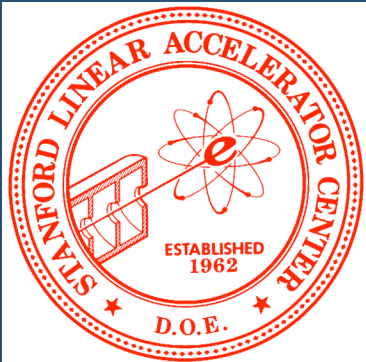
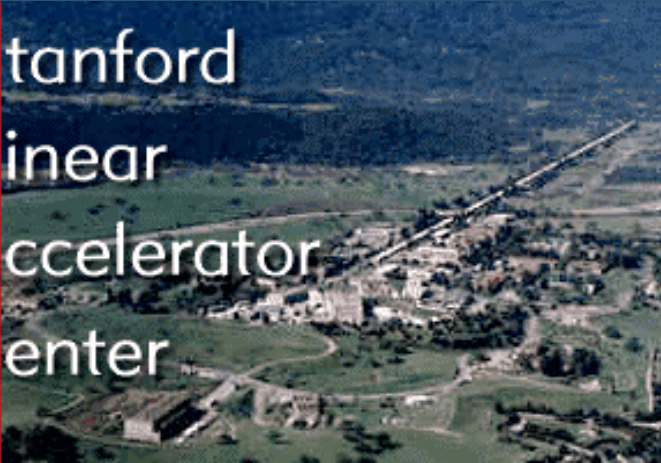


Stanford
Linear
Accelerator
Center



Faster Navigation in Voxel Geometries (DICOM)

Joseph Perl (SLAC/SCCS)

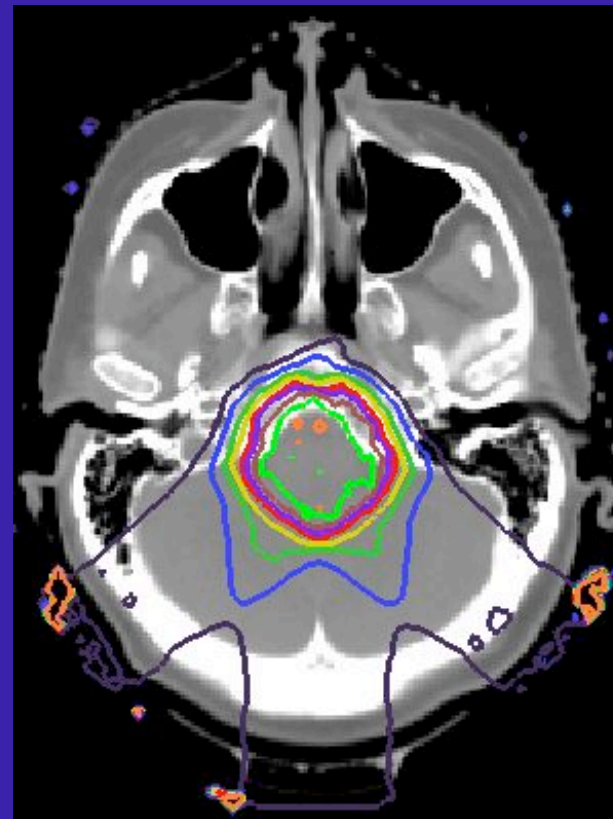
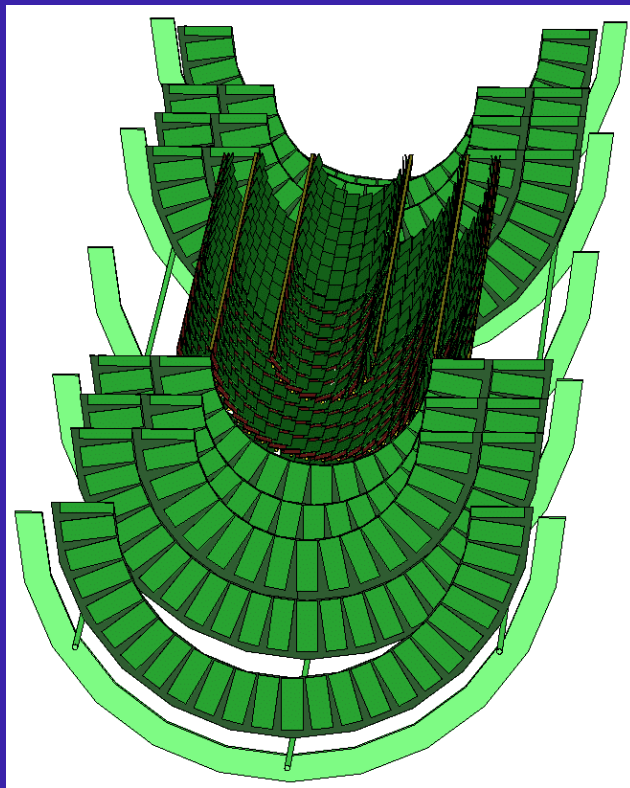
G4NAMU meeting @ AAPM Houston

27 July 2008

Geant 4

Technical Challenge #2 for Geant4 Med Apps: Patient Geometry

- HEP: constructive solid geometry
- Med Phys: constructive solid geometry + imported voxel data (DICOM)
 - different strategies for geometry and navigators



Background: Geant4 “Smart Voxels”

- The word “Voxel” can have two entirely different meanings when Geant4 is applied to Medical Physics.
 - medical physicist thinks of the voxel as a unit of a patient or phantom geometry
 - Geant4 authors use the term to refer to a way of arbitrarily subdividing the entire geometric world for navigation speed up
- Long before any application of Geant4 to DICOM, there was a need to find a way to limit how many physical volumes the navigator needs to check when tracking from one volume to another
- A technique called “smart voxels” was devised by Gabriele Cosmo
 - After the geometry has been read in, but before any particles are tracked
 - Geometry is divided into a set of “smart voxels” by a heuristic that repeatedly subdivides the overall geometry until it has no more than a few physical volumes in any one “voxel”
 - In this way, Geant4 can have millions of physical volumes in any arrangement (does not need to be regular like a DICOM) but the navigator will never need to check more than a few volumes to find the destination volume for a track step
 - The Smart Voxel feature should be turned off only in special cases where optimization is not possible, such as where motion is involved, and even then all but the moving parts should still have optimization left on

Past and Present Solutions

- Before Geant4 version 8.2 (Dec 2006), Geant4 provided:
 - Smart voxels
 - Fast navigation but huge amount of memory
 - even if you used 3D parameterization for the volumes, the “smart voxels” generated after initialization would consume huge memory
 - 3D parameterized volumes without smart voxels
 - Compact memory but very very slow navigation for typical DICOM (oversimplification, see Jiang/Paganetti paper for full discussion of this issue)
- Needed a solution to get compact memory and fast navigation
- Two efforts by medical physicists solved this first, taking advantage of the open nature of Geant4 to recode their own solutions
 - Nearest Neighbor Navigation from MGH
 - OCTREE from Université Laval
- The Geant4 collaboration has since produced two solutions
 - G4NestedParameterization
 - G4RegularNavigation

Nearest Neighbor Navigation from MGH

- Hongyu Jiang, 2004 for release 4.5.0
 - Adapted to release 4.8.1 by Christina Zacharatou Jarlskog
 - Adapted to release 4.9.0 by Cindy Hancox
- Adaptation of GEANT4 to Monte Carlo dose calculations based on CT data
H. Jiang and H. Paganetti
Medical Physics -- October 2004 -- Volume 31, Issue 10, pp. 2811-2818
- Recoding some core Geant4 classes to allow fast navigation among DICOM voxels without the memory penalty of the more generic Geant4 “smart voxels”

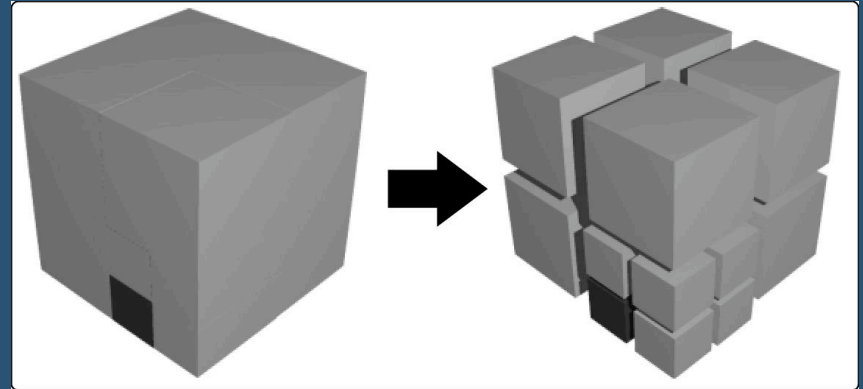
OCTREE from Université Laval

- Vincent Hubert-Tremblay, 2005

- Octree indexing of DICOM images for voxel number reduction and improvement of Monte Carlo simulation computing efficiency

[Vincent Hubert-Tremblay](#) and [Louis Archambault](#)

Medical Physics -- August 2006 -- Volume 33, Issue 8, pp. 2819-2831



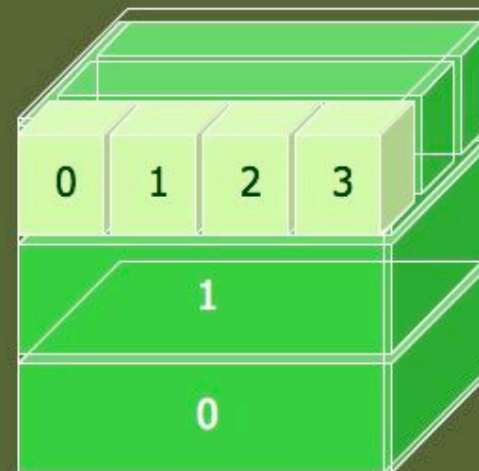
- “...a compression algorithm based on octrees: in homogeneous regions the algorithm replaces groups of voxels with a smaller number of larger voxels”
- “...ability to conserve the high-density gradient area in the volume. The octree density distribution does not contain erroneous regions at interfaces of heterogeneities. This is not the case if one only scales the original DICOM images.”
 - can be applied in any Monte Carlo
 - essentially simplifying the input to the Monte Carlo

G4NestedParameterization

- In Geant4 since release 8.2
 - Example in `geant4/examples/extended/RunAndEvent/RE02`
- Smart Voxel optimization understands how to optimize this parameterization without resulting in huge memory use

Nested parameterization

- ▶ Given geometry is defined as two sequential replicas and then one-dimensional parameterization,
 - ▶ Material of a voxel must be parameterized not only by the copy number of the voxel, but also by the copy numbers of ancestors.
 - ▶ Material is indexed by three indices.



G4RegularNavigation

- In Geant4 since release 9.1
 - Example in `geant4/examples/extended/medical/DICOM`
- From Pedro Arce of CIEMAT, Madrid, along with Gabriele Cosmo and John Apostolakis from CERN
- “Regular” not in the sense that it is the usual navigation, but in the sense that it is customized for regularly-repeating geometries
- Optionally “skip” feature:
 - ignore voxel boundaries where two voxels have the same material
- At least as fast as `G4NestedParameterization`
 - Faster if “skip” feature is enabled in a fairly homogenous phantom
- But there is an issue in how the scoring assigns dose if a volume is skipped, so wait for next release to use this (Dec 08) or contact Pedro for an update

Comparisons

- All four solutions involve restricting the number of voxels that need to be checked and/or skipping voxel boundaries where two voxels have identical material
- All have shown two or three orders of magnitude speed up compared to the un-optimized solution
- Speed comparisons are tricky, results depend on:
 - incident particle type
 - range cuts
 - size and homogeneity of the phantom
- A rigorous comparison of all four methods needs to be done
- But don't do DICOM without using one of these solutions

More hints for large numbers of voxels

Suggestions from Makoto Asai.

There is no silver bullet. You can try some/all of these options combined.

- Material map
 - Though number of materials may be small, each voxel must at least have a pointer to a material, hence huge map of these pointers.
 - Split voxel geometry into reasonable number of regions, assign a dedicated stack to each region. For example $5*5*5 = 125$ regions.
 - Load material map (from file on disk) only for one region. If a track reaches to the boundary of the region you are currently simulating, suspend the track.
 - Simulate all the tracks in one region. Once a region becomes empty, load material map for another region and simulate all tracks in that region.
 - Note that some tracks may come back to a region you have already simulated.
- Event biasing
 - In particular, geometrical importance biasing and secondary particle splitting.
 - You must validate results of your biasing options with full simulation.
- Shower parameterization
 - Instead of having a full EM shower, you may want to consider the shower parameterization in particular for the core part of the shower.
- Parallelization
 - Allocate good number of CPUs...

General Hints for Speeding Up Geant4

- We are trying to improve the speed of Geant4
 - But, since it is a general-purpose toolkit for which we give the user great control, it is possible for the user to make the simulation unnecessarily slow.
- For general applications
 - Check methods which are invoked frequently:
 - UserSteppingAction()
 - ProcessHits()
 - ComputeTransformation()
 - GetField()
 - etc.
 - In such methods:
 - avoid string manipulation,
 - file access
 - cout
 - unnecessary object instantiation or deletion
 - unnecessary massive polynomial calculations such as `sin()`, `cos()`, `log()`, `exp()`.

General Hints for Speeding Up Geant4

- For relatively complex geometry or high energy applications
 - Kill unnecessary secondary particles as soon as possible
 - You can also “Suspend” particles and decide later whether there is value in continuing to track them - see Makoto’s lecture on Geant4’s multiple stacks
 - Utilize G4Region for regional cut-offs, user limits
 - For geometry, consider replica rather than parameterized volume as much as possible. Also consider nested parameterization.
 - Do not keep too many trajectories.
- For relatively simple geometry or low energy applications
 - Do not store the random number engine status for each event.