Univel Particle Transport from Nonunivel Geometries with Juniper

by

Aaron D. Hall

A thesis proposal submitted in partial fulfillment
of the comprehensive exam requirements for the degree

of

Doctor of Philosophy

in

Computer Science

MONTANA STATE UNIVERSITY
Bozeman, Montana

April 2009

PROJECT SUMMARY

The problem of this study is to evaluate univel-based neutral particle transport simulation where the univels must be rasterized from non-univel geometries, especially as applied to the issue of designing active scanners for applications such as radioactive contraband smuggling, and with consideration for unstructured mesh geometries for multiphysics interfacing.

The specific objectives of this study are: to implement particle transport simulation with a univel-based geometric representation, supporting the extended reactions and material types necessary in evaluating active scanner designs for detecting smuggled nuclear material; to construct mesh representations of input geometries for use by outside tools for other physical simulations, supporting multiphysics problems like Gen-4 reactor design; and to evaluate the effectiveness and cost tradeoffs of antialiased univelization and transport.

Univel geometries have proven benefits for particle transport simulation; extending their reach to rasterizing non-univel input is a new approach that advances the general problem. In particular antialiased rasterization with high-dimensional vectors is a novel, untested approach. Advancement here has broader implications for applications like the above-mentioned active scanner development and Gen-4 nuclear reactor design.

Important literature topics that provide pertinent background include the particle transport problem and physics issues, the relationship of computational geometry and computer graphics to physics simulation, raster geometry in voxels and univels, raster antialiasing, vector quantization—algorithms and high-dimensional spatial indexing, mesh geometries and boundary evaluation and merging, data representations and file formats, and lastly the use of Java in scientific computing.

This study's approach is to develop and evaluate Juniper, a Monte-Carlo particle transport program, using a univelized geometry representation and written in Java. Its components provide core transport functionality, translation from MCNP input, an improved input representation, rasterization of geometries to univel form, including antialiased rasterization, and mesh generation. Once these components are sufficiently functional the study will validate and evaluate Juniper's performance against MCNP, and evaluate antialiased versus aliased univel performance.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1

INTRODUCTION AND PROBLEM

Introduction and Background

In the mid-20th century interest in the problem of subatomic particle transport physics expanded alongside technological changes. The invention of nuclear reactors raised practical particle transport problems with geometries far more complex than had been previously examined. While problems with the simplest geometries have analytic solutions, these required numerical methods. The difficulty of calculating those methods would have been overwhelming but for another timely invention, the computer.

The most prominent numerical method for solving particle transport problems is Monte Carlo. In it virtual particles are projected through the problem space. These analogues of real particles are treated as pointlike, traveling with constant velocity and energy between collisions, and colliding only with materials in the problem space, not other particles. With simplifying assumptions like these the probabilities of the various possible collisions and subsequent reactions a particle might experience are assembled in a set of cross-section data. These cross-sections can be very complex as they depend on the particle type and energy, and on the nuclides comprising each material type. Pseudorandom numbers are used to sample the probability distributions in the cross-section data to determine each virtual particle's initial state, the

distance to its next collision, and the details of the reaction at that collision. This iterates until the particle is absorbed at a collision or leaves the problem space, whereupon the process repeats, often for millions of particles. By measuring the desired properties—such as flux through a particular object—of these projected particles, the method creates a statistical sample of the true values that would result from the complete physics of the problem.

MCN, the Monte Carlo Neutron transport program, was created at Los Alamos in 1967. Its successor, MCNP, is one of the most widely used particle transport simulators in the field. Its input file format reflects that age: organized around a sequence of 80-column (punch) cards, it uses a terse language of mnemonics, specially-interpreted columns, and exceptional cases largely to work around the limitations of that format. It also represents its problem geometries with polynomial surface functions. Tracking particles through this type of geometry suffers the problem of expensive particle-surface intersection calculations. This cost can be alleviated by substituting a discrete geometry made of a regular grid of rectangular elements, univels. The Minerva radiotherapy transport program uses such a model based on the natural univelization of three-dimensional MRI or CT scan data. Where other physical processes such as heat flow are to be simulated along with radiation transport, neither of these geometry representations is appropriate. Rather, a polygon boundary mesh and then tetrahedral volumetric mesh are necessary for common finite element method (FEM) programs.

⋆ ⋆ ⋆

The problem of detecting smuggled nuclear material is an ongoing one. Existing detectors rely on the materials' naturally-emitted radiation, and can be foiled by methods like shielding. An active scanner would project neutron radiation into the scanned object (such as an ITU shipping container) and detect the induced radiation signature (Figure 1). Developing such a detector requires building a large library of radiation signatures through both experiments and simulations. A more efficient particle transport simulator would help this process; the Minerva approach is a natural fit. The next generation of nuclear powerplant reactor design, "Gen 4", is anticipated to yield improvements in safety, operating cost, waste generation, and antiproliferation issues. There is also interest in supporting high-temperature hydrogen cogeneration to contribute to a "hydrogen economy". Although more efficient particle transport simulation would also benefit this application, reactor problems are complicated by the aforementioned need to support multiple physics domains such as mechanical stress and heat flow.

⋆ ⋆ ⋆

Aliasing is a fundamental problem in rasterization, as in any form of pulse-code modulation. Signal features with frequencies beyond the Nyquist limit of the sampling density become distorted. This includes the rasterization needed to convert an input geometric model in a CSG form to a univel representation, where high-frequency features are found in small objects and sharp edges. Two-dimensional raster antialiasing

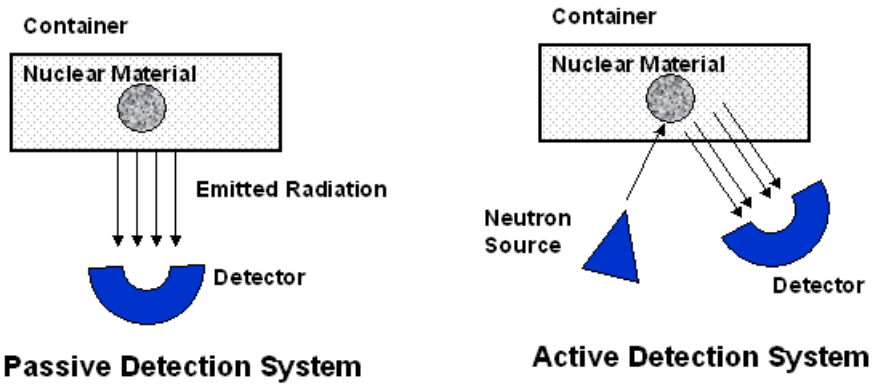

Figure 1. Nuclear material detectors.

is a well-studied problem in the world of computer graphics. There antialiasing filters remove high-frequency structure and combine three-dimensional color vectors into blended colors. When memory capacity is limited, those colors may then be quantized to a limited palette. An antialiased univelization proceeds analagously, though blending material vectors which may have more than a hundred dimensions. Memory limitations and the nature of the transport algorithm call for similar quantization to a material palette, a more difficult problem in that high a dimensionality, although high-dimensional quantization has appeared in speech compression and image-similarity applications.

Converting a CSG or surface function geometry representation into a mesh geometry is known as boundary evaluation and merging. Boundary evaluation, the simpler of the stages, converts primitive objects or surfaces into a mesh form through type-specific conversions. Merging uses mesh equivalents to CSG Boolean combining

operators to combine submeshes into the eventual world model. Once this surface mesh is complete, it can be converted to a volumetric mesh of Delaunay tetrahedra by adding interior Steiner points, forming a model appropriate for other physics domains mentioned above.

Statement of the Problem

Univel geometries have proven efficiency benefits for particle transport simulation where the input geometry is itself inherently raster, benefits that might be extensible to other geometry representations. Therefore, the problem of this study is to evaluate univel-based neutral particle transport simulation where the univels must be rasterized from non-univel geometries, especially as applying to the issue of designing active scanners for applications such as radioactive contraband smuggling, and with consideration for unstructured mesh geometries for multiphysics interfacing.

Purpose

The specific objectives of this study are

- to implement general neutral-particle transport simulation, with a univel-based geometric model representation, including support for (γ, n) and $(\gamma, \text{fission})$ reactions, and the broad library of material types, necessary for evaluating neutron- and γ-photon- based active scanner designs for detecting fissile nuclear material smuggled inside ITU cargo shipping containers;

- to construct PLC mesh representations of input geometries in parallel with univelized representations, suitable for use by outside tools for other physical simulations, to support multiphysics evaluation of problems such as Gen-4 nuclear reactor design;
- to develop a translation-friendly input file format, one that faithfully supports translation of MCNP input files to enable cross-validation between MCNP and Juniper, but more human-readable to improve flexibility, and more machine-readable with better structure and fewer special cases, using graph-based geometry and material specifications to aid factoring and reuse of common subunits;
- to support antialiased univelization and transport on antialiased geometric models, and to evaluate the tradeoffs among effectiveness, computational cost, and ultimately the utility of this approach.

Conceptual Framework

Although an early inspiration for a univel geometric model in particle transport was the inherently raster nature of patient scan data, it brings additional efficiency benefits making it attractive for other problem types. In fact for a NURBS-based geometric model 65% to 90% of particle tracking time is spent calculating particle-NURBS intersections. [1] With a univel geometry particle tracking can instead be calculated with an algorithm analogous to the Bresenham line-drawing algorithm, using mostly integer arithmetic for a significant speed improvement. A second benefit

is the indifference of the transport algorithm to model complexity—all univel models of the same size have the same transport cost, a property not true of a surface-function model where adding complexity adds intersection calculations. This is equally true for an antialiasing extension: the more complicated rasterization is more costly up front, but it doesn't add to the transport cost.

In rasterization fine features, with a spatial frequency on the order of the univelization sampling frequency or higher, are distorted by aliasing. The impact of this on the veracity of transport calculations is unknown. Antialiasing techniques, well-established in domains like computer graphics, are often used to mitigate the negative consequences of aliasing. One approach, prefiltering, eliminates high-frequency components of the signal prior to sampling. This might be accomplished spatially by "blurring" the signal image or objects, or by blurring the samples by using weighted-area sampling rather than point sampling. Another approach, postfiltering, typically uses supersampling to combine multiple point samples into one result pixel. In recursive subdivision supersampling the sampled pixel area is subdivided into, for example, four subpixels, each of which is sampled directly or subdivided recursively until some limit, and the multiple subsamples combine into an overall value. Stochastic supersampling randomly selects a certain number of point subsamples within an area around the pixel and combines those into the overall value.

Non-antialiased univel data consists of a regular grid of indices into a table of material data. When antialiased, each univel may potentially be a unique material

vector of over a hundred dimensions, requiring quantization to reduce back to index/table form. The classic algorithm for vector quantization is due to Lloyd [2], constructing an inital codebook of representative vectors, then iteratively refining those code vectors based on which signal vectors quantize to them (their "support"), in a manner reminiscent of the k-means clustering algorithm. The algorithm relies on nearest-neighbor queries of the vector space, which is difficult to perform efficiently on high-dimensional vectors. Spatial indices that support such queries on high-dimensional data is an area of ongoing research.

Computational physics methods on discretized geometries divide geometric representations into two categories, structured meshes and unstructured meshes. A univelized model, with its regular Cartesian cells, is considered a structured mesh. For particle transport univels are appropriate, but they do distort certain properties of the geometry that makes them unsuitable for certain other physics domains. This particularly includes transforming smooth boundaries into jagged ones, which changes the surface areas and normals of objects. An unstructured mesh representation, using an arbitrarily-connected graph of arbitrarily-located vertices, can represent the same original geometry in a way more appropriate to those physics domains at the cost of mesh complexity.

Assumptions and Limitations

Juniper stores the univel grid internally as a flat array. For even the largest simulations under consideration this is sufficient, and the entire world model fits in memory. However future large problems and/or smaller univel sizes may exhaust memory, requiring more complex space-partitioning data structures such as an octree or 256-tree. Juniper only handles uncharged particles with straight-line paths between collisions, neutrons and photons. This limitation is shared by MCNP. Juniper doesn't match all of MCNP's capabilities, rather, it prioritizes the subset of most-used and interrogation-specific features, especially in its current implementation. The particle-transport properties of nuclides, embodied in the cross section library, can usually but not always be assumed to be independent of the molecular context (e.g., what other nuclides are in proximity due to chemical bonding). Juniper supports only a limited set of context-sensitive nuclide cross-section data: protium (hydrogen) in water, in polymer, or in general; deuterium in D_2O only; oxygen in water or in general; and carbon-12 in graphite, in polymer, or in general. Other situations are not supported. For that matter its cross-section library contains only 84 nuclides out of the 339 that occur in nature. Just mentioned was the problem that univels don't preserve the surface area and normals of objects; these aren't necessary for particle transport calculations. There is an irreconcilable tradeoff when meshing curved surfaces, for any mesh resolution, where to locate vertices representing the

surface—to minimize the error in volume, minimize the error in surface area, minimize the surface deviation (distance between true and meshed surfaces), or minimize the error in vertex placement (distance between true surface and vertices). Which to optimize is domain-dependent, but for its purposes Juniper uses the latter, minimizing the error in vertex placement.

Questions to be Answered

- Can Juniper improve on the effectiveness/efficiency of MCNP for particle transport simulation, taking advantage of univel-based transport calculations, confirming the experience of Tirade but with additional reactions supporting broader problem types?
- Do antialiased univel models give improved simulation results versus simple univelization? In particular what is the univel resolution vs. result fidelity tradeoff with and without antialiasing?
- Can the specific properties of material vectors be used to offset their problems with high-dimensional vector quantization?
- If antialiased univel models do give improved simulation results, what is the computational cost and is it worth it? Here the interesting tradeoff is between computational time, space, and result quality.

Significance of the Study

Fundamentally, the radiation transport problem is itself significant. The planned applications to developing active detectors for smuggled nuclear material, and to developing fourth-generation nuclear power plants, are timely issues with the potential for real-world implications. Pushing the state of the art in radiation transport simulator capabilities contributes to the general problem even beyond these specific applications. Univelizing non-univel geometries for particle transport is a new approach, and while there's been some work on antialiased voxellization for computer graphics, antialiasing with high-dimensional (material) vectors has not been done before.

Definition of Terms

Card Originally an 80-column punch card, one (80-character-limited) line of text in an MCNP input file.

Cell A geometric region that may be assigned attributes such as material or tally information, defined by a Boolean combination of mathematical surfaces.

Cross-Section The probability of a reaction between a particle and some material, represented as a hypothetical cross-sectional area of material which a point-like particle will react with only if it intersects. Typically measured in barns, $10^{-28}m^2$.

DAG (Directed Acyclic Graph) A directed graph without cycles. Used by Juniper to model the CSG geometry structure as it's more general than a tree but can still represent a parent/child relationship.

Euler Angles One representation of rotation, or orientation, in three-dimensional space. There is a perilous diversity of conventions for their interpretation; Juniper follows the x convention, where the three Euler angles ϕ, θ, and ψ are applied in sequence as rotations about the z, x', and z' axes respectively, with a right-handed coordinate system and right-handed rotations. (A w' axis is what a w axis becomes after one rotation step.) For example, the rotation matrix when $\phi \neq 0, \theta = 0, \psi = 0$ would be

$$\begin{bmatrix} \cos\phi & \sin\phi & 0 \\ -\sin\phi & \cos\phi & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Geome A primitive three-dimensional geometric element in Juniper's CSG model, such as a sphere, cylinder, or parallelepiped.

Heaviside Function (Step Function) In Juniper, converts distance-function representations of geometric components into discrete inside/outside decisions. Defined as

$$H(x) = \begin{cases} 0 & : x < 0 \\ 1 & : x \geq 0 \end{cases}$$

ITU (Intermodal Transport Unit) ISO-standard shipping container. Typically a corrugated steel box $8'$ wide, $9'6''$ tall, and $20'$ or $40'$ long.

JTDL (Juniper Trial Description Language) XML-based input format for Juniper.

MCNP (Monte Carlo Neutron Photon) Particle transport simulator, supporting neutron and photon transport calculations through surface-function-defined geometric cells.

Mean Free Path The average distance a particle travels through a material between (reaction) collisions. It depends on the material type and density, and particle type and energy.

Mole Fraction The amount of a constituent substance in a mixture, measured as a ratio of *moles of constituent* to *combined moles of mixture*. A mole of a substance is the count of formula units (such as molecules) divided by Avogadro's number (about 6.022×10^{23}).

Monte Carlo A form of computational simulation using pseudorandom samples of the problem space. In particle transport, virtual particles probabilistically sample the geometry and space of possible reactions and particle paths.

Nuclide A particular substance identified by the specific makeup of its nuclei, that is, its count of protons and neutrons. Each nuclide is some isotope of some element.

NURBS (Non-Uniform Rational B-Spline) A smooth n-dimensional curve or surface shaped piecewise by control points. Can be used to represent surfaces

bounding 3-D volumes for particle transport geometric models, although ray-NURBS intersection calculations are costly.

Nyquist Limit Half a given sampling frequency. Signal frequencies higher than this limit can't be fully reconstructed from their samples, instead yielding aliasing artifacts.

PLC (Piecewise Linear Complex) A type of geometric structure more general than a polyhedron. Made up of vertices, edges, and planar facets—which may be concave and have holes or interior vertices and edges, even disconnected.

Rasterization Conversion of geometric or image information from a vector format of shapes to a regular grid of raster elements, typically pixels (in 2-D) or voxels (in 3-D).

Univel Uniform volume element. Similar to a voxel but with any aspect ratio, i.e., a parallelepiped but not necessarily a cube. Particularly useful for medical scan data made up of a stack of slices, where the between-slice distance is often not the same as the within-slice pixel size.

V-Model A geometric object representation with fuzzy boundaries. The "exterior-ness" of any point is given as a continuous value on $[0, 1]$ with the 0.5 isosurface at the conventional (crisp) object boundary.

Voxel 3-D (volumetric) analogue of a pixel, one cell of a space subdivided into a Cartesian grid of cubical cells.

REVIEW OF THE LITERATURE

This study is examining rasterized univel particle transport, the issues of data representation and file format useful to that end, the possible benefit of antialising the rasterization, and generating mesh geometries alongside univel ones for multi-physics support. Therefore this review will consider the particle transport problem and programs simulating it, the relationship between computer graphics and general computational geometry, voxels and univels in 3-d geometry, antialiasing, vector quantization as it applies to color reduction and antialiasing, high-dimensional spatial indexing, constructing geometric meshes from other representations, file formats for scientific data and geometric representations, and the issue of numerical computing in the Java language.

In simulating neutral particle (e.g., neutrons and γ photons) transport through practical geometries, analytical methods are insoluble and numerical approaches must be used insted. A common numerical approach is Monte Carlo, which simulates a finite number N of particle "histories" through the geometry, selecting reactions probabilistically from probability density functions with a pseudorandom number generator, building a tally $\hat{x}$ of some property of interest, that approximates the true value $\bar{x}$ with sufficient histories. Lewis et al. (1984) discuss this in depth. [3] One of the most prominent Monte Carlo particle transport simulators is MCNP (Monte Carlo N-Particle). [4] It supports several particle types, a large number of reaction types,

criticality analysis, general sources and tallies, and techniques for variance reduction. $G_{EANT}4$ (GEometry ANd Tracking) is another, more recent, Monte Carlo transport program. It is similarly broad-featured supporting many applications but its prime motivation was in detector design for particle accelerators, focusing on high-energy physics. [5] Then MINERVA is a more specialized transport program designed specifically for radiotherapy planning. [6] Using univel geometry for its particle transport core, that core was the inspiration and starting point for Juniper.

There is a close similarity between Monte Carlo methods for solving particle transport problems and approaches like ray tracing or radiosity for solving the rendering equation. [7] This is one of many fundamental analogies between issues in the domains of computer graphics and particle transport, being as they are highly based on three-dimensional computational geometry. This raises opportunities for making use of established computer graphics techniques toward particle transport; a rich collection of such computational geometry resources intended for graphics applications but useful to Juniper lies in the book Geometric Tools for Computer Graphics. [8]

In typical particle transport simulations with surface-function-based geometries a significant contributor to computational cost are particle-surface intersection calculations. This can be improved by substituting an transport algorithm using mostly integer arithmetic in a way analagous to the Bresenham line rasterization algorithm, making use of a univel-based geometry. [1] A univel is a three-dimensional volume element similar to a voxel; voxels are receiving increased attention for volumetric

applications in computer graphics for many of the same reasons that pixels, the 2-d raster representation, did: as memory increases to support the relatively large storage overhead, the computational advantages of raster formats become attractive. [9] This phenomenon applies also to particle transport, but where the input geometry is not already in univel form it must be converted (univelized). The most common way of doing this is point sampling: querying the input geometry at a representative point in each univel, typically the univel's center, and assigning that value to the entire univel. [10] Usually the sampling points (and univel arrangement) are selected on a Cartesian lattice, however, a similar accuracy can be obtained with 29.3% fewer samples by using a body-centered cubic lattice instead, at the cost of complexity. [11]

The problems caused by aliasing in 2-d rasterization, such as missing details and "jaggies" at boundaries, have been well-known for some time[12] and extend also to the 3-d case. Many antialiasing methods exist to reduce the harmful effects of aliasing, these include sampling more than one point per univel (supersampling), randomly jittering the sample points away from pixel/univel centers (stochastic sampling), or prefiltering the image components prior to point sampling to reduce high-frequency information. [13, 14] A prefiltering method designed specifically for volume graphics voxelization is the V-model approach. [15] This approach converts the object representations into a three-dimensional density function, where the function is chosen to have no spatial frequencies above the sampling density's limit. Point-sampling these models yields a fuzzy classification of interiorness for the voxels (or univels) without

aliasing. In the computer graphics domain voxel antialiasing serves somewhat different goals than pixel antialiasing, as the latter aims to maximize the subjective visual quality of rasterizations while the former aims to maximize the quality of subsequent volume graphics calculations. [10] The needs of the particle transport problem share goals similar to those of volume graphics.

Juniper requires the antialiased, per-univel material vectors to be reduced to indices into a much smaller table of reference vectors, a quantization problem. The algorithm due to Lloyd [2] sets the standard for constructing a vector quantization codebook from a set of training vectors. For situations like this where, once the codebook is created, only the training data need be quantized, the problem is known as "cluster analysis" and is fundamentally related to the k-means clustering algorithm. [16] Originally described for one-dimensional quantization, it can be extended to apply equally well in higher-dimensional vector spaces. [17] Vector quantization can produce a 17-fold reduction in data size in volumetric data [18], but although this paper included a discussion of higher-dimensionality issues, the results were based on 1-d vectors only. Higher-dimensional quantization for image data is described in Xiang (1997), which quantizes 2-d image pixels, but 3-d color vectors (in an RGB color space). [19] Another approach to 3-d color vector quantization instead uses fuzzy set membership to perform many-to-many mapping between data vectors and code vectors as an intermediate step, with good results. [20]

Lloyd's algorithm involves repeated nearest-neighbor queries over the vector data, queries which contribute much to the computational cost of the algorithm. That cost can be mitigated by effective spatial indexing of the vectors for efficient querying. Most spatial indexing methods contain at their core, or are derived from, the R-Tree—which is in essence a B-Tree of spatial bounding boxes. [21, 22] Multiple R-Trees are possible for the same underlying data, with different patterns of directory rectangles at the non-leaf nodes. Finding good R-Tree structures can significantly impact performance, and is the goal of the R*-Tree modification. [23] One early alternative to the R-Tree family is the Grid File, which independently subdivides each dimension of the data, then assembles convex rectangular portions of the subdivisions as buckets. [24] This approach doesn't scale well to higher dimensions, partly because each query must search every dimension's index. However all traditional spatial indexing methods must deal with the so-called curse of dimensionality—the exponential relationship between dimension number and volume. Dimensions beyond two, or certainly three, are problematic for spatial indexing. [25] Quite a few improved indexing methods have been proposed in recent years that achieve some incremental gains over the dimensionality problem. These methods variously involve using bounding hyperspheres instead of hyperrectangles at the directory non-leaf nodes (SS-Tree) [26]; using the intersection of a hypersphere and a hyperrectangle to bound non-leaf nodes (SR-Tree) [27]; calculating the Voroni nearest-neighborhood of the code vectors, then storing their bounding hyperrectangles in an R-family-Tree (NN-Cell) [28]; using the Hilbert

Curve or other space-filling curve to map n-dimensional coordinates to one dimension, then indexing that derived coordinate in a B-Tree (Hilbert Curve Index) [29]; further improving the tree structure of R*-Trees with a better node-splitting algorithm and larger directory "supernodes" (X-Tree) [30]; dividing the n-dimensional space into $2n$ hyperpyramids, then subdividing those into parallel hyperfrustra, and using a total ordering of hyperfrustra to create a one-dimensional coordinate of each object, which is indexed in a B^+-Tree (Pyramidal Technique) [31]; or modifying the Pyramidal Technique by subdividing the (hyper-)pyramids with spherical shells rather than planes (SPY-TEC) [32], an approach that has specific advantages in high-dimensional nearest-neighbor queries. [33] Lastly mention must be made of the non-index Linear Search, as the curse of dimensionality means all spatial indexing methods eventually fail to improve on it as n increases. A partial ordering of these techniques' relative performances on high-dimensional data, according to these sources, is summarized in Figure 2.

Some physical simulations in 2-d and 3-d expect an unstructured, boundary-conforming mesh—for example, the finite element method. In order for the mesh elements to be well-shaped for the simulation, a common approach is Delaunay tetrahedralization[34]: adding interior (Steiner) points to create tetrahedra conforming with the input boundary, where the choice of new points, and post-placement adjusting of the mesh, is done to maintain the Delauney criterion. In three dimensions this criterion is that no vertex lies inside the bounding sphere of a tetrahedron.

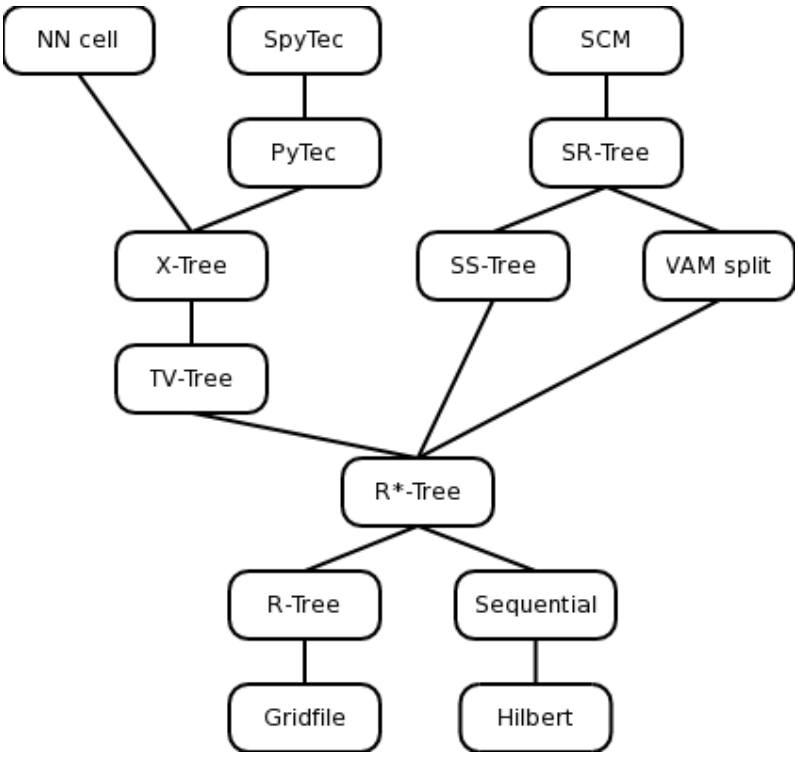

Figure 2. Relative general performance of spatial indices, as reported in the literature.

Being Delaunay isn't enough alone to guarantee a physically "good" mesh, there is for example the possibility of sliver tetrahedra, but there are refinement techniques that can remove most such problems. [35] One program that generates Delaunay tetera-hedralizations from a surface mesh in PLC form, and that Juniper's Sinnet mesher is designed to use, is TetGen. [36]

In a constructive solid geometry (CSG) representation, objects are constructed from 3-d primitives combined using regular Boolean operators, forming a tree or DAG of composed units. In a boundary representation (B-rep) geometry, objects are repre-sented by the vertices, edges, and oriented facets that constitute their boundary. [37]

The problem of converting a geometric model from a CSG representation to a B-rep representation hinges on converting the CSG primitives, a relatively straightforward prospect, and performing the equivalent of the Boolean operators on these "primitive" B-reps. [38] This process is known as boundary evaluation and merging. Eberly (1999) has a good explanation of the fundamentals of a common technique. [39] However, there are a proliferation of difficult corner cases, especially in the three-dimensional domain. Many of these are along the lines of coplanar contact, colliner contact, or point contact; where objects intersect only in a lower-dimensional way. Correct boundary evaluation and merging of these cases requires complex special handling. [40, 41] Effective boundary evaluation and merging depends on appropriate design choices for data structures of surface mesh representations, selecting the best tradeoffs among flexibility, time and space efficiency, and ease of use. [42, 43]

There is a plethora of file formats for representing 3-d geometric models, using different geometric representations and targeting different applications. The MCNP "deck" input format was designed to answer needs very close to those of Juniper. [4] But there are desirable features of an input format it doesn't support, such as the ability to represent hierarchical geometry trees, or supporting translation and exchange of geometry information through things like building on an underlying XML basis. These are some of the features provided by the GDML geometry data exchange format, designed closely with $G_{EANT}4$. [44] However with a different group of geometric primitives translating certain MCNP problems into GDML is made difficult. Another

CFG geometry input file format is the Scene Description Language used by the ray-tracing program POV-Ray. Despite being designed for the computer graphics domain there are many commmonalities in the 3-d geometric modelling requirements. [45]

One insight provided by these existing formats is that there are multiple layers of design choices in input formats: one being how to structure the geometric data, another is how to represent that structure in a binary or text file serialization. One general approach to the latter question, specifically intended for scientific data exchange, is HDF5. [46] This provides a general data model, binary file format for storing/exchanging data models, and a library and API for working with these files. Alternatively there are more general approaches that seek to define standardized binary encodings of more universal data models, on which domain-specific data structures can be layered. These include EBML and the various encodings of ASN.1. [47, 48] Yet, for reasons such as flexibility and long-term readability of archived data, experience is showing textual, rather than binary, file formats are preferable. In particular layering a higher-level data model on the well-recognized XML format is a popular and advantageous strategy. [49] McGrath (2003) examines the comparative strengths and weaknesses of XML and binary data formats for scientific data specifically. [50] For 3-d geometric models both the aforementioned GDML and X3D formats are XML-based. [44, 51] X3D is an ISO-standard data format for multimedia 3-d computer graphics, a successor to VRML intended for web publishing, visualization, and other applications.

The idea of using the Java language for computationally-intensive programs is often automatically dismissed. There are considered to be fundamental weaknesses in the language's design interfering with speed, which is often considered a tradeoff with other benefits like safety and expressiveness toward program design. [52] But with appropriate language-specific techniques for design optimization, this tradeoff can be adjusted to regain speed, though at the potential cost of reducing some of those other benefits. [53] With regards to the object-oriented paradigm in general Besset (2001) makes extensive use of both Java and Smalltalk for numerical methods computation, and in comparison with C, finds in favor of the OO approach and considers Java, at least, competitive with C. [54]

APPROACH

The Juniper program is at its heart a Monte-Carlo particle transport simulator, using a univelized geometry representation and written in Java, intended for a range of applications including evaluating neutron- and γ-photon-based active scanner designs for detecting fissile nuclear material smuggled inside standard ITU shipping containers, and exploring problems of next-generation nuclear reactor design. It draws from the INEL/MSU radiotherapy simulator Minerva and the particle transport program MCNP, and includes broader reaction and material types and provisions for translating MCNP input, generating univel geometries from nonunivel representations (including antialiasing), and generating PLC surface mesh representations of its geometries. In this way it can achieve the stated objectives of expanded univelized particle transport, multiphysics mesh support, improved input data representation supporting MCNP validation, and evaluating antialiased univelization. Juniper is formed from six modules providing the core transport functionality and associated support: Transcore, handling the particle transport; Translate, which converts MCNP input files for use with Juniper; JTDL, the input format for specifying geometries and run data; Trace, which rasterizes input geometries into univel form; Antalun, an experimental antialiasing rasterizer; and Sinnet, which converts geometries to a mesh representation to support multiphysics simulations. Figure 3 shows the relationships among these modules.

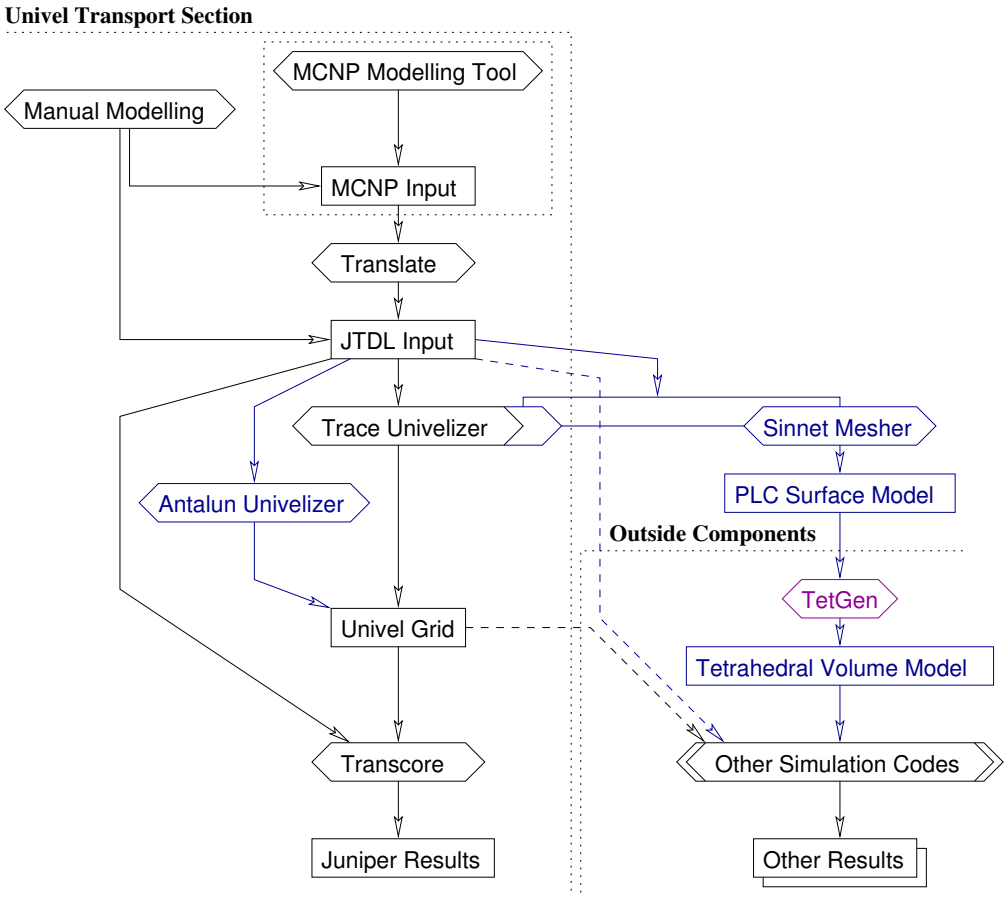

Figure 3. Juniper overall architecture.

Current Approach

Transcore

Simulating particle transport through a geometric model is the responsibility of Transcore. It draws from the INEL/MSU radiotherapy simulator Minerva's transport module, JART, as well as the MCNP transport simulator. As a Monte Carlo simulation engine it generates virtual particles and tracks them as they move, collide,

and react with the modelled geometric structure. A pseudorandom number generator selects the initial particle's state, locations of collisions, types of reactions, and other processes of the particles' tracks based on the input problem description and a database of reaction types and properties per nuclide, the cross-section database. Like many Monte Carlo transport simulators the particle tracking is not directly analogue: adjustments improve efficiency and reduce variance in the final result. Principal among these is the association of a *weight* with the virtual particles. A particle with weight 1.0 might be said to correspond with one physical particle. Then to ensure good sampling of part of the problem space, if a reaction called for a particle to be absorbed with 80% probability, the particle would instead be always kept alive but with a new weight of 0.2. High-interest areas might be sampled with many particle histories by "splitting" a particle into multiple copies, preserving the total weight. Lastly to ensure the program doesn't spend excessive time calculating the histories of very-low-weight particles not contributing much to the result, those particles are culled in a probabilistic fashion, so as to maintain conservation of weight.

Translate

The Translate module of Juniper is responsible for converting MCNP input files, based on surface and cell geometry definitions, into Juniper's input format—JTDL. To this end JTDL includes several features specifically to aid translation, described later. The first stage of translation must preprocess certain idiosyncrasies of the

MCNP input format. MCNP limits line length to 80 characters, using "continuation cards" to spread long input units across multiple lines. Some types of MCNP input form 2-D tables of data listing attributes of objects. The default table format orients the "attribute" dimension down multiple lines, with the "object" dimension along each line. This was considered inconvenient for actual cards of input, so tables may be flipped. The preprocessor converts input *cards* into abstract *lines* representing one discrete input unit each. MCNP input also contains "special syntax items" which substitute letter codes for expected numeric entries; shorthand forms for repetition, skipping defaults, linearly interpolating a range of entries, or more complicated operations. The preprocessor unpacks these codes. After preprocessing, Translate separately converts the geometry, material, source, and run parameter sections of the MCNP input.

The geometry of a problem in MCNP input files is defined in two separate sections: a list of oriented surface functions, and a list of "cells" defined by Boolean combinations of some surfaces. There may also be macrobodies: precombined surfaces forming more complicated shapes. Translate matches this structure by first translating the surfaces and macrobodies to equivalent primitive or composed CSG objects, placed in a DAG forest of components. Then it translates the cells themselves into portions of the JTDL world structure, referencing the appropriate components.

Each MCNP cell also references an entry in its material/region table. These data can be directly translated to JTDL material and region specifications, placed

in separate areas much like the geometric components. Those portions of the world structure representing cells then reference their regions in the same way. Particle sources in MCNP input are also directly translatable to their JTDL equivalents—a source occupies a region of space (either spheric, cylindric, or a cartesian box). Within that region new particles are generated with attributes that may be constant, drawn from a random sampling of probability density functions, or specified by certain functions of other parameters. Lastly, parameters controlling the specific simulation run must be translated in a more ad-hoc fashion.

JTDL

The Juniper Trial Description Language (JTDL) is an XML-based input format for Juniper. By using XML Juniper can take advantage of existing parser and emitter libraries available across multiple platforms and environments. This includes automatic input validation against a JTDL schema and composition of multiple files. JTDL was partially inspired by the Geometry Description Markup Language (GDML), an XML-based geometry data exchange format developed for use by participants in the Simulation subproject of the CERN Large Hadron Collider Computing Grid Project. Some limitations prevent GDML from being directly useful for Juniper input: having a different group of geometric primitives, translation of certain MCNP problems is more difficult; more generally their approach to geometry specification is not as flexible as one might like; and it is not clear how to smoothly integrate the

needed extensions for Juniper's handling of things like material and region attributes, particle sources, and result tallies.

The geometric model in JTDL uses constructive solid geometry to combine primitive elements ("geomes") and transformations into an overall geometry DAG analagous to a scene graph. The currently supported primitives are listed in Table 1 and shown in Figure 4. The types of primitives supported and the methods of defining them often specifically support direct translation of MCNP input files, e.g., oid and volume objects. Geomes combine by the standard CSG set functions "union", "intersect", "negate", and "subtract". These can themselves be composed with primitives or other composites. Any geometry node, primitive or composite, is transformable by any combination of the operations listed in Table 2.

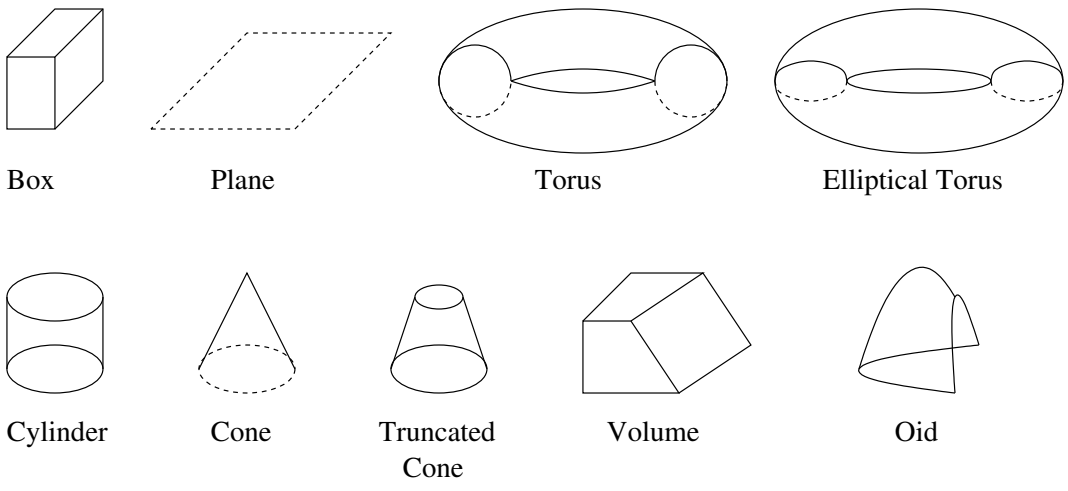

Figure 4. Geome primitives.

Table 1. Geomes in JTDL.

Name	*Description*	*How Specified*
box	orthogonal rectangular parallelepiped	defined by two opposite corner points
plane	infinite plane (extending to world boundaries)	either by normal vector and offset, or three points
torus	possibly ellipsoidal torus	center point, hole vector, one major and two minor radii
cylinder	possibly infinite solid cylinder	two end points, radius, infinite flag
cone	cone of one sheet, possibly truncated, infinite, or both	either by top and bottom points and radii; or by vertex, opening vector and angle, and possibly truncation offset(s)
volume	general hexahedron	between six and eight vertices
oid	general quadratic (spheroid, paraboloid, hyperboloid)	coefficients to quadratic distance function $Ax^2+By^2+Cz^2+Dxy+Eyz+Fzx+Gx+Hy+Jz+K=0$

Table 2. Transformations in JTDL.

Name	*Description*
translate	
scale	uniform or separately by dimension
rotate	either by axis and angle or by three Euler angles
shear	separately by three orthogonal shear planes
matrix	general transformation matrix

In addition to describing the geometrical structure of the model, JTDL supports defining a set of material types to be associated with that model. A material type may be a composition of other materials, in ratios specified by mole fraction or mass fraction. Ultimately the primitive material types are called nuclides, specific elemental isotopes. Each nuclide corresponds with a specific entry in the transport core's library of cross-section data. This hierarchical composition of materials naturally supports techniques such as combining isotopes into a library of natural-isotopic-abundance elements, combining those into common compounds, and combining those into useful mixtures, all using the same mechanism. Lastly material types combine with a density, definable in various useful units, and other attributes into a region. These regions associate with nodes in the geometric structure to determine the particle transport physical properties within that part of the geometry during the simulation.

Ultimately in a JTDL input geometry all these elements (geomes, set operators, transforms, materials, and regions) combine with cross-referencing into a singly-rooted DAG, similar to a scene graph in computer graphics, fully defining the problem structure. Region and material descriptions can be attached to arbitrary nodes at any level of the geometry. Any component can be defined in-place, in a separate section of the file, or in other files, and any node can be named for inclusion by reference elsewhere. This flexibility eases multiple approaches to input definition, whether different types of manual design, machine construction from a GUI design program, or

machine translation from other file types (especially MCNP input). MCNP only assigns its equivalent of regions and materials to top-level geometric areas, called cells, keeping those material definitions in a separate table. JTDL directly supports this approach. Also in JTDL, model portions being included by reference can be copied with variations in any parameter, allowing "template" constructs as a basis for individual variants as needed. This also directly supports translation from MCNP files. At the same time JTDL supports partitioned input, separated into predeveloped libraries of common things like material types, model components, and particle sources; a particular geometry for a particular problem; and multiple sets of run-control parameters for various types of simulations on that geometry.

Trace

Preparing the input problem model for particle transport simulation requires the CSG model be rasterized into a discrete univel representation. The Trace module performs this function. The most straightforward method is to point-sample the DAG structure along a Cartesian grid, on the centerpoints of the univels. The most-specific region attribute of the model at each point is assigned to its univel, although this region-resolution behavior can be changed in the input file for special cases. The typical ITU container problems are much larger, physically, than patient scan data. However, for transport efficiency keeping the univel model as a single in-memory array is desirable. As an opposite consideration, to avoid transport artifacts, the univel size must be smaller than the mean free path of the tracked particles, typically on the

order of $\frac{1}{2}$cm linearly. The univel grid array stores for each univel a two-byte index into a region table, while that table stores the appropriate simulation data for its region.

When visually examining model univelizations is useful, a simple display tool renders the univel grid with OpenGL fat points. It colors these points with a user-specified mapping between nuclide type and primary color, showing material combinations using color blending. Additionally, the user can highlight specific regions with individual colors as well. The user can navigate around and through the 3-D model, slicing with arbitrary cutting planes to view the internal structure. Figure 5 shows a screenshot of this program.

Antalun

The conflicting limitations of physical accuracy and memory capacity on univel size and problem size are joined by an aliasing problem: a thin slab of material (such as the wall of an ITU container) could have a thickness on the order of one univel size. After rasterization the slab's thickness would be substantially altered, and where the slab passes between sampling points it may disappear entirely. Similar problems affect any fine features of the geometry. The computer graphics world has used anti-aliasing techniques for some time, and those techniques—especially for 3-D voxellization antialiasing—can be adapted for univel rasterization. Experimenting with this approach is the domain of the Antalun module. The model shown in Figure 5

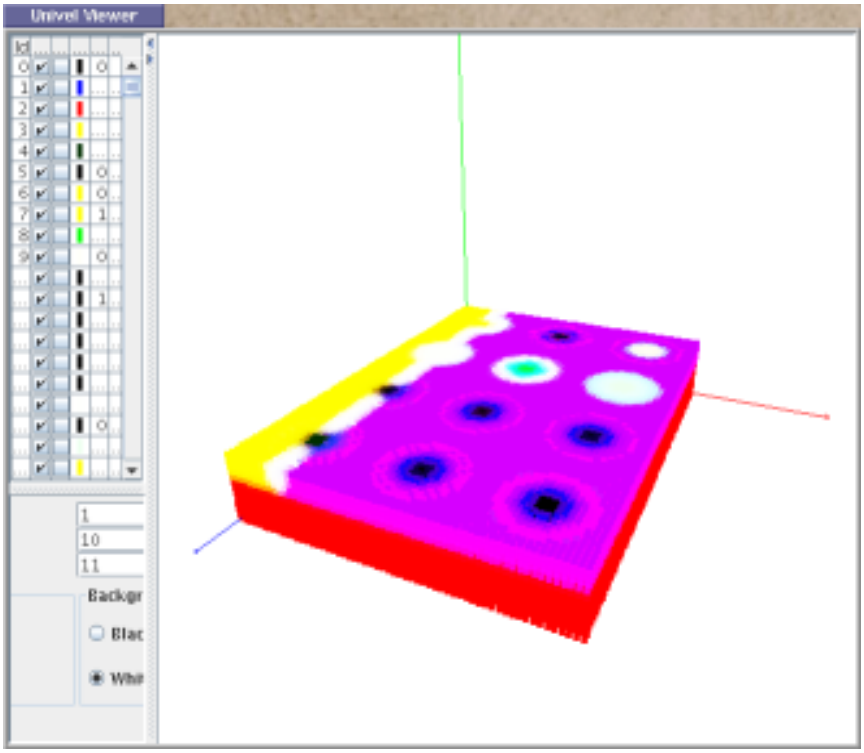

Figure 5. Antialiased univel test model "eggcrate".

has been antialiased by Antalun; note the blended transitions between colored regions (in the diagram red is mapped to hydrogen, green to carbon, and blue to nitrogen).

A principal obstacle is that, in computer graphics, each pixel or voxel value is typically a color vector in some 3-D space like RGB. The antialiasing process creates uniformly blended color vectors around high-frequency areas which, for most modern applications, can be stored completely. But univel values in Juniper's models are high-dimensionality region/material vectors. Storing a unique, blended material vector for each antialiased univel results in unworkably large model sizes. A necessary step, then, is quantizing these material vectors into a region table of at most 32,767 entries.

Antalun uses a prefiltering approach to antialiasing, where the object model itself is filtered of object details with higher frequency than the Nyquist limit of the univel resolution. It does this by switching to a V-Model representation of geomes and CSG combining functions. Conventional geome primitives can be treated as a distance function representing the distance from any point to its surface, which is then passed through a Heaviside function to flatten it to a crisp inside/outside predicate (Figure 6). In a V-Model that distance function is instead scaled onto $[0, 1]$, interpreted as a fuzzy predicate where the conventional object surface lies at the 0.5 distance. The scaling might use a linear (Figure 7), quadratic, or Gaussian transformation (Figure 8) with the "spread" tuned according to a resolution parameter. The CSG combining functions also have V-Model equivalents, shown in Table 3. The method II column is the standard interpretation of fuzzy set operations, but methods I and III can give more physically approprate results. With the CSG geometry reinterpreted as a V-Model, it is Cartesian-sampled with regular points like Trace, only this time producing an alias-free rasterization with (potentially) unique material vectors for each univel.

With over 100 unique nuclides these material vectors have a high dimensionality, although in practice each contains only a subset of the possible nuclides, a sparse vector. But these material vectors are still incompatible with the transport approach, expecting indexed regions. What's more there is the size problem: consider a 1-meter cube univelized at 0.5 cm resolution. The univel grid will have 8,000,000 univels; if

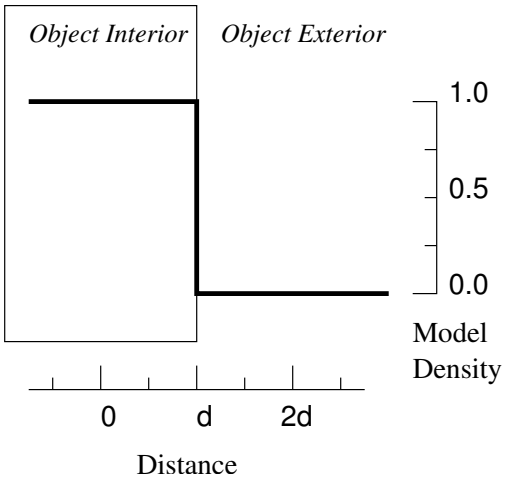

Figure 6. Distance function converted to crisp interior/exterior distinction.

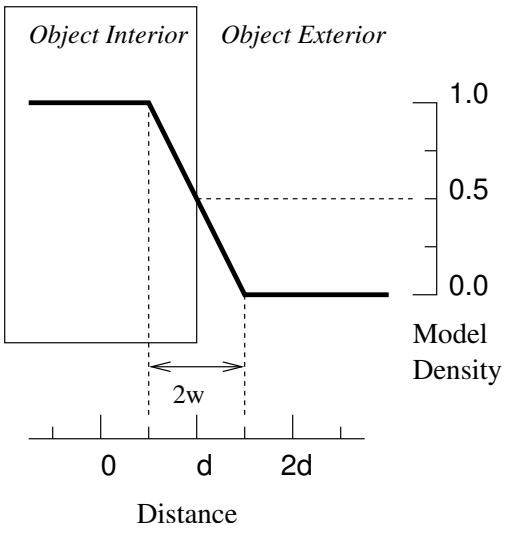

Figure 7. Piecewise linear v-model.

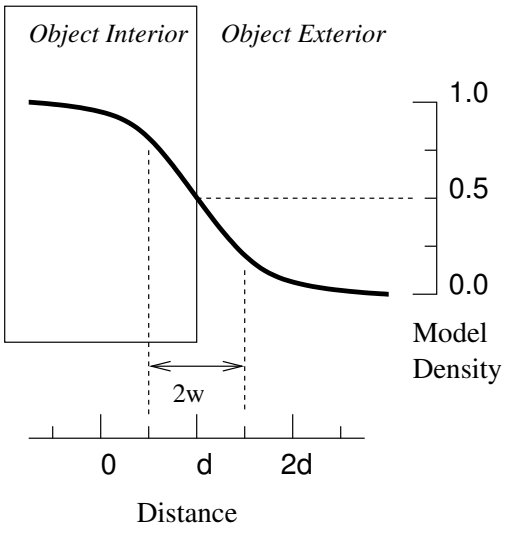

Figure 8. Gaussian v-model.

each univel has a complete material vector (76×8) bytes, the grid occupies 4.5 GiB. So the univel grid must be returned to a form where each univel references a region table entry. Here again graphics techniques are useful: until recently limitations of display hardware and storage size led image data to often be kept in a palettized format—a color table and an image map referencing that table. "Color quantization" algorithms construct a good color palette for a full-color image. Following such an approach, that 1-meter cube's material vectors would be quantized into 2^{15} regions in an antialiased region table, and 2-Byte univels, giving a 15 MiB univel grid and 19 MiB material palette, only 34 MiB total. Figure 9 shows how three original basis materials might be blended into many unique material vectors, then with a smaller number of code vectors dividing the vector space into nearest-neighborhood Voroni regions each vector in a region is mapped to the defining code vector.

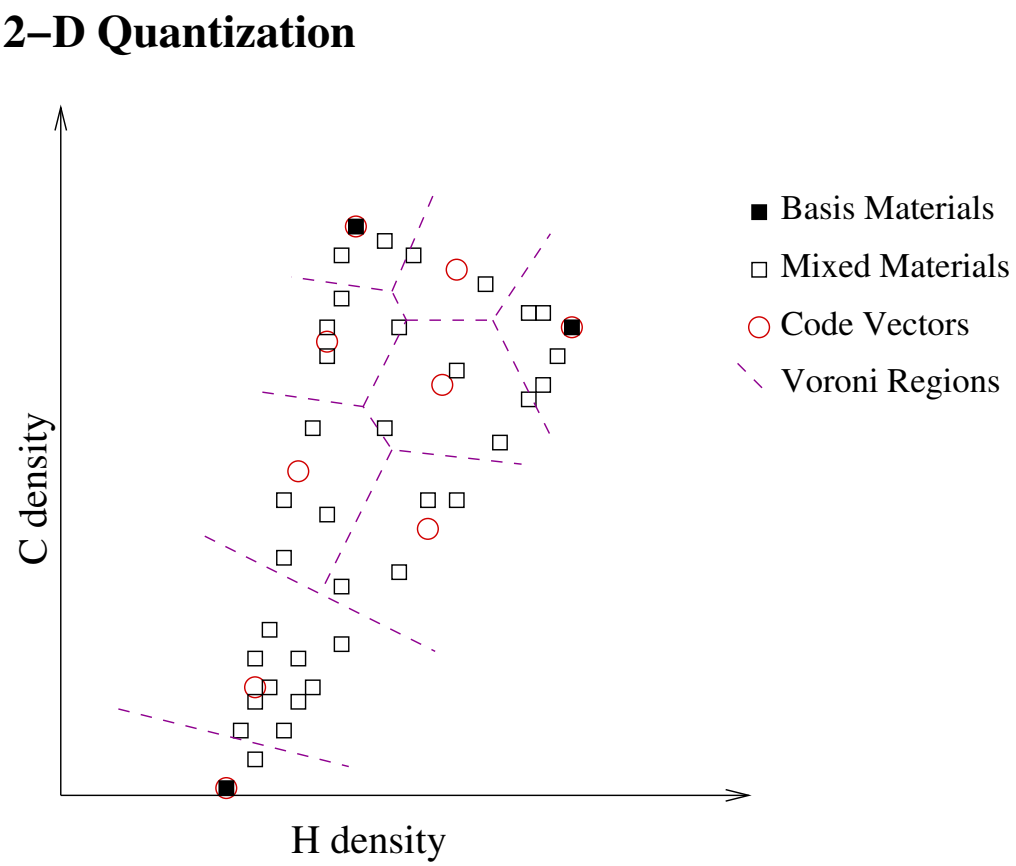

Figure 9. Vector quantization in 2-D.

Antalun uses a common algorithm for developing a vector quantization codebook from a set of training data due to Lloyd, shown in Algorithm 1. A particular challenge in applying this algorithm for high-dimension material vectors is identifying the nearest-neighbor codebook vector to an original material vector, more efficiently than a simple linear search through the codebook. Either the codebook, or the space of material vectors, or both could be indexed spatially to support more efficient nearest-neighbor queries. Over the years considerable work has gone into spatial indexing, although primarily for 2- and 3-D data due to the "curse of dimensionality"

which makes higher-dimensional data difficult to efficiently index. Currently Antalun supports two index structures beyond the null-hypothesis linear search: Spy-Tec and NnCell.

Spy-Tec (Spherical Pyramid-Technique) rescales each dimension from $[0, \infty)$ to $[0, 1)$. Then it subdivides this rescaled hyperspace into pyramids with their vertex at the center and their base covering one of the space's bounding hyperplanes. (Thus, for n dimensions, $2n$ pyramids.) If necessary it subdivides each pyramid into slices along equal distance-to-center hyperspheres. The (slices of) pyramids are numbered in a way that supports a total ordering, flattening n-dimensional vectors to a single coordinate. That coordinate can be used to index the vectors in something like a B+-tree while preserving spatial relationships in a way that supports nearest-neighbor queries. For the Nn-Cell approach each code vector has a surrounding orthogonal bounding box approximating the Voroni nearest-neighborhood. These cells can be organized in a conventional hyperdimensional spatial index (the designers suggest an X-tree or R*-tree), and nearest-neighbor queries become fast point queries.

Table 3. V-Model forms of CSG operators.

Operator	*Method I*	*Method II*	*Method III*
Union $D_{\cup}(A,p) =$	saturating add: $\min\left[1, \sum_{a\in A} d(a,p)\right]$	maximum: $\max_{a\in A}[d(a,p)]$	inclusion-exclusion sum: a long series[a]
Intersection $D_{\cap}(A,p) =$	product: $\prod_{a\in A} d(a,p)$	minimum: $\min_{a\in A}[d(a,p)]$	product: $\prod_{a\in A} d(a,p)$
Negation	$D_{\neg}(a,p) = 1 - d(a,p)$		
Subtraction $D_{-}(A,p) =$	defined in terms of the other functions: $D_{\cap}(B,p), B = \{a_0, \neg a_1, \neg a_2, \ldots\}$		

[a] which series is familiar from probability theory:

$$\sum_{a\in A} d(a,p) - \sum_{a,b\in A} d(a,p)d(b,p) + \sum_{a,b,c\in A} d(a,p)d(b,p)d(c,p) - \cdots + (-1)^{n-1} \prod_{a\in A} d(a,p),\ n = |A|$$

Algorithm 1: Lloyd's as used by Antalun

Data: X, a set of material vectors
Data: $B \subset X$, a set of basis material vectors of size r
Data: C_0, an empty codebook of n vectors ($n > r$)
Result: C_q, an appropriately-filled quantization codebook

initialize the first r entries of C_0 from B
fill the remaining $n - r$ entries by some method
for $k \leftarrow 0$ **do**
    {`quantize` X `using` C_k `and evaluate`}
    foreach $x \in X$ **do**
        find codevector $y \in C_k$ such that distortion $d(x,y)$ is minimized
        add x to S_y, the set of vectors supporting codevector y
        update overall distortion D with $d(x,y)$ by some method
    if $D < D_{ge}$ *or* $k = k_{lim}$ **then**
        {D_{ge} `is "good enough" distortion,` k_{lim} `is iteration limit`}
        return C_k
    {`mutate codebook according to centroids of supports`}
    foreach $z \in C_k$ **do**
        let z' be the centroid of S_z
        add z' to C_{k+1}
    $k \leftarrow k + 1$

Sinnet

Examining more sophisticated questions about a physical system often involves combining simulations of multiple different physical processes on the same geometric structure (multiphysics). While univel representations of models' geometric structures are useful for particle-transport calculations, they're less suited to other problems such as heat transfer where finite element methods are preferable. For this reason Sinnet, an extension to Juniper, converts any JTDL model into a matching 3-D surface mesh representation. This mesh can be passed on to a program called TetGen to fill the interiors into a solid mesh according to the Delauney criterion, producing a volumetric mesh suitable for passing on to other physics simulations. Figure 10 shows an interior cutaway of such a mesh produced from Sinnet and TetGen.

The fundamental problem Sinnet must solve is converting a CSG geometry specification into a matching surface mesh representation, a process called boundary evaluation and merging. In this Sinnet makes use of the fact that a set of CSG primitive objects, combined using Boolean operators, is equivalent to a set of planar mesh elements called PLCs combined using Euler operations. To perform boundary evaluation it converts each geome primitive into a corresponding PLC mesh. Each geome type has a custom PLC equivalent representation. Merging these PLC meshes into more complex meshes, following the structure of the Boolean operators, is the more difficult part of the problem. There are a number of special cases possible that require more specific, ad-hoc techniques to correctly merge problematic submeshes. Additionally

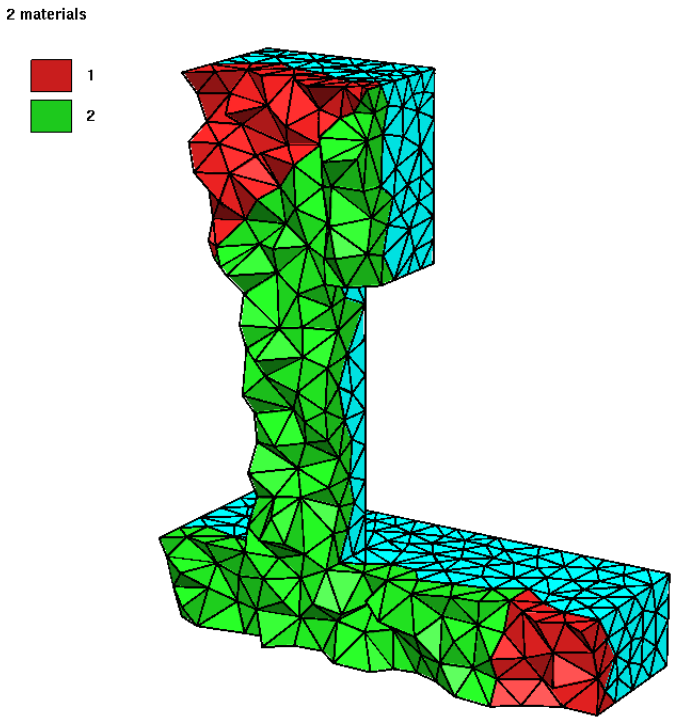

Figure 10. An example of volumetric meshing produced by Sinnet followed by TetGen.

there's a fundamental tradeoff when converting CSG primitives into PLC form where the CSG object has curved surfaces: no PLC can match the objects surfaces completely correctly. Increasing the vertex count of the PLC can improve fidelity at the cost of mesh complexity. This tradeoff is managed by Sinnet through a set of user-input constraints.

Future Work

At the time of writing, Juniper is capable of translating many MCNP input files to JTDL form, rasterizing them without antialiasing to a univel grid, and run transport simulations on that univel model. However it needs work to expand its handling of particle sources, certain reactions, and result tallies. Lacking efficiency tuning at the algorithm and implementation levels, it currently runs at about 1:1 speed compared with MCNP on a simple searchlight problem. It also needs correctness validation against MCNP simulations and experimental results. It can produce antialiased rasterizations, but this is a more speculative part of the program not integrated with the transport core. It can convert many input geometries to PLC mesh form, but handling for several degenerate cases is not yet implemented. More specifically:

Sinnet

Sinnet can generate PLC representations of the most important geomes, but doesn't yet implement torus or oid. I'll add these. Additionally, degenerate mesh-merging cases that have been left for future development include abutting objects (which intersect only along coplanar facets) and objects intersecting only along collinear edges.

Antalun

The most opportunity for future work lies in exploring antialiasing issues. The current high-dimensional spatial indices Antalun uses are designed for indexing large

counts of dynamic sets of vectors. But the antialiasing algorithm uses periods of repeated queries of a static vector set, followed by mass reorganization and more queries, iteratively. Here is an opportunity to look into "packing" algorithms for an optimizing index that takes advantage of this use pattern. The antialiasing algorithm, although following Lloyd's algorithm, has several opportunities for custom refinement in places such as generating the initial codebook and refining the error criterion, perhaps in domain-aware ways. On that subject, although blended material vectors are quite high-dimensional, they're also sparse. A domain-aware dimensionality reduction technique could improve index performance.

There is an additional question of the suitability of dithering, either as an alternative to this antialiasing or in combination with it, as a way of reducing the systematic error from univel rasterization. Antalun currently doesn't use dithering, raising first the question of whether it would be valid to the underlying particle transport physics, then—if so—what tradeoffs are involved in its use.

JTDL

Although hand-editing of JTDL input files is workable, producing more complicated input scenarios would be made greatly easier by a graphical input tool. This could be a complete bespoke modeling program, or a tool to add Juniper-specific problem information to a geometric model created using a preexisting design program. In that case there would be a need for additional translation capability to

convert between JTDL and other 3-D CAD formats. Such broader translation could also include converting to and from formats like GDML to aid data sharing.

Overall

As mentioned Antalun is a tangent project on the side of Juniper, without enough integration to run transport simulations on the antialiased univel meshes. These runs would be important to evaluating the tradeoffs and ultimately the usefulness of antialiasing in this application. For it and the rest of Juniper, as a work in progress there is significant debugging needed, as well as corner cases that have been left unimplemented along the way to getting the essential backbone functional. It will eventually need the aforementioned correctness validation, and experiment runs to evaluate the antialiasing tradeoffs. It has been designed in a way that should support parallelization across multiple processors, but this is currently unimplemented.

Timeline

My goal is to graduate in December of 2010, with four journal papers related to the research either accepted or under review by then. My timeline for accomplishing this is:

Date	Submitted	Topic
15 Sep 2009	First refereed journal paper	Either JTDL or Juniper
15 Jan 2010	Second refereed journal paper	Other of JTDL or Juniper
15 May 2010	Third refereed journal paper	Either Antalun or Sinnet
15 Sep 2010	Fourth refereed journal paper	Other of Antalun or Sinnet
15 Dec 2010	Defense of thesis	

This schedule is relatively conservative, but it includes slack to accommodate any roadblocks or resubmissions.

In addition we will be submitting papers to refereed conferences based on partial results, with the papers' timing depending on the deadlines of appropriate conferences. For example we currently intend to submit two papers to WorldComp'09 by 27 May 2009. In total my goal is to get four journal papers and four refereed conference proceedings papers from this research.

These publications will be in addition to my current published refereed journal paper[55], not related to this research.

REFERENCES CITED

[1] M.W. Frandsen. Rapid Geometry Interrogation for a Uniform Volume Element-Based Monte Carlo Particle Transport Simulation. Master's thesis, Montana State University, Bozeman, 1998.

[2] S. Lloyd. Least squares quantization in PCM. *IEEE Transactions on Information Theory*, 28(2):129–137, 1982.

[3] E.E. Lewis and W.F. Miller. *Computational methods of neutron transport*. John Wiley & Sons New York, 1984.

[4] X-5 Monte Carlo Team. *MCNP–A General Monte Carlo N-Particle Transport Code, Version 5*. Los Alamos National Laboratory, 04 2003.

[5] S. Agostinelli, J. Allison, K. Amako, J. Apostolakis, H. Araujo, P. Arce, M. Asai, D. Axen, S. Banerjee, G. Barrand, et al. GEANT4—a simulation toolkit. *Nuclear Inst. and Methods in Physics Research, A*, 506(3):250–303, 2003.

[6] CA Wemple, DE Wessol, DW Nigg, JJ Cogliati, ML Milvich, C. Frederickson, M. Perkins, and GJ Harkin. MINERVA—a multi-modal radiation treatment planning system. *Applied Radiation and Isotopes*, 61(5):745–752, 2004.

[7] J. Arvo and D. Kirk. Particle transport and image synthesis. In *Proceedings of the 17th annual conference on Computer graphics and interactive techniques*, pages 63–66. ACM New York, NY, USA, 1990.

[8] P.J. Schneider and D.H. Eberly. *Geometric Tools for Computer Graphics*. Morgan Kaufmann, 2003.

[9] A. Kaufman, D. Cohen, and R. Yagel. Volume graphics. *Computer*, 26(7):51–64, 1993.

[10] SW Wang and AE Kaufman. Volume sampled voxelization of geometric primitives. In *Visualization, 1993. Visualization'93, Proceedings., IEEE Conference on*, pages 78–84, 1993.

[11] T. Theußl, T. Möller, and M.E. Gröller. Optimal regular volume sampling. In *Proceedings of the conference on Visualization'01*, pages 91–98. IEEE Computer Society Washington, DC, USA, 2001.

[12] F.C. Crow. The aliasing problem in computer-generated shaded images. *Communications of the ACM*, 20(11):799–805, 1977.

[13] M.A.Z. Dippé and E.H. Wold. Antialiasing through stochastic sampling. *ACM SIGGRAPH Computer Graphics*, 19(3):69–78, 1985.

[14] A. Norton, A.P. Rockwood, and P.T. Skolmoski. Clamping: A method of antialiasing textured surfaces by bandwidth limiting in object space. *ACM SIGGRAPH Computer Graphics*, 16(3):1–8, 1982.

[15] M. Sramek and A.E. Kaufman. Alias-Free Voxelization of Geometric Objects. *IEEE TRANSACTIONS ON VISUALIZATION AND COMPUTER GRAPHICS*, pages 251–267, 1999.

[16] J.B. MacQueen and WESTERN MANAGEMENT SCIENCE INST UNIV OF CALIFORNIA LOS ANGELES. Some methods for classification and analysis of multivariate observations, 1966.

[17] Y. Linde, A. Buzo, and R. Gray. An algorithm for vector quantizer design. *Communications, IEEE Transactions on [legacy, pre-1988]*, 28(1):84–95, 1980.

[18] P. Ning and L. Hesselink. Vector quantization for volume rendering. In *Proceedings of the 1992 workshop on Volume visualization*, pages 69–74. ACM New York, NY, USA, 1992.

[19] Z. Xiang. Color image quantization by minimizing the maximum intercluster distance. *ACM Transactions on Graphics (TOG)*, 16(3):260–276, 1997.

[20] F. Chung and B.Y.M. Fung. Fuzzy color quantization and its application to scene change detection. In *Proceedings of the 5th ACM SIGMM international workshop on Multimedia information retrieval*, pages 157–162. ACM Press New York, NY, USA, 2003.

[21] A. Guttman. R-trees: A dynamic index structure for spatial searching. *ACM Sigmod Record*, 14(2):47–57, 1984.

[22] Douglas Comer. Ubiquitous b-tree. *ACM Comput. Surv.*, 11(2):121–137, 1979.

[23] N. Beckmann, H.P. Kriegel, R. Schneider, and B. Seeger. The R*-tree: an efficient and robust access method for points and rectangles. In *Proceedings of the 1990 ACM SIGMOD international conference on Management of data*, pages 322–331. ACM New York, NY, USA, 1990.

[24] J. Nievergelt, Hans Hinterberger, and Kenneth C. Sevcik. The Grid File: An Adaptable, Symmetric Multikey File Structure. *ACM Transactions on Database Systems*, 9(1):38–71, 1984.

[25] S. Berchtold, C. Böhm, D.A. Keim, and H.P. Kriegel. A cost model for nearest neighbor search in high-dimensional data space. In *Proceedings of the sixteenth ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems*, pages 78–86. ACM New York, NY, USA, 1997.

[26] DA White and R. Jain. Similarity indexing with the SS-tree. In *Data Engineering, 1996. Proceedings of the Twelfth International Conference on*, pages 516–523, Feb-1 Mar 1996.

[27] N. Katayama. The SR-tree: an index structure for high-dimensional nearest neighbor queries. In *Proceedings of the 1997 ACM SIGMOD international conference on Management of data*, pages 369–380. ACM New York, NY, USA, 1997.

[28] S. Berchtold, D.A. Keim, H.P. Kriegel, and T. Seidl. Indexing the Solution Space: A New Technique for Nearest Neighbor Search in High-Dimensional Space. *IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING*, pages 45–57, 2000.

[29] JK Lawder and PJH King. Querying multi-dimensional data indexed using the Hilbert space-filling curve. *ACM SIGMOD Record*, 30(1):19–24, 2001.

[30] S. Berchtold, D.A. Keim, and H.P. Kriegel. The X-tree: An Index Structure for High-Dimensional Data. *Readings in Multimedia Computing and Networking*, 2001.

[31] S. Berchtold, C. Böhm, and H.P. Kriegal. The pyramid-technique: towards breaking the curse of dimensionality. *ACM SIGMOD Record*, 27(2):142–153, 1998.

[32] D.H. Lee and H.J. Kim. SPY-TEC: An efficient indexing method for similarity search in high-dimensional data spaces. *Data & Knowledge Engineering*, 34(1):77–97, 2000.

[33] D.H. Lee and H.J. Kim. An Efficient Technique for Nearest-Neighbor Query Processing on the SPY-TEC. *IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING*, pages 1472–1486, 2003.

[34] J.R. Shewchuk. Tetrahedral mesh generation by Delaunay refinement. In *Proceedings of the fourteenth annual symposium on Computational geometry*, pages 86–95. ACM New York, NY, USA, 1998.

[35] X.Y. Li and S.H. Teng. Generating well-shaped Delaunay meshes in 3D. In *Proceedings of the twelfth annual ACM-SIAM symposium on Discrete algorithms*, pages 28–37. Society for Industrial and Applied Mathematics Philadelphia, PA, USA, 2001.

[36] Hang Si. *TetGen A Quality Tetrahedral Mesh Generator and Three-Dimensional Delaunay Triangulator*, 01 2006.

[37] N. M. Patrikalakis. Computational Geometry, 2003. MIT OpenCourseWare lecture notes 13.472J / 1.128J / 2.158J / 16.940J, how to cite?

[38] Y. Luo. Relation between Boolean operators and Euler operators. In *Proceedings on the second ACM symposium on Solid modeling and applications*, pages 477–478. ACM New York, NY, USA, 1993.

[39] D.H. Eberly. Polysolids and boolean operations. Documentation, Geometric Tools, Inc., 1999.

[40] AAG Requicha and HB Voelcker. Boolean operations in solid modeling: Boundary evaluation and merging algorithms. *Proceedings of the IEEE*, 73(1):30–44, 1985.

[41] R.B. Tilove. Set Membership Classification: A Unified Approach to Geometric Intersection Problems. *IEEE Transactions on Computers*, C-29(10):874–883, Oct 1980.

[42] Lutz Kettner. Using generic programming for designing a data structure for polyhedral surfaces. *Computational Geometry*, 13(1):65–90, May 1999.

[43] T.J. Alumbaugh and X. Jiao. Compact Array-Based Mesh Data Structures. In *Proceedings of the 14th International Meshing Roundtable*. Springer, 2005.

[44] R. Chytracek, J. Mccormick, W. Pokorski, and G. Santin. Geometry Description Markup Language for Physics Simulation and Analysis Applications. *IEEE Transactions on Nuclear Science*, 53(5 Part 2):2892–2896, 2006.

[45] P.O.V.R. Team. POV-Ray–the persistence of Vision Raytracer, 2005.

[46] M. Folk. Introduction to HDF5. *NCSA/University of Illinois at Urbana-Champaign http://hdf. ncsa. uiuc. edu/HDF5/papers*, 2000.

[47] Martin Nilsson. Extensible Binary Markup Language. Draft specification, Matroska, 2004.

[48] B.S. Kaliski Jr. *A Layman's Guide to a Subset of ASN.1, BER, and DER.* RSA Laboratories, November, 1993.

[49] D. Gruhl, D. Meredith, and J. Pieper. A case study on alternate representations of data structures in XML. In *Proceedings of the 2005 ACM symposium on Document engineering*, pages 217–219. ACM New York, NY, USA, 2005.

[50] R.E. McGrath. XML and Scientific File Formats. In *2003 Seattle Annual Meeting*, 2003.

[51] ISO/IEC FDIS 19775-1:2008. *Information Technology–Computer graphics and image processing–Extensible 3D (X3D).* ISO, Geneva, Switzerland, 2008.

[52] James W. Cooper. Is Java Fast Enough? *Java Pro Magazine*, Mar 2002.

[53] J. Shirazi. *Java Performance Tuning.* O'Reilly, 2003.

[54] D.H. Besset. *Object-Oriented Implementation of Numerical Methods: An Introduction with Java and Smalltalk.* Morgan Kaufmann, 2001.

[55] T.E. Lindley, T. Laberge, A. Hall, D. Hewett-Emmett, P.J. Walsh, and P.M. Anderson. Sequence, expression and evolutionary relationships of carbamoyl phosphate synthetase I in the toad Xenopus laevis. *Journal of Experimental Zoology Part A: Ecological Genetics and Physiology*, (3), 2007.

COLOPHON

"There is something fascinating about science. One gets such wholesale returns of conjecture out of such a trifling investment of fact."
—Mark Twain

Version Final JT5I-2 rendered from the source by LaTeX 2_ε on May 18, 2009, (BESHAHS).