UNIVERSITY OF CALIFORNIA,
IRVINE

Computational Techniques to Enable Visualizing Shapes of Objects of Extra Spatial Dimensions

DISSERTATION

submitted in partial satisfaction of the requirements
for the degree of

DOCTOR OF PHILOSOPHY

in Electrical and Computer Engineering

by

Donald Vaughn Black, II

Dissertation Committee:
Professor Stephen Jenks, Chair
Professor Daniel D. Gajski
Professor Tomas Lang

2010

ProQuest LLC
789 East Eisenhower Parkway
P.O. Box 1346
Ann Arbor, MI 48106-1346

The dissertation of Donald Vaughn Black, II
is approved and is acceptable in quality and form for
publication on microfilm and in digital formats:

_____

_____

_____

Committee Chair

University of California, Irvine
2010

Contents

List of Figures

List of Tables

ACKNOWLEDGEMENTS

This dissertation is dedicated to my wife, Dr. Noel Sturm, without whom my PhD would never have come to pass.

To all my UCI mentors of the various departments during my era of interdisciplinary study who have contributed to my formal education and from whom I continue to learn ... **Cognitive Psychology**: Mike D'zmura. **Information and Computer Sciences**: Jim Arvo; David Eppstein; Andrew Hanson (UIUC); Aditi Majumder; Gopi Meenakshisundaram; Debra Richardson; Daniel Weiskopf (VISUS); **Electrical Engineering and Computer Science**: Harut Barsamian; Pai Chu; Daniel Gajski; Michael Green; Stephen Jenks; Falko Kuester (UCSD); Tomas Lang; Joerg Meyer; Renato Pajarola (UZ); **Logic and Philosophy of Science**: Jeff Barrett; David Malament; **Mathematics**: Richard Palais; Ron Stern; **Physics**: Betsy Barton; Gregory Benford; Jim Hartle (UCSB); Bill Heidbrink; Jonathan Feng; Jon Lawrence; Riley Newman; Bill Parker; Arvind Rajaraman; Wolfgang Rindler (UT); Dennis Silverman; Virginia Trimble;

Portions of Section 2.3 of *Prior Work* are derived from prior published research by Black in 2005[1] and 2007[2]. Portions of Section 2.3 are copyright (2007) American Association of Physics Teachers. These contents may be downloaded for personal use only. Any other use requires prior permission of the author and the American Association of Physics Teachers. All figures not created by the author are licensed from Wikipedia.org under the Creative Commons Attribution-Share Alike 2.5 Generic license.

CURRICULUM VITAE

Donald Vaughn Black, II

UCI EECS	2010 Ph.D. in Electrical and Computer Engineering.
UCI ICS	2005 M.S. in Information and Computer Science.
UCI Physics	2000 Fulltime student at UCI in the Department of Physics and Astronomy.
IEEE OC	Executive Committee member and Volunteer for OC Section, Computer Society Chapter (Past Chair), Video Game Engineers (Founder & Past Chair).
PilotAge	Private pilot and founder of www.PilotAge.com, a Web resource for pilots.
Digital Choreo-Graphics	Director of R & D. Conceived and designed SnapShot, a virtual architectural rendering tool to superimpose a 3D CAD drawing onto a picture of the building site. Developed retail CAD and imaging POS software for niche markets (SPLASH!, Planscape, Floor Visions).
NASA/JPL	Consulting Engineer on *MSAT-X* digital satellite voice and data phone transceiver research project.
Calcomp, Inc.	Project manager on *Prisma* CAD graphics workstation multi-tasking OS.
Ford Aero.	Embedded realtime targeting and navigation firmware.
Edutech Project	Director of R & D. Conceived, designed and prototyped *DOTTIE*, the first *Set Top Web TV*. Other products included the PIES client/server courseware development and delivery system for mainframes, timesharing and personal computers. Used by State of Alaska, County of San Diego, Georgia Tech, Pepperdine U., Mattel Electronics.
NCSU Biomath	Conceived and coded an interactive 3D star map display on the Adage Ambilog-200 hybrid computer. Awarded North Carolina Teacher's Credential.
Junior High	Designed and built rudimentary digital computer from army surplus telephone equipment.
Other	Professional experience in business, management, media production, education, promotion, consulting, aviation, music theory, composition, and software engineering.

FIELD OF STUDY

Computer Graphics and Visualization

ABSTRACT OF THE DISSERTATION

Computational Techniques to Enable Visualizing

Shapes of Objects of Extra Spatial Dimensions

By

Donald Vaughn Black, II

Doctor of Philosophy in Electrical Engineering and Computer Science

University of California, Irvine, 2010

Assistant Professor Stephen Jenks, Chair

Envisioning extra dimensions beyond the three of common experience is a daunting challenge for three dimensional observers. Intuition relies on experience gained in a three dimensional environment. Gaining experience with virtual four dimensional objects and virtual three manifolds in four-space on a personal computer may provide the basis for an intuitive grasp of four dimensions.

In order to enable such a capability for ourselves, it is first necessary to devise and implement a computationally tractable method to visualize, explore, and manipulate objects of dimension beyond three on the personal computer.

A technology is described in this dissertation to convert a representation of higher dimensional models into a format that may be displayed in realtime on graphics cards available on many off-the-shelf personal computers. As a result, an opportunity has been created to experience the shape of four dimensional objects on the desktop computer.

The ultimate goal has been to provide the user a tangible and memorable experience with mathematical models of four dimensional objects such that the user can see the model from any user selected vantage point.

By use of a 4D GUI, an arbitrary convex hull or 3D silhouette of the 4D model can be rotated, panned, scrolled, and zoomed until a suitable dimensionally reduced view or Aspect is obtained. The 4D GUI then allows the user to manipulate a 3-flat hyperplane cutting tool to slice the model at an arbitrary orientation and position to extract or "*pluck*" an embedded 3D slice or "*aspect*" from the embedding four-space. This plucked 3D aspect can be viewed from all angles via a conventional 3D viewer using three multiple POV viewports, and optionally exported to a third party CAD viewer for further manipulation.

Plucking and Manipulating the Aspect provides a tangible experience for the end-user in the same manner as any 3D Computer Aided Design viewing and manipulation tool does for the engineer or a 3D video game provides for the nascent student.

Chapter 1

Introduction

"The symmetries of the subatomic realm are but the remnants of the symmetry of higher-dimensional space."
"...the fundamental laws of physics appear simpler in higher dimensions..."
- Michio Kaku, PhD[3]

In order to understand complex, multi-variate simulations of phenomena, it has proven to be useful to be able to observe the shape, symmetries and asymmetries of higher dimensional mathematical models which emerge from these studies. This capability allows the observer insights into the nature of complex phenomena such as: viewing relativistic spacetime interactions; preventing collision in the design of a robotic arm; exploring intersecting brane models in physics; and discovering unexpected elegant symmetries in physical laws.

Visualizing extra dimensions and models of higher dimensional objects in extra dimensions presents a number of thorny problems. Among these problems is the representation of the dimensionality beyond the three with which we are familiar such that the observer can decode the spatial relationships among the various components of the model. A second challenge is the potential complexity of the multi-dimensional data structure as well as the computational complexity of the

visualization processes.

The intent of this research has been to develop a simple yet computationally tractable data storage and visual representation strategy for the display of spatial models in greater than three dimensions that is both natural and familiar for the sophisticated user. The success of this research will be demonstrated here.

The user is presented with multiple three-dimensional (3D) views of dimensionally reduced sections of higher dimensional worlds or the objects embedded therein. The user can explore these worlds by interactively selecting the desired sub-manifold, point-of-view, or both. The proof-of-concept is demonstrated in four dimensions. Solutions for yet more dimensions are suggested and described.

Certain characteristics and attributes of the higher dimensional models can be revealed by the techniques demonstrated here.

Whether or not Extra Spatial Dimensions truly exist is irrelevant to their effective use in science. We can leave the question of their "real existence" to the philosophers. Use of extra dimensions will simplify complex mathematical formulae, just as visualization will clarify complex mathematical relationships, simplify concepts for the student, and the lay public, as well as for the researcher. Hidden symmetries will become apparent, and unexpected results can be explored.

The appearance of new professional groups such as Computational Geometry and Computational Topology may be the harbingers of the birth of a new field, one whose domain is the synthesis of Cognitive Science, Computer Graphics, Topology, Geometry, Group Theory, Physics, Scientific Visualization, and even Philosophy. These conceptual breakthroughs coupled with visual workstation performance increases make this an appropriate time to explore dimensional morphologies.

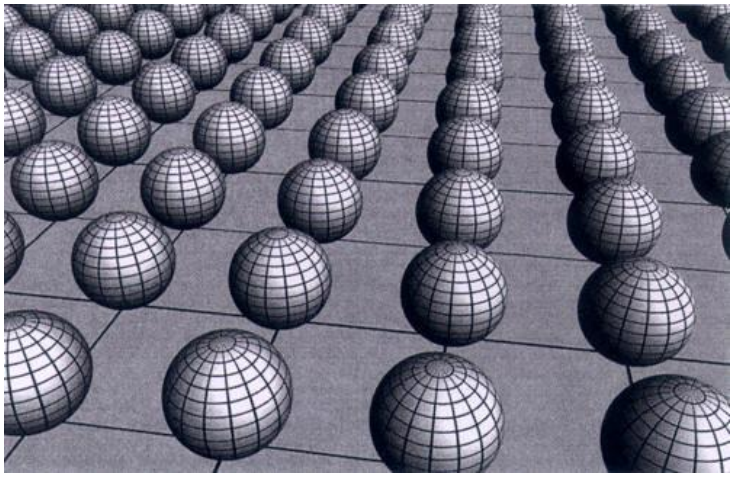


Figure 1.1: Compactified Dimensions
A Schematic Representation of a 5D Universe of 2 Compactified and 3 Infinite Dimensions.

1.1 The Conceptual Evolution of Extra Spatial Dimensions

In 300 BC the geometry of Euclid[4] introduced the concept of isometric spatial dimensions. The seventeenth century mathematician Rene Descartes introduced the cartesian coordinate system. In his 1843 paper entitled "*Chapters in the Analytic Geometry of (n) Dimensions*",[5] Cayley became one of the earliest, and perhaps the first, to publish a work on the geometry of more than three dimensions. In 1844, following Cayley's work, Grassmann published *Extension Theory*[6], encompassing the concept of an n dimensional vector space. Then in 1854, Riemann delivered his famous inaugural lecture on the foundations of geometry at the University of Göttingen. This lecture, "On the Hypothesis which lie at the Bases of Geometry"[7] in which curved metric spaces were described, extended non-Euclidean differential geometry to extra dimensions and was seminal in what has come to be known as *Riemannian Geometry*. His contribution created the mathematical framework for

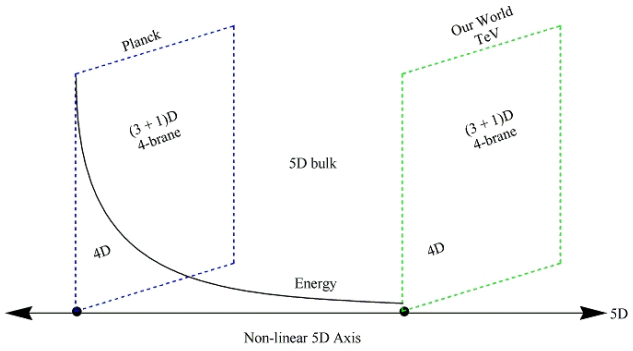


Figure 1.2: A Diagrammatic Representation of an Infinite 5D Randall-Sundrum Universe

Two 4D branes embedded in an infinite non-Euclidean 5D universe

Einstein's *General Theory of Relativity*.

Minkowski's four-dimensional (4D) spacetime[8] introduced in 1908 consists of three Euclidean dimensions and one time dimension which is asymmetric with respect to the three Euclidean spatial dimensions. Einstein's spacetime of General Relativity[9] introduced in 1916, while locally Euclidean, is globally non-Euclidean. Kaluza[10] and Klein[11] in 1926 proposed a non-Euclidean five dimensional spacetime with one compact dimension. Arkani-Hamed, Dimopolous and Dvali[12] (ADD) showed in 1998 that there could be Large eXtra Dimensions[1] that might explain the hierarchy problem. The ADD closed compact Extra Dimensions are depicted by the balls in Figure 1.1. In 1999, Randall and Sundrum[13] suggested that these Universal Extra Dimensions need be neither compact nor Euclidean, but rather could be hyperbolic as depicted schematically in Figure 1.2. Some contemporary string theories embrace the ten-dimensions of Ramanujan's mathematics[3] while other theories consider even more, as well as fewer, dimensions.

The resultant blossoming of the concept of extra spatial dimensions has caused higher dimensional representations to expand into ever greater fields and has created a rich plethora of applications. Many of these fields can represent their data

[1]Underlined terms are defined in the Glossary on page 146

Number of dimensions	*Simplex vertices*	*Simplex name*	$n-Cube$ *designator*	*n-Cube's vertices*	*n-Cube's commonname*
0	1	*point*	$0-cube$	1	*point*
1	2	*line*	$1-cube$	2	*line*
2	3	*triangle*	$2-cube$	4	*square*
3	4	*tetrahedron*	$3-cube$	8	*cube*
4	5	*pentachoron*	$4-cube$	16	*tesseract*

Table 1.1: Table of Simplices and n-Cubes

as analogues in spatial dimensions. Treating multi-dimensional spatial data can thus naturally address many of these applications.

The objective of this research is to develop strategies to interactively view and explore extra-dimensions on the personal computer. The term extra-dimensions as used here refers to more than the three common spatial dimensions. The term includes both four spatial dimensions (4D) and 3-spatial plus one temporal dimension (3+1)D, as well as 5D and (4+1)D, and yet higher dimensionality.

1.2 Terminology

Terminology is a challenge in the domain of extra dimensions. Many common geometrical terms such as face, surface, sphere, etc, can become ambiguous and must be interpreted from their context. To avoid this ambiguity, terms will be defined in the glossary on page 146, and underlined at their first occurrence.

The term m-simplex refers to the simplest possible polytope in m-space. The terms used for simplices of various dimensions are given in Table 1.1 and examples are shown in Figure 1.7.

The prefix "hyper-" will be used in this dissertation usually to indicate that the prefixed term refers to an object of a higher dimension than that of common usage. But this prefix can also be used to indicate a generic dimensionality. For example, hyper-cube could refer to any m-cube, while hyper-sphere could refer to any

m-sphere. A hyper-surface could be a 2D surface in 3-space, a 3-manifold in 4-space, or an m-manifold in (m+1)-space. A hyper-surface is a sub-manifold of codimension one, which is to say that the hyper-surface's dimensionality is one less than the space in which it is embedded.

Two new concepts and their referents shall be introduced at this point, ***Aspect*** and ***Pluck***, and described here as well as in the glossary.

Aspect - This term is used to refer to a specific 3D Slice that creates a new 3D object from an nD object dimensionally reduced to 3D, commonly referred to as a 3D view, 3D shadow, 3D silhouette, or 3D Slice of a higher dimensional object.

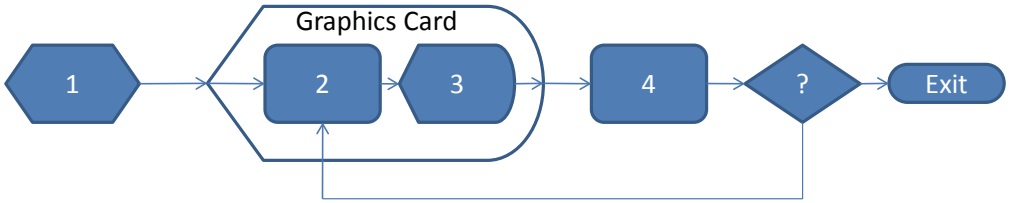
Pluck - This term refers to the operation which intersects a k-flat or kD hyperplane with an m-manifold in n-space to extract a k-manifold object, where

$$codim(k) + codim(m) = codim(k \cap m) = (n - k) + (n - m) = (n - (k \cap m)$$

for $k \leq m \leq n$. For example, (4-space - 3-flat) + (4-space - 3-manifold) = 2-manifold. The 3-flat slicer in 4-space should extract a 2-manifold from a 3-manifold embedded in the 4-space. The algorithm introduced here will pluck a triangular mesh bounding a 3D model from a tetrahedral mesh bounding a 4D model. The methodology is described in more detail in Section 6.2 Implementation.

1.3 A Brief Introduction to 3D Graphics Programming

The conventional computer graphic visualization procedure is to:



21st Century Graphics Hardware makes short work of triangles in 3D.
The hardware is designed to render 3D objects defined by bounding
pure simplicial 2-complexes – closed compact triangular meshes.

Figure 1.3: The 3D Visualization Procedure

1. **Construct** three-dimensional objects using simple two-dimensional surface primitives (usually triangles);

2. **Transform** the objects as specified via translate, rotate, and scale 4x4 homogeneous matrix operations;

3. **Render** these three-dimensional objects onto a two-dimensional viewplane as seen from a particular point-of-view.

4. **Animate** the objects by changing the relevant matrices, incrementing the image file, and repeating from Step 2 until the end of the animation.

As shown in Figure 1.3, the Transform and the Render steps of the 3D visualization procedure are now performed in hardware on the graphics card in the contemporary personal computer.

Since we are primarily interested in the shape of a 3D object, that is in how an object appears, 3D objects can be represented by their 2D surfaces. The simplest 2D surface element is a triangle. The accuracy of an object's representation can be controlled by the number and size of the triangles of which it is constructed. The process of separating an object's surfaces into triangles is called tessellation, as shown in Figure 1.4.

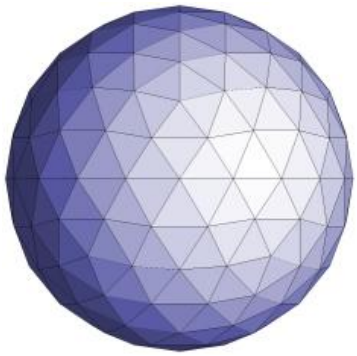


Figure 1.4: A 3D Sphere with a surface tesselated by triangles

Animation is provided by iteratively moving the point-of-view and the objects in step 4, then repeating from step 2. The simulation of the effects of physical forces acting on the objects can be manifested by modifying the matrices that determine how the objects are moved in step 2. The position and attitude of each object is defined by a three-dimensional transform matrix composed of the translations and rotations to be applied to each object. The point-of-view is defined by the composition of a three-dimensional projection matrix which also includes the type of projection (perspective, orthogonal, etc).

The transform matrix is composed by multiplying together each of the independent rotation and translation matrices, corresponding to degrees of freedom in 3-space, as in Equation 1.1. Note that the order of composition is critical since the rotation matrices do not commute in the general case. This yields one 3D transformation matrix with six degrees of freedom that can be multiplied by all the object's vertices to transform the object for each view.

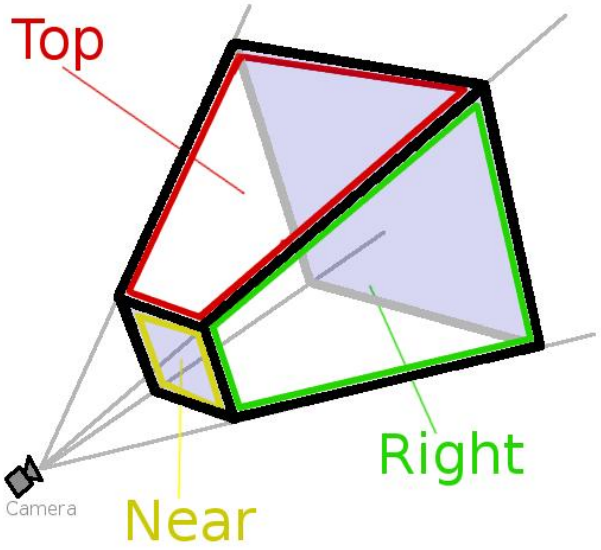


Figure 1.5: A depiction of camera showing view frustum.

$$\overline{Transform_{3D}} = \overline{Rotation_Z} \times \overline{Rotation_Y} \times \overline{Rotation_X} \times \overline{Translation_X} \times \overline{Translation_Y} \times \overline{Translation_Z} \quad (1.1)$$

The 3D to 2D graphics hardware performs scale and perspective matrix transforms.

Given that physical objects cannot be shrunk or stretched, there are no corresponding degrees-of-freedom for scaling in the implementation. However, scaling and perspective transforms may occur in rendering. For a 3D view package such as OpenGL the matrix of Equation 1.1 is handed off to the package's geometry engine's pipeline for performance of the vertex transforms as part of the rendering process. Detailed information about the 7D matrix composition methods used here is provided in Appendix A, *Matrix Composition*.

In case of a perspective projection, rendering can be compared to capturing an image with a pinhole camera. The viewable area for this camera is a *frustum*

(truncated pyramid) that encompasses the volume from near the pinhole out towards infinity. It is truncated to both limit the viewable space to a finite volume and to prevent numerical overflow. All the objects within the view frustum can be rendered onto the 2D image plane, which is represented by the position of the film within the camera. The view frustum and image plane of a camera are depicted in Figure 1.5.

Two common implementations of the rendering (3rd) step are the *polygonal method* and the *raytracing algorithm* which implement projection and intersection, respectively.

The ***polygonal method*** projects the object's 2D surface polygons through a point-of-view onto the two-dimensional viewplane, then clips and discards those polygons invisible from the point-of-view.

The ***raytracing algorithm*** traces a lightray back from the point-of-view to its origin, through each pixel of the display, out into the three-dimensional scene where it intersects with the closest three-dimensional object. For both techniques, the color (or material property) of the object is used to color the target pixel on the image plane. For the ray-tracing implementation, objects can reflect and refract, so that the lightray is filtered by each of those objects it encounters.

1.4 Extrapolating 3D Visualization to Visualizing 4D Models in 3D

Pursuant to our interest in the appearance of a 4D object in 3D space, contemporary computer graphics methods would work were it possible to extract representative 2D surfaces that faithfully represent the 4D object in 3D space. Two new strategies will be discussed and implemented here, as well as proposed alternative strategies that use these technologies to explore non-Euclidean space.

Extrapolating the 3D transform of Equation 1.1 is almost straight forward, except for the rotation. While rotation in 3-space can be characterized with a so-called *axis of rotation*, this is insufficient information to uniquely specify an axis of rotation in 4-space. For example, the rotation of the X-axis into the Y-axis in 3-space holds the Z-axis values constant as in 4-space. However, in 4-space the W-axis values are also held constant by this operation. Hence, the Z-axis is no longer the odd-man-out, so to speak. Either this can be called a rotation about the WZ-plane, or a rotation of the XY-axes. The latter terminology shall be used here, wherein the axes that are modified are used to characterize the rotation. As in 3D, composing with one matrix for each degree of freedom yields the ten degrees of freedom in 4-space as shown in Equation 1.2.

$$\begin{aligned} \overline{Transform_{4D}} \;=\; & \overline{Rotation_{XY}} \times \overline{Rotation_{XZ}} \times \overline{Rotation_{YZ}} \times \\ & \overline{Rotation_{XW}} \times \overline{Rotation_{YW}} \times \overline{Rotation_{ZW}} \times \\ & \overline{Translation_X} \times \overline{Translation_Y} \times \overline{Translation_Z} \times \overline{Translation_W} \end{aligned} \tag{1.2}$$

There is no scale or perspective transform in this hypothetical 4-space.

The higher dimensional matrices require 25 MAC[2] operations per vertex for the 4D 5x5 homogeneous matrix transforms as opposed to the 16 MAC's required for similar 3D 4x4 transforms.[3]

The concept of extrapolating 3D strategies into 4D was repeated for both raytracing where retarded time was used to select the 2D representation of a classically

[2]**MAC** - The Multiply and Accumulate operation performs a multiplication and summation of results in one operation. It is a common operation in digital signal processing (DSP) hardware. It is used here to simplify discussion.

[3]The *Spaceslice* proof-of-concept Test-Fixture requires 64 MAC's per vertex transform since it is implemented in 7D and uses 8x8 homogeneous matrix transforms. The 3D portions could be optimized to 16 MAC's and the 4D portions to 25 MAC's, but would not offer a significant performance increase if ported to a well designed multi-core processor.

relativistic 4D spacetime object; and user specified 3D shadows of 4D objects plucked from a 4D world. It shall be shown in Section 7.3 that these 3D representations are accurate within specified tolerance.

1.4.1 Matrix Composition Computational Complexity

Extrapolating the computational costs for an n dimensional implementation is as follows. One translation for each axis n, plus one rotation for each unique pair of axes $\binom{n}{2}$ requires $(n + \binom{n}{2})$ matrix multiplications, which reduces to $\mathcal{O}(n^2)$. Each matrix multiplication requires $\mathcal{O}(n^3)$ MAC's, yielding $\mathcal{O}(n^5)$ MAC operations for dimension n. Matrix composition occurs once for each viewport, so the complexity of the matrix composition is constant for a constant dimensionality. As noted above, seven dimensions are implemented in the Test-Fixture imposing a computation cost of $(7 + 1)^5$ or 32K MAC operations per viewport per frame as opposed to 1K MAC operations per viewport per frame for a 3D implementation. Detailed information about the 7D matrix composition methods used here is provided in Appendix A, *Matrix Composition*.

1.5 Constructing an Object of Extra Dimensions

A hyperobject of dimension m can be built of two extra-dimensionally non-coincident objects of dimension $(m - 1)$ in m-space, where $m > 0$, as shown in Figure 1.6. For example, a line (1D) is defined as the connecting distance between two points (two 0D objects) which are not coincident in the 1D space. A 2D square is defined by connecting the endpoints (vertices) of two lines of equal length, parallel in the extra dimension, whose distance from one-another in the extra dimensions is equal to the length of the lines. A 3D cube is defined by two 2D parallel squares a distance from one another along the extra dimension's axis equal to the length of a

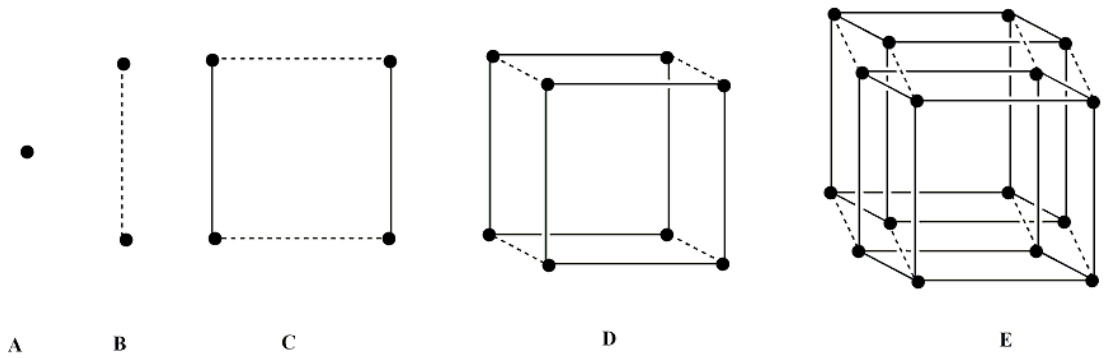


Figure 1.6: n-Cubes of dimension - A) 0D; B) 1D; C) 2D; D) 3D; E) 4D
A) 0-cube; B) 1-cube; C) 2-cube; D) 3-cube; E) 4-cube
A) Point; B) Line; C) Square; D) Cube; E) Hypercube.

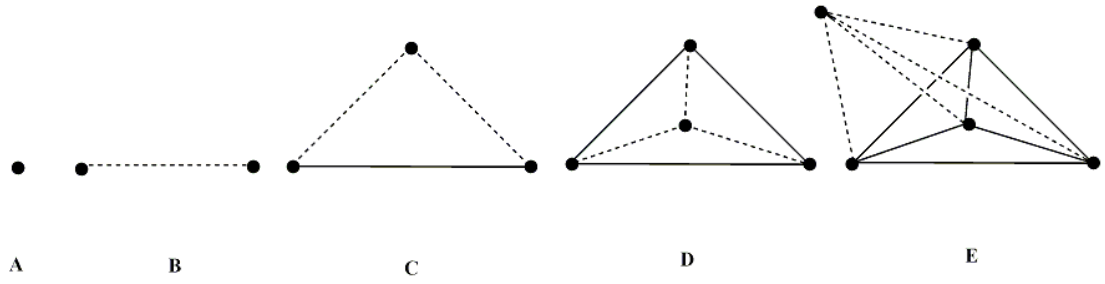


Figure 1.7: n-Simplices of dimension - A) 0D; B) 1D; C) 2D; D) 3D; E) 4D
A) 0-simplex; B) 1-simplex; C) 2-simplex; D) 3-simplex; E) 4-simplex.
A) Point; B) Line; C) Triangle; D) Tetrahedra; E) Pentachoron.

side whose 4 adjacent vertices are connected by lines perpendicular to the plane of the square. A 4D hypercube is defined by two identical parallel squares which are a distance from each other equal to the length of one of their sides, and whose vertices are connected by lines perpendicular to the hyper-planes in which the cubes lie. Thus a line is two connected points, a square is two connected lines, a cube is two connected squares, and a hypercube is two connected cubes in four dimensions. One could also define a 5D cube as two connected (4D) hypercubes.

In the database structure to be introduced here, the n-simplex will be used as the fundamental geometrical element to described the nD objects. A progression of nD simplices from the 0D 0-simplex through the 4D 4-simplex is shown in Figure 1.7. Each simplex is formed from the prior lower dimensional simplex by adding a vertex on the newest axis, and then connecting this vertex to all the existing vertices by a

1D line. The nomenclature for these n-simplices is given in Table 1.1.

1.6 Contributions

Describing and viewing extra-dimensions and their morphology on the personal computer has been a challenge due to a lack of an elegant extensible methodology to represent higher dimensional models as well as the processing requirements of an interactive solution. It is felt that if these challenges can be addressed effectively, then the research can be expanded to include 7D and (6+1)D implementations. It is hoped that the research can lead to computationally tractable methods to address up to 11D and (10+1)D environments. Exploring extra temporal dimensions will also be an interesting challenge.

This dissertation will describe a technology to address these issues with an extensible mechanism and associated computationally tractable algorithms that will convert higher dimensional models into a ubiquitous three dimensional model representation for use on contemporary personal computers.

As noted earlier, the existence or non-existence of extra dimensions is irrelevant to this research. It is expected that the ability to visualize and explore higher dimensions, in addition to the obvious pedagogical advantages, will yield insights into existing philosophical, metaphysical, physical, mathematical, topological, geometric, and other scientific domains.

Three contemporary technologies used to address the above issues will be explored in detail in this dissertation. It will be shown how a modification to the third technology developed here is superior given the performance constraints and visual objectives described above.

Chapter 2

Prior Work

> "... dimension 4 is an unstable boundary case: the dimension is big enough to have room for wild things to happen, but the dimension is too small to allow room to tame and undo the wildness." - Alexandru Scorpan, PhD[14]

Two earlier projects by this author leading to the research described here, are as follows. These are mentioned here in order to put the current research into historical perspective.

2.1 Realtime 4D Rotation Demo

In 1985, while developing the graphics package for a Computer Aided Design (CAD) Workstation, a simple yet comprehensive demonstration of the capabilities of the MC-68010 based Prisma graphics hardware was created: A module to rotate and shade a colored 4D hypercube in realtime. The demonstration performed the ubiquitous 4x4 matrix operations to rotate and dimensionally reduce the 4D model to 3D, culled back facing polygons, then performed the usual 2D projection.

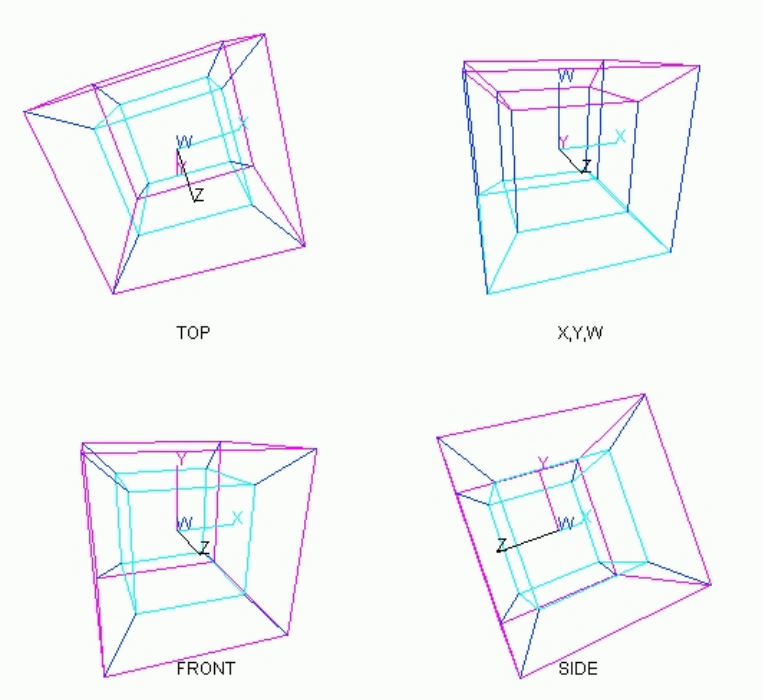


Figure 2.1: Four Views of Four Euclidean Dimensions
Views are manipulated via thumb-sliders.

2.2 Interactive Multi-Aspect 4D Rotator

In 2001, a software device[15] posted on the internet to interactively rotate a 4D hypercube was modified[16] to introduce multiple aspects where each of four 3D viewports contained a different set of three dimensions, each a different *Aspect* in 3-space of the same 4D hypercube. Figure 2.1 shows a 4D tesseract which has been projected with perspective into 3D and then projected once more onto a 2D viewport for display. The three viewports labeled TOP, FRONT, and SIDE, have been arranged so as to be similar to a 3D CAD-like view. The forth viewport in the upper right corner is an alternate 4D view, the XYW Aspect in this case.
For the discussion here, consider this tesseract (4-cube) to have been constructed from a pair of unit 3-cubes one unit distant from each other on the W-axis, one red and one cyan. The cyan cube is on the -W side of the tesseract and the red cube is

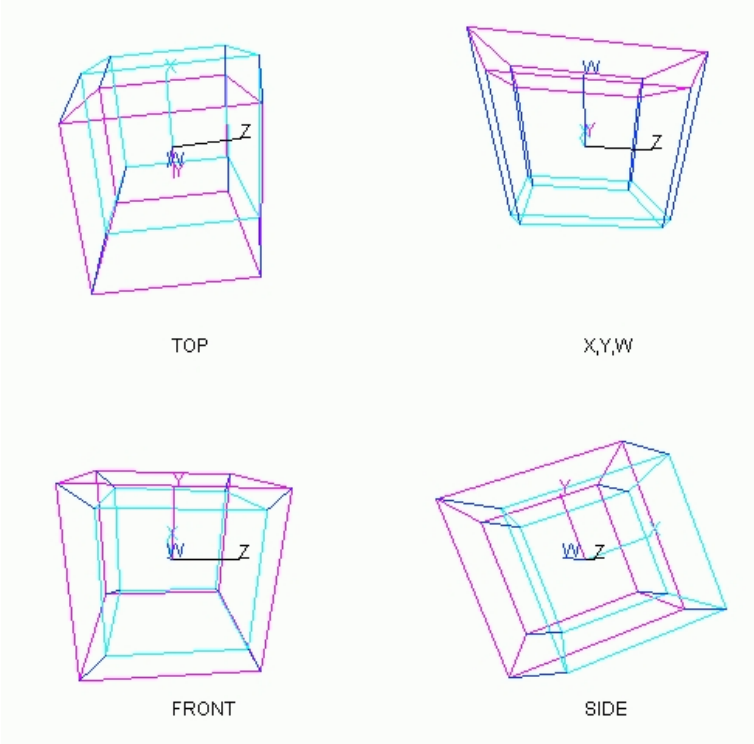


Figure 2.2: Four Views of Minkowski 4D Spacetime
Views are manipulated via thumb-sliders.

on the $+W$ side. Projecting with perspective along the W-axis from the red positive end towards the cyan negative end would, at the proper distance, cause the distal cyan cube to appear to be inside the red cube as depicted in the TOP, FRONT and SIDE viewports of the figure. The three views show the 3D shadow of the 4-cube as projected along W-axis. The relative size of the two cubes give information about their relative positions along the projection axis.

The upper right viewport labeled X, Y, W shows the tesseract rotated so as to project in perspective along the Y-axis providing an XZW Aspect of the tesseract. The relative sizes of the inner and outer cubes suggest the relative distances along the Y projection axis.

Figure 2.2 depicts the ubiquitous hypercube in a (3+1)D Minkowski spacetime.

2.3 Spacetrace - 4D Spacetime Raytracing

"..attempts to clarify this paradox satisfactorily
were condemned to failure as long as the axiom ...
... was rooted unrecognized in the unconscious."
- Albert Einstein on absolute time

2.3.1 Introduction

This section describes prior work wherein a ray tracing algorithm had been used to visualize a Euclidean 4D model of Minkowski spacetime.

For simplicity, a flat spacetime and ***temporal homogeneity*** with no acceleration were assumed, and lighting effects were not considered. Under these conditions flat Minkowski spacetime is Euclidean for an inertial observer. The corresponding model can then be viewed and animated based on 4D raytracing.

Temporal extrusion of an inertial 3D object into 4-space along its normalized velocity 4-vector (worldline), followed by the ***Lorentz transformation*** (length contraction and time dilation) of the object into the inertial reference frame of the stationary camera were used to model object behavior. The camera was then moved along the time axis, raytracing the 4D space, and creating an image collection that was subsequently composited into a video sequence capturing the time-varying effects.[1]

In the following sections the fundamental assumptions will be discussed as well as the Minkowski 2D and 3D spacetime diagrams; the model will be described as well as the construction of 4D objects from 3D objects; and finally, the resulting animations of 3D objects in 4D spacetime will be demonstrated.

2.3.2 Implementation

Terrell[17] and Penrose,[18] demonstrated the visual phenomena explored here can be described as the combination of a pre-relativistic purely optical effect due to finite lightspeed that was discovered by Roemer in 1677,[19] and special relativity's four dimensional spacetime discovered by Minkowski in 1908.[8] The finite speed of light leads to effects analogous to those of sound, as in the case of locating the position of a fast high flying jet by the sound of its engines. Finite and invariant lightspeed requires the physical phenomena predicted by special relativity: time dilation and length contraction. Time dilation is observable only if there is a variation in the object during the viewing period, as in the muon particle's decay. Length contraction is observable by differences in the geometry of a relativistic object at rest and in motion.

Background

Relativistic 4D spacetime (t, x, y, z) consisting of both space and time, is often labeled (3+1)D, referring to three spatial dimensions (x, y, z) and one time dimension t. Similarly, a 3D spacetime (t, x, y) could be referred to as (2+1)D, which is to say containing two spatial dimensions (x, y) and one time dimension t. The most convenient units for the purposes discussed here are ***relativistic units*** where $c = 1$. The benefit of using relativistic units is that the units along all the spacetime axes have the same scale, resulting in a lightray traveling one unit along the spatial axes for each unit it travels along the time axis, similar to the isometries assumed in Sections 2.4 and 6. A lightray c can thus be represented in a Minkowski 2D spacetime diagram as a 45° bisector, or in a 3D spacetime diagram as the surface of a right circular cone, both shown in Figure 2.3. The light-second, the distance light travels in a second, will be used as the basic unit of measure in this section. An object's ***worldline*** is its 4D path through spacetime. The instantaneous

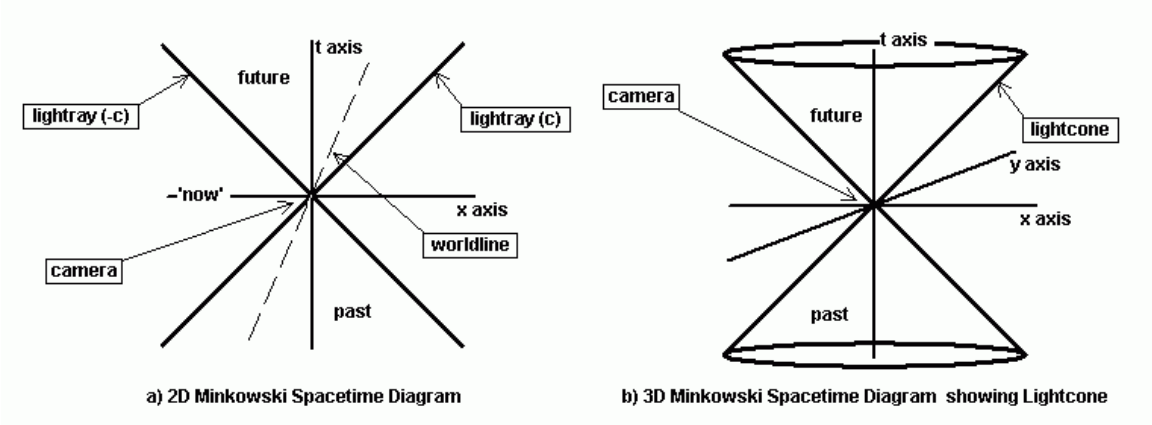


Figure 2.3: (1+1)D and (2+1)D Minkowski spacetime diagrams
A camera at the origin can only 'see' an event in the past whose lightray passes from that event through the camera at the origin.

direction of an object's worldline is the object's proper time axis. The slope of this proper time axis in the Minkowski diagram represents the object's speed. The worldline through flat spacetime of an object with a constant velocity is a straight line. The normalized tangent to an object's worldline is the object's instantaneous velocity 4-vector.

A 3D object can be created by ***extruding*** a 2D object in a direction perpendicular to the 2D plane in which the object lies (for example, by extruding a square from the X, Y plane along the Z axis). Likewise, a 4D object can be created by extruding a 3D object in a direction orthogonal to the 3D hyperplane in which the object lies. A 4D example would be the extrusion of a cube from the X, Y, Z 3-space, along the T axis. Two examples are shown in Figure 2.4. This operation is termed ***temporal extrusion*** when a 3D object is extruded along its 4D worldline.

Raytracing[20] is a geometric 3D image rendering algorithm that colors the pixel on a viewplane by sending a ray from the viewpoint, through a pixel on the viewplane, and out into the scene's 3-space where it may intersect the 2D surface element (such as a triangle) used to define the boundaries of a 3D object. The color of the object's surface at the intersection is used to color the corresponding pixel in the viewplane.

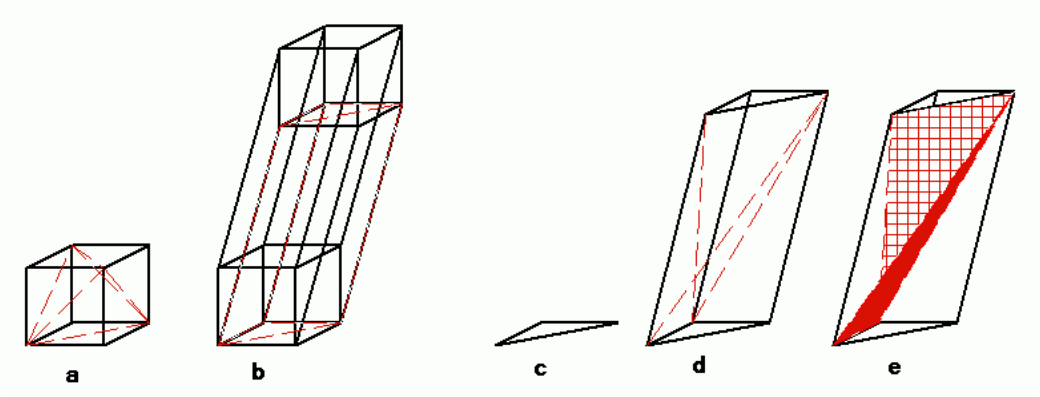


Figure 2.4: Cube & triangle: Extruded then tessellated

This procedure is repeated for each of the pixels in the viewplane. Howard[21] adapted the open-source 3D raytracer POV-Ray Version 2.0 to relativistic raytracing by changing the angle of incidence as a light ray passes from one inertial reference frame to another. It was necessary to increase the model's flexibility in order to demonstrate the difference between finite lightspeed effects and relativistic effects. A four dimensional raytracer was developed by globally extending a 3D raytracer's[22] vector math package from 3D to 4D and adding a fourth component t to the coordinate system. The lightrays were constrained to lie on the negative lightcone so that the ray traveled through the model at lightspeed. The resulting 4D raytracer can image a Euclidean 4D space of 4D objects.

It can be shown that the length of an object with an arbitrary constant relativistic velocity $\beta = \frac{v}{c}$ will contract in the direction of motion by a factor of $\frac{1}{\gamma} = \sqrt{1-\beta^2}$. It can also be shown that the proper duration between any two events on the relativistic object's worldline will expand (dilate) by the Lorentz factor $\gamma = \frac{1}{\sqrt{1-\beta^2}}$. This is known as ***length contraction*** and ***time dilation***, respectively.

Object Construction

Any 3D object defined by bounding triangles such as the cube in Figure 2.4a can be ***temporally extruded*** into a 4D hyperobject and inserted in the scene's 4-space by extruding each f of its n individual triangles as follows. Assuming that the triangle's vertices are defined by their 3D coordinates in 3-space, insert a t component into each of the vertex coordinates and set t to some constant value, say t_0.

$$(x_i, y_i, z_i)_f \rightarrow (t_0, x_i, y_i, z_i)_f.$$

When performed on all three vertices i, the 2D triangle f will have a unique location in 4-space.

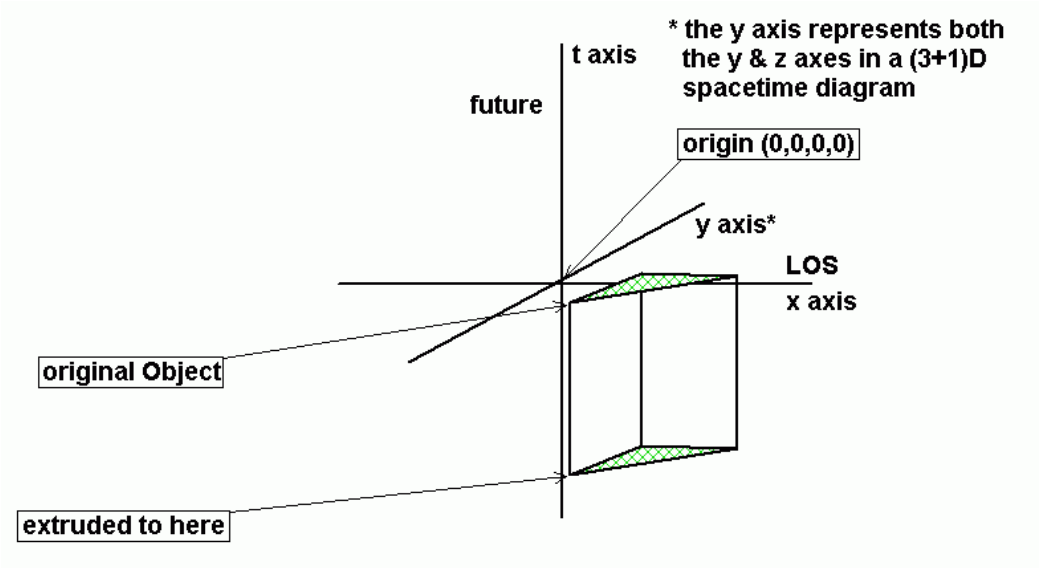


Figure 2.5: Temporal extrusion: Triangle at rest extruded into prism

The object now lies embedded in the XYZ hyperplane that is orthogonal to the t axis at t_0 (*original Object* in Figure 2.5). Each of these triangles f, and hence the object composed from them, can be extruded into the 4th dimension by duplicating the vertices of the triangles with lesser (or greater) values for the t components. If the object is at rest in the camera frame, a constant Δt can be added to the t component of each of the object's original triangles in the t_0 hypersurface to create

an f' duplicate triangle to be used as the object's position in the $t_0 + \Delta t$ hypersurface.

$$(t_1, x_i, y_i, z_i)_{f'} \;=\; (t_0, x_i, y_i, z_i)_f + (\Delta t, 0, 0, 0) \tag{2.1}$$

Where $f = \{1..n\}$ refers to each of the original triangles, $f' = n + \{1..n\}$ to each of the corresponding extruded triangles, and $i = \{1, 2, 3\}$ to each of the corresponding vertices that define each triangle pair.

As shown in Figure 2.5 where $\Delta t < 0$, connecting the three vertices ($i = 1, 2, 3$ in Equation 2.1) of the original triangle f with the respective vertices of the extruded triangle f' creates a 3D prism from the original triangle. Thus the triangle f exists only between t_0 and t_1, inclusive.

The prisms are then tessellated [1] into three adjacent tetrahedra as shown in Figure 2.4e. The 3D simplices are necessary for the barycentric algorithm (described below) used to determine where, on the 3-manifold surface of the 4D object the intersection with the lightray occurs, as well as the intersection algorithm of Section 6

An object's velocity is represented by changing the position of the extruded end of the triangle (Figure 2.6) with respect to the original end: $x_{end} \;=\; x_{beg} + \Delta x$ spatial units. The speed in the camera frame would thus be $\frac{\Delta x}{\Delta t}\ \frac{spatial\ units}{time\ unit}$. Canceling the units yields the dimensionless fraction $\frac{\Delta x}{\Delta t}$. A lightray's slope $c \;=\; \pm 1.0$ is represented by both the diagonal lines and the surface of the lightcone of Figure 2.3. For the general 3D case, where the distance traveled in time Δt is $\Delta d \;=\; \sqrt{\Delta x^2 + \Delta y^2 + \Delta z^2}$, the speed would be $\frac{\Delta d}{\Delta t}$, and Equation 2.1 would become:

$$(t_1, x_i, y_i, z_i)_{f'} \;=\; (t_0, x_i, y_i, z_i)_f + (\Delta t, \Delta x, \Delta y, \Delta z) \tag{2.2}$$

[1] See Tessellation in Glossary.

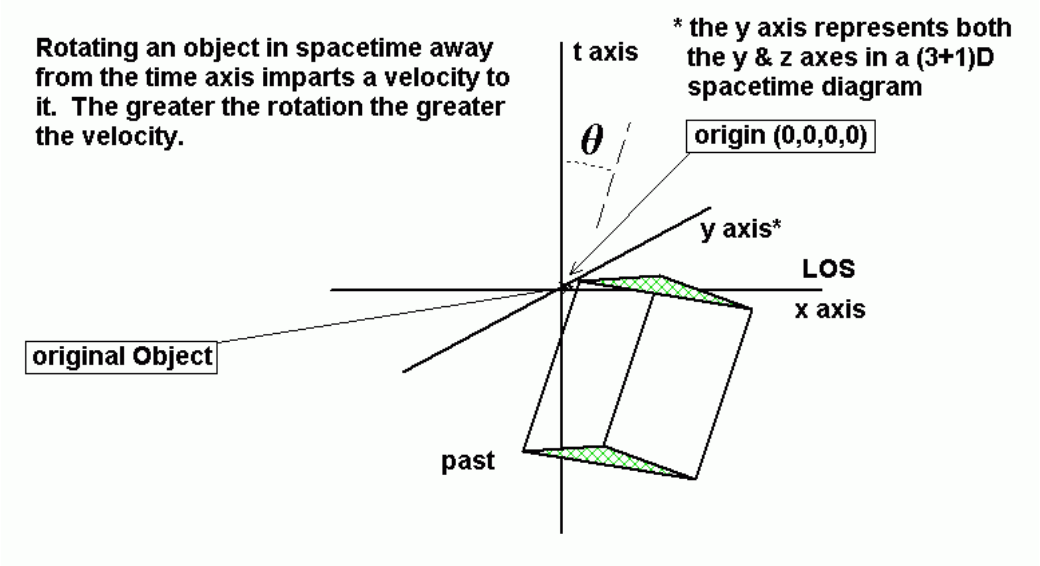


Figure 2.6: Temporal extrusion not parallel to 't' axis
Object has moved: $velocity = \frac{\Delta x}{\Delta t}$

Viewing 3D Objects in (3+1)D Spacetime

Consider a camera at the origin, whose line-of-sight (LOS) is collinear with the x axis. Since a lightray's worldline as depicted in the spacetime diagram lies on the lightcone, an object must cross the lightcone in the diagram in order to be visible to the camera. In fact, the object is visible to the camera only while it is intersecting that lightcone whose apex is coincident with the camera (assuming the camera is pointing at the object) as shown in Figure 2.7.

Figure 2.7 depicts a right circular hypercone in 4-space, whose symmetric axis is collinear with the $-t$ axis, and whose apex is coincident with the camera at the origin $(0, 0, 0, 0)$. This hypercone's hypersurface, depicted by the inverted cone, has 3 dimensions, sufficient to contain the camera's focal point and the lightrays entering its lens. Although a 3-manifold in 4-space, this hypercone is known as a ***lightcone***. A lightcone is thus the locus of all points that satisfy:

$$(t_p, x_p, y_p, z_p) = (-\sqrt{x_p^2 + y_p^2 + z_p^2}, x_p, y_p, z_p) \tag{2.3}$$

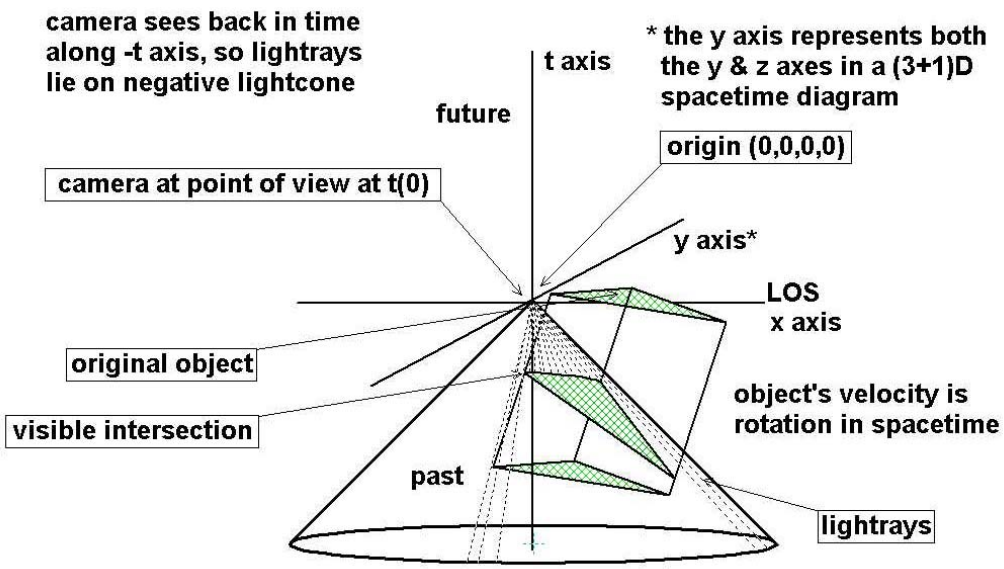


Figure 2.7: Temporal extrusion of moving object with lightcone
A non-relativistic spacetime: finite lightspeed but no Lorentz transform

Note that the light travels from the object to the lightcone's apex at the origin. As depicted by the broken lines representing lightrays in Figure 2.7, a camera located at the apex in this 4D model can see only those 3D objects whose extruded triangles (tetrahedra triads) intersect the lightcone. The only visible objects are those with vertex extrusion pairs (t_0, x_i, y_i, z_i) of the original object and (t_1, x_i, y_i, z_i) of its extruded end-cap, where

$$t_0 \geq \sqrt{x_i^2 + y_i^2 + z_i^2} \geq t_1, \forall \ \ \{(t_0, x_i, y_i, z_i) \ \ \& \ \ (t_1, x_i, y_i, z_i)\} \tag{2.4}$$

The intersecting portion of the extruded triangle is depicted by the triangle labeled *visible intersection* in Figure 2.7. Note that geometric distortion in the object is caused by the intersection of the triangle and the lightcone. An object in the lightcone is easily detected since a straight line can be intersected with a 3D object in Euclidean 4-space in the same manner as a straight line is intersected with a 2D object in 3-space.

Animating Spacetime Objects

There is no mathematical or geometric limit to an object's speed in the model, its velocity being the slope of the temporal extrusion vector. For real physical objects, some physical mechanism must accelerate the object to the speed with which the object enters the model's laboratory inertial frame. It can be assumed with some confidence that this speed must be less than that of light. The physical objects will then maintain an extrusion vector with a $\frac{|\Delta x|}{\Delta t}$ slope of less than 1.0, or an angle of less than 45° with respect to the t axis on the Minkowski diagram as shown by θ in Figure 2.6. Since only uniformly moving objects are being considered, the specifics of the spacetime rotation that yield the extrusion angle can be ignored.[2]

Two classes of 4D objects have been implemented: one for the finite lightspeed objects and one for relativistic objects. The first is inserted into the scene without length contraction or time dilation as shown in Figure 2.7, and the second is inserted with the Lorentz transformation as shown in Figure 2.8. Conceptually, the former may be considered to have been measured in the laboratory inertial reference frame's subjective units (it was already length-contracted and time-dilated), while in the latter case the object was measured in its own rest frame. The relativistic objects therefore must be length contracted and time dilated prior to insertion.

The animation procedure is straight forward. For example, to generate 20 seconds of animation at 10 frames per second ($\Delta t = 0.1 seconds$), the procedure is as follows.

1. Beginning with the camera at (t_0, x_0, y_0, z_0), a view is rendered and saved.
2. The camera is moved forward along the time axis to (t_i, x_0, y_0, z_0), where $t_i = t_{i-1} + \Delta t$ and the view is rendered and saved;
3. Repeat from step 2 while $t_i < 20$.

[2]The temporal homogeneity assumption obviates the need for a discussion of hyperbolic rotation.

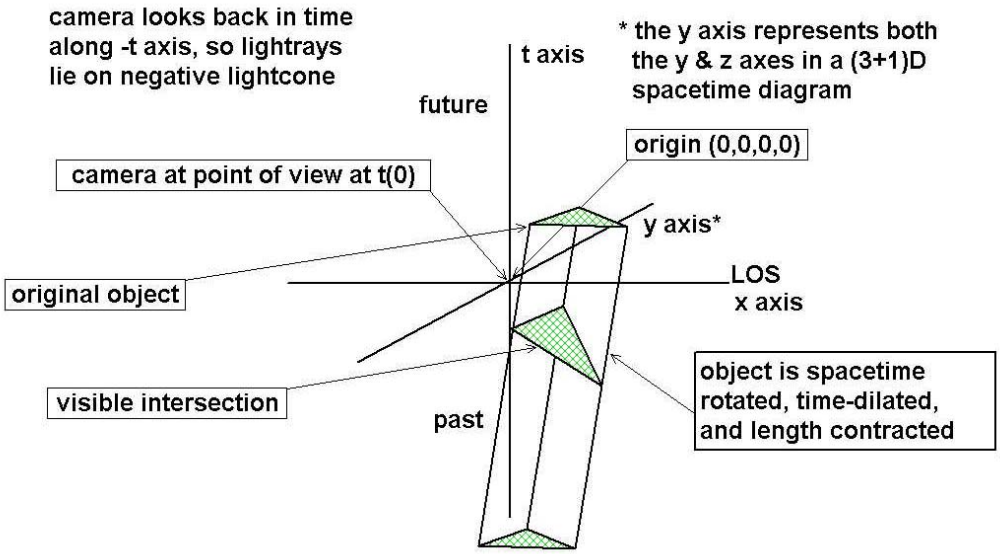


Figure 2.8: Lorentz transformed object
A velocity in the neighborhood of 86.6% of c yields a γ factor around 2. The prism is shown length contracted by $\sim \frac{1}{2}$ and its proper time axis is dilated by ~ 2

Notice that the pinhole camera's spatial components (x, y, z) do not change, only the time component of the camera position. Crucial to the simplicity of the procedure is the fact that the 4D object's bounding surfaces (and the tessellating tetrahedra that comprise those surfaces) ***do not change***. The 4D world is ***static***. Only the point of intersection of the lightcone changes as the camera and its lightcone progress along the t axis.

4D Intersection Algorithm

Lightcone crossing events are detected by solving for the intersection of a lightray with each of an object's bounding tetrahedra. The set of lightrays is defined as that set of 4D straight lines passing from the camera through each of the pixels in the viewplane's pixel grid and out into 4D space. Using a 4D implementation of the barycentric algorithm to compute the intersections of the ray with all tetrahedra faces, we select the intersecting event nearest to the camera (with the t value closest to 0.0). The array of 1D lightrays that originate from the gridded viewplane results

in a 2D image of the objects projected onto that viewplane.

Since the objects have been Lorentz transformed prior to the intersection, such that their geometry is correct for the camera frame in which the intersection occurs, the geometric components of the lighting model, the surface normal and the reflection angle, can be used to approximate the pixel shade just as with a conventional lighting model in 3D rendering.

Photorealistic rendering requires the addition of lighting effects such as Doppler shift[23] and the searchlight effect, which could dominate the rendered image and obscure the visualization of the object's geometry.[24] For this reason, these effects were not implemented in this model.

2.3.3 Examples of Ray Traced 4D Spacetime

Three models of relativistic motion are displayed in the sequential images in Figure 2.9. The object displayed is a flange (angle bracket) 2 light-seconds wide by 2 light-seconds deep by 4 light-seconds tall. Its thickness is negligible (being constructed of four 2D triangles). The top row shows the traditional ray-tracing technique, where the lightspeed is effectively infinite. In the middle row, the pre-relativistic optical effects are shown, while in the bottom row, the relativistic effects are displayed. The finite lightspeed camera (top row) was moved ahead in time 18.675 seconds, an amount equal to the lightspeed delay from the center of the stage to the camera, so that the flanges appear to be in approximately the same positions.

The scene is set upon a stage with an overhead light source, both at rest in the camera frame. Two flanges approach, cross, and depart the centerline of the scene at $0.866c$. The geometric distortions of the center row are due exclusively to classical aberration. Those of the third row are due to relativistic aberration.

The stage's mirrored backdrop shows the reflections of the flanges from behind.

Note the difference in the positions of the reflections in the three rows. The top row shows the instantaneous reflections of the flanges, while the middle and bottom rows show the retarded reflections due to the lightspeed delay imposed by the added distance to be traveled by the lightray from the object to the mirror and back. The distances modeled are on the order of the size of the Jovian system.[3]

Note the bottom flanges *appear* to cross each other before the top flanges. Note also, that even with this head start, the top flanges arrive at their respective edges at the same time as the bottom flanges. The bottom flanges appear to approach faster and retreat slower than the top flanges. This is the visual evidence of the pre-relativistic optical effect known as classical aberration. The flanges approaching the centerline of the stage are obliquely approaching the camera. Aberration causes the angle from the centerline to the flanges to appear smaller than the proper angle of incidence, resulting in the object appearing closer to the centerline, or ahead of the object's proper position as depicted in the top view.

This is true for both the leading and the trailing edges of the flange, independently. As a result, the leading edge, which is closer to the centerline, appears to have moved further than the trailing edge, giving the impression of a wider flange. The opposite effect occurs as the flanges move away from the centerline. The flanges appear to incrementally speed up and simultaneously contract as they move relativistically away from the camera. These aberration effects are apparent in the bottom two panels of each image of Figure 2.9.

2.3.4 Observations

It is possible to accurately visualize 4D spacetime with a simple linear algorithm. However, there are significant performance issues that could be addressed with more

[3]At the recorded animation rate (your playback may differ), the size of the stage is about 6 million Km on a side (20 light-seconds), easily large enough to encompass the planet Jupiter and the orbits of its four largest moons.

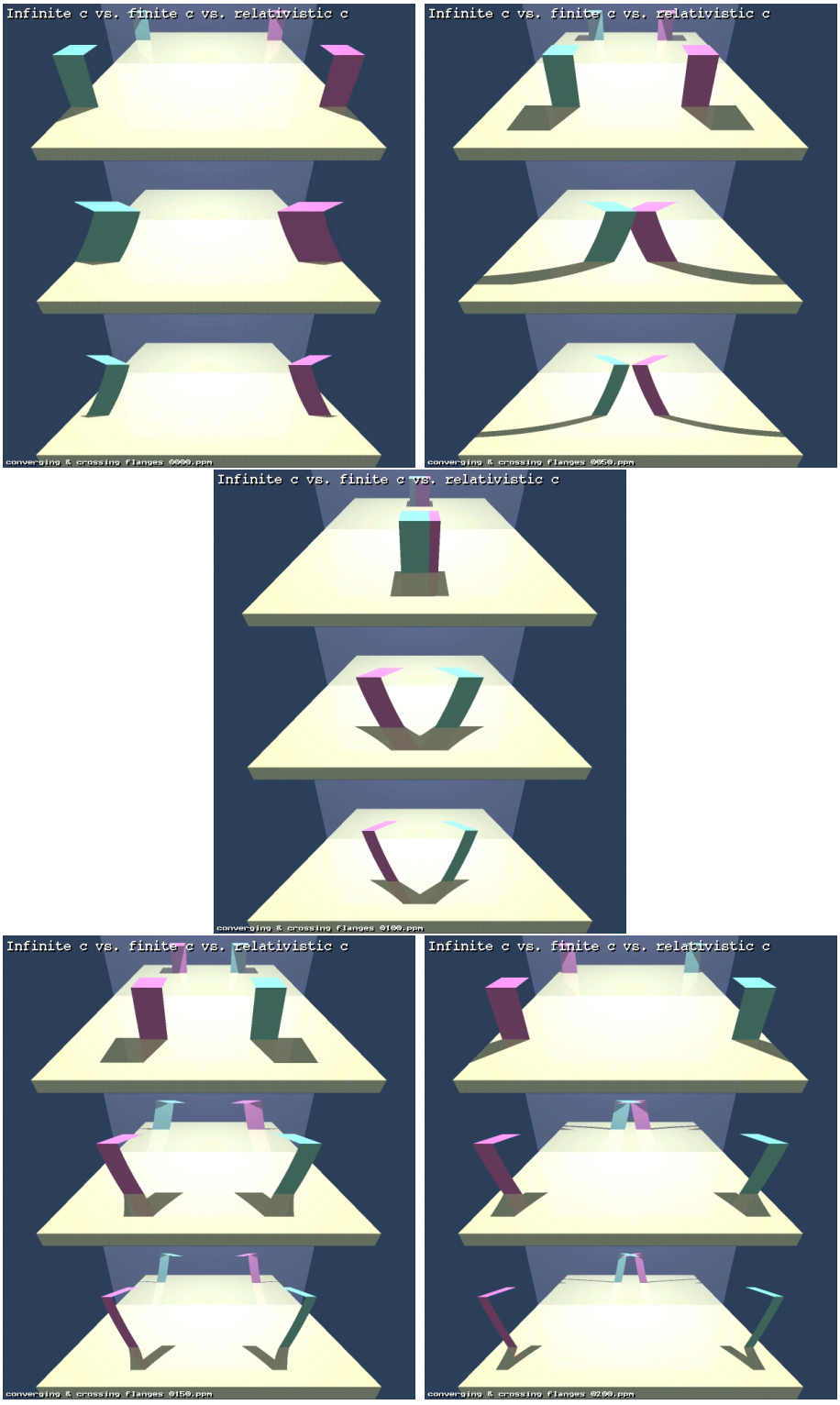


Figure 2.9: Two 4D Objects Crossing at 0.866c

Sequential images of two 4D objects converging then crossing at 0.866c on a mirrored background Top row: infinite lightspeed 4D raytracing; Center row: pre-relativistic spacetime; Bottom row: relativistic spacetime

Accompanying video filename figure-2.9.mov. Video ID: teaser-AA,sceneH.1e

sophisticated algorithms, as will be demonstrated in Sections 2.4 and 6

It is possible to implement a simple algorithm that consists of a finite lightspeed component and a length contraction component to yield special relativistic spacetime visualizations.

2.4 Spacegrid - nD Multi-view Lattice

The Challenge: visualize curved spacetime

2.4.1 Introduction

Space tells matter how to move

Matter tells space how to curve

- Misner, Thorne & Wheeler[25]

The objective here was to explore the very shape of space itself.

A discrete spacetime is hypothesized, consisting of regular isometric "hypervoxels" whose hyper-volume is equal to the fourth power of the length of the edge: l^4. Each hypervoxel is represented by a point or dot analogous to a test particle consisting of only "passive mass"[4] at its geometric center.

To probe this four-space, consider this force to be an analogue of Newtonian gravity. Hypothesize a unipolar attractive force whose strength falls off with the inverse of the square of the distance of the lattice cell from the origin of this force as with Newtonian gravity.

In the domain of Einstein's Theory of General Relativity, the positions of these cells and their perturbation due to gravity will represent the curve of spacetime due to the presence of matter. These perturbations will be representational only since the

[4]**Passive Mass** - mass can be divided into three categories: active, inertial, and passive. Active mass generates gravity; Passive mass is affected by gravity; Inertial mass is affected by inertia. Note that all three masses are empirically identical.

result of gravity will be approximated by Newton's simple Equations of Motion rather than the more complex Einstein Field Equations. This state of affairs is deemed acceptable since the intent is to demonstrate a view strategy rather than physical laws. Explaining the theories of relativity is beyond the scope of this dissertation. The curious reader is directed to *Gravitation*[25] and the citations of Section 1.1.

In order to examine the very shape of space itself, it is necessary to develop both the visual tools and perceptual basis. This module implements a four-dimensional (4D) visualization of a dynamically warped discrete spacetime. Included is a capability to extend the visualization to five dimensions.

Motivation - Viewing the Shape of Space

Computer Aided Design (CAD) packages view 3-space from multiple viewports, each representing alternate projections of 3-space into the view frame's 2-space. Engineers, technicians, CAD operators, and video game aficionados are well versed in decoding the information presented to them as interactive 3D views. Likewise, the view of 4-space may be approximated by using the CAD paradigm to view alternate 3D projections of the 4-space into 3-space. Daily experience is of a locally flat isometric 3-space. Beyond the local manifold, the Universe is a gravitationally warped spacetime.

Flat Isometric Space Euclidean space is flat, that is to say, it is isometric.[5] This space could be easily represented by a regular lattice stretching to infinity in all directions, including the time direction. The lattice of green equidistant tick marks shown in Figure 2.10 is used to represent a flat isometric 4-space.

[5]Newtonian gravity lives in Euclidean 3-space.

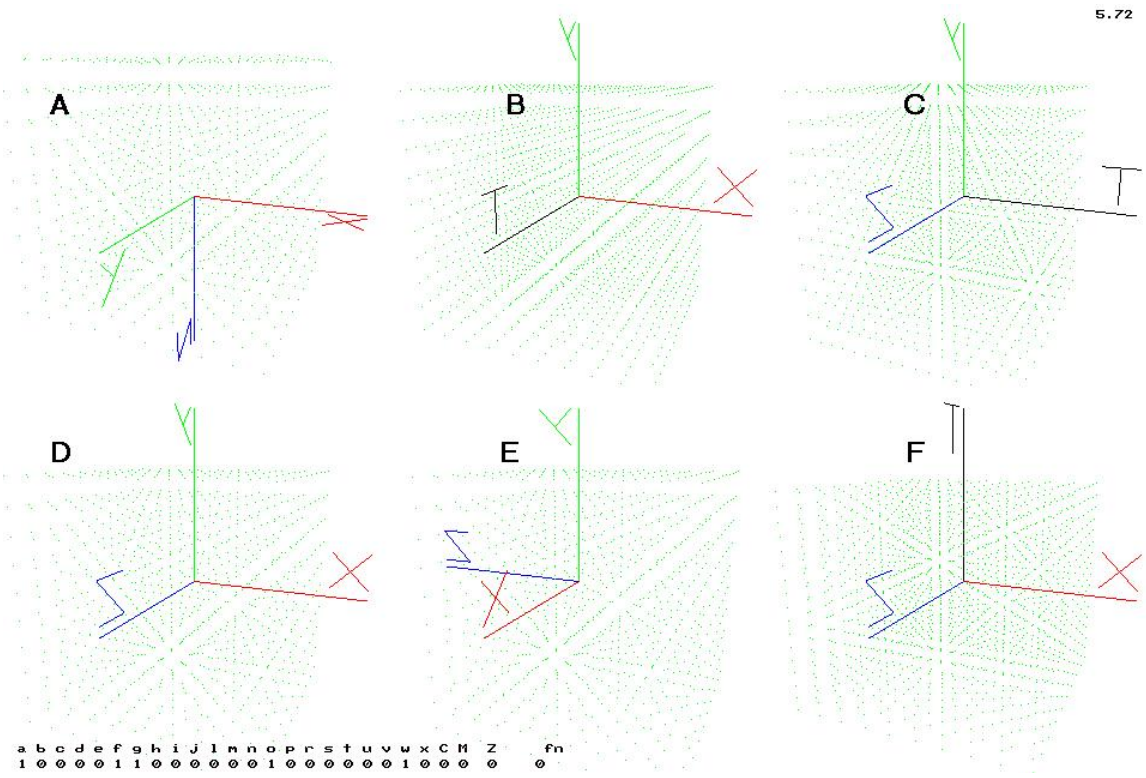


Figure 2.10: Static Euclidean 4D Spacetime Lattice
Green dots show undisturbed state
Standard CAD Views: A) - Top; D) - Front; E) - Side;
Project 4D to 3D along: B) - Z axis; C) - X axis; F) - Y axis

Retarded Time Many applications, especially those in the physical sciences, require an historical (time) component in order to process the contribution of any causal phenomena.[6]

Spacetime Not all spaces are symmetric. For example, Minkowski spacetime is asymmetric - the time axis is not symmetric with the three space axes. But by treating this fourth T axis as a symmetric 'history' axis rather than as a 'time' axis, we can visualize this 4-space isometrically.

Curved Space Applications requiring non-Euclidean space are not uncommon. For example, Einstein's spacetime is non-Euclidean since it can be warped by the

[6]Both classical and relativistic spacetime require a time axis for retarded physical phenomena such as light propagation or gravitational effects.

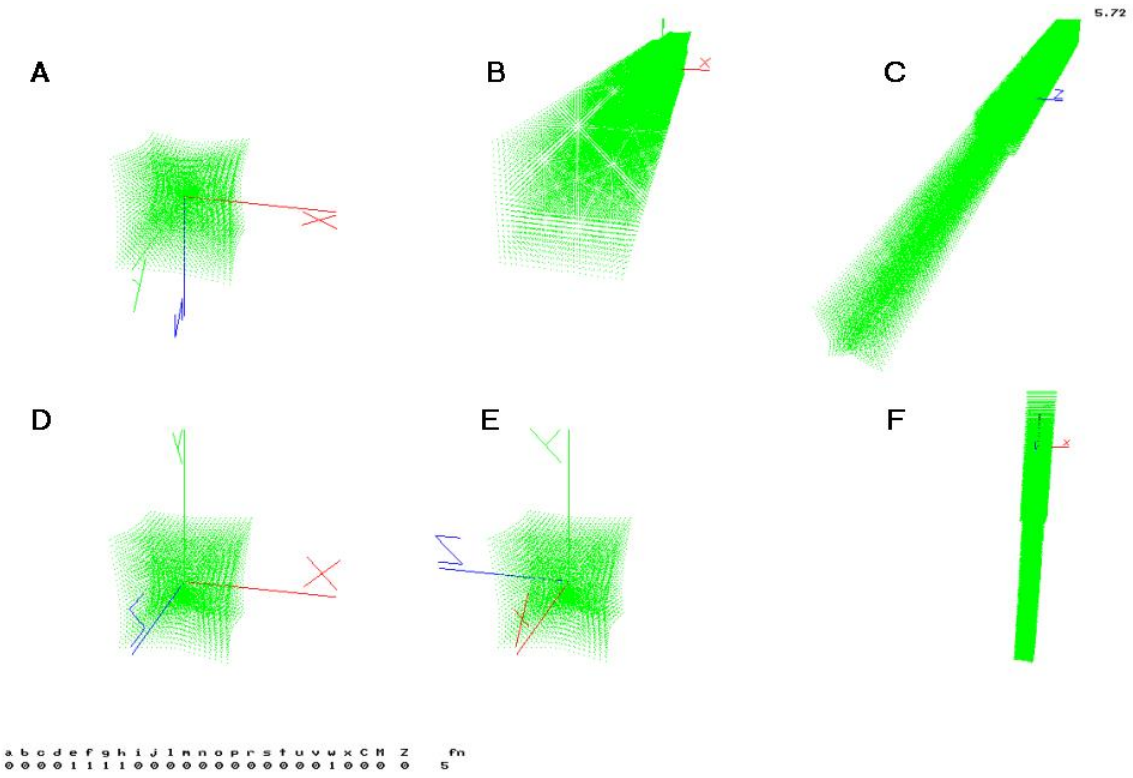


Figure 2.11: Perturbed 4D Spacetime Lattice
Shifted green dots show spacetime positions. Spacetime event history is displayed along the fourth or T (time) axis. The lattice of uninteresting unperturbed dots clutter up the display.

presence of a massive particle. In response to the question as to how the shape of this 'warped' spacetime around a massive particle can be represented, perturbations were added to the positions of the tick marks to represent distortions in the spacetime metric. As shown in Figure 2.11, the warp of spacetime can be depicted. But visualization of a discrete multi-dimensional spacetime introduced clutter and other problems which must be addressed.

Design Goals

This module was designed to explore multiple viewports of alternate POV's and alternate dimensions or Aspects. First, a warped lattice in 4-space was explored, then the algorithm was be extended to five dimensions.

The visualization package is required to:

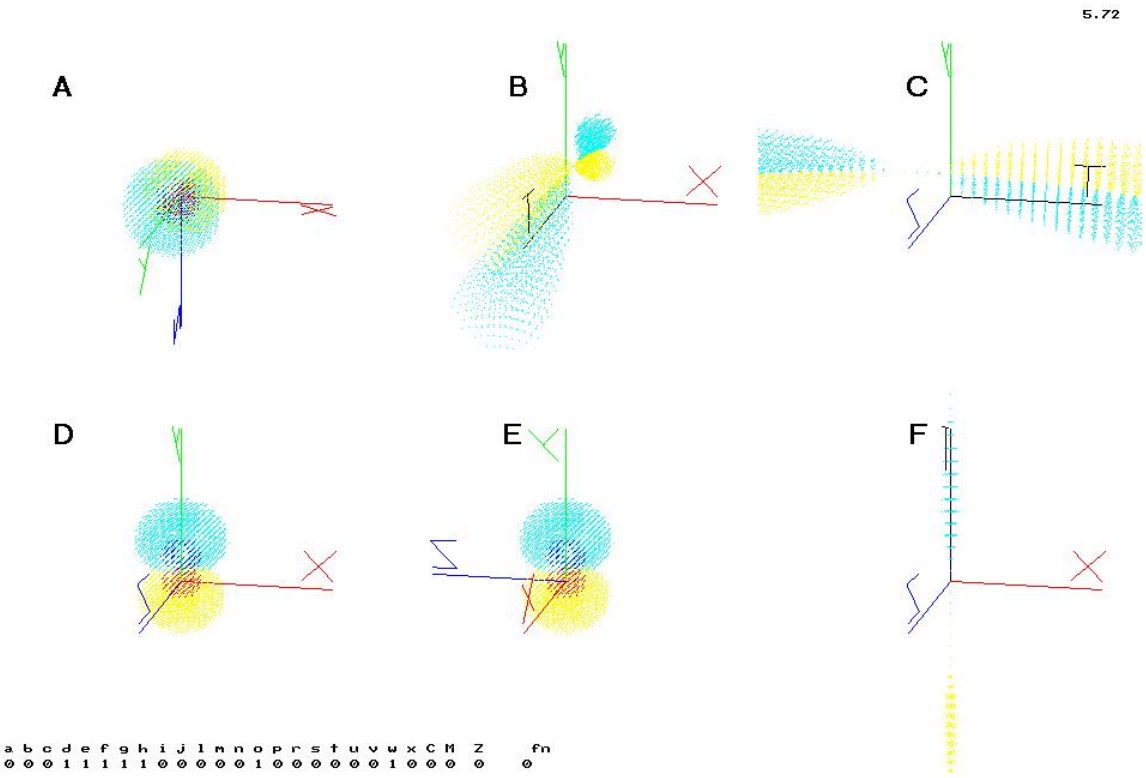


Figure 2.12: 1st Order Differential 4D Spacetime Lattice
Red shaded dots show spacetime compressing towards the event (or increasing attractive unipolar field strength) while blue shaded dots show spacetime expanding away from event (or decreasing attractive unipolar field strength). Spacetime event history is displayed along the fourth or T (time) axis.

1. View four or more dimensions simultaneously;

   Multiple 3D viewports must be displayed, where each contains an alternate Aspect of the n-space from a different Point-Of-View and with a different subset of dimensions.

2. Rotate, scroll, pan and zoom interactively via mouse click & drag;

   The views of each viewport, both individually and collectively, must be able to be rotated about each axis and translated along each axis in 3D via mouse click & drag actions.

3. Visualize causal retarded time contributions;

   Retarded causal signals, such as light waves or gravity, suffer propagation

delays from the source due the finite speed of transmission. A capability, similar to the emergent retarded time effects of 4D ray tracing (as requested in Section 2.3.4), is required if any relativistic effects are to be depicted.

4. Visualize dynamically warping spacetime;

   A method must be implemented to visualize dynamic spacetime warping such that the effects of mass curving space can be seen. The method should be extensible to an arbitrary dimensionality.

5. View velocity, acceleration, and jerk effects.

   A method must be identified to visualize the dynamics of spacetime warping that is extensible to an arbitrary dimensionality.

6. Test and demonstration

   A process must be selected to predictably perturb the lattice grid.

7. The Fifth Dimension

   A goal is to extend the algorithm into extra dimensions, 5D for example.

8. Animation generation and storage

   A mechanism to record the animated sequences and store them as viewable videos is required.

2.4.2 Implementation

The system was implemented as a 5D floating point array of two 7D vectors and three scalars at each lattice cell. The extent of each dimension in the array represents the size of the world-space in that dimension - for example, the default X dimension is -10 to +10 in steps of 0.5 units. The T axis is a 4D FIFO of the saved

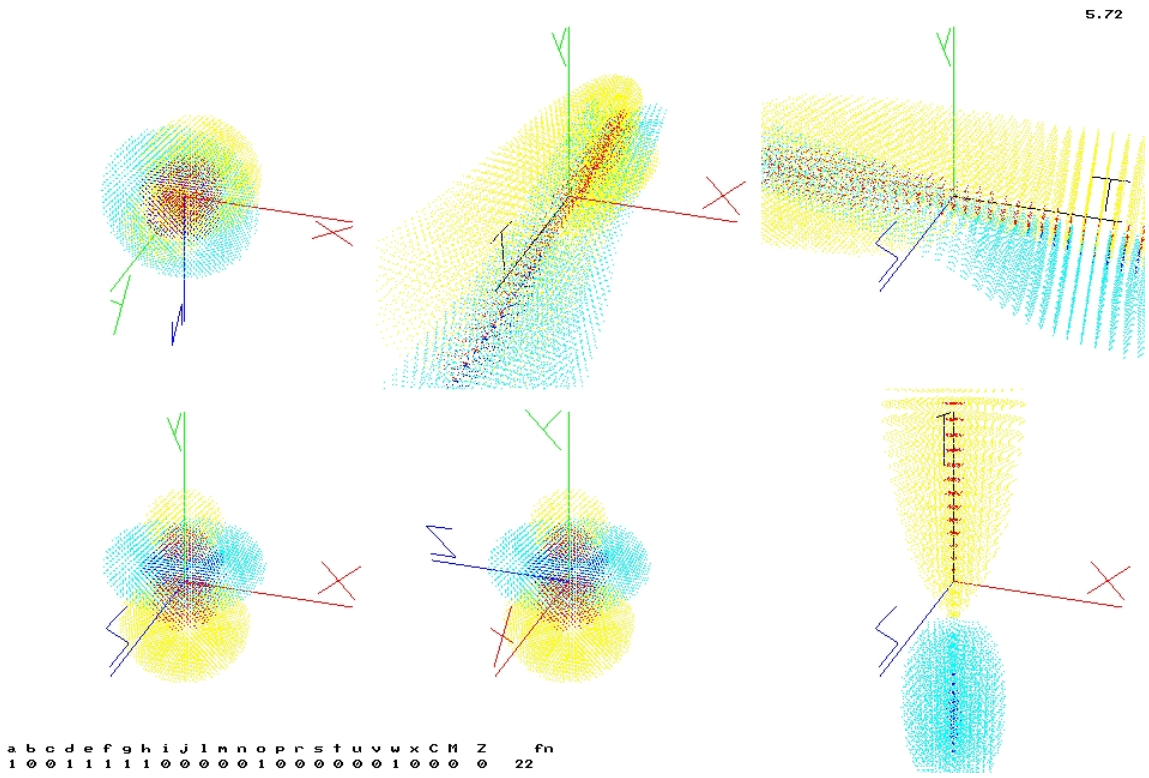


Figure 2.13: 2nd Order Differential 4D Spacetime Lattice
Red shaded dots show spacetime acceleration towards the event (an increase in the rate of the increase in attractive unipolar field strength) while blue shaded dots show spacetime accelerating away from event (a decrease in the rate of decrease of strength).

4-space history of user specified duration. This is a classical rather than a Minkowski spacetime T axis.

The display shows a green colored tick mark for each lattice cell in its default position (Figure 2.10). The cell position can be modified as described below and shown in Figure 2.11. The clutter from the other unperturbed cells makes it difficult to see the relevant information.

View four or more dimensions simultaneously

As shown in Figure 2.12 four dimensions are displayed simultaneously. Three 3D CAD-like representations are shown as Top, Front and Side in viewports A,D, & E, respectively. These 3D views are all slices of the T axis orthogonal to the T axis at its initial minimum position. Viewports B, C, & F depict the projections of the

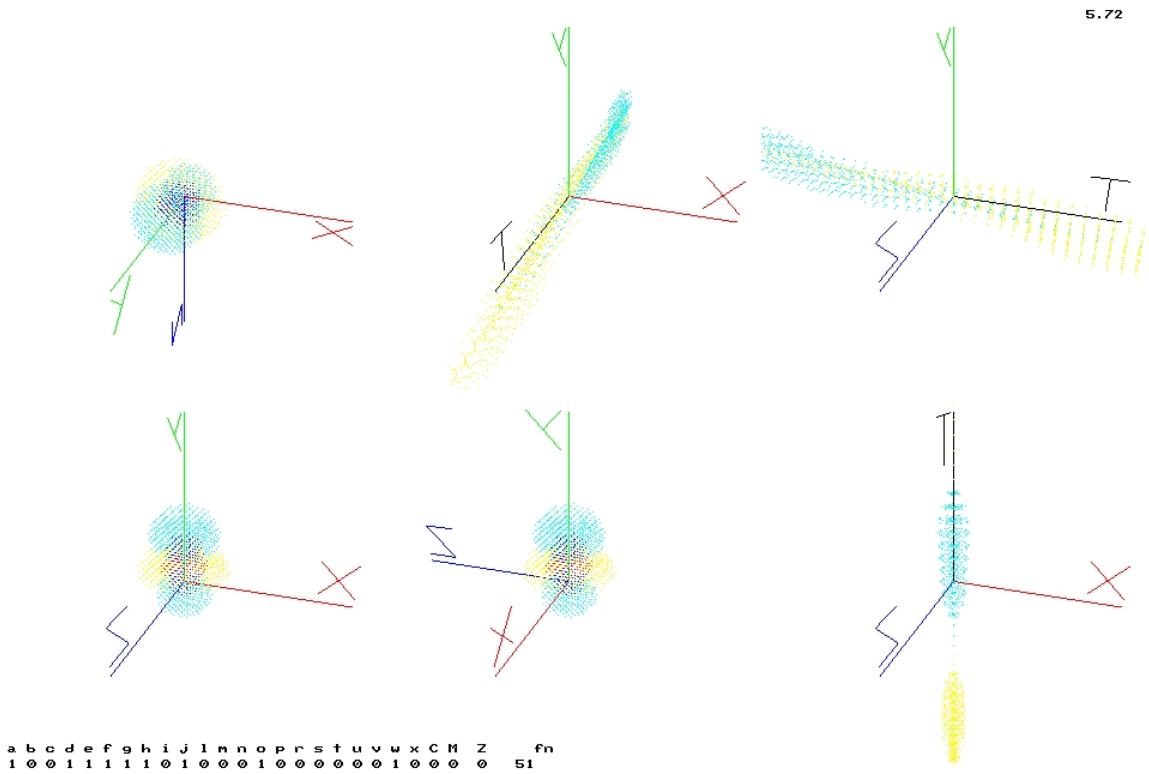


Figure 2.14: 3rd Order Differential 4D Spacetime Lattice
Red shaded dots show spacetime "jerk" towards the event (an increase in the rate of the increase's increase in attractive unipolar field strength) while blue shaded dots show spacetime accelerating away from event (a decrease in the decrease's rate of decrease of strength). The spacetime event history shows the "bow wave" along the fourth or T (time) axis.

4-space to 3-space along the Z, X, & Y axes, respectively.

The interactive operation and recorded animation sequences will show the 3D depiction in motion while the 3D slice propagates along the time axis. The time axis, which contains the 3D (or 4D) history can be used by the relativistic mode to compute the delayed-time contributions to the depiction.

Figure 2.15 is an example of the 3x3 5D viewport format with the fifth or W axis in the rightmost column. The three rightmost viewports depict from top to bottom the $WXY\_$, W_YZ, & WX_Z 3D views, respectively. The dimensional reduction is performed conceptually by slicing the 5-space orthogonal to the T axis at t_0 and then slicing the resultant 4-space orthogonally to each of the views complementary 4D spatial axes: Z, X, & Y. The displayed value of the W axis represents the time

dilation required of the T dimension in order for the hyper-voxel volume to be Lorentz invariant.

Rotate, scroll, pan and zoom interactively via mouse click & drag

The user can manipulate the 3D view matrices for each of the six viewports via the mouse. While viewports B, C, & F can be manipulated individually, viewports A, D, & E share the same matrix, which is composed with the B, C, & F matrices so that all six viewports move in concert when A, D, & E are changed.
The *left mouse click & drag*[7] action rotates the view to follow the mouse, middle mouse click & drag performs pan & scroll (translation), while right mouse click & drag will zoom.
Some of these operations may be easier to use with constraints imposed on how the cursor follows the mouse. For example, allow the user the option of using sliders, thumb-bars, and even text fields to specify rotation angles and position in a control panel as implemented in the online HyperCuber Applet.[15]

Visualize causal retarded time contributions

A Euclidean time axis, symmetric with the spatial axes, should suffice as a record of the spatial dimension's history, to be used for retarded time signals. This implementation should not aversely affect interactive performance. Alternative asymmetric non-Euclidean approaches for representing the T (time) axis relevant to Special Relativity and General Relativity are given in Section 8 *Future Work*.

[7]Left [Middle or Right] mouse click & drag refers to depressing the left [middle or right mouse button while the mouse cursor is over the object in question and moving the mouse without releasing the button until the desired operation is complete.

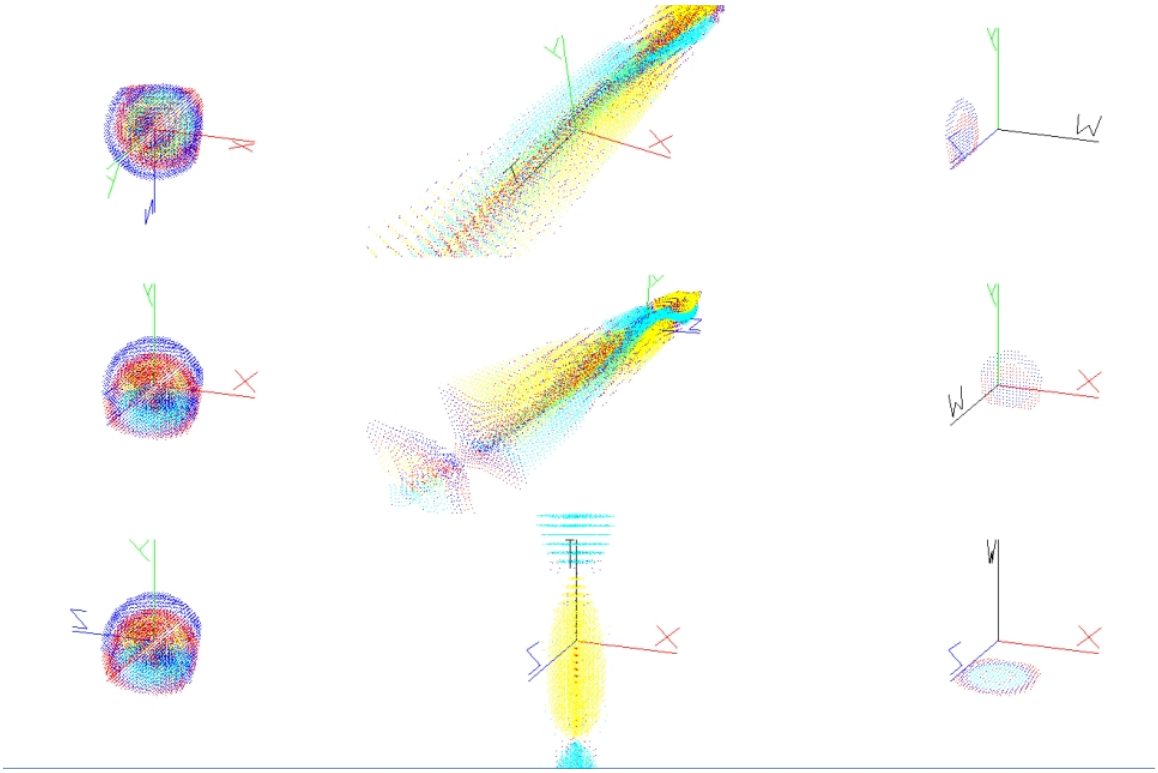


Figure 2.15: 2nd Order Differential in 5D Spacetime Lattice
The rightmost column depicts a 3D slice of the 5-space at t_0 and Z=0, X=0, & Y=0 for each of Top, Middle, & Bottom, respectively As in other figures, shaded dots show cell's acceleration status. Note the bow-wave on the T axis.

Visualize dynamically warping spacetime

The visualization provides realtime graphic double-buffer updating. The observer is privy to the dynamic changes in the position of the spacetime cells - or that is to say, the warpage or shape of space. Note that only the spacetime effects of the hypothetical gravitational source (the Source) are viewed, the Source itself is never displayed. The user must interpret the position of the Source from its effects on the lattice. This is intentional.

View velocity, acceleration, and jerk effects

The amplitude of the difference in the cell's prior position and the cell's perturbed position can be computed and colored by value. The cells whose difference in

position is below some threshold ϵ are not displayed, while cells in specified ranges are colored to indicate the range the cell's perturbation lies within, as in Figure 2.12. Since the cell's history is saved for at least six iterations, the second order derivative (Figure 2.13) and third order derivative (Figure 2.14) of cell's displacement and hence of the perturbing function can be similarly displayed via color coding. The display of the lower order differentials below some arbitrary delta value can also be suppressed.

Differential Display Modes

1. Differential display:

   Tick marks for static cells are removed, showing only dynamic cells' tick marks.

2. Stochastic display:

   Aliasing artifacts are suppressed by randomizing a tick mark's position within its cell.

3. Velocity, a first order differential display:

   The first order differential of lattice grid's position is displayed as red shaded[8] when compressing towards event, or blue shaded[9] when expanding away from event.

4. Acceleration, a second order differential display:

   The second order differential of the lattice grid's position is displayed. The cell is shown as red shaded when compression is increasing (accelerating towards

[8] **Red shaded** - a bright red dot indicates an increase in the numerical value in the direction of the event while yellow indicates a lessor value. Green indicates the nominal or 0 differential within some ϵ

[9] **Blue shaded** - a bright blue dot indicates an increase in the numerical value away from the event while cyan indicates a lessor value. Green indicates the nominal or 0 differential within some ϵ

the event), or blue shaded when compression is decreasing (accelerating away from event). This mode has a higher precedence than and overwrites the first order differential mode.

5. Jerk, a third order differential display:

   The third order differential of the lattice grid's position is displayed. The cell is shown as red shaded (increasing acceleration towards event) or blue shaded (decreasing acceleration away from event). This mode is the highest precedence differential display.

Demonstration

The demonstration mechanism warps the grid by moving each lattice cell with respect to a hypothetical unipolar attractive source (Source).

Gravitation A simple function to perturb the positions of the cells has been introduced by hypothesizing an attractive force at the Source that affects each cell as a function of the inverse of the square of distance from the Source, conceptually similar to a Newtonian gravitational field generated by a gravitating mass at the Source. A scalar value proportional to the inverse of the square of the distance from the center of the cell to the position of the Source is stored with the cell in the lattice array. In addition the position of the cell is perturbed towards the Source by a distance proportional to this scalar value. Effectively, this value is used to scale the normalized *cell-to-Source* direction vector which is then added to the position. Each cell is thus offset towards the Source as a function of its distance from the Source. The colors are then set as a function of the difference between the cell's current scalar value and the cell's prior scalar values, depending on the order of the differential. Only the color is a function of the order of the differential, not the cell's position.

Oscillation The origin of the attractive force, call it a gravitating mass (Source), can be oscillated via a sine function along the Y axis to approximate the cyclical sinusoidal acceleration of a massive particle.

Orbit The sine oscillation can be converted to a circle in the XY plane by adding a cosine generator to the X axis.

Velocity A velocity in an arbitrary direction can be added to the oscillating or circling mass.

The Fifth Dimension

A fifth dimension can be enabled in the lattice array and in the display. The fifth axis W shows the Lorentz dilation of the T axis required to maintain a Lorentz invariant spacetime volume.

Animation generation and storage

An animation capability is provided where each frame is one slice along the fourth or T (time) axis. Each individual frame of the animation sequence is output as a .ppm images file and can then be compressed into a QuickTime .mov video file.

2.4.3 Observations

Sample frames of the animations are provided as figures 2.10 through 2.14.
Figure 2.10 shows the background of the static Euclidean E^4 4-space via a standard six viewport 3x2 representation of four dimensions. The conventional CAD three viewport *Top, Front, and Side* representation of the XYZ Aspect of a 3D object is depicted in the three viewports labeled A, D, & E, respectively. The remaining three viewports labeled B, C, & F show projections of the 4-space into 3-space along the Z, X, and Y axes, respectively. The origin on the T-axis, the t_0 position, is the

minimum value of t (there are no values less than 0.0). The position of the axes in the views containing the T-axis is the mid-point of the T-axis, not its origin. The XYZ view may be considered to be a 3D slice through the $TXYZ$ 4-space at t_0. Introducing an event into the lattice, such as a unipolar attractive Source represented by a perturbation of the tick marks, is hidden by the clutter of the lattice as shown in Figure 2.11.

Removing all the ticks whose perturbation is less than some ϵ cleans up the image nicely showing only those areas in spacetime that are of interest. Furthermore, adding a color code to those ticks whose differential lies within certain bands can depict relevant information about the spacetime perturbation. For example, in Figure 2.12 the voxels experiencing an increase in the attractive force $\geq \epsilon_r$ are red shaded, while those experiencing a decrease in the attractive force $\leq \epsilon_b$ are blue shaded. This figure uses the same viewport representation as Figure 2.10, where A, D, & E are 3D XYZ views, and B, C, & F are 4D projections into 3D.

Pursuing this strategy further suggests that second and third order differentials can be likewise represented by red or blue shading. Figure 2.13 demonstrates a 2nd order differential, comparable to acceleration in the gravity analogy, with two red shade intensities represented by red and yellow indicating an increase in velocity (acceleration) of the tick mark towards the Source, while the two blue shade intensities represented by blue and cyan indicate a decrease in the velocity (deceleration) towards the Source.

A 3rd order term, jerk (sometimes known as jolt, shock, surge, lurch or super-acceleration), which is the derivative of acceleration with respect to time is represented in Figure 2.14 in the manner of the prior two data representations with regard to red shading and blue shading.

Jounce or snap are sometimes used to refer to the fourth order term, and are mentioned here for completeness. While crackle & pop are sometimes facetiously

used (as in snap, crackle, & pop) for the fifth and sixth derivatives of position with respect to time, respectively.

To remove aliasing artifacts introduced by the regularity of the grid, a stochastic display mode was implemented as an option that randomly positions each dot within its cell. This stochastic randomization procedure could have used a Gaussian distribution to conform to the quantum ground state in the Standard Model, but the compute time was deemed too expensive for the visual return. The stochastic feature was not used for figures 2.10-2.14.

There is no storage compression algorithm in this implementation. The storage complexity is $\mathcal{O}(L^n)$ where n is the number of visualized dimensions, and L is the length of the edge of the n-space. In the examples here, L is 10 and n is 4, requiring 10,000 storage cells. For the example if Figure 2.15, L is 10 and n is 5, requiring 100,000 storage cells. The Spaceslice implementation could effectively compress storage requirements by displaying only those cells containing a vertex of the nD object reducing storage complexity to V_{nD}.

Spacegrid performance data for the above 4-space examples is provided in Table 7.1 of Section 7.4.1 in the *Results and Evaluation* chapter, below. The matrices are composed as described in Appendix A, *Matrix Composition*, with computational complexity is as described in Section 1.4.1, *Matrix Composition Computational Complexity*.

It is important to note that a visualization system has been constructed here that depicts a function by the perturbation of the spacetime in which it is embedded. There is no explicit visual representation of the function, its source, or its cause. However, displaying the effect provides an unambiguous representation of the source. This is intentional and significant.

Chapter 3

Literature Review

This review is divided into three categories with regard to the type of visualization technology being addressed: Spatial Dimensions - where the dimensions to be viewed are treated as distances as in 3D space; Hypervolumes - where ray casting technology displays a 4D volume in 3-space; and Parametric Data - where a variable's value is represented, rather than a distance.

3.1 Spatial Dimensions

Software techniques for viewing four-dimensional objects had been implemented as early as 1963 when Noll plotted hypercubes with an IBM-7094.[26] In 1965 Noll[27] animated his earlier work by creating a computer-generated animation of the 3D projection of a rotating hypercube with a 16mm movie camera. According to Noll, "*Although no actual mental visualization of the fourth dimension resulted from the computer-generated displays, it was at least possible to visually display the projections and be puzzled in attempting to imagine the rigid four-dimensional hyperobject.*" In 1978 Banchoff[28] presented a film of a rotating wireframe of a hypercube at the International Congress of Mathematicians. In 1984, Black[29]

developed a real-time animation of a multi-colored solid 4D hypercube as a demo for the Calcomp Prisma[30] workstation.

In 1988, Beshers and Feiner[31] described a real-time implementation on an HP graphics workstation to rotate and project hypercubes and other 4D objects of comparable complexity into 3D. The implementation used W-axis near and far clipping planes to select a viewable 4D "hyper slice" orthogonal to the W-axis. 3-Flat intersection with a 4D object boundary, as described in Appendix B, was not implemented. In 1990, Feiner and Beshers explored interacting with 4D objects as nested heterogeneous coordinate systems[32].

Banks[33] also explored 4-space interactions via manipulation of 2D surfaces in 1992 in which he described 4D projection, the transparency performance challenge[1], and the binary space partition (BSP) challenge[2] for planes in 4-space. Both of these challenges were addressed by Chu, et al.[34] in 2009 with the GL4D API. Interesting questions raised by Banks that will be answered here include "*Are there effective ways to shape, to position, and to display the volume or its boundary interactively?*" and his concern about "*..loss of geometric content that transparency produces*" proposing the exploration of 4D silhouettes. Banks acknowledged *The Feynman Lectures on Physics*[35, 36, 37] as his inspiration for the physical principles of illumination of large codimensions in his 1994 paper.[38]

Andrew Hanson has provided comprehensive solutions to many 4D viewing challenges, not only to 4D geometry, but also 4D lighting. In addition to his technical contributions of more than 20 papers in the last two decades, Hanson advocated *The Visualization Principle*[39] as a working definition of success: "*A useful data depiction must allow the viewer to reconstruct a consistent and relevant model of the original data*".[39]. Hanson stated in 1992 "*..nor do we know that the*

[1] **Transparency challenge** - expensive in computational resources and impacts interactive performance.

[2] **BSP challenge** - a codimension one construct, a 3-flat, is required to divide a 4-space, whereas a 2-flat cannot.

desired interpretation of the images is cognitively feasible within the limitations of the human intellect".[40] In 1993, he worried that "*Non-convex or multiple objects would need additional attention to avoid the display of scene portions that should be occluded*".[41] The Spaceslice implementation described here solves this problem in real-time. While Lange and Dickson sculpted into plastic certain complex curves proposed by Hanson in 1990 and 1991, these models are not representative of higher dimensional objects in the same sense that Spaceslice's plucked 3D objects are representative of 3-manifolds. Hanson also explored higher dimensional geodesic interpolations and splines. The Meshview program was adapted by Hanson's group to viewing two-dimensional manifolds in the fourth dimension in the late 1990's.[42] Future plans included interactive 4D rotations as currently performed by Spaceslice. Like Lonsway[43] in 2002, Hanson was also interested in multi-modal 4D exploration including haptics as evidenced by a 2005 paper[44] describing the exploration of a two-torus in four-space similar to the two-torus of Figure 6.3. More recently, Chu, Hanson, et al. in 2009 described their GL4D[34] API that adapts a state-of-the-art rendering pipeline to hypervolume ray-casting. Their future plans include adapting contemporary video game haptic technology and time-dependent data to 4D visualization.

In their survey of multidimensional multivariate visualization[45] in 1997, Wong and Bergeron concluded that "*We must learn what approaches actually lead to more accurate results, enhanced productivity, and better understanding of the underlying science.*" This theme has been repeated consistently as noted in the summary, below.

4D illumination models were also explored by Hanson[40] in 1992 and by Banks[38] in 1994 by extrapolating 3D smooth shading using the surface normal into a strategy using the hyperplane equation whose coefficients were constructed from the minors of the determinant as described in Appendix B, *Derivation of Intersection of*

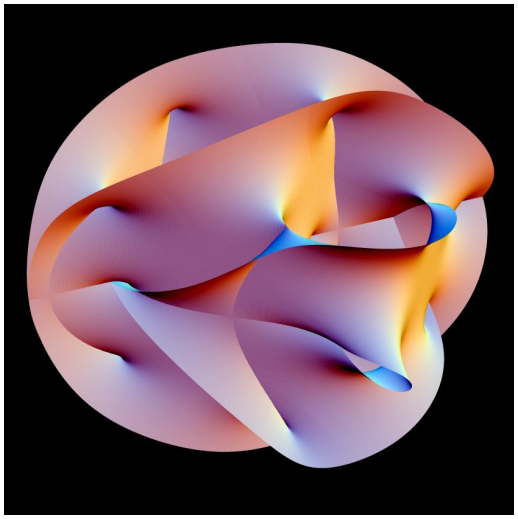


Figure 3.1: Depiction of 6D Calabi-Yau Space
An example of the Direct to 2D Technique.

a Line with a 3-Flat. Banks also used Kajiya's teddy bear fur[46] to display 3D vector fields. In his conclusions, when Hanson states, "*We can argue that* some intellect*, possibly superhuman, can in fact understand the 4D images we produce*", he is implying that there is enough information in the 2D images to unambiguously reconstruct the 4D model.

In 1993 Hanson & Cross[41] suggested thickening as a visualization tool, for example to extrude lines to tubes and planes to 3-manifolds "*...by adding a circle to each point of the surface's normal plane...*". As stated by Hanson & Munzer[47] in 1994 and quoted in the next chapter: "*Mathematical visualization is the art of creating a tangible experience with abstract mathematical objects and concepts*". *Graphics Gems IV*[48] contains a compendium of 4D formulas and insights also by Hanson[49].

In 1999 Hanson proposed to add "*interactive 4D rotations and volume rendering of 3-manifolds embedded in 4D*" to the Meshview 4D package developed at University

of Illinois at Urbana-Champaign.

In 2000, D'Zmura[50] of UCI created a 4D virtual world and reported, "*.. we have chosen to represent boundaries using a set of geometric primitives. This is the sort of representation that has proven so successful in 3-D computer graphics.*" He proved that his test subjects could indeed learn to find their way around in the 4-space maze of corridors.

McGuigan stated in the conclusion of his paper on viewing 4D Quantum Chromo Dynamics (QCD) data[51] in 2000 that, "*As there are many possible slices[,] animation can help to choose the interesting ones. The projection approach shows all the data at once. Rotating through the data uncover features obscure in a particular four dimensional view.*" In future work, he wished to include 5D effects to simulate parity, and the ability to visualize topological defects, as well as real-time steering of calculations.

Lonsway[43] in 2002 suggested the use of perceptual manifestations via additional sense modalities such as echoes, breezes, smells and temperature to express space.

In his 2004 paper on multiple view visualization of higher dimensional data[52], Tomov used projection to reduce dimensionality. Tight coupling among the different viewports and resultant coordinated exploration allowed the authors to extract insights about the data.

In 2007, Elmqvist[53] while studying 3D occlusion noted that it was necessary for "..users to be aware of occluded content without compromising visual content and imposing a high cognitive load on the user." He recognized that a multiple-viewport strategy wherein the various viewports allow augmented views of user selected targets would improve user comprehension. Spaceslice implements the strategies recommended by Tomov and Elmqvist, as will be shown later.

In his *Views on Visualization*2006 van Wijk made a few interesting observations, as follows. Visualization research can be considered a science in its own right, and that

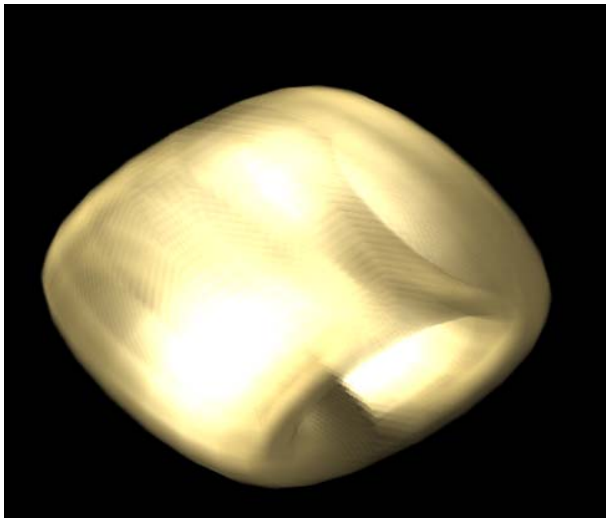


Figure 3.2: 4D Hypervolume Depiction of Hypertorus

elegance and beauty are worthy goals for this new science, as are effectiveness and efficiency. Visualization research should also lead to predictive theory and laws. Aguilera[54] in 2008 reiterates a quote (from Allen's *Human Spatial Memory: Remembering Where*[55] of 2004) questioning "*how the increasing use of videogames and more sophisticated virtual environments will change the way people think about and remember space and place.*"

3.2 Hypervolume Visualization

In 1998 Bajaj decided to abandon rendering and explore hypervolume visualization to discover a "*dimension independent viewing system that scales nicely with the geometric dimension of the dataset and that can be combined with classical approaches like isocontouring and animation of slices of nD data.*"
Most recently in 2009, Chu, Fu, Hanson, & Heng[34] combined nD volume rendering with nD lighting as shown in Figure 3.2.

3.3 Parametric Dimensions

In the domain of parametric space, Asimov[56] described a method of automatically "*touring*" a parametric space with a 2-frame (2D view plane).

Wegman[57] provides powerful pragmatic real-world results for nD visualization strategies for 4 through 68 dimensions and 22 to 280,000 data points. Tools included grand-tours, scatter plots and parallel coordinates.

Artero[58] in 2004 explored clustering of large multi-dimensional data using parallel coordinates.

Tomov[52] explored non-spatial data in a multi-view spatial visualization performing dimensional reduction by projection and intersection.

The unnaturalness of these non-intuitive parametric space strategies places an unnecessary cognitive load on the user attempting to interpret the data. Providing an analog to a real world experience, such as our experience with the size and shape of 3D objects, is preferred.

3.4 Summary of Researchers' Published Comments

> Hitching our research to someone elses driving problems, and solving those problems on the owners terms, leads us to richer computer science research.
>
> - Fred Brooks, PhD[59]

Following fifty years of research from Noll's expression of puzzlement[26] to Hanson's *The Visualization Principle*[40] leads to the objective of this project: to provide an ability for the users to interactively and tangibly manipulate and explore the puzzling aspects of the extra dimensional models before them.

Geometric primitives[50] have been used *to shape, to position, and to display the*

volume or its boundary interactively[33] using the data structure defined in Section 6.

McGuigan's animation statement[51] adds support for the feature set of all three implementations as discussed in sections 2.3, 2.4 and 6, specifically the animated rotating intersection option as shown in the frames of the sliced 3-sphere in Figure 6.8 and the sliced 3-torus in Figure 6.9.[3]

Research into *computer graphics and visualization* cannot exist in a vacuum. Rather it exists by, and for, its relation to its sister sciences. An algorithm needs an implementation, and an implementation needs an application. Developing strategies to interactively visualize the theories of relativity is a challenge interesting enough to motivate these algorithms.

While speaking of an application specific visualization, Hanson[62] recommended that the interesting physical effects be incorporated into the visualization "*.. in ways that are intuitive and qualitatively correct without being obtrusive.*" This concept can be extended to extra dimensions as well as relativity. Furthermore, Weiskopf[63] offered "*.. intuition is particularly improved by establishing a tight link between the user and the visualization environment. Therefore, real-time applications and interactivity should be employed whenever possible*".

[3]Videos of the animations may still be available either on the accompanying Compact Disc (CD) or on the internet[60, 61] at http://visualization.com/videos/uci/figure-6.8.mov and http://visualization.com/videos/uci/figure-6.9..mov

Chapter 4

Problem Definition

"Mathematical visualization is the art of creating a tangible experience with abstract mathematical objects and concepts"

Tamara Munzer, PhD[47]

Visualizing and exploring the shape of objects of dimension greater than three has been a thorny issue for generations. Prior to the research here, there has been no easy way to explore models of extra dimensions.

For a solution to be comprehensive it is required to include: An efficient internal data representation and associated display technology that can support interactive frame rates;[1] An intuitive extra-dimensional (ED) Graphics User Interface that can seamlessly support 4D, 5D, and higher dimensions; A view methodology that can seamlessly display 4D, 5D, and higher dimensional objects. An easy to code Off-The-Shelf Open API would be a perk.

Given the success of interactive viewing of 3D models, 3D model animation, videogames and virtual reality, it is reasonable to adapt that technology to viewing objects of extra dimensions.

Consider, one can remember the result of a 3D operation such as catching a ball, or slicing an apple. One can also extrapolate those operations to similar activities such

[1] **Interactive frame rate** values acceptable by the US military are defined in MIL-STD-1472F [64] and shown in Figure 7.9.

as bouncing a ball or slicing a pear, or even slicing a donut. These 3D manipulations are considered to be intuitive, perhaps because one remembers a history of similar activities to which the proposed manipulation can be compared. Unlike remembering a dynamic 3D event, like catching ball, or slicing an apple, envisioning the result of manipulation of 4D events in extra dimensions is not intuitive. While most of us can navigate a multi-storey building or catch a ball, few of us have had experience negotiating a 4D maze[65] or observing the shape of a sliced 4D object. Even an activity as simple as examining a 3D ball has no empirical counterpart in 4-space.

Envisioning extra dimensions, unlike imagining a 3D dynamic event such as catching a ball, cannot be intuitive without comparable experience. One can speculate that experience with manipulating extra dimensions may enhance one's intuition about objects of extra dimensions.

How does one go about manipulating 4D objects in higher dimensions? As noted in 1.3 - *A Brief Introduction to 3D Graphics*, a well developed 3D CAD technology already exists. Furthermore, many users are sufficiently facile with the technology to lend credence to the claim that interactive manipulation of a mathematical object via computer technology, such as *click & drag* for example, gives the user a tangible experience with the object in question. A tangible experience is the ultimate goal. Therefore, devising a mechanism to convert a 4D object into a 3D environment for handoff to a CAD-like or video-game system is the penultimate objective.[2]

The challenge is to convert a 4D object in 4-space into a representative 3D model in 3-space. We shall identify a 4D model in the same manner as we would identify a 3D model. If it looks like an apple, it is an apple. If it looks like a sliced donut, it is a sliced donut.

[2]A user study is not necessary to demonstrate that 3D CAD is an effective 3D model visualization and manipulation strategy. A quarter century of industrial adoption is adequate proof of the technology's efficacy. Discovery of better strategies is left as future work.

The external shape of the object is characterized by the convex hull. In the case of a 3D apple, it is the appearance of the shape of the 2D surface encompassing the 3D apple that identifies the object as an apple. For the sliced donut, it is the symmetries of the cross-section of the object, the slices, that characterizes the object as a donut.

The proposed solution should also include the capabilities of the prior work as given below in sections 4.1 and 4.2.

4.1 Viewing: 4D Spacetime

If the four dimensions are considered to be a 4D spacetime, rather than a spatial 4-space, then there are special challenges with respect to classical retarded time. That is to say that the light from the points of the object furthest from the Point-of-View (POV) will take longer to arrive at the viewer than points closer to the POV. This effect can be detected with large objects at sufficient distances even at non-relativistic velocities.

If relativistic objects in spacetime are to be viewed accurately, then retarded time must be taken into account. The representation of the 4D object should emerge dimensionally reduced to 2D, and a physically correct view of classical aberration should emerge from the method. There are geometrical issues that must be addressed in order to view the object as perturbed by relativistic effects.

4.2 Exploring: the Warp of Space

How can one view distortions, warps, and curves in the very *shape of space* itself? Space and spacetime can be non-Euclidean, and dynamically so. The challenge is compounded since it would be of interest to see the dynamic relationship between voxels in space, and hyper-voxels in spacetime and extra dimensions. More

information can be gleaned from the differentials of the voxels' spatial relationships - that is not only their warps, but their rates of warpage to the second and third orders. In a spacetime predicted by the Theory of General Relativity this would provide information about the magnitudes and directions of matter and energy flows.

A fourth dimension should be able to be treated as temporal or spatial, and displayed accordingly.

4.3 Exploring: nD Models in n-Space

The challenge to be addressed here is the creation and display of nD models. Firstly, a strategy must be defined to efficiently encode an nD model or n-manifold with boundary in an easy to use database from a mathematical, programmatic, or interactive description.

Secondly, a method must be selected to display the database model on the desktop computer ultimately so as to make it tangible for the user. There user should be able to see the total extent of the nD model in each of its dimensions so as to be able to grasp its size and shape in nD. Given that the computer has a 2D viewport, a technique must be devised to display it in a form comprehensive, comprehensible, and memorable to the user.

The user should be able to explore the object's shape in all its dimensions, interactively.

4.4 Manipulating: mD Models in n-Space

In order to manipulate an mD model in extra dimensions, we must have a Extra Dimensional graphics user interface (ED-GUI) of dimensionality equivalent to the dimensionality of the n-space in which the mD object is embedded where $m \leq n$.

This implies that an ED-GUI is required even for manipulating a 2D object in all possible degrees of freedom in 4-space, as well as for a 4D object in 4-space.

It should be possible for the user to manipulate the mD model in some manner in order to explore the interior of the mD model's n-space. A method must be devised to provide an interactive tool to the user that enables penetration as well as exploration.

A useful feature would be the ability to select and export a 3D object to a file for editing with a conventional 3D CAD or 3D viewing package. For example, an object can be exported to a .ply file for importation by a PLY file viewer.

Chapter 5

Approach and Contributions

Intelligence Amplification is better than Artificial Intelligence

- Fred Brooks, PhD

The research here describes and demonstrates a mechanism for the visualization, exploration, and manipulation of extra dimensional models on the personal desktop computer. Provided here are a system of algorithms to create, view, and manipulate four-dimensional models in both Euclidean four-space and Minkowski spacetime. Furthermore the foundation is laid for algorithms to manipulate yet higher dimensional mathematical models in non-Euclidean spaces. Since these mechanisms are a proof-of-concept rather than a marketable product, certain well-considered shortcuts have been taken to reduce the traditionally labor intensive software development time.

5.1 Multi-Aspect Viewer Approach (Spacegrid)

A 4D lattice provides views of models of an empty static or an empty dynamic curved or warpable non-Euclidean 3-space. In the latter case, the position of a perturbing object can be inferred from the relative perturbation of the 3-space itself as indicated by the perturbation of the lattice points. The realtime feature allows

the user to introduce dynamic variation in the display for exploration and comparison. The implementation is easily expanded into a 5D lattice representation of five-space, and perhaps more, limited only by memory and compute power.
This strategy also provides visualization of retarded time effects that are critical to relativistic visualization and inherent in the raytracing implementation, but at a level of performance not possible with contemporary raytracing.
The hypervoxels in n-space can be represented by points in an nD dynamic lattice and easily sliced parallel to the extra dimension to perform a simplistic dimensional reduction. Furthermore, the points' position in n-space can be dynamically perturbed to represent a *warp in space* as described as an approximation to the Theory of General Relativity or other theoretical physical force.
This fourth dimension can be treated as temporal or spatial, and displayed accordingly.
It should be possible to explore by using multiple viewports, each of which shows an alternate Aspect of the n-space, and rotating the 3-space, n-space, or the point-of-view (POV) through the space using the mouse and a click & drag paradigm.

5.2 Projection & Intersection Approach (Spaceslice)

In this approach the geometric model of an mD object is created and then loaded into an nD workspace for viewing. The viewpoint is managed by the user through an Extra Dimensional Graphics User Interface (ED-GUI) designed for use with mD models in n-space. For this implementation, $4 \leq m \leq n \leq 5$.
A description of a GUI for extra dimensions and a description of the construction of the hyperobject will be followed by a discussion of the two dimensional reduction

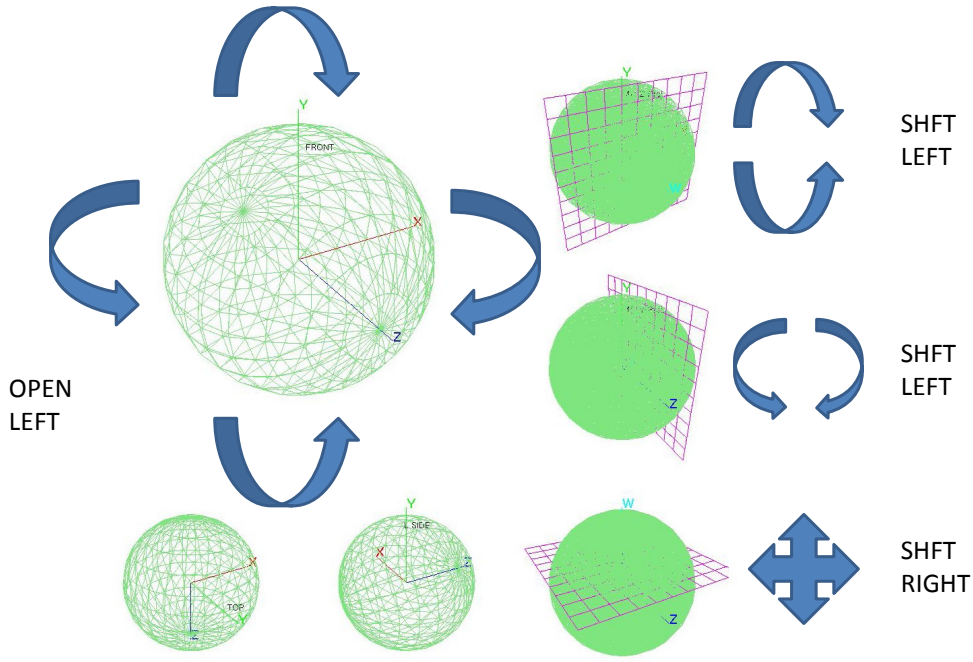


Figure 5.1: The 4D GUI

techniques used here: projection and intersection.

5.2.1 ED-GUI: Extra Dimensional Graphics User Interface

An nD Graphics User Interface (ED-GUI) allows the user to rotate any axis into any other axis of the nD object, and to move (translate) the nD object along any axis. The ED-GUI supports rotation in any viewport by a left-click[1] in any viewport, and a drag[2] of the view up or down, and left or right. The viewed object rotates to follow the cursor. Rotating the large upper-left viewport causes all viewports to rotate or translate in concert. Right-click and drag will translate the object in the viewport to follow the cursor. The large upper-left viewport addresses all the viewports simultaneously as did the rotation operation. The user can thus

[1] **Left-click** - place the cursor on the object and click the left mouse button.
[2] **Drag** - drag the cursor with the mouse while not releasing the mouse button.

reposition any view via rotation or translation.

The nD ED-GUI is also required to allow the user to specify the nD orientation and nD position of the intersecting hyperplane in the model's n-space. A 3D grid-like icon visible in the right column of Figure 5.1, has been created as a 3D cursor to represent the orientation and position of the three-flat in hyperspace. This three-flat icon is depicted in each of the viewports by the Aspects appropriate to that set of dimensions. For example, in a viewport where the Z axis had been compactified so that only W, X,& Y are visible as in the top-right viewport of Figure 5.1, the three-flat icon consists of only its X & Y dimensions in this orientation, and so appears as a gridded plane in the XY 2-space with no extent along the W-axis. In the lower-right WXZ viewport the three-flat icon consists only of its X & Z dimensions, and is compactified along the W-axis, as depicted by the gridded 2D plane in XY 2-space.

Each view of the three-flat icon will allow the user to *click & drag* the icon through the displayed axes within the convex hull, and hence reposition the intersecting three-flat hyperplane at any orientation or position within the n-space, effectively providing the hyperplane with $n + \binom{n}{2}$ degrees of freedom[3].

The interactive implementation of this strategy gives the user a tangible experience with the 4D model. The user shall be able to explore the object by interactively repositioning the 3-flat and observing the resultant model of the slice in 3D.

5.2.2 nD Object Definition

A 4D object , or a closed compact three-manifold describing a 4D object in a closed finite metric Euclidean 4-space, is intersected by a three-flat and the intersection displayed as a two-manifold in three-space. Similarly, a 5D object in five-space can be intersected and the intersection displayed as a three-manifold in four-space, then

[3]**Degrees of Freedom** (DoF) - one translation for each axis (n), plus one rotation for each unique pair of axes ($\binom{n}{2}$). 3D requires 6 DoF and 4D requires 10 DoF

again intersected to yield a two-manifold in three-space. The implication is that the algorithm can be extended to yet higher dimensions, as processing resources become available. The details are described in Section 6.2.2.

5.2.3 Exploration: Dimensional Reduction by Projection onto Hyperplane

In lay terms, in this and the following approach, a model of an nD object is created as described above and displayed via dimensional reduction in 3D. Multiple viewports display different aspects of the nD model. The model is dimensionally reduced by an orthogonal projection onto a user positioned and oriented 3D hyperplane or 3-flat. The resultant 3D convex hull is displayed. The user shall be able to explore the object by interactively repositioning the 3-flat and observing the model in 3D. The rendering of the nD object shall be accomplished by rotating the object into the hyperplane's space, projecting into the 3-flat's 3D space along the 3-flat's orthogonal axes, then de-rotating the resultant 3D object back into the 3D view frustum. The matrix composition is described in detail in Section A.
The view of the object shall be that obtained by the 3D observer within the Euclidean 3-flat. Unfortunately performance is impaired since the projection operation causes all the object's vertices and connecting edges and faces to be rendered, although only the outer vertices and faces are visible.

5.2.4 Manipulation: Dimensional Reduction by Intersection with Hyperplane

In this as in the previous approach, the mD object, an m-manifold with boundary, is specified by defining its closed compact boundary via a pure simplicial (m-1)-complex. The mD object is then dimensionally reduced for display in 3D via

projection. Multiple viewports display different Aspects of the mD model. The 4D model is dimensionally reduced by both projection as described above, and intersection with the same user positioned and oriented 3-flat hyperplane. The 3-flat slices the 4D object and displays the 3D slice to reveal heretofore hidden internal 3D structure. The mD model will be sliced by intersecting the (m-1)-simplices with an (m-1)-flat. This process is repeated for higher-dimensional objects for $m > 4$ while decrementing m. For $m = 4$, the manifold's bounding 3D tetrahedra are intersected with the 3-flat to yield bounding 2D triangles describing the closed compact 2D surface of the revealed 3-manifold with boundary.

The resulting 3-manifold with boundary (2-manifold) can then be manipulated in the usual ways with a conventional 3D CAD system, or rotated and viewed with the embedded *click & drag* 3D viewer. The user can thus drag the 3-flat through the model to explore the internal structure of the 4D (or higher) model yielding a tangible 3D experience as defined above in Section 4.

An additional feature to enhance the tangibility of the object will be the ability for the user to export any of the resultant plucked 3D models to a 3D CAD system for visualization and editing.

5.3 This Research's Contributions to the Art

This research has seamlessly merged novel approaches to create a comprehensive solution to the interactive visualization challenges as follows.

5.3.1 A Computationally Tractable Method to Explore Euclidean Four-space

Euclidean four-space (and five-space) is probed with a 3D hyperplane to reveal and extract 3D structure hidden within the higher dimensional spaces in realtime.[4]
A new simple data structure and algorithm yields realtime performance with 4D models, and near realtime performance with 5D models. In both cases, the performance enhances the generation of animated videos on the personal computer. The data structure and algorithm are configured to allow for the implementation of multi-processor multi-threaded strategies. More threads will enable more interactive extra dimensions.

5.3.2 Making the Intangible Tangible

Simple and natural interactive metaphors - rotate, pan, and zoom via ubiquitous *click & drag* - allows the exploration and manipulation of extra dimensional objects to become tangible to the user in the same manner that an engineer's 3D CAD construct becomes tangible to the designer. The capability of being able to export the "*plucked*" 2-manifold as a 3D object to a 3D viewer or third-party CAD package lends an accessibility and reality to the 3-manifold heretofore unavailable to the user.

5.3.3 Multi-Dimensional Viewports

The common CAD type viewport has been adapted to the visualization of higher dimensional objects via multiple viewports, each displaying an alternate set of dimensions or *aspects* of the same extra dimensional object.
The uncluttered attractive views and the easy-to-use click & drag Graphics User Interface reduces the cognitive load on the user promoting creative thinking and

[4]Realtime as defined by MIL-STD-1472F[64] and shown in Figure 7.9.

problem solving.[63]

5.3.4 Non-Euclidean n-Space

Displaying projections and intersections of non-Euclidean n-space is an inherently difficult problem. It can be greatly simplified by approximating the curve, warp and distortion with n-simplices at some arbitrary resolution. Including the prior work of the *Spacegrid* implementation described in Section 2.4 could provide projections and intersections of non-Euclidean n-space at an additional computational complexity of $\mathcal{O}(V_{4D})$. The individual vertices could be perturbed during the vertex duplication phase of the intersection process. Furthermore, the resultant simplices could be colored to represent the value of their first, second, or third order derivatives as was the *Spacegrid* display. Thus the foundations for a computationally tractable method to explore a dynamic non-Euclidean n-space can be established here.

Chapter 6

Solution - Plucking an Aspect

"The frank realization that physical science is concerned with a world of shadows is one of the most significant of recent advances."

Sir Arthur Stanley Eddington

6.1 Introduction

Embedded within a 3D sphere (or 2-sphere) are an infinite number of 2D circles (or 1-spheres). A circle can be extracted from the sphere by slicing the sphere with a plane as in Figure 7.5. Topologists can describe this result mathematically for 3-space:

$$\begin{aligned} codim(sphere \cap plane) &= codim(sphere) + codim(plane) \\ dim(sphere \cap plane) &= 1\text{-}manifold \end{aligned} \tag{6.1}$$

In like manner, a 4D sphere (or 3-sphere) contains an infinity of 3D spheres (or 2-spheres). Extracting the 2-sphere with a 3D hyperplane (3-flat) can be described

mathematically for 4-space as:

$$\begin{aligned} codim(\text{3-}sphere \cap \text{3-}flat) &= codim(\text{3-}sphere) + codim(\text{3-}flat) \\ dim(\text{3-}sphere \cap \text{3-}flat) &= \text{2-}manifold \end{aligned} \tag{6.2}$$

Presented here are a fundamental data structure and attendant library of tools to define and display an n-dimensional (nD) model by describing its (n-1)-dimensional bounding simplices. The Test-Fixture as described and demonstrated here[1] can both generate the bounding three-simplices bordering a mathematical four-dimensional (4D) model at a specified level-of-detail; and allow the user to interactively explore this three-manifold in real-time by selecting three-dimensional (3D) projections and intersections while viewing the resultant 3D object just as does an architect, engineer, or video gamer.

6.2 Implementation

The design goal of interactivity was met by transforming the intersecting hyperplane rather than transforming the database of vertices, and by using an infinite hyperplane for intersection thus relieving the software of the necessity of clipping the nD objects.

Given that an m-manifold in n-space, consisting of s simplices, will require $\mathcal{O}(s)$ vertices[2] and n components per vertex, $\mathcal{O}(s \times n^2)$ MAC's would be required to transform the object's s vertices by the $n \times n$ matrix described in Section 1.4.1. Furthermore, each of the s vertices would have had to have been clipped against $2n$ clipping planes for an additional $\mathcal{O}(s \times n)$ operations.

[1] While the Test-Fixture was implemented in seven Euclidean dimensions, and is extensible to yet more, only four dimensions are described and demonstrated here.

[2] As with a 2D triangular strip bounding a closed compact connected 3-manifold, a 3D tetrahedral strip bounding a closed compact connected 4-manifold will require one vertex for each additional simplex connected to the initial simplex.

However, only one *vector*×matrix multiplication is required for the hyperplane equation to transform the 3-flat, and no clipping operations. Only the vertices of the plucked 3D object need be transformed and clipped for viewing, and these 3D vertices are amenable to interactive manipulation in the usual manner by the OpenGL 3D package.

The data structure is described first followed by a description of the Test-Fixture within which it is implemented.

6.2.1 Data Structure

In conventional 3D computer graphics, whether video games or computer aided design, 3D objects are typically described by their bounding two-dimensional (2D) faces. For example, a tetrahedron could be graphically defined by its four bounding triangular faces. The four bounding 2D faces comprise an approximation to a compact two-manifold[3] and the enclosed 3D volume is a *three-manifold with boundary*.[4] In like manner, a 4D object can be approximately defined by its bounding three-manifold of 3D hyper-faces[5] and would be characterized as a three-manifold[67].

Since the shape of an m-manifold with boundary can be described by its bounding (m-1)D hyper-faces, and the (m-1)D hyper-faces can be tesselated by (m-1)-simplices, then the mD object can be described by a hyper-surface of adjacent (m-1)-simplices.

The fundamental data structure element is the $\underline{m\text{-simplex}}$ or $\underline{m\text{-cell}}$, stored as a list

[3]**Compact Manifold** - It should be noted that the term "compact manifold" often implies "**manifold without boundary**", which is the sense in which it is used here.[66]

[4]**Manifold with boundary** - is a manifold with an edge - i.e. a 2-manifold with boundary is homeomorphic to a disk where the bounding circle is the edge and a 4-manifold with boundary would be homeomorphic to a 4D ball where the 4D sphere would be the edge.

[5]**hyper-** when prefixed to a geometrical term, refers to an extrapolation from the common geometrical object into a generic dimensionality. For example an infinite hyperplane would indicate a co-dimension one isometric object of infinite extent. A hypercube in the context of four-space would indicate a 4D cube, while in a 5-space context would indicate a 5D cube, yet in a 2-space would indicate a square.

Pseudo-Code of Sample Data Structures

```
struct Vec7 {                          // 7D double precision vector
    double      t,x,t,z,w,v,u;         // 7 components of vector
};

struct Object {                        // Linked list and OpenGL color info
    OBJ_TYPE objType;                  // Triangle, Tetrehedra
    Material    material;              // OpenGL material color/lighting info
    Object      *next;                 // Link to next singly linked-list object
};

struct Triangle : public Object {      // 2D Triangle Object of 3 vertices
  vecArray *Verts;
  int iA, iB, iC;                      // Indices of 3 vertices into *iVectors array
};

struct Tetrahedron : public Object {   // 3D Tetrahedron Object of 4vertices
  vecArray   *Verts;
  int        iVert[4];                 // Indices of 4 vertices into *iVectors array
};

struct vecArray {
  int        vecLen;                   // Number of used entries
  int        maxLen;                   // Allocaed array size
  int        *fixed;                   // flags and counts for sums & average
  Vec7       *vecPtr;                  // The 7D vectors
  double     fClose;                   // Used to merge shared fClose vertices
};
```

Figure 6.1: Structures used to slice 4D tetrahedra into displayable 3D triangles. The Tetrahedron is sliced into one or two Triangles. The Object structure contains the lighting model data required by OpenGL. One 7D iVectors vertex array is shared by all 4D objects and a second is shared by all the sliced 3D displayable objects. Vertices of adjacent nD simplices are identical, so most vertices are shared.

of (m+1) vertices of n components in Euclidean n-space (E^n), where $n \geq m$. A list of all the unique nD vertices is maintained as an indexed array (vecArray in Figure 6.1) in order to minimize vertex transforms and memory manipulation. This is discussed in Section 6.2.2 on page 72.

Data Complexity

The fundamental element of this data structure is the simplex which contains (m+1) vertices. A vertex in n-space requires n components. A connected closed compact m-manifold (pure simplicial m-complex) built of S bounding simplices in n-space will require storage of $S * n * (m + 1)$ components. The storage requirement of the data structure for an mD object in n-space bounded by S simplices can be restated as $\mathcal{O}(S * n * m)$ floating-point components or $\mathcal{O}(S * m)$ vertices.

6.2.2 Test-Fixture

The Test-Fixture consists of two independent modules: the *Object-Generator* constructs an mD object in n-space by defining its compact (m-1)D boundary from (m-1)-simplices; and the multi-point-of-view *nD Object-Viewer* which constructs three-dimensional viewable objects from the (m-1)-manifold and simultaneously displays the object from multiple viewpoints and sub-dimensions in multiple view-ports via an OpenGL[6] based 3D viewer.

The two classes of m-manifolds with boundary demonstrated here were selected for their interesting four-dimensional symmetries and asymmetries, as well as their simple and recognizable 3D intersection properties.

[6]**OpenGL** - The Open Graphics Library, at the time of this writing, is an interoperable standard API and library for writing applications that produce 2D and 3D computer graphics.

Object Generator

The mD objects (m-manifolds with boundary) are constructed out of simplices[7] in the *Object-Generator* module from a mathematical description of the hyper-surface of the m-manifold's boundary. As used here, a hyper-surface is a pure simplicial complex formed by joining contiguous (m-1)-simplices at their shared (m-2)-faces into a closed (m-1)-surface[8] in the same way that a 3-manifold with boundary such as a tetrahedron is created from four mutually adjacent 2D triangles enclosing the tetrahedron's inner 3-space. A "closed-up" pure simplicial (m-1)-complex may be considered connected, compact and without boundary since all (m-2) edges are shared with adjacent simplices.

Conceptually, the m-manifold is generated by iterating though the 4D (or 5D) hyperspherical[9] coordinates around the mD object's center, and embedding the contiguous bounding (m-1)-simplices onto the mD hypersurface. The generation algorithm iterates around and through the hypersphere's mD hyper-surface via mD hyperspherical coordinates which are converted to rectilinear coordinates during embedding.

The m-manifold's vertices are generated via parametric functions of hyper-spherical coordinates and converted to E^4 or E^5 Euclidean coordinates. The resultant vertices of the pure simplicial m-complex so generated are then insertion sorted via a *red-black balanced binary search tree* algorithm into the *vecArray*, which is an indexed list of unique 7D vertices. The *Object* linked-list of simplices is constructed of either *Triangle* data elements with three indices or *Tetrahedron* data elements

[7] **Simplex creation** is described in detail in Section 2.3.2 and in sections A and B of [2]

[8] **Closed surface** - "A surface is closed if it is compact, connected and has no boundary; in other words it is a compact, connected Hausdorff space in which each point has a neighborhood homeomorphic to the plane."[68]. This definition will be used, with suitable extension, to describe the hyper-surface.

[9] **Hyperspherical** - A coordinate system in an n-dimensional Euclidean space which is analogous to the spherical coordinate system defined for 3-dimensional Euclidean space, in which the coordinates consist of a radial coordinate r, and (n-1) angular coordinates.

with four indices as documented in Figure 6.1. *Pentachoron* data elements with five indices were added for 5D boundaries. The m-manifolds are not limited to convex shapes and need not be simply-connected as demonstrated by the hyper-torus.

Storage and Computational Complexity The number of simplices S for a specific simplex resolution R increases exponentially with the dimensionality k of the k-manifold. In the case of an k-cube for example, this would be $S = \mathcal{O}(R^k)$ where R is the number of 1D simplices in the cube's 1D edge. Consider the bounding m-manifold surface of the k-manifold where m=k-1. For the bounding m-manifold of the k-cube, the number of simplices is one dimension lower, and so $S_m = \mathcal{O}((S_k)^{\frac{k-1}{k}})$ or $S = \mathcal{O}(R^m)$.

The S simplices are embedded serially in n-space by the object generator resulting in storage complexity $\mathcal{O}(m * S * n)$. The storage is reduced to $\mathcal{O}(S * n)$ at the computational cost $\mathcal{O}(m * S * n * log(m * S * n))$ by using a balanced binary search tree (BBST) to build and merge vertices that are identical within some ϵ into an indexed list. In these calculations, the embedding space n is also the number of vertex components. The generation and compression procedure needs to occur only once at object creation time, and will further optimize the future intersection and display processing by reducing computational complexity from $\mathcal{O}(m * S * n)$ to $\mathcal{O}(S * n)$ or to $\mathcal{O}(S)$ per view for constant n.

The Hypersphere: 3D two-sphere and 4D three-sphere The first class of nD object selected is the m-sphere[10] where m=n-1, defined as:

$$\mathcal{S}^m = \{\overline{x} \in \mathcal{R}^n : \|\overline{x}\| = 1\}. \tag{6.3}$$

Just as a 3D two-sphere $\mathcal{S}^2$ is described by its bounding two-dimensional surface, so

[10]**Sphere vs. Ball** - as used here, a sphere is hollow while a ball is solid, regardless of the number of dimensions. A sphere is the ball's bounding hyper-surface.

is a 4D three-sphere $\mathcal{S}^3$ described by its bounding three-dimensional hyper-surface. The closed hyper-surface boundary of a 3-sphere in 4-space can be modeled as described by the pseudo-code segment in Figure 6.11.

The Hypertorus: 3D two-torus and 4D three-torus A second class of n-manifold with boundary demonstrating interesting visual symmetries and asymmetries is the m-torus where $m = n - 1$, defined as:

$$\mathcal{T}^m = \underbrace{\mathcal{S}^1 \times \mathcal{S}^1 \times \ldots \times \mathcal{S}^1}_{m\ times} \subseteq \underbrace{\mathcal{R}^2 \times \mathcal{R}^2 \times \ldots \times \mathcal{R}^2}_{m\ times} \tag{6.4}$$

As shown by (6.3), the one-sphere can use two dimensions for its embedding:

$$\mathcal{S}^1 \subseteq \mathcal{R}^2 \tag{6.5}$$

Ergo, by (6.4), a two-torus $\mathcal{T}^2$ can span four dimensions while a three-torus $\mathcal{T}^3$ can span six dimensions.

$$\mathcal{T}^m \subseteq \mathcal{R}^{2m} \tag{6.6}$$

Two-Torus If the each of the two S^1's are embedded in the XY and the ZW planes, then merging these two 2-spaces into a 4-space will create the rather uninteresting 4D structure as shown in Figure 6.2 when intersected with the 3-flat. A simple oblique orthogonal projection of the $\mathcal{T}^2$ in $\mathcal{R}^4$ onto $\mathcal{R}^3$, as shown in wire-frame in Figure 6.3, can yield a self-intersecting 3D representation of the two-torus. In the FRONT view, the wire-frame reveals this is an unphysical object since the surfaces interpenetrate near the dimple at the top and bottom of the

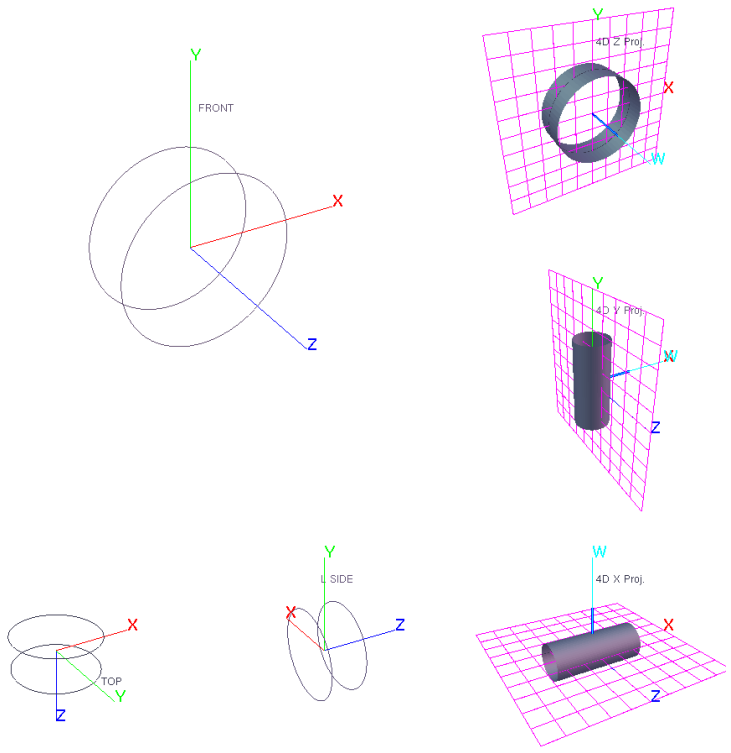


Figure 6.2: Two-torus in 4D and sliced orthogonal to W by a three-flat
Accompanying video filename figure-6.2.mov. Video ID: TorusI_06m04s33r664t0_hTorus_twist-S

three-manifold.

The familiar donut shape of a $\mathcal{T}^2$ torus is the result of mapping of one component of the four-space $\mathcal{R}^4$ torus into the three-space $\mathcal{R}^3$ of the observer. In Figure 6.4, for example, the W component of the (W, X, Y, Z) $\mathcal{R}^4$ Euclidean four-space is mapped onto the X & Y components of the (X, Y, Z) $\mathcal{R}^3$ Euclidean three-space as described by the pseudo-code in Figure 6.12.

Three-Torus The four-dimensional torus or three-torus $\mathcal{T}^3$ embedded in four-space was selected to demonstrate the capabilities of the Test-Fixture since certain three-flat cross sections reveal three-dimensional two-tori embedded within the object as seen in Figure 6.9. As with the three-sphere, the three-torus is defined by its bounding three-surface.

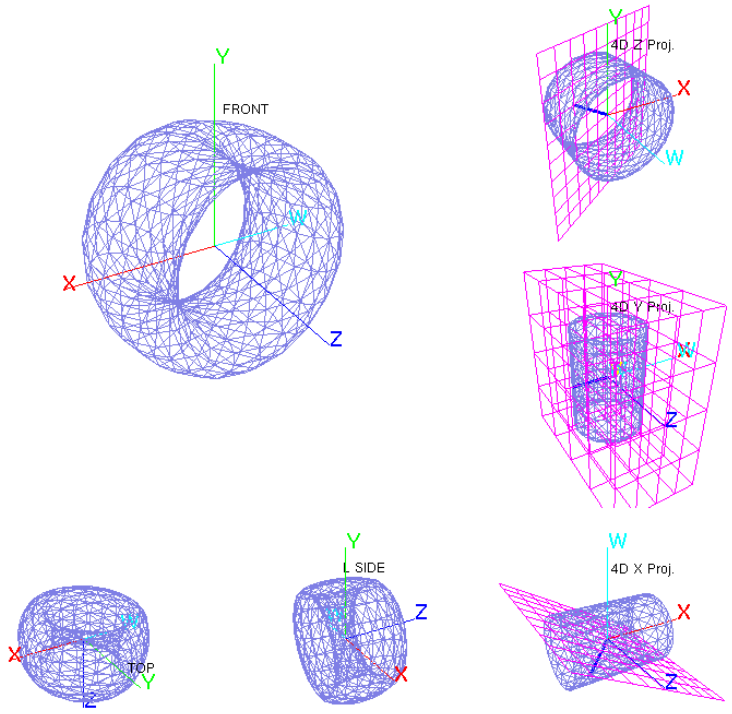


Figure 6.3: Two-torus in 4D and projected obliquely onto a three-flat
Accompanying video filename figure-6.3.mov. Video ID: TorusI_06m04s13r664t0_hTorus_twist-R

Similar to the two-torus, the components of a $\mathcal{T}^3$ can be mapped to embed the $\mathcal{T}^3$ from $\mathcal{R}^6$ (Equation. 6.6) onto $\mathcal{R}^4$. For example, labeling the components of $\mathcal{R}^6$ as (X, Y, Z, W, V, U), the components V and a piece of Z are mapped into X and Y, while U and the remainder of Z are mapped into Z, as shown by the pseudo-code in Figure 6.13.

The astute programmer can detect the code for the two-torus enmeshed in the three-torus pseudo-code shown here.

Object Viewer

The Viewer reduces the n-dimensional object to three-dimensions for viewing with the interoperable OpenGL 3D computer graphics library. The OpenGL library can

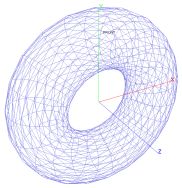


Figure 6.4: Two-torus embedded in three-space

then provide the ubiquitous three-dimensional viewer functions such as rotate, pan, zoom, and shade.

Intersection The Transversal Intersection of an (n-1)D hyperplane in n-space with an (n-1)D simplex can yield an (n-2)D simplex. The intersection of this same (n-1)D hyperplane in n-space with a closed-up pure simplicial (n-1)-complex can yield a closed-up pure simplicial (n-2)-complex. The intersection of two transversal submanifolds of Y is again a submanifold.[69]

$$codim(X \cap Z) = codim(X) + codim(Z). \tag{6.7}$$

In this manner an (n-1)-manifold with boundary can be created from the intersection of an (n-1)-hyperplane with an nD manifold with boundary. In this sample implementation as shown in Figure 6.6, a 3D isometric hyperplane of infinite extent (a three-flat) when intersected with a three-manifold (a closed compact three-manifold) in four-space will generate a closed compact two-manifold that bounds a three-manifold with boundary. Likewise, a 3-flat in 4-space intersecting a 1D line can yield a 0D point.

As explained above in Section 6.2.2, the 4D object is described by its bounding 3D manifold of tetrahedra. If the three-flat of infinite extent is constrained to penetrate the convex hull of the four-space of the bounding pure simplicial three-complex,

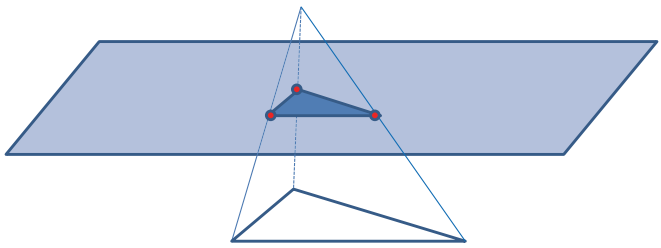


Figure 6.5: Constructing a Triangle from Intersected Tetrahedral Edges.

then an intersection with at least one tetrahedron must occur.

If two of the non-traversal intersection degenerate cases[11] are ignored, a 3D by three-simplex intersection in four-space will result in the generation of one of three objects:

1. **no object** if no intersection occurs;
2. a **plane** if the three-flat transversally intersects the tetrahedron as in Figure 6.5;
3. the original **tetrahedron** if the three-flat and tetrahedron share the same three dimensions - i.e. a non-transversal intersection which is degenerate.

To visualize the structure of the 3-manifold in 4-space as a sequence of 2D surfaces, the solution must intersect a 3-flat with this 3-manifold and yield a 2D surface. By moving the 3-flat, different slices of this 3-manifold are selected. Since this 3-manifold is a pure simplicial 3-complex, it consists of linear simplices. Using this information the intersection operation can be simplified to performing the intersection on the edges of each 3-simplex with the 3-flat, and then connecting these intersection points according to the connectivity of their parent simplex to get a 2D mesh as depicted in Figure 6.6. The resultant 2D mesh comprises the pure

[11] **Two degenerate cases:** A non-transversal intersection of a 3-flat with a vertex will yield a 0D point, while a non-transversal intersection of a 3-flat with an edge will yield a 1D line. These intersections are unstable or degenerate.

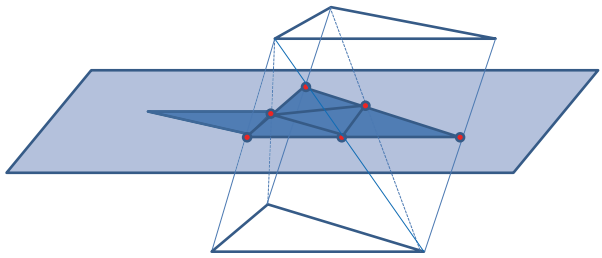


Figure 6.6: Plucking a Triangular Mesh from a Tetrahedral Mesh.

simplicial 2-complex which will be rendered. A pseudo-code description of this process is given in Figure 6.14.

Parallel Performance The block diagram in Figure 6.7 identifies in green those paths appropriate to parallel processing. Since each of the simplex edge intersection operations are independent, it is possible to implement this algorithm with up to t threads where $1 \leq t \leq s$ (up to one thread per simplex) to yield a nearly linear performance improvement of $O(n * s/t)$, assuming trivial per-polygon graphics engine display times as with contemporary graphics engines. An alternative parallel approach for a machine with fewer threads would be to implement one thread for each of the six tetrahedra edge intersection operations. For the latter case the operations are likewise independent but share the same 3-flat plane equation and same vertex list.

Projection An (n-1)-dimensional "shadow" or "silhouette" of the n-manifold can be created from the nD object by projecting the object along an arbitrary vector orthogonally onto an (n-1)-dimensional viewscreen. In the sample implementation here, this would be the projection of a four-dimensional object into a three-dimensional view space resulting in a new three-dimensional object from the superposition of multiple vertices, edges, and faces of the object's three-simplices. Hanson[49] suggests discarding the extra component of the four component vertex

(set W to zero, for example). The resultant superposition can cause many redundant graphic operations, but it will project the three-manifold along the W axis onto the XYZ image-hyperplane. A shadow can be projected along any arbitrary direction vector by performing a 4D rotation of the W-axis to the selected direction in n-space and discarding this W component.

Raytracing While not an OpenGL function, the algorithm's raytracing feature described in Section 2.3 is included here for completeness to demonstrate the versatility of this data structure and algorithm.
Raytracing is implemented via a barycentric algorithmic solution to the lightray-tetrahedron intersection.[2] The barycentric algorithm solves for the intersection of a line and the 4D object's hypersurface represented by the tetrahedra, and then determines the 4D spacetime position within the tetrahedron where the intersection occurs. All solutions along the ray are considered and all but the hyper-voxel nearest to the POV are discarded, resulting in a 2D image of the 4D spacetime as seen from the POV. The resultant pixel is shaded via a conventional raytracing lighting model as applied to the extruded object's surface at the point of intersection.

Visualizing Special Relativity Visualizing the geometric effects of Special Relativity can be implemented by constraining the reversed lightray to lie along a negative lightcone in Minkowski four-space.

Parallel Performance Traditional raytracing has lent itself well to parallel processing where each light-ray is processed independently by one thread. This strategy is also applicable for 4D or higher dimensional spacetime raytracing.

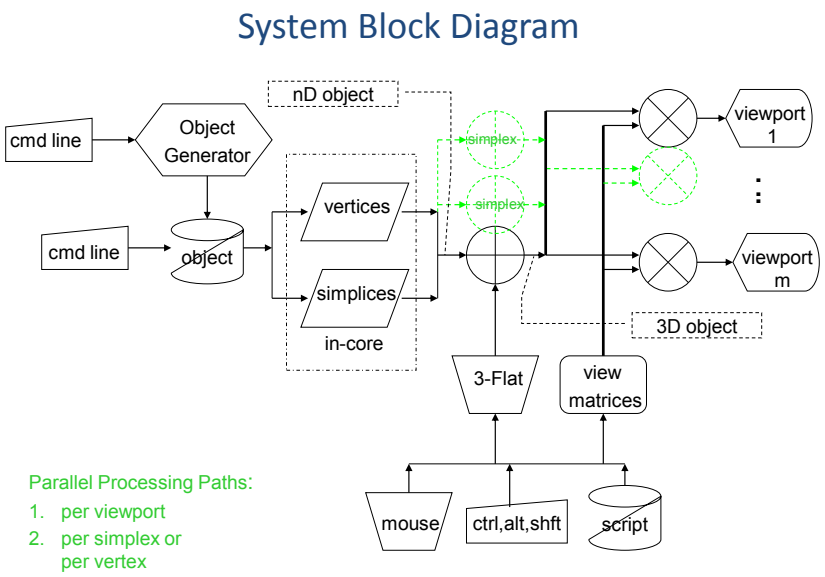


Figure 6.7: Test-Fixture System Block Diagram

6.2.3 System Block Diagram

As depicted in Figure 6.7, the Test-Fixture provides the following capabilities:

1. Simultaneous multi-Aspect multi-Point-Of-View viewports;
2. Multi-dimensional graphics user interface via mouse & keyboard;
3. Interactive slice, pan, tilt, zoom and shade viewing;
4. Batch mode animation scripting language via text file.

Green paths in the figure indicate optional parallel multi-thread implementations in future versions of the Test-Fixture.

Multi-Aspect Points-Of-View As represented by the transform ($\otimes$) and viewport ($<\supset$) symbols in the right of Figure 6.7, the Test-Fixture supports an arbitrary selection of possible points-of-view, selection of dimensions, dimension reduction techniques, and display modes. Sample display screens in figures 6.2 and 6.3 show six simultaneous views of a two-torus in four-space in a Computer Aided

Design (CAD) type orthogonal viewport format. The right column of three viewports are dimensionally reduced via projection of 4D to 3D and represents, from top to bottom:

1. the (W, X, Y) components of the four-space projected along the Z axis;
2. the (W, Y, Z) components of the four-space projected along the X axis;
3. the (W, X, Z) components of the four-space projected along the Y axis.

This projection strategy provides a complete view of all four dimensions of the 4D object via a combination of its various 3D shadows or silhouettes. simultaneously. The left three views are conventional *Front, Top, Side* CAD views of the three-dimensional object that is created by the three-flat intersection with a three-manifold (four-manifold with boundary). Conceptually, the top-left viewport is a picture of the 3D world within the 3D hyperplane. The four labeled axes in the left 3D views map to the corresponding (W,X,Y,Z) axes in the four-space. The relative lengths and angles of the labeled axes indicate the spatial contributions to the extent of the 3D object created by the three-flat to three-manifold intersection. If there is no fourth axis label visible, the 3D view is orthogonal to the invisible 4D axis.

Figure 6.10 offers an alternative CAD format for the display of a four-manifold in five-space and is described in Section 6.3.3.

Extra-Dimensional Graphics User Interface The intersecting three-flat's position and attitude in four-space is indicated by the red grid icon shown in the three viewports of the right column in figures 6.2 and 6.3, and in the right nine viewports of Figure 6.10. As depicted in Figure 5.1, the user can rotate the three-flat by holding down the SHIFT-Key and LEFT-Mouse-Button while dragging the icon with the mouse. The three-flat can be translated by use of the mouse on the icon with the SHIFT-Key and the RIGHT-Mouse-Button.

Interactive viewing The 4D algorithm as implemented is interactive on contemporary desk-top computers[12]. The desk-top user can observe the leftmost three viewports containing the resultant three-dimensional object metamorphose as the position and attitude of the three-flat intersector of the three-manifold is repositioned interactively with the mouse and keyboard. Note that interactivity seriously degrades when 5D objects are sliced. A parallel implementation should address this degradation.

Animation scripting language Experience with this GUI has shown that while interactivity is effective for discovering expected and unexpected geometries, a scripted animation is superior for smooth demonstrations. Hence a scripting language has been implemented that provides consistent incremental translations and rotations of the intersecting hyperplane, and provides output of the resulting three-dimensional intersection display to sequentially numbered files. These files are then converted by the Linux *ffmpeg* animation package to create a *QuickTime* (.mov) video of the demonstration sequence.
The animation feature was used to generate the sequences of frames shown in the figures in Section 6.3. Animation sequences are available online via the Internet[70, 60, 61].

6.3 Examples of Exploring Four-Manifolds with Boundary

For brevity only a hyper-sphere and a hyper-torus are displayed. Frames of an animation of these two closed compact three-manifolds in four-space as output by the Test-Fixture are shown and described below.

[12]The author's **contemporary desk-top computer** is a dual-core 2.6GHz AMDx2 w/ 4GB under XP-64. The application was implemented as a single thread.

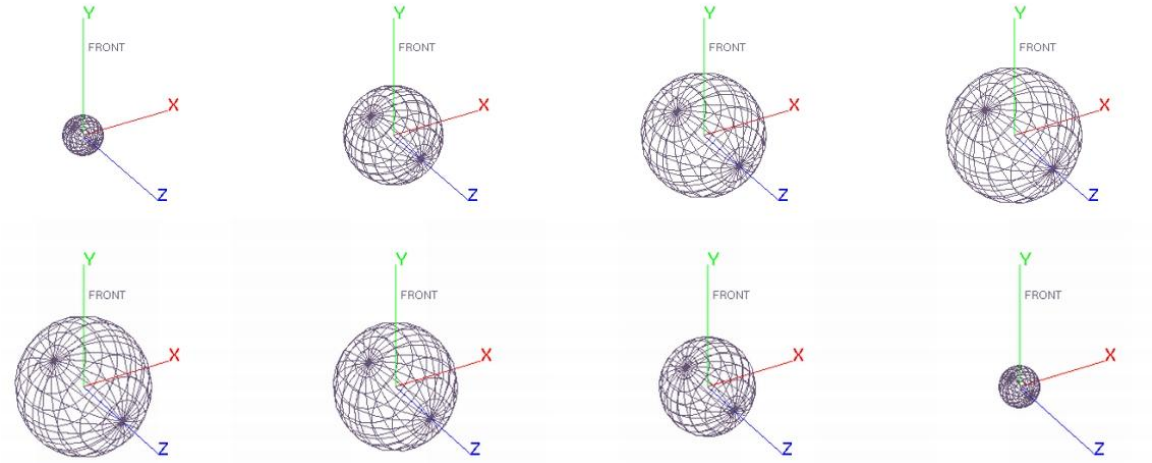


Figure 6.8: A 4D Sphere sliced eight times along the W axis by a 3D hyperplane. The sliced $\mathcal{S}^2$'s diameter changes with the $\mathcal{S}^3$'s diameter
Accompanying video filename figure-6.8.mov. Video ID: hSphereI_04s11r633t0_hSph02-633_sliceW

6.3.1 Slicing a Three-sphere

Shown in Figure 6.8 is a three-sphere of radius 10.0 world-units that has been sliced by a three-flat orthogonal to the W-axis as one would slice a loaf of bread with a knife. The knife in this case is a three-flat, an intersecting three-dimensional hyperplane of infinite extent, stepped along W from +9.6w to -9.6w in eight equal steps yielding eight three-dimensional plucked slices of a hyper-sphere. The slices have no W component, thus as can be seen in Figure 6.8, each slice is a 3D sphere of common experience. But to carry the analogy further, the result of this slicing is not a slice of bread, but rather the infinitesimal (n-1)D space between the loaf's slices. The plucked sphere can be exported as shown in Figure 7.6.

6.3.2 Slicing a Three-torus

Figure 6.9 displays a sequence of three-dimensional plucked slices of a three-torus. As the three-flat slicer progresses along the W axis from +9.6w to -9.6w in eight equal steps, the ellipsoid end-cap metamorphoses into a sequence of an oblate genus one torus whose symmetric axis is collinear with the W-axis, then into a genus two

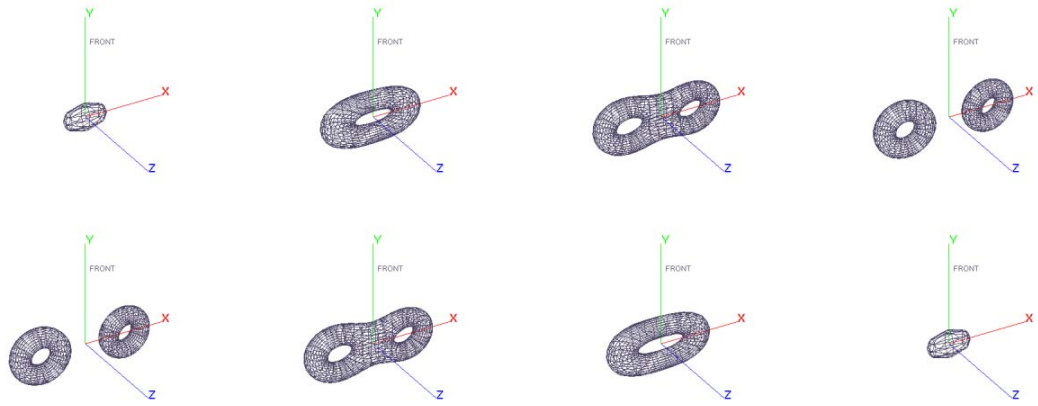


Figure 6.9: A 4D Torus sliced eight times along the W axis by a 3D hyperplane $\mathcal{S}^1 \times \mathcal{S}^1 \times \mathcal{S}^1 \rightarrow \mathcal{R}^6$ is mapped onto $\mathcal{R}^4$ by pseudo-code in Figure 6.13
Accompanying video filename figure-6.9.mov. Video ID: hTorusI_36m05s13r521t0_hTor05I_cutWXYZr521p

torus, then into a pair of tori symmetric about the XZ plane. The process then reverses itself through the same sequence back to ellipsoid end-caps. A genus zero 3D object can thus metamorphose though a genus one configuration into a genus two then into two genus one 3D objects, and then back again. The plucked objects can be exported as shown in Figure 7.10.

6.3.3 Slicing A Five-dimensional Object

The data-structure and algorithm are designed such that they can be extended to allow visualization of objects of extra spatial or temporal dimensionality. While only two four-dimensional objects were demonstrated here, the algorithm can display five-dimensional or (4+1)-dimensional objects as shown in Figure 6.10. Shown here is a 4D closed compact three-manifold homeomorphic to a three-sphere that has been extruded along the t-axis to yield a three-manifold with boundary (not closed-up) spanning five dimensions.

The ubiquitous three-sphere is depicted in a (4+1) Minkowski spacetime with a zero

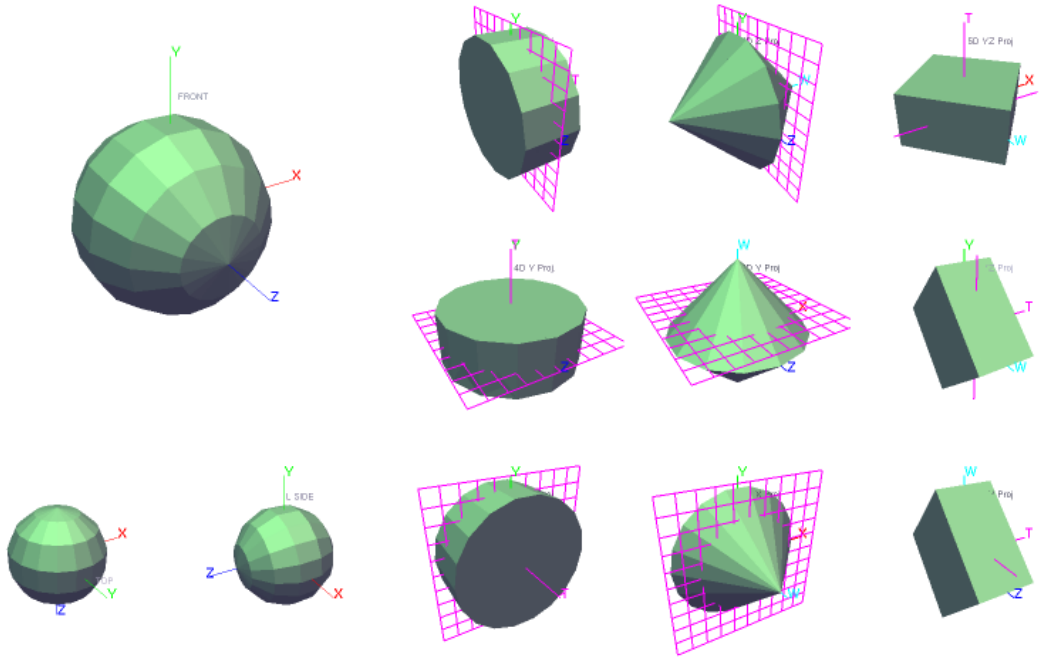


Figure 6.10: A 5D Torus Shown in 15 3D Viewports
Col 1: 3D views clockwise: (Front,Side,Top). The remaining views are projections along axis pairs into 3-space as follows: Col 2:(WX, WY, WZ). Col 3:(TX, TY, TZ). Col 4: (YZ, XZ, XY). The views are described in Section 6.3.3
Accompanying video filename figure-6.10.mov. Video ID:
hSphere7_24sa090702r9t-7_V7.7_hSphereI_CutWXYZr9.65p_n12-L-null-edit

velocity.[13] The object was extended by extruding each of the bounding tetrahedron's faces into the T dimension and tessellating each face into three 3D tetrahedra for each triangular face of each tetrahedron. The result is a 5D object with boundary constructed of a 3D tetrahedral mesh that may be characterized as a hyper-wireframe.

Note the rightmost column of Figure 6.10 where the TWX, TWY, and TWZ aspects are plotted, top to bottom respectively. In this column the XYZ 3-flat, being a codimension-two structure in $TWXYZ$ 5-space, is depicted as a line, which is a codimension-two structure in 3-space. This orientation of the 3-flat has no extension in the T or W dimensions and so is depicted as a one 1D line in the X, Y,

[13] A **relativistic velocity** would require a Lorentz Transform resulting in length contraction and time dilation which is beyond the scope of this phase of the research.

and Z dimensions, top to bottom, respectively.

Note the second column from the right in Figure 6.10 where the WYZ, WXZ, and WXY aspects are plotted, top to bottom, respectively. In this column, the XYZ 3-flat, shares two dimensions with each of the aspects and so is depicted as a gridded plane. The top through bottom aspects share the YZ, XZ and XY planes, respectively. This column is comprehensive W without T.

The second column from the left in Figure 6.10 depicts TYZ, TXZ, and TWY aspects, top to bottom, respectively. In this column, the XYZ 3-flat, shares two dimensions with each of the aspects and so is again depicted as a gridded plane. The top through bottom aspects share the YZ, XZ and XY planes, respectively. This column is comprehensive T without W.

The leftmost column of Figure 6.10 depicts anti-clockwise from the top, conventional *Computer Aided Design* (CAD) *Front*, *Top*, and *Side* projected views of the plucked 3D object (two-manifold) resulting from the 3-flat intersection. The intersection of the 3-flat with the 4D hyper-wireframe results in a two-manifold as given here:

$$\begin{aligned} codim(4D \cap 3D) &= codim(4D) + codim(3D) \\ (4D \cap 3D) &= 5 - ((5-4) + (5-3)) \\ (4D \cap 3D) &= two\text{-}manifold \end{aligned} \tag{6.8}$$

The resultant depiction is the convex-hull of the plucked object projected into 3-space.

6.4 Observations

A complex three-manifold in Euclidean four-space can be interactively explored in real-time on a personal computer by visualizing sequential three-dimensional slices of the four-dimensional model. While only two four-dimensional objects were demonstrated here, the algorithm and implementation support higher dimensional objects as is discussed in Section 6.3.3 and shown in Figure 6.10. The data-structure and algorithm are designed such that they can be extended to allow visualization of objects of yet greater spatial dimensionality. For an nD object with s simplices, an (n+1)D object at the same simplex resolution[14] will require $s^{\frac{n}{n-1}}$ simplices[15]. However, only one vertex is required for each additional simplex. The 5D example suggests that twice the number of viewports are required for each additional dimension displayed, depending on the art of the developer. The nature of the algorithm will allow for a parallel implementation on contemporary multi-GPU (Graphics Processing Unit) devices.

As suggested by Figure 6.10, the algorithm lends itself to the creation and exploration of an interactive Minkowski spacetime diagram in four dimensions or more spatial or temporal dimensions.

[14] **Simplex Resolution** is a count of the number of simples required to describe one dimension of the closed compact connected manifold without boundary.

[15] An (nD object requires an (n-1)-manifold, hence an n-manifold is required for an (n+1)D object. Simplex requirements increase exponentially

Pseudo-Code

$initialize\ prevX, \delta$
$for\ \ \alpha\ =\ 0\ to\ 2\pi\ step\ \delta\ do$
  $for\ \ \beta = 0\ to\ \pi\ step\ \delta\ do$
    $for\ \ \gamma = 0\ to\ \pi\ step\ \delta\ do$
      $X = radius * \sin(\gamma) * \sin(\beta) * \sin(\alpha)$
      $Y = radius * \sin(\gamma) * \cos(\beta) * \sin(\alpha)$
      $Z = radius * \sin(\gamma) * \cos(\alpha)$
      $W = radius * \cos(\gamma)$
      $nextX = Vec4(X, Y, Z, W)$
      $makeCubeBetweenTwo4DCoordinates(prevX, nextX)$
      $tessellateCubeTo6Tetrahedra^{\ddagger}(prevX, nextX)$
      $prevX = nextX$

Figure 6.11: Generate a three-sphere bounding surface in four-space
tessellateCubeTo6Tetrahedra‡ - a description of the tessellation of a cube into six tetrahedra is shown in [2] by Figure 2 and accompanying text.

Pseudo-Code

$initialize\ prevX, \delta$
$for\ \ \alpha\ =\ 0\ to\ 2\pi\ step\ \delta\ do$
  $for\ \ \beta = 0\ to\ 2\pi\ step\ \delta\ do$
    $X = radius * \sin(\alpha)$
    $Y = radius * \cos(\alpha)$
    $Z = tube \quad * \sin(\beta)$
    $W = tube \quad * \cos(\beta))$
    $nextX = Vec4(X + W * \sin(\alpha), Y + W * \cos(\alpha), Z, 0)$
    $makeSquareBetweenTwo3DCoordinates(prevX, nextX)$
    $tessellateSquareTo2Triangles(prevX, nextX)$
    $prevX = nextX$

Figure 6.12: Generate a two-torus bounding surface in three-space

Pseudo-Code

initialize $prevX, \delta$
for $\alpha = 0$ *to* 2π *step* δ *do*
  for $\beta = 0$ *to* 2π *step* δ *do*
    for $\gamma = 0$ *to* 2π *step* δ *do*

$$X = radius * \cos(\gamma); \qquad Y = radius * \sin(\gamma); \quad // \ \mathcal{S}_0^1$$
$$Z = depth * \cos(\alpha); \qquad W = depth * \sin(\alpha); \quad // \ \mathcal{S}_1^1$$
$$V = tube * \cos(\beta); \qquad U = tube * \sin(\beta); \quad // \ \mathcal{S}_2^1$$
$$x_1 = X + (V + Z * \cos(\beta)) * \cos(\gamma)$$
$$x_2 = Y + (V + Z * \cos(\beta)) * \sin(\gamma)$$
$$x_3 = (U + Z * \sin(\beta))$$
$$x_4 = W$$

$nextX = Vec4(x_1, x_2, x_3, x_4)$
$makeCubeBetweenTwo4DCoordinates(prevX, nextX)$
$tessellateCubeTo6Tetrahedra(prevX, nextX)$
$prevX = nextX$

Figure 6.13: Generate a three-torus bounding surface in four-space
Two different three-torus shapes can be generated by swapping the x_3 and x_4 components.

Pseudo-Code

Perform interactive GUI directed 3-flat 4D matrix transforms

initialize OpenGL 3D display list
for each Tetra in 4D Object do
  for each edge in Tetra do
    intersect edge with 3-flat
  build a polygon from resultant vertices
  tesselate polygon into 3D triangle(s)
  append 3D triangle to OpenGL 3D display list

for each 3D triangle in new object's 3D display list do
  perform interactive GUI directed viewing transforms

Figure 6.14: Reveal three-dimensional slice in four-dimensional model
Intersect 4D Model with 3D hyperplane.

Chapter 7

Results and Evaluation

> Delays of longer than a few seconds [by the system] can result in the disruption of thought and short term memory for the next planned action(s).
>
> - C. Marlin Brown [71]

As specified in the opening Section of this dissertation, the objective is to obtain a clear comprehensive and representative view of a higher dimensional object, specifically of a closed compact 3-manifold bounding a 4D object. The three operations of 1) intersection, 2) 4D to 3D dimensional reduction, and 3) the multiple POV's, must be performed in less than a few seconds in order to be interactive, or at a rate of better than 0.25 frames per second.

7.1 Overview

7.1.1 Stage One - Raytracing Spacetime

The first stage explored the visualization of a known physical four dimensional phenomena, the apparent effects on the visualization of the geometry of an object moving at a relativistic velocity with respect to the observer: Special Relativistic Visualization. This experiment, described in Section 2.3, showed that certain

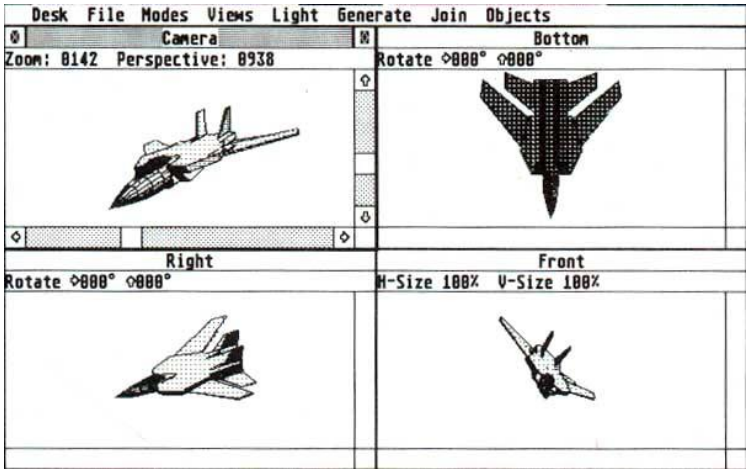


Figure 7.1: Example Multiple Points-of-View on 3D CAD Workstation

technologies developed specifically for the implementation were both effective and delivered accurate visualizations as predicted by Special Relativistic theory. These technologies included: a data structure to approximate the geometric shape of a 3-manifold in 4-space extruded from a 2-manifold in 3-space describing a 3D model of a 3D object with a velocity with respect to the camera; a Barycentric algorithm crafted to retrieve the color for a pixel on a 2D image plane from within the object's 4D spacetime 3-manifold. From this effort it was determined that animation, along with providing a conventional mirror to reflect an additional view of the object back to the camera as shown in Figure 2.9, were effective visualization strategies, and were indispensable to view clearly unambiguous geometrical information about the model. In addition it was found that introducing additional viewports displaying alternate points-of-view of the same test case as shown in Figure 7.2 provided more, and in some cases necessary, information about the physics and geometry of the visualization.

Figure 7.2 is an example of this point. The viewports show non-relativistic, relativistic, and top-down situational views from left to right, respectively. Each

viewport is a visualization of three columns of colored dashed lines moving towards the camera on the left, stationary in the center and moving away from the camera on the right. The leftmost viewport shows a non-relativistic visualization, the center viewport visualizes the dashed lines moving relativistically, while the rightmost viewport depicts the situation from a point-of-view directly over the moving dashed lines without any relativistic or delayed signal effects.

The raytracing algorithm's performance precluded interactivity, but the method was necessary for depicting the delayed signal propagation. To address this challenge an animation capability was introduced to provide the user with additional depth cues and a sense of continuity as the camera smoothly progressed from view to view. The Test-Fixture allows the sequence of .ppm frames to be converted to .mpeg and .mov video files for later replay. Sample videos are available on the referenced website [60, 61, 70] The takeaways from this stage are:

1. The need for better performance;
2. More than one point of view; and
3. Animation.

Given that at least the last two generations of professional engineers have successfully used 3D computer graphics to obtain clear comprehensive representative views of 3D objects, views adequate to replicate and build these objects in 3-space, it does not take a leap of faith to accept that 3D CAD visualizations deliver comprehensible depictions of the models they represent.

In the research that has been described here various methods of representation have been implemented and examined for applicability to the stated goal. An analysis of the properties of CAD visualization software packages yields certain common functionality. 3D CAD programs provide multiple 2D points of view of the 3D object. An effective common format is three 2D projections along the third axis

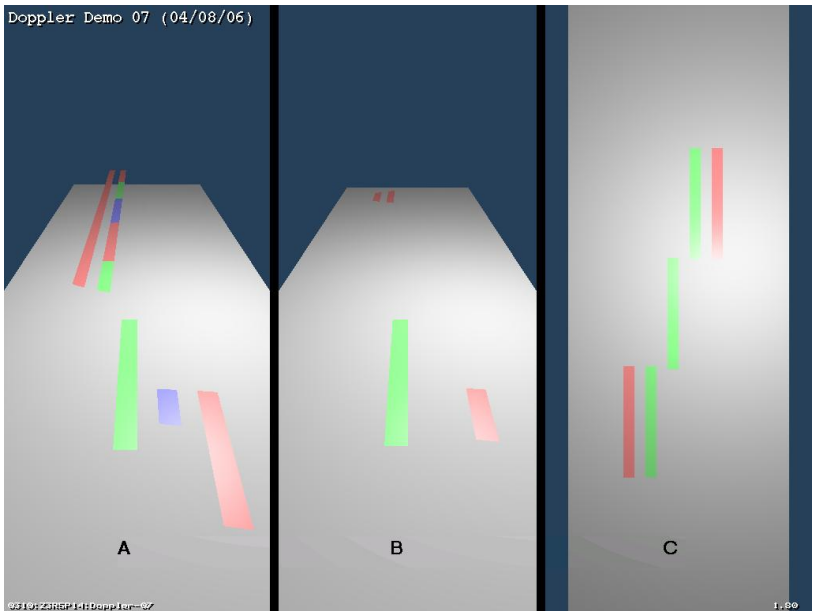


Figure 7.2: Spacetrace showing Three Viewports
A) non-relativistic, B) relativistic, and C) god's-eye views of the same spacetime event.
Accompanying video filename figure-7.2.mov. Video ID: Doppler-07a,Dash-07a.

thus yielding Front, Top, and Side views of the object resulting from projections along the Z, Y, and X axes, respectively, as in Figure 7.1. A forth viewport often provides a perspective projection as shown in the referenced figure, an isometric projection or even an arbitrary 2D cross-section of the 3D model.

7.1.2 Stage Two - nD Lattice

In the second stage both Euclidean and non-Euclidean space was visualized from multiple viewports. This experiment, described in Section 2.4, introduced the ***Aspect*** concept wherein viewports depicted 3D views of dimensionally reduced 4-space dimensions. Each of the six viewports delivered a 3D view of the 4-space, but each view delivered a different selection of three of the four possible dimensions thus reducing the dimensionality of the view. The first three viewports as in Figure 2.14 followed the CAD convention to display anti-clockwise from the top-left, the Top, Front, and Side views of the nominal XYZ 3-space, as can be seen,

respectively, from the labels on the axes. The remaining three viewports depicted three alternate Aspects, that are the other combinations of three of the four dimensions. Continuing anti-clockwise those Aspects are TXZ, TYZ, and TXY. In addition to multiple aspects, interactive performance was introduced. Each of the six viewports could be rotated, panned and zoomed independently and collectively. This greatly enhanced the ability of the user to explore this 4-space. Certain other characteristics of the non-Euclidean nature of the 4-space could be explored such as those discussed in Section 8 *Future Work*, which are beyond the scope of this dissertation.

For example, retarded time effects could be viewed interactively by propagating wavefront-like perturbations in the 4D lattice over time at a specified rate. That is to say, an arbitrary inverse distance squared relationship could be assumed between adjacent lattice cells such that the displacement of one cell affects the displacement of adjoining cells at a specified rate, similar to the propagation of waves at a finite rate in a discrete toy spacetime. The history of the source cell for the duration of the experiment must be saved to be accessible in order to compute the retarded signal's contributions to the current spacetime state. The time axis is necessary to save the 3-space's history in order to display the 3-space history as if it were a 4D spacetime. It is assumed that the 4D lattice is in the lab frame and the observer is at rest, while the source may be in motion with respect to the lab frame. Interesting non-symmetric radiation effects were discovered as the speed of causal propagation and the source acceleration were varied in the toy spacetime.

Rendering via projection was not considered due to excessive cell clutter. Trivial axis intersection views were computed by discarding all 3-space cells not in the 3-space orthogonal to the axis at the axis position defining the particular view. Interactive arbitrary intersection could be accomplished by resurrecting a Bresenham Digital Differential Analyzer (DDA) algorithm adapted from a line in

3D to a hyperplane in 4D, similar to a texture mapping algorithm.

It was learned that removing static cells, or equivalently cells whose delta was below some minimum value, cleared up the clutter as in figures 2.12 through 2.15. In addition, color encoding higher order differentials of the cell's deltas provided even more useful information as in figures 2.12 and 2.13. This latter technique allowed for exploration of acceleration and jerk, both useful when examining radiation and relativistic accelerations.

The takeaways from this stage are:

1. Interactivity is necessary and useful;
2. Multi-viewports with alternate Aspects are necessary and useful;
3. More sophisticated cell-to-cell interactions will require more memory, better data compression, superior performance, or all three.

7.1.3 Stage Three - Tangible Cross-sections of 4D Models

The third stage was implemented in four steps:

1. Model data structure definition;
2. Convex hull 3D viewer;
3. Extra dimensional Graphics User Interface design;
4. 3D Plucker.

Stage Three, Step One - 4D Model Data Structure

In step one of the third stage models of 4D objects were created by describing their bounding 3-manifolds in the same way that 3D objects are described by their bounding 2-manifolds. This implementation allows the bounding 3-manifold of the

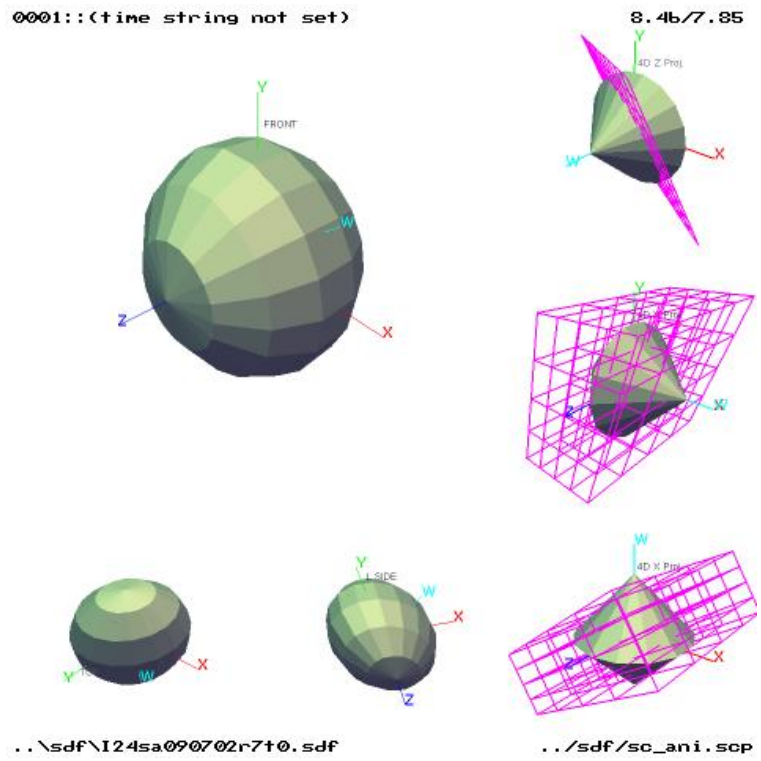


Figure 7.3: Spaceslice - 4D GUI Example

4D model to be described by a parametric function of the four spherical coordinates α,β,γ, and radius r as shown by the pseudo-code in Figure 6.11. For ease of use in describing interesting 4D objects, the implementation allows the explicit function six arguments: the three hyper-spherical angles α, β, γ; and their three associated radii r_α, r_β, and r_γ similar to that shown for the three-torus pseudo-code of Figure 6.13 where these latter three radii parameters are labeled *radius*, *depth*, and *tube*.

Stage Three, Step Two - 4D GUI

In step two of the third stage these 4D models, that is four-manifolds with boundary, were visualized and explored. As in the first two stages, multiple interactive viewports depicting different Aspects and Points-of-View of the models of the 4D objects were viewed, explored, and manipulated. A convex hull for each of

the four Aspects was generated from the projection of the 4D object along one of the four axes of the 4-space in which the 4D object was embedded: X,Y,Z, and W. Each of the resulting 3D objects could then be manipulated in 3-space by a conventional 3D GUI. In addition, the 4D object could be rotated in 4-space to provide new Aspects for each of the 3D views, as shown in Figure 7.3. Expected and unexpected symmetries were thus revealed. But the interiors of these 4D objects were still mysteries. The ED-GUI is described in Section 5.2.1.

Stage Three, Step Three - 4D Model Convex Hull Shadows

In step three of the third stage a four-dimensional graphics user interface (ED-GUI) was developed and implemented. The ED-GUI is similar to those of 3D CAD packages in that it uses the ubiquitous *click & drag* paradigm both without and with control keyboard characters to position the Point-of-View and the Slicer Icon, respectively, in each of the multiple Aspects of the 4D object. Each viewport provides both a view of the convex hull of one Aspect of the 4D object and a iconic representation of the 4D position and orientation of the 3-flat that will be used to intersect the 4D object and define a 3D view of the intersection.

Stage Three, Step Four - Plucking 3D Cross-sections from 4D Models

In step four of the third stage an intersection algorithm has been implemented to allow the user to explore the "interior" of the 4D object and "pluck" out a 3D cross-section of same. The position and orientation of the intersecting Euclidean hyperplane of infinite extent, or 3-flat, is manipulated by the user via the 3-flat icon. The resulting 3D cross-section is displayed and exported as a 3D object in the form of a closed compact 2-manifold. The ability to manipulate the cross-section as a 3D CAD object gives the user the same sense of the object's tangibility as does a 3D CAD program with any virtually engineered 3D object since it is now indeed a 3D

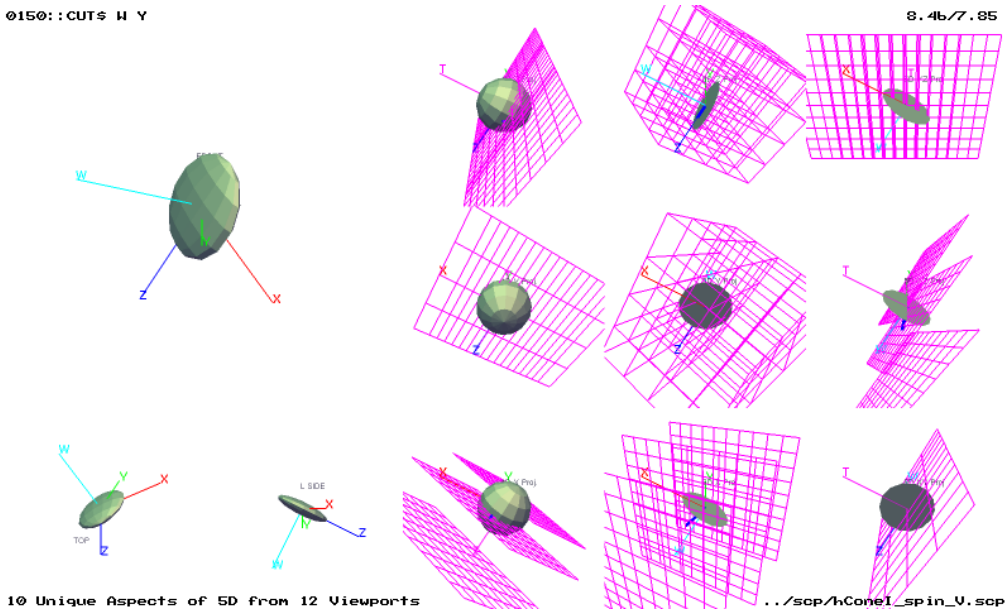


Figure 7.4: Example Multiple Points-of-View and Multiple Aspects for 4D Viewports
This is the convex hull of a 4D torus - a 3-torus

object.

7.2 Viewing 3D Aspects of nD Objects

Multiple Points-of-View of a 3D object, as in a CAD system (Figure 7.1), provide the CAD operator with a clear perspective of the 3D object under observation. The conventional rotate, pan, and zoom operations allow the operator to explore unseen portions of a convex object. Additional tools are available to the CAD operator to explore hidden areas of concave and hollow objects by operations such as slicing and transparency.

Multiple views of each of the 4D object's aspects as shown in Figure 7.4, when taken together yield a clear perception of the convex hull of the 4D object.

Viewing a 3D object on a 2D computer screen is most effective if the object is rotating, or even slightly oscillating about an axis, so that the eye can interpret any

seeming parallax as 3D depth.

7.3 The ED-GUI Makes the Intangible Tangible

As claimed on page 55 in Section 4, converting a 4D object to 3D and enabling manipulation of the 3D object gives the user a greater sense of the object's shape. As described in Section 1.2, a 3D aspect is the dimensional reduction of an nD object to 3D. As with the view of a 3D object, slight motion conveys more depth and shape information than a stationary image. Couple the 3D object's motion to conventional mouse *click & drag* so that the user sees the motion as immediate feedback, and the user has a tangible experience with the 3D object.

In like manner, the *click & drag* slicer ED-GUI can provide immediate feedback, so that the user has a tangible experience with the 4D object. The user can pluck a 3D object out of hyperspace in realtime via *click & drag*.

It was claimed that 3D slices are representative of the 4d object at the point of intersection. This is demonstrated in Figure 7.6 where the 4D hypersphere is intersected by the 3-flat at grid lines corresponding to $\sim \pm 4.0$ on both the X-axis and Y-axis. The 3D sphere thus plucked from the 4D hypersphere will have a radius of ~ 4.0.

Since the 3-sphere is symmetrical about its center with respect to all four dimensions, the expected radius of the plucked 3D object will be identical to the radius of the hypersphere minus the minimum distance from the center of the 4D object to the intersecting 3-flat within a tolerance equivalent to the 3-simplex size. See Figure 7.5.

A simpler method is to use the ED-GUI to position the intersection so as to produce an expected value, such as a radius of 4.0 as follows. Note that in Figure 7.6 the 20x20 grid lines on the 3-flat icon are on two-unit centers. Position the 3-flat in the

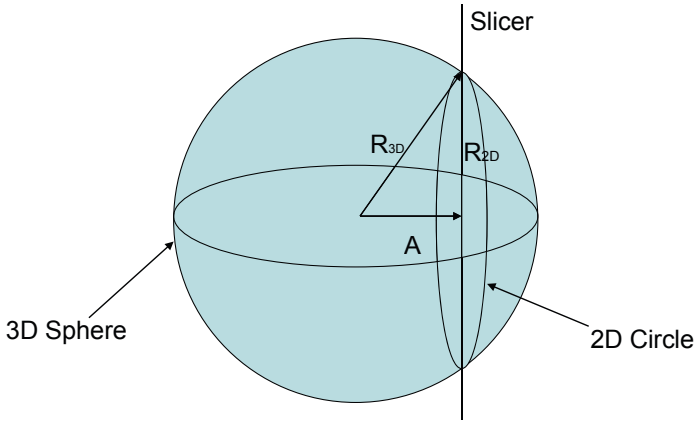


Figure 7.5: Plucking a 1-sphere (2D circle) from a 2-sphere (3D sphere)
A plane intersects a sphere of radius $radius_{3D}$ (R_{3D}) at position A along the Z axis scribing a circle of radius $radius_{2D}$ (R_{2D}).

upper-right corner viewport so as to intersect the hypersphere's surface at the X=-4, X=+4, Y=-4, Y=+4 grid lines as shown. Thus the plucked sphere's radius will be ∼4.0. This can be verified in the right-hand section of the figure where a third party PLY 3D viewer displays the minimum and maximum extents of the loaded figure. The extents, and hence the radius, of the plucked sphere can be confirmed by the numerical display in the third party PLY viewer in the right panel of Figure 7.6.

7.3.1 Evidence

The 3D slices are representative of the real sub-manifold structure of the 4D models. The measurements of the 3D object's size are evidence. The relative sizes can be measured. The display in Figure 7.6 shows minimum values of (-3.97,-3.942,-4.0) and maximum values of (+3.97,+3.942,+4.0). The average of these six radii is 3.97.

7.3.2 Proof

It can be shown that the specific slices are representative of the general case. Consider Figure 7.5 containing a 3D sphere of radius $radius_{3D}$ centered at the

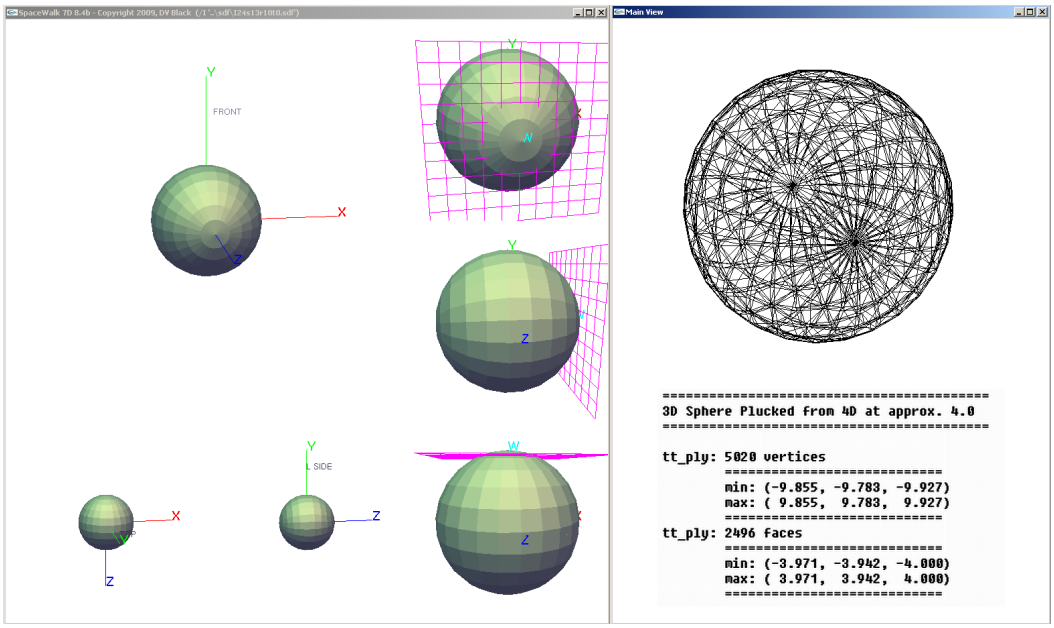


Figure 7.6: Plucking a 2-sphere (sphere) from a 3-sphere (hypersphere)
A plane intersects a 3-sphere of radius $radius_{4D}$ at position B along the W axis scribing a sphere of radius $radius_{3D}$. The radius of the 4D sphere at intersection is selected via the icon to be ~ 4.0 world-units. The 3D .ply Viewer confirmed a radius of 3.971 world-units.

origin, and an intersecting plane (a 2-flat) constructed parallel to the XY plane, which encounters the sphere at a position A along the Z axis. The intersection is a circle of radius $radius_{2D}$ as predicted in equations 7.1 and 7.2.

$$(radius_{3D})^2 = x^2 + y^2 + z^2 = A^2 + (radius_{2D})^2 \tag{7.1}$$

$$(radius_{2D})^2 = (radius_{3D})^2 - A^2 \tag{7.2}$$

Extrapolating from 3D to 4D as is our practise, it can be seen in Figure 7.6 which shows a 4D hypersphere (3-sphere) of radius $radius_{4D}$ centered at the origin, and being intersected by a hyperplane (a 3-flat) constructed parallel to the XYZ hyperplane (3-space) at a position B along the W-axis, that the intersection is a sphere of radius $radius_{3D}$ as predicted by the derivation in equations 7.3 and 7.4.

$$(radius_{4D})^2 = x^2 + y^2 + z^2 + w^2 = B^2 + (radius_{3D})^2 \tag{7.3}$$

$$(radius_{3D})^2 = (radius_{4D})^2 - B^2 \tag{7.4}$$

The numerical values in this example are $(radius_{4D})^2 = 9.850$ world-units and $B^2 = 81.18$ yielding $(radius_{3D})^2 = 14.21$, or $radius_{3D} = 3.98$. This is within 1% of the measured value shown in Figure 7.6.

7.4 Comparison of Performance Among Technologies

7.4.1 Frame Rates

The performance of the three technologies can be generalized from their frame rates. Each of the technologies output an animation as a series of sequential files, one file for each frame. The frame rate in Table 7.1 was determined from the file system time stamp on each output frame. All three software packages output at the same screen resolution. Spacetrace and the Spaceslice used the same input object file to minimize discrepancies in processing demands. The Spacegrid program does not process an object, but does process n-space itself. Each of the technologies' performance curves respond to different classes of input as listed in the column labeled *Critical value* in Table 7.1.

Spacetrace's performance degrades with the size of the image plane.

Spacetrace will require significant optimization to perform interactively. The Raytracing technology responds well to multiple threads on parallel processors since each pixel on the screen can be processed independently from all other pixels. Low resolution 3D raytracing renderers have been optimized to be

	frame rate (fps)	Displayable dimensions	Viewports available	Critical value
Spacetrace	0.00079	4	3	screen size
Spacegrid	2-9	4-5	10	cell count
Spaceslice	3-14	4-5	15	simplex count
Spaceslice (5D)	0.5	5	15	simplex count

Table 7.1: Frame Rate for Each Visualization Technology

interactive when ported to high-performance GPU based graphics cards. It is expected that such a port could provide an interactive capability for Spacetrace.

Spacegrid's performance degrades with an increase in the volume of the n-space. Spacegrid's performance would improve with an appropriate memory optimization scheme. But the compression strategy must allow for higher order differentials in the data display, so a history axis would be necessary. Merging this technology with Spaceslice would significantly reduce the storage requirements. For further discussion, see Section 8.2 in *Future Work*.

Spaceslice provides a range of adequate response times as marked by the shaded ellipses in the performance curves of figures 7.7 and 7.8. These frame rates were profiled by the Test-Fixture. Spaceslice performance does degrade with the number of simplices bounding the mD object. Spaceslice's performance could be improved significantly with a parallel thread optimization strategy, due to the parallel and independent nature of the algorithm. A thread could be devoted to each simplex, and to each edge within the simplex.

The faster and more responsive the performance, the more tangible is the experience for the user. [71] As is highlighted by the shaded ellipse in Figure 7.7 and

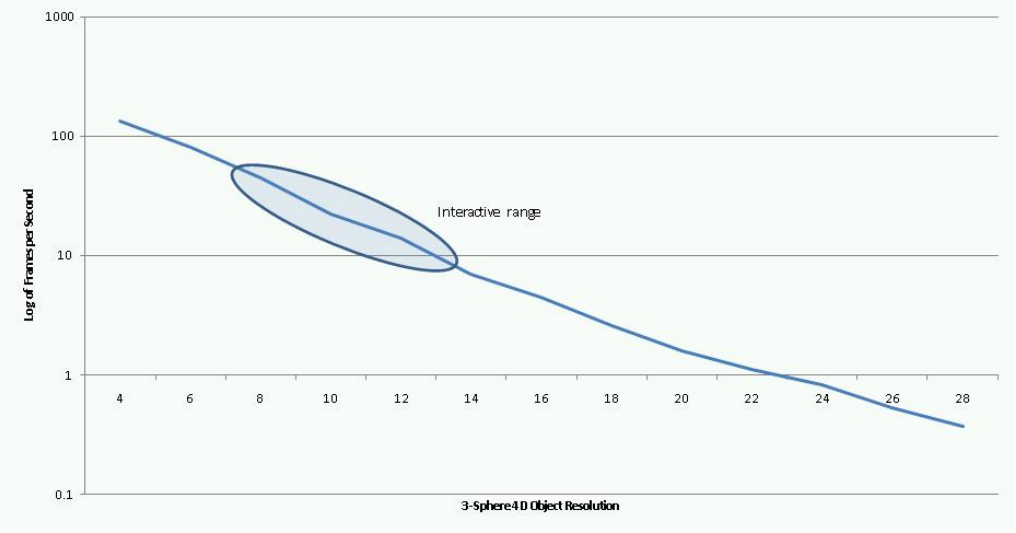


Figure 7.7: 3-Sphere complexity vs. log(frames-per-second)
Frame rates determined from internal software profiling.

in Figure 7.8 there is a range of 4D object complexity that provides both adequate object complexity to view the object's shape, and adequate response time as measured by frame-rate to also provide a tangible experience for the user. Note that the response times meet or exceed those response times specified by the US Military Standard MIL-STD-1472F in Table XXII for real-time systems as shown in Figure 7.9.

7.4.2 Dimensional Reduction

Each of the technologies provides dimensional reduction at different levels. Spacetrace only knows how to render 3D and (3+1)D objects into a 2D frame buffer. The usual reflection, refraction, and shadow capabilities of raytracers are available to enhance the depth cues for the user. The other two display packages use the Aspect paradigm to display a subset of the available dimensionality via projection or intersection. Hence, there is no theoretical limit to the number of dimensions that can be displayed given adequate screen real estate and performance

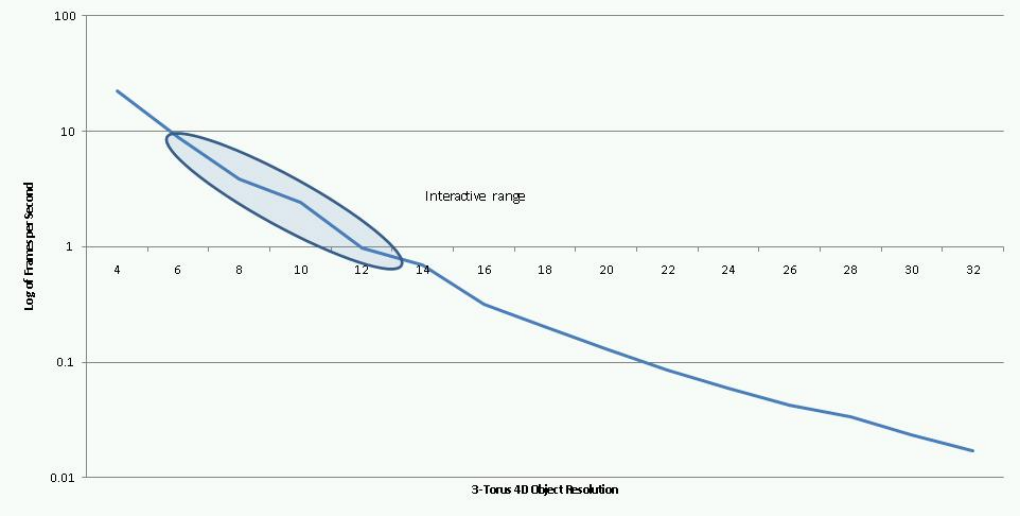


Figure 7.8: 3-Torus complexity vs. log(frames-per-second)
Frame rates determined from internal software profiling.

to support many 3D viewports.

	Viewports											
	Dimensions											
Spacetrace	1	2	3	4								
	4	4	4	4								
Spacegrid	1	2	3	4	5	6	7	8	9			
	3	3	3	4	4	4	5	5	5			
Spaceslice	1	2	3	4	5	6	7	8	9	10	11	12
	3	3	3	4	4	4	5	5	5	5	5	5

Table 7.2: Number of Viewports and Dimensions for Each Visualization Technology
Upper row is number of viewports. Lower row is corresponding number of viewable dimensions.

Spacetrace can support an unlimited number of viewports, each of which is a single point-of-view from an Inertial Reference Frame. Non-relativistic views, such as a god's-eye, are also available. But these are all the same Aspect - that is a 3D view of (3+1)D-spacetime.

Spacegrid supports up to nine Aspects (3D-views) of five dimensions in its current

Table XXII. **Maximum Acceptable System Response Times**

System Interpretation	Response Time Definition	Time (Secs)
Key Response	Key depression until positive response, e.g., "click"	0.1
Key Print	Key depression until appearance of character	0.2
Page Turn	End of request until first few lines are visible	1.0
Page Scan	End of request until text begins to scroll	0.5
XY Entry	From selection of field until visual verification	0.2
Function	From selection of command until response	2.0
Pointing	From input of point to display point	0.2
Sketching	From input of point to display of line	0.2
Local Update	Change to image using local data base, e.g., new menu list from display buffer	0.5
Host Update	Change where data is at host in readily accessible form, e.g., a scale change of existing image	2.0
File Update	Image update requires an access to a host file	10.0
Inquiry (Simple)	From command until display of a commonly used message	2.0
Inquiry (Complex)	Response message requires seldom used calculations in graphic form	10.0
Error Feedback	From entry of input until error message appears	2.0

Figure 7.9: Military Standard MIL-STD-1472F Table XXII Response times.

implementation. Additional dimensions could be coded with additional viewports, given sufficient memory and compute power.

Spaceslice supports up to twelve Aspects (3D-views) of five dimensions. The current implementation can support up to seven concurrent dimensions. Additional viewports and Aspects for up to seven dimensions can be added by updating the viewport display tables used by the Test-Fixture.

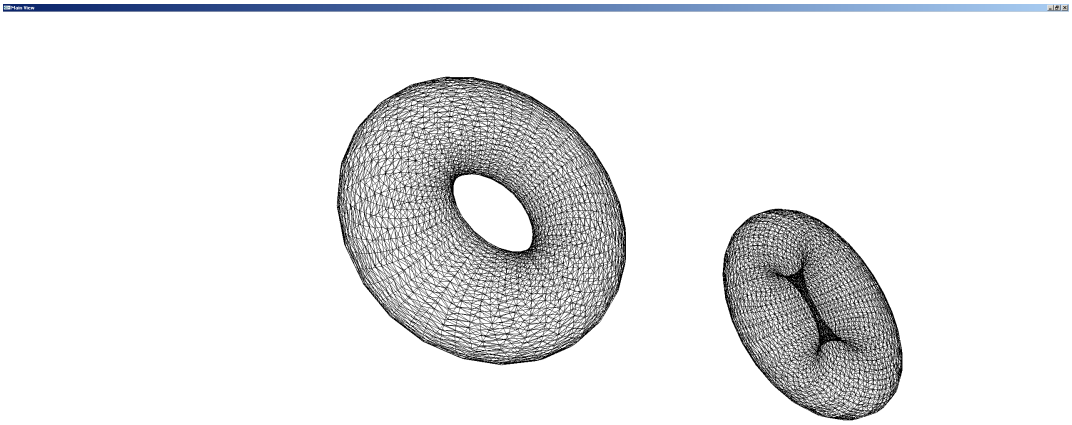


Figure 7.10: Donut Pair - 3D Tori Plucked from a 4D Torus by Intersection with the 3-flat

7.4.3 Tangible Results

A 3D sphere plucked from its embedding 4D sphere as shown in Figure 7.6 or a 3D tori pair plucked from their embedding 4D torus as shown in Figure 7.10 are obvious tangible results. Should the user be able to reconstruct these two common figures from memory, then the research has fulfilled the criteria for Hanson's Visualization Principle.[39]

7.5 Slices are Representative

As discussed in sections 7.3 and 7.4.3, tangible results can be seen in Figure 7.6 in which a sphere was plucked from its embedding in the 3-manifold. The 3rd party view shows a tangible result. Another example is shown by the pair of tori in Figure 7.10 which were plucked from their embedding 4D torus in 4-space shown in Figure 7.11.

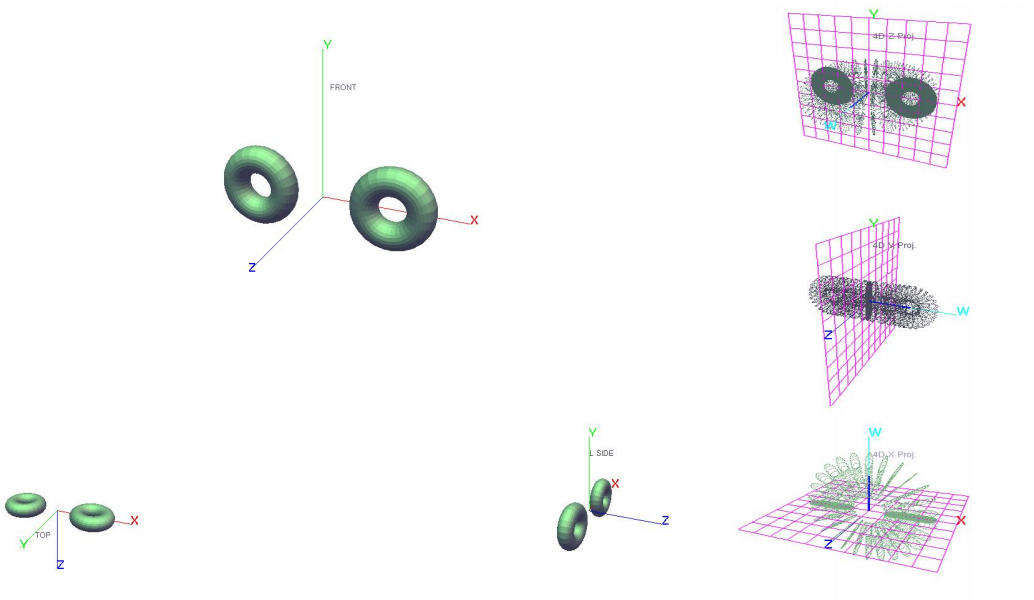


Figure 7.11: 4D Torus - The Source of the Plucked Donut Pair

Chapter 8

Future Work

Visualizing the morphology of Extra Dimensions can be treated as a field in its own right, and as such there is a wealth of research that is yet to be performed. Firstly, consolidation of prior research is essential.

8.1 User Studies

There is an interesting juxtaposition between Hanson's statement, "*We can argue that* some intellect*, possibly superhuman, can in fact understand the 4D images we produce*", and Aguilera's repetition of Allen's question about how the effect "*..of the increasing use of videogames ... will change the way people think..*". The readers may draw their own conclusions. As reminded by Weiskopf, "*..it is a grand goal to find some kind of (formalized) metric to assess the effectiveness..*" of visualizations. It is imperative that Weiskopf's formalized metric be found so that visual effects such as transparency, lighting, shadows, perspective, animation, depth-of-field, focus, fog, blur, red-shift, brightness, subtended-angle, multiple Points-of-View, ghosting, multiple-aspects, and so forth, can be efficiently applied as engineering tools to craft an effective visualization of extra dimensions. At that time it may be possible to answer these questions of Hanson and Allen.

8.2 Exploring Dynamic Non-Euclidean Space

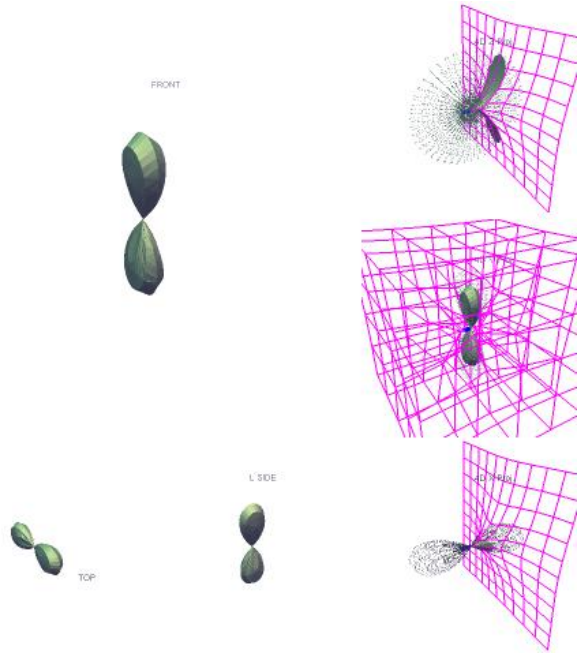


Figure 8.1: Slice of a 4D Torus in a non-Euclidean 4-space.
Accompanying video filename figure-8.1.mov. Video ID: I36m09s21sa05r51.52t0_ts8o15w_KVhoeA

Physics and mathematics are a wealth of dynamic non-Euclidean spaces. Solving non-linear equations analytically is fraught with challenges whether performed by hand, by computer, or computer assisted. As described in Section 4.2, a combination of the *Spacegrid* and the *Spaceslice* implementations by the simple expedient of perturbing the vertices of the *Spaceslice* model using the *Spacegrid*'s algorithms, either pre- or post- slicing will efficiently address this challenge. In either case the 3-flat code is still a linear intersection with an approximation of the warped n-space visualization that can be generated interactively in realtime. Shown in Figure 8.1 is an example of slicing a three-manifold in non-Euclidean 4-space. Figure 8.2 is the same slice in Euclidean space. It is expected that this strategy would provide a tangible experience with curved space comparable to the *Spaceslice* Euclidean implementation. Figure 8.1 is a proof-of-concept of a

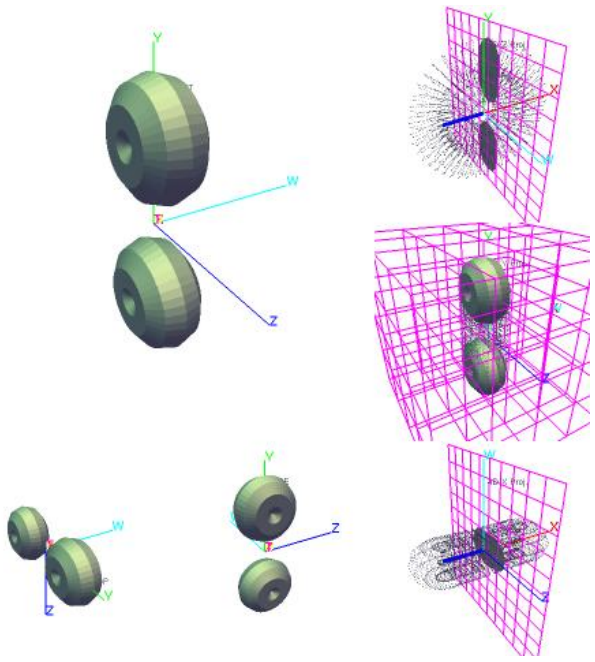


Figure 8.2: Slice of a 4D Torus in Euclidean 4-space.
Accompanying video filename figure-8.2.mov. Video ID: I36m09s21sa05r51.52t0_ts8o15w_KV

non-Euclidean display showing the result of a merger of the Spacegrid and Spaceslice paradigms. Further work is required to pluck an E^3 artifact from a non-Euclidean 4-space.

8.3 Open Questions

8.3.1 Emergent Surface Normals for Simplices of Arbitrary Dimension

A mechanism to generate and intersect generic n-simplices for an arbitrary dimension n would be useful. The intersection mechanism should also provide emergent surface normals for the simplex sub-manifold without additional processing.

8.4 Pedagogical Applications for Extra Dimensional Visualization

8.4.1 The Ehrenfest Paradox

Consider a traditional merry-go-round rotating at a relativistic velocity. Paradoxically, the circumference must Lorentz contract in the direction of rotation, while the radius must stay a constant length. At 0.866 c, for example, the length contraction is 50%.

A 4D spacetime representation of the "relativistic merry-go-round" may provide interesting and comprehensive insight into the Ehrenfest paradox.[1] The *Spaceslice* package has the capability to *sweep*[2] a rotating 2-sphere into 4D spacetime. The package can also perform a Lorentz transform during the sweep operation. The resultant 4D object would then lend itself to visualizing the Ehrenfest Paradox of the relativistic merry-go-round.

8.4.2 (6+1)D Phase-spacetime

In undergraduate physics, phasespace is used to analyze a 3D object's equation-of-motion in a 1D world. With the upgrade of *Spaceslice* into 7D, a 3D object's or particle's phasespace can be analyzed in all six degrees of freedom: the object's 3D position and 3D velocity or momentum.

A 7D visualization could be created by a 3D velocity space crossed with a (3+1)D spacetime to yield a (6+1)D phase-spacetime. Visualizing geodesics may yield interesting views. (6+1)D phase-spacetime may also provide new insights into the Ehrenfest Paradox.

[1] An in depth discussion of the **Ehrenfest Paradox** is beyond the scope of this work.

[2] **Sweep** - A CAD-like operation to extrude an object along an orthogonal axis into a higher dimension while simultaneously performing affine matrix transformations on the object, usually from 2D to 3D, but 3D to 4D as used here. May be extended to include Lorentz transforms.

8.4.3 Lorentz Invariant Hypervoxel Volumes

According to Einstein's Relativity, a 4D hypervoxel's hypervolume is invariant. This could be interactively demonstrated and explored with suitable 4D models.
It would be interesting to observe the dynamic symmetries of perturbed Lorentz invariant spacetime hypervoxels in the vicinity of high energy densities (mass). Visualizing the geodesics (around black holes, for example) may yield visually interesting symmetries of General Relativity.

8.4.4 Inflation and Expansion of the Universe

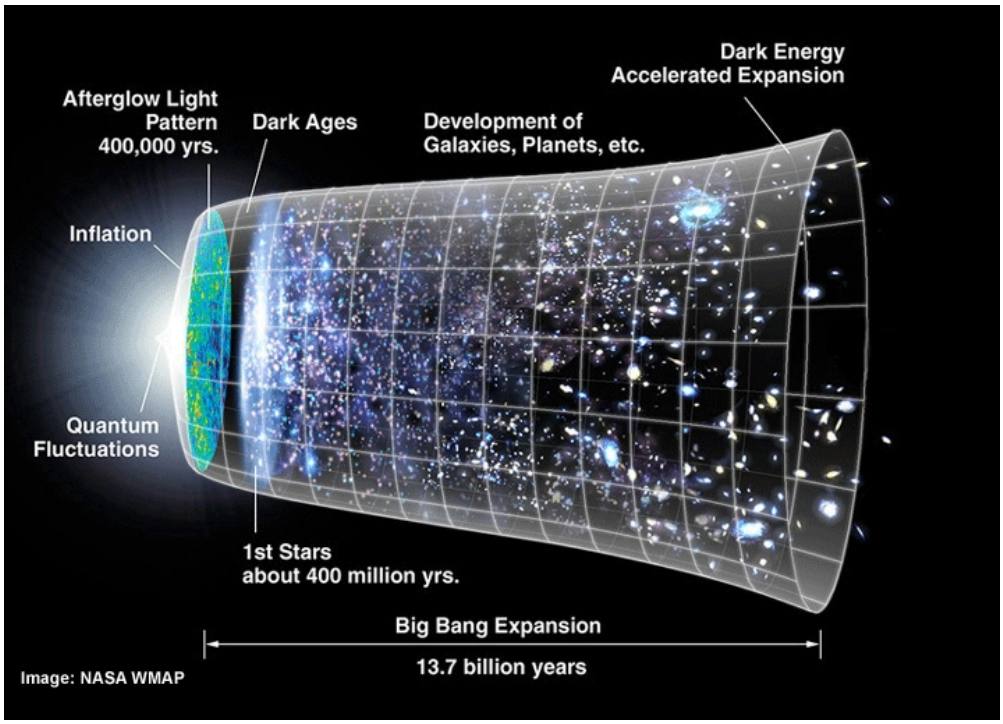


Figure 8.3: Big Bang, Cosmic Inflation, and the Cosmic Microwave Background
Timeline of the Universe. Credit: NASA/WMAP Science Team

The expansion of the Universe could be interactively visualized and explored with a (3+1)D spacetime diagram. Exploring the Universe's shape from the Big Bang,

through Cosmic Inflation, up to the present era could be quite insightful for a curious student.

Chapter 9

Conclusion

The ultimate goal has been to provide the user a tangible and memorable experience with mathematical models of four dimensional objects such that the user can see the model from any vantage point.

By use of a 4D GUI, an arbitrary convex hull or 3D silhouette of the 4D model can be rotated, panned, scrolled, and zoomed until a suitable Aspect is obtained. The 4D GUI then allows the user to manipulate a 3-flat hyperplane cutting tool to slice the model at an arbitrary orientation and position to extract or "*pluck*" an embedded 3D slice or "*aspect*" from the embedding four-space.

This plucked 3D aspect can be viewed from all angles via a conventional 3D viewer using three multiple POV viewports and optionally exported to a third party CAD viewer for further manipulation.

Plucking and Manipulating the Aspect provides a tangible experience for the end-user in the same manner as any 3D Computer Aided Design viewing and manipulation tool does for the engineer.

In order to achieve this objective, the following steps were taken.

1. Codified 3D to 4D Extrapolation Strategies:

   This technique to envision solutions to higher dimensional challenges is

necessary for 3D observers. Developing a methodology to extrapolate a sphere's bounding triangular mesh to a hyper-sphere's bounding tetrahedral mesh is an example of this strategy.

2. Reviewed visualization of non-Euclidean Minkowski spacetime:
   Examined the formation of a data structure to support 3D objects extruded into 4D spacetime in Section 2.3. This identified an algorithm to describe the hypersurfaces of higher dimensional mathematical models, to reduce these model's dimensionality and to visualize the lower dimensional representations in realtime.

3. Reviewed multiple points-of-view:
   Test fixtures were developed to demonstrate that multiple points-of-view are useful to visualize non-intuitive phenomena such as special relativity, and by extrapolation, extra dimensions as discussed in Section 7.1.1.

4. Discovered Aspects - multidimensional 3D views of 4-space objects:
   The concept of a 3D Aspect of a 4D object was discussed in Section 2.4 as a 4D extrapolation of multiple points-of-view. A test fixture was created to test and confirm the concept. This new realtime multi-aspect technology was added to the conceptual Test-Fixture for visualizing and exploring extra dimensions.

5. Discovered an efficient data structure:
   An extension to the spacetime data structure reviewed in Item 2 was added to the software Test-Fixture to represent objects of extra dimensions. A module to use this data structure to craft objects of extra dimensions was also added to the Test-Fixture in Section 6.

6. Developed an Extra Dimensional GUI:

A GUI was crafted to allow the user to manipulate the orientation and position of a 4D or 5D object in 4-space or 5-space in realtime. The 3-flat or 4-flat intersection tool (the slicer) can be positioned and oriented via a 3D gridded icon. The ED-GUI is described in Section 6.

7. Developed the Pluck methodology to extract an embedded 3D object from 4-space:

   It was shown that the plucked 3D object was representative of the 3-manifold. The Test-Fixture was modified to implement a file output function to export the plucked object in a common 3D file format to a third party CAD viewer as described in Section 6.

The ultimate goal has been reached, along with a few penultimate en passant achievements as listed above. A tool has been developed to allow the user to interactively explore three-manifolds in realtime.

Appendix A

Matrix Composition

Bars over variables or functions (for example: $\overline{matrix}$) indicate that they are matrices or return matrices, respectively. Mtx7h is a 7D (8x8) homogeneous matrix of real numbers. It is dimensionally reduced by the display module to the appropriate 3D (4x4) homogeneous matrix when it is passed to the OpenGL renderer for each individual viewport. All mouse *click & drags* are associated with a specific viewport.

- Mtx7h makeRotate7D(toAxis, fromAxis, radians)

  Make a 7D rotation matrix to rotate the (*toAxis*,*fromAxis*) plane from axis *fromAxis* to axis *toAxis* by *radians* radians.

```
/*****************************************/
/* Create a 5D rotation matrix from indices */
/* of the two axes comprising the rotation  */
/* plane and the angle (in degrees).        */
/* The following example is a rotation of   */
/* the XW plane A radians from X into W.    */
/* or 1 -> 4 of A degrees                   */
```

```
/*      0    1    2   3   4   5   6          */
/*  0   1    0    0   0   0   0   0          */
/*  1   0   cA    0   0  -sA   0   0          */
/*  2   0    0    1   0   0   0   0          */
/*  3   0    0    0   1   0   0   0          */
/*  4   0   sA    0   0   cA   0   0          */
/*  5   0    0    0   0   0   1   0          */
/*  6   0    0    0   0   0   0   1          */
/*                                           */
/* Cos[] are on the diagonals                */
/* Sin[] are antisymmetric about diagonal    */
/*******************************************/
```

- Mtx7h reduce7Dto3D(which3Axes)

  Compose a 7D matrix to shift the 3 components specified in *which3Axes* to the X,Y,Z components.

```
//  Compose a 7D matrix to capture the 3 Axes specified
//  by whichAxes.  For example to map (t,Z,X,Y) of a 4D
//  homogeneous matrix to (X,Y,Z,0):
//  { 0, 0, 0, 0, 0}    The coefficients other than
//  { 0, 0, 0, 1, 0}    X,Y,Z in the target matrix
//  { 0, 1, 0, 0, 0}    are irrelevant since only the
//  { 0, 0, 1, 0, 0}    (X,Y,Z) components are used by OpenGL
//  { 0, 0, 0, 0, 0}    once the matrix is reduced to 3D
```

- Mtx7 matrixPOVn(iViewport)

  Make a matrix for the viewport *iViewport* from the mouse position using open keyboard (no control, shift or Alt keys pressed).

H_l,V_l Horizontal & vertical mouse positions w/ left button

H_m,V_m Horizontal & vertical mouse positions w/ middle button

V_r Vertical mouse position w/ right button

$$\overline{rotate(\angle H_l)} * \overline{rotate(-\angle V_l)} * \overline{translate(H_m, V_m, V_r)} \tag{A.1}$$

- Mtx7h matrixSlice(iViewport, invertFlag)

  Make a matrix for the 3-flat from viewport number *iViewport*. Use the mouse position captured when the keyboard SHIFT key was pressed, using the same mouse symbol convention as in $\overline{matrixPOVn()}$, above. The values of $h, \hat{h}, v, \hat{v}$ correspond to the indices components of the horizontal matrix and the vertical matrix, respectively, that will be rotated; and a,b,c correspond to the components of the homogeneous matrix that will be used for translation. The values of these indices as specified by *iViewport*, determine the input characteristics of the Viewport for which the mouse commands are being processed.

  The 3-flat is rotated by $\overline{Rot}$ for each viewport, then intersected with the 4D model. The intersection algorithm is discussed elsewhere.

$$(h, \hat{h}, v, \hat{v}, a, b, c) = func(iViewport) \tag{A.2}$$

$$\overline{Rot_{iViewport}} = \overline{rotate(h, \hat{h}, -\angle H_l)} * \overline{rotate(v, \hat{v}, -\angle V_l)} * \overline{translate(a, b, c, -H_m, V_m, V_r)} \tag{A.3}$$

  When *invertFlag* is set, the de-rotation matrix is built here. The de-rotation matrix is used to restore the intersecting 3-flat to its nominal Euclidean state. Consequently the *plucked* 3D object appears undistorted to the observer in its Euclidean space.

$$\overline{deRot_{iViewport}} = \overline{translate(H_m, -V_m, -V_r)} * \overline{rotate(\angle V_l)} * \overline{rotate(\angle H_l)} \quad \text{(A.4)}$$

The OpenGL view transform matrix is composed as follows. The 7D matrix is decomposed into the OpenGL homogeneous 3D matrix format. The $\overline{Viewport}$ matrix is a viewport specific constant initialized at startup. The OpenGL view matrix is composed from the mouse position within the specified viewport.

$$\overline{OpenGL} = \overline{Viewport} * \overline{rotate(\angle H_l)} * \overline{rotate(-\angle V_l)} * \overline{translate(H_m, V_m, V_r)} \quad \text{(A.5)}$$

Each viewport is set to display a fixed set of dimensions at startup. A viewport table (panel[]) contains this information and is referenced by the display driver when each viewport is displayed.

$$\overline{reduce7D_{iViewport}} = \overline{reduce7Dto3D(thisViewportsAxes_{iViewport})} \quad \text{(A.6)}$$

In addition to the mouse and keyboard, the orientation of the nD object can be modified by the control panel. A *soccerball* rolling ball GUI updates the $\overline{rotateBall}$ matrix with its rotation in the specified plane. The control panel table of axis-to-axis rotation fields also affect the $\overline{rotate7D}$ via the $\overline{makeRotate7D()}$ table of view matrices as shown below.

$$\overline{rotate7D} = \overline{rotateBall} * (\prod_{j,k=0}^{j,k=6} (\overline{makeRotate7D(j, k, rotateValue_{j,k})})) * deRot \quad \text{(A.7)}$$

These latter two viewport specific matrices are used to transform the vertices prior to being handed off to the OpenGL pipe for rendering.[1]

$$vert3 = \overline{reduce7D_{iViewport}} * \overline{rotate7D} * vert7 \tag{A.8}$$

Each 7D vertex $vert7$ is rotated and reduced as by Equation A.8 to yield a 3D vertex $vert3$, and then passed on to be rendered by the OpenGL pipe, which was initialized with the $\overline{OpenGL}$ matrix defined in Equation A.5, above.

[1] $\overline{reduce7D}$ is computed just-in-time prior to the drawObject() invocation, so there is no need or use of a storage array.

Appendix B

Derivation of Intersection of a Line with a 3-Flat

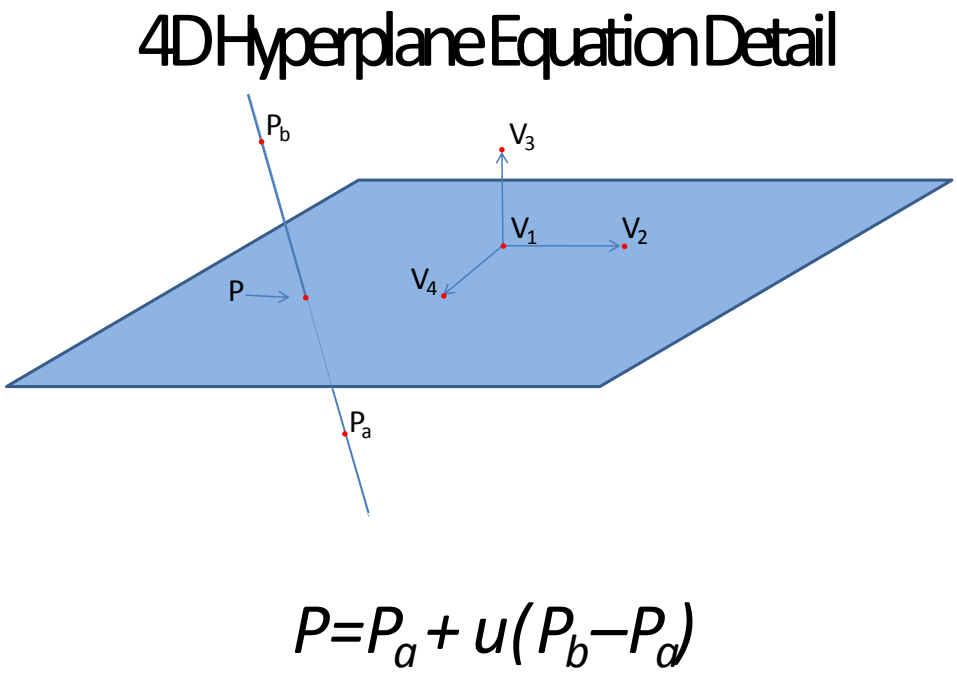


Figure B.1: 4D 3-Plane Equation Derivation

B.1 3-Flat Intersection with Line in 4-space

Consider a 3-flat defined by the four vertices V_1, V_2, V_3, V_4 as depicted in Figure B.1, to be intersected by the line defined by the two points P_a and P_b. The intersecting point P on the 3-flat as represented by the shaded plane in Figure B.1, is found by plugging the coordinates of the vertices and points into the determinants of Equation B.3 and solving for P in Equation B.2.

$$0 = \begin{vmatrix} \hat{t} & \hat{x} & \hat{y} & \hat{z} & 1 \\ V_1^t & V_1^x & V_1^y & V_1^z & 1 \\ V_2^t & V_2^x & V_2^y & V_2^z & 1 \\ V_3^t & V_3^x & V_3^y & V_3^z & 1 \\ V_4^t & V_4^x & V_4^y & V_4^z & 1 \end{vmatrix} \tag{B.1}$$

$$P = P_a + u(P_b - P_a) \tag{B.2}$$

Solving yields:

$$u = \frac{\begin{vmatrix} 1 & 1 & 1 & 1 & 1 \\ V_1^t & V_2^t & V_3^t & V_4^t & P_a^t \\ V_1^x & V_2^x & V_3^x & V_4^x & P_a^x \\ V_1^y & V_2^y & V_3^y & V_4^y & P_a^y \\ V_1^z & V_2^z & V_3^z & V_4^z & P_a^z \end{vmatrix}}{\begin{vmatrix} 1 & 1 & 1 & 1 & 0 \\ V_1^t & V_2^t & V_3^t & V_4^t & P_b^t - P_a^t \\ V_1^x & V_2^x & V_3^x & V_4^x & P_b^x - P_a^x \\ V_1^y & V_2^y & V_3^y & V_4^y & P_b^y - P_a^y \\ V_1^z & V_2^z & V_3^z & V_4^z & P_b^z - P_a^z \end{vmatrix}} \tag{B.3}$$

B.2 4-Flat Intersection with Line in 5-space

Consider a 4-flat defined by the five vertices V_1, V_2, V_3, V_4, V_5 to be intersected by the line defined by the two points P_a and P_b. The intersecting point on the 4-flat is found via the following simultaneous equations in a manner similar to that given in Section B.1:

$$0 = \begin{vmatrix} \hat{t} & \hat{x} & \hat{y} & \hat{z} & \hat{w} & 1 \\ V_1^t & V_1^x & V_1^y & V_1^z & V_1^w & 1 \\ V_2^t & V_2^x & V_2^y & V_2^z & V_2^w & 1 \\ V_3^t & V_3^x & V_3^y & V_3^z & V_3^w & 1 \\ V_4^t & V_4^x & V_4^y & V_4^z & V_4^w & 1 \\ V_5^t & V_5^x & V_5^y & V_5^z & V_5^w & 1 \end{vmatrix} \tag{B.4}$$

$$P = P_a + u(P_b - P_a) \tag{B.5}$$

Solving yields:

$$u = \frac{\begin{vmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ V_1^t & V_2^t & V_3^t & V_4^t & V_5^t & P_a^t \\ V_1^x & V_2^x & V_3^x & V_4^x & V_5^x & P_a^x \\ V_1^y & V_2^y & V_3^y & V_4^y & V_5^y & P_a^y \\ V_1^z & V_2^z & V_3^z & V_4^z & V_5^z & P_a^z \\ V_1^w & V_2^w & V_3^w & V_4^w & V_5^w & P_a^w \end{vmatrix}}{\begin{vmatrix} 1 & 1 & 1 & 1 & 1 & 0 \\ V_1^t & V_2^t & V_3^t & V_4^t & V_5^t & P_b^t - P_a^t \\ V_1^x & V_2^x & V_3^x & V_4^x & V_5^x & P_b^x - P_a^x \\ V_1^y & V_2^y & V_3^y & V_4^y & V_5^y & P_b^y - P_a^y \\ V_1^z & V_2^z & V_3^z & V_4^z & V_5^z & P_b^x - P_a^z \\ V_1^w & V_2^w & V_3^w & V_4^w & V_5^w & P_b^w - P_a^w \end{vmatrix}} \tag{B.6}$$

If one assumes that there is no vertex V_5 that is not co-hyper-planar with the 3-flat then this 4-Flat equation can be simplified to the 3-Flat equation.

Appendix C

Model Builder Documentation (mkhyper V7.5)

This section shows the command line usage documentation and sample output for mkhyper V7.5, and demonstrates the input format expected by the nD viewer. This module produces the 4-manifold with closed compact 3-manifold boundary. User can select to output a model of one of the following in 4-space: 2-sphere; 3-sphere; spinning 2-sphere; 3-torus, 4-cube. User controls the radii of the objects and all three radii of the 3-torii. The models can be output in 4-space or into a 4D spacetime. The velocity vector of spacetime models can be specified, as well as the angular velocity vector of spinning models. Models are described by their boundaries of pure simplicial 2-complexes (triangles) or 3-complexes (tetrahedra), depending on if a wireframe or shaded visualization is intended, respectively. The module generates a count of, and then the list of 7D vertices to be used by model. In this example, a list of triangles is then generated, defined by their five indices into the vertex table. The first three indices select the triangle's vertices. The last index selects an offset vector, while the second to the last index selects a velocity vector. Note that in this example all triangles share the same offset and

velocity. The offset is zero, while the velocity is 7.0 along the t axis indicating a duration of 7.0 units.

Lighting information is also provided for each triangular face or tetrahedral hyperface.

```
================================================================
================================================================
mkhyper.exe Command Line Usage

/?        - Help (this)
/a  <n>   - Regular (0); odd (1) or Even (2) object tetra's
/b  <f>   - Beta velocity ( fVelocity = dTime * Beta)
/c  <n>   - Number of cycles in time /t [0]
/d  <f>   - Angular Delta V
/h  <n>   - Hue <n>
/i        - Use Indexed Vector List [no]
/o  <n>   - generate object 'n' [0]
     0    - 2-sphere in 3D
     1    - 3-torus  in 4D v2.0 in 5-space (XYZW) - wire
    21    - 3-torus  in 4D v2.0 in 5-space (XYZW) - solid
     2    - 3-torus  in 4D v2.0 in 4-space (TXYZ) - wire
    22    - 3-torus  in 4D v2.0 in 4-space (TXYZ) - solid
     3    - 3-torus  in 4D v2.1 in 5-space (XYZW) - wire (Gopi)
    23    - 3-torus  in 4D v2.1 in 5-space (XYZW) - solid (Gopi)
     4    - 3-sphere in 5-space (XYZW)
    24    - 3-sphere in 5-space (XYZW)
     5    - 3-sphere in 4-space (TXYZ)
     6    - 2-torus  in 3-space (XYZ)
     7    - 3-torus  in 4D v1.0 (XYZW)
     8    - 3-torus  in 4D v1.1 (XYWZ)
     9    - 3D Calabi-Yau surfaces
    10    - 3-torus  in 4D v3.0 ([T]XYZW)
```

```
    11    - 3-torus  in 5D v1.1 ([T]XYZW)
    12    - 3-torus  in 5D vx.x ([T]XYZW)
    13    - 5D cube  v 3.0 in 5-space ([T]XYZW)
    14    - 2-sphere in 4D Rotating via 4-space (TXYZ)
/r  <f>   - Radius scale [5.0]
/ra <f>   - Radius A scale [2.0]
/rb <f>   - Radius B scale [2.0]
/s  <f>   - steps per PI in three sphere [7.5]
/t  <f>   - timeaxis extrusion [1.0]
/v  <f>   - angular velocity
Y:\4D\mkhyper>debug\mkhyper /o 0 /t 7 /s 9 /r 7 /i > ..\sdf\mkndx_00s9r7t7.sdf
## debug\mkhyper /o 0 /t 7.000000 /r 7.000000 /ra 2.000000 /rb 2.000000 /s 9.00000
## 2-sphere in XYZ 3-space
##Allocated 174 Vectors (10K)
```

```
=================================================================
=================================================================
mkhyper.exe File (.SDF) Sample Output:


## debug\mkhyper /o 4 /t 7.000000 /r 7.000000 /ra 2.000000 /rb 2.000000 /s 7.00000
##V1.5#################################
         snip
#######################################
## 3-sphere in XYZW 4-space
vectors 794
(7.000000,0.000000,0.000000,0.000000,0.000000,0.000000,0.000000)
(0.000000,0.000000,0.000000,0.000000,0.000000,0.000000,0.000000)
(0.000000,0.000000,0.000000,0.000000,7.000000,0.000000,0.000000)
(0.000000,0.000000,0.000000,3.037186,6.306782,0.000000,0.000000)
(0.000000,0.571766,1.187284,2.736410,6.306782,0.000000,0.000000)
(0.000000,0.000000,1.317786,2.736410,6.306782,0.000000,0.000000)
         snip
(0.000000,-0.000000,0.000000,0.000000,-7.000000,0.000000,0.000000)
###End of Vector Records
## 3-sphere in XYZW 4-space
itriangle 2,3,2,0,1
diffuse      [0.5, 0.5, 0.90]
specular     [1, 0.9, 0.9]
reflectivity 0.4
Phong_exp    20
itriangle 3,3,2,0,1
```

```
diffuse      [0.5, 0.5, 0.90]
specular     [1, 0.9, 0.9]
reflectivity 0.4
Phong_exp    20
itriangle 2,4,2,0,1
diffuse      [0.5, 0.5, 0.90]
specular     [1, 0.9, 0.9]
reflectivity 0.4
Phong_exp    20

=====================================================================
```

Appendix D

Lattice Viewer Documentation (Spacegrid V5.72)

This module generates the lattice display of a 4-space and a 5-space as multiple Aspects in multiple viewports as specified by the /N parameter.

```
==================================================================
Command line parameters:
3: Use 3DSMax GUI      [off]
D: Debug level         [0]
G: set Grid limits     [-10,10,0.5,40]
I: Input filename      [sc_ani.sdf]
L: Label (N/I)
N: Number of dimensions [6]
O: Output filename     [V:/ppm/sc_ani]
P: Panel (not image) size [160,160]
R: Rotation (T/Z): 0-Euclidean; 1-Minkowski [1] 2-Theory, 3-Test
S: Swap T & Z axes [1]
T: Time (N/I)
```

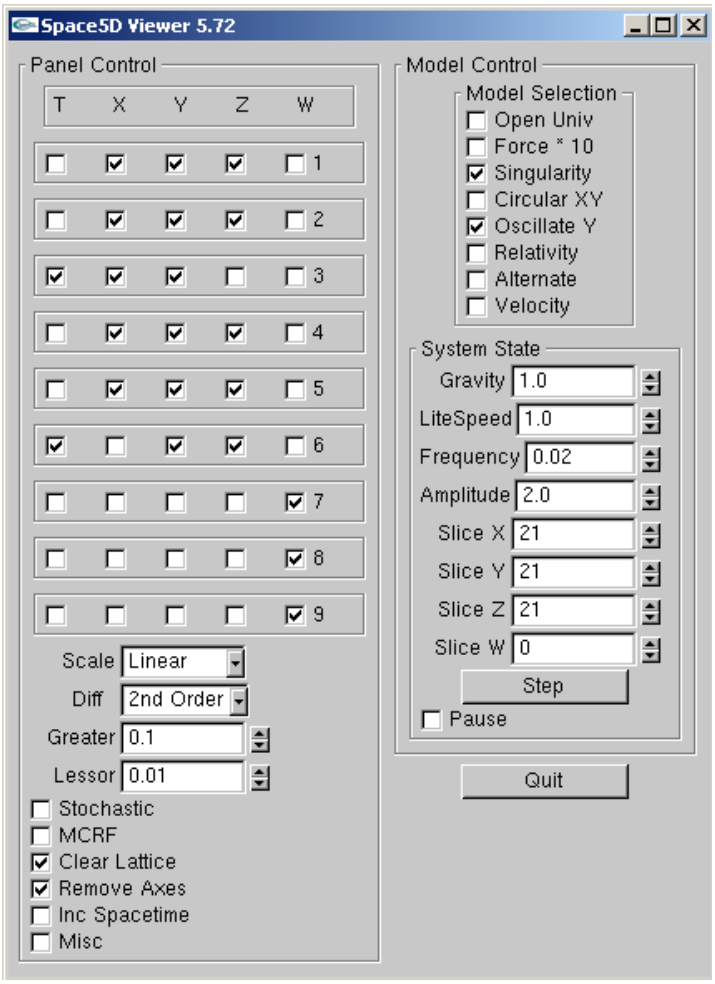


Figure D.1: Spacegrid V5.72 Control Panel

```
?: List this menu

Example:

  Ray4D /O frame_no /I ../conf/setup.sdf

Note: Last parm processed has precedence

=====================================================================
```

```
================================================================
Spacegrid V5.72 4D: ( 40, 42, 42, 42)
     requesting 361.758 MB as 40 blocks of 9.044MB
     Using a Grid of -10.000000, 10.000000, 0.500000, 40.000000


Usage:
Available one character keyboard commands from view window:
q - S: Quit
A - D: toggle Axes display [ON]
C - M: Move BH to Center
D - S: toggle color Debug
M - P: toggle Mark high values w/ circle
R - D: Reset to nominal position
S - S: Enable Statistics output [off]
a - D: toggle showing Acceleration [off]
b - M: Toggle open Boundary [on]
c - D: Track BH (N/I)
d - D: toggle Delta [off]
e - D: toggle x 100 Exaggeration [off]
f - S: File this sequence [off]
g - D: toggle Grid display [on]
h - M: toggle black Hole [on]
i - D: Exponential (w/Log) [off]
j - D: toggle Jerk (3rd order) [off]
k - S: Enable status display [off]
l - D: toggle Lorentz Transform mode [off]
m - S: toggle Miscellaneous flag [accel]
```

```
n - M: toggle Circle Oscillation [off]

o - M: toggle Oscillator [off]

P - S: toggle debug Print

p - S: toggle debug Pause

q - S: Quit

r - M: toggle Relativistic mode[off]

s - D: toggle Stochastic mode [off]

t - D: Increment thru XYZ,TXY, TYZ, TXZ

u - M: Use alternative mod [off]

v - M: toggle Velocity mode [off]

w - P: toggle draw static grid marks

x - D: toggle show 4 Volume on W axis

z - M: Zero model lattice[off]

0 - Select all panels[0]

1..6 - Select panel number [0]

? - This menu list

Left Mouse  - Rotate

Mid Mouse   - Pan & Scroll

Right Mouse - Zoom

CTRL/Left   - Rotate T/Z



State: /R Minkowski(1)

=====================================================================
```

Appendix E

Model Viewer Documentation (SpaceWalk V7.50)

This module imports the output of the mkhyper module and displays the 4D and 5D convex hulls as multiple Aspects in multiple viewports as specified by the /N parameter.

```
SpaceWalk7D V7.5 Command Line Usage:
=====================================================================
SpaceWalk7D V7.50
For commandline invocation usage, enter:
debug\SpaceWalk7D /?
For interactive character command usage, enter:
?
Command line parameters:
3: Use 3DSMax GUI      [off]
A: Automated animation mode file name [none]
B: Animation filename beginning number [0]
D: Debug level         [0x0000]
```

```
       0x2000 - Print FlatC_WikiM Vectors
       0x4000 - Print Sliced Poly Vectors
       0x8000 - Print Dets, Inv's & Xpose Matrices
F: Toggle Special (Fn) key 'n' [none]
G: set Grid limits     (-1.00  1.00  0.25 10.00)
I: Input filename      (../sdf/sc_ani.sdf)
L: Label (N/I)
M: 5D size             [1.0]
N: Dim/Display Number (1-6) [6]
   2 (1);       3: (2x2)3D; 4: (2x3)4D;  5: (2x3)5D;  6: (3x3)5D;
   7: (3x3)4D;  8: (3x4)5D; 9: (2x5)5D; 10: [3x3)5D; 11: [3x4)4D
  12: [3x5)5D; 13: [3x3)4D;
O: Output filename     (../ppm/Space5D/sw7.50_)
P: Set image Size via Panel size (wd,ht) [240,240]
R: Rotation (T/Z): 0-Euclidean; 1-Minkowski [1] 2-Theory, 3-Test
T: Transparency [1.0]
Y: Yank in & execute animation script file (../sdf/sc_ani.scp)
?: List this menu with defaults
Example:
  SpaceWalk7D /O frame_no /I ../conf/setup.sdf
Note: Last parm processed has precedence
=====================================================================
```

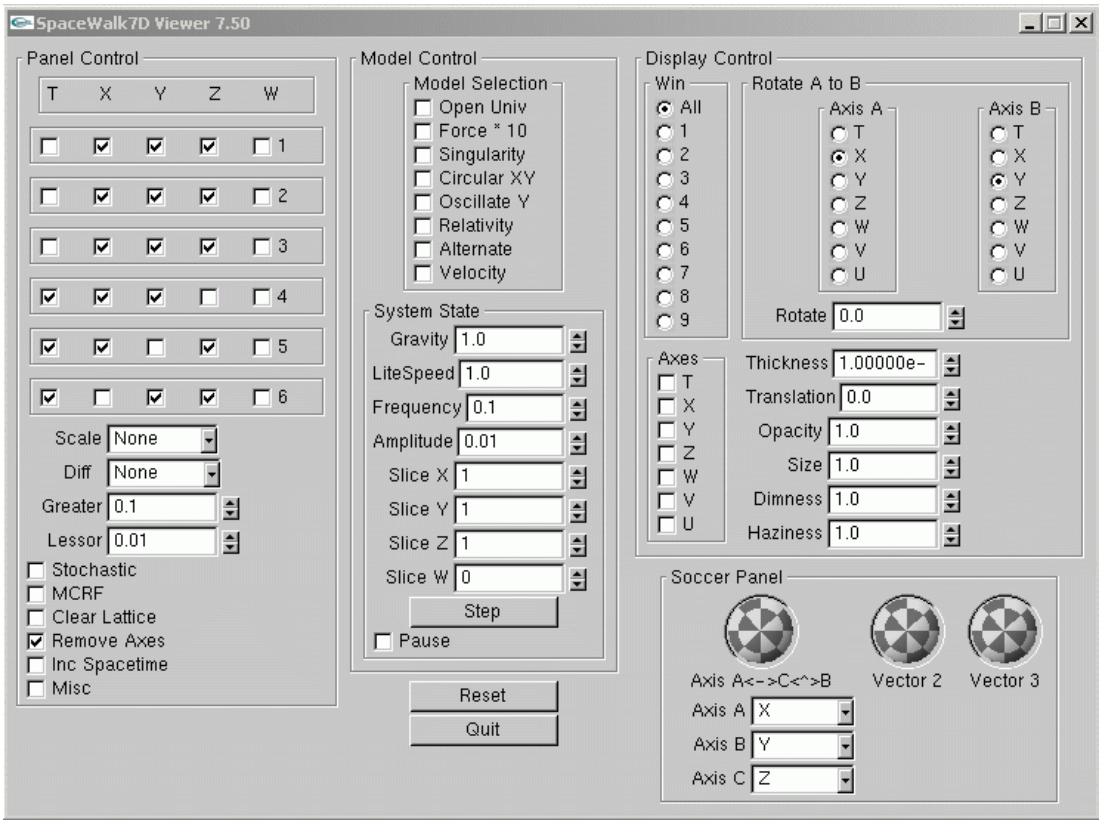


Figure E.1: SpaceWalk V7.50 Control Panel

```
======================================================================
SpaceWalk7D V7.5 Console output:


Spacegrid V7.50 4D: ( 10, 12, 12, 12)
              requesting 2.109 MB as 10 blocks of 0.211MB
              Using a Grid of -1.000000, 1.000000, 0.250000, 10.000000


. Total cameras available: 1., camera mask 00000000
/L '' TS: '(null)'
Build GLUT display. Register callbacks. Windows for 6D. Register display callback.


SpaceWalk7D V7.50
```

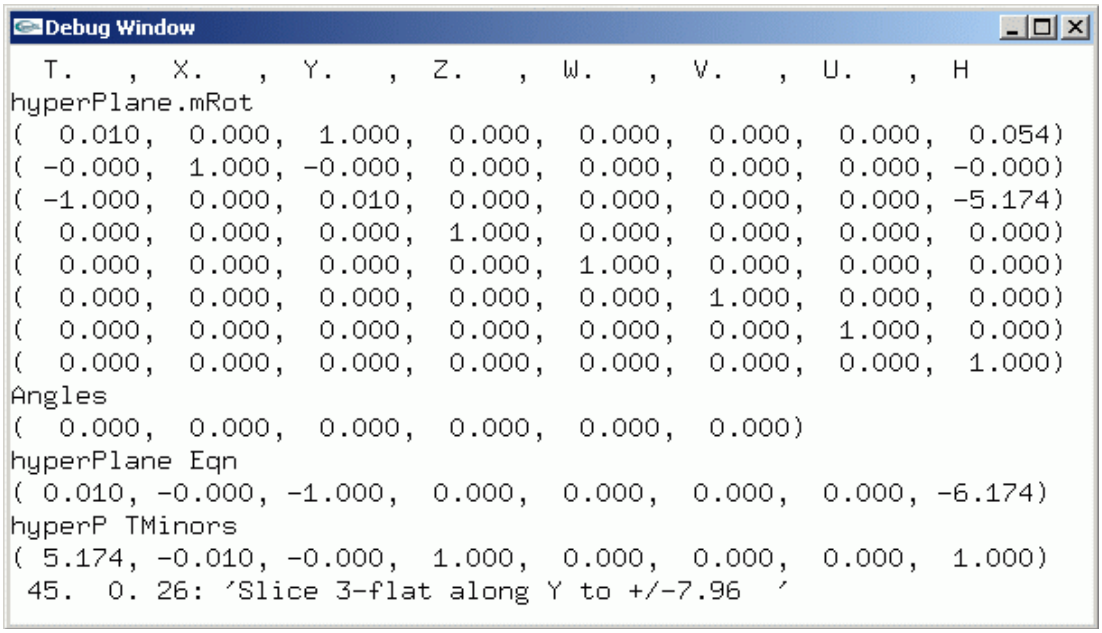


Figure E.2: SpaceWalk V7.50 Matrix Debug Window

```
For commandline invocation usage, enter:

debug\SpaceWalk7D /?

For interactive character command usage, enter:

?



Timing: Init: elapsed 1 seconds [00:00:01]

Timing: Copy: [00:00:00] elapsed seconds  0.000 ms 0/0

Timing: Draw: [00:00:00] elapsed seconds  0.000 ms 40/0

Timing: Run : [00:00:07] elapsed seconds (7)

================================================================
```

```
================================================================
SpaceWalk7D V7.5 Interactive Keyboard Commands:


Usage:
Available one character keyboard commands from view window:
q - S: Quit
A - D: toggle Axes display [ON]
C - M: Move BH to Center
D - S: Input Debug Hex Value
       0x2000 - Print FlatC_WikiM Vectors
       0x4000 - Print Sliced Poly Vectors
       0x8000 - Print Dets, Inv's & Xpose Matrices
E - M: increment Einstein spacetime [0]
H - D: show HyperSlice matrix window
I - D: Invert (derotate) sliced view via rotate7D [on]
L - D: toggle lighting mode [ON]
M - P: toggle Mark high values w/ circle
N - D: toggle clipNdx vs copyNdx slicing [clip]
P - D: draw polygons, not lines [false]
R - D: Reset to nominal position
S - M: Slice w/ 3-brane [off]
T - D: Slice T (not W) axis [ off]
W - D: Slice W (not T) axis [on]
Y - D: Reset hyperSlice to nominal state
a - D: toggle showing Acceleration [off]
b - M: Toggle open Boundary [on]
c - D: Track BH (N/I)
```

```
d - D: toggle Delta [off]
e - D: toggle x 100 Exaggeration [off]
f - S: File this sequence [off]
g - D: toggle Grid display [on]
h - M: toggle black Hole [on]
i - D: Exponential (w/Log) [off]
j - D: toggle Jerk (3rd order) [off]
k - S: Enable status display [on]
l - D: toggle Lorentz Transform mode [off]

m - S: toggle Miscellaneous flag [accel]
n - M: toggle Circle Oscillation [off]
o - M: toggle Oscillator [off]
p - S: toggle debug Pause
q - S: Quit
r - M: toggle Relativistic mode[off]
s - D: toggle Stochastic mode [off]
t - D: Increment thru XYZ,TXY, TYZ, TXZ
u - M: Use alternative mod [off]
v - M: toggle Velocity mode [off]
w - P: toggle draw static grid marks
x - D: label vertices [off]
y - D: Toggle scripting state [/Y set]
z - M: Zero model lattice[off]
0 - Select all panels[0]
1..9 - Select panel number [0]
? - This menu list
```

```
Left Mouse  - Rotate
Mid Mouse   - Pan & Scroll
Right Mouse - Zoom
CTRL/Left   - 3-brane Rotate (X,Y)
CTRL/Right  - 3-brane Zoom   (  Y)
F1  - Inc Show Slicer [off,on,3D]
F2  - Test for XSCT_ALL
F3  - Show clipNdx
F4  - Original 3D object: 0-Show!3D!F3,2-Hide,3-Hide 4 slice,4-Show ALL [0]
F5  - Show TFace/Slice/F05_2/F05_3
F6  - 1: Blend 2: Persp 3: Blend & Persp 4: 3D [0]
F7  - Use Mathworld/Wiki Slicer Algorithm [MW]
F8  - Test smooth shade
F9  - Select one of 6 slice matrices [0]
F10 - Test Determinant
F11 - Label More Vertices [off]
F12 - Walk Scene
=====================================================================
```

Appendix F

Contents of Accompanying CD

These are the video files associated with the figures in the PhD Dissertation by Don V Black, entitled: "*Computational Techniques to Enable Visualizing Shapes of Objects of Extra Spatial Dimensions*".
The figures in the dissertation are frames from these associated videos. The videos were generated by converting from .ppm format images to Apple QuickTime format .mov videos by the Linux FFmpeg video converter (ffmpeg) program executed under the Ubuntu 8.10 Linux distribution.
The sequences of .ppm frames were created programmatically as described in the dissertation. This accompanying CD was generated via the *MicroSoft XP-Pro 64 bit* operating system updated to SP-3 on an off-the-shelf DVD burner. The *Nero Home Essentials* retail CD-ROM software "burned" the files to the CD-ROM.

CD-ROM Contents:

Figure Filename	Video ID Information
figure-2.9.mov	PHD_teaser-AA.mov
figure-6.2.mov	PHD-62_TorusI_06m04s33r664t0_hTorus_twist-S_Z2.mov
figure-6.3.mov	PHD-63_TorusI_06m04s9r673t0_hTorus_twist_Z2.mov
figure-6.8.mov	PHD_3-sphereW_hSphereI_04s11r633t0_hSph02-633_sliceW.mov
figure-6.9.mov	PHD_3-torusW_hTorusI_36m05s13r521t0_hTor05I_cutWXYZr521p.mov
figure-6.10.mov	PHD-5D-LA_hSphere7_24sa090702r9t-7_V7.7_hSphereI_CutWXYZr9.65p_n12-edit
figure-7.2.mov	PHD_Dash-07a.mov
figure-8.1.mov	PHD-Warp-V84d-LA2-g100_I36m09s21sa05r51.52t0_ts8o15w_KVhoeA.mov
figure-8.2.mov	PHD-Warp-V84d-LA2-Control_I36m09s21sa05r51.52t0_ts8o15w_KV.mov
README.txt	This CD-ROM documentation.

Glossary

k-Flat	A k-dimensional (kD) Euclidean hyperplane of infinite extent in n-space, where $k \leq n$., 6
m-Cell	An m-dimensional cell that is a subset of a space whose interior is homeomorphic to an open m-ball in R^m [72]., 69
m-Cube	A cube of dimension m in n-space where $m \leq n$. A 3-cube is a common 3D cube, while a 2-cube is a square, and a 4-cube is a tesseract., 6
m-Manifold	A topological or mathematical space of m dimensions that is locally Euclidean. An m-space with each point's neighborhood homeomorphic to $\mathbf{E}^m$., 5
m-Simplex	A convex hull of (m+1) independent points in an n-dimensional (nD) Euclidean space E^n (where $n \geq m$). A triangle is a two-simplex, while a tetrahedron is a three-simplex. A simplex need not be regular., 68

m-Sphere	A connected compact closed smooth m-manifold in n-space, where $n > m$. A 2-sphere is the common hollow sphere in 3D which can be defined as the locus of the points $x^2 + y^2 + z^2 = radius^2$. A 3-sphere is a closed compact 3-manifold in 4D which is the locus of the points $w^2 + x^2 + y^2 + z^2 = radius^2$. An m-sphere is thus the locus of $\sum_{i=0}^{i=m} x_i^2 = radius^2$., 6
mD Object	As used in this dissertation will refer to a closed m-manifold with boundary, whose boundary is a compact closed (m-1)-manifold or hyperobject approximated by a pure simplicial (m-1)-complex., 60
nD	n-dimensional., 6
(3+1)D	4D Minkowski spacetime of three spatial dimensions and one temporal dimension., 5
(4+1)D	5D Minkowski spacetime of four spatial dimensions and one temporal dimension., 5
2D	two-dimensional., 6
3D	three-dimensional., 6

3D Slice	A three-dimensional (3D) sub-space or 3-space generated by the intersection of a 3-flat with an n-space or m-manifold in n-space. Usually, $n \geq m \geq 3$. When the intersected 4D object, or 3-manifold, is a pure simplicial 3-complex, the resulting 3D Slice is a 3D object defined by its bounding pure simplicial 2-complex commonly known as a bounding triangular mesh., 6
4D Object	As used in his dissertation will refer to a closed four-manifold with boundary, whose boundary is a compact closed three-manifold approximated by a pure simplicial three-complex., 62
5D Object	As used in this dissertation will refer to a closed five-manifold with boundary, whose boundary is a compact closed four-manifold approximated by a pure simplicial four-complex., 62
Aspect	Refers to a dimensionally reduced view of an nD object or n-space, usually displayed in 3D., 6
Co-dimension	The complementary dimensionality: the codimension of an m-manifold in n-space is an $(n-m)$-manifold., 6

Extrusion	A Computer Aided Design (CAD) construction operation that converts an n dimensional object into an $n+1$ dimensional object. The vertices, edges, and faces of the object are all extruded into the next higher dimension along the extrusion vector and then connected by edges, planes and prisms, respectively., 20
Flat spacetime	Uncurved or 'flat' n-space - objects obey Newton's Laws, and a geodesic (straight line) is straight., 18
Hierarchy Problem	The force of gravity is on the order of 10^{-43} times weaker than the force of electromagnetism. Physicists consider this large difference to be anomalous., 4
Hyperobject	An m-manifold in n-space open or closed, with or without boundary, where $m \leq n$., 6
Large eXtra Dimensions	At the time of this writing 'large' means larger than the planck length - perhaps on the order of a millimeter., 4

Lightcone	The lightcone is a right circular 4D hypercone whose symmetric axis is collinear with the time axis. It is swept out of 4-space along the time axis by the expanding 3D spherical light wave front., 21
LXD	See Large eXtra Dimensions., 4
Pluck	The intersection of a k-flat with an m-manifold in n-space to extract a k-manifold object, where $codim(k) + codim(m) = codim(k \cap m) = (n-k)+(n-m) = (n-(k \cap m))$ for $k \leq m \leq n$. For example, (4-space - 3-flat) + (4-space - 3-manifold) = 2-manifold. The 3-flat slicer in 4-space should extract a 2-manifold from a 3-manifold embedded in the 4-space. If the original 3-manifold is a closed compact tetrahedral mesh bounding a 4D model, the plucked 2-manifold can be exported as the conventional triangular mesh boundary of a 3D object., 6
Pure simplicial m-complex	A set of aligned non-intersecting simplices in n-space where $m \leq n$, formed by joining contiguous (m-1)-simplices at their shared (m-2)-faces into a closed compact connected (m-1)-surface without boundary., 6

Pure simplicial complex	A generic pure simplicial m-complex where m has not been specified., 6
Relativistic velocity	Used to describe the relative speeds between two reference frames of $0.866c$ or greater. The adjective **relativistic** also describes an object that is moving with a relativistic velocity with respect to the camera frame in which the observer is at rest., 21
Temporal homogeneity	Implies the object's visible elements do not change during the viewing period. Object shape, color, size, attitude, as well as velocity, are constant., 18
Tessellation	An the operation whereby a hyperplane (a prism for example) is tiled with a pattern (a tetrahedron for example) in such a way as to leave no region uncovered. The covering hyper-tiles (tetrahedra) need be neither regular nor congruent., 7
Transversal Intersection	Two regular surfaces S_1 and S_2 intersect transversally if whenever $p \in S_1 \cap S_2$ then $T_p(S_1) \neq T_p(S_2)$,[73] where $T_p(S)$ is the tangent of S at p, 77
UED	See Universal Extra Dimensions., 4

Universal Extra Dimensions	Can be of infinite extent and non-linear - e.g. hyperbolic or exponential., 4

Bibliography

[1] Don V. Black. Visualization of classical and relativistic spacetime geometery. Masters thesis, University of California at Irvine, 2005.

[2] Don V. Black, M. Gopi, Frank Wessel, Renato Pajarola, and Falko Kuester. Visualizing flat spacetime: Viewing optical versus special relativistic effects. *American Journal of Physics*, 75(6):540–545, 2007.

[3] Michio Kaku. *Hyperspace: A Scientific Odyssey Through Parallel Universes, Time Warps, and the 10th Dimension*. Anchor, 1995.

[4] Euclid of Alexandria. *The Elements*. Theon of Alexandria, Hellenistic Alexandria, Greece, circa 300BC.

[5] Arthur Cayley. *The collected mathematical papers of Arthur Cayley [microform]*, chapter Chapters in the Analytic Geometry of (n) Dimensions. University Press, Cambridge [Eng.], 1843.

[6] Hermann Grassmann. *Extension Theory*. American Mathematical Society, 1844.

[7] Bernhard Riemann. On the hypotheses which lie at the bases of geometry. *Nature*, 8:14–17; 36–37, 1873.

[8] H. Minkowski, H.A. Lorentz, A. Einstein, and H. Weyl. *The Principle of Relativity*, chapter V: H. Minkowski. Space and Time. Cologne, 1908. Dover Publications, 1908.

[9] Albert Einstein. *Relativity, The Special and General Theory*, chapter Section 26, The Space-Time Continuum of the Spectral Theory of Relativity Considered as a Euclidean Continuum. Random House, Inc., New York, 1916, 1916.

[10] TH Kaluza. On the unity problem in physics (1921). In Thomas Appelquist, editor, *Modern Kaluza-Klein Theories*. Addison-Wesley Publishing Company, 1921.

[11] Oskar Klein. Quantum theory and five dimensional theory of relativity (1926). In Thomas Appelquist, editor, *Modern Kaluza-Klein Theories*. Addison-Wesley Publishing Company, 1926.

[12] N. Arkani-Hamed, S. Dimopoulos, and G. Dvali. The hierarchy problem and new dimensions at a millimeter. *Phys. Lett. B*, 429:263, 1998.

[13] L. Randall and R. Sundrum. A large mass hierarchy from a small extra dimension. *Phys. Rev. Lett.*, 83:3370, October 1999.

[14] Alexandru Scorpan. *The Wild World of 4-Manifolds*. American Mathematical Society, Providence, Rhode Island), Year = 2005,.

[15] Greg Ferrar. Hypercuber applet. http://www.flowerfire.com/ferrar/java/hypercuber/HyperCuber.html, 1995.

[16] Don V Black. Modification in greg ferrar's hypercuber 4d viewer. http://hyperdimensia.com/viewer4d.html, 2001.

[17] James Terrell. Invisibility of the lorentz contraction. *Physical Review*, 116(4):1041–1045, November 1959.

[18] Roger Penrose. The apparent shape of a relativistically moving sphere. In *Proceedings of the Cambridge Philosophical Society*, volume 55, pages 137–139, 1959.

[19] Olaf Roemer. A demonstration concerning the speed of light. *Philosophical Transactions*, XII(136):893–894, June 1677.

[20] Andrew S. Glassner. *An introduction to ray tracing*. Academic Press Limited, San Diego, CA, 1989.

[21] Andrew Howard. Relativistic ray-tracing: simulating the visual appearance of rapidly moving objects. Technical Report 95/21, The University of Melbourne, July 1995.

[22] D. Kirk and J. Arvo. The ray tracing kernel. In M. Gigante, editor, *Proceedings of Ausgraph 1988*, pages 75–82, ACM, Melbourne, Australia.

[23] P-K Hsiung, R.H. Thibadeau, C.B. Cox, R.H.P. Dunn, M. Wu, and P.A. Olbrich. Wide-band relativistic doppler effect visualization. *Proceedings of the First IEEE Conference on Visualization, 1990*, pages 83–92, October 1990.

[24] D. Weiskopf, U. Kraus, and H. Ruder. Searchlight and doppler effects in the visualization of special relativity: A corrected derivation of the transformation of radiance. *ACM Transactions of Graphics*, 18(3):278–292, July 1999.

[25] Kip S. Thorne Charles W. Misner and John Archibald Wheeler. *Gravitation*. W. H. Freeman and Company, New York, 1973.

[26] A. Michael Noll. A computer technique for displaying n-dimensional hyperobjects. *Communication ACM*, 10(8):469–473, 1963.

[27] A. Michael Noll. Computer-generated three-dimensional movies. *Computers & Automation*, 14(11):20–23, Nov 1965.

[28] Thomas Banchoff and Charles Strauss. Computer animation and the geometry of surfaces in 3- and 4-space. In *Proceedings of the 1978 International Congress of Mathematicians*, 1978.

[29] Don V Black. Realtime 4d hypercube viewing software. CALCOMP, Inc. Prisma Demo Utility, Sep 1985.

[30] Carl Machover and John Dill. "new products". *IEEE Computer Graphics and Applications*, 7(9):72–76, Sept 1987.

[31] C. M. Beshers and S. K. Feiner. Real-time 4d animation on a 3d graphics workstation. In *Proceedings on Graphics interface '88*, pages 1–7, Toronto, Ont., Canada, Canada, 1988. Canadian Information Processing Society.

[32] S. K. Feiner and Clifford Beshers. Worlds within worlds: metaphors for exploring n-dimensional virtual worlds. In *UIST '90: Proceedings of the 3rd annual ACM SIGGRAPH symposium on User interface software and technology*, pages 76–83, New York, NY, USA, 1990. ACM.

[33] David Banks. Interactive manipulation and display of surfaces in four dimensions. In *SI3D '92: Proceedings of the 1992 symposium on Interactive 3D graphics*, pages 197–207, New York, NY, USA, 1992. ACM.

[34] A. Chu, Chi-Wing Fu, A. Hanson, and Pheng-Ann Heng. Gl4d: A gpu-based architecture for interactive 4d visualization. *Visualization and Computer Graphics, IEEE Transactions on*, 15(6):1587–1594, Nov.-Dec. 2009.

[35] Richard Feynman, Robert B. Leighton, and Matthew Sands. *The Feynman Lectures on Physics, Volume I.* Addison-Wesley, 1963.

[36] Richard Feynman, Robert B. Leighton, and Matthew Sands. *The Feynman Lectures on Physics, Volume II.* Addison-Wesley, 1963.

[37] Richard Feynman, Robert B. Leighton, and Matthew Sands. *The Feynman Lectures on Physics, Volume III.* Addison-Wesley, 1963.

[38] David C. Banks. Illumination in diverse codimensions. In *SIGGRAPH '94: Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, pages 327–334, New York, NY, USA, 1994. ACM.

[39] A. J. Hanson and P. A. Heng. Visualizing the fourth dimension using geometry and light. In *Proceedings of Visualization 1991*, pages 321–328. IEEE Computer Society Press, 1991.

[40] A. J. Hanson and P. A. Heng. Illuminating the fourth dimension. *Computer Graphics and Applications*, 12(4):54–62, July 1992.

[41] A. J. Hanson and R. A. Cross. Interactive visualization methods for four dimensions. In *In Proceedings of Visualization 1993*, pages 196–203. IEEE Computer Society Press, 1993.

[42] Andrew J. Hanson, Konstantine I. Ishkov, and Jeff H. Ma. Meshview: Visualizing the fourth dimension. Technical report, Computer Science Department, Indiana University, 1999. http://www.cs.indiana.edu/ hanson/papers/meshview.pdf.

[43] Brian Lonsway. The mistaken dimensionality of cad. *Journal of Architectural Education*, 56:23–25, 2002.

[44] Andrew J. Hanson and Hui Zhang 0006. Multimodal exploration of the fourth dimension. In *IEEE Visualization*, page 34, 2005.

[45] Pak Chung Wong and R. Daniel Bergeron. 30 years of multidimensional multivariate visualization. In *Scientific Visualization, Overviews, Methodologies, and Techniques*, pages 3–33, Washington, DC, USA, 1997. IEEE Computer Society.

[46] J. T. Kajiya and T. L. Kay. Rendering fur with three dimensional textures. *SIGGRAPH Comput. Graph.*, 23(3):271–280, 1989.

[47] A. J. Hanson, Tamara Munzer, and George Francis. Interactive methods for visualizable geometry. *IEEE Computer*, 27(7):73–83, July 1994.

[48] Andrew J. Hanson, Tamara Munzner, and George Francis. Interactive methods for visualizable geometry. *Computer*, 27(7):73–83, 1994.

[49] A. J. Hanson. *Geometry for n-dimensional graphics*, chapter In Paul Heckbert, editor,Graphics Gems IV, pages 149–170. Academic Press,Cambridge, MA, 1994.

[50] Michael D'Zmura, Philippe Colantoni, and Gregory Seyranian. Virtual environments with four or more spatial dimensions. *Presence: Teleoper. Virtual Environ.*, 9(6):616–631, 2000.

[51] M. McGuigan, G. J. Smith, and S. Ohta. Visualization of four-dimensional quantum chromodynamic data. In *In Proceedings S. Klasky and S. Thorpe (Eds.), Visualization Development Environments 2000,Princeton, NJ*, pages 159–164, 2000.

[52] Stanimire Tomov and Michael McGuigan. Interactive visualization of higher dimensional data in a multiview environment. *CoRR*, cs.GR/0405048, 2004.

[53] Niklas Elmqvist and Philippas Tsigas. A taxonomy of 3d occlusion management techniques. *Virtual Reality Conference, VR'07, IEEE*, 0:51–58, march 2007.

[54] Julieta C. Aguilera and Mark U. SubbaRao. Voluble: a space-time diagram of the solar system. volume 6804(1), page 68040A. SPIE, 2008.

[55] Gary Allen (Ed). *Human Spatial Memory: Remembering Where*. Lawrence Erlbaum. Associates, Mahwah, New Jersey, 2004.

[56] Daniel Asimov. The grand tour: a tool for viewing multidimensional data. *SIAM J. Sci. Stat. Comput.*, 6(1):128–143, 1985.

[57] J. Wegman and Jeffrey L. Solka. On some mathematics for visualizing high dimensional data, 2002.

[58] Almir Olivette Artero and Maria Cristina Ferreira de Oliveira. Viz3d: Effective exploratory visualization of large multidimensional data sets. In *SIBGRAPI '04: Proceedings of the Computer Graphics and Image Processing, XVII Brazilian Symposium*, pages 340–347, Washington, DC, USA, 2004. IEEE Computer Society.

[59] Fred P. Brooks Jr. The computer scientist as toolsmith ii. volume 39, pages 61–68, New York, NY, USA, March 1996. ACM.

[60] Don V. Black. Sliced 4d three-sphere - original video. http://www.hypervisualization.com/videos/uci/Figure-4.mov, 2009.

[61] Don V. Black. Sliced 4d three-torus - original video. http://www.hypervisualization.com/videos/uci/Figure-5.mov, 2009.

[62] Andrew J. Hanson, Chi wing Fu, and Eric A. Wernert. Very large scale visualization methods for astrophysical data. In *Data Visualization 2000*, pages 115–124. Springer Verlag, 2000.

[63] Marc Borchers, Martin Falk, Oliver Fechtig, Regine Frank, Frank Grave, Andreas King, Ute Kraus, Thomas Muller, Hans-Peter Nollert, Isabel Rica Mendez, Hanns Ruder, Tobias Schafhitzel, Sonja Schar, Corvin Zahn, and Michael Zatloukal. Explanatory and illustrative visualization of special and general relativity. *IEEE Transactions on Visualization and Computer Graphics*, 12(4):522–534, 2006. Member-Weiskopf, Daniel and Member-Ertl, Thomas.

[64] USA Department of Defense. *MIL-STD-1472F, Human Engineering*, volume MIL-STD-1472F. 1999.

[65] Michael D'Zmura. Navigation in 4-d virtual environments. Spring, 2005, 2005.

[66] John Renze, Todd Rowland, and Eric W. Weisstein. "compact manifold.". 2010.

[67] Maurice G. Kendall. *A Course in the Geometry of n Dimensions*. Charles Griffins & Company Ltd, London, England, 1961.

[68] M A Armstrong. *Basic Topology*. Springer Science+Business Media, 1983.

[69] Victor Guillemain and Alan Pollack. *Differential Topology*, page 30. Princeton-Hall, Inc, Englewood Cliffs, New Jersey, 1974.

[70] Don V. Black. Videos referenced in "phd dissertation ...". http://www.HyperVisualization.com/videos/uci, 2009.

[71] C. Marlin Brown. *Human-computer interface design guidelines*. Intellect Books, Exeter, UK, UK, 1999.

[72] Michael P. Hitchman. *Geometry with an Introduction to Cosmic Topology*. Jones and Bartlett Publishers (Sudbury, Mass), 2008.

[73] Manfredo P. Do Carmo. *Differential Geometry of Curves and Surfaces*. Prentice-Hall, 1976.

[74] Edwin A. Abbot. *Flatland, A romance of many dimensions*. Public Domain, 1884. http://www.GravityWaves.com/reprints/4D_Phil/Flatland/Flatland.html.

[75] Arthur Cayley. *The collected mathematical papers of Arthur Cayley [microform]*. University Press, Cambridge [Eng.], 1889.

[76] Charles Hinton. *The Fourth Dimension [Frontispieces]*. London and New York, 1904.

[77] T. Kaluza. On the unity problem of physics. *Physik.-Mathema Klasse*, pages 966–972, 1921.

[78] Max Born. *Einstein's Theory of Relativity*. Dover Publications, 1962, 1924. Chapter VI, Section 4.

[79] A. R. Forsyth. *Geometry of Four Dimensions*. Cambridge U. Press, 1934, London, England, 1934.

[80] Marsden Morse. *The Calculus of Variations in the Large*. American Mathematical Society, London, England, 1934.

[81] Sir Arthur Eddington. *The Philosophy of Physical Science*. 1939.

[82] Hassler Whitney. The singularities of a smooth n-manifold in (2n- i)-space. *Ann. of Math.*, 45, 1944.

[83] E.A. Abbott. *Flatland*. Dover Publications, Inc., 1952.

[84] S. Hilbert, D.and Cohn-Vossen. *Geometry and the Imagination*. Chelsea, New York, 1952.

[85] J. Semple and G. Kneebone. *Algebraic Projective Geometry*. Clarendon Press, Oxford, 1952.

[86] D.M.Y. Sommerville.

[87] John A. Wheeler. *Logic, Methodology and Philosophy of Science*, chapter Curved Empty Space-Time as the Building Material of the Physical World: An Assessment. Stanford University Press, 1962.

[88] J. Levine. Imbedding and immersion of real projective spaces. *Proc. Amer. Math. Sot.*, 14, 1963.

[89] John Milnor. *Morse Theory*. Princeton University Press, 1963.

[90] A. Cobham. The intrinsic computational difficulty of functions.

[91] J. Hartmanus and R. E. Stearns. On the computational complexity of algorithms. In *Transactions of the American Mathematical Society*, volume 117, pages 285–306, May 1965.

[92] Barrett O'Neil. *Elementary Differential Geometry*. Academic Press, New York, 1966.

[93] A. Michael Noll. A computer technique for displaying n-dimensional hyperobjects. *Commun. ACM*, 10(8):469–473, 1967.

[94] Andrew J. Hanson. Dual n-point functions in pgl(n-2,c)-invariant formalism. *Phys. Rev.*, D5:1948–1956, 1972.

[95] Steven Weinberg. *Gravitation and Cosmology: Principles and Applications of the General Theory of Relativity*. John Wiley & Sons, New York, 1973.

[96] M. Henon. A two dimensional mapping with a strange attractor. *Comm. in Mathematical Physics*, 50(1):69–77, 1976.

[97] Peter Tanner Kenneth Evans and Marceli Wein. Tablet-based valuators that provide one, two, or three degrees of freedom. In *SIGGRAPH '8I Proc*, pages 91–97. SIGGRAPH, 1980.

[98] Greg Abram Henry Fuchs and Eric Grant. Near real-time shaded display of rigid objects. In *SIGGRAPH '83 Proc*, pages 65–69. SIGGRAPH, 1980.

[99] Turner Whitted. I. Lane, Loren Carpenter and Jim Blinn. Scan line methods for displaying parametrically defined surfaces. *Communications of the ACM*, 23(1):23–34, Jan 1980.

[100] Michael Potmesil and Indranil Chakravarty. Synthetic image generation with a lens and aperture camera model. *ACM Transactions on Graphics*, Apr 1982.

[101] Dino Schweitzer and Elizabeth Cobb. Scanline rendering of parametric surfaces. In *SIGGRAPH '82 Proc*, pages 265–271. SIGGRAPH, 1982.

[102] William Armstrong and Robert Burton. Perception cues for n dimensions. *Computer Graphics World*, pages l1–28, May 1985.

[103] T. F. Banchoff. Visualizing two-dimensional phenomena in fourdimensional space: A computer graphics approach. In E. Wegman and D. Priest, editors, *Statistical Image Processing and Computer Graphics*, pages 187–202, New York, 1985. Marcel Dekker, Inc.

[104] Eric Bier. Skitters and jacks: Interactive 3d positioning tools, 1985.

[105] Thomas Banchoff Htiseyin Kocak, Frederic Bisshopp and David Laidlaw. Topology and mechanics with computer graphics: Linear hamiltonian systems in four dimensions. *Advances in Applied Mathematics*, 7:282–308, 1985.

[106] Gregory Nielson and Dan Olsen. Direct manipulation techniques for 3d objects using 2d locator devices, 1985.

[107] K. Shoemake. Animating rotation with quaternion curves. In *Computer Graphics*, volume 19, pages 245–254. Proc. Siggraph, ACM Press, 1985.

[108] Jeffrey R. Weeks. *The shape of space : how to visualize surfaces and three-dimensional manifolds*. M. Dekker, New York:, 1985.

[109] R. P. Burton S. A. Carey and D. M. Campbell. Shades of a higher dimension. *Computer Graphics World*, pages 93–94, Oct 1987.

[110] G. K. Francis. *A Topological Picturebook*. Springer Verlag, 1987.

[111] Teresa McBennett Susan Gauch. Rich Hammer, Dals Krams and Dabby Saltzman. An evaluation of factors affecting rotation tasks in a three-dimensional graphics system. *TR87-002 Dept Comp. Sci.*, TR87-002, 1987.

[112] K. V. Steiner and R. P. Burton. Hidden volumes: The 4th dimension. *Computer Graphics World*, pages 71–74, Feb 1987.

[113] Michael Chen, S. Joy Mountford, and Abigail Sellen. A study in interactive 3-d rotation using 2-d control devices. In *SIGGRAPH '88: Proceedings of the 15th annual conference on Computer graphics and interactive techniques*, pages 121–129, New York, NY, USA, 1988. ACM.

[114] Matthew Moore and Jane Wilhelms. Collision detection and response for computer animation. In *SIGGRAPH '88: Proceedings of the 15th annual conference on Computer graphics and interactive techniques*, pages 289–298, New York, NY, USA, 1988. ACM.

[115] Robert Burton. Raster algorithms for cartesian hyperspace graphics. *Journal of Imaging Technology*, 15(2):89–95, Apt 1989.

[116] Henry Fuchs ef al. Pixel-planes 5: A heterogeneous multiprocessor graphics system using processor-enhanced memories. In *SIGGRAPH '89 Proc.*, pages 79–88, New York, NY, USA, 1989. ACM.

[117] A. S. Glassner. How to derive a spectrum from an rgb triplet. *IEEE Computer Graphics and Applications*, 9(4):95-99, July 1989.

[118] Ivars Peterson. A different dimension. *Science News*, 135(21).

[119] T. F. Banchoff. Beyond the third dimension: Geometry, computer graphics, and higher dimensions. *Scientific American*, 1990.

[120] David Baraff. Curved surfaces and coherence for nonpenetrating rigid body simulation. In *SIGGRAPH '90 Proc.*, pages 19–28, New York, 1990. ACM.

[121] David Ellsworth. Howard Good and Brice Tcbbs. Distributing display lists on a multicomputer. In *Proc. 1990 Symp Interactive 3D Graphics*, 1990.

[122] S. K. Feiner and Clifford Beshers. Visualizing n-dimensional virtual worlds with n-vision. *SIGGRAPH Comput. Graph.*, 24(2):37–38, 1990.

[123] Paul Haeberli and Kurt Akely. The accumulation buffer: Hardware support for high-quality rendering. In *SIGGRAPH '90 Proc.*, pages 309–318, New York, 1990. ACM.

[124] Pat Hanrahan and Paul Hacberli. Direct wysiwyg painting and texturing on 3d shapes. In *SIGGRAPH '90 Proc.*, pages 215–224, New York, 1990. ACM.

[125] A. J. Hanson, P. A. Heng, and B. C. Kaplan. Techniques for visualizing fermat's last theorem: A case study. In *Proceedings of Visualization 1990*, pages 97–106. IEEE Computer Society Press,San Francisco, 1990.

[126] A. J. Hanson, P. A. Heng, and B. C. Kaplan. Visualizing fermat's last theorem, 1990. 3:37 minute video animation.

[127] P.-K. Hsiung, R.H. Thibadeau, and M. Wu. T-buffer: Fast visualization of relativistic effects in spacetime. In *Computer Graphics,1990 Symposium on Interactive 3D Graphics*, volume 24(2), pages 83–88. SIGGRAPH, ACM, 1990.

[128] P. Shirley and A. Tuchman. A polygonal approximation to direct scalar volume rendering. In *SIGGRAPH '90 Proc.*, volume 24, pages 63–70, New York, 1990. ACM.

[129] E. Bedford and J. Smillie. Polynomial diffeomorphisms of c2: Currents, equilibrium measure and hyperbolicity. *Inventiones Mathematicae*, 103:69–99, 1991.

[130] Edward Adelson William Freeman and David Heegar. Motion without movement. In *Proc. 1991 Symp Interactive 3D Graphics*, pages 27–30, 1991.

[131] C. M. Hoffmann and J. Zhou. Some techniques for visualizing surfaces in four-dimensional space. *Computer Aided Design*, 23:83–91, 1991.

[132] S. Hollasch. Four-space visualization of 4d objects. Masters thesis, Arizona State University, 1991.

[133] Don Mitchell. Spectrally optimal sampling for distribution ray tracing. In *SIGGRAPH '91 Proc.*, pages 157–164, New York, NY, USA, 1991. ACM.

[134] Jarke van Wijk. Spot noise-texture synthesis for data visualization. In *SIGGRAPH '91 PROC.*, pages 309–318, New York, NY, USA, 1991. ACM.

[135] J.E. Fornaess and N. Sibony. Complex henon mappings in c2 and fatou bieberbach domains. *Duke Mathematical J.*, 65:345–380, 1992.

[136] A. J. Hanson and P. A. Heng. Four-dimensional views of 3d scalar fields. In *In Proceedings of Visualization 1992*, pages 84– 91. IEEE Computer Society Press, 1992.

[137] Andrew J. Hanson. Video animation of knotted spheres in four dimensions, 1993.

[138] Robert A. Cross and Andrew J. Hanson. Virtual reality performance for virtual geometry. In *VIS '94: Proceedings of the conference on Visualization '94*, pages 156–163, Los Alamitos, CA, USA, 1994. IEEE Computer Society Press.

[139] Rodney G. Downey and Michael R. Fellows. Parameterized computational feasibility. In *Feasible Mathematics II*, pages 219–244. Birkhauser, 1994.

[140] K. L. Duffin and W. A. Barrett. Spiders: a new user interface for rotation and visualization of n-dimensional point sets. In *In Proceedings of Visualization 1994*, pages 205–211. IEEE Computer Society Press, 1994.

[141] A.J. Hanson. A construction for computer visualization of certain complex curves. *Notices of the Amer.Math.Soc.*, 41(9):1156–1163, November and December 1994.

[142] R. A. Cross and A. J. Hanson. Virtual reality performance for virtual geometry. In *In Proceedings of Visualization 1994*, pages 156–163. IEEE Computer Society Press, 1994.

[143] Paul S. Heckbert and Ed Andrew Glassner. *Graphics gems IV*. Academic Press.

[144] A. J. Hanson and H. Ma. Quaternion frame approach to streamline visualization. *IEEE Trans. on Visualiz. and Comp. Graphics*, 1(2):164–174, June 1995.

[145] A. J. Hanson. *Rotations for n-dimensional graphics*, chapter In Alan Paeth, editor,Graphics Gems V, pages 55–64. Academic Press, Cambridge, MA, 1995.

[146] A. J. Hanson and H. Ma. Space walking. In *In Proceedings of Visualization*, pages 126–133. IEEE Computer Society Press, 1995.

[147] W. Chen. Y. Zhou and Z. Tang. An elaborate ambiguity detection method for constructing isosurfaces within tetrahedral meshes. *Computers & Graphics*, 19:355–364, 1995.

[148] Moving coordinate frames for representation and visualization in four dimensions. *Computers & Graphics*, 20(6", pages = 905-919, year = 1996, note = Medical Visualization, issn = 0097-8493, doi = DOI: 10.1016/S0097-8493(96)00060-X, url = http://www.sciencedirect.com/science/article/B6TYG-3VTK1P3-F/2/0f525fdc80b03fbdb60ade3c08120960, author = R. Egli and C. Petit and N.F. Stewart).

[149] Kenneth Krane. *Modern Physics*. John Wiley & Sons, 1996.

[150] Chris Weigle and David C. Banks. Complex-valued contour meshing. In *In Proceedings of Visualization '96*, pages 173–180. IEEE Computer Society Press, 1996.

[151] David C. Banks. Screen-parallel calculation of surface intersections, 1997.

[152] A. J. Hanson and E.Wernert. Constrained 3d navigation with 2d controllers. In *Proceedings of Visualization*, pages 175–182. IEEE Computer Society Press, 1997.

[153] C. M. Savage and Antony C. Searle. Visualizing special relativity. http://www.anu.edu.au/Physics/Searle/, 1997. on-line video.

[154] Antony C. Searle. Backlight ray tracer, 1997. http://www.anu.edu.au/Physics/Searle/.

[155] Antony C. Searle. Seeing relativity, 1997. http://www.anu.edu.au/Physics/Searle/.

[156] Daniel Weiskopf. Institute for visualization and interactive systems, virtual reality software download, 1997. http://wwwvis.informatik.uni-stuttgart.de/eng/research/fields/current/relativity/specialrelativity/vr/vr-eng.html.

[157] C. Bajaj, V. Pascucci, G. Rabbiolo, and D. Schikore. Hypervolume visualization: a challenge in simplicity. In *In Proceedings of 1998 Symposium on Volume Visualization*, pages 95–102. ACM Press, 1998.

[158] Y.F. Gong et al. Recovering strange attractors from noisy interspike intervals of neuronal firings. *Physics Letters A*, 258:253–262, 1999.

[159] A. J. Hanson. Constrained optimal framings of curves and surfaces using quaternion gauss maps. In *Proceedings of Visualization*, pages 375–382. IEEE Computer Society Press, 1998.

[160] Steve A. Hill, Jonathan C. Roberts, Local Cell Tilers, England Uk, and England Uk. Generating surface geometry in higher dimensions using local cell tilers, 1998.

[161] A. Johnson. Ray tracing the complex henon map. Technical report, Univ. of Kansas, 1998.

[162] R. Rau, D. Weiskopf, and H. Ruder. Special relativity in virtual reality, mathematical visualization. In K. Polthier H.-C. Hege, editor, *Visualization and Mathematics*, pages 269–279. Springer-Verlag, 1998.

[163] C.M. Savage and A.C. Searle. Visualizing special relativity, 1998.

[164] Chris Weigle and David C. Banks. Extracting iso-valued features in 4-dimensional scalar fields. *Volume Visualization and Graphics, IEEE Symposium on*, 0:103–110, 1998.

[165] L. Cao. Coexisting attracting basins in complex holomorphic dynamics. Technical report, Univ. of Kansas, 1999. Dept. of Mathematics.

[166] Ulrike Stege Rodney G. Downey, Michael R. Fellows. Computational tractability: The view from mars. In *Bulletin of the European Association for Theoretical Computer Science*, volume 69, pages 73–97),, 1999.

[167] Bruce Edmonds. *The Evolution of Complexity*, chapter What is Complexity?-The Philosophy of Complexity Per Se With Application to Some Examples in Evolution. Kluwer Academic Publishers, 1999.

[168] Michael D'Zmura, Phillipe Colantoni, and Gregory Seyranian. Virtual environments with four or more spatial dimensions. *Presence*, 1999.

[169] Jonathan C. Roberts and Steve Hill. Piecewise linear hypersurfaces using the marching cubes algorithm, 1999.

[170] D. Weiskopf. A texture mapping approach for the visualization of special relativity. In H. Hagen A. Varshney, C.M. Wittenbrink, editor, *IEEE Visualization '99 Late Breaking Hot Topics*, pages 41–44. ACM Press, October 1999.

[171] D. Weiskopf. An immersive virtual environment for special relativity. Technical Report SFB 382 - Report 108, University of Tbingen, January 1999.

[172] Eric A. Wernert and Andrew J. Hanson. A framework for assisted exploration with collaboration. In *In Proceedings of the IEEE Conference on Visualization*, pages 241–248. IEEE Computer Society Press, 1999.

[173] R.Wenger P. Bhaniramka and R. Crawfis. Isosurfacing in higher dimensions. In *Proceedings of Visualization*, pages 267–273. IEEE Computer Society Press, 2000.

[174] Michael D'Zmura, Philippe Colantoni, and Gregory D. Seyranian. Visualization of events from arbitrary spacetime perspectives. volume 3960(1), pages 35–40. SPIE, 2000.

[175] S. Fang and H. Chen. Hardware accelerated voxelization. *Computers & Graphics*, 24(3):433–442, 2000.

[176] J.R. Miller E.A. Gavosto and J. Sheu. Immersive 4d visualization of complex dynamics. In *Immersive 4D Visualization of Complex Dynamics," Proc. Workshop New Paradigms in Information Visualization and Manipulation (NPIV 98)*, pages 62–64. ACM, 2000.

[177] Penny Rheingans and Marie Desjardins. Visualizing high-dimensional predictive model quality. In *In Proceedings of IEEE Visualization 2000*, pages 493–496, 2000.

[178] A. Tufaile and J.C. Sartorelli. Hnon-like attractor in air bubble formation. *Physics Letters A*, 275:211–217, 2000.

[179] Daniel Weiskopf. daniel weiskopf - homepage, 2000. http://wwwvis.informatik.uni-stuttgart.de/ weiskopf/.

[180] D. Weiskopf, D. Kobras, and H. Ruder. An image-based approach to special relativistic rendering. Technical Report SFB 382 - Report 145, University of Tbingen, March 2000.

[181] D. Weiskopf, D. Kobras, and H.Ruder. Real-world relativity: Image-based special relativistic visualization. In *IEEE Visualization 2000 Conference Proceedings*, pages 445–448, October 2000.

[182] D. Weiskopf and M. Ansorg. Visualization of the general relativistic rigidly rotating disk of dust. *Annalen der Physik*, 9 (2000) Spec. Issue:179–185, 2000.

[183] D. Weiskopf, U. Kraus, and H. Ruder. Illumination and acceleration in the visualization of special relativity: A comment on fast rendering of relativistic objects. *The Journal of Visualization and Computer Animation*, 11(4):185–195, 2000.

[184] D. Weiskopf. An immersive virtual environment for special relativity. In *WSCG Conference Proceedings, V. Skala (Ed)*, pages 337–344. University of West Bohemia, Pilsen, February 2000.

[185] D. Weiskopf. Fast visualization of special relativistic effects on geometry and illumination. In R. van Liere W. de Leeuw, editor, *Proceedings of the EG/IEEE TCVG Symposium on Visualization*, pages 219–228. Springer, 2000.

[186] D. Weiskopf. Four-dimensional non-linear ray tracing as a visualization tool for gravitational physics. In A. Varshney T. Ertl, B. Hamann, editor, *IEEE Visualization 2000 Proceedings*, pages 445–448. ACM Press, 2000.

[187] D. Weiskopf. Non-linear ray tracing as a visualization tool for gravitational physics. In *Proceedings of the IEEE Visualization 2000 Conference*, pages 445–448, 2000.

[188] H. Ruder D. Weiskopf, D. Kobras. Real-world relativity: Image-based special relativistic visualization. In *IEEE Visualization 2000 Proceedings, T. Ertl, B. Hamann, A. Varshney (eds.)*, pages 303–310. ACM Press, October 2000.

[189] J. Diepstraten, D.Weiskopf, and T. Ertl. Minkrelvis v0.2b software package, 2001. http://www.vis.uni-stuttgart.de/relativity/minkowski/.

[190] M. D'Zmura, P. Colantoni, and G. Seyranian. Virtual environments with four or more spatial dimensions. *Presence 9*, pages 616–631, 2001.

[191] C. Everitt. Interactive order-independent transparency, 2001.

[192] A. J. Hanson and D. Weiskopf. Course 15: Visualizing relativity. In *SIGGRAPH 2001 Tutorials*. ACM Press, 2001.

[193] D. Kobras, D. Weiskopf, and H. Ruder. Image-based rendering and general relativity. In *WSCG Conference Proceedings, V. Skala (ed.)*, pages 130–137, February 2001.

[194] M. J. Kligard E. Lindholm and H. Moreton. A user-programmable vertex engine. In *SIGGRAPH 2001*, pages 149–158.

[195] Alex Pang. Visualizing uncertainty in geo-spatial data. In *In Proceedings of the Workshop on the Intersections between Geospatial Information and Information Technology*, 2001.

[196] S. Tzvetkov K. Proudfoot, W. R. Mark and P. Hanrahan. A real-time procedural shading system for programmable graphics hardware. In *SIGGRAPH 2001*, pages 159–170.

[197] Daniel Weiskopf. *Visualization of Four-Dimensional Spacetimes*. Ph.D. dissertation, University of Tübingen, 2001.

[198] Don V. Black. Raytracing special relativity with the searle backlight software package. http://www.HyperVisualization.com/videos/black, 2002.

[199] J. Diepstraten, D. Weiskopf, and T. Ertl. Automatic generation and non-photorealistic rendering of 2+1d minkowski diagrams. In V. Skala, editor, *WSCG Conference Proceedings*, pages 139–146, University of West Bohemia, Pilsen, February 2002.

[200] Alfred Inselberg. Multidimensional visualization and it's applications. In *SIGGRAPH 2002 Tutorials*. ACM Press, 2002.

[201] D. Kobras, D. Weiskopf, and H. Ruder. General relativistic image-based rendering. *The Visual Computer*, 18(4):250–258, 2002.

[202] N. Neophytou and K. Mueller. Space-time points: 4d splatting on efficient grids. In *In Proc. of IEEE Symposium on Volume Visualization and Graphics*, pages 97–106, 2002.

[203] J.G. Shannonhouse. A framework for building high performance computing environments. Masters thesis, Univ. of Kansas, 2002.

[204] M. D'Zmura and M. Ge. 4d structure from motion: a computational algorithm. *Computational Imaging (Proc SPIE/IS&T)*, 5016:13–23, 2003.

[205] Michael D'Zmura. The 4d web page, 2003. http://www.vrlab.uci.edu/dzmura/4D/.

[206] Sanjeev Seahra. Physics in higher-dimensional manifolds. Masters thesis, University of Waterloo, 2003.

[207] Miller and Gavosto. The immersive visualization probe for exploring n-dimensional spaces. *IEEE Comp. Graph. and App.*, 24:76–85, 2004.

[208] Catherine Plaisant. The challenge of information visualization evaluation. In *AVI '04: Proceedings of the working conference on Advanced visual interfaces*, pages 109–116, New York, NY, USA, 2004. ACM.

[209] A. J. Hanson and D. Weiskopf. Course 15: Visualizing relativity. In *SIGGRAPH 2001 Tutorials*. ACM Press, 2005.

[210] A. J. Hanson. Course notes: Visualizing quaternions. In *SIGGRAPH 2005 Tutorials*. ACM Press, 2005.

[211] Jeffrey R. Spies, Jeffrey R. Spies, and Jeffrey R. Spies. Hdtreev: a multidimensional visualization tool, 2005.

[212] Daniel Weiskopf. T.: Particle and texture based spatiotemporal visualization of time-dependent vector fields. In *In VIS '05: Proceedings of the 8th conference on Visualization '05 (2005), IEEE Computer*, pages 647–654. Society Press, 2005.

[213] Daniel Weiskopf, Marc Borchers, Thomas Ertl, Martin Falk, Oliver Fechtig, Regine Frank, Frank Grave, Andreas King, Ute Kraus, Thomas Muller, Hans-Peter Nollert, Isabel Rica Mendez, Hanns Ruder, Corvin Zahn, Michael Zatloukal, Tobias Schafhitzel, and Sonja Schar. Visualization in the einstein year 2005: A case study on explanatory and illustrative visualization of relativity and astrophysics. *Visualization Conference, IEEE*, 0:74, 2005.

[214] Cheng-Chih Yang, Cheng-Chieh Chiang, Yi-Ping Hung, and Greg C. Lee. Visualization for high-dimensional data: Vishd. In *IV '05: Proceedings of the Ninth International Conference on Information Visualisation*, pages 692–696, Washington, DC, USA, 2005. IEEE Computer Society.

[215] Marc Borchers, Martin Falk, Oliver Fechtig, Regine Frank, Frank Grave, Andreas King, Ute Kraus, Thomas Muller, Hans-Peter Nollert, Isabel Rica Mendez, Hanns Ruder, Tobias Schafhitzel, Sonja Schar, Corvin Zahn, and Michael Zatloukal. Explanatory and illustrative visualization of special and general relativity. *IEEE Transactions on Visualization and Computer Graphics*, 12(4):522–534, 2006. Member-Weiskopf, Daniel and Member-Ertl, Thomas.

[216] Kedar Bhumkar. Interactive visualization of hyperbolic geometry using the weierstrass model. Masters thesis, University of Minnesota, 2006.

[217] E. Eisemann and X. Decoret. Fast scene voxelization and applications. In *In Proc. of the 2006 symposium on Interactive 3D graphics and games*, pages 71–78, 2006.

[218] Andrew J. Hanson and Ji-Ping Sha. A contour integral representation for the dual five-point function and a symmetry of the genus four surface in r6. *MATH.GEN.*, 39:2509, 2006.

[219] Ben Shneiderman and Catherine Plaisant. Strategies for evaluating information visualization tools: multi-dimensional in-depth long-term case studies. In *BELIV '06: Proceedings of the 2006 AVI workshop on BEyond time and errors*, pages 1–7, New York, NY, USA, 2006. ACM.

[220] Hanan Samet. *Foundations of Multidimensional and Metric Data Structures*. Morgan Kaufman, Oxford, UK, 2006.

[221] Jarke J. van Wijk. Views on visualization. *IEEE Transactions on Visualization and Computer Graphics*, 12(4):421–433, 2006.

[222] P. Brown and B. Lichtenbelt. Ext geometry shader4 extension specification, 2007.

[223] Ivan Hip. Interactive visualization package for 4d lattice field theories, 2007.

[224] I. Llamas. Real-time voxelization of triangle meshes on the gpu. In *In ACM SIGGRAPH 2007 sketches*, page 18, New York, NY, USA, 2007. ACM.

[225] Brian Fisher Daniel Keim David Laidlaw Silvia Miksch Klaus Mueller William Ribarsky Bernhard Preim Anders Ynnerman Joerg Meyer, Stephan Diehl. From visualization to visually enabled reasoning. *Scientific Visualization*.

[226] Harri Siirtola. *Interactive Visualization of Multidimensional Data*. PhD thesis, University of Tampere, apr 2007.

[227] Sidharth Thakur and Andrew Hanson. A framework for exploring high-dimensional geometry. *Lecture Notes in Computer Graphics*, (4841):804–815, 2007.

[228] Hui Zhang and Andrew Hanson. Shadow-driven 4d haptic visualization. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1688–1695, 2007.

[229] L. Bavoil and K. Myers. Order independent transparency with dual depth peeling, 2008.

[230] A. J. Hanson and J.-P. Sha. A tessellation for algebraic surfaces in CP3. *ArXiv e-prints*, April 2008.

[231] Colin B. Macdonald and Steven J. Ruuth. Level set equations on surfaces via the closest point method. *J. Sci. Comput.*, 35(2-3):219–240, 2008.

[232] Hsin-Chia Cheng. 2009 tasi lecture-introduction to extra dimensions, 2009.