web site can be generated in three ways:

1. It can be custom made.
2. It can be automatically generated.
3. It can emerge from users' pattern of traversal.

We have created two sample visualizations, WebMap and WebFan, using the first two methods. WebMap visualizes Website activities that can be mapped to physical structures. The custom-created layout is derived from a floor plan as shown in Figure 1. Each Web user is represented by a dot. As users move among pages, their dots move correspondingly through the WebMap. Other features include:

- Distinguishing users from different domains by color.
- Showing WebMap user's own traversal in context of others'.
- Indicating how long a user has been on a page and what other pages the user has visited.
- Accumulating user accesses over time to identify Web pages that are visited more often.
- Allowing users to navigate directly using WebMap.

WebFan visualizes Web activities using a fan-like hierarchical structure. This abstract structure allows a large set of Web pages with multiple levels to be represented at the same time for overview and comparison. Users can also interactively explore the fan structure to find out more about individual pages.

Figure 2 shows a WebFan visualization of a Web-based message board, which contains user postings and replies that each reside on a separate Web page. WebFan can also be used to visualize a generic Web site by first extracting its hierarchical structure using a Web crawler.

The authors would like to thank Judith Donath for her helpful discussions.

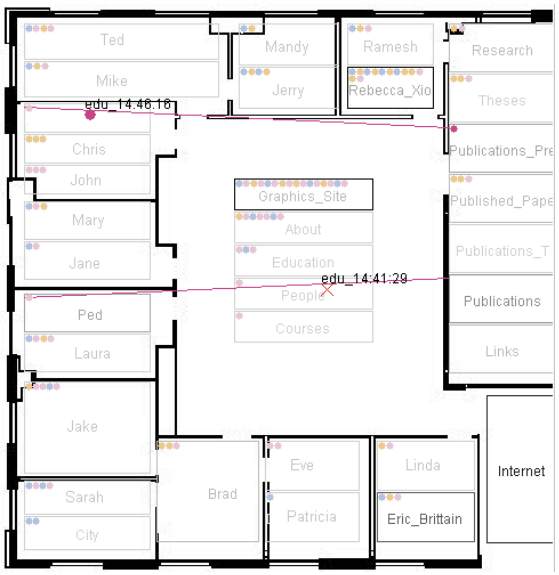


Figure 1
Web Map visualizes the Web site for a research group. Each office contains one or more rectangular areas that represent group or individual members' home pages. Each visitor to a Web page is represented by a dot in the corresponding rectangular area.

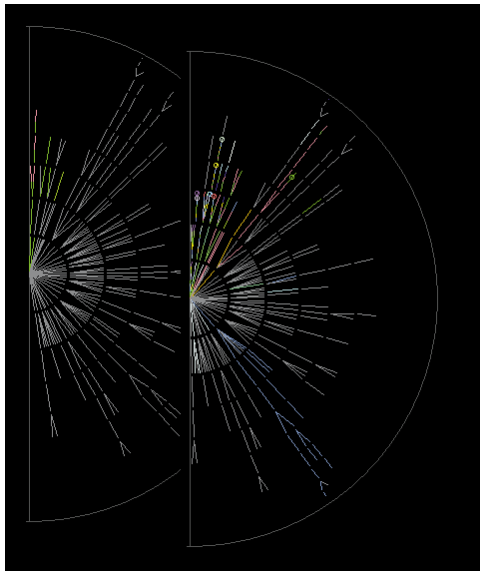


Figure 2
WebFan visualizes activities on a Web-based message board. It uses the message-reply structure to lay out the message in a fan-like fashion. Each time a Web visitor reads a message, the line segment representing the message will change to the visitor's color. A circle on the colored section indicates that the user is currently viewing the page.

Methods for Preventing Cloth Self-Intersection

John McDonald
DePaul University
johncmcdonald@hotmail.com

Robert Welland
Northwestern University

Cloth modeling has long been an active subject in computer simulation and animation. Diverse industries such as clothing and textiles, as well as film and animation, have all contributed to this growing body of knowledge. But realistic and stable computation of cloth self-intersection remains one of the challenges.

Since the early 1990s, animators have used large particle systems or mass-spring lattices to model the behavior of cloth.[2,3,4] Whether the animations use explicit integration techniques such as Runge-Kutta or implicit techniques such as backward Euler,[1] the models are similar.

Many techniques for handling self-intersections calculate when a vertex has penetrated a triangle and then either adjust the velocity of the vertex with a rigid collision or introduce a stiff spring force to correct the penetration. To eliminate self-intersections completely, such algorithms must handle edge-edge collisions similarly, and track vertex-triangle orientations to ensure surface integrity.[5]

We propose an alternative approach. The triangular mesh in these simulations is only an approximation to the surface itself. Our forces use a different approximation that allows us to handle collisions and friction. Cloth-cloth interactions are highly complex. Colliding patches of cloth push on and slide against each other. Our method prevents self-intersections and reacts in a realistic manner as the cloth collides with itself. Our approach does not sacrifice the visual integrity of the simulation and yields realistic self-collisions in cloth.

The Model

Consider a sheet of material whose physical properties are constant along its surface. We decompose the sheet into a triangular mesh of masses and damped springs. We chose a hexagonal decomposition to avoid introducing features inherent to square grids. To introduce stiffness to the system, both for bending and twisting, we attach damped springs that span the center vertex of each hexagon.

For self-intersections, we approximate the fabric by a lattice of patches centered at each vertex v. At v, we place a repulsive force with limited range, which can apply to every other particle in the fabric except v's neighbors. This places spheres about all vertices, which collectively cover the sheet. While these forces tend to make the sheet a bit thick for coarse grids, they work exceptionally well for fine grids (> 30 x 30). We are currently working on other force configurations that will yield thinner fabric for coarse grids. Using a hierarchical decomposition of the fabric can dramatically improve the efficiency of the algorithm.

We describe the various force laws we used for repulsion, including a linear spring force and a Lennard-Jones-type 2/4 potential law, each of which we modify by cutting off their attractive components. In addition, we describe our use of adaptive step sizes to counter instabilities in the system.

Each of these force laws has its own benefits and disadvantages. The stiff spring force is simple, and its stability is easy to calculate and predict. On the other hand, a vertex with enough energy can overcome a stiff spring. In such an event, we are forced either to re-compute the entire animation with a new spring constant and step size, or to introduce a barrier beyond which the particle cannot pass (a sphere within the force sphere). In either case, we get a repulsive force for friction calculations.

A Lennard-Jones type force law gives us a repulsive force that more accurately models two pieces of cloth in contact. The force gives a little upon initial contact and then rapidly increases until the pieces of cloth can come no closer. The Lennard-Jones 6/12 potential law yields just such a force, but we found it to be too unstable for our purposes and decided to work with a similar force whose repulsive term is not quite as extreme as d -13. We counter any remaining instability with an adaptive integration step size. This approach is best suited to situations where cloth-cloth collisions are rare, as when modeling clothing or cloth wrapped around stationary objects.

References

1. D. Baraff and A. Witkin. Large Steps in Cloth Simulation, Computer Graphics Proceedings, Annual Conference Series, pp. 137-146, 1996.
2. D. Hutchinson, M. Preston, W.T. Hewitt. Adaptive Refinement for Mass/Spring Simulations, Proceedings of Seventh Eurographics Workshop on Animation and Simulation, Poitiers, 1996.
3. Xavier Provot. Deformation Constraints in a Mass-Spring Model to Describe Rigid Cloth Behavior, Graphics Interface '97, pp. 147-155, 1997.
4. R. Szeliski and D. Tonnesen. Surface Modeling with Oriented Particle Systems, Computer Graphics (Proc. SIGGRAPH '92), pp. 185-194, July 1992.
5. P. Bolino, M. Courchesne, N. Magnenat-Thalmann. Versatile and Efficient Techniques for Simulating Cloth and Other Deformable Objects, Computer Graphics (Proc. SIGGRAPH '95), pp. 137-144. 1995.

Teresa Larsen
David Goodsell
The Scripps Research Institute
Tlarsenphd@home.com
www.scripps.edu/pub/olson-web/people/larsen/HIVmodel.htm

Dru Clark
Michael Bailey
San Diego Supercomputer Center

Ernest Stewart
Matrix Enterprises

Modeling HIV

The digital model (center) provided the geometry to manufacture the physical components. Clockwise from lower left: the laminate block before liberating pieces, the individual pieces before coating and sanding, coated parts, and the laminated model, painted.

In the effort to conquer AIDS, scientists have collected more data about the Human Immunodeficiency Virus (HIV) than any other organism. Yet textbooks, journal publications, and other educational materials still use schematic representations to illustrate HIV.

In 1995, collaborating with David S. Goodsell, using in-house tools of Arthur Olson's Molecular Graphics Lab at The Scripps Research Institute, and employing sophisticated graphics tools at the Advanced Scientific Visualization Laboratory at the San Diego Supercomputer Center, Teresa Larsen and David Goodsell published a scientifically accurate model of HIV based on the data available in the literature. It incorporated the known structural features and used color, transparency, and texture mapping to illustrate the details of the model known at the time. The model renounces the traditional simplification of molecular components customarily used in schematic illustrations, such as spiral ribbons to represent the RNA and spheres for reverse transcriptase (RT) and other components. The renderings provide contextual relationships between the virus and its host in vivo and provide a visual basis for explanations of its life cycle and functional mechanisms.

Since 1995, a considerable amount of new data has come to light, and the sketch has been updated. We have now created an accurate physical model of HIV at a scale of one million times actual size. We imported each virtual component of the model as a single geometry file into the SDSC Tele-Manufacturing Facility's Laminated Object Manufacturer (LOM). As the LOM machine operates much like a pen-plotting device, proper interpretation of the 3D data requires that each piece consist of a contiguous surface of polygons. Any flaws or irregularities in the geometry can cause unpredictable results as the laser carves out each layer of the laminate to the geometry specifications.

We can now put this physical model into a researcher's or a student's hands so they can assemble and disassemble its various components and learn their structural interrelationships. The laminated model from the LOM machine, as well as digital versions of the geometry, serve as originals from which a manufacturer can create molds for mass producing the pieces. In collaboration with Matrix Enterprises, we have learned some practical limitations to maintaining the level of accuracy in the physical model portrayed in the virtual model. Consequently, the manufacturing engineers at Matrix Enterprises have provided guidance in tailoring the components to the particular requirements of the molding process.

In this physical model, we have chosen textures and colors of polymer materials to distinguish structural components. In each case, the choices also reflect durability and assembly criteria.

HIV

The Morphological Cross-Dissolve

Kevin Novins
University of Otago
novins@cs.otago.ac.nz

James Arvo
California Institute of Technology
arvo@cs.caltech.edu

We define a new operator, called the morphological cross-dissolve, that blends binary or grayscale images using mathematical image morphology.[2] The operator gradually changes shapes in the source image until they match corresponding shapes in the target image. In the case of gray-scale images, the morphological cross-dissolve also changes gray levels while distorting shapes and achieves more natural transitions than traditional cross-dissolves. The new operator is completely automatic, requiring no user-supplied feature lines or correspondences.

To illustrate, consider the problem of gradually transforming one binary image into another, as shown in the top row of Figure 2. Each point x that is "on" in the source image but "off" in the target image must be turned off at some step in the transformation and vice-versa. By appropriately timing each transition, we create the illusion of a continuously moving boundary between the shapes.

In Figure 1, the solid and dashed curves represent the boundaries of source and target shapes, respectively; here, point x will eventually be turned off, and point y will eventually be turned on. The morphological cross-dissolve times the transition of each point using the distance to the source boundary and the distance to the target boundary. In particular, the transition time is the ratio of the former to the sum of the distances.

For discrete images, distance information is readily obtained from the *morphological dilation* operator.[2] The distance from the boundary of a set S to a point $x \notin S$ is the minimum number of dilations needed for S to reach x. The distance from the boundary of S to a point $x \in S$ is the minimum number of dilations needed for the complement of S to reach x.

So that expanding boundaries do not pass through all gaps between the shapes, distances in the expanding or contracting regions are constrained to lie within those regions. When the source and target shapes do not overlap, the transitions are timed so that the source shapes gradually erode away while the target shapes simultaneously undergo the reverse process.

To extend the technique to N-level gray-scale images, we partition the image G into N-1 binary level sets G_i, where $G_i(x)$ is "on" if and only if $G(x)>i$. We compute morphological cross-dissolves for each level set independently and recombine them.

Figure 2 compares a morphological cross-dissolve to a standard cross-dissolve. While both methods are fully automatic, only the former shows smooth transitions of boundaries rather than fades. For image pairs in which corresponding features overlap when superimposed, the results are comparable to feature-based morphs.[1]

www.cs.caltech.edu/~arvo/mdissolve

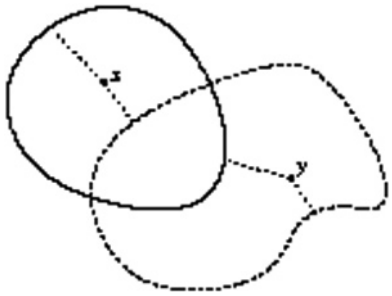


Figure 1
The geometry of a morphological cross-dissolve.

Figure 2
(top) Several steps in the morphological cross-dissolve of two binary images. (middle) A morphological cross-dissolve of two gray-scale images. (bottom) An ordinary cross-dissolve of the same images. Differences are most evident on the left side of the face and the hair line. (Images from the database of faces, AT&T Laboratories, Cambridge.)

References

1. Thaddeus Beier and Shawn Neely. Feature-Based Image Metamorphosis, Computer Graphics, 26(2):35-42, July 1992.
2. J. Serra. Image Analysis and Mathematical Morphology, Academic Press, New York, 1982.

Michael Patrick Johnson
Massachusetts Institute of Technology
aries@media.mit.edu

Multi-Dimensional Quaternion Interpolation

Motivation

Rose, Bodenheimer, and Cohen describe a promising method for reusing a small number of example animations in an interactive system.[1] They use multi-dimensional interpolation of orientations using radial basis functions (RBFs) on an Euler angle parameterization. Our work is similar to this work, but our system uses quaternions as the representation of orientation, avoiding the known problems with interpolating Euler angles. Our goal was to discover a multi-variate version of Shoemaker's famous "slerp" function. (See Watt & Watt[2] for an introduction to quaternions and quaternion interpolation.)

Euclidean Multidimensional Interpolation

The problem of multivariate interpolation in Euclidean space consists of finding a continuous, smooth function $F(\mathbf{x}) : \mathbb{R}^n \rightarrow \mathbb{R}^m$ from a set of example datapoints $\{(\mathbf{x}_i, \mathbf{y}_i)\}$ that the function must pass through (formally, $F(x_i) = \mathbf{y}_i$). RBFs are one technique for finding this function.

Multidimensional Quaternion Interpolation

We explored two methods for applying existing Euclidean multi-variate interpolation techniques to the (non-Euclidean) quaternion manifold: extrinsic and intrinsic. Our input examples will be of the form $\{x_i \in \mathbb{R}^n\}$ and our output (quaternion) examples of the form $\{\mathbf{q}_i \in \mathbb{Q}\}$ where $\mathbb{Q}$ is the unit quaternion group. As with *slerp,* the examples must all lie on the same (local) hemisphere of $\mathbb{Q}$. A simple preprocessing step is used to enforce this.

Euclidean Reprojection

A simple trick is to interpolate the examples as if they were Euclidean points in $\mathbb{R}^4$, then project the result on $\mathbb{Q}$ (by renormalizing). The problem with this method is that a constant speed change in the input parameters will not produce a constant speed interpolation of the reprojected points, which was the motivation for *slerp* originally.

Tangent Space Interpolation

Quaternions have an exponential form: $\mathbf{q} = [\cos(\theta/2), \sin(\theta/2)\hat{\mathbf{n}}] = e^{\frac{\theta}{2}\hat{\mathbf{n}}}$ where $\mathbf{q}$ represents a rotation of θ around axis $\hat{\mathbf{n}}$. Therefore, $\ln \mathbf{q} = \frac{\theta}{2}\hat{\mathbf{n}}$. The log of a quaternion lies in the tangent space anchored at the identity element. This tangent space is $\mathbb{R}^3$, a Euclidean vector space. Additionally, the ln of our examples will also lie in a local hemisphere of the tangent space, within the solid ball of radius π. (Note that tangent points need to be within $\pi/2$ of each other, however, since antipodal points on the *surface* of this ball are identified.)

We create our interpolation function F from the tangent space description (ln) of the examples. We interpolate in this space and then "lift" the interpolated tangent vector back into $\mathbb{Q}$ by exponentiating. Explicitly, the interpolated vector is $\mathbf{q}(x) = e^{F(x_i\{(\ln x_i, \mathbf{q}_i)\})}$.

This function clearly interpolates examples, and the author has proven that it reduces to slerp in the case of one input parameter and two examples, as desired. A more technical description of this technique is available on the author's Web site.[3]

Results and Future Work

We have implemented both of these methods (using RBFs) to continuously interpolate between several kinematic animation examples that differ along emotional input axes, such as happiness and fatigue. The extrinsic version was incorporated in the Swamped! interactive exhibit at SIGGRAPH 98, and the intrinsic version is used in the Millennium Motel (SIGGRAPH 99). Future work will add quaternion-based inverse kinematics and constraints to this animation technique.

References

1. Rose, C., Bodenheimer, B., Cohen. M. Verbs and Adverbs: Multidimensional Motion Interpolation, IEEE Computer Graphics and Applications, Sep/Oct 1998.
2. Watt, A. and Watt, M. Advanced Animation and Rendering Techniques. ACM Press. 1992. pp. 356-368.
3. www.media.mit.edu/~aries/quats/quats.html

Multifluid Finite Volume Navier-Stokes Solutions for Realistic Fluid Animation

John A. Turner
Andre C. Mazzone
Blue Sky Studios, Inc.

It is commonly believed that solution of the full 3D Navier-Stokes equations is too computationally intensive for computer graphics applications. Previous approaches have typically used either the simplified shallow-water approximations[2] or, most recently, two-stage approaches involving a low-resolution 3D Navier-Stokes solution followed by a height-field solution.[1] However, recent advances in incompressible free-surface-flow algorithms and methods for the solution of linear systems of equations make high-resolution solution of the full 3D Navier-Stokes equations possible.

A computational tool, referred to here as MIZU* has been developed at Los Alamos National Laboratory for simulation of casting processes (filling, cooling, and solidification of molten fluid in molds with complex geometry**). Such simulation involves modeling physical phenomena such as unsteady, incompressible, or slightly compressible flow of multiple, immiscible fluids; interface physics (for example, surface tension); convective, diffusive, and radiative heat transfer; solidification of multicomponent alloy systems; microstructural physics (for example, nucleation, dendrite growth); and material response effects (for example, stress, distortion, shrinkage, plastic flow).

MIZU solves the 3D, incompressible, variable-density Navier-Stokes equations on generalized-connectivity unstructured (GU) meshes. The use of GU meshes, which can contain hexahedra, tetrahedra, prisms, and pyramids, enables simulation of arbitrarily complex geometry, which is crucial for both mold filling and computer-graphics applications. Note that these are volumetric meshes, so the availability of tools for generating high-quality GU meshes from surface meshes created by CAD software and modeling tools such as Maya and Softimage is essential.

The flow algorithm is a 3D extension of recent advances in projection methods[4] and is based on a colocated, cell-centered, finite volume formulation that is second-order in both time and space.*** The algorithm is implicit, so it requires the solution of linear systems of equations at each timestep. The solver library used[5] makes use of recent advances in iterative-solution techniques, specifically preconditioned Krylov subspace methods.

Since the flow involves multiple materials, a critical aspect of the simulation involves tracking the interface between materials. Note that the flow of wine into a glass is a multimaterial problem (wine and air) and that material properties such as density vary by several orders of magnitude over an extremely small distance at the interface. In addition, the interface itself is topologically complex, and physical properties such as surface tension play a significant role in the behavior of the flow. Hence, realistic fluid simulation requires accurate modeling of these interfaces. MIZU uses a volume tracking, or volume-of-fluid (VOF), approach. Interfaces are tracked on 3D generalized hexahedral cells and localized over one cell width at each timestep. They are assumed to be planar within each cell, yielding a globally piecewise planar approximation to the actual interface.

Once the flow conditions are computed, the results are rendered as spherical metaballs (blobs), with blob radius determined by the volume fraction of the appropriate fluid. While this smears some of the fine detail of the solution, and would likely not be appropriate for scientific visualization, it results in motion realistic enough for film and commercial applications.

The results were rendered using Blue Sky Studios, Inc.'s proprietary ray-tracing renderer, CGI Studio. Figure 1 shows a selected frame from an animation of a fluid being poured into a glass box.

As an initial test, we simulated a fluid being poured into a glass box eight centimeters on each side. Since the geometry of this situation is simple, a 32x32x32 orthogonal mesh was used, resulting in 32,768 computational cells of 0.25 cm on each side. The calculation thus requires solution of linear systems involving this number of unknowns at each timestep. While this mesh is somewhat coarse, it nevertheless yields fairly realistic results.

The simulation was mostly performed using 250 MHz R10000 processors on an SGI Origin 2000 (although MIZU provides for parallel execution in either SMP or distributed modes via MPI, these calculations were performed in serial mode). CPU time requirements were significant, but not unreasonable. Early in the simulation, when the flow is complex and difficult to compute, roughly an hour of CPU time was required for each frame of animation. Near the end of the simulation, when the flow is nearing steady-state, only a few minutes per frame were required.

The resulting animation sequences are quite realistic, particularly the degree to which the complex topology of the interface is captured. A selected frame is shown in Figure 1.

Future Improvements

Currently, blobs are placed at the centroid of each cell. More realistic results could be achieved if they were placed at the centroid of the region of the cell occupied by the fluid being rendered. Since a planar interface is reconstructed in the course of the flow calculation, this information is available.

In addition, blobs need not be spherical. Ellipsoids that "fit" the cell region in question, using cell geometry and the reconstructed fluid interface, can be used. This would further enhance the realism of the final animation.

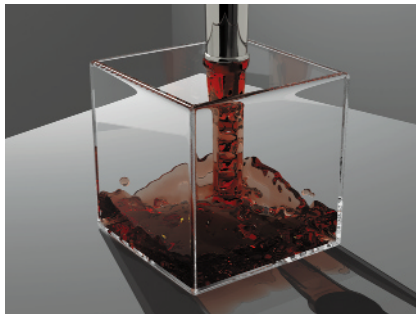


Figure 1
Simulation time: 0.77 seconds.

References

1. N. Foster and D. Metaxas. Realistic Animation of Liquids, Proceedings GI, pp. 204-212, 1996.
2. M. Kass and G. Miller. Rapid, Stable Fluid Dynamics for Computer Graphics, Proceedings of SIGGRAPH 90, in Computer Graphics, volume 24, pp. 49-57, 1990.
3. W. J. Rider, D. B. Kothe, S. J. Mosso, J. H. Cerruti & J. I. Hochstein. Accurate Solution Algorithms for Incompressible Multiphase Fluid Flows, Technical Report AIAA 95-0699, AIAA, 1995. Presented at the 33rd Aerospace Sciences Meeting and Exhibit.
4. W. J. Rider, D. B. Kothe, E. G. Puckett & I. D. Aleinov. Accurate and Robust Methods for Variable Density Incompressible Flows with Discontinuities, In V. Venkatakrishnan, M. D. Salas & S. R. Chakravarthy, editors, Workshop on Barriers and Challenges in Computational Fluid Dynamics, pp. 213-230, Boston, MA, 1998. Kluwer Academic Publishers.
5. J. A. Turner, R. C. Ferrell & D. B. Kothe. JTpack90: A Parallel, Object-Based Fortran 90 Linear Algebra Package, Proceedings of the Eighth SIAM Conference on Parallel Processing for Scientific Computing, Minneapolis, MN, March 14-17, 1997.

* Japanese for water.

** It must be noted that MIZU has not been released publicly and is in a state of development. It was only available to us due to the fact that one of us (Turner) was involved in its development while at LANL. More information on MIZU can be found at www.zephyr-group.com/mizu/

*** That is, if the mesh spacing is decreased by a factor of two, accuracy improves by a factor of four.[3]

Ramesh Raskar
University of North Carolina at Chapel Hill
www.cs.unc.edu/~raskar/Oblique/

Oblique Projector Rendering on Planar Surfaces for a Tracked User

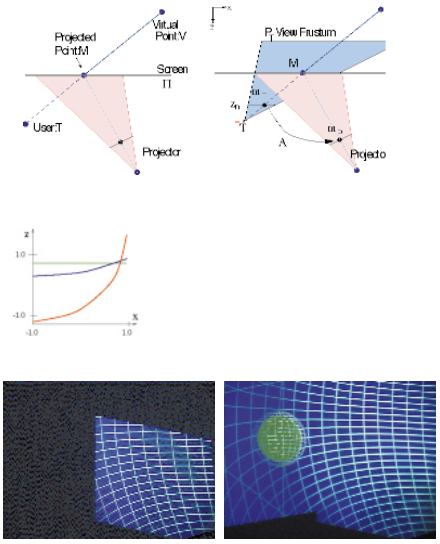


Figure 1: (a) Oblique projectors create keystoned imagery.
(b) The modified projection matrix achieves off-axis projection P_T and a collineation A_{4x4}.
(c) Depth buffer values along a scan line for points along constant depth. Using P_T (green). After collineation (red) and with depth-buffer approximation (blue).
(d) An oblique projector (e) and its contribution to overlapped projectors.

$$A_{4x4} = \begin{bmatrix} a_{11} & a_{12} & 0 & a_{13} \\ a_{21} & a_{22} & 0 & a_{23} \\ 0 & 0 & 1 & 0 \\ a_{31} & a_{32} & 0 & 1 \end{bmatrix} \quad (1)$$

$$A'_{4x4} = \begin{bmatrix} a_{11} & a_{12} & 0 & a_{13} \\ a_{21} & a_{22} & 0 & a_{23} \\ 0 & 0 & 1 - |a_{31}| - |a_{32}| & 0 \\ a_{31} & a_{32} & 0 & 1 \end{bmatrix} \quad (2)$$

Projectors are typically mounted so that their optical axis is perpendicular to the planar display surface. Such configurations also are used in immersive environments to render perspectively correct imagery for a head-tracked moving user. These environments include CAVEs, PowerWalls, and ImmersaDesks (back-lit and tilted desktop workbenches). By design, typical display systems try to maintain the image plane parallel to the plane of the display surface. However, this leads to a need for constant electro-mechanical alignment and calibration of display systems.

In this sketch, we show that it is possible to allow oblique projection on planar display surfaces and still render correct imagery of 3D scenes without additional computational cost. We describe how oblique projection can replace frequent alignment with simple calibration. We use a traditional graphics pipeline with a modified projection matrix and an approximation of the depth buffer.

Oblique Projection
Consider rendering the virtual point V for the user at T using an oblique projector as shown in Figure (a). For planar display surfaces, the images m_T and m_P of the virtual point V can be computed by first finding projection of V onto the display surface, M. A simple observation is that the two images of a common virtual point are related by a collineation, which is well known to be a 3 x 3 matrix defined up to scale. This observation allows us to create a new projection matrix during rendering for the projector as a product of a traditional off-axis projection matrix, P_T (from the user's viewpoint) and a matrix, A_{4x4} (from the 3 x 3 collineation matrix).

Without a loss of generality, let us assume that the display plane, Π, is defined by z = 0. There are various ways to create P_T and A_{4x4}. We will use a method that updates P_T as the user moves but the collineation matrix remains constant. We create an axis aligned rectangle S on Π bounding the keystoned quadrilateral illuminated by the projector. Define a view frustum by first creating a pyramid with T and the four corners of S and then truncating it with a near plane, $z = T_Z - z_n$, and a far plane, $z = T_Z - z_f$. This is similar to OpenGL's *glFrustum* setup. The projection matrix for this view frustum is, $P_T = Frustum(T, S, z_n, z_f)\, Translate(-T)$.

Next we calculate the collineation between images of V: m_T due to P_T, and its image in the projector, m_P. If the 3D positions of points on Π illuminated by four or more pixels of the projector are known, the eight parameters of the collineation matrix, $A = [a_{11}, a_{12}, a_{13}; a_{21}, a_{22}, a_{23}; a_{31}, a_{32}, 1]$, can be easily calculated. We create a new matrix, A_{4x4} to transform the pixel coordinates but try to keep the depth values intact. The complete projection matrix is $A_{4x4} P_T$. See equation (1).

Depth Buffer Approximation
Although the naive approach described above creates correct images of virtual 3D points, it is important to note that the traditional depth buffer cannot be effectively used for visibility and clipping. The depth values of virtual points between near and far planes due to P_T are mapped to [-1,1]. Let's say, $[m_{Tx}, m_{Ty}, m_{Tw}]^T = P_T[V,1]^T$ and $m_{Tz} / m_{Tw} \in [-1,1]$. After collineation, the new depth value is actually $m_{Tz} / (a_{31} m_{Tx} + a_{32} m_{Ty} + m_{Tw})$ which (i) may not be in [-1,1], resulting in undesirable clipping and (ii) is a function of pixel coordinates, changes quadratically, and hence cannot be linearly interpolated during scan conversion for visibility computation (Figure (c). In general, we cannot achieve two hyperbolic interpolations for the depth values with a single 4 x 4 matrix. In other words, we must first compute an image with P_T (``divide by w''), and then warp the resultant image. This requires a two-pass rendering method: first render the image and load it in texture memory and then achieve warping using texture mapping. However, we can achieve the rendering and warping in a single pass using an approximation of the depth buffer. Note that $m_{Tz} / m_{Tx} / m_{Tw}$ and $m_{Ty} / m_{Tw} \in [-1,1]$ for points rendered inside the rectangle S. Hence $(1 - |a_{31}| - |a_{32}|)\, m_{Tz} / (a_{31} m_{Tx} + a_{32} m_{Ty} + m_{Tw})$ is guaranteed to be in [-1,1]. Further, by construction of P_T, the angle between the projector's optical axis and the normal of the planar surface is the same as the angle between the optical axis and the retinal plane of the frustum for P_T. Thus, if this angle is small (i.e. $|a_{31}|$ and $|a_{32}| \ll 1$), the depth values are modified but the changes are monotonic and almost linear across the framebuffer as shown in Figure (c). See equation (2).

Applications
The modified projection matrix can be easily calculated by measuring the tracker-sensor at the four corners of the illuminated quadrilateral. The effect of quadratic changes in depth values is minimized when the projector is almost perpendicular. In CAVEs or ImmersaDesks, special effort is taken to align projector pixels to the pre-defined corners. Using the technique described, a rough positioning followed by a simple calibration is sufficient to render correct images.

Acknowledgments
I would like to thank Gary Bishop and my colleagues in the Office of the Future group at the University of North Carolina at Chapel Hill (Mike Brown, Ruigang Yang, Wei-Chao Chen, Herman Towles, Greg Welch, Brent Seales and Henry Fuchs) for helpful discussions and prototype implementation.

Occlusion Culling with Optimized Hierarchical Buffering

Ned Greene
Ned Greene Consulting
ned@ngreene.com

One approach to occlusion culling in z-buffer systems, by which we mean the culling of occluded geometry prior to rasterization, is to replace the z-buffer with a z-pyramid and perform hierarchical z-buffering[1,2] rather than traditional z-buffer scan conversion. However, makers of z-buffer hardware have been slow to adopt this approach because it requires a major architectural revision. Here we describe a simpler and cheaper alternative, adding to a conventional z-buffer pipeline a culling stage that employs hierarchical z-buffering optimized for conservative culling (culling that sometimes fails to cull occluded geometry but never culls visible geometry).

In this architecture (Figure 1), a culling stage is inserted between the transformation and rendering stages of a conventional z-buffer pipeline. The culling stage receives transformed primitives, hierarchically tiles them into a z-pyramid,[2] thereby culling most occluded primitives, and passes visible and nearly visible primitives on to a conventional rendering stage. The rendering stage establishes visibility definitively at each image sample using a standard z-buffer.

One big advantage of this architecture is that it enables hierarchical z-buffering to be highly optimized for conservative culling. Toward this end, we employ a very compact z-pyramid to reduce storage cost, memory traffic, and tiling computations. Within a standard z-pyramid having 4x4 decimation, 4x4 tiles are stored as arrays of full-precision z values. Instead, we use 16-bit z values, and as diagrammed in Figure 2, at the finest pyramid level we represent 4x4 tiles as two z values and a coverage mask: a zfar value for the tile (zfar_tile), a coverage mask indicating samples covered by primitives encountered since the last update of zfar_tile, and the zfar value of the samples covered by the (zfar_mask). Since zfar_mask is at or behind the corresponding occluders, culling performed with this data structure is conservative.

This compact z-pyramid requires only about 10 percent of the storage of a standard z-pyramid and culls approximately 90 percent of occluded primitives. Simulations on a variety of simple and complex scenes show that hierarchical tiling into this z-pyramid generates only about 10 percent as many reads and writes as standard hierarchical z-buffering, and only 1-2 percent as many as traditional z-buffering. These figures assume that primitives are traversed in approximately front-to-back order, which can be accomplished by organizing the scene in bounding boxes and traversing the boxes front to back.

To illustrate the dramatic bandwidth reductions that are possible with this architecture, we rendered the scene of Figure 3, which consists of 167 million replicated polygons organized into bounding boxes. With front-to-back traversal of bounding boxes, the culling stage reduced the depth complexity of polygons processed by the rendering stage from 41.5 to 2.3, an 18-fold reduction. To reduce geometry traffic in the pipeline, we tested bounding boxes for visibility with the culling stage prior to sending their primitives through the pipeline.[1] In rendering Figure 3, the culling stage generated only .58 reads and writes of z-pyramid values per image sample, on average.

Bandwidth simulations on this and other scenes show that this occlusion-culling architecture could extend the real-time rendering performance of z-buffer hardware to much more deeply occluded scenes.

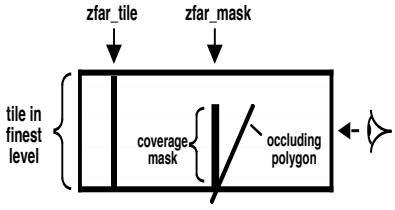


Figure 2. Side view of a finest-level 4x4 tile within the z-pyramid. To reduce memory traffic and storage, finest-level tiles are represented as two z values and a coverage mask.

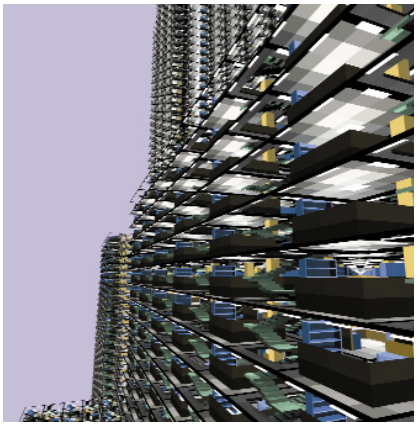


Figure 3. For this scene, the culling stage reduces the depth complexity of polygons that need to be rendered from 41.5 to 2.3.

References

1. N. Greene, M. Kass, and G. Miller. Hierarchical Z-Buffer Visibility, Proceedings of ACM SIGGRAPH 93, 231-238.
2. N. Greene. Hierarchical Polygon Tiling with Coverage Masks, Proceedings of ACM SIGGRAPH 96, 65-74.

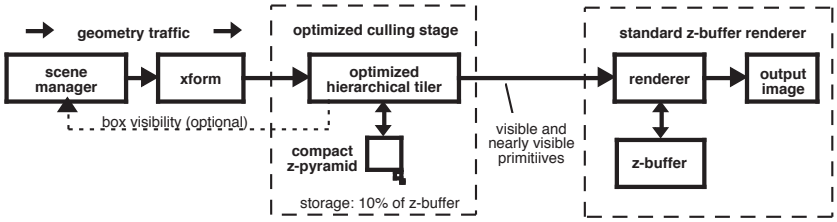


Figure 1. Block diagram of the proposed architecture. Most occluded geometry is culled with a separate culling stage that is highly optimized for conservative culling.

Paul Hansen
SGI
hansen@sgi.com

OpenGL Texture-Mapping With Very Large Datasets and Multi-Resolution Tiles

On OpenGL systems without the "clipmap" extension, a tiled approach can be used to provide the correct resolution image data for different parts of a 3D visualization, where very large texture maps are desired. The application, called "EarthView" is a planetary browser, which can be used to view arbitrary amounts of geo-referenced image and terrain elevation data. The proper number and size of bi-linear texture-maps that can fit into available memory are used as a resource, constantly recycled by being "sub-loaded" with new data as the user browses into new areas of the globe.

EarthView uses recursive descent, window and horizon culling, and a quad-tree structure to focus in on the part of the globe currently in view. However, all vertex and texture coordinates are not pre-stored, but rather calculated on-the-fly, based on a simple cosine table. This allows EarthView to support close-in zooms to any part of the globe, and high-resolution insets can be supplied for desired areas. Terrain elevation data are also utilized on a multi-resolution basis, which allows the terrain to "morph" as it is viewed from different positions. This allows EarthView to browse at very high frame rates.

EarthView mimics the texture-mapping calculations performed in OpenGL in order to decide which image resolution tile to use for a particular area. This creates a high-quality result similar to what could be accomplished with a single, very large, mipmapped texture, and thus avoids memory limitations. EarthView can browse very large datasets even on systems with less texture memory, and browsing of the same areas repeatedly is enhanced on systems with larger memory.

EarthView has an unusual projection for polar regions, designed to save diskspace, look better, and run faster. Typical cylindrical projections stretch the polar regions, creating an imbalance between the amount of data corresponding to longitude versus latitude. Also, polar geometry is tesselated differently, avoiding the typically thin "pie-slices" at the poles. Various OpenGL and window-system features and extensions are used to enhance the view and speed up performance, including texture objects and subloads, the "clamp-to-edge" extension, the "packed pixels" extension, the "texture-LOD" extension, vertex arrays, and asynchronous I/O.

The EarthView application was designed to demonstrate the advantages of a new Intel-based hardware platform called the SGI Visual Workstations 320 and 540. A 320 system can be configured with 1Gb of system memory, and with the Integrated Visual Computing (IVC) architecture, more than 900 Mb of this can be allotted to textures, an unprecedented amount. This, combined with a high textured fill rate, a high textured polygon rate, and the ability to move data in from the disk in parallel with other operations, gives outstanding results.

For the first time, a system costing less than $5K can browse planetary data with speed and visual realism that was previously available only to government and industry personnel on very expensive systems. As a dedicated platform, this alone would be well worth the price; when not running the browser, users still have a general-purpose, high-performance graphics workstation.

Phong Shading at Gouraud Speed

Greg Rivera
Virtus Corporation
greg.rivera@virtus.com

Purpose

Since its presentation in 1973, Phong shading[1,2] has been the standard technique for rapidly rendering specular highlights. Because it supports interior highlights, Phong shading retains a high degree of visual quality even with low-polygon-count models. However, despite 25 years of improvements,[3,4] real-time Phong shading is so costly that it's still only available on high-end hardware. This sketch presents a new implementation, "Rapid Phong," that is finally fast enough in both setup and execution to be easily implemented in real-time software and low-end hardware. By using a few matrix rotations during polygon setup, this algorithm reduces Phong's per-pixel renormalization and exponentiation to a single per-pixel linear interpolation and fast multiply.

Basic Phong Shading

Basic Phong-style shading $(\vec{R} \bullet \vec{V})^n$ consists of four main steps per pixel:

1. Linearly interpolate the 3D-reflection vector $(R'_x R'_y R'_z)$ across the polygon.
2. Renormalize the interpolated reflection vector $\vec{R}'$ per pixel.
3. Calculate the highlight by dotting the reflection and view vector $\vec{R}' \bullet \vec{V}$.
4. Sharpen the specular falloff by using a specular exponent (n).

New Rapid Phong Implementation

Per triangle, we will transform the reflection vectors to eliminate the per-pixel highlight dot product, remove $\vec{R}'_z$ interpolation and $\vec{R}'$ renormalization using the Pythagorean theorem, simplify the specular falloff, and then roll to remove the $\vec{R}'_y$ per-pixel interpolation.

View-Transform the Reflection: Transform the view and reflection vectors into view (eye) space, so that the new $\vec{V}' = (0,0,1)$. Then $\vec{R}' \bullet \vec{V}$ becomes $\vec{R} \bullet (0,0,1)$, reducing the per-pixel highlight dot product to the single component $\vec{R}_z'$. The regular view transform can be used, since it's usually orthonormal. (Note for later that the roll around this "z" axis is arbitrary).

Remove $\vec{R}_z'$ interpolation and $\vec{R}'$ renormalization: Since , $\vec{R}_x'^2 + \vec{R}_y'^2 + \vec{R}_z'^2 = 1$, only interpolate the "x" and "y" components of the transformed reflection vectors and extract $\vec{R}_z'$ via the Pythagorean theorem. By definition, $\vec{R}_z'$ will already be normalized. For speed, generate the $\vec{R}_z'$ highlight via $1 - (\vec{R}_x'^2 + \vec{R}_y'^2)$, removing the square root. This also reduces the up-to-30-percent linear interpolation error to under six percent.

Simplify the specular falloff: A good approximation for the specular falloff exponentiation is to premultiply the reflection vectors by the square root of the specularity exponent (n). The resulting highlight value of $1 - n * (\vec{R}_x'^2 + \vec{R}_y'^2)$ stays accurate down to 50-percent brightness and only needs a clamp to zero per pixel.

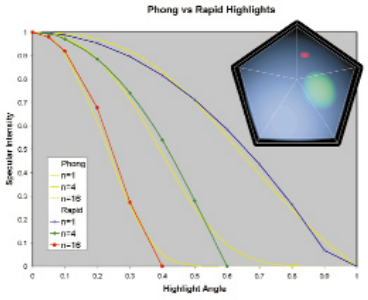
Remove $\vec{R}_y'$ per-pixel interpolation: Since $1 - (\vec{R}_x'^2 + \vec{R}_y'^2)$ is circularly symmetric, rotate the 2D $(R_x' R_y')$ reflection components into $(R_x'' R_y'')$, aligning them with the screen's x rows and y columns. As a result, $\vec{R}_y''$ stays constant within each row. (As noted in the view-transform step, the roll around "z" was arbitrary.)

Per-Pixel Calculations:

We obtain the per pixel highlight value from $1 - (\vec{R}''^2_x + \vec{R}''^2_y)$. Since $\vec{R}''^2_y$ is constant within each row, only $\vec{R}''_x$ needs interpolating per pixel. Its squaring can be changed to an add via finite differences. The specular falloff was already included earlier, but still requires a clamp to zero. The final cost is only 3 adds and one clamp per pixel.

Finally

The reasonable setup overhead and low per-pixel cost of Rapid Phong allows it to run at speeds approaching Gouraud. Hopefully, refinements like this will finally permit the higher-quality rendering of Phong shading to become widely available, even on real-time consumer level systems.



References

1. Phong, B. T. Illumination For Computer-Generated Images, PhD Dissertation, Department of Computer Science, University of Utah, Salt Lake City, 1973.
2. Foley, J. D. and A. Van Dam. Fundamentals of Interactive Computer Graphics, Addison Wesley, Reading, MA., 1983.
3. Bishop, Gary and Weimer, David. Fast Phong shading, ACM SIGGRAPH 96 Conference Proceedings.
4. Schlick, Christophe. A Fast Alternative to Phong's Specular Model , Graphics Gems IV, Academic Press, Inc., 1994.

Victor Ng-Thow-Hing
University of Toronto
victorng@dgp.toronto.edu

Physically Based Anatomic Modeling for Construction of Musculoskeletal Systems

In recent years, we have seen several anatomically inspired modeling systems, such as those created by Scheepers et al.[3] and Wilhelms and van Gelder.[4] Although these modeling systems may be suitable for defining superficial body shape, the choice of shape primitives is not sufficiently accurate for medical applications because they do not capture details of muscle fiber architecture nor allow specification of complex attachment areas of musculotendon to bone. Furthermore, these muscles are incapable of generating forces for physical simulation.

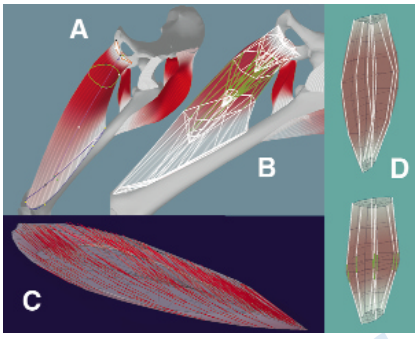


Figure 1
(a) Profile curves are used to specify muscle attachment areas and lines of action. (b) Green viscoelastic units in muscle can be selectively contracted. (c) Generated fibers show correct orientation in isolated human soleus specimen. (d) Relaxed B-spline solid (top) has selected green fibers contracted while preserving volume (bottom).

We are developing an anatomically based modeling system that integrates both the physical and geometric properties of muscle and tendon for the purpose of constructing and simulating musculoskeletal systems. Our goal is to create a muscle model that can be embedded with physical properties to enable animation of active muscle contraction and other inertial effects as the muscles move with their underlying bones. Essentially, we are incorporating low-level, physically based muscle models, such as those introduced by Chen and Zeltzer[1] with the primarily geometric, anatomically based modeling systems mentioned above.

Muscle Model

We use B-spline solids to represent muscle and tendon. The B-spline solid is the volumetric triple tensor product function of three material coordinates, *u, v,* and *w* whose domain maps to the spatial volume:

$$x(u,v,w) = \sum_{i=0}^{l}\sum_{j=0}^{m}\sum_{k=0}^{n} B_i^u(u)B_j^v(v)B_k^w(w)q_{ijk}.$$

A 3D spatial point x(*u,v,w*) in the solid is produced by the weighted sum of control points q_{ijk} and the product of B-spline basis functions. The volumetric representation allows individual muscle fibers to be visualized as iso-parametric curves embedded in the solid. It also enables an efficient point-inclusion test that can be used for collision detection with other muscles and bone geometry.

The control points are mapped to a unique set of mass points embedded within the solid, allowing physical reaction constraints[2] and internal muscle forces to be applied at these mass points.

We use a Lagrangian formulation for the equations of motion of the solid to enforce global volume conservation simultaneously with the local attachment and non-interpenetration reaction constraints and the internal muscle force models.

Attachment of Musculotendon to Bone

We use these muscle models as force actuators in a simulation framework that allows muscles to be visualized and simulated with articulated skeletons made up of rigid bones and various joint constraints. To produce an initial shape for muscle, pre-built muscles from an anatomical library can be loaded into the system or the muscles can be interactively created by sketching profile curves directly onto the bone surfaces by direct manipulation.

These curves can be subsequently adjusted while maintaining bone attachment to provide high-level shape manipulators for a practitioner who may wish to specify non-trivial attachment areas or adjust the various lines of action of muscle.

Conclusion

The combination of geometric and physically based properties in a muscle model allows closer study of the interrelationships between anatomical form and function. Virtual reconstructions of both real and non-existent musculoskeletal systems can be created and simulated.

References

1. David T. Chen and David Zeltzer. Pump It Up: Computer Animation of a Biomechanically Based Model of Muscle Using the Finite Element Method, Computer Graphics (SIGGRAPH 92 Proceedings), pp. 89-98, July 1992.
2. John C. Platt and Alan H. Barr. Constraint Methods for Flexible Models, Computer Graphics (SIGGRAPH 88 Proceedings), pp. 279-288, August 1988.
3. Ferdi Scheepers, Richard E. Parent, Wayne Carlson and Stephen F. May. Anatomy-Based Modeling of The Human Musculature, Computer Graphics (SIGGRAPH 97 Proceedings), pp. 163-172, August 1997.
4. Jane Wilhelms and Allen Van Gelder. Anatomically Based Modeling, Computer Graphics (SIGGRAPH 97 Proceedings), pp. 243-252, August 1997.

Postprocess 2D Motion Blur for Cel Animation

Peter Cucka
DreamWorks Feature Animation
pcucka@anim.dreamworks.com

Motion blur arises naturally in cinematography as a result of movement in the scene while the camera's shutter is open. In its absence (for example, when a scene is illuminated by a strobe light), movement is perceived to lack fluidity and coherence. In traditional cel animation, motion blur is simulated with streaks ("speed lines") or by instancing objects multiple times in a drawing ("vibrations"),[1] but these techniques are labor-intensive and are practical only for localized motion (not for motion arising from a camera pan, for example). Algorithms for simulating motion blur in computer-generated animation have been available for some time,[2] and their effectiveness in reducing "strobing" and enhancing photorealism and the sensation of motion is well-recognized. Such algorithms typically take advantage of the known velocities of objects in a 3D scene by filtering along the objects' motion paths. In contrast, velocity information is generally not directly available in 2D cel animation. Fortunately, though, "optical flow" estimation (deriving velocity information from image sequences) has received considerable attention in the computer-vision community. This sketch describes a postprocess 2D motion blur algorithm based on optical-flow estimation.

An optical-flow field is a vector field that describes the motion in an image by assigning to each pixel of the image a vector representing its velocity at a given instant in time. Ideally, each pixel's velocity reflects the velocity of the point in the 3D scene of which the pixel is a projection, but this need not be the case (consider an image of a featureless, rotating sphere). In evaluating a number of different algorithms for estimation of optical flow, Anandan's correlation-based algorithm[3] was found to perform well on animation cels. Correlation-based algorithms derive velocities from the displacements of matching neighborhoods of pixels in successive images, where the quality of a match is determined by cross-correlation of the intensity distributions in the neighborhoods.

Anandan's algorithm employs a multilevel Laplacian image pyramid,[4] in which matching takes place between two images at multiple resolutions. Results from lower-resolution matches are used as initial guesses for higher-resolution matches, which greatly reduces the search space required to recover large displacements. To single-pixel accuracy, the velocity of a pixel in the first image is its displacement from the pixel in the second image for which the sum of squared differences (SSD) in intensity between their two neighborhoods is minimized. Subpixel accuracy is achieved by minimizing a quadratic approximation to the SSD surface over a search space centered at this integer displacement. The algorithm also imposes a smoothness constraint to minimize variations in velocity both between and within pyramid levels.

Simulated motion blur is applied to an image via line-integral convolution[5] of its optical-flow field. Satisfactory results are obtained with a simple box filter kernel, whose length and center point can be adjusted to vary properties of the effect, such as overall blurriness. (Line-integral convolution integrates along streamlines of the optical flow field, and it should be noted that these do not in general correspond to motion paths of objects in the scene, but the resulting blur is visually more appealing than that produced by, for example, a DDA line-drawing kernel.) Where possible, motion blur is applied to individual scene elements independently, as this reduces unnecessary computation and allows for greater flexibility in compositing.

The accompanying figures illustrate the process: Figure 1 is a frame from the animated feature film "The Prince of Egypt" prior to motion blurring. Figure 2 is a visualization, by line-integral convolution with a random noise field, of the computed optical-flow fields for two scene elements. Figure 3 is the final frame with motion blur.

Thanks to Lance Williams for the inspiration for this work and for his invaluable input.

References

1. F. Thomas and O. Johnston. Disney animation: The Illusion of Life, New York, Abbeville Press, 1981, pp. 116-117.
2. M. Potmesil and I. Chakravarty. Modeling Motion Blur in Computer-Generated Images, Computer Graphics (SIGGRAPH 83 Proceedings) 17, 1983, pp. 389-399.
3. P. Anandan. A Computational Framework and An Algorithm For the Measurement of Visual Motion, International Journal of Computer Vision 2, 1989, pp. 283-310.
4. P.J. Burt and E.H. Adelson. The Laplacian Pyramid as a Compact Image Code, IEEE Transactions on Communications 31, 1983, pp. 532-540.
5. B. Cabral and L. Leedom. Imaging Vector Fields Using Line Integral Convolution, Computer Graphics (SIGGRAPH 93 Proceedings) 27, 1993, pp. 263-272.

Claudio Pinhanez
Frank Nielsen
Kim Binsted
Sony Computer Science Laboratory
pinhanez@media.mit.edu

Projecting Computer Graphics on Moving Surfaces: A Simple Calibration and Tracking Method

We are currently investigating the possibilities of transforming every surface in a space into a display and, in particular, of projecting graphics on movable surfaces. In this sketch, we present a simple method to convert tracking points from a camera into positions on the image to be projected. The main advantage of our method is the simplicity of a calibration process that does not require knowledge of the intrinsic parameters (focal length, dimensions of the imaging device)of the camera or the projector.

Camera and Projector Calibration

The relationship between points observed on a planar surface from two different cameras is known to be a *homography.*[1] A homography is a 3x3 matrix that defines a linear application in the projective space that, for a given planar surface of the real world, maps all projected points in one camera's image into the other camera's image.

The fundamental observation is that from a geometric point of view, "ideal" pinhole projectors and cameras are identical (see Figure 1). Let H denote the homography that relates the image of the projector image frame to the camera image frame. This means that a 2D-point on the camera image $\overline{c} = (x_c/z_c, y_c/z_c)$ matches a 2d point $\overline{p} = (x_p/z_p, y_p/z_p)$ on the projector image as follows:

$$p = \begin{bmatrix} x_p \\ y_p \\ z_p \end{bmatrix} = Hc = H \begin{bmatrix} x_c \\ y_c \\ z_c \end{bmatrix}$$

A homography is completely defined if the projection of four 3D points of the world on both image planes is known. To determine the homography between a camera and a projector, we need simply to obtain the four needed points while manually aligning a projection of the surface with the real surface (see Figure 1).

The homogeneous coordinates of the four points to be projected, $p_i = (x^i_p, y^i_p, 1)$, i=1,2,3,4, are determined arbitrarily, making sure that the points are visible, and there is a way to move the real surface so it aligns with the projection. Then, we consider the homogeneous coordinates of the four points on the camera image as sensed by the tracking system, $c_i = (x^i_c, y^i_c, 1)$, i=1,2,3,4. Taking the matrices corresponding to these two sets of four points, $P=(p^T_1 p^T_2 p^T_3 p^T_4)$ and $C=(c^T_1 c^T_2 c^T_3 c^T_4)$, we obtain $P=HC$, whose solution is

$$H=PC^T(CC^T)^{-1}$$

During run-time, we simply take a point in the camera image $c = (x_c, y_c, 1)$, project through the homography H obtaining $p=Hc =(x_p, y_p, z_c)$, and compute the position on the projector's image plane, $(x_p/z_p, y_p/z_p)$.

Surprisingly, this calibration step is numerically stable even with only four points and can be done, in practice, in a few seconds. We believe that the stability is also related to the fact that in our experiments the projection centers of the camera and the projector are close to being aligned. Notice that there is no need to determine either the camera's intrinsic parameters or the projector's.

Tracking the Projection Surface

In our experiments, we have used plain markers on the projection surface. In particular, we employed infrared LEDs that can be easily tracked by a camera with an infrared filter. However, if we move the mask too quickly, we observe that the projected image "falls behind" the moving surface. That is, there is a "shifting" effect where the observations at discrete time t on the camera image $c(t)$ are displayed by the projector at time $t+dt$ using the estimate at time t, p(t)=Hc(t). To reduce the "shifting" problem, we employ a predictive Kalman filter[2] that estimates the most likely position of every point at time $t+dt$, using the equations of dynamics as the underlying model of the Kalman filter, as shown in Figure 1. The parameter dt, corresponding to the average delay between sensing and displaying, is determined experimentally. The Kalman filtering approach proved to be very effective in our experiments.

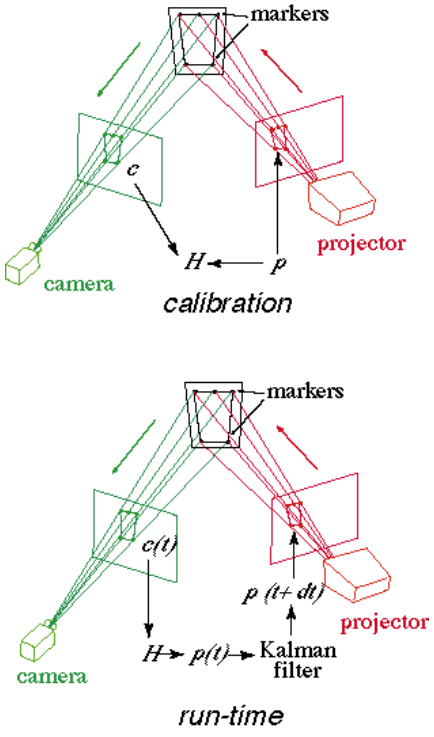


Figure 1
Calibration process and run-time system.

References
1. O. Faugeras. Three-Dimensional Computer Vision: A Geometric Viewpoint, The MIT Press, Cambridge, Massachusetts. 1993.
2. A. Gelb (ed.). Applied Optimal Estimation, The MIT Press, Cambridge, Massachusetts. 1974.

Prototype System of Mutual Telexistence

Yutaka Kunita
The University of Tokyo
kuni@star.t.u-tokyo.ac.jp

Masahiko Inami
Taro Maeda
Susumu Tachi

This prototype allows for face-to-face communication between remote users.[1]

Unlike most conventional tele-conferencing systems, which limit the users' viewpoint to a fixed position, a mutual telexistence system should provide images of the other users that correspond to the change of a user's eye position in the computer-generated 3D space. On the other hand, with "image-based rendering," complex photo-realistic images can be effectively synthesized at arbitrary viewpoints. However, with this technique, the system must synthesize the image in real-time for smooth communication. This prototype system includes the following features to fulfill this real-time request:

1. Geometric information such as a "depth map" is needless. Only one parameter of the object's distance is required. So, the system does not include a time-consuming process such as pattern matching. Data processed by the rendering computer are reduced in advance at the stage of capturing the source image. This means that the rendering computer doesn't have to deal with bulky data of the "plenoptic function."

2. Fast rendering is enabled by using graphic hardware acceleration of texture mapping. The system is comprised of relatively low-cost general graphic hardware.

The prototype system includes a camera unit, the rendering PC, and the control PC. In the camera unit, 12 small color CCD cameras are aligned horizontally in a row on the linear actuator at intervals of 54mm. These cameras rotate around their optical axes, so the direction of their scanning lines is vertical. Moreover, all the cameras are synchronized by the same genlock signal, and a video switch is installed between the camera unit and the rendering PC. This design allows the rendering PC to selectively capture only one scanning line among 12 video streams, which reduces data and the number of video capturing devices required. The rendering PC synthesizes the images of the object from arbitrary viewpoints by texture-mapping long tile images on a transparent plane in the computer-generated 3D space. The control PC indicates the channel of this switch and controls the motion of the linear actuator.

Figure 2. shows a video sequence synthesized by the system. Moving human figures are successfully rendered in real-time.

More specific technical data is available at: www.star.t.u-tokyo.ac.jp/projects/mutel/

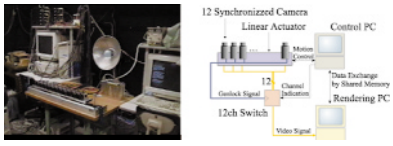


Figure.1
Left: overview of the prototype system.
Right: block diagram of the prototype system.

Figure.2 Synthesized video sequence.

Prototype System

Reference
1. Susumu Tachi, Taro Maeda, Yasuyuki Yanagida, Masaaki Koyanagi, and Hiroki Yokoyama: A Method of Mutual Tele-Existence in a Virtual Environment. In Proceedings of ICAT 98, pp9-18, 1996.

Eugene Lapidous
Guofang Jiao
Trident Microsystems Inc.
eugene@tridmicr.com

Quasi-Linear Z Buffer

Figure 1

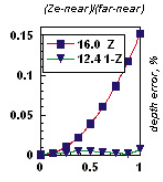


Figure 2

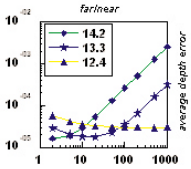


Figure 3

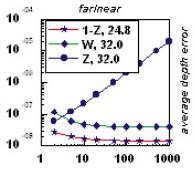


Figure 4

This sketch presents a new type of depth buffer with quasi-linear mapping from eye to screen space that significantly improves depth resolution and is easy to use with standard 3D hardware and APIs.

Perspective projection of eye-object distance to depth in screen space is usually bound by normalization to [0,1] range, preservation of lines and planes, and preservation of the sign of distance change.[1] Non-linear mapping under these conditions results in a loss of roughly $\log_2(r)$ bits of the depth precision for a given ratio r of the distances to the far and near clipping planes.[2] It causes severe visual artifacts in open environments with large dynamic range r, typical for flight simulators and games.[3] If the W value after projective transformation is proportional to the Ze value in the eye space, linear mapping can be accomplished by storing W values in the depth buffer.[4] However, correct implementation of the W buffer requires costly high-precision per-pixel division and isn't widely supported; it isn't suitable for scenes with very low dynamic ranges.

The main factors that affect eye-space depth resolution are mapping of the eye-space distance (Ze) to the screen-space depth (D) and precision of the format used to store D in the depth buffer:

$$\delta Ze(Ze)=\frac{dZe}{dD} * \delta D(D)$$

Linear mapping and integer storage format keep W-buffer resolution independent from the distance to the object. We achieve a similar result by using a combination of simple mapping and format precision functions that compensate each other's non-linearity, instead of requiring both of them to be linear.

Screen depth D is computed using all mapping constraints except distance change sign; the result is complementary to the screen Z in the same range [0,1]:

$$D = 1-Zs = \frac{d}{f-d} * \left(\frac{f}{Ze} - 1\right)$$

where f and d are the distances to the far and near clipping planes in the eye space.Depth value is stored in the floating-point format:

$$D = \begin{cases} 2^{e-2^n} \cdot 1.f \ldots (e>0) \\ 2^{1-2^n} \cdot 0.f \ldots (e=0) \end{cases}$$

where n is a number of bits of exponent (e) and m is a number of bits of mantissa. Depth errors due to mapping and storage format compensate each other when D is close to 0 (the object is close to the far plane), increasing overall resolution. For instance, cloth-body errors in the scene with dynamic range 1000 are visible with 16-bit screen z buffer (Figure 1, left), but not with floating-point complementary z (Figure 1, right).

The above equations define a range of ratios f/d that minimize depth error for the known format (m,n). For instance, 12.4 16-bit format (m = 12, n=4), supported by many 3D accelerators, is optimal to make depth precision of the floating-point complementary z buffer almost independent from Ze at f/d = 100, decreasing errors more than 10 times in comparison with screen z (Figure 2).

Average depth error in the view volume depends on f/d ratio, storage format and sampling distribution. Results for complementary z and uniform sampling (Figure 3) show that 12.4 format is the best for f/d>200, while 13.3 is better for 200>f/d>20. To keep quasi-linear mapping for the wide range of f/d ratios, we propose to support a set of storage formats with different exponent sizes. The best format for the current scene is derived from its dynamic range and region of interest. In many practical cases, it may be enough to use integer format (n = 0) for scenes with small dynamic ranges and four bits of exponent for open environments.

When per-pixel size of the depth storage increases, precision becomes limited by per-vertex input format, usually, 32-bit IEEE float. In this case, the complementary z buffer is 2.8 times better than W buffer, without an associated cost/performance penalty (Figure 4).

References

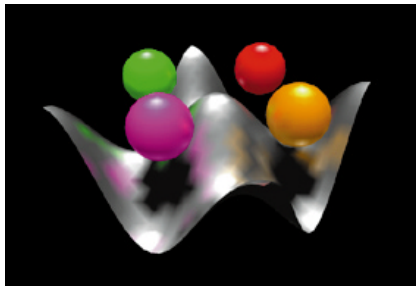
1. Newmann W.M. and Sproull R.F. Principles of Interactive Computer Graphics, 1981 New York, McGraw-Hill.
2. Kempf R., Frazier C. OpenGL Reference Manual, ver.1.1, 1997 Addison-Wesley, p. 164.
3. Swarovsky J. Extreme Detail Graphics, Proceedings of the Game Developers Conference 1999, pp. 899-904
4. Microsoft Corp. What are Depth Buffers? DirectX6 SDK, 1998, msdn.microsoft.com/library/sdkdoc/directx/imover_4xwk.htm

Real-Time Shadows, Reflections, and Transparency using a Z Buffer/Ray Tracer Hybrid

Abe Megahed
Hypercosm, Inc.
abe@hypercosm.com

This sketch describes techniques for merging ray tracing with real-time hidden-surface algorithms such as the z buffer algorithm. The technique offers a smooth transition from interactive display with approximate soft shadows, reflections, and transparency to full ray tracing with all of its associated effects. This is done by using the ray tracer only for tracing shading rays at the vertices of polygons and using another hidden-surface technique for hidden-surface removal in conjunction with Gouraud interpolation.

As computer graphics have evolved, we have seen a schism emerge in rendering techniques between the so-called "real-time" techniques and the "photorealistic" techniques. The result: most interactive computer graphics have a characteristic look, with diffuse shading and highlights included because they can be done quickly and easily, but a noticeable absence of shadows and reflections. Photorealistic images, on the other hand, tend to be characterized by having a lot of reflective surfaces in addition to shadows and transparency, simply because these effects are possible. The approach we have taken integrates these techniques in a uniform way so that as faster hardware and multiprocessing become available, we can have a smooth transition from the traditional z buffer to more realistic ray tracing algorithms.



The Rendering Algorithm

If we trace rays at select points distributed across the surface of the object and interpolate the color across the object, then we may only need to trace a few hundred or thousand rays per frame. This can be done at interactive rates. In addition, we can use high-performance z buffer and Gouraud interpolation hardware to perform the tasks of hidden-surface removal and color interpolation, respectively. With this technique, we have found that on current workstations it is possible to compute shadows, reflections, and transparency in scenes with a few hundred or thousand polygons at acceptable interactive rates.

Benefits of Interpolation

One of the most striking things about the images that are produced is that the interpolation technique causes reflective objects to have the appearance of being diffuse reflectors. The same effect causes shadows and transparency to appear "soft." Previously, this effect could only be achieved through extremely expensive distributed ray tracing. Although the diffuse reflections, transparency, and shadows computed through interpolation are not physically accurate, they give a close enough illusion of the phenomena to be convincing.

Disadvantages of Interpolation

There are two major disadvantages with ray tracing at select vertices and interpolating the color in between. The first problem is that the appearance of the object is dependent upon the underlying tessellation. The second problem is that we are using a smooth interpolation to approximate the specular and transmitted components of the surface of the object, which often change very abruptly compared to the diffuse component. This leads to shading anomalies.

Comparison with Alternative Techniques

There are two primary techniques for simulating shadows with the z buffer. The simplest type of shadowing effect is ground-plane shadows. These shadows are created by projecting the object onto the ground plane and drawing dark polygons where the projections lie. These shadows have hard edges and are only cast onto the ground plane, which limits their usefulness. A more general solution is available through shadow mapping. Unfortunately, the shadow maps use a large amount of memory and are prone to artifacts. Reflections are often simulated using environment maps, images of the scene from the viewpoint of the object, which are then mapped onto the object as reflections. Although these look like reflections at first glance, they do not have the proper geometry for true reflections.

Transparency can be simulated by z buffering machines with special frame buffer hardware for "alpha-blending." However, the distortions caused by refraction of light through different media cannot be accurately simulated. Although these techniques are very useful in the hands of a skilled and experienced computer artist, they are not as general-purpose and easy-to-use as a ray tracer. In addition, after these additional features are added, the initial simplicity and speed of the z buffer algorithm is lost. These same effects can all be computed simply and precisely through ray tracing. It therefore makes sense to restrict the z buffer to what it does best (hidden-surface removal) and restrict the ray tracer to the tasks that only it can perform.

Conclusions

We have demonstrated the practicality and usefulness of the algorithms described here in a prototype system. We have found that inclusion of reflections and shadows in a real-time system not only contributes to a better understanding of spatial relationships, but also increases the gamut of possible surface appearances. While this technique may not be appropriate for every real-time application, we have found that in certain circumstances, it can be very effective.

Yan Zhuang
John Canny
University of California, Berkeley
yzhuang@cs.berkeley.edu

Real-Time and Physically Realistic Simulation of Global Deformation

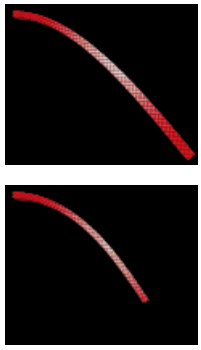


Figure 1: The left end of the beam is fixed. The top image shows its *distorted* deformation under gravity, using linear strain. The bottom image shows the *undistorted* deformation, under the same gravitational force, using quadratic strain.

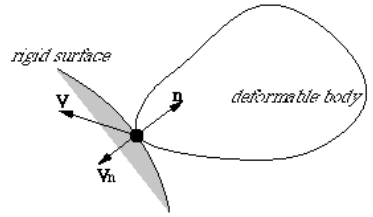


Figure 2:
A flexible body collides with a rigid body.

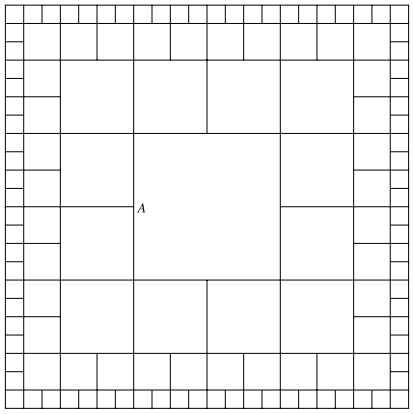


Figure 3:
A 2D example of graded mesh.

References

1. David Baraff and Andrew Witkin. Dynamic Simulation of Non-Penetrating Flexible Bodies, Computer Graphics: Proceedings of SIGGRAPH, pages 303—08, ACM, 1992.
2. Edward John Nicolson. Tactile Sensing and Control of a Planar Manipulator, PhD thesis, EECS, University of California, Berkeley, 1987.
3. D. Terzopoulos, J. Platt, A. Barr, and K. Fleischer. Elastically Deformable Models, Computer Graphics, 21, July 1987.

Global Deformation Using Nonlinear FEM

We apply the displacement-based finite element methods (FEM) to simulation of large motions and global deformations of deformable objects. A linear strain model leads to unacceptable distortion (Figure 1). To avoid this problem, we apply a quadratic strain instead (Figure 1). Essentially. this requires solving the following nonlinear system of differential equations

$$\mathbf{M}\ddot{\mathbf{u}} + \mathbf{D}\dot{\mathbf{u}} + \mathbf{R}(\mathbf{u}) = \mathbf{F} \qquad (1)$$

where $\mathbf{u}$ is the nodal displacements; $\dot{\mathbf{u}}$ and $\ddot{\mathbf{u}}$, the respective velocities and accelerations; $\mathbf{F}$, the external force; $\mathbf{M}$, the mass matrix; $\mathbf{D}$, the damping matrix; and $\mathbf{R}(\mathbf{u})$, the *nonlinear* internal force due to deformation. For a soft material such as live tissue, the material stiffness is small. This makes explicit the time integration scheme appropriate because we can take large time steps. In particular, we apply a Newmark scheme to equation (1). We diagonalize the mass matrix $\mathbf{M}$ and the damping matrix $\mathbf{D}$. This leads to a decoupled system of nonlinear equations, which requires no matrix inversions to solve.

Collision Integration Scheme

Simulating deformable object collisions using a penalty method[3] requires tiny time steps to produce visually satisfactory animations. A general impulse collision[1] is considered more efficient and accurate but still requires more computation than collision-free dynamics. For deformable object collisions, the collision time can be assumed finite (unlike the instantaneous collision of rigid bodies). By recognizing this, we propose an efficient way to handle collisions.

Consider the collision between a deformable body with a stationary rigid body (figure 2). Assume at time t_n, the node p, with velocity $\hat{\mathbf{v}}(p)_n$, is colliding with a rigid surface of outward normal $\hat{\mathbf{n}}$. Then the non-penetration constraint at node p can be enforced by setting the normal component of $\hat{\mathbf{v}}(p)_{n+1}$ to zero as following:

$$\hat{\mathbf{v}}(p)_{n+1} = \hat{\mathbf{v}}(p)_n + (\hat{\mathbf{v}}(p)_n \cdot \hat{\mathbf{n}})\hat{\mathbf{n}} \qquad (2)$$

If we choose $\triangle t_{n+1} = \triangle t_n$ for the Newmark scheme, we have

$$\hat{\mathbf{u}}_{n+2} \cdot \hat{\mathbf{n}} = \hat{\mathbf{u}}_n \cdot \hat{\mathbf{n}} \qquad (3)$$

This shows that the non-penetration constraint is exactly enforced after two time steps, *without* solving a constrained problem. For a decoupled system, this collision handling scheme can be easily generalized to multiple point collision constraint. And unlike the coupled system, no explicit impulse calculation is necessary for frictionless collisions. When friction has to be considered, the equivalent impulse can be easily computed without matrix inversion. This collision-handling integration scheme can be easily generalized to collisions between a deformable body and a moving rigid body, and to collisions between deformable bodies. It is worth noting that a deformable object's resting on a surface can be handled exactly the same way.

Graded Mesh

While 2D FEM has achieved great success in achieving real-time performance in computer graphics applications, the computational cost is much higher for 3D applications, mainly due to the increase in the number of elements in the mesh. In a roughly uniform 2D finite element mesh, the number of elements is about $O(n^2)$, where n is the average number of elements in each principle direction. However, a similar 3D mesh would have $O(n^3)$ elements, which leads to a much larger system of equations. Nicolson[2] shows that the cutoff spatial frequency of an object in response to external loads decreases faster than $1/d$ in terms of the distance d away from the surface. Therefore, if we use a 3D graded mesh similar to the 2D example in Figure 3, we will lose little accuracy with respect to static forces on the surface, while reducing the complexity of the problem from $O(n^3)$ to $O(n^2)$. A similar grading property also applies to tetrahedral mesh. To maintain the geometric compatibility at the element interface, we simply set the displacement of edge constrained nodes, such as node A in Figure 3, to the midpoint of the corresponding edge. Similar constraints apply to the face-constrained nodes in a 3D mesh. On a 400MHz Pentium II PC, a uniform mesh of 1331 elements needs about 0.11 seconds per time step. The graded mesh with the same accuracy needs only 0.06 seconds per step.

*Supported by a Multi-Disciplinary Research Initiative grant for 3D Visualization, sponsored by BMDO with support from ONR.

Real-Time Principal Direction Line Drawings of Arbitrary 3D Surfaces

Ahna Girshick
Victoria Interrante
University of Minnesota
{girshick, interran}@cs.umn.edu
www.cs.umn.edu/Research/graphics

Computer-generated line drawings,[8] are becoming an increasingly popular method of non-photorealistic rendering. While many factors contribute to an effective line drawing, line direction may be among the most critical for conveying surface shape. The principal directions hold particular promise in this respect.[2] In a recent study, human subjects consistently displayed biases toward perceiving surface contours (lines drawn on the surface) as being aligned with the principal directions.[5] Despite the potential for principal directions and their suggestion by previous authors,[1,2,7] there are currently no available techniques for using them to render geometrically invariant line drawings of arbitrary 3D surfaces. Markosian et al.[4] handled arbitrary 3D surfaces, but line direction was viewpoint-dependent and lines clustered on silhouette edges with little interior curvature information.

Interrante[2] used a 3D principal direction texture to show the shapes of transparent isosurfaces in volume data. Stalling[6] suggested line drawing based on 3D streamlines following the principal directions. Figure 1, derived by these methods, was a motivating factor for this research. However, this work focuses on the application to polygonal meshes instead of volume data. The major contribution of this work is to motivate the use of principal directions in line drawings and show how to apply them to arbitrary polygonally defined surfaces. The result is a geometrically invariant set of shaded strokes in 3D which can be rendered in real-time.

Estimating Principal Directions on a Polygonal Mesh

Much of the difficulty of this problem lies in accurately computing smoothly continuous estimates of the principal directions at arbitrary points on a potentially coarse polygonal mesh. The method we use was proposed by Taubin.[6] First, vertex normals are computed. Then for each vertex vi, we construct a symmetric matrix Mvi based on the weighted sum of the directional curvatures of the edges in the neighborhood of vi. The eigenvectors of Mvi are the normal and two principal directions, which together form an orthonormal basis at vi. To determine the principal directions, we restrict Mvi to the tangent plane and then diagonalize the 2x2 submatrix of the transformed Mvi. The principal curvatures can be calculated with a simple linear transformation on the corresponding eigenvalues.

Tracing Lines Over the Surface

The collection of estimated principal directions results in a vector field over the surface. We trace long curved strokes whose directions are determined by the flow of the vector field. To maintain an approximately even-line density and avoid line crossings. We extend the placement strategy of Jobard et al.[3] from 2D images to the 3D surface. Line starting points are chosen randomly on the surface with the constraint that they lie at a minimum distance from existing lines. In umbilic regions where the surface shape is planar or spherical, the principal directions are undefined because there is equal curvature in all directions. In such cases, we tri-linearly interpolate. If there are no neighboring well-defined principal directions, or if there is a sudden change in surface curvature, the line is terminated.

We implemented this technique in OpenGL using shading and hidden line removal. Rendering parameters, line density, length, or waviness, can be varied for different effects. Figure 2 shows the technique applied to a coarse triangular model of a vase.

Conclusions and Future Work

We are engaged in ongoing efforts to improve the principal direction estimation algorithm and apply this technique to more complex models. Future work will incorporate methods for maintaining continuity through umbilic regions and aesthetically merging opposing lines of force. The perceptual effectiveness of principal direction lines for the understanding of surface shape will hopefully be a further area of research.

Acknowledgments

This work was supported by a University of Minnesota Grant-in-Aid of Research, Artistry, and Scholarship. Thanks to Detlev Stalling and Todd LeMoine.

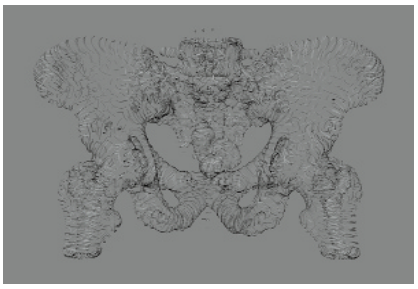


Figure 1 Principal direction line drawing (without hidden surface removal) of a bone/soft tissue boundary surface in a CT volume data set.

Figure 2 Principal direction line drawing of a vase. Mesh consists of 400 triangles.

References

1. R. Coutts and D. Greenberg. Rendering With Streamlines, Visual Proc. (SIGGRAPH 97), p.188.
2. V. Interrante. Illustrating Surface Shape in Volume Data Via Principal Direction-Driven 3D Line Integral Convolution, Computer Graphics (Proc. SIGGRAPH 97), pp.109-116.
3. B. Jobard and W. Lefer. Creating Evenly-Spaced Streamlines of Arbitrary Density, Proc. of 8th Eurographics Workshop on Visualization in Scientific Computing, 1997, pp.45-55.
4. L. Markosian, M. Kowalski, S. Trychin, L. Bourdev, D. Goldstein, and J. Hughes. Real-Time Nonphotorealistic Rendering, Computer Graphics (Proc. SIGGRAPH 97), pp.415-420.
5. P. Mamassian and M. Landy. Observer Biases In the 3D Interpretation of Line Drawings, Vision Research, 38 (1998), pp. 2817-2832.
6. G. Taubin. Estimating The Tensor of Curvature of a Surface From a Polyhedral Approximation, Proc. 5th International Conference on Computer Vision (ICCV), 1995, pp.902-907.
7. G. Winkenbach and D. Salesin. Rendering Parametric Surfaces In Pen and Ink, Computer Graphics (Proc. SIGGRAPH 96), pp.496-476.
8. D. Stalling. Personal Communication.

Tsukasa Noma
Kyoji Oishi
Hiroshi Futsuhara
Hiromi Baba
Takeshi Ohashi
Toshiaki Ejima
Kyushu Institute of Technology
noma@ai.kyutech.ac.jp

Real-Time Translation of Human Motion from Video to Animation

Figure 1: Walking sequence (top: video source; bottom: animation output)

Recently, considerable effort has been devoted to animating a virtual human from real human motion in video image sequences.[1] For real-time processing, however, existing systems have limitations in possible human postures and environmental/lighting conditions.

This sketch presents a robust top-down approach to translating human motion in video images to computer animation. In existing approaches, all the motion data are obtained directly from video images, frame by frame. Such a pure motion-capture approach often forces unnatural postures and/or causes sudden postural changes. In our approach, the system makes conjectures on what a real human subject is doing from the information taken from video images, and then motion generators[2] animate a virtual human smoothly and appropriately.

Approach

A real human subject is assumed to move on a horizontal floor, and a single video camera is placed above the floor. The camera data (for example, height, dip, etc.) are known in advance. Our prototype system is composed of two modules: a region detector and an animation generator. In the region detector, three blobs corresponding to head, trunk, and legs are tracked frame by frame. Our blob-tracking algorithm is similar to Wren et al.,[3] but we have improved the blob-labeling equations based on the Maximum A Posteriori Probability (MAP) method.

In our animation generator, possible human motions are categorized, and the motions and their transitions are modeled as a state/motion transition graph. Its conditions for state transition are described with relative positions of the blobs. The animation generator first computes the horizontal position of the human body assuming that the bottom of the legs blob touches the floor. Next, from the relative positions of the blobs, the system infers the currently performed motion by making transitions on the state transition graph.

Even if the system fails to track some blobs, our system makes rough conjectures on the human posture by treating all the remaining blobs as a single blob representing the whole human body. With this feature, our virtual human is animated even if the human figure in the source images is on the order of several pixels. Our approach is thus applicable to a much larger environment than existing approaches.

More technical information is available on the Web.[4]

Results

If the input video image size is 160x120, experiments show that our system can generate about 11 frames per second on a single SGI O2 (R5000). The animated virtual human can walk, sit, and lie as the real human subject performs in an uncontrolled environment like a computer room. (See Figure 1 and additional materials on the Web.[4])

Conclusion & Future Work

Our system can translate human motion in video images to computer animation in a more robust fashion than existing methods. To animate the human as robustly as possible, our current implementation has some frame delay between real video images and animations. This design decision is made because the system was primarily developed as a privacy-oriented surveillance system, where its observer can understand what its subject is doing without knowing who the subject is. In such a system, some frame delay would be acceptable. To apply our system to other applications (for example, interactions in virtual environments), the delay will be minimized in the near future.

Acknowledgement

This work was supported by ANIMA Electronics Co., Ltd.

References

1. K. Ebihara. Shall We Dance? SIGGRAPH 98 Conference Abstracts and Applications, 124, 1998.
2. T. Noma and N.I. Badler. A Virtual Human Presenter, Proc. IJCAI-97 Workshop on Animated Interface Agents, 45-51, 1997.
3. C.R. Wren, et al. Pfinder: Real-time Tracking of the Human Body. IEEE Trans. on PAMI, 19(7):780-785, 1997.
4. Human Motion Translator Web page. www.pluto.ai.kyutech.ac.jp/~noma/mottrans/

Rendering 3D Objects into Photographs Taken by Uncalibrated Perspective Cameras

Qian Chen
Gérard Medioni
University of Southern California
qianchen@usc.edu
csuri.usc.edu/~chenqian/research.htm

One important element in virtual/augmented reality applications is the insertion of synthetic objects into real images. For this purpose, the camera pose must first be estimated. Typical approaches rely on 3D position tracking devices, precise calibration, or fudicials. Recently, affine invariant has been used to provide calibration-free augmented reality. However, the affine representation makes it difficult to describe the objects in a traditional way (using positions, distances, or angles, etc). Another aspect of the same problem, which has been largely ignored, is the mutual occlusion (or other geometrical interactions) between the virtual (inserted) and the real (existing in the scene) objects. To make this happen, accurate Euclidean models of the real objects must also be recovered.

In our research, we extended the idea of using invariance to perspective cameras, adopting a more general concept: projective reconstruction. We developed new self-calibration algorithms to transform the projective reconstruction into a Euclidean one to gain the freedom of conventional object placement while bypassing the calibration process. Rather than fudicials, image features such as corners of objects were used. As a result, object models and camera pose were obtained at the same time. Finally, special attention was paid to take care of registration errors caused by the deviation in feature detection. The overall workflow has four steps:

1. Monocular image features are detected automatically.

2. Correspondences across images are established interactively.

3. Object structure and camera pose are recovered.

4. Original images are augmented with inserted objects that may cut into the real objects.

In a projective framework, camera projection can be described by a three-stage process: projective-to-Euclidean, Euclidean-to-camera, camera-to-image. We have developed a novel projective reconstruction algorithm using matrix factorization. We have also developed a new self-calibration algorithm that separates the projective-to-Euclidean matrix from the projective reconstruction, thus giving the Euclidean reconstruction. This result is shown in Figure 1. However, using the so-far-obtained cameras for object insertion generates images with severe misalignment (left two images, Figure 2). The error comes from the deviation in feature detection. We addressed this problem by introducing a 3D affine distortion matrix right before the camera-to-image projection. This preserves the Euclidean structure of the world. The final result is shown in the right two images of Figure 2.

Figure 1
Two examples of a terminal sequence, its Euclidean reconstruction together with the recovered cameras.

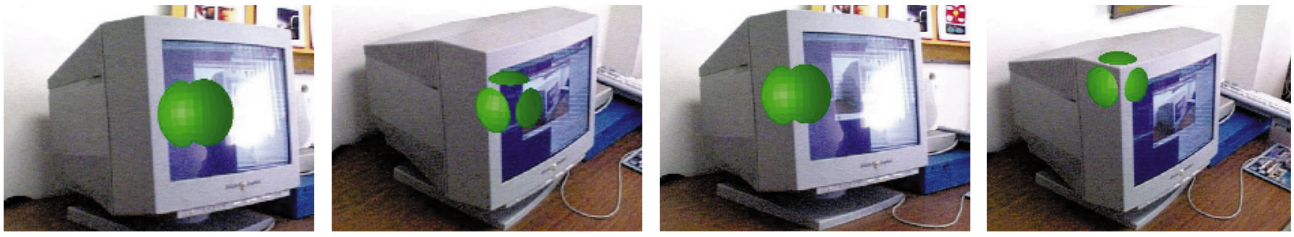


Figure 2
Misalignment of real and virtual objects (left), and correct registration after error compensation (right).

Perspective Cameras

Teruaki Iinuma
Hideki Murota
Yasuo Kubota
Dai-Nippon Printing Co., Ltd
Inuma@lab.cio.dnp.co.jp

Hiroshi Tachihara
High-Tech Visual Promotion Center

Representation of the Tactile Surface Texture of an Object Using a Force Feedback System.

The "digital archives project" is being pursued worldwide. This project preserves information on cultural properties such as national treasures, important cultural properties, and works of art as digital information. Because it is saved as digital data, the information can be used indefinitely.

Our appreciation and enjoyment of objects that have a significant 3D structure (for example, sculpture, statues of Buddha, traditional ceramic ware, etc.) can be greatly enhanced by not only being able to look at them, but also to touch and move them. Such virtual tactile feedback systems have been developed with force feedback control technology and CG display technology. But these systems have difficulty representing the physical texture of rough surfaces on an object.

In this approach, the force exerted on the fingertip by the force feedback system is calculated using the location of the finger tip relative to the polygons representing the object. The force is calculated by the normal vector of the polygon, so the same force is felt everywhere on the polygon. For example, when the system displays a "brick," we expect a rough texture, but the current system can not provide feedback for this. So we developed a method to allow the fingertip to feel rough texture using the virtual tactile feedback system.

The dynamic frictional resistance feedback to the fingertip in the system is calculated as follows:

Fd = - Abv
Fd:dynamic frictional resistance
A:constant
b:dynamic frictional resistance coefficient(0.0-1.0)
v:fingertip velocity vector projected onto the polygon surface

This equation means that if the fingertip moves faster on the object surface, more feedback force is returned. If the dynamic friction resistance coefficient is set to be large, the fingertip feels more resistance force from the system. We simulate the texture of the surface by alternating, at 100Hz, between two dynamic frictional resistance coefficients. We used a "brick" mapped with a finely textured surface. The two coefficients for this surface, FD1 and FD2, are changed by increments of 0.2, so 15 combinations are evaluated (FD1 = 1.0-FD2 = 0.8 to 0.0, FD1 = 0.8-FD2 = 0.6 to 0.0, FD1 = 0.6-FD2 = 0.4 to 0.0, FD1 = 0.4-FD2 = 0.2 to 0.0, FD1 = 0.2 -FD2 = 0.0). The results showed that when the difference of the two values is less than 0.2, the texture feels "sticky." When the difference is 0.4 to 0.6, the texture feels "rough." And when the difference is 0.6 to 0.8, the texture feels "jagged." These results show that alternately applying the two coefficients to the object can enable us to feel subtle surface texture.

The fingertip moves across the surface against the two alternating resistance forces. If the resistance force is large, it is difficult to move. If the force becomes small, it becomes easy to move. If the difference of the two alternating forces is larger, the move distance with small resistance force becomes large. So you feel a large rough texture (jagged). On the the other hand, if the difference of two forces is smaller, the move distance against smaller resistance becomes small, so you feel a small, fine, rough texture.

We used Sensable Technology Inc.'s PHANTOM as the force feedback equipment.

Reference
High-Tech Visual Promotion Center Foundation development committee, annual report 1997 (March 1998).

Shading and Shadow Casting in Image-Based Rendering Without Geometric Models

Akihiro Katayama
Yukio Sakagawa
Hiroyuki Yamamoto
Hideyuki Tamura
Mixed Reality Systems Laboratory Inc.
katayama@mr-system.co.jp
www.mr-system.co.jp/

Image-based rendering (IBR) methods such as light-field rendering[1] and the ray-space method[2] succeed in photo-realistic representation of complex objects without geometric models. Now the key to IBR's future is how to accomplish functionalities of conventional geometry-based approaches. We have already reported how to interact with image-based objects in the virtual environment.[3] The next goal is how to make objects without polygon models react to transitional lighting conditions. This sketch describes a real-time rendering method that changes the shading of image-based objects and casts appropriate shadows according to the motion of viewpoint or objects and transitions in local lighting.

Method

A set of multiple pictures taken by a moving camera under a specific lighting condition is called an imageset. Our idea is based on a simple method of selecting the imageset with the lighting condition nearest to that of the virtual environment from the imagesets taken under various lighting conditions. Let I_c be the imageset taken under ambient light. An imageset $I(l)$ stores only the luminance component of pictures of real objects taken under a lighting condition l. While changing values of l, the image data are stored into the corresponding imageset. To render an image from a given viewpoint, corresponding pixels are extracted from $I(l)$ and I_c. Then, the chrominance part of I_c and luminance part of $I(l)$ are applied to the pixel in order to get the final shading. Note that as the technology of IBR is used in this process, an image seen from any point of view can be reproduced. The shadow cast by an object is represented by the texture mapping of the image contour, which is viewed from the light position, onto the surface the object resides on as shown in Figure 1.

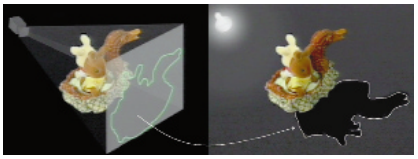


Fig. 1 Generation of cast shadow

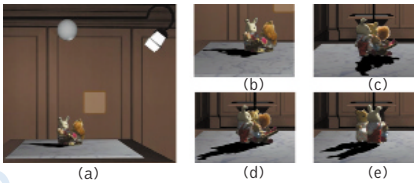


Fig. 2 Views of the objects under different lighting conditions

Implementation

A system called CyberMirage 99 places ray-based data in the geometry space. A light source can be switched on and off, and moved. Of course, translation and rotation of objects and observers' walk-throughs are implemented. Note that all of these functions are realized in real-time. Figure 2(a) shows an image of objects lit from a certain light source. Note that the objects are appropriately shaded and cast an appropriate shadow. Figure 2(b) is the image when an observer comes near to the objects, and Figure 2(c) is the image when an observer approaches another side of the objects. Figures 2(d) and 2(e) are images when the light source is moved and the objects are rotated, respectively. These figures show the effect when the shading and cast shadow are appropriately adjusted.

Future Work

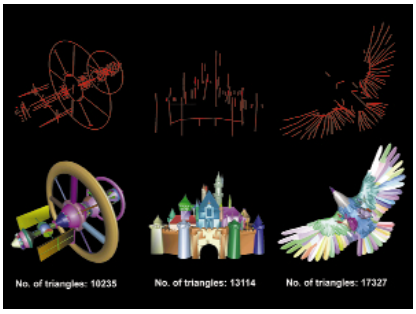
The next step of our study will explore how to expand the system so that it can accommodate multiple light sources and react to the change in intensity of light. Theoretically, this can be accomplished by preparing a lot of imagesets corresponding to the number of light sources and their intensity levels. However, it requires a new horizon of data-compression methodology in order to realize the real-time interaction.

References

1. M. Levoy and P. Hanrahan. Light Field Rendering, Proc. SIGGRAPH 96, pp.31-42, 1996.
2. T. Naemura, et al. Ray-Based Creation of Photo-Realistic Virtual World, Proc. VSMM'97, pp.59-68, 1997.
3. S. Uchiyama, et al. Presentation and Interaction of Virtual 3D Objects Without Geometric Model, Proc. HCI'97, Vol.2, pp.869-872, 1997.

Tong-Wing Woon
Xuetao Li
Tiow-Seng Tan
National University of Singapore
woontong@comp.nus.edu.sg

Shape Extraction for a Polygon Mesh



This sketch reports on our new technique for deriving geometric shape for both organic and inorganic models. This information can be used in computing data structures related to the object (for example, bounding volume hierarchy for collision detection and ray tracing, and level-of-detail for different parts of an object).

Several previous studies have reviewed generation of polygon meshes from a given point set.[1] However, given a polygon mesh, there are no well-known studies of computing a good hierarchical arrangement of polygons. A portion of our work[2] is the first attempt in this direction. Though the work has achieved good results in partitioning objects efficiently into smaller parts, it is sometimes affected by user parameters and modeling variations. In this sketch, we seek to devise a technique that is automated, efficient, and robust with respect to modeling variations.

Shape Extraction

Shape extraction involves two parts: computing atomic parts (each consisting of a set of polygons) that make up the input model and arrangement of these parts into a hierarchy. The proposed algorithm addresses these using a skeleton derived from the model.

Atomic Computation

A skeleton of a given model is a collection of vertices and edges resulting from a sequence of operations (such as edge contraction) that collapses polygons of the model. Each element e of an edge or vertex of the skeleton is associated with a set of polygons $T(e)$ of the given model collapsed to e. The union of all $T(e)$'s of a connected part of the skeleton is considered an atomic part of the model and is a leaf in the shape hierarchy, H.

Our previous work on using a simplified model derived from model simplification[2] is a variant of a skeleton. In another development,[3] a series of edge-collapse operations is applied to the input model to obtain a so-called skeleton for a different purpose. Our experiment on directly applying this algorithm to compute the atomic part was not satisfactory, in particular, for organic objects. Its drawback is absorption of small (but distinct) features of a model into bigger components. Our method uses weights to control the order of collapse and makes use of previous edges to guide the collapsing.

Hierarchy Building

Each atomic part is now a leaf of the shape hierarchy H. The next step is to link these leaves into the top few levels of H. Construct G, where each vertex in G represents an atomic part and each edge connects two vertices if their associated set of polygons shares some edges or vertices. Assuming G is a single tree, hierarchy building begins at the leaf nodes of G by "folding" up until a single node is reached. At each step, create a node m in H to contain the leaf nodes and their immediate predecessors, and delete the leaf nodes from G.

The resulting H may not be binary and can be made binary with an approach[2] for computing bounding volume hierarchy. Also, similar ideas can be applied to connect various disconnected graphs G (each can be made as a tree by removing edges in cycles) from the skeleton.

Preliminary Results

Our preliminary results show good partitioning of geometric models, both organic and inorganic. The figure shows examples of skeletons (first row), and atomic parts (second row, where each part is shown in a different color) obtained by the current implementation. Future work includes the use of the shape information for various applications such as bounding volume hierarchy computation.

References

1. Amenta et al. A New Voronoi-Based Surface Reconstruction Algorithm, SIGGRAPH 98, pp 415-421.
2. Tan, Chong and Low. Computing Bounding Volume Hierarchy Using Simplified Models, Proc. Symp. on Interactive 3D Graphics, April 1999, pp. 63-69.
3. Deussen et. al. An Illustration Technique Using Intersections and Skeletons, Proc. Graphics Interface, June 1999.

Speedlines: Depicting Motion in Motionless Pictures

Maic Masuch
Universität Magdeburg
masuch@isg.cs.uni-magdeburg.de

How do we present the motion of objects in computer-generated still images? Generally, we don't. Or, if we do, we use motion blurring to simulate a real-world camera.[1] Motion blur costs a great deal of extra rendering time, and the only thing it does is blur the objects so that their contours are unrecognizable. The goal, to convey information about the movement per se, is not entirely met. If, however, we consider the application of non-photorealistic rendering techniques, it is promising to adopt successful illustrative techniques from comics to depict past and future motions of objects in a single image.

Presenting Motion

Speedlines are an important stylistic element in comics, and although they are not based on an exact physical model, their use is well known to all of us.[2] Other means of depicting motion in still images include motion arrows and contour repetitions. In the latter case, earlier positions of an object are drawn with less detail or slightly faded, whereas the current state of the object is presented in all detail.

Besides using those techniques in still images, their use has also been adapted in many classical animations.[3]

Generating Speedlines, Contours, and Arrows

For an automated generation of speedlines, arrows, and contour repetitions, models coming from typical animation systems (such as 3D Studio) provide all necessary data. After executing a rendering pipeline for line drawings[3], we used 3D information gathered during the rendering in order to establish the appropriate visibility and the perspective correctness of the speedlines. Here, a 2D-based approach would be futile, since we have to keep track of the contour of every single object. We then calculate a motion path from the given keyframe data.

As these speedlines were invented by human artists, their generation requires some heuristics about where and how to draw them. In general, speedlines and contour lines:

- Are drawn in the opposite direction of the movement (thus reaching into the past).
- Start at "characteristic" points of the moving object.
- Embrace the minimum and maximum extent of the object.
- Are more or less equally sized, shaped, and directed, without intersecting each other.
- Are uniformly, but not too regularly distributed.

Candidates for starting points of speedlines are vertices of the 3D mesh which yield the outline of the object. The algorithm divides the shape into a number of stripes where each stripe can hold one speedline. If a stripe contains no such vertex, additional vertices are generated on the object's outline. If we do not restrict ourselves to the outline and also take into account inner contour lines, the result can be further improved. In addition, the speedlines should not stick to an object: the algorithm should draw them with an offset. The computation of these effects takes only a small fraction of the rendering time.

Drawing Lines with Style

The output of the renderer consists of a vector-oriented description of the image. Actual lines are drawn using line styles which, in turn, simulate hand-drawn pen strokes by the superposition of the drawing path and the chosen style deviations.[4]

Although our system supports parameter control in every detail, only very little user interaction is necessary, since nearly all speedline parameters can be computed based on the animation data. Therefore, length, width, distribution, number, etc. are determined by applying given default rules.

Some Results

The pictures show different models with speedlines applied to them. We have chosen to render quite simple models to concentrate on the effects of speedlines and contour repetitions. Also, we applied line styles only to the speed elements. Further examples including a short animation can be found at isgwww.cs.uni-magdeburg.de/~masuch/gallery.html

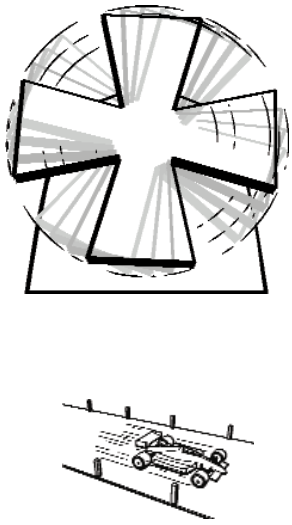


Figure 1

References

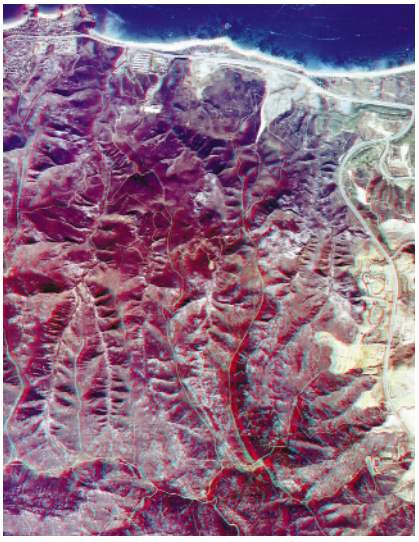
1. M. Potmesil, I. Chakravarty. Modeling Motion Blur in Computer Generated Images, In SIGGRAPH 83, pp. 389-399.
2. S. McCloud. Understanding Comics - The Invisible Art, Harper Collins, New York, 1993.
3. F. Thomas, O. Johnston. The Illusion of Life: Disney animation, Hyperion, 1981.
4. T. Strothotte et al. Computational Visualization: Graphics, Abstraction, and Interactivity, Springer, 1998.

Jason Rosenberg
Gena Hillhouse
Younian Wang
ERDAS, Inc.
jason@erdas.com
www.erdas.com

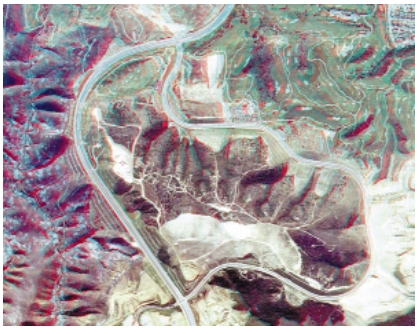
Stereo Analyst: Visualizing Large Stereoscopic Imagery in Real-Time

Stereo Analyst is a new software package designed for visualizing stereoscopic imagery in real-time and for collecting 3D features in stereo. The initial targeted users for this product are those who want to visualize terrestrial image data, both in the form of aerial photography and satellite imagery. However, the tool is quite practical for visualizing close-range photography as well. Stereo Analyst includes a set of editing tools for creation of feature databases, such as road maps and 3D building outlines.

One of the primary requirements for the application is that it provide smooth real-time pan and zoom navigation. It utilizes stereo viewing hardware, such as liquid-crystal shutter glasses, in combination with OpenGL's interface for creating stereo-in-a-window on the desktop. If stereo viewing hardware is not available, the system will also render the imagery using a color anaglyph mode.



Color anaglyph stereo screen capture from Stereo Analyst. The image is best viewed using a pair of red-blue glasses, in order to see the 3D stereoscopic effect. The imagery is from a pair of high-altitude aerial photographs taken near Laguna Beach, California.



The tool allows the user to display any pair of images with overlapping areas in stereo. The user can position, rotate, and scale each image interactively relative to the other, so that a proper epipolar alignment can be achieved, and stereoscopic viewing becomes possible.

If camera position is available for the imagery, Stereo Analyst will automatically provide on-the-fly epipolar projection for the images. This is accomplished by using OpenGL's 3D transformation pipeline, coupled with hardware texture mapping, ensuring that this occurs interactively in real-time.

Another requirement for this application was that it be able to handle arbitrarily large volumes of image data. In this case, it is clearly not possible to read an entire image into main memory and then to texture memory. The problem is further complicated by the fact that with stereo imagery, there are two image sources, so twice as much texture data is needed at any given time. A texture-caching scheme is used, whereby only image data for the current area of interest is paged from disk. A multi-threaded design is used to separate slow disk access from the interactive rendering thread.

Multi-resolution decomposition of the data is used to optimize real-time performance. In this case, we want to render the data at reduced resolution when the available screen resolution is less than the available resolution of the data to be rendered from the current viewpoint. For the imagery data, we pre-process the image files to create pyramid layers, which can be thought of as a pyramid of sub-sampled versions of the imagery, with each successive level providing a power-of-two reduction factor.

Interactive fallback modes allow the application to perform well on a full range of graphics hardware. For instance, if only a small amount of texture memory is available, the system can be set to use reduced resolution data while the data are in motion, such as during panning and zooming. When the movement is halted, the full resolution is rendered, possibly at a sub-real-time redraw speed.

Stereo Analyst was developed using OpenGL, and runs on the Intel-based Windows operating systems. It is designed with a "plugin-able" architecture, meaning that it is easily extensible for domain-specific applications, both in terms of imagery sources and overlaid data collection and visualization.

Tangible Modeling System

Ken-ichi Kameyama
Toshiba R&D Center
kenichi.kameyama@toshiba.co.jp

A 3D CAD system, in which users can feel as if they actually deform a real mockup and directly paint some pictures on it, has been developed using the following techniques.

One is the simple augmented-reality technique whereby a user's real hands and virtual objects are made to appear to self-occlude each other. The system uses a physical prop with the rough shape of the virtual object as an input device. A half-silvered mirror with 30-percent transmittance and a 3D tracker in the input device are used as the corresponding image of the input device is always overlaid on it.

There were many systems with this structure,[1,2,3] but only the tangible modeling system has realized appropriate occlusions between real and virtual objects in a very simple manner. To achieve this feature, the system utilizes the fact that a human cannot distinguish a darker object or image when a brighter image or object is overlaid on it.

The input devices and the walls inside the system are coated with deep-black cloth called "Haimiron" and illuminated with spotlights. The brightness of an object covered with Haimiron is 0.6-1.2 [Cd/m2] when it is lighted up with the spotlights, whereas the brightness of human hands in the same condition is 28.6-57 [Cd/m2]. Therefore, the user's hands in the workspace can be seen through the half-silvered mirror, but the devices and the walls inside cannot. Meanwhile, the brightness of the white image (reflected on a half-silvered mirror) is 12 [Cd/m2]. The brightness of human hands overlaid with the white image is 21-30 [Cd/m2]. Therefore, hands can occlude the image to some extent when they are in front of the input device (the virtual object). Hands are automatically occluded when they are behind the input device (Figure 1).

Another technique of this system is that the input device has a force/torque sensor for directly measuring hand action for object deformation. Currently, the sensor can detect hands as they twist, stretch, shrink, bend, or shear. The actual shape of the device cannot be changed, but its corresponding image is deformed using the FFD (free-form deformation) algorithm along with the classified patterns. Users can passively feel the resistant force from the device when they hold, push, and deform it.

In one previous system, the hand device can be regarded as a virtual object itself,[4] but such a system does not support object deformation. The tangible modeling system can be applied to direct shape modeling and 3D painting or sticker pasting for early industrial design (Figure 2).

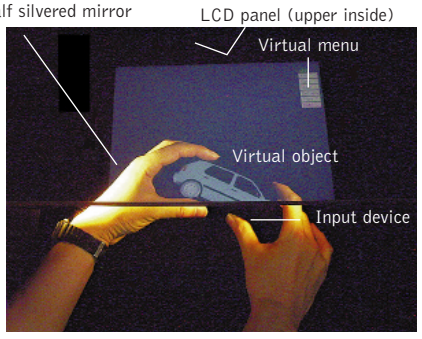


Figure 1: Integration of user's hands and a graphic image.

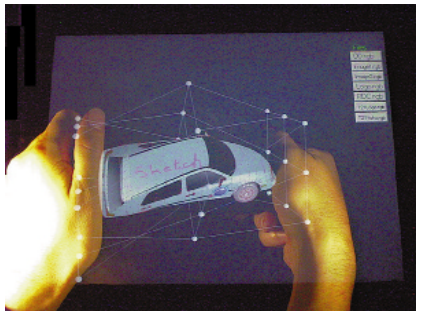


Figure 2: Interactive deformation of a 3D model.

References

1. Schmandt, C. M. Spatial Input/display Correspondence In a Stereoscopic Computer Graphic Work Station, Computer Graphics, 17(3), 253-262, 1983.
2. Ullmer, B. and Ishii, H. The MetaDESK: Models and Prototypes for Tangible User Interface, Proc. UIST'97, 223-232, 1997
3. Poston, T. and Serra, L. Dexterous Virtual Work, Communication of the ACM, 39(5), 37-45, 1996.
4. Goble, J. C., Hinckley, K., Pausch, R., Snell, J. W., and Kassel, N. F., Two-handed Spatial Interface Tools for Neurosurgical Planning, IEEE Computer, 28(7), 20-26, 1995.

Victor B. Zordan
Jessica K. Hodgins
Georgia Institute of Technology
victor@cc.gatech.edu

Tracking and Modifying Human Motion with Dynamic Simulation

The top row shows a simulated character and the human actor whose motion is being tracked. The second row shows a variety of tracked behaviors. The third row shows an example of combining balancing and tracking on the left and simulations playing patty cake with dynamic impacts on the right.

Subtle details in the motion of animated, human-like characters affect the believability, aesthetics, and impact of an animation or virtual environment. However, none of the available techniques for animating characters make including these stylistic details in the generated motion easy. Motion capture produces characters with rich detail in their motion, but the data are difficult to modify for new characters and situations. Dynamic simulation generates physically correct motion for characters that can respond interactively in a changing environment. However, the controllers required for simulated characters are difficult to construct because we do not know how to specify the details of human motion procedurally.

In this sketch, we describe a technique that uses a dynamic simulation to track and modify motion capture data of human upper-body movements. By combining simulation and motion capture, we hope to retain the interactivity and realism of dynamic simulation and the subtle details of the human data while avoiding the disadvantages of each approach.

Our system uses motion-capture data as the desired inputs to a tracking controller for a dynamic simulation of the character. The simulation is created using the mass, moment of inertia, and limb lengths of the animated character. Thus, the generated motion is physically correct for that character. Driven by the tracking controller, the simulation follows the input motion with similar trajectories, thereby maintaining the style of the human data. This system was used to perform a variety of gesturing and signalling behaviors for several humanoid models.

Three additions to this basic system allow it to be used to generate a richer variety of behaviors. First, a collision handler is added to generate realistic dynamic impacts such as hands clapping in a game of patty cake. The dynamic model applies reaction forces, and the hands collide in a believable way.

Second, specialized task controllers are added to allow editing at the task or behavior level. As a simple example, consider the task of moving a staff up and down rhythmically. Kinematic differences between the motion capture subject and the animated character will cause the staff to wave around rather than move vertically. A task controller fixes this problem by setting the desired angle of the wrist to keep the hand in the correct orientation. The remaining degrees of freedom of the upper body track and maintain the style of the input motion. In a similar way, the system uses a balance controller for the degrees of freedom in the lower body and tracks motion capture data for the upper body to produce full-body motion.

Finally, by modifying the input data, our technique allows the user to edit motions at a high level while relying on the simulation to maintain physical correctness. We used a patty cake example to experiment with several such modifications. We adjusted the input data for the arms with inverse kinematics to ensure contact. Using this input, the tracking system generated physically plausible clapping sequences. We generated arbitrary clap sequences by re-ordering and time scaling segments of the input data. The system created physically realistic transitions between two distinct sequences by interpolating between two input motions and tracking the resulting data. The transitions are smooth because the resulting sequence obeys a consistent set of physical constraints for the duration of the motion.

Our preliminary results demonstrate that this approach to combining simulation and motion capture data maintains the style of the original motion while allowing the data to be modified for new characters and situations. Animations and a technical report with details on the system implementation, results, and related research can be found at: cc.gatech.edu/gvu/animation/Areas/datadriven/datadriven.

Video Embodiment - MovieSpiral: Towards Intuitive/Comprehensive Interfaces for Digital Video Interaction

Akihito Akutsu
NTT
email: akutsu@aether.hil.ntt.co.jp

Our new paradigm of handling digital video means moving from computer-oriented forms to user-oriented forms. In the computer-oriented approach, digital videos are represented just as a simple dataset, which means they have titles and codes. Adding semantic script information makes the datasets more user-oriented, but we feel that this is insufficient. A suitable body, a 3D representation, is needed to realize an attractive and powerful support system for human interaction.

Goals of Video Embodiment
Video embodiment is an attempt to create a body that expresses video content and structure. The most important issue is how to reflect video content and structure in the shape of the body. The shape should be compact while expressing the information that allows us to judge how suitable the video may be. The body should also simplify editing of the material.

Our Approach
This sketch presents our vision of video embodiment and introduces design activities including MovieSpiral to illustrate our key concepts:

1. Automatic feature extraction and semiautomatic structuring, to extract the important implicit features of video, and to construct structures using features and knowledge.

2. Transformation rule, to define how to transform the extracted features into an attractive and compact shape.

3. Direct and intuitive operation, to realize more direct control of editing operations with direct feedback.

4. Basic function of editing and processing, to design interaction tools that offer comprehensive and intuitive functions.

DNA-Like MovieSpiral
In nature, deoxyribonucleic acid (DNA) is a very compact form of sequential information. It consists of spirals within a super spiral, so it is spatially compact and easy to access. We shape virtual video to imitate DNA, as described below. A video is a linear sequence, but it has a hierarchical structure consisting of story, scenes, and shots. The DNA-like spiral structure is created by making first-order loops for cuts and second-order loops at scenes. The size of each shot loop represents the length of the corresponding shot, and the size of a scene loop represents scene length. Figure 1 shows the MovieSpiral of a commercial message. The total number of shots is 18, and the story consists of one scene. Another sample is a famous Japanese movie. The movie consists of 349 shots and 7 scenes.

Features
- Compact representation and montage visualization. When we see an entire MovieSpiral, we can grasp the technique the director used in making the video.
- Seamless interaction. We can obtain more detailed information. The MovieSpiral provides clues about content and structure, so it is hard to lose your way in the video information space.
- New interaction-style viewing operation. MovieSpiral is compact and expresses a lot of information that may allow us to discover attractive videos. We move around in space and watch and touch the shape to get more information.
- Identification. In fact, it seems possible to identify the style of a movie from the shape of its MovieSpiral. For example, a shape consisting of many short shots means an action scene. We may find that some directors produce works that have their own distinctive MovieSpirals.
- Manipulation based on MovieSpiral. The core of manipulating video is to interact with temporal information. MovieSpiral provides an environment in which video can be manipulated as it provides a better understanding of content and structure.

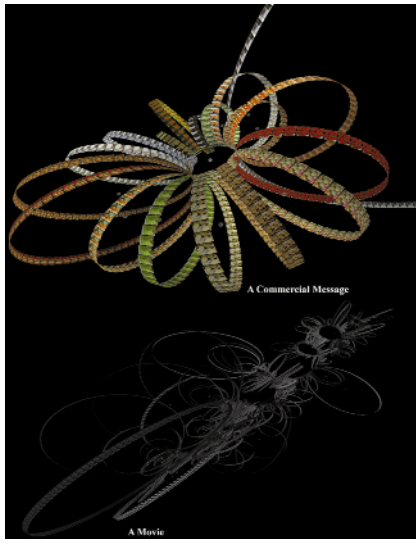


Figure 1
The MovieSpirals of a commercial message and a movie.

Christoph Stratmann
ART+COM Medientechnologie und Gestaltung AG
strat@artcom.de

Virtual Car

This sketch describes development of a sales support tool that utilizes virtual reality (VR) technology. The project was commissioned by DaimlerChrysler.

Objective
The Virtual Car system presents all possible variants of the vehicle. As an alternative catalogue, it includes all extras, colors, and fabrics, and it is intended to support every customer's individual Mercedes-Benz configuration. Therefore, a very high standard of visualization of the vehicle and its features is of cardinal importance. An important aspect of the system is its employment in the showroom, where it is intended for long-term use. It must not look and perform like a computer terminal, but must incorporate a user-friendly interface. Ergonomic criteria such as comfort and usability were given special consideration.

Figure 1
Fundamental concept

Fundamental Concept
At first, the basic concept was: "One World - One Window." In the showroom, the real vehicle is replaced by a virtual car at 1:1 scale (Figure 1). Using a flat touchscreen mounted to a swivel arm, the user navigates through the car's interior and exterior. The visual representation is updated in real-time in coordination with the user's movements. The touchscreen enables further interaction with the virtual car.

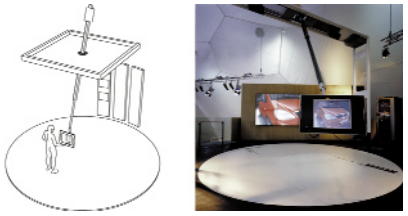


Figure 2
Hardware user interface design

Hardware User Interface Design
Traditional VR is still characterized by helmet and glove. In our case, the intended purpose of the tool did not allow that interface solution. It would have meant an unacceptable compromise of comfort, and it would have excluded the user from the real ambience.

Since all other commercially available input and output devices were not acceptable, a novel type of interface had to be developed for the Virtual Car application. In contrast to the classical approach in VR, the user is not immersed in the virtual environment. Instead, the virtual representation becomes immersed in the real world. Utilizing a LCD with integrated touch screen that can be freely moved in 3D space, the user experiences the car through a window into the virtual world (Figure 2). The display is framed and mounted to a swivel arm, and the 20-inch LCD (1280x1024 pixels, 16 million colors) displays almost true colour. The sensors in the axes of rotation record movements within six degrees of freedom. In order to allow operation of the touch screen the swivel arm is automatically locked in any position when the user lifts a hand to operate the touch screen.

Figure 3
Car configuration

Interactivity
Design of the integrated hardware and software systems was closely coordinated throughout the development process. All aspects of the car may be examined in detail. Using the touchscreen, the user can re-configure the car to individual specifications. If the user selects a piece of equipment, it is shown all options also available for the real car. Everything from colour, engine, wheel rims, and seats, down to the style of the gear shift knob can be selected according to the user's individual taste (Figure 3).

Figure 4
Employment of the system

Software
Special authoring tools were required for compilation of the 3D model data and the essential interaction programming of the actual application. Intelligent culling tools permit interactive presentation of the model at a minimum of 12 frames/second. For each frame, one graphic pipe of an SGI Onyx2 InfiniteReality System visualizes an average of 350,000 textured polygons.

Outlook
In the next Virtual Car product cycle, the ergonomics of the integrated input and output device will be further improved. The system will also be integrated into the sales infrastructure of the manufacturer, and the process of data integration will become more cost-efficient.

Reference
1. "The Virtual Car Project" (www.artcom.de/projects/vrf).

Volumetric Modeling of Artistic Techniques in Colored Pencil Drawing

Saeko Takagi
Tokyo Institute of Technology
sae@cs.titech.ac.jp

Issei Fujishiro
Ochanomizu University

Masayuki Nakajima
Tokyo Institute of Technology

Little attention has been given to colored pencil drawing (CPD), which has been regarded as secondary in relation to painted or finished work. However, CPD has become an artform in its own right, and it is often used for package illustration and picture books.

A CPD-like image could probably be generated by combining various functions provided by existing digital painting systems. A number of common CPD techniques exist,[1] and direct and faithful reproduction of these techniques is essential in order to represent the charm of CPD.

We have consistently taken a volume graphics[2] approach to modeling of CPD. Our previous CPD model consists of two sub-models.[3] One describes the 3D micro-structure of paper. The other defines a method for pigment distribution, formulated as the interaction between the paper and the lead of a pencil. In this sketch, we incorporate into the model pigment redistribution sub-models for two common CPD techniques: watering and adding eraser effects.

Watering
When using water-soluble colored pencils, watering is a real CPD technique for redistributing pigment by running a wet brush over a CPD-surface. After watering, a certain amount of pigment is spread and blended, depending mainly on the uneven conditions of pigment distribution, the quality of the bonding agent in the pigment, and the quantity of water.

In this model, voxels accessible from a brush are located using volume accessibility. If a pigment voxel in the stroke region is accessible, the pigment in the voxel dissolves (becomes a "source voxel"). Pigment in a voxel can also begin to dissolve if the voxel touches a source voxel, depending on the stroke pressure and the bonding agent in the lead.

After all the source voxels are located, the brush stroke vector is used for each of the voxels as a local streamline for our 3D extended version of Cabral's line integral convolution,[5] and the pigment movements are simulated. For each "sink voxel," in which the dissolved pigment collects, the contribution weight of the source voxel is determined according to the distance between the two voxels.

Using this sub-model, we added watering effects to a CPD volume, which was generated using our previous CPD model.

Eraser Effects
Among many CPD eraser techniques, we put a particular focus on the reproduction of the soft tint effect, because this technique has no substitute and is therefore imperative for creating CPDs with a soft impression. A soft tint is obtained by applying lightweight eraser strokes to CPD surfaces. Volume accessibility again plays a key role in evaluating which voxels are accessible from the eraser. In this case, an eraser is assumed to move only in the vertical direction. The top few layers of accessible pigment voxels are emptied by an eraser stroke. The thickness of the emptied layers is estimated according to the softness of the eraser and the pressure of the eraser stroke.

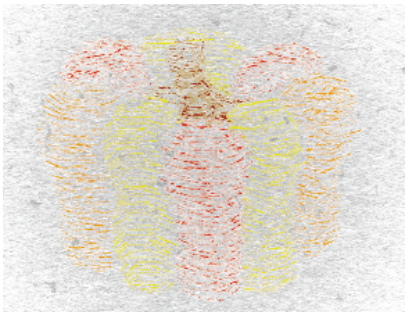


Figure 1
A volume visualizer called VolVis6 was used for volume rendering. The original pencil stroke looks somewhat scratchy, whereas pigment extended smoothly by the wet brush stroke results in the expression of a gentle feeling similar to real water-soluble CPDs.

Figure 2
A CPD volume in which an eraser effect was added around the tip of a leaf. In addition to the scratchy strokes peculiar to CPDs, the areas where thin layers of pigment remain present a lighter feeling or impression.

References
1. Martin, J. The Encyclopedia of Colored Pencil Techniques, Running Press, April 1997.
2. Kaufman, A., et al. Volume Graphics, IEEE Computer, vol. 26, no. 7, pp. 51-64, July 1993.
3. Takagi, S. and Fujishiro, I. Microscopic Structural Modeling of Colored Pencil Drawings, SIGGRAPH 97 Visual Proceedings (Technical Sketches), p. 187, August 1997.
4. Miller, G. Efficient Algorithms for Local and Global Accessibility Shading, Proc. SIGGRAPH 94, pp. 319-326, July 1994.
5. Cabral, B. and Leedom, L. C. Imaging Vector Fields Using Line Integral Convolution, Proc. SIGGRAPH 93, pp. 263-272, August 1993.
6. Avila, R., et al. Volvis: A Diversified Volume Visualization System, Proc. IEEE Visualization '94, pp. 31—8, October 1994.

Armin Bruderlin
Sony Pictures Imageworks
armin@spimageworks.com

Wet and Messy Fur

One of the many challenges in modeling, animating, and rendering believable mammals in computer graphics has been the generation of realistic-looking hair and fur. Approaches can be divided into two basic categories: modeling and rendering. In the former case, individual hair primitives are geometrically defined, whereas in the latter case, the geometry of hairs is "faked" through special rendering techniques. In this sketch, we discuss two new effects we implemented as part of our hybrid geometric/rendering hair/fur pipeline:

1. Production of a wet fur look by clumping of individual hairs.
2. A breaking up of combed hairs along fur tracks on the skin.

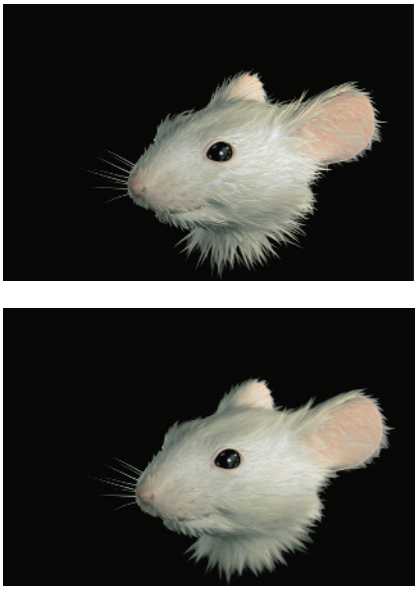


Figure 1: Two frames of an animated static clumping sequence. The model is defined by 34 NURBS patches and has approximately 450,000 hairs.

Clumping of Hairs

Clumping of hairs can occur when the fur gets wet. The effect is that the tips of a set of neighboring hairs (clump hairs) tend to gravitate towards the same point (clump-center hair), creating a kind of cone-shaped "super-hair,"or circular clump. We have implemented two kinds of clumping: static area clumping and animated area clumping. The former generates hair clumps in fixed predefined areas on the model, whereas the latter allows clumping areas to move on the model. In both cases, we provide parameters that can be animated to achieve various degrees of dry-to-wet fur looks.

Static Area Clumping

There are four required clumping input parameters: clump-density, clump-size, clump-percent, and clump-rate. Clump-density specifies how many clumps should be generated per square area. Clump-size defines the area of a clump in world space. Our clump method also has an optional clump-size noise parameter to produce random variations in the size of the clumps. Finally, our clump method also assigns a clump-percent and clump-rate value to each clump hair. The values for both range between 0 and 1. Clump-percent specifies the degree of clumping for a clump hair: a value of zero means that the hair is not clumped at all. It is like a "dry" hair. A value of one means that the hair is fully attracted to its clump-center hair. The tip of this hair is in the same location as the tip of the clump-center hair. Clump-rate defines how tightly a clump hair clumps with its clump-center hair. Both clump-percent and clump-rate can also be animated to provide continuous control for dry-to-wet-to-dry fur looks. This is illustrated in Figure 1, which shows two frames from an animated clump-percent and clump-rate sequence. In the top image, clump-percent is 0.9 and clump-rate is 0.3, which results in a moderate wet look. In the bottom image, clump-percent and rate are both 1.0, which produces a very wet look.

Animated Area Clumping

Animated area clumping is desirable if we want to simulate spouts of water or raindrops hitting the fur and making it increasingly wet. Animated clumping areas are defined through particles hitting surface patches. These particles originate from one or more emitters, whose attributes determine, for instance, the rate and spread of the particles. Once a particle hits a surface patch, a circular clumping area is created on the patch at that location, with clump-percent, clump-rate and radius determined by a creation expression. Each area contains several actual clumps, depending on its radius and on clump-size. Runtime expressions then define clump-percent and rate, to determine how quickly and how much the fur "gets" wet.

Breaking of Hairs

Hair breaking occurs because fur sometimes breaks along certain lines (fur-tracks) on the skin. We have identified and implemented two kinds of breaking: symmetric and one-sided. In symmetric breaking, hairs on both sides of a fur-track "break" towards that track, whereas in one-sided breaking, hairs on one side of the track break away from the track. Fur-tracks are specified as curves-on-surfaces. Each track has a radius, break-percent and break-rate for symmetric and one-sided breaking, and an additional break-vector for the latter.

Which Way Is the Flow?

David L. Kao
NASA Ames Research Center
davidkao@nas.nasa.govls

Introduction

The line integral convolution (LIC) technique has been known to be an effective tool for depicting flow patterns in a given vector field.[1] There have been many extensions to make it run faster and reveal useful flow information such as velocity magnitude, motion, and direction.[2] There are also extensions to unsteady flows and 3D vector fields. In this sketch, a new method for highlighting flow direction in LIC images is presented. The method gives an intuitive impression of flow direction in the given vector field and reveals saddle points in the flow. The method presented is an automatic approach for highlighting flow features, which often represent the most important and interesting physical flow phenomena.

Directional LIC

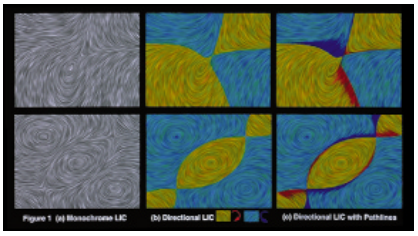
A fast and simple method to depict overall flow direction in the given vector field is presented. The method, which is referred to as directional LIC, colors LIC flow texture based on its orientation and uses local streamlines computed in LIC to show the actual flow direction. The method proceeds as follows.

For each pixel in the vector field, the orientation of the flow is classified as either clockwise or counterclockwise. The direction of orientation can be determined by the curl of the local velocity vector V. Let $V=(u,v,w)$; then $curl\ V = [dw/dy-dv/dz\ \ du/dz-dw/dx\ \ \ dv/dx-du/dy]^T$; $curl\ V$ represents the axis of rotation and the amount of rotation. For 2D vector fields, $curl\ V = [0\ \ \ 0\ \ \ dv/dx-du/dy]^T$ and it represents a vector perpendicular to the XY plane. Directional LIC uses a color input noise, where the R, G, and B primary colors are convolved individually, and the texture color at each pixel is set according to $curl\ V$. If the z component of $curl\ V$ is positive (i.e., the flow is rotating in counter-clockwise direction), then the red primary color is set to zero; thus the texture would be colored in some variation of cyan hues. Otherwise, the blue primary color is set to zero to create yellow flow textures when $curl\ V$ is negative. This results in a color LIC image colored according to flow direction. However, it only shows regions with the same flow direction, not the actual particle path in each region.

A straightforward method is to release particles/dyes randomly in the input field and overlay the particle pathlines on the LIC image. Depending on the seed locations and the length of the pathlines, the underlying LIC flow texture may be obscured. A more ideal selection of seeding is one where the flow is changing from one orientation to another (neighboring pixels have opposite signs of $curl\ V$). Pixels in these regions are in the transient zones. Figure 1 compares directional LIC (1b) with traditional monochrome LIC (1a) using two data sets. A benefit of the directional LIC method is that it automatically highlights saddles in the flow. As shown in Figure 1b, there is one saddle point in the first data set and two saddle points in the second data set. It can also highlight flow separation and flow reattachment, since the flows are approaching from opposite directions. The transient zones are the boundaries where the texture color changes from yellow to cyan. Figure 1c shows directional LIC with pathlines released from transient zones. Particles are released from all pixels in the transient zones. The particles are then tracked during the positive convolution to form pathlines. At the end of each convolution, every other pathline is tracked continuously until there are no more pathlines. Furthermore, pathlines are terminated when they reach another transient zone. With these pathlines, the flow directions are even more clear.

Topology LIC

Another choice of seed locations for the directional flow lines are those near the critical points (points where the velocity is zero). Algorithms for computing critical points are well known in tensor field visualization. We propose to further enhance directional LIC by choosing seed points based on the critical point locations. This would give us additional insights about the flow.



(a) (b) (c)

References

1. Brian Cabral and Casey Leedom. Imaging Vector Fields Using Line Integral Convolution, in Proceedings of ACM SIGGRAPH'93, August 1993, pp. 263-270.
2. Kwan-Liu Ma, Brian Cabral, Hans-Christian Hege, Victoria Interrante, and Detlev Stalling, Texture Synthesis With Line Integral Convolution, Course notes no. 8, Aug. 1997, ACM SIGGRAPH '97.

Sonny E. Kirkley
Richard B. Dietz
S. Chris Borland
Indiana University
sonny@worldboard.org
www.worldboard.org/misc/SIGGRAPH99

WorldBoard: Enabling a Global Augmented Reality Infrastructure

Information in Places

WorldBoard is a family of technologies that enables association of digital objects (HTML, Java applet, image, sound files, etc.) to any physical location on the planet and to predefined tagged objects. In its simplest form, WorldBoard can be thought of as a global bulletin board containing geocoded messages or Web pages that are attached to a particular location (longitude, latitude, and altitude) or tagged object. In its ultimate incarnation, WorldBoard provides a global augmented reality infrastructure in which virtual content is perceived to be co-registered and present in the physical world.

Imagine:

- Going to a museum and seeing virtual post-it notes or 3D sounds attached to exhibits.
- Entering an office building with a virtual red carpet leading you to your eight o'clock meeting.
- Shopping for groceries and having information and nutrition analysis tools virtually attached to each product by the retailer, the manufacturer, your physician's prescribed health plan, and even Weight Watchers.
- Kids on a field trip to a wetland with virtual research and analysis tools attached to different locations.
- A construction worker looking at the ground and seeing the underground gas lines.

WorldBoard Forum

The WorldBoard concept was originally conceived, and some early prototype work was conducted, in the Apple Research Labs in 1996. Since 1997, a multidisciplinary team of researchers at Indiana University has been conceptualizing and spearheading the development of the requisite infrastructure. This infrastructure is designed to support any individual or business desiring to develop content, software, or hardware that takes advantage of location. These tools were recently released and are currently being reviewed by the WorldBoard Forum, an expanding group of university and business collaborators who are interested in the implications of WorldBoard technologies. The forum is open to anyone who wants to participate in development or discussions.

Infrastructure Tools

WorldBoard is simply an extension of the Web. Therefore, all of our tools are designed to fully leverage current and emerging standards and protocols. The following are the key components we are developing:

- Mediated Reality Markup Language (MRML), a draft standard based on XML for geocoding documents, describing how the client uses the data, and how the client renders the objects.
- WorldBoard Local Server, a database and parser that runs on a normal Web server. It parses MRML documents and notifies the search engine of key information from the documents.
- WorldBoard Search Engine, a central search engine to route users to every geocoded document on known local servers.
- WorldBoard Clients, designed to flexibly support many types of applications. Viewers can be created to render many types of data for various devices.

This infrastructure is designed to flexibly support development of any type of WorldBoard-enabled device or content. It does not dictate what kinds of I/O devices a user will have (for example, GPS, wireless LAN triangulation to determine position, or Aether wire), what the current state of the technology is, or the infrastructure that is available in a particular location. Instead, it enables developers to create the software tools and assemble the hardware in ways that make sense for the user and the content that the user will access. A primary goal of the WorldBoard Forum is to create and facilitate access to WorldBoard materials and assure that WorldBoard content is easy to create, access, and navigate. Anyone should be able to attach anything to any location or object on the planet. In keeping with this philosophy, all infrastructure components are freely available via an open-source license from the above URL.