Framework Technologies & Methods for Large Data Visualization

Tutorial T1, EG2000

21st August 2000

W T Hewitt
University of Manchester

I Curington
AVS Inc

Interlaken, Switzerland

Overview of Today

- Who are we?
- What are we going to talk about?
- Multidimensional data visualization (WTH)
- Volume Visualization (IC)
- Case Studies (IC)
- Parallel Strategies (WTH)
- Parallel Volume Visualization (WTH)
- Optimising Visualization Systems (IC)
- Conclusions (The Future, Q&Q, wrap-up)

How did we get together?

- The International AVS Centre
  - www.iavsc.org
  - Repository of Modules and Projects
  - Over 1,000
  - Free!

IAC manchester

Acknowledgements

- Current and Previous Colleagues
  - Steve Larkin
  - Andrew Grant
  - Peter Kelly
  - Mikael Jern
  - Matt Cooper
  - Marcello Zuffo
  - Paul Lever
  - Jo Leng
  - Mary McDerby
  - ...

Contact Info

Mr W T Hewitt
Manchester Visualization Centre
University of Manchester
Manchester M13 9PL
United Kingdom

Tel: +44 161 275 6095,
Fax: +44 161 275 6800
w.t.hewitt@man.ac.uk
http://www.man.ac.uk/MVC

Contact Info

Mr I Curington
AVS Inc
Montrose House,
Chertsey Boulevard,
Hanworth Lane
Chertsey KT16 9JX
United Kingdom

Tel: +44 1932 566 608
Fax: +44 1932 568 842

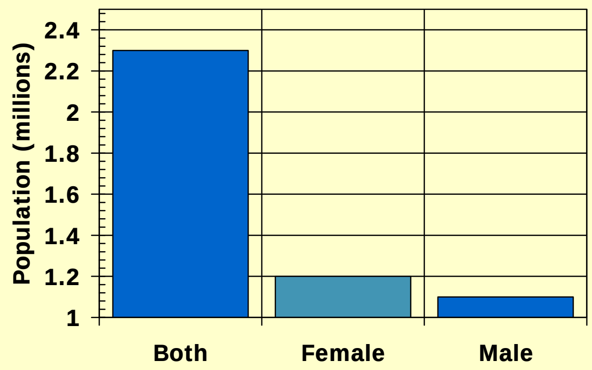
ianc@avs.com

http://www.avs.com

Scientific Aims of Visualization

- It is concerned with [interactively] graphically exploring data to gain an insight into the results
  – Hamming R.W. Numerical methods for scientists and engineers, 1962
- "The purpose of computing is insight, not numbers"
- It differs from presentation graphics:
  – Visualization: understanding the data
  – Presentation: communicating the results

Why Draw a Graph?

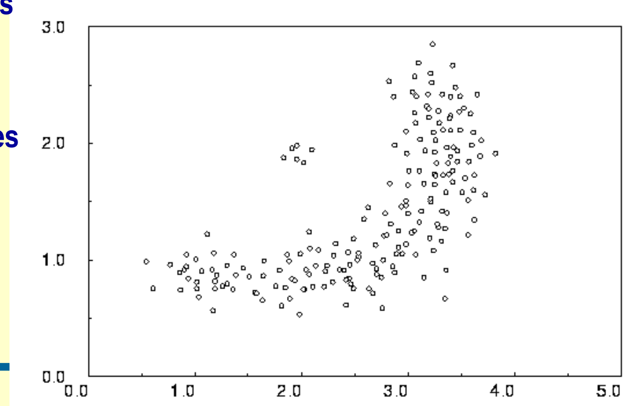


What's Wrong with that Graph?

- Please fill in your answers

A Better Graph

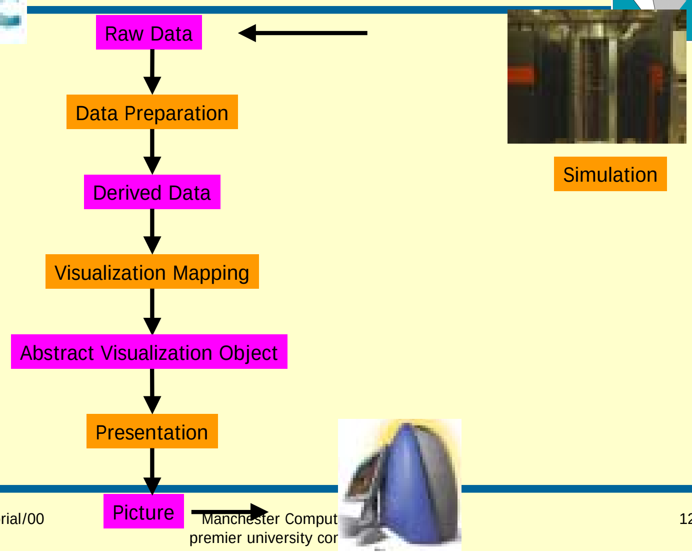
- Summarizes data
- Reveals outliers
- It communicates



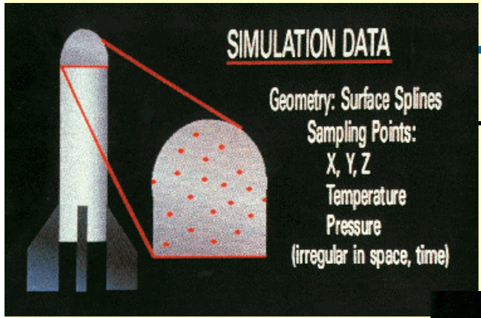
Effective Graphs

- Reasonable amount of data
- Describe behaviour
- Be truthful

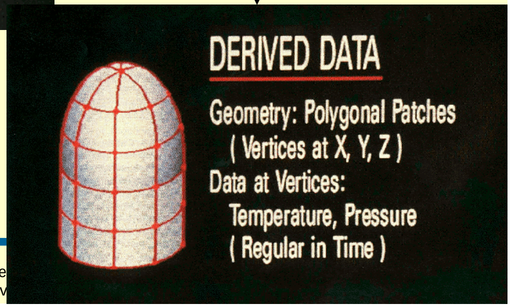
Haber & McNabb Reference Model



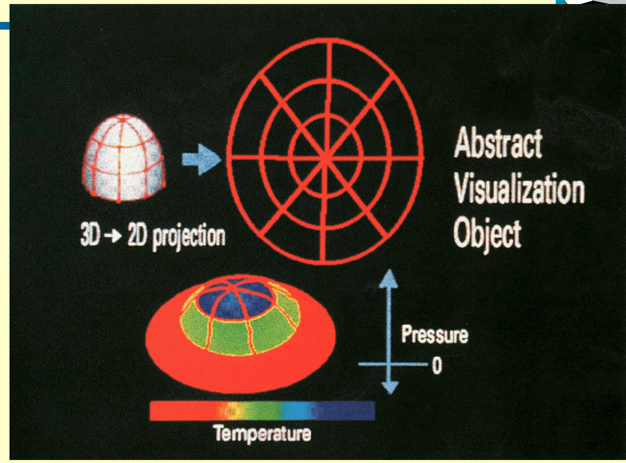
Simulation & Derived Data



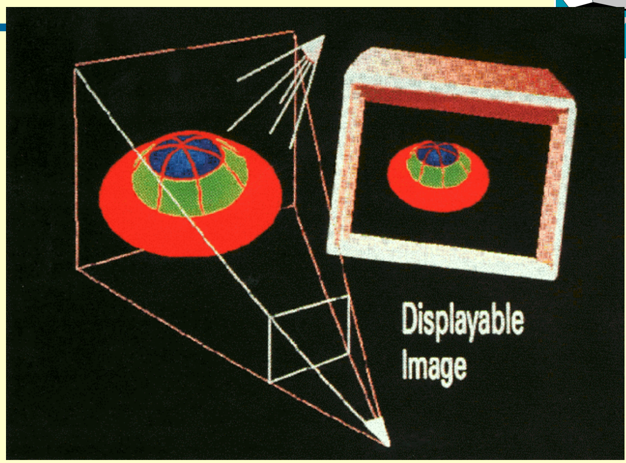
Data Preparation



Abstract Visualization Object



Display



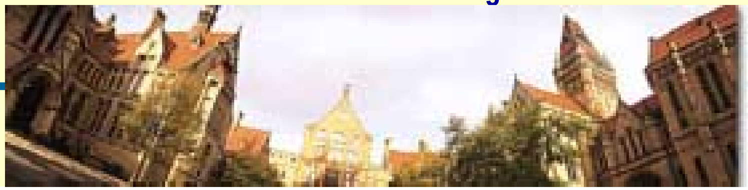
Introduction
to
Manchester Computing

W T Hewitt

Director Manchester Visualization Centre

&

CSAR User Services Manager



University of Manchester

- **Established 1851**

University of Manchester

- **One of the largest in the UK**
  - Student numbers
  - Research Income
  - ...
- **One of the best in the UK**
  - Research quality
  - Graduates getting jobs
  - Teaching

Also in Greater Manchester

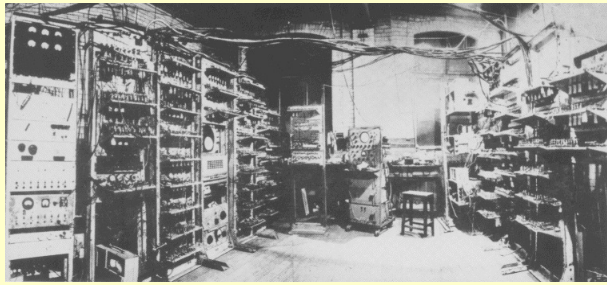
- **UMIST**
  - University of Manchester Institute of Science & Technology
  - Was a faculty of the University of Manchester
  - Now a separate institution
- **Manchester Metropolitan University**
- **University of Salford**

Computing at Manchester

- **Fundamental Computer Science**
  - Next slide
- **Applications of computing, e.g.,**
  - Molecular modelling, chemistry, engineering,...
  - Medicine, social anthropology, ageing process
- **Computer Services**
  - Manchester Computing

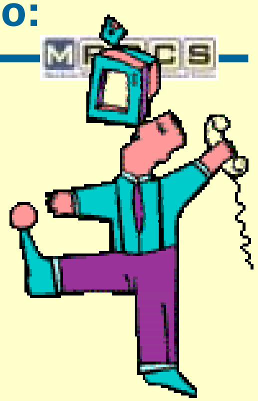
Manchester Computer Innovations

- **World firsts:**
  - Stored program 1948
  - Commercial computer
  - Index registers
  - Virtual Memory
  - Dataflow machine



Manchester Computing provides services to:

- **University of Manchester**
  - including administrative computing
- **UK Academia**
  - Supercomputing (CSAR)
  - Information & data services (MIMAS)
  - Major node in UK Academic Network

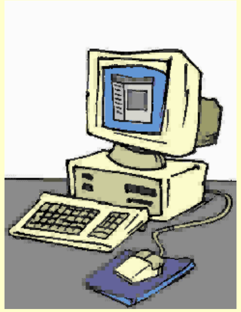


Manchester Computing provides services to:

- **International Services**
  - International AVS Centre
  - MIMAS (formerly MIDAS)
- **Government, Commerce & Industry**
  - Supercomputing
  - Internet exchange point
  - Networks
  - Multimedia (Advanced Telematics Centre)
  - Consultancy
  - R&D

Manchester Computing

- **Main Groups - 210 Staff**
  - MCISO
  - Communications, Operations & Systems
  - National Services
  - Information Services
  - Teaching & Learning Support
  - Manchester Visualization Centre
  - Manchester Research Centre in Computational Science

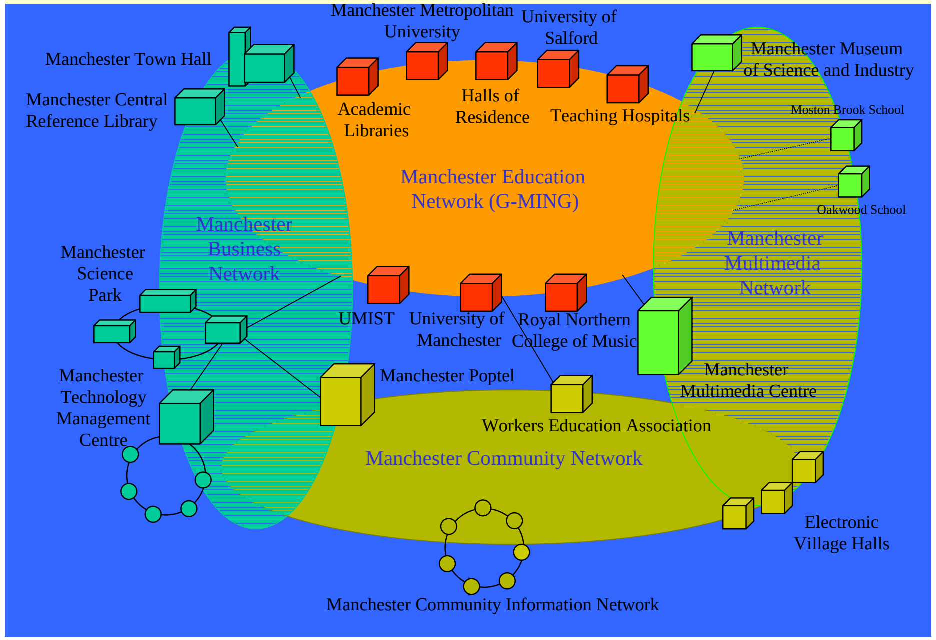


- **10,000 computers on local area network**
- **25,000 users**
- **Used by over 150 Universities**

Networking

- **Lead site for**
  - G-MING
  - Network NorthWest (Cumbria to Keele)
  - JANET
  - SuperJANET
- **Own dial-up service**
  - 3,000 subscribers

Building a City Infostructure

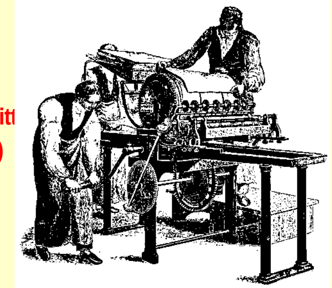


Supporting World Class Research & Teaching

- **Software & Hardware**
- **Services**
  - Web, graphics, multimedia, databases, visualization, email, news groups, video, video conferencing, word processing
- **Support**
  - Helpdesk
- **Consultancy**
- **Training & Education**
  - 30 students/day
  - ECDL
  - IT skills for all new students

Manchester Computing Machine Room

- **Cray T3E-1200E, 816 PE**
- **Fujitsu VPP300**
- **IBM SP2 90 PEs**
- **Origin2000**
  - 16 PE (CSAR), 44 PE (Hillier/Hewitt
  - 16 PE (Brass) 40 PE (Local HPC)
- **Sun E6500 24 PE (MIMAS),**
  - E4500, E4000 (JSTOR)
- **100s PCs for other services**
- **Two large tape Silos (150 TB)**



Supercomputing

- **National HPC services continuously since 1972**
- **National HPC (CSAR)**
- **National Class 3**
- **Local HPC**
- **R&D in HPC, Visualization,**
- **Datasets**

CSAR & CfS: Who and What?

- **Computing Services for Academic Research**
- **provided by Computation for Science (through PFI)**
  - University of Manchester
  - Computer Sciences Corporation
  - Cray Research/Silicon Graphics

MIMAS (formerly MIDAS) Services

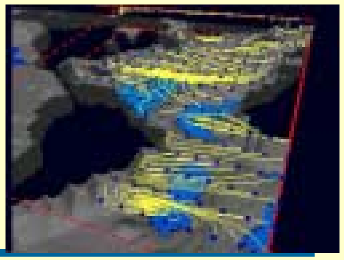
- On-line access to:
  - Electronic Journals (JSTOR Mirror service)
  - Bibliographic data (COPAC)
  - Chemical Information (Beilstein CrossFire)
  - UK Censuses of Population & Surveys
  - Time series databanks (OECD)
  - Digital map data & satellite images (SPOT)
  - ISI Web of science
- Data analysis/manipulation service
- Specialist support services
  - Documentation, training & user support

Research & Development Areas

- Scientific visualization
- Applications of HPC
- Distributed & Meta Computing
- Datasets & data mining
- High Performance Computing Technologies
- Applications of High speed Wide Area Networks
- NURBS, Animation
- 3D Graphics, Radiosity, & Ray-tracing
- WWW & Collaborative working

Manchester Visualization Centre

- Graphics, visualization, multimedia, and image processing services
  - Since 1974
- National Video facility
- The International AVS Centre
- Research & Development

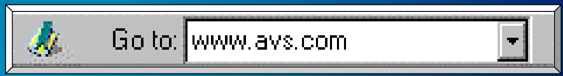


Manchester Research Centre for Computational Science (MRCCS)

- Focus for HPC activities throughout the University of Manchester
  - Computational Chemistry
  - Centre for Novel Computing (Computer Science)
  - Manchester Computing: Manchester Visualization Centre
  - Manchester Computing: CSAR
  - School of Engineering
- Coordinated research programme
- Summer School
- Summer Scholarship programme
- Seminars (with live broadcast)

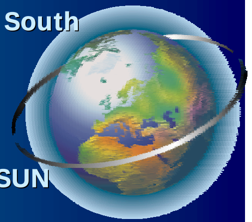
About Advanced Visual Systems

ADVANCED VISUAL SYSTEMS INC.



About AVS

- Established in 1992
- Pioneer/Industry Leader in Data Visualization
  - **Leading edge technology**
- Offices Worldwide
  - **Corporate Headquarters : Waltham, MA**
  - **US Offices : Arkansas, California, Washington, Virginia**
  - **International : Denmark, England, France, Germany, Italy**
  - **Distributors : Austria, Australia, Japan, Korea, South Africa, Switzerland**
- World Class Customer Base
- Strong Industry Partnerships
  - **Compaq, ESRI, Hewlett Packard, Oracle, SGI, SUN Microsystems**



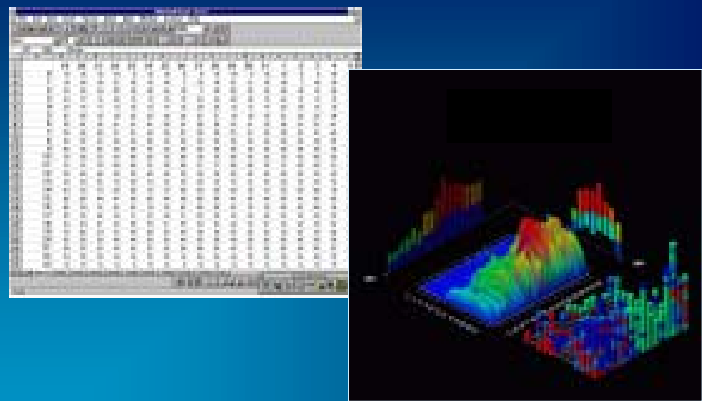
Company Mission

To be the preeminent supplier of Visualization Technology and the Professional Services to assist in its deployment.

Objective : Deliver technology and services to enable people to make better and faster decisions.

Customers : End Users, Internal Developers, Corporate IT organizations, Systems Integrators, Independent Software Vendors and OEMs in selected market segments.

Data Visualization



The science of transforming complicated data into visual insight.

Major Customers

BT
vodafone
JPMorgan
MOTOROLA INC.
HEWLETT PACKARD
ADAC
ADAC LABORATORIES
GEC-Marconi
Ford

AVS Technology Base

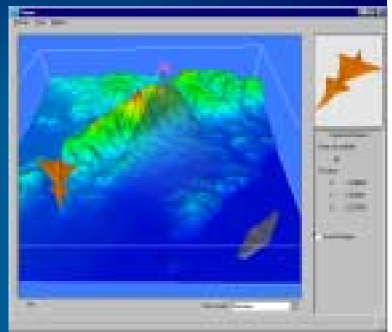
- 3D interactive graphics - 10+ yrs
- Complex visualization algorithms - 10+ yrs
- Artifact-free presentation-quality 2D & 2½D graphics - 15+ yrs
- 250+ person yrs/ 5+ Million Lines of Code
  - **2D & 3D Geometry** **- Animation**
  - **Images** **- Rendering**
  - **Volumes** **- Charting**
  - **Graphing** **- Hardcopy**
  - **Web** **- Data Import**

AVS Visualization Products

- End-User Visualization Applications
  - **AVS5**
  - **AVS/Express Visualization Edition**
  - **Gsharp**
- Product Development Environments
  - **AVS/Express**
- Libraries
  - **AVS/Express**
  - **Toolmaster**
- Components
  - **OpenViz™**

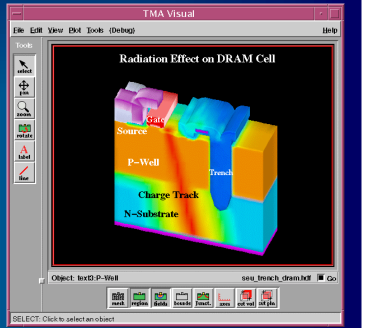
Defense/Intelligence Applications

- Applications:
  - **Remote Sensing, Mission Planning, Radar Analysis, Communications Analysis, Range Instrumentation, Force on Force Simulation**
- Representative Customers:
  - **CSC, E-Systems, Raytheon, TRW, DRA, GEC Marconi**



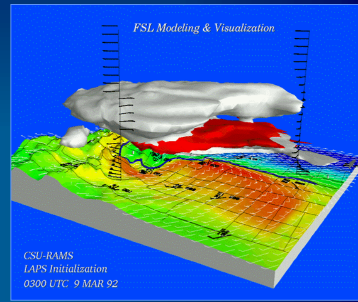
Engineering Applications

- Applications:
  - Computer Aided Design, Electronic Design Automation, Test and Measurement, Fluid Dynamics, Manufacturing Engineering
- Representative Customers:
  - ADAM Net, AEA Technology, CIRA, FIAT-Avio, Ford, Technology Modeling Assoc.,



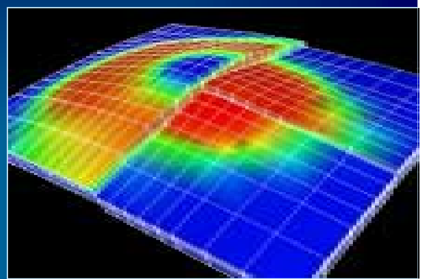
Environmental Applications

- Applications:
  - Weather Forecasting, Climate Control, Air Quality, Hydraulic Modeling, Ocean Studies, Resource Mgmt., Geological Surveys, Site Remediation
- Representative Customers:
  - Deutsch Wetterdienst, NOAA, Delft Hydraulics, GE/NBC, Ctech, Danish Hydraulic Inst.



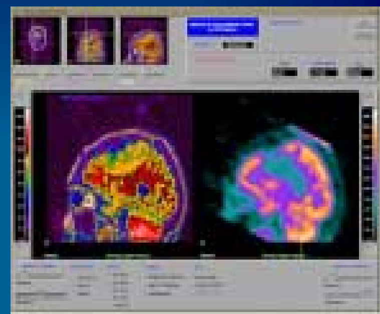
Oil & Gas Applications

- Applications:
  - Reservoir Modeling, Seismic Interpretations, Well Log Analysis
- Representative Customers:
  - CMG, Mobil, Shell, AGIP, BP, GECO, Western GEO, Schlumberger, Exxon, PGS Tigress



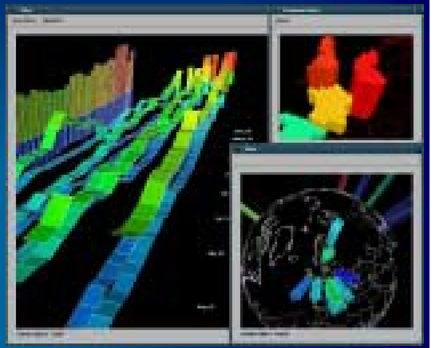
Medical Applications

- Applications:
  - Treatment Planning, Medical Diagnostics, Microscropy, Biomedical Engineering
- Representative Customers:
  - RSA, ADAC, Duke University, Focus Graphics, Integrated Medical Images (iMIP), John Hopkins University, Radionics Software, University of Washington



Telecom Applications

- Applications:
  - RF Propagation Modeling, Network Monitoring and Control, Network Planning and Simulation
- Representative Customers:
  - Ericsson, GEC Marconi, Motorola, DeTeMobil, Vodafone, CRIL, Bristish Telecom, MCI



Techniques for Multidimensional Data

W T Hewitt
Manchester Visualization Centre
University of Manchester

Overview

- What is multidimensional data (mDv)?
- A look at the problems and some examples
- Techniques and use
- Some visualization systems which cater for multidimensional data
- Conclusions and summary

What is mDv?

- M dimensional data, V data components but also referred to as:
  - Multidimensional multivariate data
  - range data or extremes
- Some examples are:
  - Traditional areas such as Census data
  - We will also treat 2nd order tensors as a class of mDv
  - National Power: 300 companies, 13 cost components, each bidding to sell electricity to them in units of 30 minutes. How do you provide timely analysis?
  - Sociology data: a researcher has collected data over the last 20 years from people who held office in Medieval times. It contains person, year and position.
    - He needs to analyse job movement, promotion/demotion, and kinship/nepotism.

Stock Exchange Data

- Vast amounts of data which changes every day and has complex relationships.

Traditional Techniques

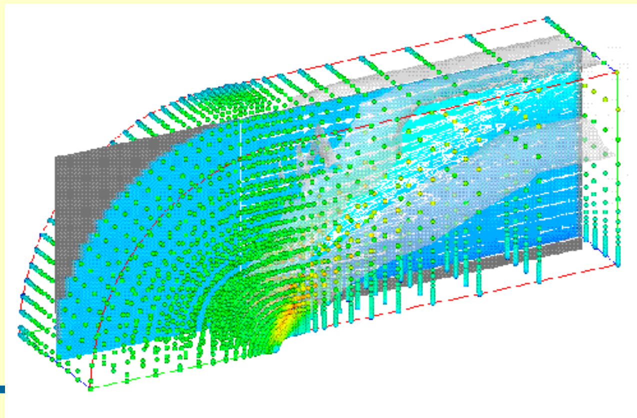
- Why not use these techniques for multidimensional data?
- These techniques are very useful for 2D, 3D scalar and vector datasets
- Problems still occur from perceptual issues:
  - arrows in 3D
  - Colour
  - locating/probing values in 3D space
- But in the majority valid assumptions can be made from the figures produced for this class of data

Applying traditional techniques

- The multidimensional data components can be viewed separately using these techniques
- Correlations can be made by stacking or overlaying results
- Careful use is need as they can produce cluttered and incomprehensible results
- We will see more examples in the techniques section

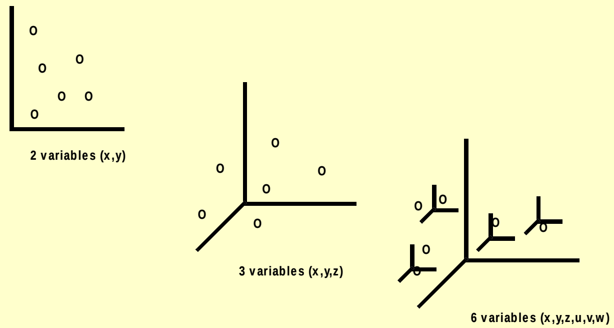
Using Traditional Techniques

- Flow of air over a fin: density, stagnation and momentum



Coping with greater than 3D

- It becomes hard to navigate, relate and compare values
- We will introduce some other techniques



Some techniques

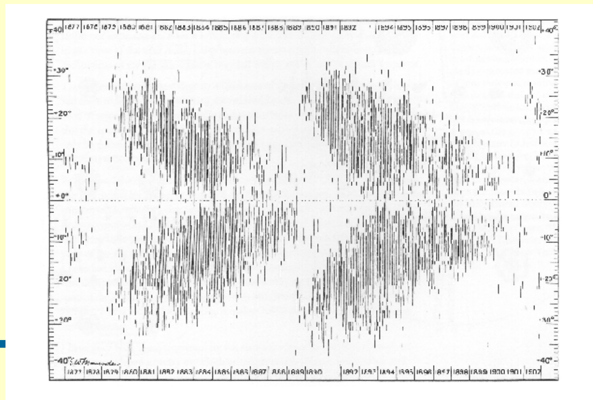
- Glyphs [1], [2], [4], [5], [6], [16]
- Textures [3], [9], [13], [17]
- Tables and Stacked Plots [2], [12]
- Scatterplots [2], [7]
- Andrews curves [10]
- Permutation Matrix [8]
- Parallel coordinates [11]
- Data Sonification [18], [20]
- Virtual Reality [22]

What are you looking for?

- The techniques sometimes produce results which appear to be very cluttered
- The viewer is specifically looking for:
  - unexpected results or anomalies (spotting a stranger)
  - grouping or clusters
  - identifying patterns or trends and correlations
- These techniques require the viewer to be trained in their use and application
- Their effectiveness is very dependent upon the viewer

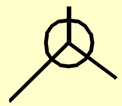
Sunspots

- The sunspots travel towards the equator of the sun over time. The figure only shows the vertical dimension of each sunspot.



Glyphs

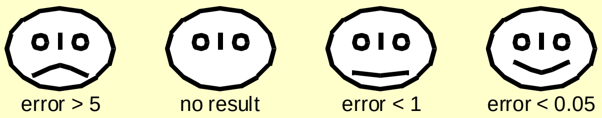
- Graphical icons (glyphs) are not new



- 1957: Edgar Anderson - circular icons with rays
- 1966: Pickett White - triangle with sides and orientation related to different variables
- 1973: Chernoff - used a traditional 2D scatterplot with facial characteristics to represent 3,4,5,...,22 variables
- Referred to as “Chernoff Faces”

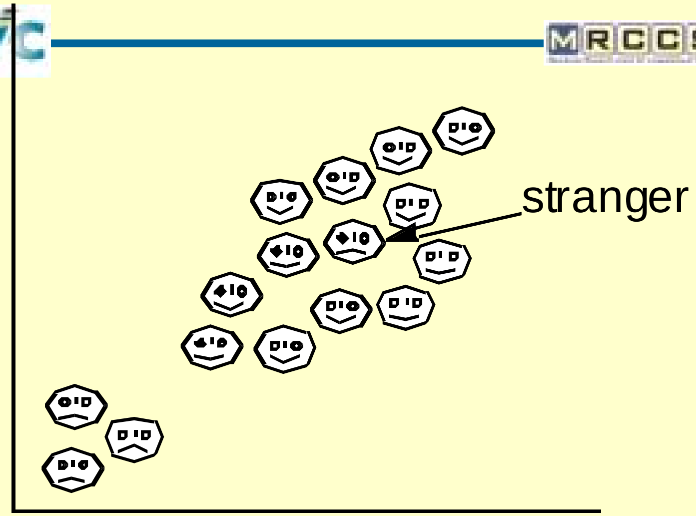
Chernoff Faces

- **The variations are normally grouped into distinct classes:**



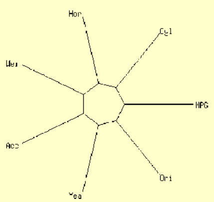
Encoding error as variation of the mouth

- **Allows the viewer to try and spot trends or strangers as it relies on the fact we are good it recognising faces**



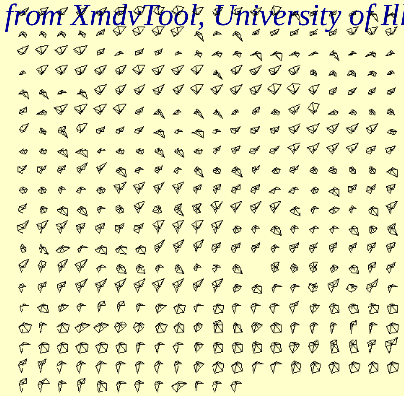
Star Glyphs

- **Each dimension in the dataset is represented as a "prong" in the star, [19]**
- **For each datapoint a star is drawn with the size of the "prongs" representing the value in each dimension for that particular point**



Star Gylphs

- *Produced from XmdvTool, University of Illinois*



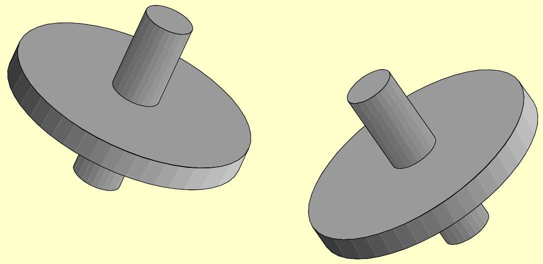
Haber Glyphs

- Used to visualize the stress-strain in a tensor
- Split the tensor into symmetric and anti-symmetric parts
  - J(s) is the stress-strain tensor

$$J = J^{(s)} + J^{(a)}$$

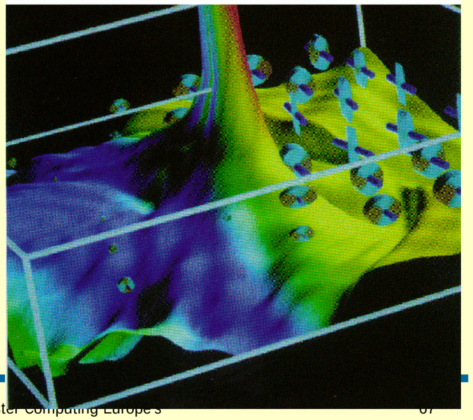
- Glyph is a cylinder and an ellipse
- Cylinder axis direction shows major principal direction, ellipse axes show the other two
- Cylinder and axis lengths show stretching in each axis.

Haber Glyph



Example of Haber Glyphs

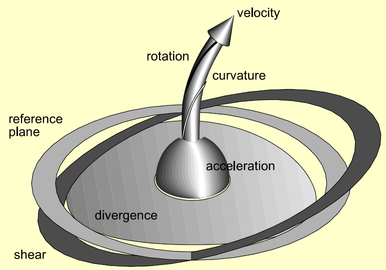
- Haber R B, "Visualization Techniques for Engineering Mechanics",



de Leeuw and van Wijk glyphs

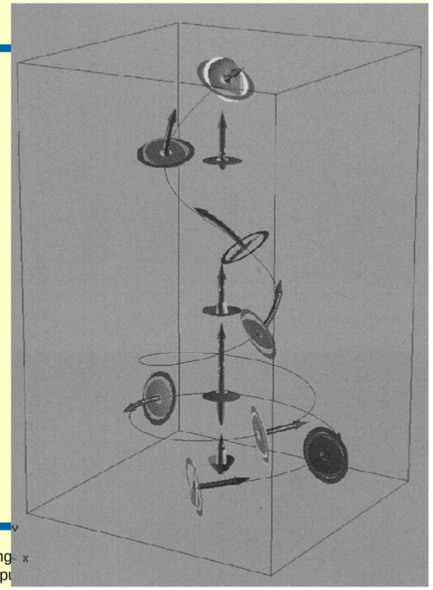
- Visualise the tensor field in the context of the associated velocity field
- Steady state flows only
- Best used as a probe or small multiple
- Constructs local coordinate axis as with Haber glyphs
- Decompose tensor into *parallel* & *perpendicular* components
- Extract further components from these
  - acceleration, shear, curvature (parallel)
  - torsion, divergence (perpendicular)

de Leeuw and van Wijk glyphs



de Leeuw and van Wijk glyphs

- Flow in a vortex



Textures

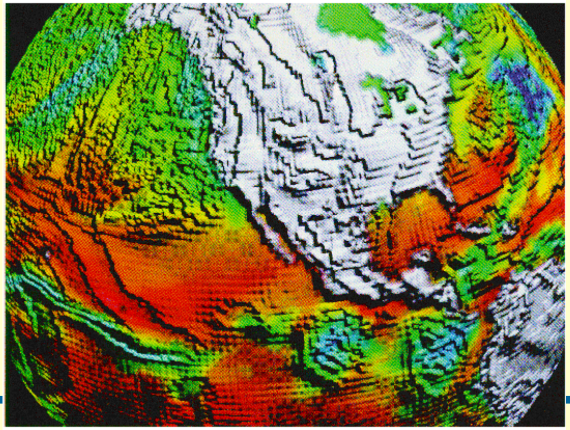
- In addition to surface height, colour and vectors we can use texture (bump mapping)
- Bump map is a collection of bumps (texture) used to add additional
- information to a graphical primitive
- Interactive adjustment of parameters is desirable to obtain best results
- Careful use is needed as additions to an already rough surface can be distracting

Climate Model Example

- Climate model produces a number of components:
  - wind velocity
  - heat (outgoing long wave radiation from earths surface)
  - surface height
- We want to correlate these components:
  - Reference map (surface plot): surface heights
  - Colour of Reference map: heat (blue - red)
  - Bump mapping: wind velocity (smooth - rough)

Climate Model using Texture

– Crawfis R A, Allison M J, LLNL, [13]

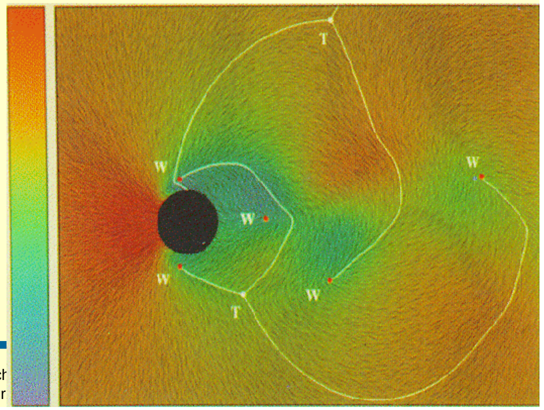


More use of textures

- Texture maps can be used to represent more information about vectors and tensors than just magnitude, [9], [17]
- The process is called "Line Integral Convolution"
- You take:
  - a vector field defined on a cartesian grid
  - a texture map of the same dimensions
- "The output image is a one-one correspondence of a 1D convolution of a filter kernal and texture pixels along a local streamline in the vector field"
- More simply the texture is "smeared" in the direction of the vector field

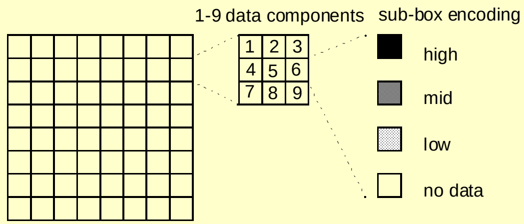
Texture for tensor fields

- texture is the eigen vector of the stress tensor
- colour is the magnitude of the compressive force
  - Demarcelle T, Hesselink L, Stanford University, [9]



Tables

- Each point in the dataset is represented as a rectangle
- The rectangle contains encodings for the value of the point in each particular dimension in the dataset [1]



Magnetosphere and solar wind

- The readings were taken every hour over a number of days from NASA Goddard Space Flight Center.
- 13 parameters of magnetosphere and solar wind data
  - Beddow J, Microsimulations Research, [1]

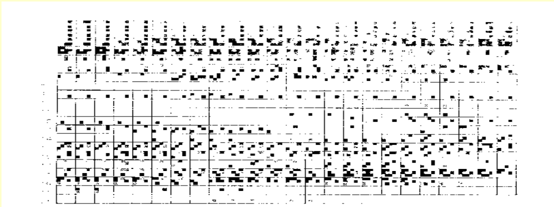
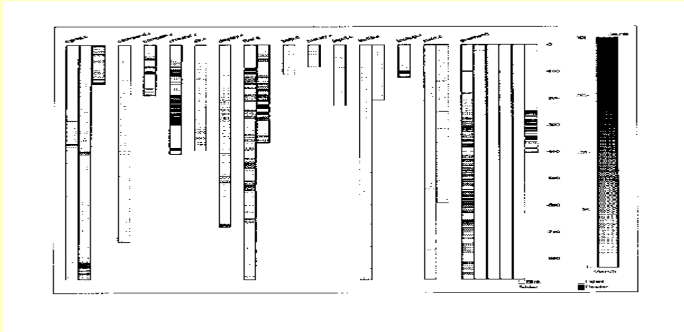


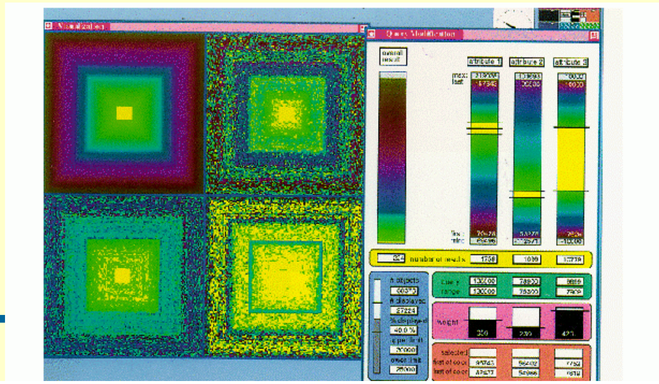
Table - Profiling Code

- Displays "hot-spots" in programming code
  - Eick S G, Steffen J L, AT&T Bell Labs [12]



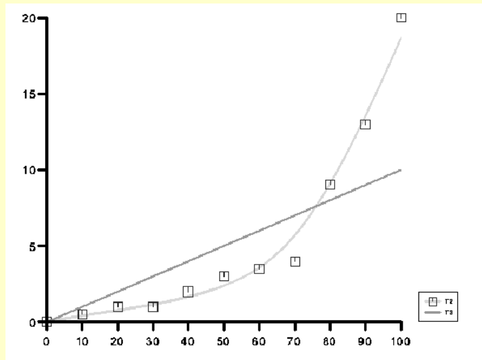
Querying Databases

- Each data item in the database is represented as a pixel where the colour indicates the relevance for the query
  - Kiem D A, Kiegel H P, Seidl T, University of Munich [29]



Simple dataset

- We will use a simple dataset of temperature, pressure and velocity:

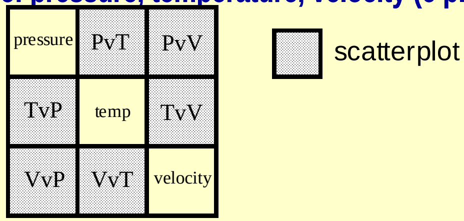


Complex dataset

- **The dataset is taken from a selection of 406 different cars:**
- **(http://www.hensa.ac.uk)**
  - This data set is a version of the CRCARS data set of Donoho, David and Ramos, Ernesto (1982), ``PRIMDATA: Data Sets for Use With PRIM-H”
- **8 Variables**
  - MPG, # cylinders, engine displacement, horsepower, vehicle weight, time to accelerate from O to 60 mph, model year
  - origin of car (1. American, 2. European, 3. Japanese)

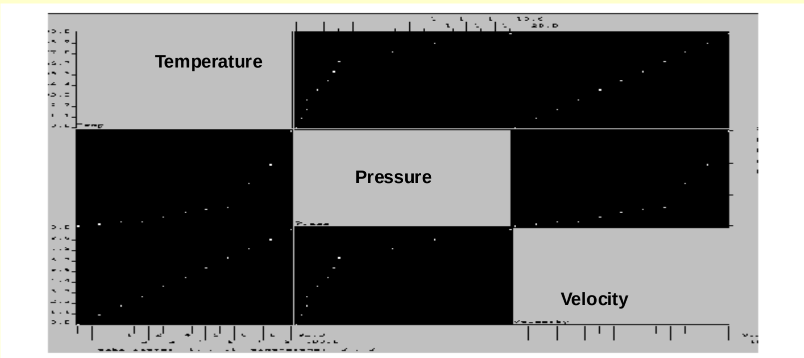
Scatterplot Matrix

- **Scatter plot shows the relationship of 2 variables**
- **Addition of colour can represent a 3rd variable**
- **A scatterplot matrix of n variables are projected onto n*(n-1) scatter plots**
- **For example: pressure, temperature, velocity (6 plots)**

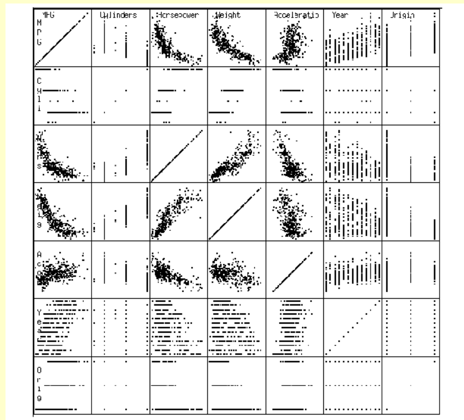


Scatterplot Matrix – Simple

- **Shows pressure, temperature, velocity**



Scatterplot Matrix – Complex



Andrews Curves

- Introduced by D Andrews in 1972
- Each multidimensional point x (x1,x2,...,xm) is mapped to a periodic function G(t):

$$G(t) = \frac{F1}{\sqrt{2}} + F2\sin(t) + F3\cos(t) + F4\sin(2t) + F5\cos(2t) + \dots$$

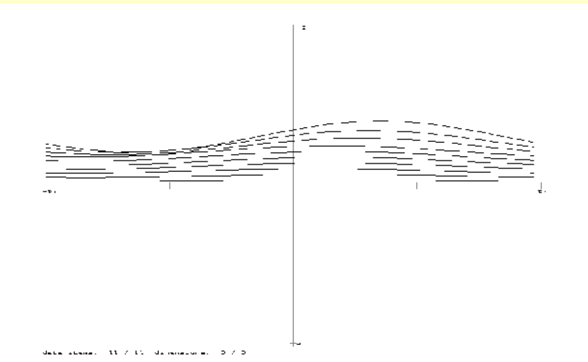
- The curves are plotted over the range -PI..PI

What do the curves show?

- Produces an iconic representation of each point through multidimensional space
- Clusters of points map to similar shaped curves
- It is not possible to pinpoint single data components i.e., all the data components are combined into one function

Andrews curves – Simple

- Points through pressure, temperature, velocity

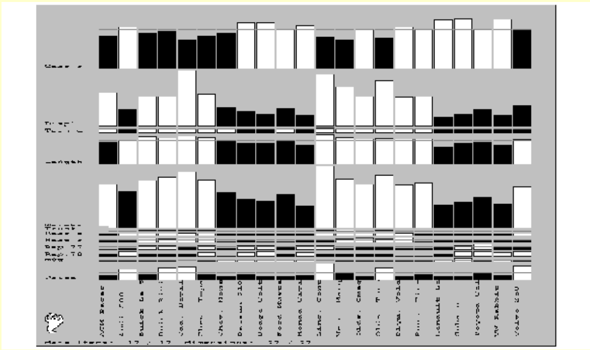


Permutation Matrix

- The dataset is transformed into a matrix of graphical elements where the rows and columns correspond to:
  - dimensions in the dataset
  - points in the dataset
- The chart has three main parts:
  - a line indicates mean value
  - black bars are values below mean
  - white bars are values above mean
- This matrix reveals structure of the whole dataset
- Individual points and dimensions can be identified

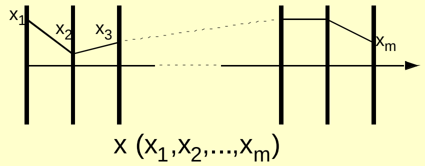
Permutation Matrix

- Shows individual cars and their characteristics



Parallel Coordinates

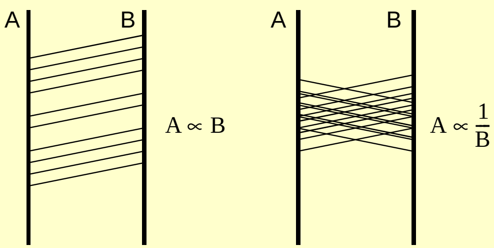
- Introduced by Alfred Inselberg
- Organise each axis vertically and for each multidimensional point x (x1,x2,...,xm) mark the appropriate axis
- Join the marks with line segments



- Therefore a m dimensional point is represented as a line through m parallel coordinates

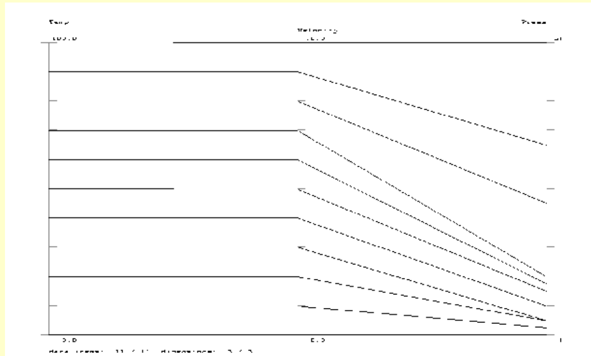
What are you looking for?

- The results seem extremely cluttered
- Systems which provide this technique allow interactive marking and highlighting of groups of lines
- There are some patterns/shapes to look for:



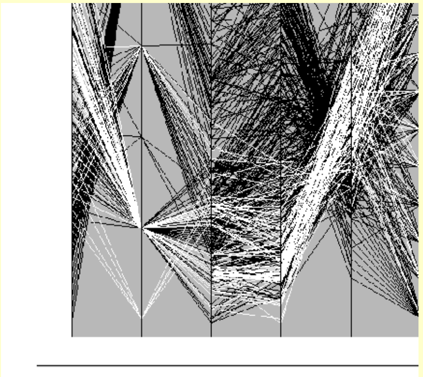
Parallel coordinates – Simple

- Shows pressure, temperature, velocity



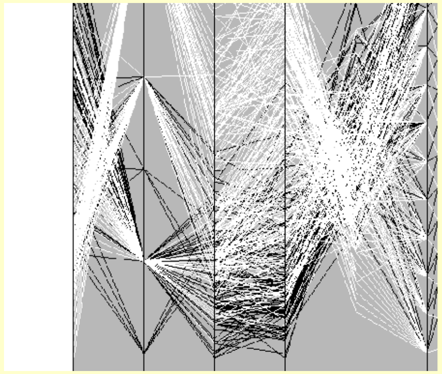
Parallel coordinates – Complex

- We have highlighted all Japanese cars



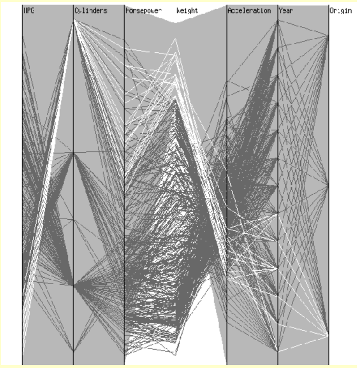
Parallel coordinates

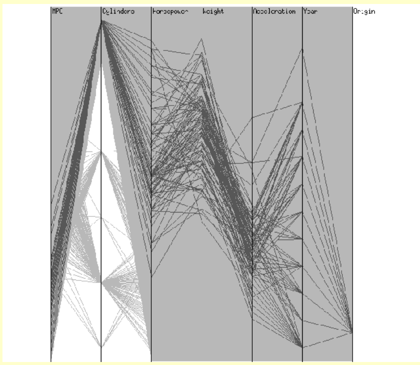
- We have highlighted all US cars



Parallel coordinates

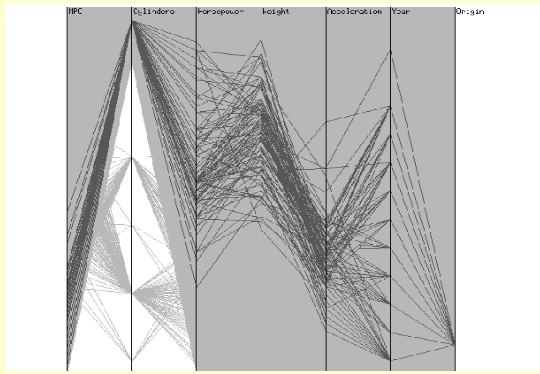
- We now look at top weight and top cylinder





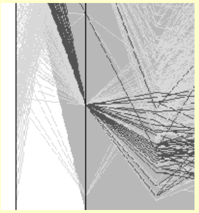
Parallel Coordinates

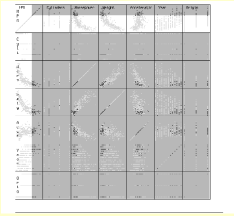
- ...and now we take a look at low MPG



Brushing

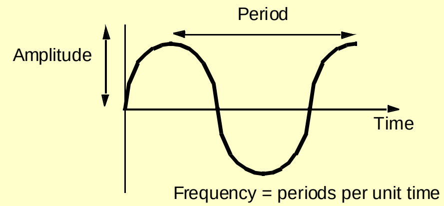
- In all the techniques which we have seen we make extensive use of facilities to highlight data which falls between certain ranges
- The XmdvTool [21] implements N dimensional brushing
- Links can be made between views of same data





Data Sonification

- The use of sound to complement a graphical representation
- But what is sound?
- It is the sensation of pressure variations in air caused by a vibrating source:



- The assimilation to data is simple, or is it?

Sound Attributes

- Pitch
  - logarithmic changes in frequency = linear changes in pitch
  - intuitive for relating to magnitude of a scalar component
  - similar problems as with colourmaps; adjacent values are difficult to distinguish
- Loudness
  - variations in amplitude
  - it is not linear as it is also affected by frequency and timbre changes
- Timbre
  - waveform: different instruments playing the same pitch/loudness
  - used to differentiate between data components

Sound Attributes

- Location
  - physical location of the sound source which is affected by acoustics of the surrounding environment
  - can provide locational cues to results
- Rhythm
  - music is organised around a periodic event rate or pulse
  - can be used to represent temporal separation between time stamped events or behavioural cycles
- Duration
  - hard to distinguish unless exaggerated
  - not a quantitative measure but useful to identify outliers or activity lifetimes

Sound Attributes

- Melody
  - "the first thing remembered, the last thing forgotten"
  - What constitutes a melody is the subject of considerable research
  - Certain patterns of notes are more "melodic" than others
  - Therefore the choice of scale or starting pitch is significant
- Conclusions
  - sound is as complex a medium as other more traditional ones for visualization e.g., colour
  - There are many pitfalls
  - You have to be aware of the "tone deaf" equivalent of a "colour blind" user

Some Real Examples

- Analysis of climate data - a probe samples data components and assimilates them to sound:
  - wind: varying the pitch of a siren
  - rain: varying the amplitude (loudness) of the sound of "rain"
- Audio Cues to signal an event or condition has been reached:
  - data component outside a specified range
- Application to Stanford ParalleL Applications for SHared memory benchmark suite (SPLASH)
- type of process (system, network, application) = pitch
  - process's time quantum = duration
  - processor = instrument

Systems

- The application builders (Modular Visualization Environments) have little support for these techniques:
  - AVS has some public domain modules for sonification
  - IBM Data Explorer: Some of Inselberg's group are working in this area
- Systems being developed primarily for this analysis e.g.,
  - VisuLab: Hans Hinterberger, Institute for Scientific Computing, ETH, Zurich
  - XmdvTool: Computer Science Department, Worcester Polytechnic Institute, MA, US (ftp://ftp.wpi.edu:/contrib/Xstuff/XmdvTool2.tar.gz)
  - Porsonify: A Sonification toolkit, Madhyastha & Reed, Dept. of CS, University of Illinois
  - XmdvTool and Visulab were used to produce some of these figures and we wish to acknowledge the developers of these software packages.

The Future - Virtual Reality?

- This can integrate traditional techniques for visualization with other less familiar media
  - Sound
  - tactile (touch)
  - Olfactory (smell)
  - taste?
- Some more current and real examples are:
  - NASA Ames Virtual Wind Tunnel
  - CAVE: The Virtual Reality Theatre
  - Advanced Interfaces Lab: Dept. of CS, University of Manchester

References

1. Beddow J, “Shape Encoding of Multidimensional Data”, Proceedings of IEEE Vis ‘90, pages 238-246
2. Tufte E R, “Envisioning Information”, Graphics Press, 1990
3. Cabral B, Leedom L C, “Imaging Vector Fields using Line Integral Convolution”, SIGGRAPH ‘93 Proceedings, pages 263-272
4. de Leeuw W C, van Wijk J J, “A Probe for Local Flow Field Visualization”, Proceedings of IEEE Visualization ‘93, pages 39-48
5. Haber R B, “Visualization Techniques for Engineering Mechanics”, Computing Systems in Engineering I, 1990, pages 37-55
6. Ellson R, Cox D, “Visualization of Plastic Injection Moulding”, Simulation 51, 5, 1988, pages 184-188
7. Cleveland M, “Elements of Graphing Data”, Wadsworth, 1985
8. Bertin J, “Semiologie graphique”, Editions Gauthier-Villars, 1967

References

9. Delmarcelle T, Hesselink L, “The topology of 2nd order tensor fields”, Proc Vis ‘94
10. Andrews D, “Plots of Higher Dimensional Data”, Biometrics, March 1972
11. Inselberg A, “Parallel Coordinates - A Tool for visualizing multi-dimensional geometry”, Proceedings of IEEE Visualization ‘90
12. Eick S G, Steffen J L, “Visualizing Code Profiling Line Oriented Statistics”, Proceedings of IEEE Visualization ‘92
13. Crawfis R A, Allison M J, “A Scientific Visualization Synthesiser”, Proc. Vis ’91
14. Gardiner V L, Lazarus R B, Stein P R, “Solutions of Diophante Equation x3 + y3 = z3 - d”, Math Comp 18, 1964,
15. Nielson G M, “Modelling and Representing Multivariate Data”, Course Notes on Advanced Techniques for Scientific Visualization , SIGGRAPH ’94.

16. Chernoff H, “The use of faces to represent points in k-dimensional space graphically”, Journal of American Statistical Association 76, June 1973
17. Forsell L K, “Visualizing Flow over curvilinear grid surfaces using Line Integral Convolution” Proceedings of IEEE Visualization ‘94
18. Scaletti C, Craig A B, “Using sound to extract meaning from complex data”, http:// www.ncsa.uiuc.edu/VR/VR/Papers/sound.ps
19. Siegel J H, Farrell E J, Goldwyn R M, Friedman H P, “The surgical implications of physiological patterns in myocardial infractions shock”, Surgery Volume 72, 1972
20. Madhyastha T M, Reed D A, “Data Sonification: Do you see what I hear?”, http:// bugle.cs.uiuc/edu/Papers/IEEEsound.ps
21. Ward M O, “XmdvTool: Integrating Multiple Methods for Visualizing Multivariate Data”, Proceedings of IEEE Visualiztion ‘94

References

22. Bryson S, Levit C, “The virtual windtunnel: an environment for the exploration of 3D unsteady fluid flows”, CG & A, 1992
23. Keller P R, Keller M M, “Visual Cues - Practical Data Visualization”, IEEE Computer Society Press, 1993.
24. Kaufman A E, “Proc. Vis ‘90”, IEEE Computer Society Press 1990
25. Kaufman A E, Nielson G M, “Proc. Vis ‘92”, IEEE Computer Society Press 1992
26. Nielson G M, Bergeron D, “Proc. Vis ‘93”, IEEE Computer Society Press 1993
27. Bergeron R D, Kaufman A E, “Proc. Vis ‘94”, IEEE Computer Society Press 1994
28. Earnshaw R A, Watson D, “Animation and Scientific Visualization - Tools and Applications”, Academic Press, 1993
29. Keim D A, Kriegel H P, Seidl T, “Visual Feedback in Querying Large Databases”, Proc. Vis ‘93, IEEE Computer Society 1993.

Volume Visualisation (1)

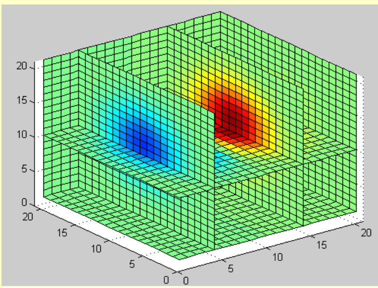
W T Hewitt
Manchester Visualization Centre
Manchester Computing
University of Manchester

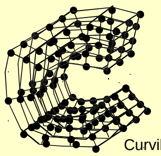
Volume Visualization

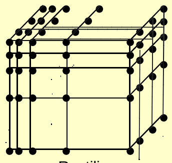
- Volume Visualisation Terminology
- Basic Techniques
- Some Algorithms
- Conclusions and Examples

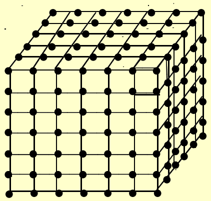
What is a Volume?

- In general 3D Scalar Fields



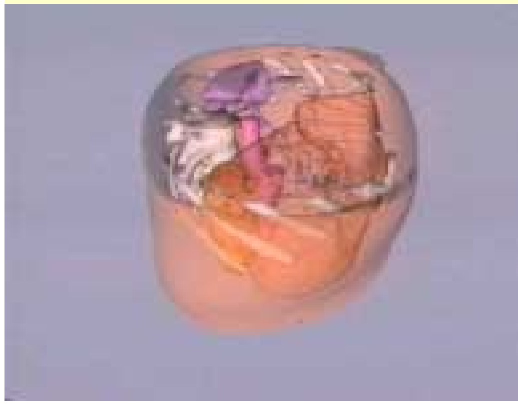

Curvilinear

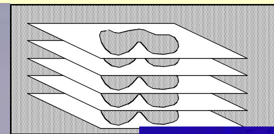

Rectilinear

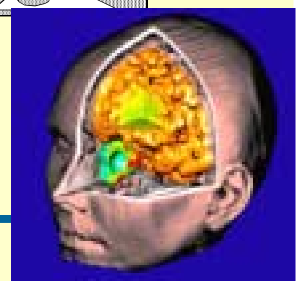

Regular

Why Volume Visualisation?

- Concerned with the representation & analysis of volume data
- To see internal structure/topology for minimal cost



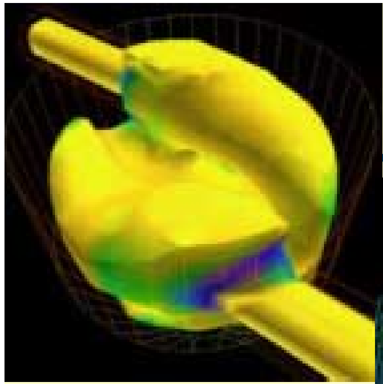


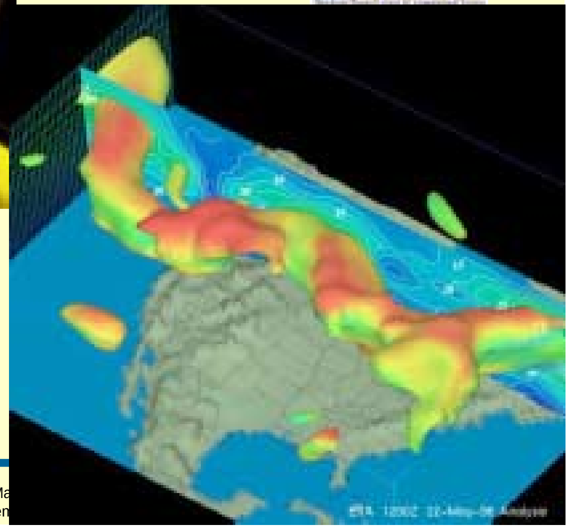


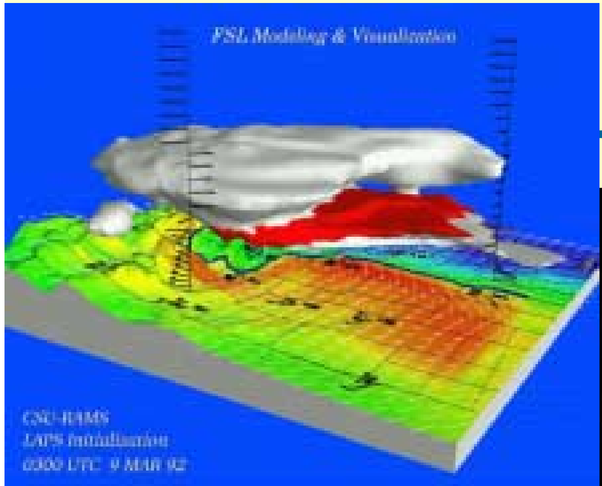
Introduction
Application Areas

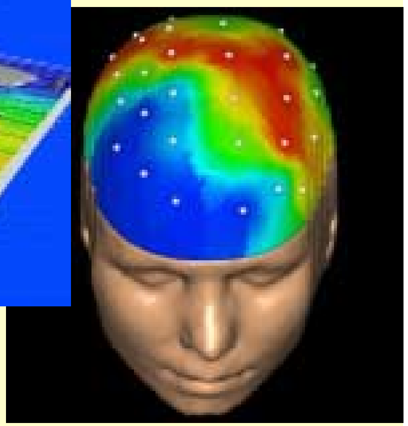
- **Medical**
  - Craniofacial, clinical diagnosis, radiation treatment planning, non-invasive surgery, medical education, neurology
- **Molecular modelling**
- **Non-destructive Evaluation**
- **Astrophysics, Meteorology**
- **Confocal Microscopy**
- **Seismic Geophysics Interpretation**

- **Typically data volumes are very large**

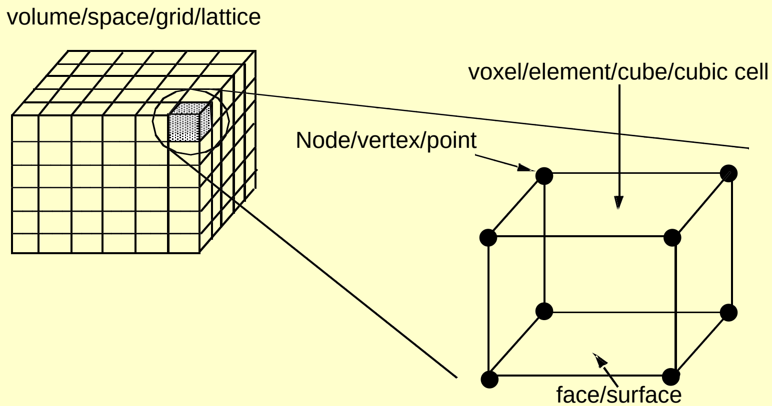






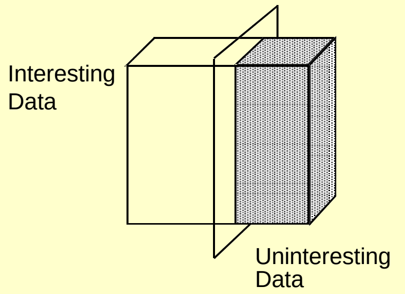


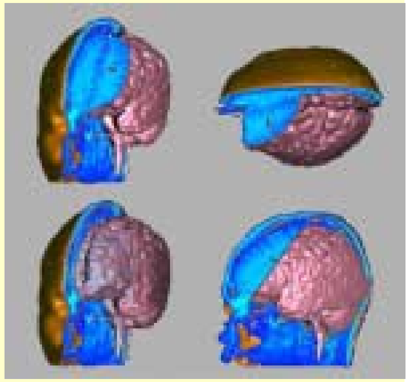
Volume Visualisation Terminology



Volume Visualisation Terminology

Extent Planes/ Cut Planes/ Excavate



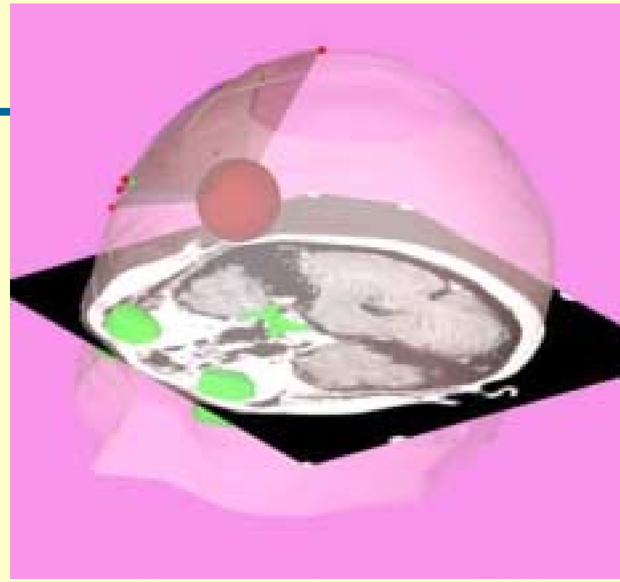


Volume Visualisation Terminology

- **Projection Method**
- **Feed Forward/ Object Order traversal**
  - The date volume is traversed and each voxel in the volume is projected onto the image plane
- **Feed Backward/Image order traversal**
  - The pixels in the image plane are traversed and imaginary rays are cast through each pixel into the volume. The path of the ray determines the value of the pixel

Volume Visualisation Terminology

- **Opacity**
  - A Material Property that prevents light from passing through an object (a=1)
- **Transparency**
  - A Material property that allows light to pass through an object (a=0)
- **Translucency, semi-transparency**
  - Graded or blurred transparency (0<a<1)

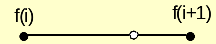


Volume Visualisation
Basic Techniques

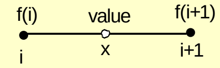
- Resampling Methods
- Gradients
- Lighting and Shading
- Colour Classification
- Opacity Classification

Basic Techniques
Resampling Methods

Nearest Neighbour

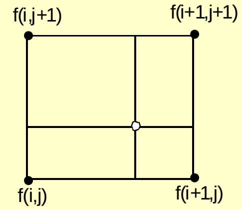


Linear Interpolation



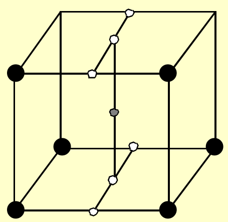
x=i+(value-f(i))/(f(i+1)-f(i))

Bilinear Interpolation



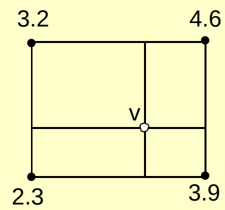
Basic Techniques
Trilinear Interpolation

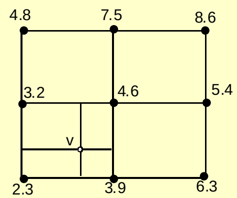
- Often done several times for each voxel
- Involves seven linear interpolations



Basic Techniques
Resampling Methods

- The Bigger Picture



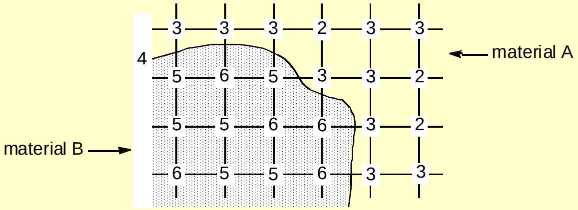


v= 4.11

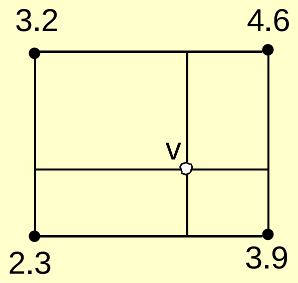
v=?

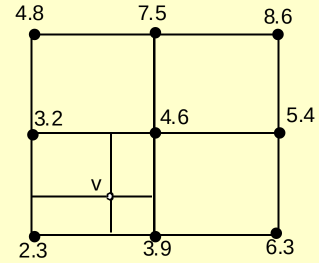
Basic Techniques
Gradient Approximations

- Used to approximate surface normals for shading calculations
- Also to determine surface "strength"



Basic Techniques
Resampling Methods





v= 4.11

v=?

Basic Techniques
Gradients

- Generally calculated using central differences

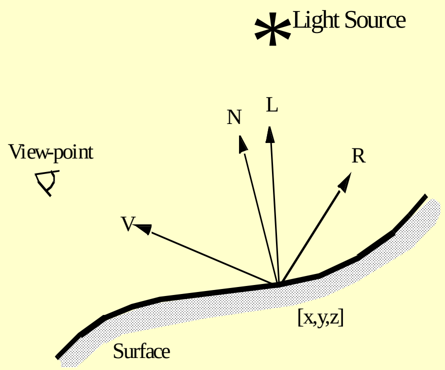
$$G_x(i,j,k) = \frac{f(i+1,j,k) - f(i-1,j,k)}{\Delta x}$$

$$G_y(i,j,k) = \frac{f(i,j+1,k) - f(i,j-1,k)}{\Delta y}$$

$$G_z(i,j,k) = \frac{f(i,j,k+1) - f(i,j,k-1)}{\Delta z}$$

- Surface Normal is then given by $\vec{N} = \frac{\vec{G}}{|\vec{G}|}$

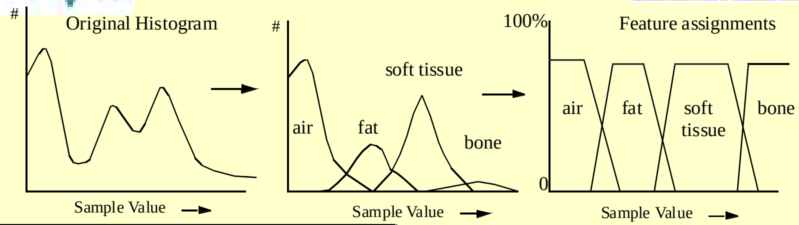
Basic Techniques
Lighting and Shading



$$I = K_a I_a + K_d \Sigma(N.L_j)I_j + K_s \Sigma(R_j.V)^n I_j$$

I = Intensity
K = Weight Constant
L = jth Light Vector
N = Surface normal vector
R_j = jth reflected light vector
V = View vector

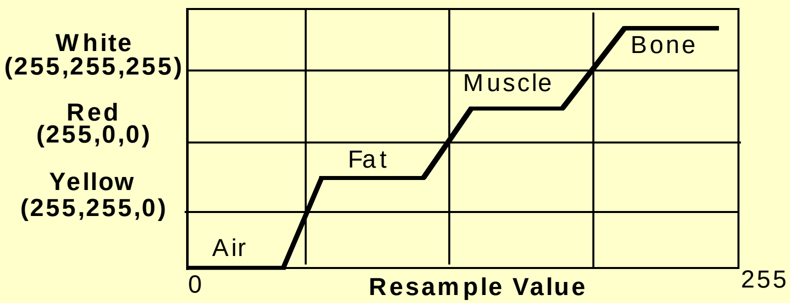
Data Classification & Feature Assignment



Computing Europe's
sity computer service

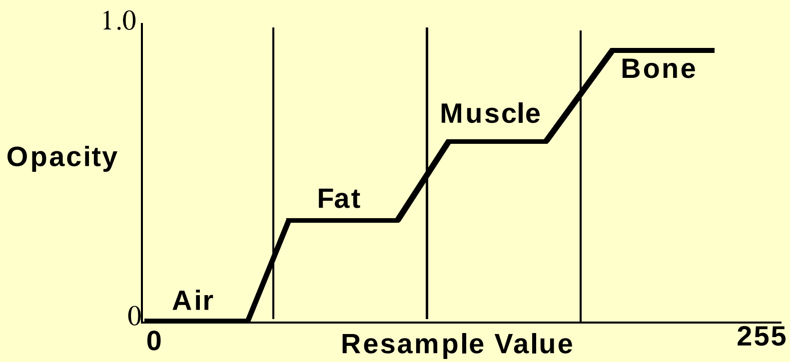
Basic Techniques Colour Classification

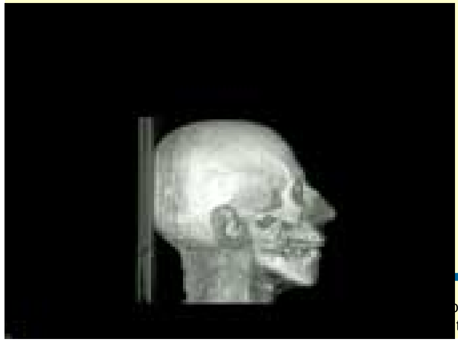
Colour= $C_{xfer}(f(i,j,k))$

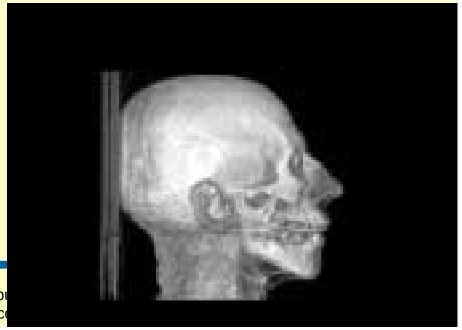


Basic Techniques Opacity Classification

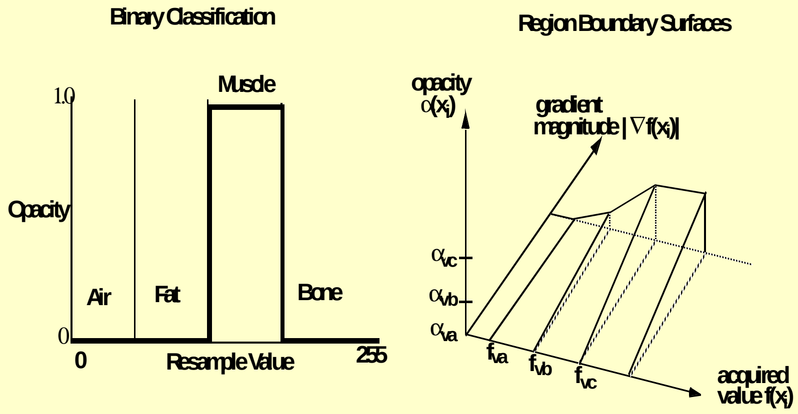
Opacity = $\alpha_{xfer}(f(i,j,k))$



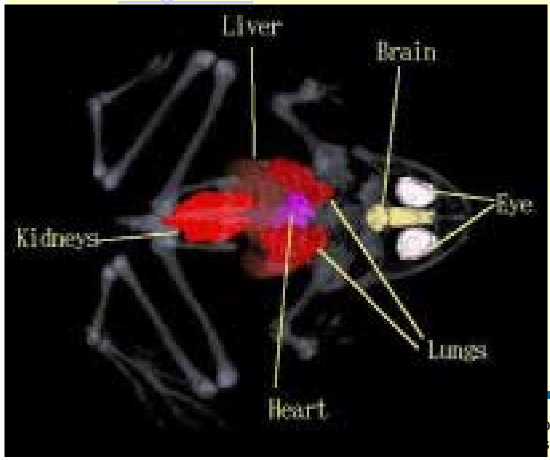




Basic Techniques
Opacity Classification



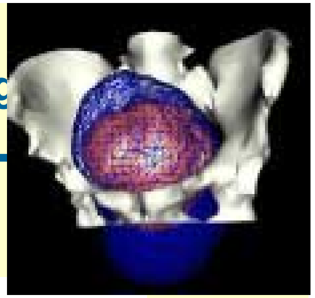
- Frog Movie



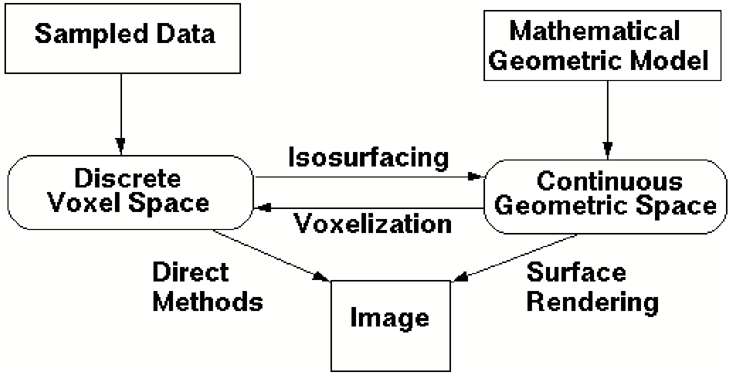
Volume Visualisation Algorithms
Broad Catergorisation

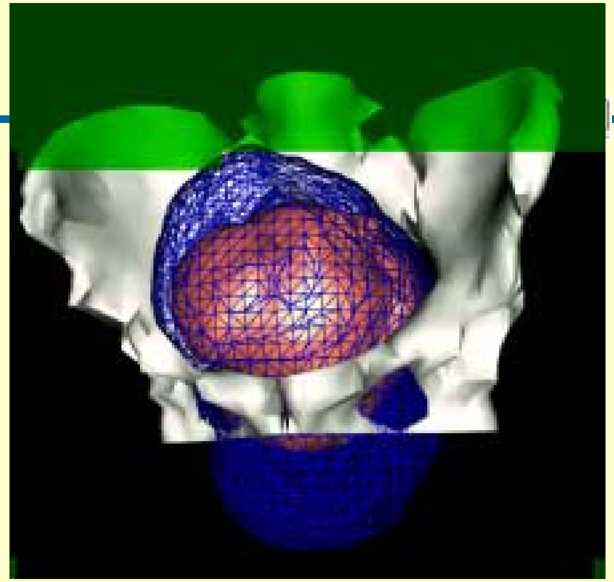
- **Surface Extraction**
  - Objective is to extract single valued (iso-) surfaces from the volume data
  - Convert the volume data into geometric primitives (triangles) to be displayed
- **Direct Volume Rendering**
  - Objective is to render an image of the volume without extracting geometric primitives
  - Generally considered to be more accurate, but also more time consuming

Volume Visualisation Alg



- Flow diagram [2]



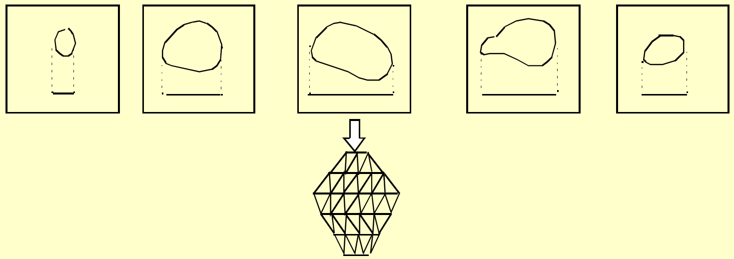


Volume Visualisation Algorithms

- **Surface Extraction Methods**
  - Contour tracking [3][4]
  - Marching Cubes [6]
  - Dividing Cubes [7]
- **Surface Extraction vs. Direct Methods**
- **Direct Methods**
  - Ray Casting [1]
  - Splatting [8][9][10]
- **Mixing Geometry and Volume Data**
  - [11][12][13][14][15]

Surface Extraction Methods Contour Tracking

- Trace Isovalued contours around a sequence of 2D slices
- Construct triangle mesh between slices
- Render the triangles



Surface Extraction Methods
Contour Tracking

- **Keppel [3]**
  - Find path in Directed Graph using heuristics
- **Fuchs [4]**
  - Find minimum cost path in Directed Toroidal Graph
  - Graph theory gives solution without heuristics

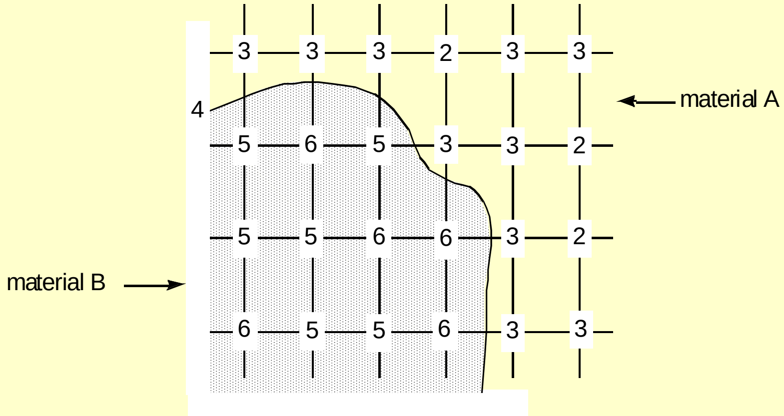
Surface Extraction Methods
Contour Tracking

- **Inaccurate**
- **Handles small features poorly**
- **Can't handle branching**
- **Requires binary classification**
- **Quick to render, using known computer graphics methods**
- **Change of view only requires re-rendering of triangles**

Surface Extraction Methods
Marching Cubes

- **For each voxel through which the surface passes**
  - create small polygons approximating the surface within the voxel
- **Surface intersects edges whose vertices bracket the surface value**
- **Assumptions**
  - maximum of one intersection per edge
  - maximum of four triangles per voxel

Surface Extraction Methods
Marching Cubes

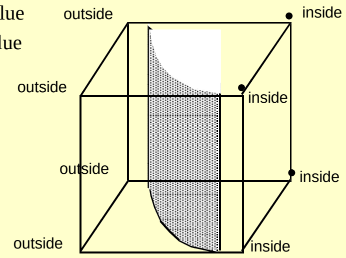


Surface Extraction Methods
Marching Cubes

- **Algorithm Summary**
  - Classify each grid point in the volume as being inside or outside the surface
  - Build an index for each voxel
  - Create a table of intersecting edges
  - Interpolate intersection points
  - Calculate and interpolate surface normals

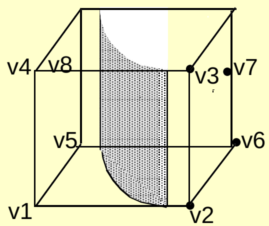
Surface Extraction Methods
Marching Cubes

- **1. For each voxel in the volume classify each vertex**
  - Classify each vertex of the cube as to whether it lies outside the surface or inside the surface
    - Outside (0) if vertex value < surface value
    - Inside (1) if vertex value >= surface value



Surface Extraction Methods
Marching Cubes

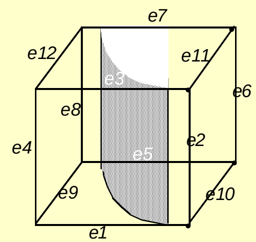
- **2. Create an 8 bit index from the binary labelling of each vertex**



0	1	1	0	0	1	1	0
v8	v7	v6	v5	v4	v3	v2	v1

Surface Extraction Methods
Marching Cubes

- **3. Create a table of intersecting edges**



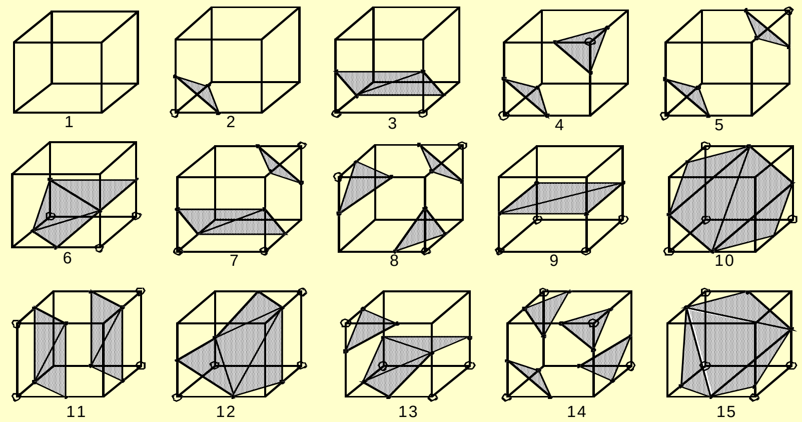
Index	*Edge List*
00000000	.
00000001	*e1 e4 e9*
00000010	*e1 e2 e10*
00000011	.
.	.
.	.
.	.
.	.
.	
.	
11111111	

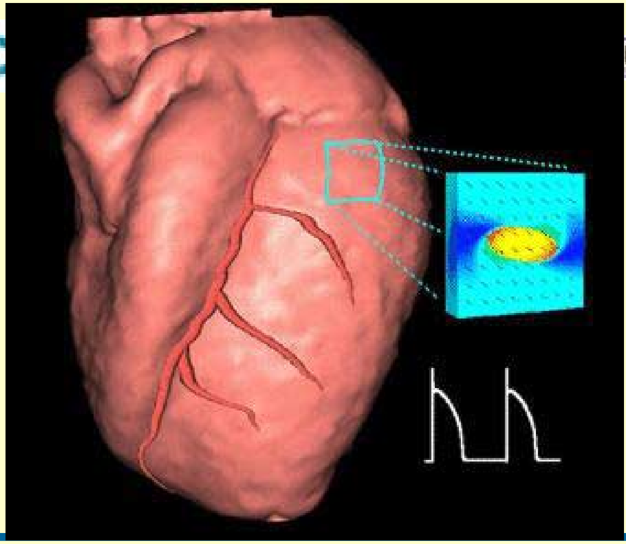
- **Since 8 bit index, the table will have 256 entries**

Surface Extraction Methods
Marching Cubes

- A given index provides access to a list of voxel edges that contain triangle vertices
- All 256 cases can be generated by 15 base case
  - By complimentary cases (swapping 0's and 1's), reduces to 128
  - By rotational symmetry (reduces to 15)
- Thus the edge intersections can be defined by the 15 cases and all other cases can be generated by symmetry

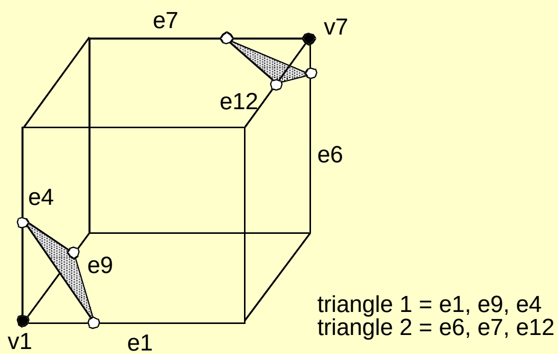
Surface Extraction Methods
Marching Cubes





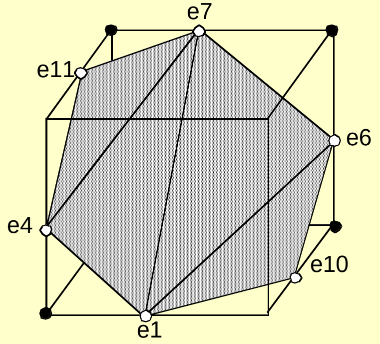
Surface Extraction Methods
Marching Cubes

- Case 4, index 01000001



Surface Extraction Methods
Marching Cubes

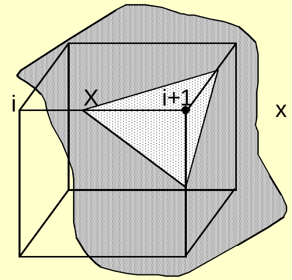
- **Case 9, Index 10110001**



triangle 1 = e4, e7, e11
triangle 2 = e1, e7, e4
triangle 3 = e1, e6, e7
triangle 4 = e1, e10, e6

Surface Extraction Methods
Marching Cubes

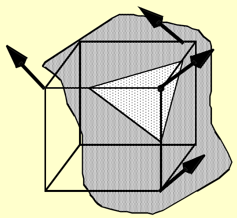
- **4. Interpolate Triangle Vertices**
  - **For each triangle edge, find the vertex using linear interpolation of the sample values**



$x = i + (\mathit{isovalue} - D(i)) / (D(i + 1) - D(i))$

Surface Extraction Methods
Marching Cubes

- **5. Calculate and interpolate normals**
- **For each triangle edge, find normals at cube vertices from gradient of density data using central differences**
- **Interpolate the normals at the point of intersection**



$G_x = D(i+1,j,k) - D(i-1,j,k)$

$G_y = D(i,j+1,k) - D(i,j-1,k)$

$G_z = D(i,j,k+1) - D(i,j,k-1)$

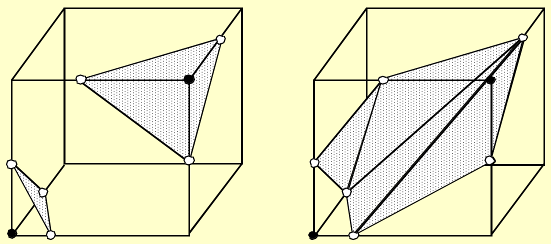
$$\vec{N} = \frac{\vec{G}}{|\vec{G}|}$$

Surface Extraction Methods
Marching Cubes

- **Originally developed to produce surfaces for rendering**
- **Ambiguous cases can results in holes**
- **Solutions to ambiguous cases have been proposed by a number of authors [16],[17]**

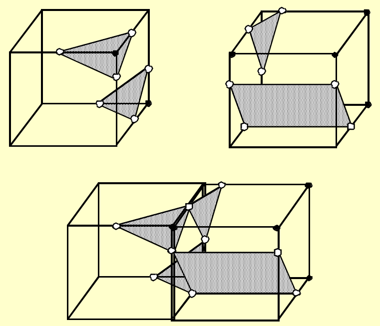
Surface Extraction Methods
Marching Cubes

- **Ambiguous Cases**
  - Occur on any voxel face that has adjacent vertices with different states, but diagonal vertices in the same state
  - There are six of these



Surface Extraction Methods
Marching Cubes

- **Ambiguous Cases**



Volume Visualisation (2)

W T Hewitt
Manchester Visualization Centre
Manchester Computing
University of Manchester

Surface Extraction Methods
Dividing Cubes[7]

- Observation: the size of many polygons produced by Marching Cubes are often smaller than pixel size
- Basic idea
  1. Classify each voxel as inside, outside or on the surface
  2. Subdivide surface voxels to image resolution
  3. Output points with normals

Surface Extraction Methods
Dividing Cubes

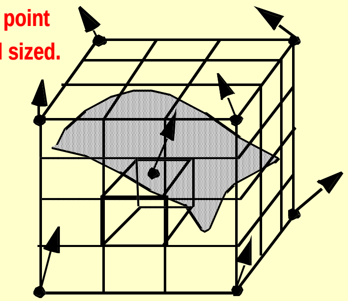
- 1. Classify each voxel in a volume
  - Voxel is interior if all data values are above surface value
  - Voxel is exterior if all data values are below surface value
  - Otherwise surface intersects the voxel

Surface Extraction Methods
Dividing Cubes

- 2. Subdivide Surface voxels to image resolution
  - In x, y, and z create small voxels that correspond to the final image resolution. If original data is 256x256x256 and final image is 512x512, subdivide twice in x, y and z
  - Resample the intensities for the new voxels using tri-linear interpolation
- For each new voxel, test if surface voxel as in step 1

Surface Extraction Methods
Dividing Cubes

- 3. Calculate Normals
  - For each surface voxel interpolate gradient vectors for surface intersections using central differences
  - Calculated surface normal at each surface point
  - Output points with normal, points are pixel sized.



Surface Extraction Methods
Marching Cubes vs. Dividing Cubes

- Both extract surface models from volume data
- Marching Cubes
  - Creates triangles
  - Models rendered using conventional graphics techniques
  - Efficient for small volumes
- Dividing Cubes
  - Creates points with normals
  - Efficient for larger volumes

Surface Extraction Methods
Summary

- Advantages
  - Rendering methods are known: shadows, depth cueing etc.
  - Changing views/lights requires only re-rendering of geometric primitives
  - Can be done in hardware
- Disadvantages
  - Can introduce artifacts and ambiguities
  - Throws away data between surfaces
  - Not capable of imaging volume interiors
  - Not capable of imaging continuum

Direct Rendering Methods
Introduction

- To render the volume directly without recourse to intermediate geometry
- To allow the display of weak and fuzzy surfaces
- Can relax the condition that a surface is either present or not
- Two methods will be covered
  - Ray Casting [1]
  - Splatting [10]

Direct Rendering Methods
Ray Casting

- **Basic idea**
  1. Shade the acquired data to obtain a volume of colour values
  2. Classify the data to obtain a volume of opacity values
  3. Cast a ray through both volumes and takes samples along the ray
  4. At each sample point calculate the colour and opacity using tri-linear interpolation
  5. Composite the colour and opacity samples along the ray to produce a total colour for the ray.

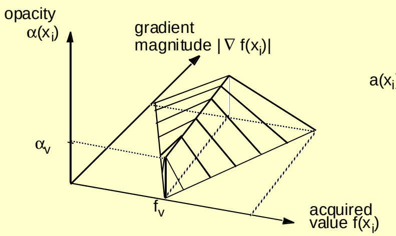
Direct Rendering Methods
Ray Casting

1. Shade the acquired data to obtain a volume of colour values
- Calculate normals for each voxel using central differences
- using Phong or some other shading method, calculate a colour for each voxel, creating a volume of colours

2. Classify the data to obtain a volume of opacity values
- Use one of the classification techniques shown earlier (binary, region boundary surfaces etc.) to assign an opacity for each sample location
- Isovalue contour surfaces — Good for display of multiple concentric surfaces
- Region boundary surfaces — Good for data with lots of fuzzy surfaces

Direct Rendering Methods
Ray Casting

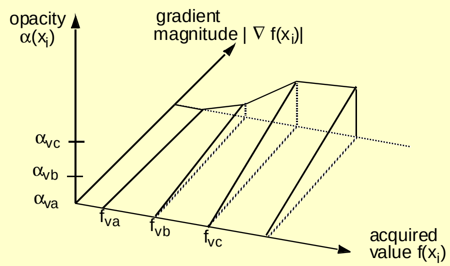
- **Opacity Classification — Isovalue Contour Surfaces**
  - Ensures that data values near to fv are mapped to similar opacities
  - Tails off in inverse proportion to gradient magnitude i.e surface strength



$$a(x_i) = \alpha_v \begin{cases} 1 & \text{if } f(x_i) = f_v \\ 1 - \frac{1}{r}\left|\frac{|f_v - f(x_i)|}{|\nabla f(x_i)|}\right| & \text{if within range} \\ 0 & \text{otherwise} \end{cases}$$

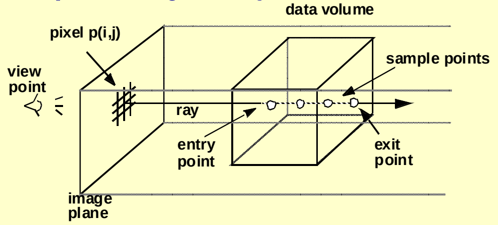
Direct Rendering Methods
Ray Casting

- **Opacity Classification - Region Boundary Surfaces**
  - Assigns opacities to a number of fixed sample values
  - Interpolates between them so that every voxel has an opacity

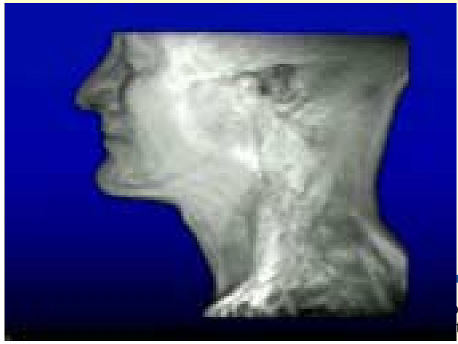


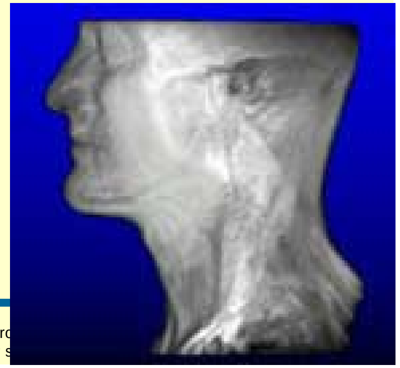
Direct Rendering Methods
Ray Casting

3. Cast a ray through the volume of opacities and colours and take samples along the ray



4. At each sample point, calculate the colour and opacity using trilinear interpolation of the voxel vertex values

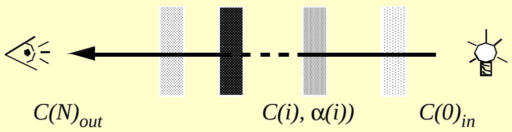




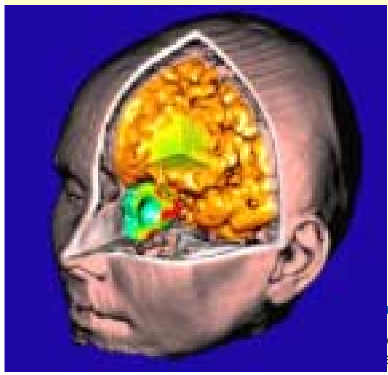
Direct Rendering Methods
Ray Casting

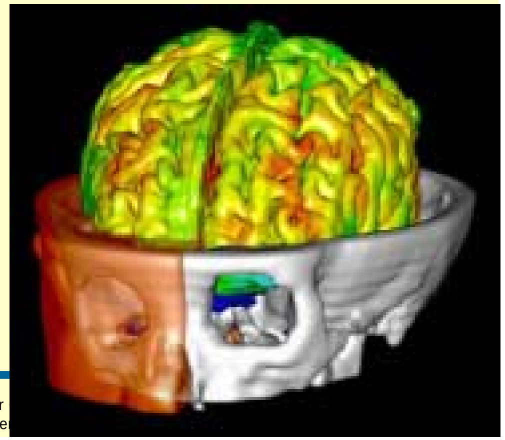
5. Composite the colour and opacity along each ray

- Composite along each ray to give a total colour for the ray and hence the originating pixel
- Uses the Porter and Duff "Over" operator[27]



$$C_{out} = C_{in}(1-\alpha(i)) + C(i)\alpha(i)$$



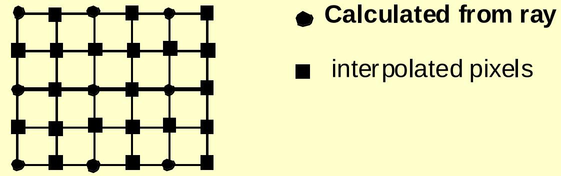


Direct Rendering Methods
Ray Casting — Serial optimisations

- **Early Ray Termination [19]**
  - Stop tracing when total opacity = 1, since no point tracing any further
- **Hierarchical Spatial Enumeration [18]**
  - Pyramids, Octrees,
  - Avoids tracing through parts of the volume that don't contain any relevant data

Direct Rendering Methods
Ray Casting — Serial optimisations

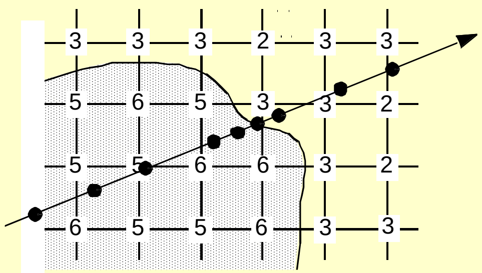
- **Adaptive Screen Sampling[18]**



  - Calculate a subset of the pixels
  - Interpolate intermediate values
  - Replace interpolated values in areas of high gradient

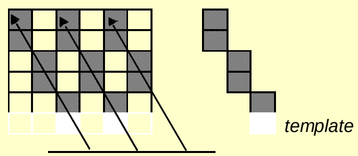
Direct Rendering Methods
Ray Casting — Serial optimisations

- **Adaptive Ray Sampling [20]**
  - Sampling rate is adjusted according to the significance of the traversed data
    - Sampling rate <-> gradient magnitude
    - Sampling rate <-> material opacity



Direct Rendering Methods
Ray Casting — Serial optimisations

- **Templates [21]**
  - rays in parallel viewing perform the same set of steps which is saved as a template



- **Space leaping [22]**
  - techniques to construct a bounding object around the interesting part of the volume, so that empty space can be quickly traversed

Direct Rendering Methods
Splatting [8][9][10]

- **Can be likened to throwing a snowball onto a glass plate**
- **Basic Idea**
  - Determine in what order to traverse to volume
  - Classify and shade the voxels in each slice
  - Project or splat each slice onto the image plane
  - Determine the extent of the splat's contribution and attenuate according to some filter
  - Merge the attenuated splats to calculate the pixel contributions

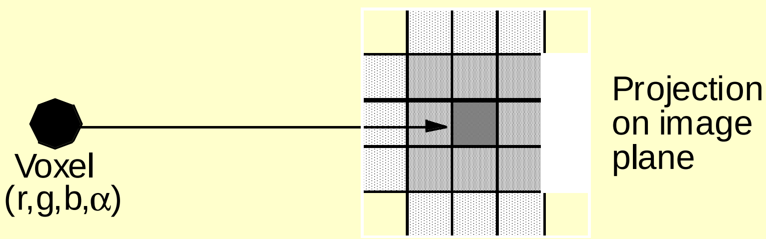
Direct Rendering Methods
Splatting

1. Determine in what order to traverse the volume
- Voxels are projected in order of their distance from the image plane a slice at a time
- Therefore need to calculate which slice is nearest the image plane
- This ensures that close voxels obscure distant voxels

2. Classify and Shade the voxels in each slice
- Classify the voxels according to some opacity transfer function or look up table
- Calculate the gradient using central differences and use this information in the shading formula

Direct Rendering Methods
Splatting

3. Project each voxel in a slice onto the image plane
- A reconstruction filter, usually a Gaussian is used to determine the extent of the splat (its footprint) on the image plane



Direct Rendering Methods
Splatting

4. Attenuate according to the filter
- The classified, shaded voxels (r,g,b,a tuples) are then attenuated according to where they fall within the footprint
- Voxels projected into the middle of the footprint will have the highest contribution

5. Merge the attenuated splats
- Composite the attenuated r,g,b,a tuples that fall within footprint to give a pixel value for the image buffer

Direct Rendering Methods Summary

- Advantages
  - Non-binary classification
  - Shows structure between surfaces
  - Displays small and poorly defined features
  - Readily parallelisable
- Disadvantages
  - Expensive - cost is proportional to volume size (n3)
  - Combining volume and geometry data is difficult
  - Can't take advantage of computer graphics hardware

Mixing Volume and Geometry Data Surface Based Intermixing [24]

- Convert volume data to geometric data using surface reconstruction algorithms
- Mix with the required geometry and render Advantages
  - Can use graphics hardware and normal computer graphics techniques
  - Low memory and storage needs
- Disadvantages
  - Can't see internal data
  - Sensitive to object complexity

Mixing Volume & Geometry Data Volume Based Intermixing [23]

- Voxelise the geometry data
- Mix with the sampled volume & render using direct methods
- Advantages
  - Access to entire volume not just surfaces
  - Insensitive to object complexity
- Disadvantages
  - Large amount of memory and storage
  - Long processing time
  - Classification is difficult

Volume and Geometry Data Point Based Intermixing [25]

- Adds a "point" to the set of geometric primitives where a point is a 3D location and normal
  - Normals are calculated for each voxel using central differences
  - Thresholding can be used to cut down the number of points
- Advantages
  - Interactive
  - Can take advantage of computer graphics hardware
- Disadvantages
  - Repeated normal estimation for every change
  - Holes may appear

Volume and Geometry Data Z-merging [23]

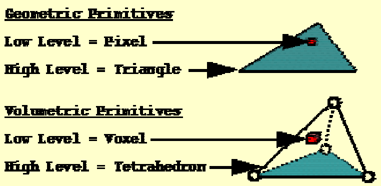
- Render volume using surface reconstruction or direct methods
  - Render geometry separately
  - Merge using some depth storing technique
- Advantages
  - Can use either direct or indirect methods
  - Can be implemented in hardware
  - Interaction
- Disadvantages
  - Re-renders all the data if the model is changed

Volume and Geometry Data Ray Merging [28]

- Trace rays through volume and geometry datasets producing two vectors of values
- Combine the two vectors to produce pixel values
- Advantages
  - Handles transparency
  - Uses complete volume
  - Handles small features and complex objects
- Disadvantages
  - Not interactive
  - Lots of memory required

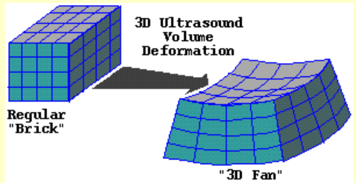
APIs for Volume Visualization

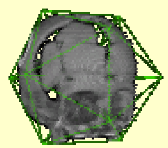
- OpenGL Volumizer (www.sgi.com/software/volumizer)
- Developed by SGI
- Library of C++ classes
- Aimed at O2, Octane and Onyx2
- I.e., needs 2D and/or 3D texture hardware
- Adds volumetric primitive

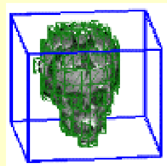


Volumizer Supports

- Mixing primitives
- Volume deformation
- Co-resident volumes
- Time-varying volumes
- Arbitrary regions of interest







Cochlea Implant

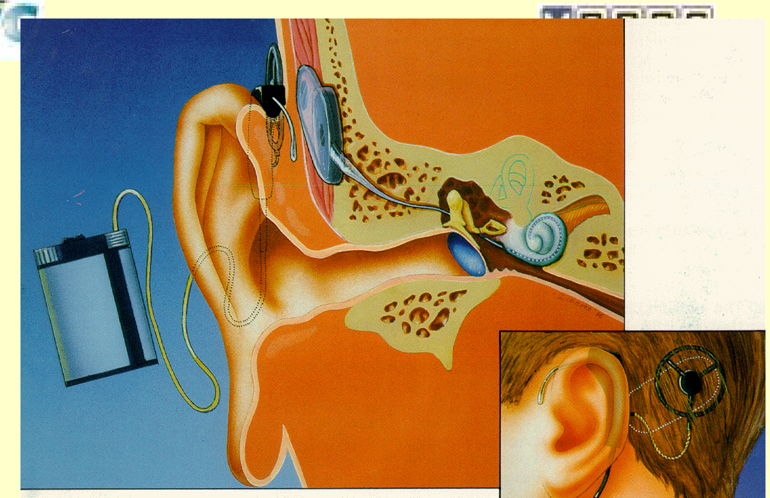
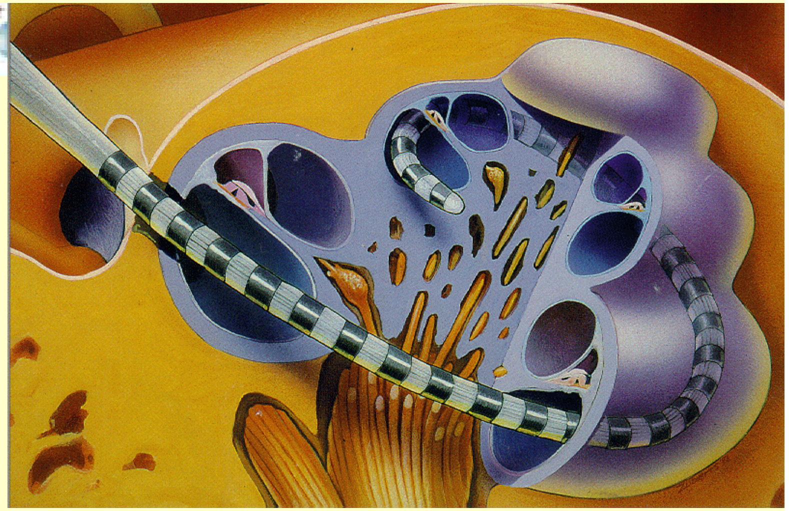
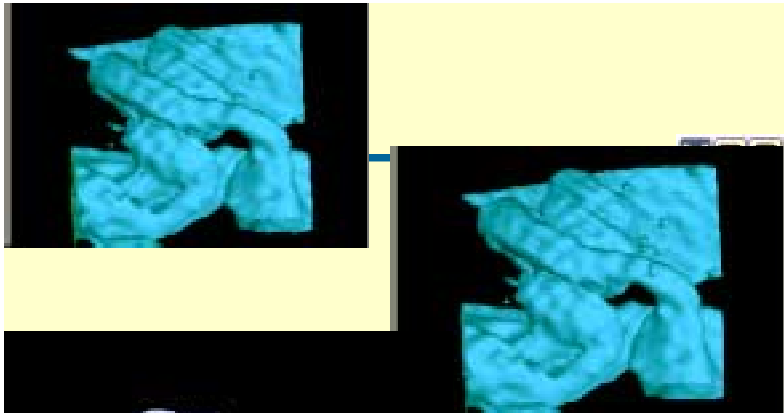
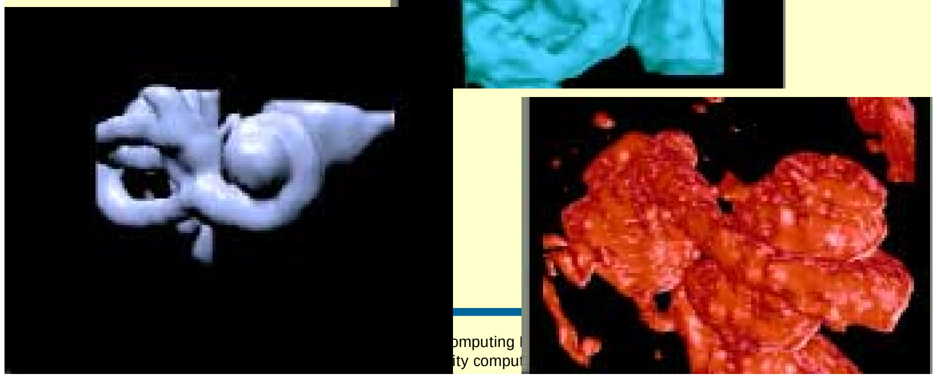


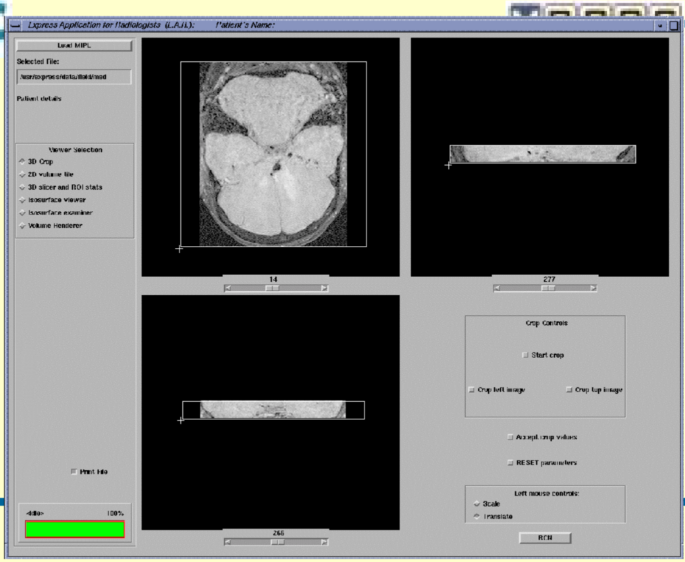
Fig. 2 The Nucleus 22 Channel Cochlear Implant System

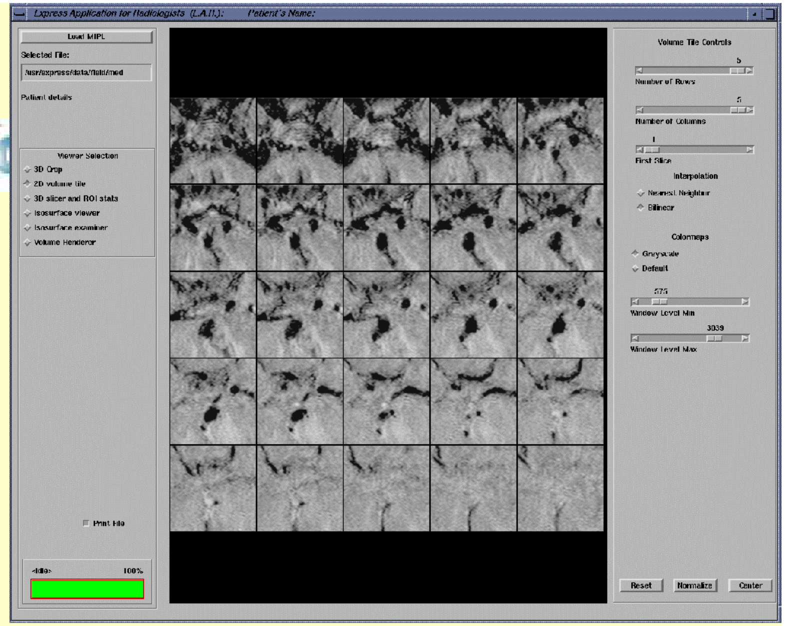


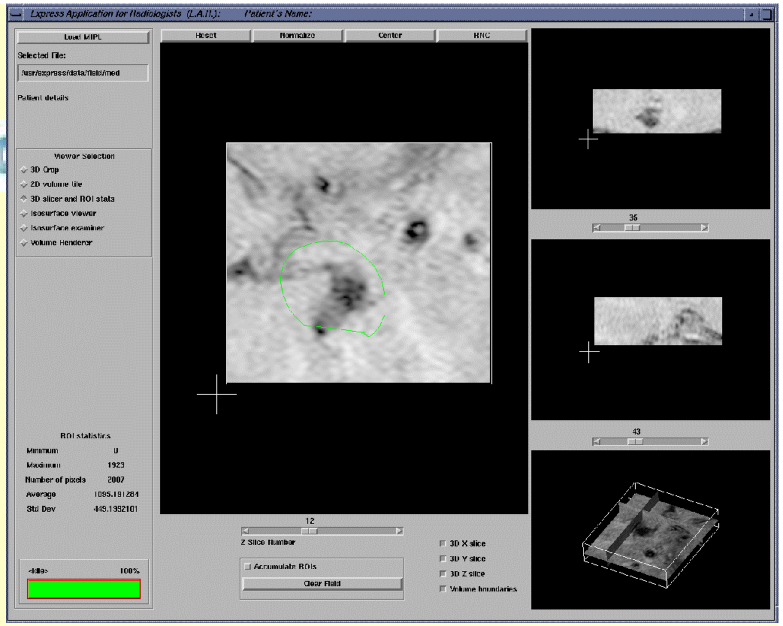


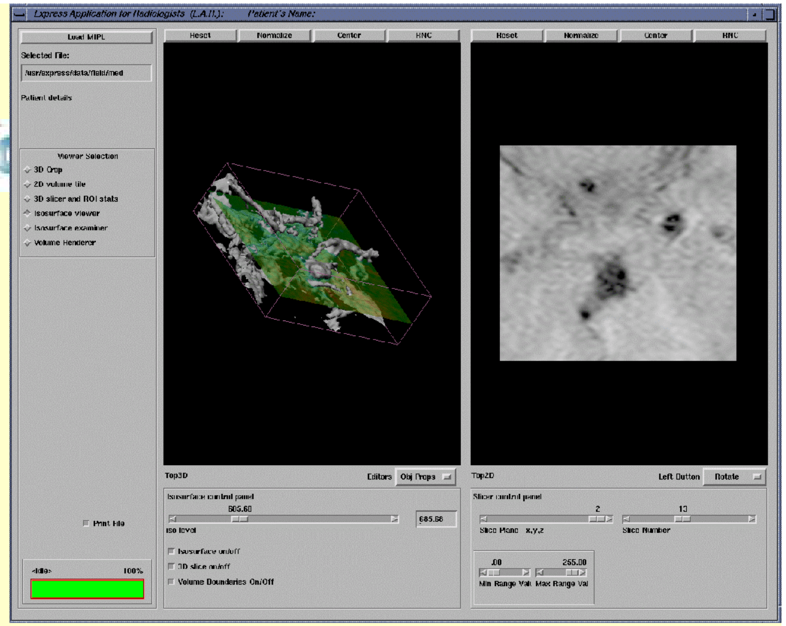


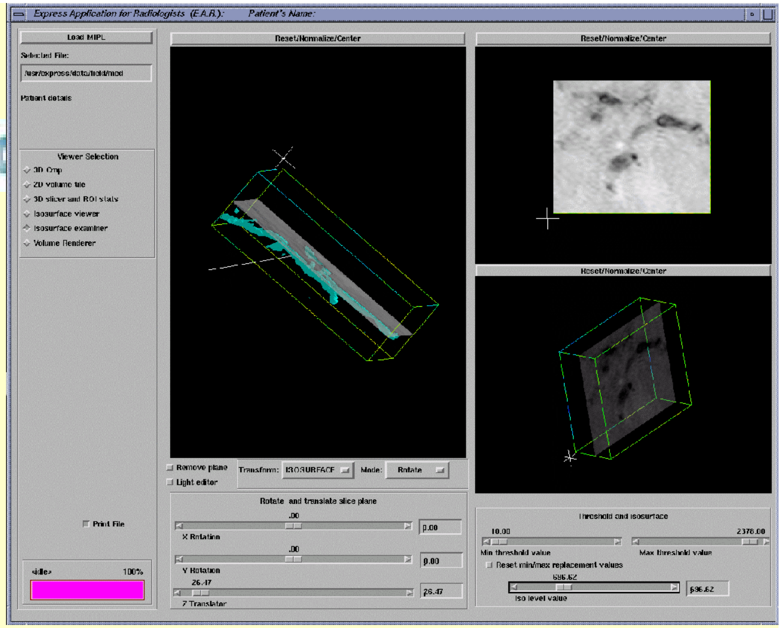
Aneurisms in the Brain



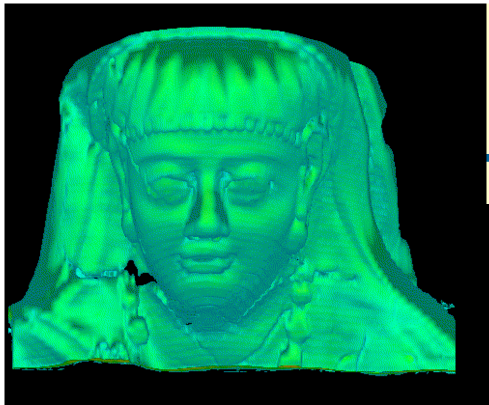


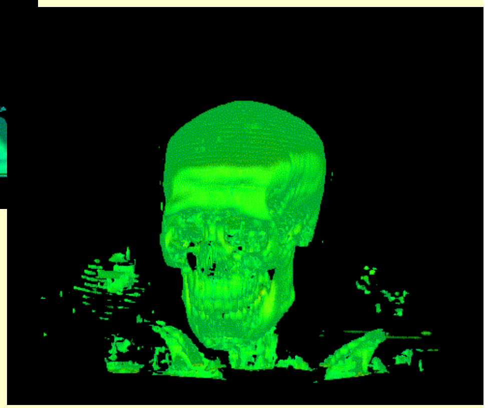




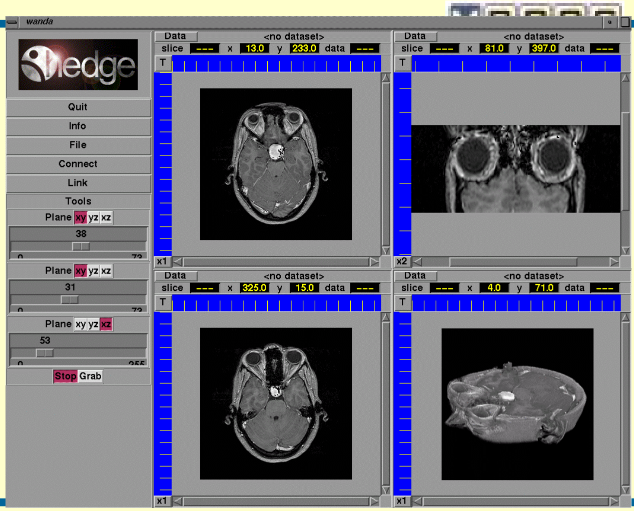

Express Application for Radiologists (E.A.R.):
Patient's Name:
Load MPL
Selected File:
Patient details
Viewer Selection
3D Crop
2D volume tile
Isosurface viewer
Isosurface examiner
Volume Renderer
Reset/Normalize/Center
Reset/Normalize/Center
Reset/Normalize/Center
Remove plane
Light editor
Transform:
ISOSURFACE
Mode:
Rotate
Rotate and translate slice plane
X Rotation
Y Rotation
Z Translation
Threshold and isosurface
Min threshold value
Max threshold value
Reset min/max replacement values
Iso level value
Print File

My Mummy





Remote Radiologists Workbench



Volume Visualisation References

W T Hewitt
Manchester Visualization Centre
Manchester Computing
University of Manchester

References

1] M Levoy, "Display of Surfaces from Volume Data" IEEE Computer Graphics and Applications Vol 8 No 3 May 1988 pp29-37
[2] A Kauffman, "Volume Visualisation `91" IEEE Computer Society Press 1991
[3] E Kepple, "Approximating Complex Surfaces by Triangulation of Contour Lines" IBM Journal of Research and Development Vol 19 No 1 Jan. 1975
[4] H Fuchs, et al. "Optical Surface Reconstruction from Planar Contours" Communications of the ACM Vol 20 No 10 Oct. 1977
[5] G T Herman, H K Lui, "Three Dimensional Display of Human Organs from Computed Tomograms" Computer Graphics and Image Processing Vol 9 No 1 Jan. 1979
[6] W Lorenson, H Cline, "Marching Cubes: A high resolution 3D Surface Construction Algorithm" Computer Graphics Vol 21 No 4 July 1987
[7] Cline et al. "Two Algorithms for 3 Dimensional Reconstruction of Tomograms" Medical Physics Vol 15 No 3 May/July 1988
[8] L Westover, "Footprint Evaluation for Volume Rendering" Computer Graphics Vol 24 No 4 August 1991
[9] L Westover, "Splatting, A Parallel, Feedforward Volume Rendering Algorithm" Dept. of Computer Science, UNC at Chapel Hill Tech Report TR91-029 July 1991
[10] L Westover, "Interactive Volume Rendering" Proceedings of the Chapel Hill Workshop on Volume Visualisation 9-16 May 1989
[11] A Kauffman et al. "Intermixing Surface & Volume Rendering" 3D Imaging in Medicine K Hoehre, H Fuchs & S Pizer (eds) Springer Verlag pp217-227 1990
[12] A Kauffman, Shimony E "Efficient Algorithms for 3D scan conversion of parametric curves, surfaces and volumes", Computer Graphics 21, 4, 171-179 (July 1987)

References

[13] Johnson & Mosher, "Integration of Volume Rendering & Geometric Graphics" Proceedings of Chapel Hill Workshop on Volume Rendering Chapel Hill NC 1-8 May 1989
[14] M Levoy, "A Hybrid Ray Tracer for Rendering Polygons & Volume Data" IEEE Computer Graphics and Applications 1990
[15] G Nielson et al. "The Asymptopic Decider: Resolving the Ambiguity in Marching Cubes" Proceedings of IEEE Visualisation 1991 Conference IEEE Computer Society Press Oct. 1991
[16] E Wu, R He, W Wang, "Disambiguation in Marching Cubes through Direct Topological Detection"
[17] M Levoy, "Volume Rendering by Adaptive Refinement" The Visual Computer Vol 6, No 1, pp 2-7 Feb 1990
[18] M Levoy "Efficient Ray Tracing of Volume Data" ACM Transaction on Graphics Vol 9 No 3 pp 245-261 Feb. 1990
[19] P Hanrahan, "Hierarchical Splatting: A Progressive Refinement Algorithm for Volume Rendering" Computer Graphics Vol 25 No 4 July 1991
[20] R Yagel, "Template Based Volume Viewing" Computer Graphics Forum Vol 4 No 3 pp153-157 Sep. 1992
[21] R Yagel, "Accelerating Volume Viewing by Space Leaping" OSU-CISRC-3/93-TR10 Dept. of Computer Science Ohio State University March 1993
[22] T Porter & Duff, "Composing Digital Images" Computer Graphics Vol 18 No 3 July 1984
[23] P Hanrahan, Salem J et al. "Volume Rendering" Computer Graphics ,22, 4, 65-74 (August 1988). Proceedings of the San Diego workshop on volume visualisaton
[24] Laur D, Hanrahan P, Grant E, "Image Rendering by Adaptive Refinement" Computer Graphics 20, 4,29-37 (August 1986)
[25] M Levoy, "Volume Rendering by Adaptive Refinement", The Visual Computer, 6, 1, 2-7 (February 1990)
[26] P Sabella "A Rendering Algorithm for Visualising 3D Scalar Fields", Computer Graphics, 22, 4, 51-58 (August 1988)
[27] P Shirley, A Tuchman et al. "Area and Volume Coherence for Efficient Visualisation of 3D Scalar Functions", Computer Graphics, 24,5,27-33 (November 1990)
[28] R Yagel , A Kaufman, Q Zhang, "Realistic Volume Imaging" Proceedings of Visualisation 1991 San Diego 226-231 October 1991

Framework Technologies and Methods for Large Data Visualization

Ian Curington
Advanced Visual Systems Ltd.

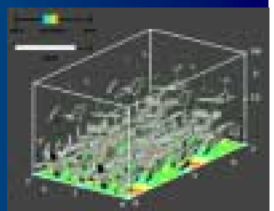
ADVANCED VISUAL SYSTEMS INC.

Case Studies

- Fluid Dynamics Animation
- Astrophysics Simulation
- Aerospace Flow Visualization
- Wind Tunnel Store Separation Tests
- Oil Reservoir Simulation Visualization
- Radar Vulnerability Analysis
- Microwave Field Visualization
- EDA - IC Layout Review

CFD Visualization

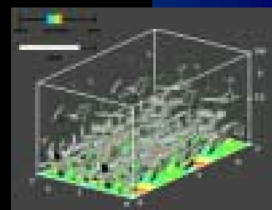
- Laminar-Turbulent Transition in a Supersonic Boundary Layer
- Award Winning Animation: APS/DFD Physics of Fluids Journal
- 200 Hours on NEC SX3
- Adaptive Colormap Scheme
- 193 GigaBytes of Data Visualized



CFD Visualization

C. Mielke, L. Kleiser, J. Favre
ETH Zurich, CSCS

(movie)



Example: CFD Data Format (DLR)

Data by Ch. Mielke, ETH-Z, Institute of Fluid Dynamics

Expensive File Format:

For each time step, 18 fields are stored:

u-velocity	x-vorticity	v-velocity-x
v-velocity	y-vorticity	v-velocity-y
w-velocity	z-vorticity	v-velocity-z
pressure	u-velocity-x	w-velocity-x
temperature	u-velocity-y	w-velocity-y
density	u-velocity-z	w-velocity-z

Computation of Derived Quantities

Using (V) array notation (*as in Fortran 90*)

```
Sec_inv_data => ((((nd[18]*nd[14])+
  ((nd[18]+nd[14])*nd[10]))-
  (nd[13]*nd[11])+
  (nd[12]*nd[16])))-
  (nd[15]*nd[17]));
```

[scalar: 2nd invariant of velocity gradient tensor]

Disk-mapped I/Os for Data > RAM

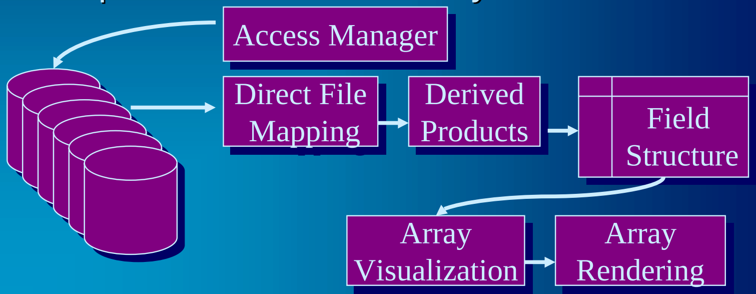
- Grids of up to 10 million points
- 600 Mbytes per timestep
- "FILE" Access Objects
- Data accessed on demand at runtime
- Caching is Optional - Excellent for Derived Qty.
- Cropping in K direction is trivial

File Access Objects are Faster than Loading Data into RAM

- Accessing a 3x3 velocity gradient tensor and computing its second invariant,
- Extracting isosurfaces and mapping the pressure field onto them,
- Displaying the isosurfaces (often consist of meshes of 250K+ triangles
- 36% Speedup for large grids

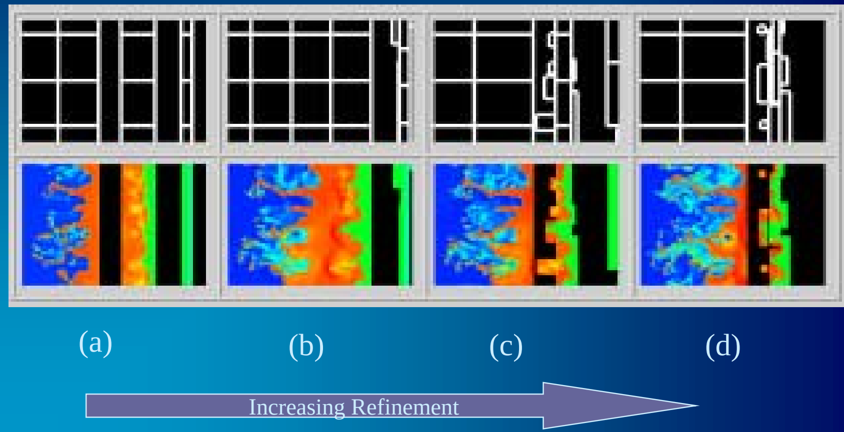
Segmented File Access

- Ordered File Archive
- Direct array to file mapping
- Derived products created ‘on the fly’
- Populate Hierarchical Array structure

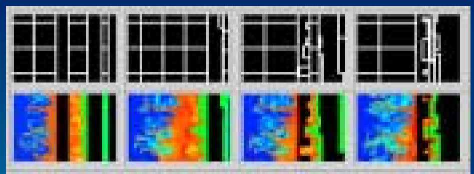


Visualization for Multi-grid Simulations in Astrophysics

Adaptive Grid Refinement - Multi-Grid Hierarchy



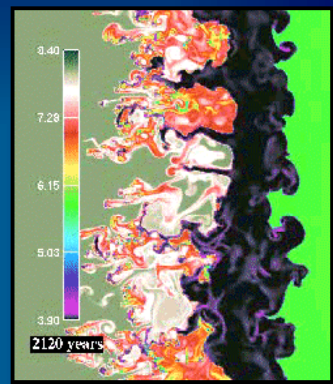
Visualization for Multi-grid Simulations in Astrophysics



- 5 levels of hierarchical refinement, using Multi-block data structure
- Stability of radiative shocks
- Typical of supernova remnants or planetary nebulae
- Up to 200 blocks / time step

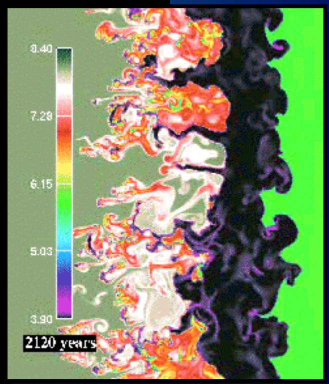
Large Data: Astrophysics Simulation

- Multi-block Macros for Visualization Operators
- *FILE* Objects
- 618 time steps
- Custom Colormap
- Video Animation



Large Data: Astrophysics Simulation

Dr. Jean Favre, CSCS, ETH Zurich, Switzerland



(movie)

CIRA
Centro Italiano Ricerche Aerospaziali

- NAPLES, Italy Site
- Founded in 1984 for:
  - **Italian Aerospace Industry Support**
  - **Regional Companies**
- 250 people , 50% researchers
- Founding member of AEREA consortium (Aeronautic European Research Establishment Association) with:
  - **DLR (Germany), DRA (UK), FFA (Sweden), NLR (Holland) ,**
  - **INTA (Spain), ONERA (France)**

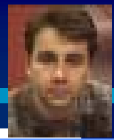
Principle Research Areas

- Aerodynamics
- Aeronautics
- Aero-thermodynamics
- Aero-acoustics
- Ice Protection
- Space Propulsion
- Vehicle Mechanics
- Structures and Materials
- Scientific Computations and Visualization

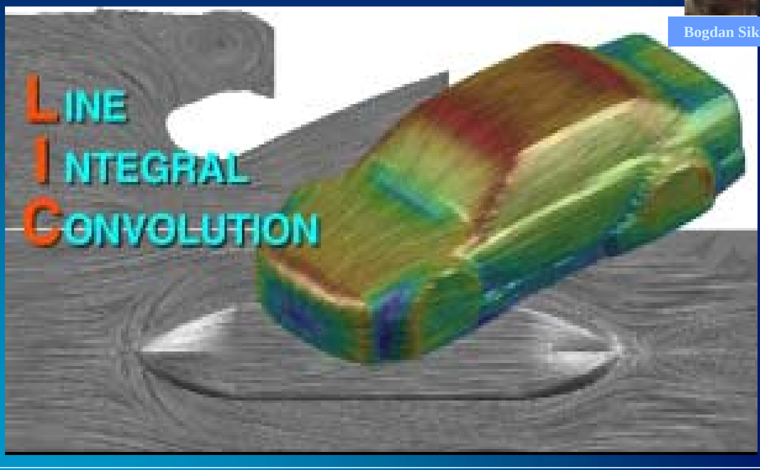
Experimental Facilities

- Wind Tunnels:
  - **Plasma Wind Tunnel**
  - **Low Speed Wind Tunnel**
  - **Icing Wind Tunnel**
- Propulsion cryogenics
- Vehicle Crash test lab, scale 1:1
- Supercomputing Facility
  - **SGI Power Challenge with 16 R10000 4GByte ram**
  - **Convex 3880 with 8 processors 1 GByte ram**

Innovative Visualization Technique Development - CFD

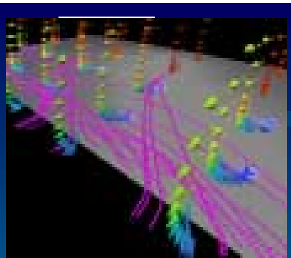


Bogdan Sikorski



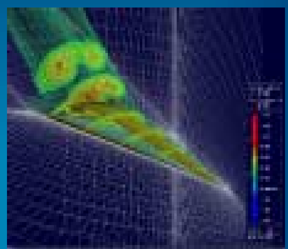
LIC - surface velocity field

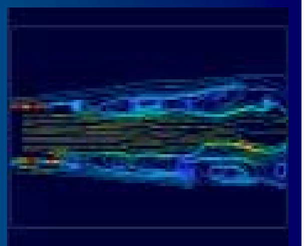
- Boundary surface velocity field visualization
- Complex geometry
- Full Surface, not just stream lines



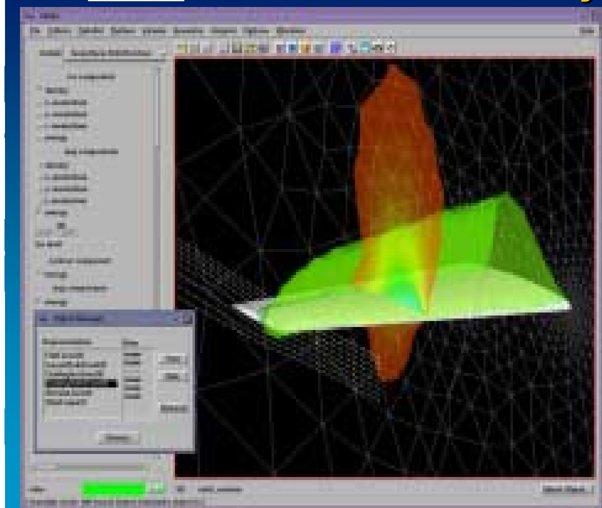
Visualization for Structured Grids: *Flovis®*

Previous Version: C, C++, OGL





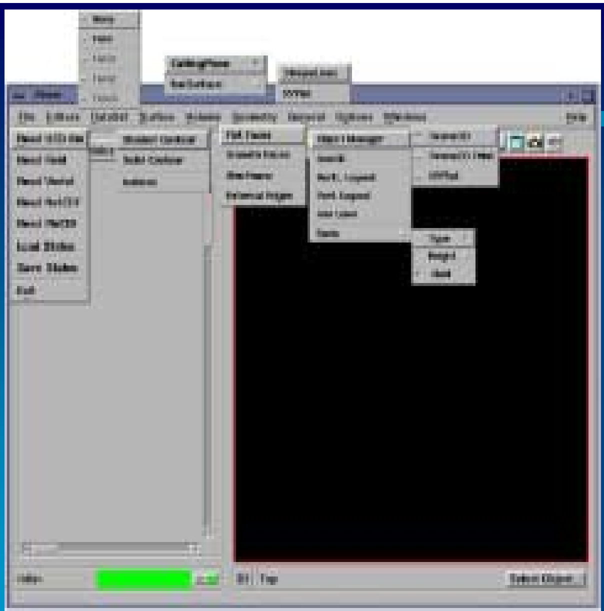
FLOVIS new visualization system



based on AVS/Express

FLOVIS

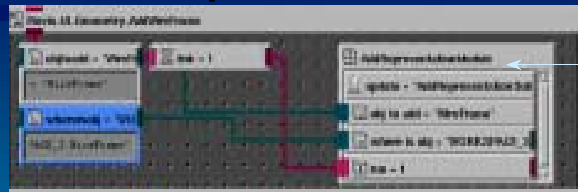
Structured,
Unstructured,
Multi-block



FLOVIS Menu

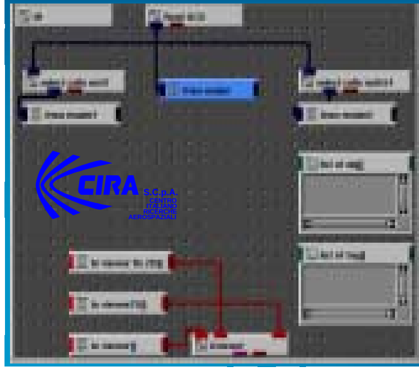
Developed with UI Kit

AddRepresentation Module

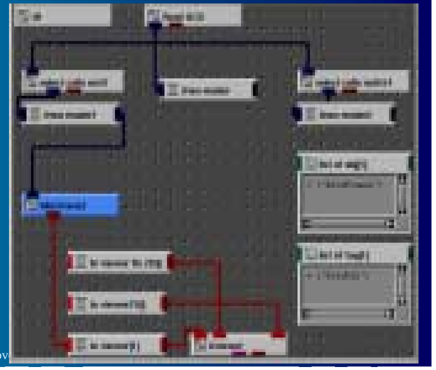


- Makes OM Calls for Dynamic State

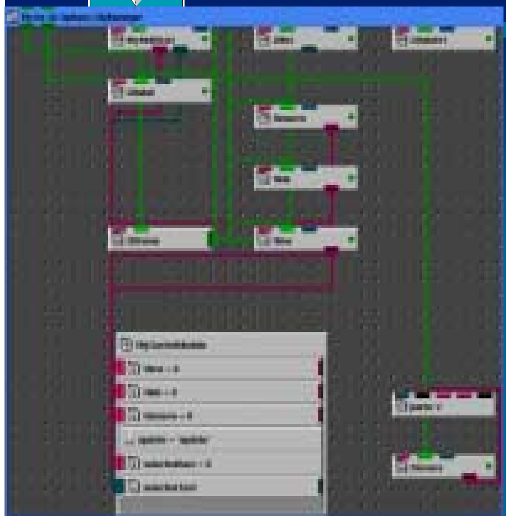
Before



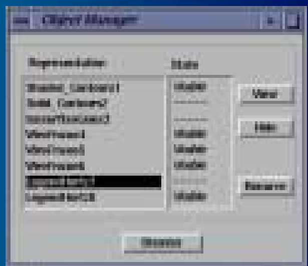
After



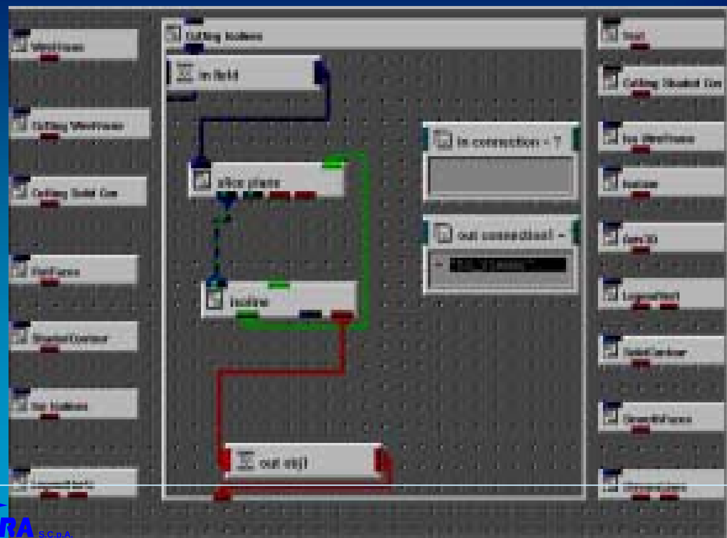
Object Manager



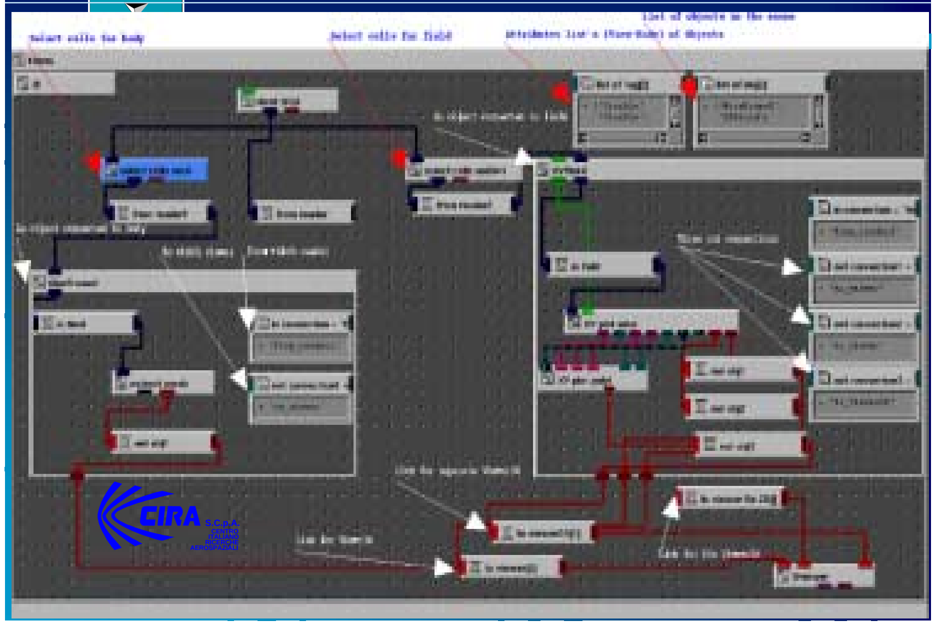
User Interface for Dynamic Network Control



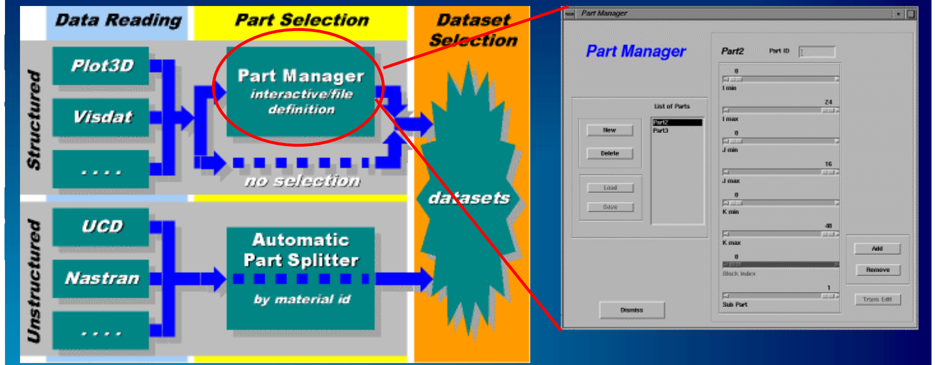
Flovis Application Macros



Flovis Application workspace



Parts/Materials/ Substructures



- Allows navigation to region of interest
- Supports visualization of large models - Part access model

McDonnell Douglas (now Boeing)

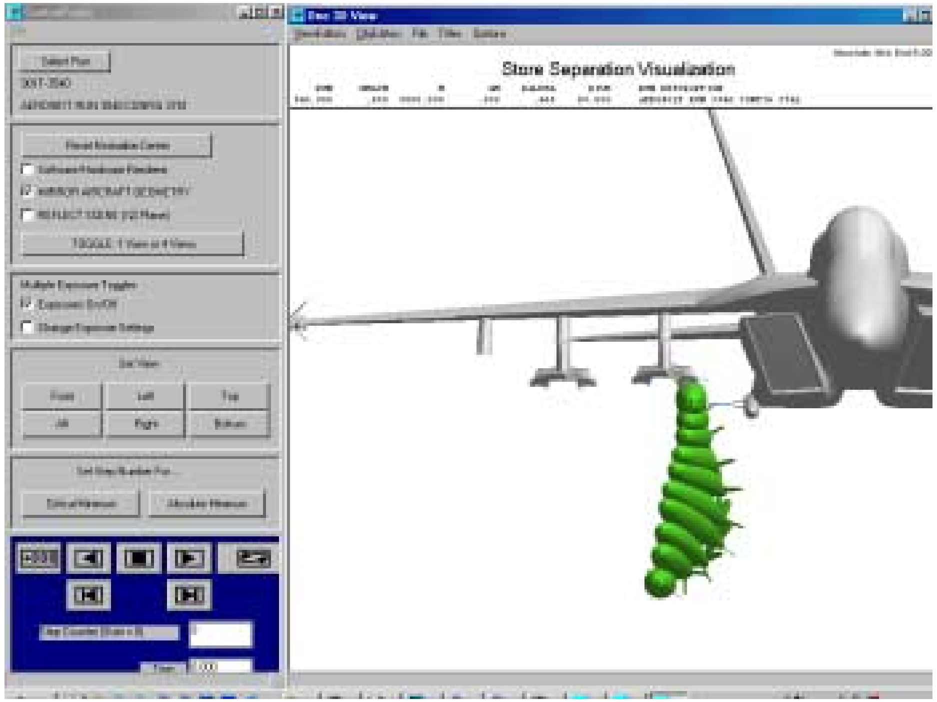
- Store Separation Analysis Application
- Wind Tunnel / Simulation Data: 1000's of runs

F/A-18E/F fighter/attack aircraft program

McDonnell Douglas "TMAN"

Time History ANimation Project

- AVS/Express Based
- 3D Reporting Tool
- Playback of 3D structural dynamics
- Critical Distance Visualization
- Multiple Exposure mode
- "HESS" Panel Method CFD Format



Oil Reservoir Example

Schlumberger Geoquest

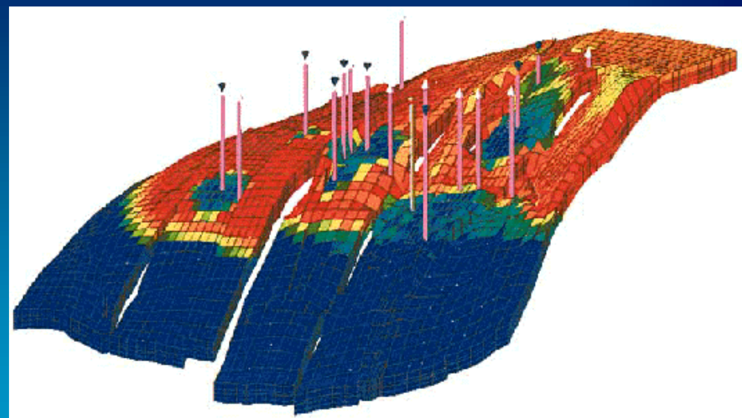
- http://www.slb.com
- Geologic, Oil / Gas / Water Modeling
- Reservoir Simulation - Time Histories
- Products:
  - **ECLIPSE, FloGrid, Schedule, Office, FloViz**
- Large Grid sizes (500K)
- Large number of time steps (15+ years)

Oil Reservoir Example (2)

Schlumberger Geoquest

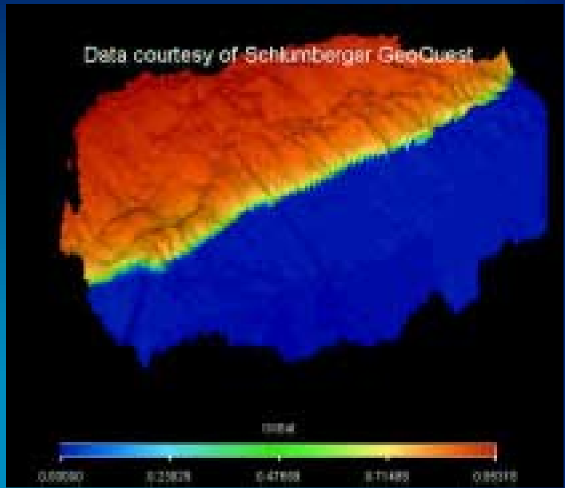
- North Sea example
- 300K Cells - Geologic Model
- 60K “Active” Cells
- Time Series Animation of Values
- 850 Frame Animation Movie

Reservoir Simulation Example



Data & Images Courtesy of Schlumberger Geoquest

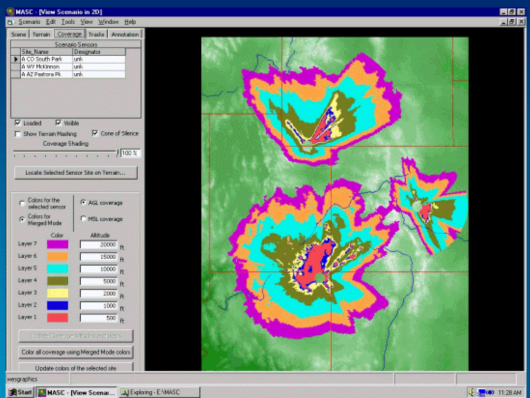
Reservoir Simulation Example



CSC - NORAD: MASC Project

- Military Radar Vulnerability Analysis
- MASC: Model for Analysis of Sensor Coverage
  - **Terrain Masking**
  - **Theater Surveillance using Unmanned Airborne Vehicles (UAVs)**
  - **Satellite line-of-site masking**
- MASC, replacement of 2D program
- CSC: Computer Science Corporation
- NORAD: North American Aerospace Defense Command

CSC - NORAD

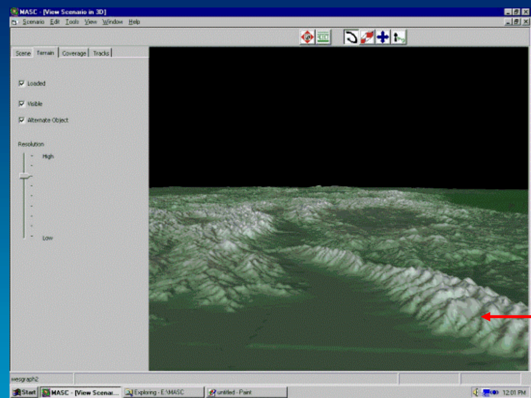


Radar & Terrain Coverage Visualization

•Easy to Use Interface
•Direct Access to Data

2D plan view coverage from 3 radar sites in Western USA

CSC - NORAD

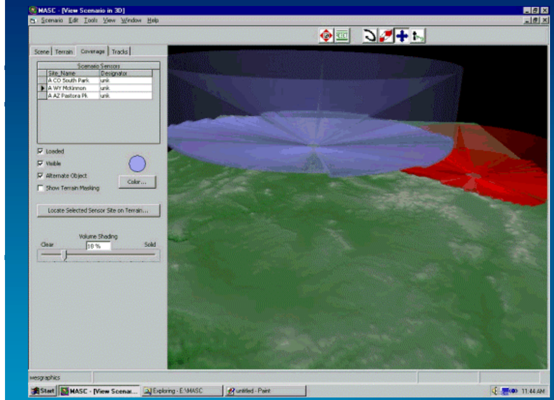


Terrain Review,
Site Planning,
Aircraft Tracks

Sangre de Christo Range

3D Perspective View to the North, Colorado Rocky Mountains

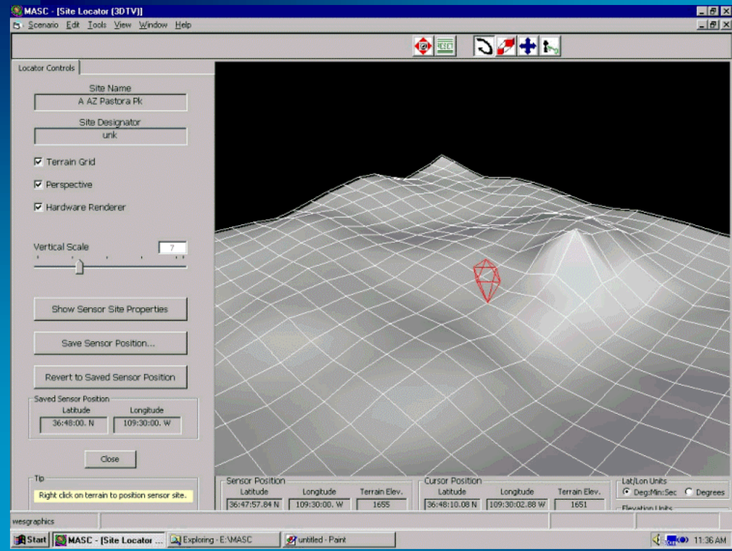
CSC - NORAD



- AVS/Express based Visualization Environment
- PC-based Application
- Integrated Government Algorithms

MASC view showing 3D view of RADAR coverage volumes over Rocky Mountains

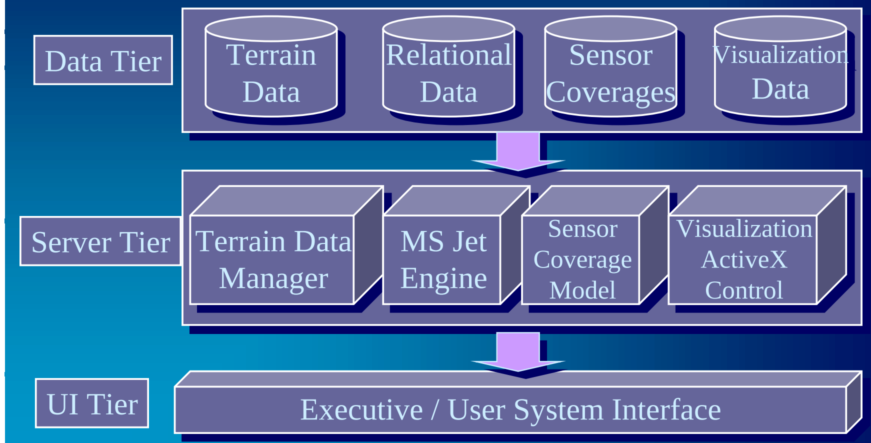
CSC - NORAD



Interactive Sensor Positioning on high resolution terrain

CSC – NORAD

MASC: Architectural Components



KCC Application Overview

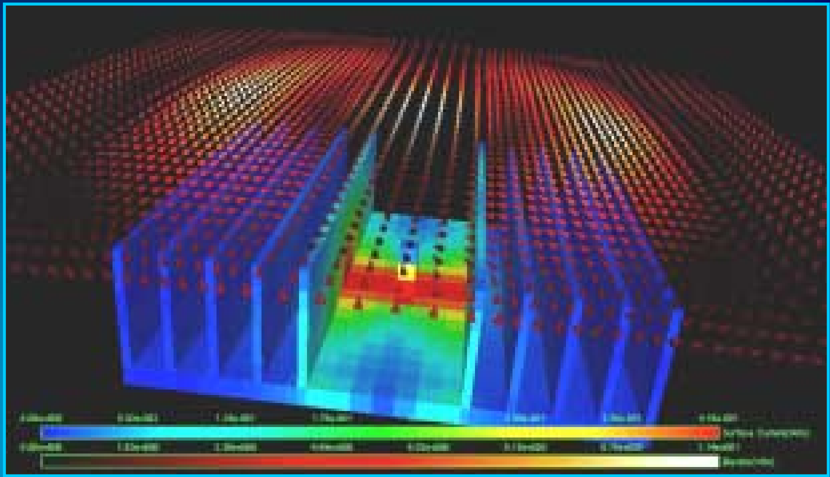
Kimberley Communications Consultants

Kimberley Communications Consultants Ltd

- Micro-Strips Technique
- RF Field Simulation
- Microwave Devices
- EMC

(Now merged with Flomerics Ltd.)

KCC Field Plot Application: Heat Sink Example



Heat Sink Currents & Electric Field

KCC Visualization Strategy

- Competitive Advantage of Improved Viz
- Increases sales potential to KCC's product
- AVS Solution Partnership
- AVS/Express Development Seat + Professional Services + Deployment

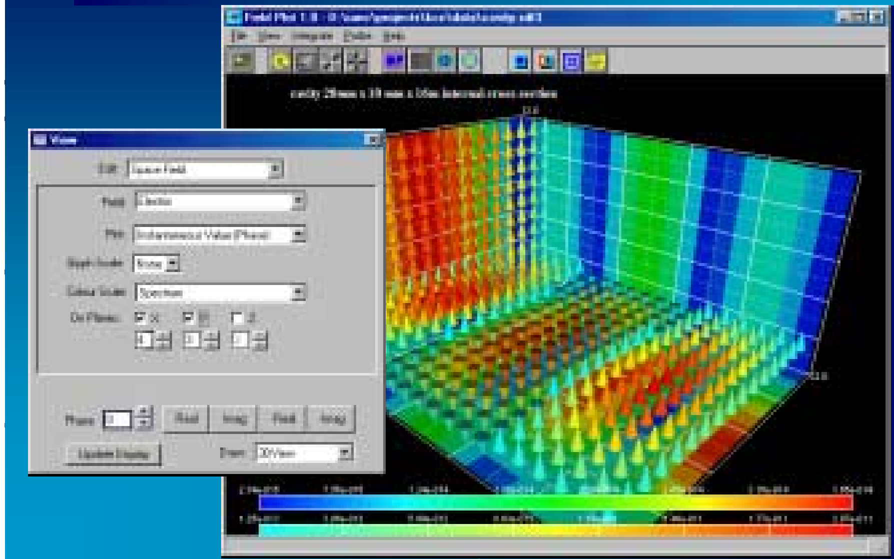
KCC UK Ltd. Microwave Simulation

- Needed 3D Real-time Interaction
- Needed Cross platform solution (UNIX/NT)
- non-uniform cells with cell-based data
- Needed circular probe
- Needed Cone glyphs
- Needed Culling of back facing surfaces

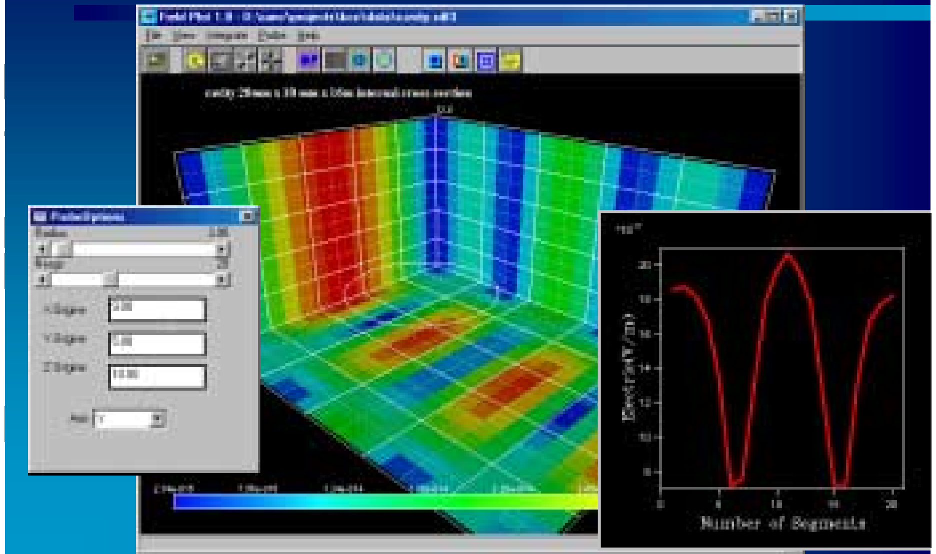
KCC Development Phase

- Similar to HP Eesof
- KCC staff developed intelligent reader
  - **Large Data Handling**
  - **Computation of Derived Results**
- Total development time to runtime delivery: **10 weeks**
- Electric / Magnetic Field Vectors (Real/Imag)
- Surface Current Visualization
- Interactive Phase adjustment

KCC Field Plot Vector Display



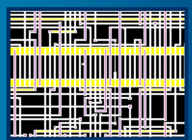
KCC Circular RF Field Probe



AVS & EDA Tool Development

Visualization for VLSI Layout

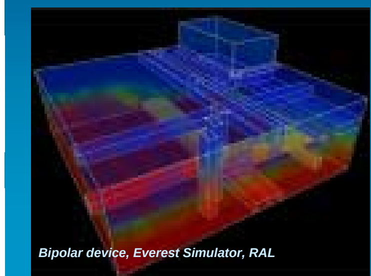
Ian Curington
Advanced Visual Systems Inc.
ianc@avs.com



AVS Examples in EDA:

Electronic Design Automation

- **Electronics - EDIF**
- **Electromagnetic - RF**
- **TCAD**
- **Optical Path Correction**
- **FAB Process QA**
- **MEMS**



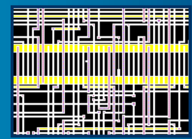
Bipolar device, Everest Simulator, RAL

AVS - EDA *GDSII Viewer* Features

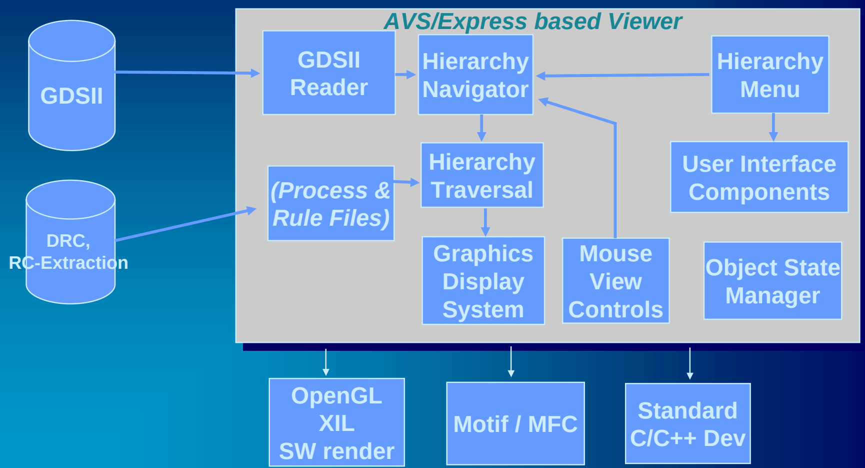
- Direct Access of GDSII Geometry
  Hierarchy Navigation, Display
- Single/Multiple View windows as needed
- Cross platform GUI - Motif (UNIX), MFC (NT)
- 2D Graphical Update Acceleration Available through Hardware (OpenGL, XIL)
- High Level Application Architecture for Rapid Refinement

GDS-II Viewer

- GDSII: Large, Deeply Hierarchical Structure
- Defines Geometric Layout for Chip
- Custom "Render-Method" Traverses Hierarchy
- Level-of-Detail display control
- 26 Million Graphics Prims in one view on a PC
- Custom Zoom/Pan Navigation
- Feature Display Mode Editor

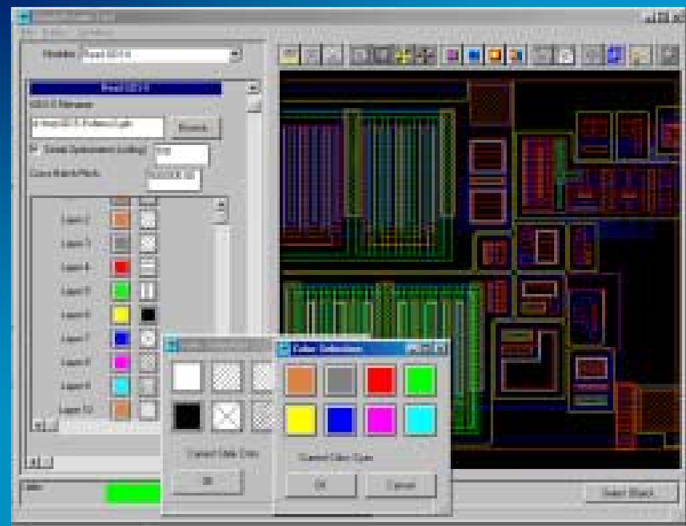


Architecture



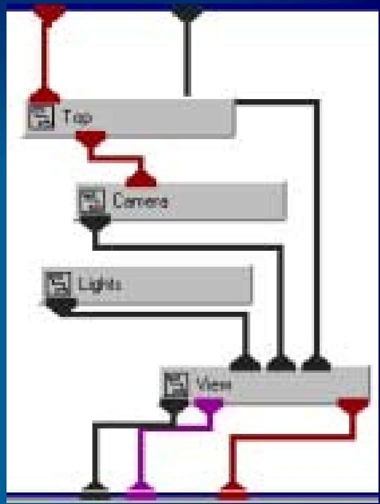
AVS Visualization Frameworks

Layout Viewer



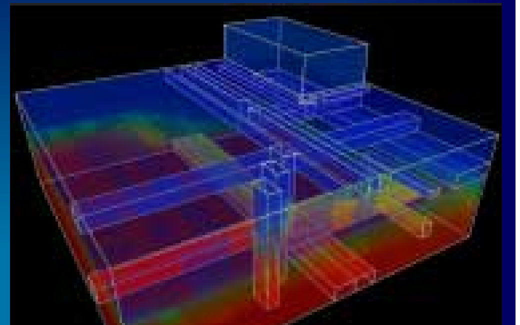
Graphics Pipeline

- Minimum Memory Profile
- Large Data Management
- Thin OpenGL Layer
- Not Scene-Tree Dependent
- Multi-Pass “Chunking”
- Objects register “render-methods” at runtime
- Graphics APIs through “virtual renderer” I/f
- Allows Procedural Objects



AVS Visualization Frameworks

- Layout view can be extended to 3D
- Technology CAD,
- Process Optimization
- Device Characterization



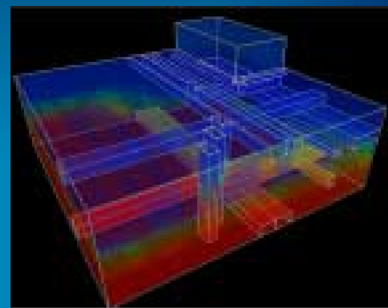
3D Bipolar TCAD model
Everest Simulator, RAL

Mask Layout Review Framework

- Direct Access to GDSII hierarchy
- Single/Multiple views
- Cross-platform Unix/NT
- High-level architecture
- Interactive level-of-detail control
- Unlimited overlays
- Exploits flexible graphics pipeline
- Streaming display yields low memory profile for very large models
- Application template available

AVS in EDA

<< The End >>



Parallelization Strategies for Volume Visualization

W T Hewitt
Manchester Visualization Centre
Manchester Computing
University of Manchester

Issues in Parallel Computing

- Load Balancing
- Levels of Granularity
- Nature of Parallelism
- Data Coherence
- Data Access
- Scalability

Load Balancing

- To encourage an equal distribution of work throughout the processors
- Each processor is used as effectively as its neighbours
- Equal amounts of work mean that all processors finish their work at the same time
- Typically address by task partitioning:
  - Static assignment of large tasks to processor
  - Dynamic assignment of smaller tasks

Static Task Assignment

- Typically the number of tasks is equal to the number of processors
- All tasks are estimated to take approximately the same amount of time
- Advantages
  - Communication overhead small due to large tasks
  - Task startup cost is minimised, and scheduling overhead reduced
- Disadvantages
  - Requires some pre-processing to ensure that tasks are roughly the same size

Dynamic Task Assignment

- Number of tasks T >> number of processors P
- During run time processors are assigned tasks from a pool of tasks waiting to be executed
- Processors work on tasks until the task pool is empty, then processor stays idle until all tasks complete

Dynamic Task Assignment

- Advantages
  - Idle time is usually small and has minimal impact
  - Task size does not need to be calculated a priori
  - Load balancing solved dynamically
- Disadvantages
  - If task *granularity* is too small then may lead to excessive communication

Levels of Granularity

- Granularity — A measure of the size of an individual task to be executed on a parallel processor

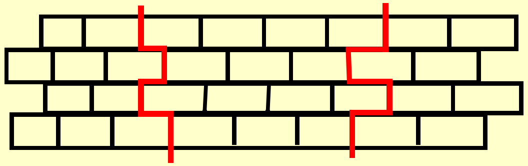
  Stone, High Performance Computer Architecture
- Coarse
  - Execution of P modules in parallel on P processors i.e, large tasks
- Medium
  - Execution of N modules on P processors in parallel where N>>P
- Fine
  - Parallel computations of loop iterations in parallel.
    e.g., for each pixel in an image

Nature of Parallelism

- Principal types
  - Data Parallelism (Geometric)
  - Functional Parallelism (Procedural)
  - Farm Parallelism

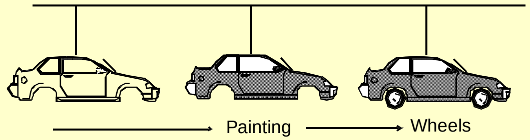
Data Parallelism

- Divide up data amongst processors
- Process different data segments in parallel
- Maybe requirement for communication at borders
- Maybe inefficient if data access patterns are unknown



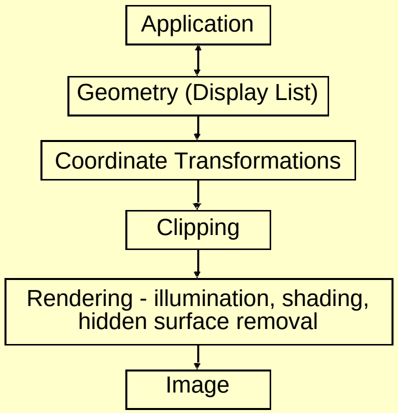
Functional Parallelism

- Different threads of control
- Decompose the algorithm into different sections, assigned to different processors
- Pipelining is a form of functional parallelism



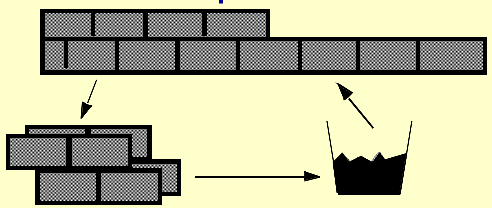
Functional Parallelism

- E.g. Graphics Pipeline



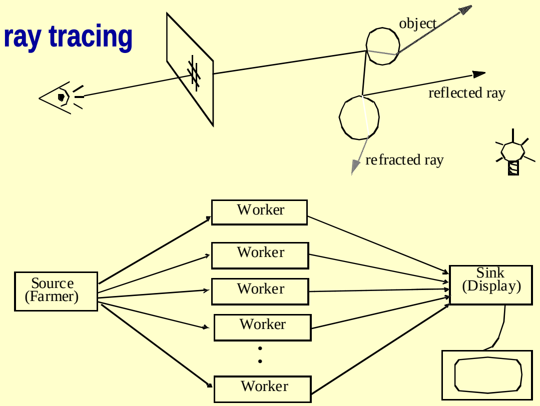
Farm Parallelism

- Usually used with dynamic task assignment. Similar tasks are generated by a “source” process and maintained in a pool
- “Worker” Processors repeatedly take tasks from the pool and perform the calculation and pass them onto a “sink”



Farm Parallelism cont.

- Example use in ray tracing



Data Coherence

- Concerned with exploiting coherence in data to avoid re-computations and number of remote accesses
- Pixel level coherence
  - High probability that neighbouring pixels have similar values in an image
- Area level coherence
  - Areas of pixel are likely to have similar values in an image, and will therefore use similar data.
  - Maybe groups of scan lines

Data Coherence Cont.

- Frame level/Temporal coherence
  - Generally there is little change between neighbouring frames in an animation sequence
  - Some algorithms/machines can exploit this data coherency between frames

Data Access

- Concerned with movement of data between processors
- Try to avoid access to data on remote processors where possible. Differences in access times between local and remote data can be dramatic, particularly on networks of workstations
- Exploit data coherence where possible

Overview of Parallel Architectures

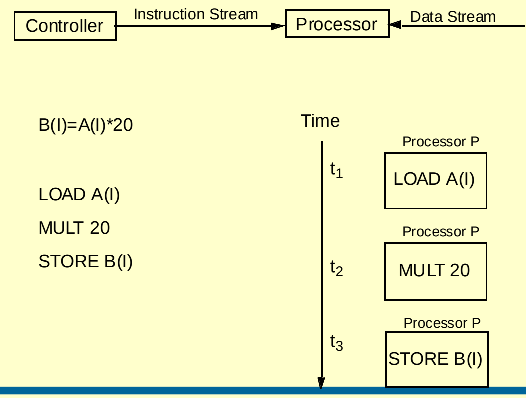
- A classification scheme for architectures

Flynn in 1972

- Outdated, however still widely used
- Four categories:
  - SISD - Single Instruction stream, Single Data stream
  - MISD - Multiple Instruction stream, Single Data stream
  - SIMD - Single Instruction stream, Multiple Data stream
  - MIMD - Multiple Instruction stream, Multiple Data Stream

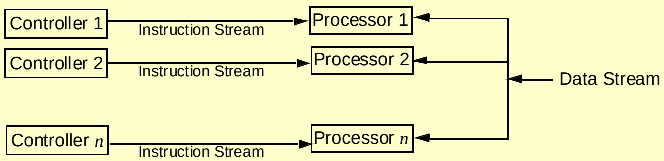
Single Instruction, Single Data Stream

- Conventional Uni-processor architecture

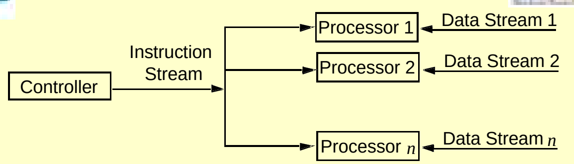


Multiple Instruction, Single Data stream

- Several processors simultaneously execute different instructions on one data stream
- Used in pipelines, e.g graphics hardware



Single Instruction, Multiple Data Stream

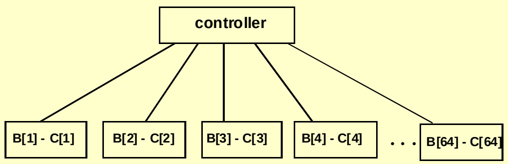


- Single instruction executed on several data streams simultaneously in lock step manner.
- In general processors are very simple and act like mindless clones.
- Used in processor arrays like the DAP, Connection Machine CM2. Also in Pixel Planes machine

SIMD Example

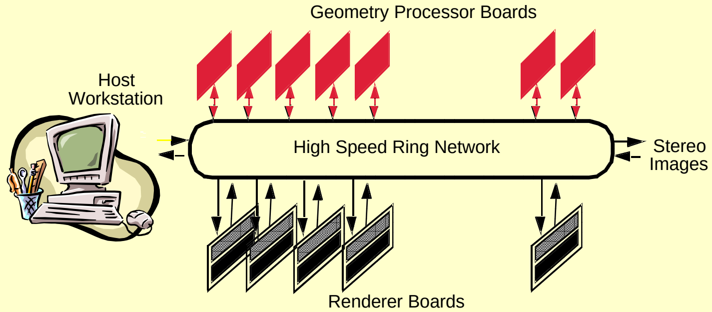
- Each element of the array is calculated simultaneously

```
for i = 1, 64 do
      answer[i] := b[i] - c[i];
```



Example Machine — Pixel Planes 6

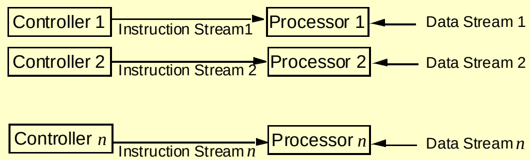
- Developed at the University of North Carolina, now marketed by Division Ltd.



Example Machine — Pixel Planes 6

- Each Geometry Processor (GP) contains two processors each with 32MBytes of local memory. A subset of the graphics database is distributed to each GP.
- All GP's operate in parallel to transform the graphics database and send rendering commands to the Rendering Processors (RPs)
- Each Rendering Processor board has 64 custom chips each with 256 processors, giving 16,384 processors per board
- The RP's work in a lock-step mode rendering up to 1 GPixels per second per board.

Multiple Instruction, Multiple Data Stream



- Essentially separate computers working together to solve a problem
- Includes networks of workstations
- All other classes are sub-classes of MIMD
- We will concentrate on MIMD

Other Architectural Issues
Connection to Memory

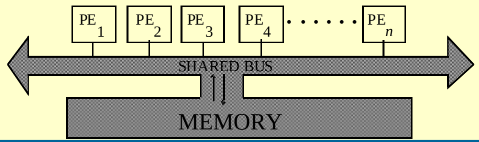
- Shared Memory
  - Bus based
  - Interconnection network
- Distributed Memory
  - Message passing
- Virtual shared memory

Shared Memory

- One common memory block between all processors
- Connection either by shared bus or inter-connection network

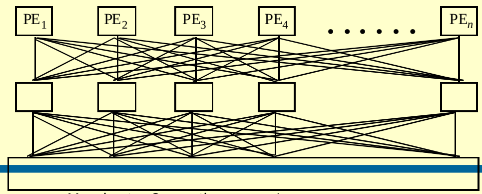
Bus Based Shared Memory

- Since bus has limited bandwidth, number of processors which can be used is limited to a few tens of processors
- Examples include Encore, Sequent, SG Power series



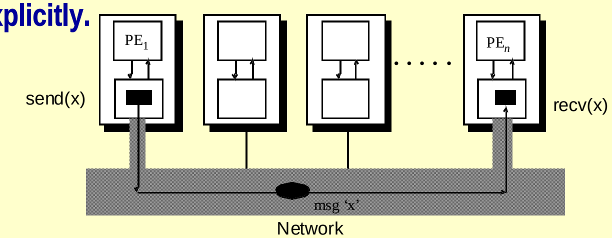
Switch based shared memory

- Utilises complex inter-connection network to connect processors to shared memory modules.
- May use multi-stage networks
- Increases bandwidth to memory over bus-based systems
- Every processor still has access to global store
- In general provide Non-Uniform Memory Access



Distributed Memory

- Message Passing. Memory physically distributed throughout the machine. Each processor has private memory
- Contents of private memory can only be accessed by that processor. If required by another processors then it must be sent explicitly.

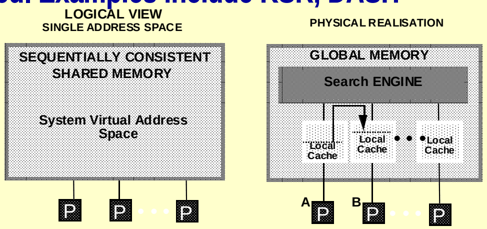


Distributed Memory

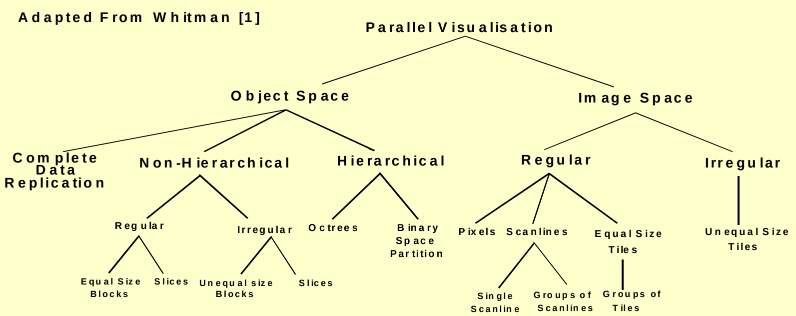
- In general machines can be scaled to hundreds/thousands of processors
- Considered difficult to program, due to message passing, and difficulty of debugging.
- Examples include Intel Paragon, Meiko CS2 and Cray T3D, IBM SP2

Virtual Shared Memory

- Objective: scalability of distributed memory with the programmability of shared memory
- Global address space mapped to physically distributed memory
- Data moves between processors "on demand", i.e as it is accessed. Examples include KSR, DASH



Taxonomy of Parallel Visualisation Decompositions

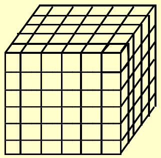


Object Space Complete Data Replication

- All data is held locally at each processor
- Simple parallelisation since multiple instances of the same sequential algorithm
- No communication during the compute phase
- Wasteful of memory. Does not scale with size of data set
- Impractical on massively parallel machines since high cost in initial data distribution
- Only useful for read-only data

Object Space Non-Hierarchical

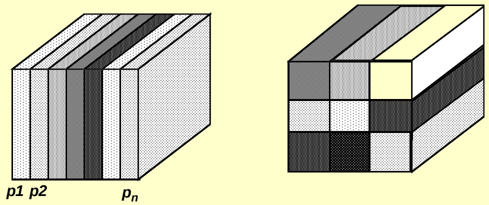
- **regular decomposition — equal size blocks**
  - Break down the data set into regular 3D regions
  - Each processor works on separate 3D regions



  - Can lead to poor performance if load balancing is not handled correctly

Object Space Non-Hierarchical

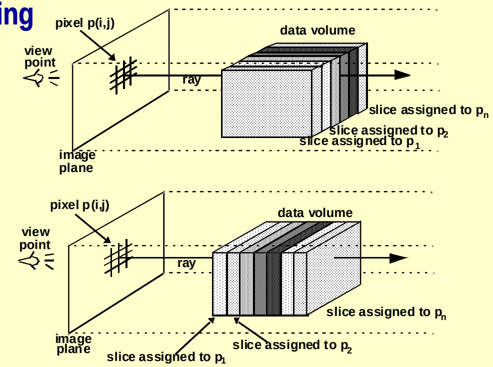
- **Regular Decomposition — Slices, Slabs or Shafts**



  - Suited to architectures which are ring based or chains since only nearest

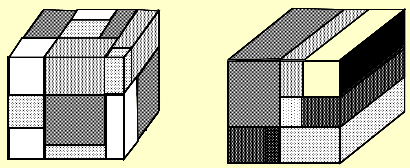
Problems

- **Means that for some algorithms performance is view-dependent e.g., ray casting**



Object Space Non-Hierarchical

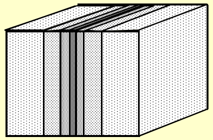
- **Irregular Decomposition — Unequal sized blocks**
  - Break the data set up so that each block represents similar amounts of work



  - Hard to estimate this, requires pre-processing

Object Space
Non-Hierarchical

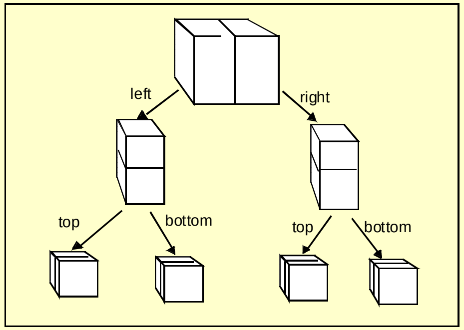
- Irregular Decomposition — Slices or Slabs



  - Slices may be different widths to aid load balancing
  - Again requires pre-processing to determine slice width

Object Space
Hierarchical Approaches

- Kd Tree Data Partitioning [2]



Object Space
Hierarchical Approaches

- Octree Subdivision [3]

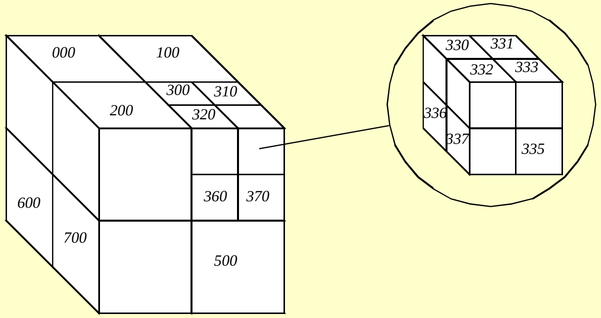
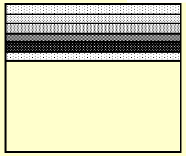


Image Space
Regular

- Processor per pixel
  - very fine grained parallelism
  - In general used on SIMD machines
  - Time to render one pixel may be several orders of magnitude different to time to render another
  - Should be load balanced if many more pixels than processors (i.e task queue)

Image Space
Regular



- **Scanlines**
  - Single
  - Groups
- **Assumptions**
  - Distribute Scan lines interleaved in round-robin fashion
  - Some pixels on a scan line are computed trivially, while others take longer to compute
  - Hence, should be intrinsically load balanced

Image Space
Regular

- **Regular Spaced Pixel tiles**
  - Straightforward to implement
  - Exploits area coherency in neighbouring pixels
  - Each block is generally independent of the other blocks, so it is not necessary for any communication between blocks to take place
  - Can vary the size of the blocks to suit the granularity of the machine
  - Often implemented using Task-Farm parallelism

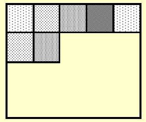


Image Space
Irregular

- **Again exploits area coherency**
- **Uses pre-processing to calculate load for each pixel tile**
  - Subsample the image plane, and keep a note of the time taken to calculate the subsampled pixels
  - Cluster tiles together to form new tiles with roughly equal times to render
  - Should be better load balanced

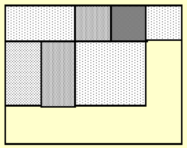
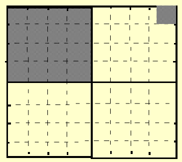


Image Space
Irregular

- **Groups of tiles**
  - Each processor works on a block of square tiles held in its queue
  - When a processor finishes rendering the tiles in its queue, it steals tiles from other processor queues



Parallel Volume Visualization Algorithms

W T Hewitt

Manchester Visualization Centre

Manchester Computing

University of Manchester

Introduction

- **Parallel Surface Extraction**
  - Marching Cubes on a SIMD machine
  - Marching Cubes on a DM-MIMD machine
  - Marching Cubes on a VSM-MIMD machine
- **Parallel Direct Methods**
  - Ray casting on a network of workstations
  - Ray casting on DM-MIMD
  - Ray casting on VSM-MIMD
  - Splatting on DM-MIMD
- **Not a full survey of parallel volume visualisation algorithms**

Parallel Surface Extraction

- **Recall Marching Cubes Algorithm:**
  - Create a bit index by classifying each voxel as being inside or outside the surface
  - Look up edge intersection in a pre-computed table
  - Interpolate edge intersection and gradients
  - Construct Triangles
- **"Marching Cubes on a Connection Machine CM2" Hansen et al. [1]**
- **Target Architecture CM2**
  - SIMD machine with 64,000 processors. Each processor can simulate a number of virtual processors (VPs). Lock step parallelism
  - Supports fast nearest neighbour communication
  - Each processor has only 256k of memory

Parallel Isosurface Extraction SIMD

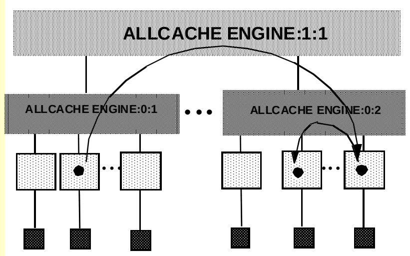
- **Parallelisation Strategy**
  - Each voxel is distributed to a different virtual processor - essentially one voxel per processor - very fine grained
  - Each virtual processor then performs communication with its nearest neighbours to obtain the neighbouring voxels required for edge intersection and gradient calculations
  - Edge intersections are calculated locally using trilinear interpolation. Edge intersections are then communicated to neighbouring virtual processors until a complete triangle list is produced.

Parallel Isosurface Extraction SIMD

- **Implementation**
  - The full 256 case look up table is used rather than 15 base cases to avoid handling special cases in lock-step manner. All VPs need access to the table but don't have sufficient memory to hold it
  - On CM2 groups of 32 processors can share memory. Therefore the table is stored once for every 32 processors. This allows the table to be accessed in parallel by the 32 processors
- **Observations**
  - The algorithm is not portable to other machine as it exploits explicit characteristics of the CM2
  - The lock step nature of the machine means that performance is bounded by the number of triangles in a given voxel, rather than the total number of triangles

Parallel Isosurface Extraction VSM

- **"Marching Cubes on a Virtual Shared Memory Architecture", Grant et al. [2]**
- **KSR Architecture**



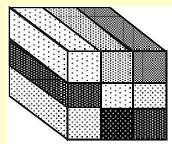
Data Location	Latency
PE on a different AE:0	30 µs
Different PE same AE:0	8.7 µs
Local-cache	0.9 µs
sub-cache	0.1 µs

Parallel Isosurface Extraction VSM

- **KSR Architecture**
  - Each processor has a 32 Mbyte cache and a 0.5 Mbyte subcache
  - Data is moved on demand when it is referenced. To distribute data around the machine, distribute the iteration space rather than use explicit message passing
  - The mapping between an address and a physical location is handled by the hardware
  - The unit of data movement is a subpage of 128 bytes. i.e when an address is referenced then that address and the surrounding subpage are fetched

Parallel Isosurface Extraction VSM

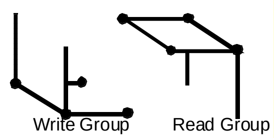
- **Parallel Design**
  - Crucial on this machine to exploit data coherency to minimize the amount of non-local communication
  - Need to avoid writing to shared data structures where possible
  - Split the data volume into shafts along the first dimension - this takes advantage of data coherency since neighbouring voxels in the shaft are neighbours in contiguous memory blocks.



  - Each processor is responsible for calculating the triangles within a shaft

Parallel Isosurface Extraction VSM

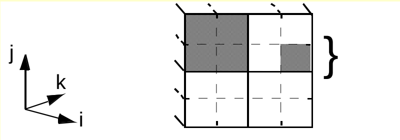
- **Implementation**
  - Each voxel shares edges with other voxels and so at the edges of the shafts there is a need to exchange edge information.
  - In reality each voxel has a “read” and a “write” group, where triangles are only calculated for its write group. Dummy references are maintained for the other edges.



  - Each processor will maintain a triangle list for its subvolume, which has some dummy references. In the final stage these lists are combined and the references resolved.

Parallel Isosurface Extraction VSM

- **Optimisations**
  - Since the vertices in each write group are stored contiguously, then when the corresponding read group references the first true vertex, all the vertex values will be fetched in the 128 byte subpage. This decreases amount of data referencing
  - Distributed task queue.



  - Each processor assigned a queue of subvolumes. Processors take subvolumes from their own queue until empty and then steal from other processor’s queues
  - Grouping like this means that more data references will be local

Parallel Ray Casting Workstation Clusters

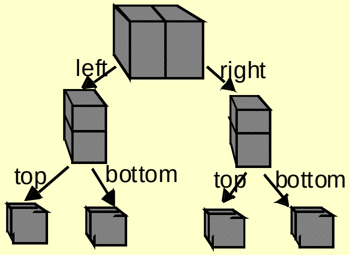
- **“A distributed parallel algorithm for ray traced volume rendering” Ma et al. [3]**
  Environment
  - A network of high performance heterogeneous workstations connected by Ethernet
  - non-dedicated computing cycles
  - message passing communication using the PVM libraries
- **Motivation**
  - Datasets often too large for a single workstation
  - Clusters are prevalent and much cheaper than MPPs

Parallel Ray Casting Workstation Clusters

- **Recall the Ray casting algorithm**
  - Shade the acquired data to obtain a volume of colour values
  - Classify the data to obtain a volume of opacity values
  - Cast a ray through both volumes and takes samples along the ray
  - At each sample point calculate the colour and opacity using tri-linear interpolation
  - Composite the colour and opacity samples along the ray to produce a total colour for the ray.
- **Parallelisation Strategy**
  - Distribute data amongst processors using a hierarchical subdivision method
  - Distribute viewing information and classification tables
  - Each processor calculates a partial image for its local subvolume
  - Partial images can then be merged in parallel, ensuring load balancing

Parallel Ray Casting
Workstation Clusters

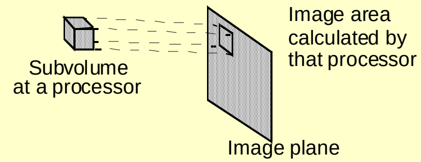
- **Data Distribution**
  - Need to establish an unambiguous back-to-front ordering, to ensure correct compositing
  - Uses the Kd Tree Method



  - Difficult to ensure load balancing, since each subvolume may not represent equivalent amounts of work

Parallel Ray Casting
Workstation Clusters

- **Calculation of the partial images**
  - Each processor has information on the view position and orientation of the image plane. Only rays within the image region corresponding to the subvolume are cast.



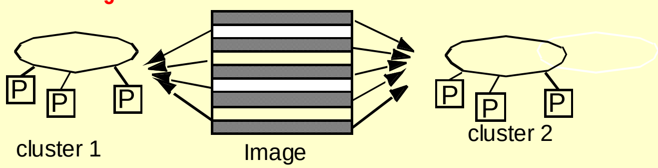
  - Ray casting takes place independently for each subvolume, so that no communication is required during this phase. End result is that each processor holds a partial image
  - Use a parallel compositing technique (see later) to reconstruct the total image from the partial images

Parallel Ray Casting
Workstation Clusters

- **Summary**
  - Workstation clusters can provide a useful environment when large datasets are to be rendered
  - Achieving real time in this kind of environment is difficult due to high latency communications
  - Algorithm is scalable due to the distribution and parallel compositing method.
  - Algorithm is also applicable to MPP systems
  - Load balancing is difficult using this kind of decomposition. Alternatives using static or dynamic partitioning can lead to either more communication or more pre-processing.

Parallel Ray Casting
DM-MIMD

- **"Volume rendering on a distributed memory parallel computer" Montani et al. [4]**
  - Implementation on a nCube 2 with 128 nodes, 4Mbytes of memory per node
- **Parallelisation Strategy**
  - Group processors into clusters. Replicate the volume in each cluster
  - Divide the image space by scanlines, so that each cluster is responsible for rendering a subset of the scanlines



Parallel Ray Casting DM-MIMD

- **Data Decomposition**
  - A slice strategy is used to decompose the copies of the data volume within each cluster
  - The justification is that the partitioning is straightforward and the communication patterns are simple
- **Ray Dataflow**
  - Rays are cast into the data volume. When a ray reaches the bounds of its slice partition it must be packaged and communicated to adjacent nodes
  - The identifier of a processor within a cluster provides a unique address for the slice partitions it holds

Parallel Ray Casting DM-MIMD

- **Load Balancing**
  - A static load balancing scheme is used.
  - Initially each processor has a uniform number of slices. A subset of rays are calculated and the load redistributed according to the partial results
- **Observations**
  - Load balancing scheme requires pre-processing step
  - Since a slice strategy is use the performance of the algorithm is view dependent
  - As more processors are added slice partitions become narrower, and rays reach their bounds sooner. This means more ray communication

Parallel Ray Casting DM-MIMD

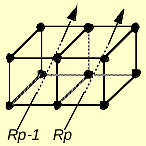
- **"Parallel Volume Rendering and Data Coherence" Corrie et al. [5]**
- **Environment**
  - Fujitsu AP1000
  - DM-MIMD message passing architecture with up to 1024 nodes. Host/node programming model
  - wormhole routed 2D mesh network
- **Parallelisation Strategy**
  - Image space task partition
  - Implementation of distributed shared memory for data distribution

Parallel Ray Casting DM-MIMD

- **Image space task partitioning**
  - Uses square pixel tiles to exploit data coherence. More tiles than processors
  - To improve load balancing implement a work item timeout period. When a processor is struggling the work is re-distributed to other processors
- **Data distribution using DSM**
  - Each processor maintains to lists - persistent and cache
  - Persistent list - Use to hold part of the volume which its serves to other processors
  - Cache list - An LRU cache which is used to hold data which is fetched from other processors persistent lists

Parallel Ray Casting
DM-MIMD

- **Data coherency**
  - Since pixels in a neighbourhood of an image will use the same data value then once the data is cached for the first ray it should be present for the second ray



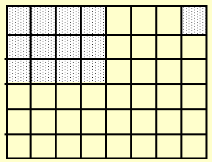
  - If the LRU cache is too small can lead to thrashing
- **Performance**
  - Efficiencies of between 80 and 95% obtained on up to 128 processors
  - Overhead of using DSM less than 20%

Parallel Ray Casting
VSM-MIMD

- **“Volume rendering on a scalable shared memory MIMD Architecture” Nieh et al. [6]**
- **Environment**
  - Stanford DASH machine. Up to 64 high performance RISC processors
  - Physically distributed memory providing a single address space logical view for the programmer. Processors are grouped into clusters
  - Each processor has a local cache which forms part of the larger global memory. If memory requests can’t be handled locally they are referred to other processors in the local cluster and then to other clusters
  - The DASH supports explicit placement of data unlike the KSR

Parallel Ray Casting
VSM-MIMD

- **Parallel Implementation**
  - Distribute pages of the data in a round robin fashion amongst the processors. This ensures no data hotspots
  - Adopt a task queue image partition scheme. Each processor assigned a block of image tiles. When its tile are completed it can grab tiles from other processors



  - Since corner pixels may be required by two different processors a scheme is adopted to avoid duplicating the computation of the corner pixels.

Parallel Ray Casting
VSM-MIMD

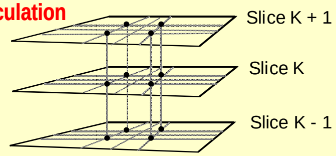
- **Summary**
  - The scheme shows good parallel efficiencies (over 80% on 48 processors) and is well load balanced
  - It takes advantage of both image coherency and volume coherency
  - For animation sequences inter-frame temporal coherency is also exploited, since neighbouring frames use broadly the same data values which are already in local caches

Parallel Splatting
DM-MIMD

- "Volume Rendering on a Distributed Memory Parallel Computer" Elvins et al. [7]
  - Distributed Splatting algorithm on distributed memory nCube2 machine
  - nCube2: 128 processors, 16 Mbytes per processor, host/node approach
  - Recall sequential splatting
    - Determine in what order the volume will be traversed
    - Classify each voxel according to colour and opacity look up tables
    - Project (splat) each voxel into image space. Use a reconstruction filter to determine extent of the splats contribution (footprint)
    - Attenuate the colour-opacity tuple with the reconstruction filter
    - Composite the attenuated tuples into an image buffer

Parallel Splatting

- Parallel strategy
  - Object space data decomposition using master-slave parallelism
- Master Process
  - Read dataset from disk and distribute slices to slave processors in round-robin fashion.
  - Actually need to distribute 3 slices for each slice that is to be splatted, since this data is required for the gradient calculation



  - Collects Image contributions from each slave and composite in correct order

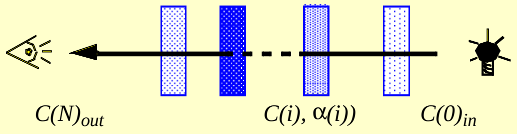
Parallel Splatting

- Slave Processors
  - Slaves performs the splatting operation on the slices that they have been assigned
  - Image contributions are then passed back to the master for composition
  - If more slices still to be rendered then obtain more slices from master
- Performance Optimisations
  - Split the distribution and composition roles of the master to avoid a bottleneck
  - *Image Coherency*. Each slave calculates the non-black sections of the image contributions it generates and just passes these back. Saves on communication and reduces the composition task
  - *Distribute groups of slices*. Instead of sending out groups of 3 slices, send out larger groups depending on memory capacity. Reduces short communications and also unchokes the image contribution bottleneck.

Parallel Compositing

“Over” Operator

- Porter and Duff “Over” operator widely used for compositing, or “alpha blending”

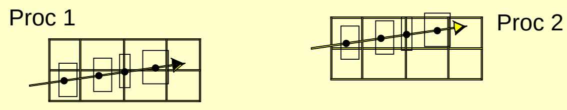


- For a sample location S(i), with colour C(i) and opacity a(i), then

S(i)over S(j) = S(I) + (1- alpha(I))S(j)

“Over” Operator

- The “over” operator is associative
- This means that samples which have been computed by different processors can be composited is the correct order is maintained

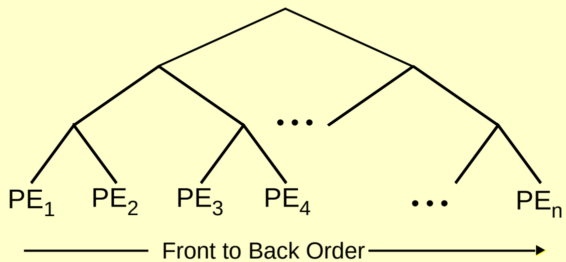


$S(P_1) = S(1) over ...S(k)$ $S(P_2) = S(K+1) over ...S(N)$

$S(P) = S(P_1)\ over\ S(P_2)$

Binary Compositing

- Data required for a particular section of the image is distributed amongst the processors
- Data must be re-composed in the correct order



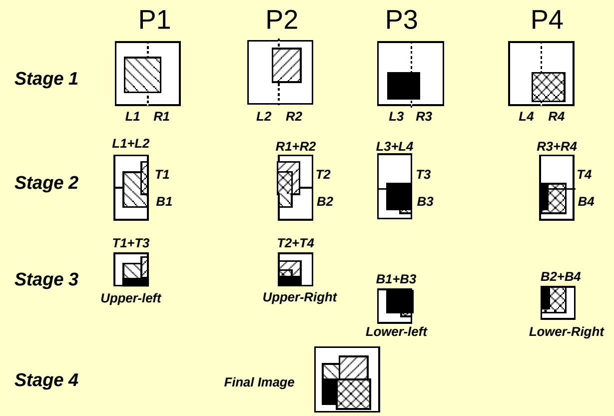
Binary Compositing

- Composite data at each stage of the tree starting at bottom
- At each stage half the processors become idle after they pass their data on. Poor load balancing
- Final compositing step is done by one processor — a significant bottleneck
- Better to adopt a parallel approach, where each processor is kept busy until the final stage. e.g. Painter and Hansen’s Binary Swap Compositing[8]

Binary Swap Method [8]

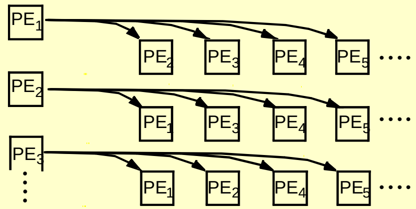
- At each stage of the compositing tree, swap half of the PE's image with a partner
- At each stage the size of the image at each PE becomes smaller
- All processors are active throughout the composite phase

Binary Swap Compositing



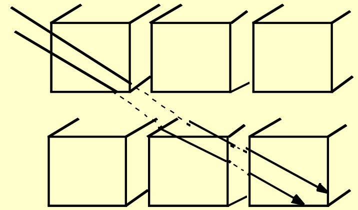
Direct Send Compositing[9]

- Split the tasks of rendering and compositing. For each rendering processor allocate a compositing processor.
  - These may or may not be the same processors that are doing the rendering
  - Subdivide the image
  - After the rendering process completes all compositing is performed in parallel
- Maintains load balance



Ray Dataflow Approach

- If ray dataflow has been used as the parallelisation method, then composite as the rays progresses



- Alternatively wait until all processors have rendered their sub images and then propagate a ray to do the compositing as a post process

Ray Dataflow Approach

- **Observations**
  - Integrated with the rendering stage
  - If done as a post process, then a relatively small number of messages
- **Disadvantages**
  - Same as ray dataflow approaches i.e not scalable
  - means that some processors are idle waiting for rays to progress to them

Parallel Volume Visualization

References

Parallel Methods References

[1] S Whitman "Multiprocessor Methods for Computer Graphics Rendering" Jones and Bartlett, Boston 1992

[2] J Bentley "Multidimensional Binary Search Trees used for Associative Searching" Comm. ACM 18,8 (Sept. 1975)

[3] L Doctor et al. "Display techniques for Octree-Encoded Objects" IEEE CG&A (July 1981)

[4] S Green "Parallel Processing for Computer Graphics" Pitman, London 1991.

[5] H Stone "High Performance Computer Architectures" Addison-Wesley, 1993

[6] I Foster "Designing and Building Parallel Programs", Addison-Wesley, 1994

Parallel Volume Visualization

[1] C Hansen, P Hinker, "Massively Parallel Isosurface Extraction" Proceedings of Vis'92

[2] A Grant, W Haslam "Marching Cubes using Virtual Shared Memory", Technical Report MVC, University of Manchester

[3] K Ma, J Painter et al. "A Data Distributed, Parallel Algorithm for ray Traced Volume Rendering" Proceedings of the 1993 Parallel Rendering Symposium. San Jose. IEEE

[4] C Montani, R Perego et al., "Parallel Volume Visualization on a Hypercube Architecture", Proceedings of the 1992 Workshop on Volume Visualization (1992), 9-16, Boston

[5] B Corrie, P Mackerras, "Parallel Volume Rendering and Data Coherence", Proceedings of 1993 Parallei Rendering Symposium, (1993), IEEE

[6] J Nieh, M Levoy "Volume Rendering on Scalable Shared Memory MIMD Architectures Proceedings of the 1992 Workshop on Volume Visualization

[7] T Elvins, "Volume Rendering on a Distributed Memory Parallel Computer" Proc.Vis'92, Boston

[8] K Ma, J Painter et al. "A Data Distributed, Parallel Algorithm for ray Traced Volume Rendering" Proceedings of the 1993 Parallel Rendering Symposium, San Jose, IEEE

[9] M Levoy, "Efficient ray Tracing for volume data" ACM Transactions of Computer Graphics 9(3), July 1990

[10] G Cameron, P Underill, "Rendering Volumetric medical Image data on a SIMD Architecture Computer", Proc. of the Third Eurographics workshop on Rendering, 1992

[11] J Rowland, et al., "A Distributed, Parallel, Interactive Volume Rendering Package", Proc. Vis'94, IEEE

[12] T.Ford, A.J.Grant, "Volume Rendering on the Computing Surface", Proceedings of the Parallel Computing & Transputer Applications Conference, Barcelona, 1992

[13] A.J.Grant, M.K.Zuffo "Approaches to Direct Volume Rendering on Distributed Memory and Virtual Shared Memory Parallel Computers", Proceedings of BCS Conference on Parallel Processing for Graphics and Scientific Visualization, Edinburgh, May 1993

[14] M.K.Zuffo, A.J.Grant, "RTV: A Package for the Visualization of three dimensional medical data", Proceedings of SIBGRAPI'93, Pernambuco, Brazil,1993.

[15] U. Neumann, "Parallel Volume Rendering Algorithm Performance on Mesh Connected Multicomputers" Proceedings of the 1993 IEEE Parallel Rendering Symposium, San Jose 1993

[16] J Challinger, "Parallel Volume Rendering on a Shared Memory Multiprocessor", Technical report UCSC-CRL-91 - 23, University of California, Santa Cruz,1992

[17] S Green, D Paddon, "Exploiting Coherence for Multiprocessor Ray Tracing", IEEE CG&A 9(6), 1989

Framework Technologies and Methods for Large Data Visualization

Ian Curington

Advanced Visual Systems Ltd.

ADVANCED VISUAL SYSTEMS INC.

Optimizing Visualization Systems

Outline

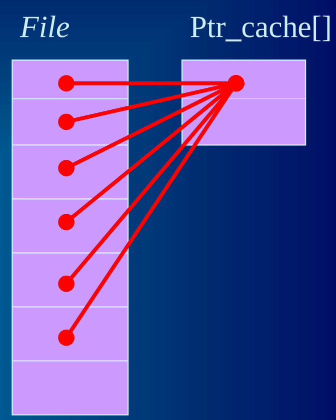
- File Access Methods
- Active Data Repository
- Data Management Middleware
- Large Unstructured Mesh Methods
- Large Data Management System Design
- Geometric Surface Reduction
- Multi-Pipe / Multi-Channel Rendering

File Access Methods

- Out-of-Core Visualization, beyond Virtual Memory
- ASCII
  - **Pre-Parse to identify counts, sections, then seek**
  - **Offline converter to binary**
- BINARY
  - **Endian Issues - Portability**
  - **Contiguous Regions**
  - **Structured Access**
- Split Files - not practical for > 2000 files/dir
- » Basic I/O Optimization Effects Visualization!

Binary File Pointer Caching

- Common Uses:
  - **Start of Substructures**
  - **Time Steps**
  - **Scalar / Vector Variable Selection**
  - **Node Displacements**
- Application Specific Virtual Memory Cache
- Scalable, Portable
- Easy to Implement



fseek(ptr_cache[i],…)

Map your File to Virtual Memory

“MMAP”

- Available on UNIX
- Very Simple to use, file becomes *array[i]*
- Powered by system virtual memory manager
- File size limited by VM
- Handles random access easily
- Eats up system resources
- Not available on Windows

Databases and Systems Software for Multi-Scale Problems

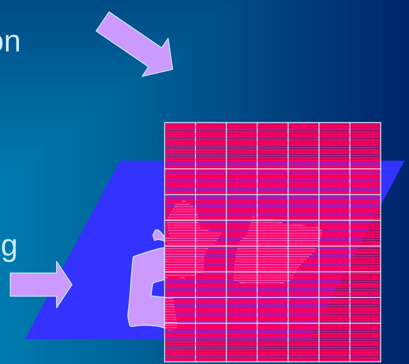
Joel Saltz
University of Maryland College Park
Computer Science Department
Johns Hopkins Medical Institutions
Pathology Department
NPACI
(presented at LDV’99)

Active Data Repository (ADR)
Common Themes

- Spatial/multidimensional multi-scale, multi-resolution datasets
- Multiple spatio-temporal queries
- Complex preprocessing
- Dataset exploration or program coupling

Typical Query

Output grid onto which a projection is carried out

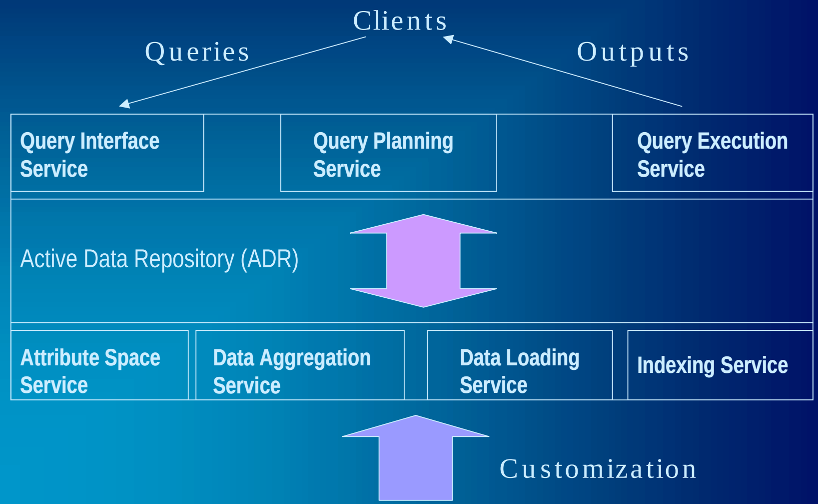


Specify portion of raw sensor data corresponding to some search criterion

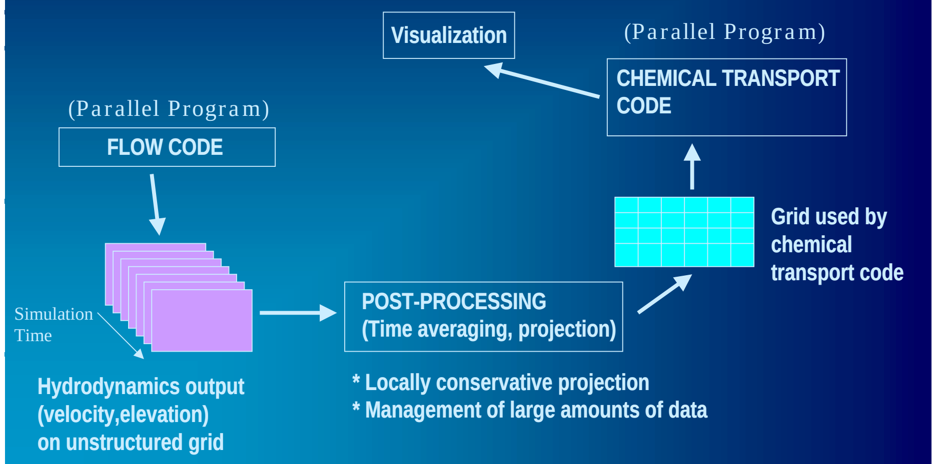
Components of System Software Architecture

- Spatial Queries and filtering on distributed data collections
  - **Spatial subset and filter (ADR')**
  - **Load disk caches with subsets of huge multi-scale datasets**
- Toolkit for producing data product servers
  - **C++ toolkit targets SP, clusters**
  - **Compiler front end**
    - **extension of inspector/executor**

Architecture of Active Data Repository

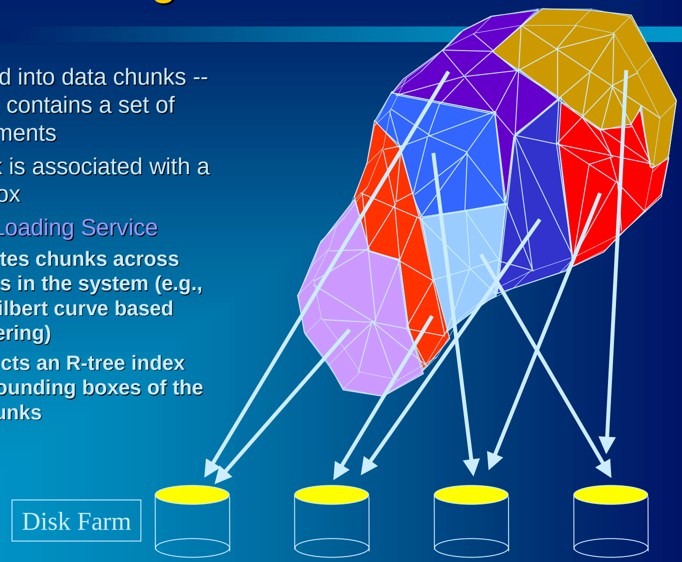


Water Contamination Studies

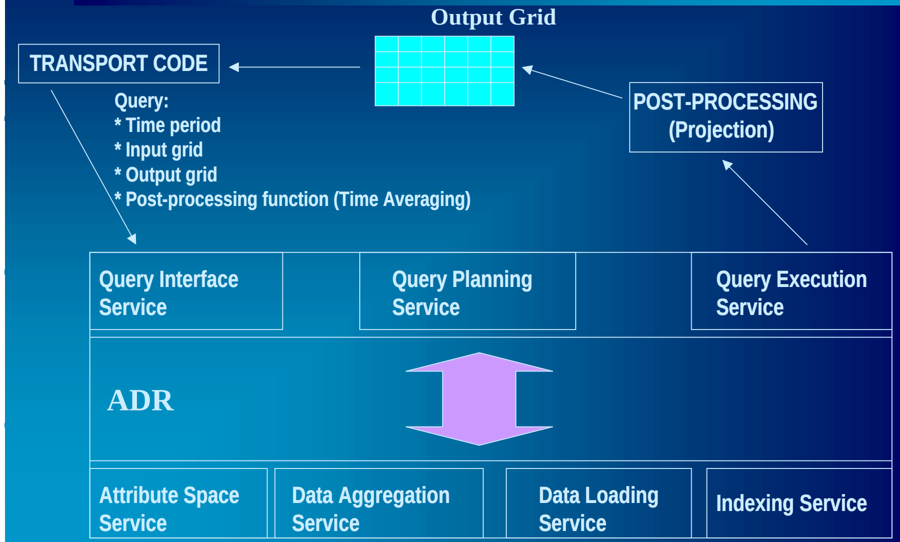


Loading Grids into ADR

- Partition grid into data chunks -- each chunk contains a set of volume elements
- Each chunk is associated with a bounding box
- ADR Data Loading Service
  - **Distributes chunks across the disks in the system (e.g., using Hilbert curve based declustering)**
  - **Constructs an R-tree index using bounding boxes of the data chunks**



Water Contamination Studies



Data Management Systems for Large Scale Visualization

Reagan W. Moore
San Diego Supercomputer Center
National Partnership for Advanced Computational Infrastructure (NPACI)
http://www.npaci.edu/DICE
(presented at LDV'99)

Data Management Middleware

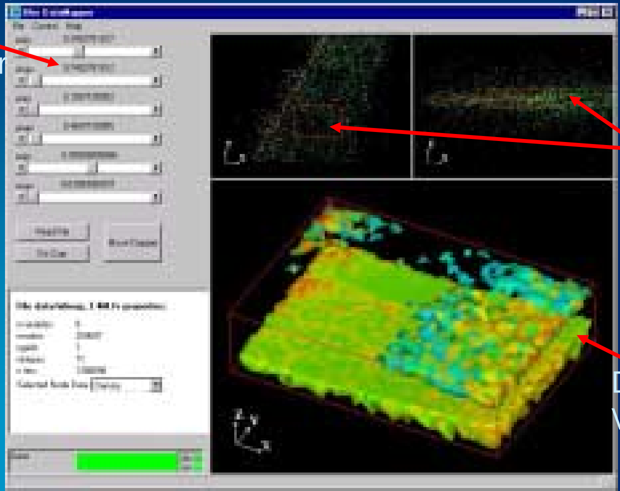
- Data Access
  - Information discovery system - Mediators
  - Distributed collection management system
  - Authentication system
  - Authorization system
- Data Movement
  - Remote execution environment for data sub-setting
  - Data manipulation support
    - Encapsulation of data subset as digital object
  - Data resource protocol support

Large Unstructured Data Visualization

- Typical of Large Aerospace CFD meshes
- Unstructured tetrahedral/prism/triangle elements
- Abstract visual spaces (pressure fields)
- Lack of orientation queues
- Selective Region/Quantity Navigation

Large Tetrahedral Mesh Navigation

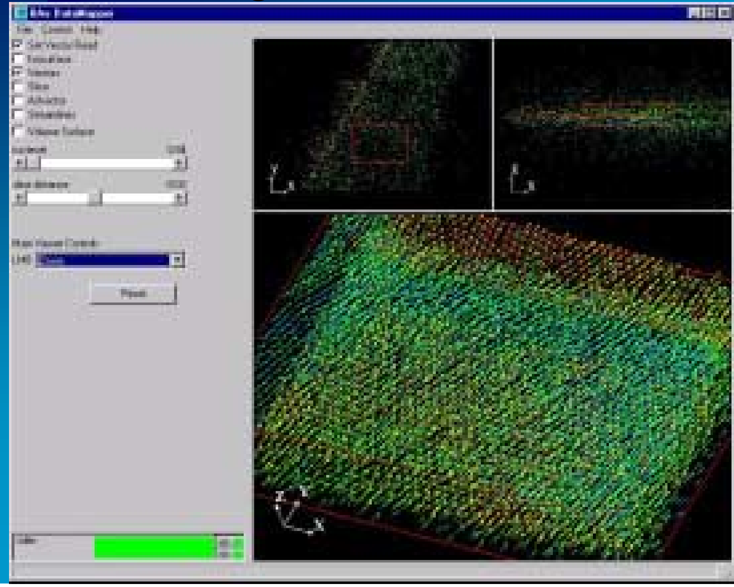
Visualization Tool Parameter Selection



Global Region AOI

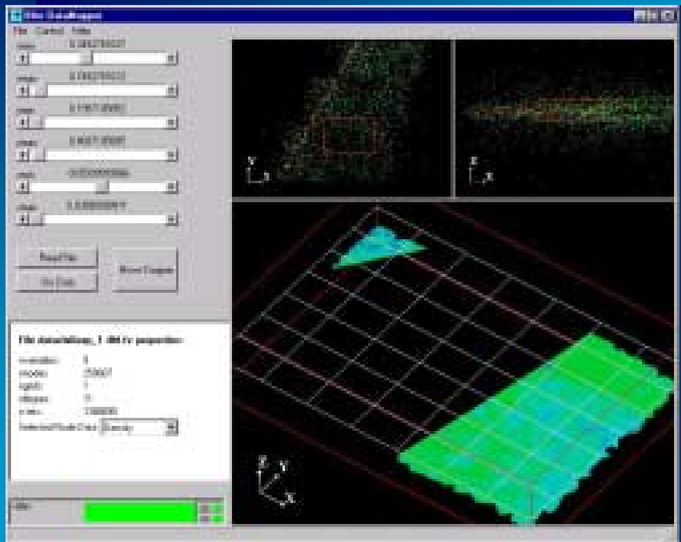
Detail Visualization

Large Tetrahedral Mesh Navigation



Localized Vector Field

Large Tetrahedral Mesh Navigation



Arbitrary Sub-Region Cross-section

Large Aircraft Dataset Example

- Aircraft External CFD Mesh data
- 50-200MB + dataset sizes typical
- Custom Module writing with C++ api
- C Style File Pointers to node data and coordinate attributes
- Downsizing of Connectivity and Mesh Coordinate Information

Nodes, Conn, 9 Vars:

1.4M Tets - 39.7MB
2.2M Tets - 63.4MB
3.8M Tets - 106.9MB
5.8M Tets - 166.3MB
8.2M Tets - 236.3 MB

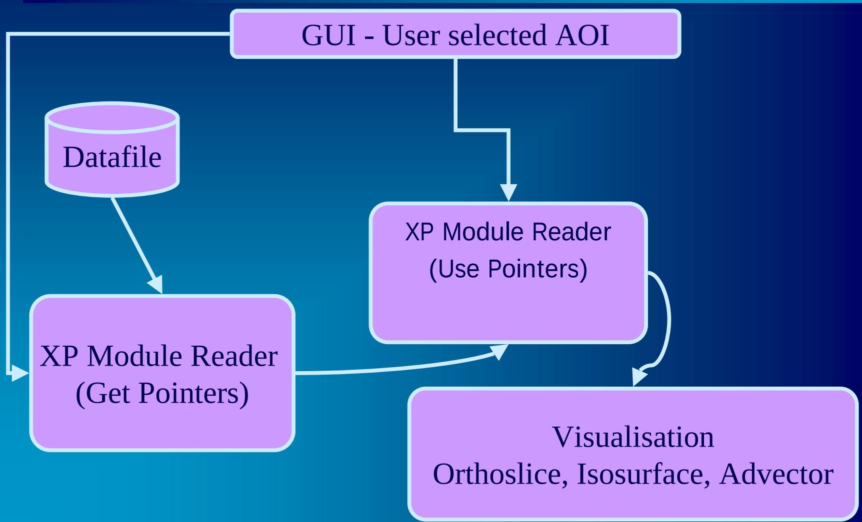
Example Test Case: Large Aircraft Mesh - Functionality

1) Sample Mesh Coordinates - Create Summary Point Mesh and Display
2) User selects Area of Interest (AOI) using the point mesh
3) AVS/Express Reads relevant part of coordinate datafile (including connectivity) to create tetrahedral mesh
4) A default component is read and pointers set up for the node datas within the datafile.

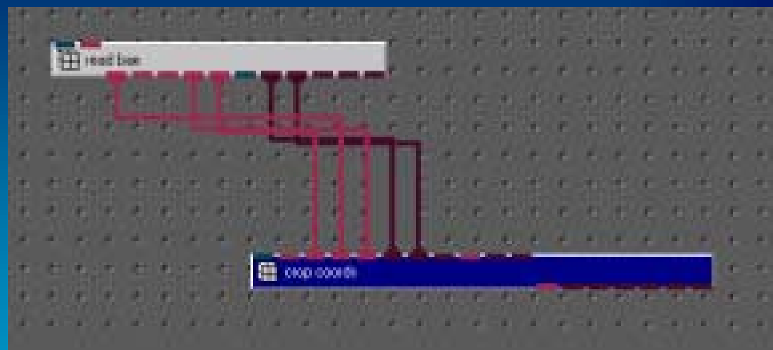
Test Case, Continued...

5) Rendering of Scene in Viewer
6) Selection of various visualization techniques (eg Orthoslice, Isosurface, Streamlines)
7) User can select new scalar/vector node data which is read directly from file
8) User can reread for different AOI

Methodology



Express Environment



Reader passes the appropriate file pointers to the crop coords which selects data from a datafile. In this way memory consumption within the application is minimized.

Case Study: LadMan

- LadMan:
  A Large Data Management System
- Coupled to visualization system
- Vistec Software, Berlin
- http://www.vissoft.de

Large Data Management

- **Situation**:
- you Generate large amounts of data
- you want to Store the data
- you want to Access the data

Large Data Management

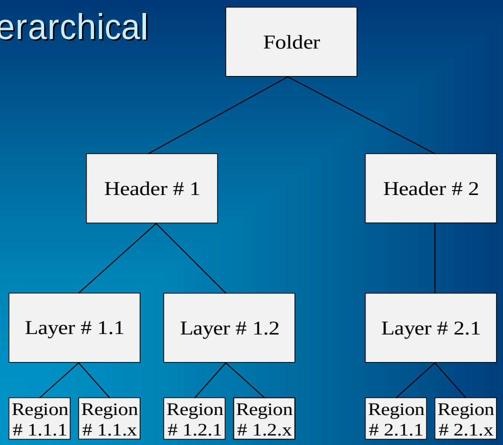
- **Resource Problems**:
- Requires large disk space to store the data
- Difficult to access your large datafiles
- Easily run out of memory processing the data

Large Data Management

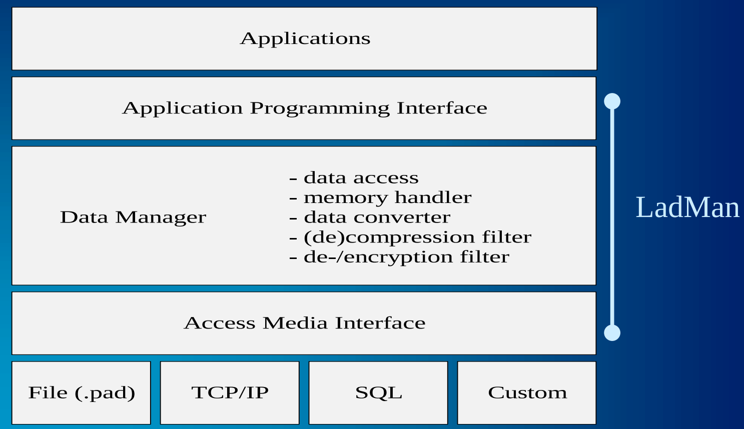
- **Solution Example**:
- Lossless data compression to save diskspace
- Smart readers to access the compressed data
- Subsampling while reading to save memory

Large Data Storage Model

- Hierarchical



Architecture



Data Types

- Mesh Types:
  - **structured data (multidimensional)**
  - **unstructured data (spatial)**
- Storage Classes:
  - **char, uchar, short, ushort, int, uint,**
  - **long, ulong, float, double**
  - **scalar, vector, tensor**

Data Partitioning Methodology

Structured:

- Multi-dimensional
  - **uniform fields**
  - **rectilinear fields**
  - **irregular fields**

Unstructured:

- Spatial data
  - **points**
  - **lines**
  - **triangles**
  - **meshes**
  - **cells**
  - **tetrahedra**
  - **...**

Data Partitioning Methodology (2)

- Structured data are tiled in dimensions
- Unstructured data are tiled in space
- System accesses one region at a time
- Multi-dimensional/multi-spatial tiling
- User defines region dimensions according to needs of accessing the data

Data Compression

- Five lossless compression algorithms built-in
- Developers can integrate own algorithms
- User selects his favorite algorithm, or
- by default LadMan selects the one with the best compression rate

Security: Data EnCryption

- Encryption of database access:
  - **all hierarchies are encrypted but the data itself**
  - **increases the access speed**
- Encryption of the complete database:
  - **all data are encrypted**
  - **prevents unauthorized users to look at the data using tools like vi, more, hexdump, etc.**
- Performance / Security trade-off

Large Data Storage Mechanics

- LadMan stores its data in an platform independent way
- Databases are identical on all platforms
- A database is accessible from all platforms
- fast conversion from LadMan to native format when using builtin compression algorithms (much faster than xdr)

Data Access: Memory Handler

- user defines LadMan memory limit
- LadMan caches region data in memory
- LadMan frees the oldest or less used region if the defined caching limit is reached

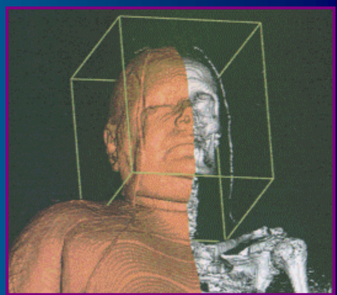
Data Sampling, Access Methods

- cropping
- downsizing
- interpolating
- zooming
- mirroring
- stretching
- slicing

(Operations directly on file-access, not in memory)

LadMan Storage Example

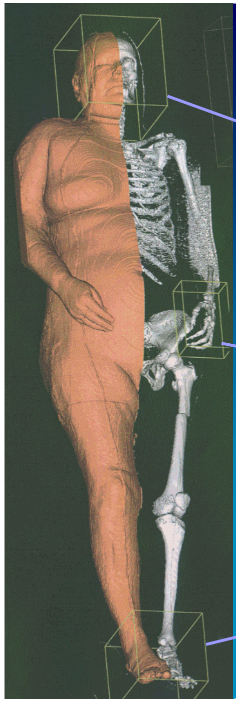
- Visible Human DataSet (Female)
  US Department of Health & Human Serv.
- CT-Scans: 1734 slices, 512x512x(16-bits)
- Original Size: 910 Mbytes
- Ladman Size: 285 Mbytes

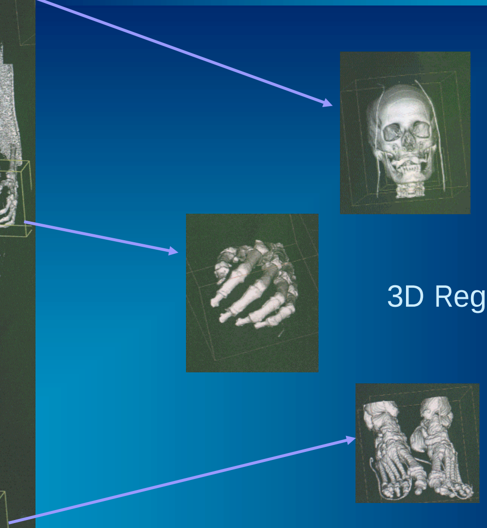


Visible Human

- Arbitrary Slice
- Crop Region Selection Control
- Overview of whole
- High Resolution Detail access

Visible Human Example (2)





3D Region Selection

INDEX Project

INDEX <=> LESS IS MORE

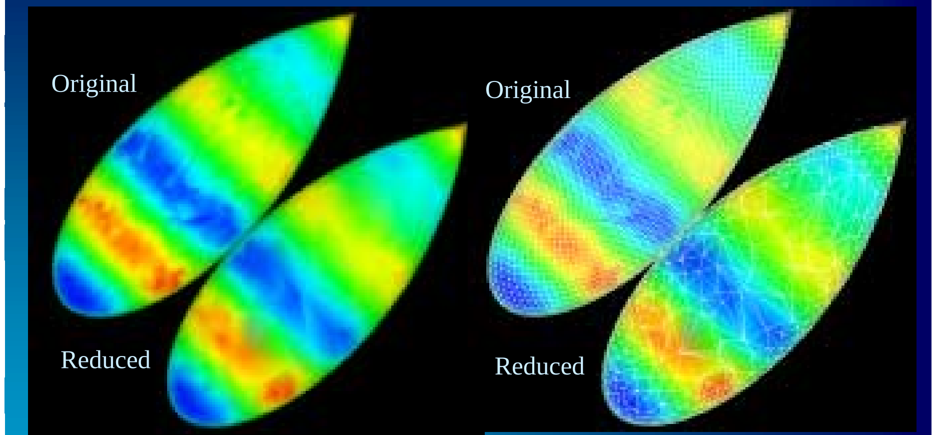
- EC Project 1996 - 1998
- Selective Data Reduction
- Automatic Compression Methods
- Alternative Vector / Scalar Representations
- Partners:
  - Advanced Visual Systems
  - British Aerospace
  - RUS, Stuttgart
  - Daimler Benz
  - Manchester Visualization Centre
  - OGS, Trieste

Surface Simplification

INDEX <=> LESS IS MORE

- **Decimation and geometric surface reduction**
  - Reduces polygon counts, memory size
  - Increases display performance
  - Secondary mesh data constrained decimation, to retain gradients
- ***decimate***
  - controlled, high quality reduction with quantifiable error measure
- ***simple-decimate***
  - less error control but is much faster

Surface Simplification



Smooth Colors

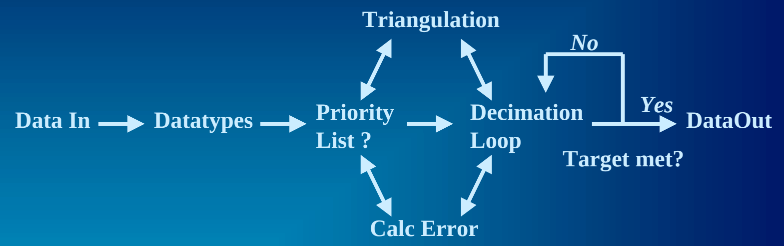
Colors and Line Mesh

Electromagnetic Surface Current Study

Surface Simplification

- Large Polygonal Models
- Models from CAD
- Large Isosurfaces
- Transport over the Web
- Interactive Response
- Preprocessing for VRML
- *Klein* "Mesh Reduction with Error Control"
- *Schroeder* "Decimation of Triangle Meshes"
- Supported in AVS/Express 5.0

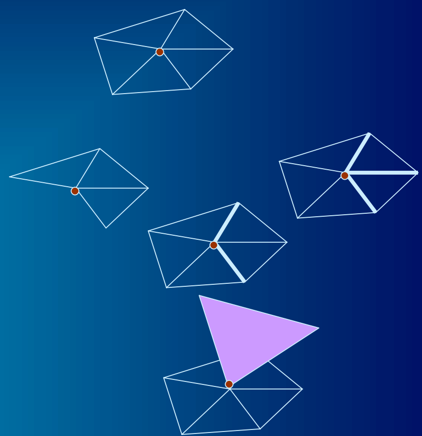
Decimation Method Overview



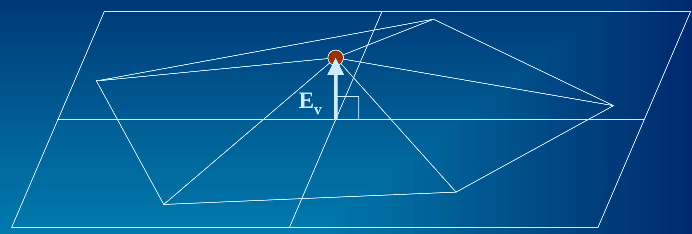
- Datatypes - efficient vertex and triangle types
- Priority List - not necessary, used only by Klein method

Schroeder - Vertex Classification

- Easily removed
- Special consideration
  - **boundary**
  - **feature edge**
  - **feature corner**
- Can't be removed
  - **complex (non-manifold)**



Schroeder - Error Calculation

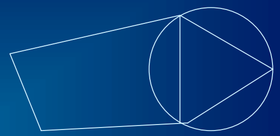


Average plane through neighbouring vertices

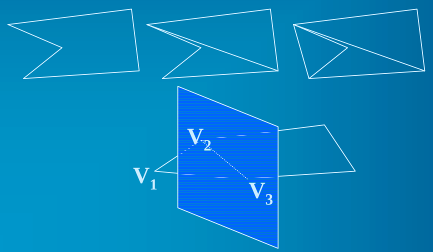
- E_v perpendicular distance the average plane
- Simple, fast but inaccurate

Schroeder - Trianglulation

- Delaunay method
  - **circumscribe triangle**
  - **only if triangle vertices contained**



- Convex Polygons
  - **v_1 opposite side to other vertices**



Schroeder - Decimation Loop

```
E0 initial error, e error increment, Ei = E0
for each iteration i
    for each remaining vertex v
            calc Ev
            if Ev < Ei
                    add Ev to each neighbouring vertex
                    triangulate hole, remove v
    Ei = E0 + e*I
    if %vertices removed > target of || no vertices removed
            stop
```

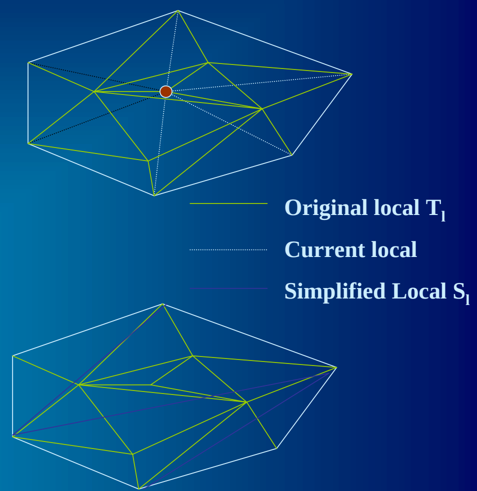

Klein - Error Calculation 1

- Hausdorff distance max(d(T,S), d(S,T))
  - **d(X,Y) is distance V_X, L_X, T_X -> V_Y, L_Y, T_Y**

- d can be calculated locally to a vertex
- Special cases lead to simplification

Klein - Error Calculation 2

- Each vertex must remember the original local mesh
- d is calculated between S_l and T_l



Klein - Priority List

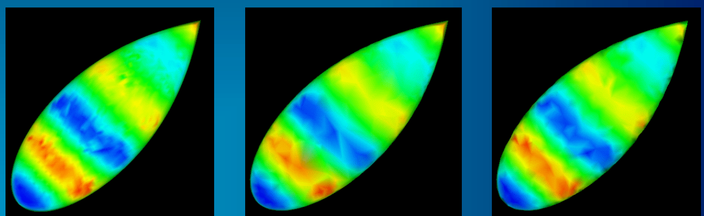
- Precalculate the potential error of vertices E_v
- Create ordered list with lowest E_v at top
- Increases accuracy and efficiency

Klein - Decimation Loop

```
E_o maximum global error
while E_v (first v in list) < E_o and % reduction unreached
    remove v
    retriangulate hole
    for all neighbouring vertices v_i
        remove from list
        update T_I references
        recalculate E_vi (slowest part)
        reinsert in list
```


Data Dependent Decimation

- Combine Geometric and Data error criteria
  - $E = \alpha.E_g + (1 - \alpha)E_d$
- E_d - gradient or curvature estimation



Original Geometric Geometric + Data

AVS / SGI
Multi-Pipe Visualization Project

ADVANCED VISUAL SYSTEMS INC.

Ian Curington

Director, Technical Marketing

www.avs.com

AVS MPU Project Research Partners

Advanced Visual Systems (AVS)

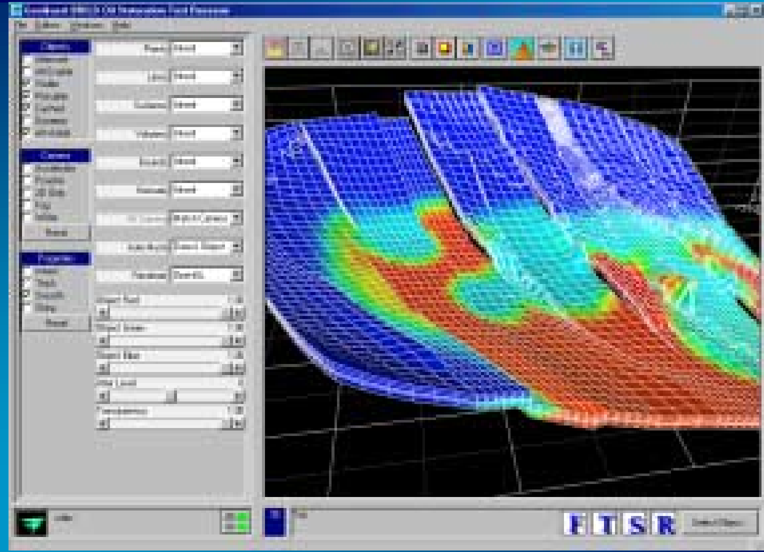
-Visualization Software & Solution Vendor

Kubota Computer Graphics, Inc. Partner

Graphics Technology Partner (MPU)

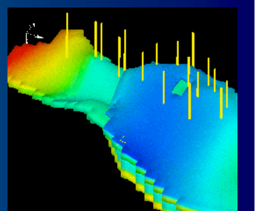
Manchester Visualization Centre,
IAC: International AVS Centre
(Research Partner)

Graphically Demanding Applications

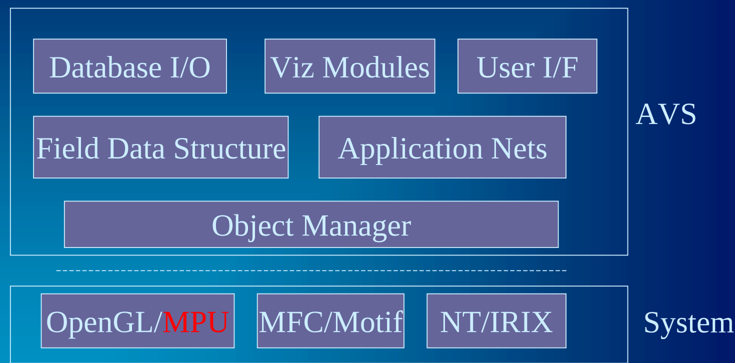


AVS/Express Rendering System

- OpenGL 1.1
  - **High-performance data structures**
  - **2D/3D Textures**
  - **BTF Volume Renderer**
  - **Stereo**
- AVS Software renderer
  (X-server, Printing, Postscript, CGM)



AVS Architecture



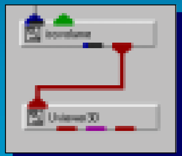
AVS Viewer Structure

Viz Module

Field Struct

GD Object

Render Methods



Viewer Macro

Scene Tree
Lights
Cameras
Interactors

View

OM

Virtual Renderer

Independent setup
(tri-strip, normals)

OpenGL dependent

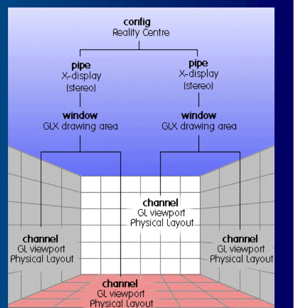
OpenGL/**MPU**

Multi-Pipe Utility (MPU)

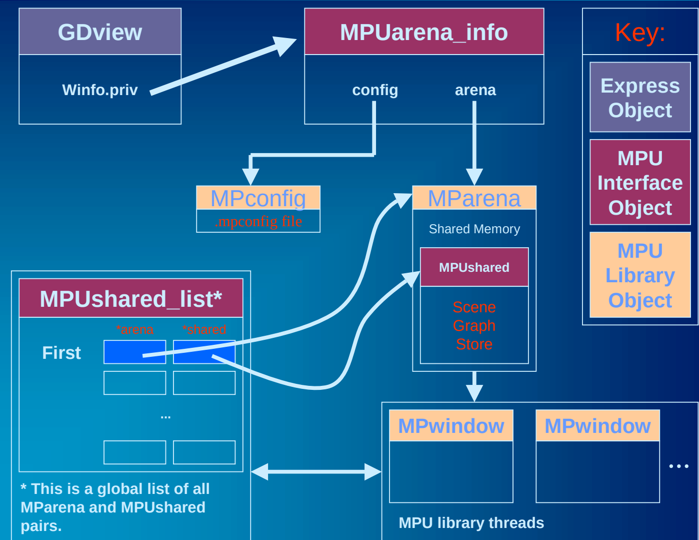
- The SGI Multi-pipe Utility (MPU) is a multi-pipe programming interface for OpenGL.
- Key features :
  Flexible pipe, window and channel configuration
  Easy integration in OpenGL application framework
  Inherent support for Stereo and Head Motion Tracking
  Transparent parallelization of rendering

MPU Key Benefit for AVS

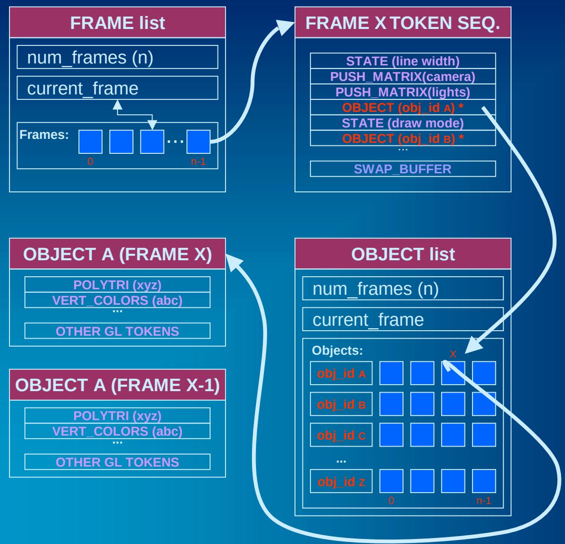
- ASCII configuration file format, the MPU provides run-time portability of OpenGL-based applications between single-user and large-scale environments such as
  - **CAVE Environments**
  - **RealityCenter Curved Screens**
  - **PowerWall, HoloDesk**
  - **ImmersaDesk**
- Single Executable Deployment



AVS/MPU Object Architecture for a Single View

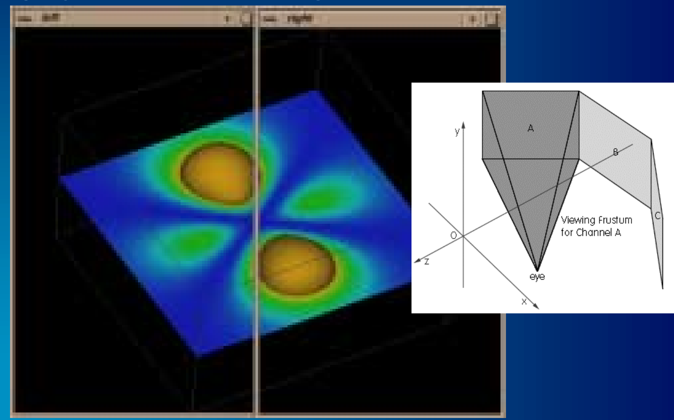


Shared Memory - Scene Graph Store Architecture



MPU external camera config

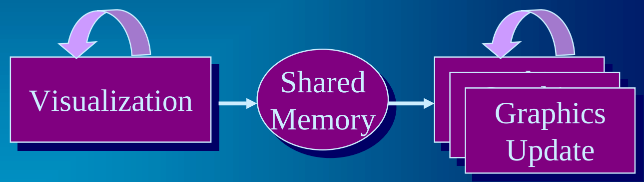
Multiple Display Channels for Single Scene



Single Executable App

MPU software pipelining

- Generation of Visualization content in parallel with graphics update
- Graphics (head tracking) de-coupled from visualization



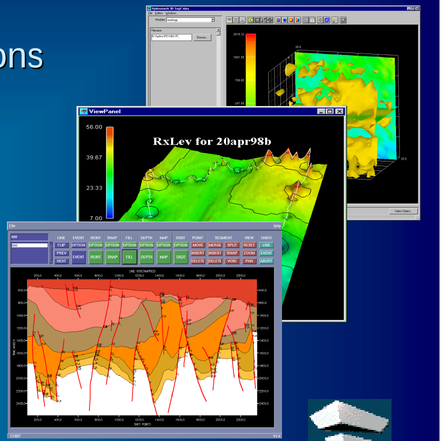
Challenges, Issues

- Performance Characterization
- Not just another graphics API
- Not just another platform
- Interaction Methods - Immersion
- Visualization techniques for large views
- Computational Steering
- Distributed, Cooperative Visualization

Conclusion: Visual Applications Developed with AVS Multi-Pipe Facility...

Goals to Provide:

- Configurable Applications
- Hi-End Visualization
- Professional Services
- Application Templates
- Research Platform



REFERENCES

Case Studies & Optimization

References

- J Saltz, "Databases and Systems Software for Multi-Scale Problems", NSF/DOE Workshop on Large-Scale Visualization and Data Management, May 1999
- R Moore, "Data Management Systems for Large Scale Visualization", NSF/DOE Workshop on Large-Scale Visualization and Data Management, May 1999, http://www.npaci.edu/DICE
- J Favre, "Visualization of Astrophysical Data with AVS/Express", proceedings of 40th Cray User Group Conference, Stuttgart, Germany, June 1998

References (2)

- J Favre, "Visualization of a Laminar-Turbulent Transition in a Supersonic Boundary Layer" An award-winning video submission at the American Physical Society (APS) in the Annual Gallery of Fluid Motion, November 1998
- P Leoncini, "Requisiti del Sistema di Visualizzazione per il Numerical Wind Tunnel" CIRA-TN-96-084, 1996
- C Meilke, CFD Research Summary, http://www.ifd.mavt.ethz.ch/members/Mielke/index.html

References (3)

- W Schmeing, "LadMan - a Large Data Management System", AVS User Group Proceedings, 1998, http://www.vissoft.de