Pure Tracking and Geometry in GEANT 4

P. Kent

11 April 1995

1 Introduction

The document describes the analysis work carried out for the 'Geometry' subsystem of GEANT 4, during the first quarter of 1995. The topics discussed include elementary tracking, the detector's geometrical representation and tracking optimisation methods.

2 Tracking Principles

In GEANT 3, particles are moved using small steps, determined by physics processes or by the detector geometry. This has the advantage of minimising the number of steps taken, and is known to be efficient and implementable.

In the early stages of discussion, the prospect of a time driven simulation was discussed. This approach was quickly discarded for the following reasons:

- The time step, effectively the resolution, would have to be smaller that the shortest time-scale on which events occured. This would probably result in a significantly greater amount of computations being performed, resulting in a lower performance. Probabilities and intersection distances would have to be updated, recalculated or re-evaluated at each small step.
- No major advantages were foreseen for physics computations.

However, a significant advantage of the time-driven approach would be a more accurate tracking in magnetic fields, because of the smaller step size. This is an area where the current step driven implementation is less robust owing to conflicts between a particle's real trajectory (helical), it's approximation (small linear steps), and the detector geometry. Possible algorithms for magnetic field tracking are discussed in Section 5

3 Detector Representation

One of the major challenges faced by the simulation is to store, in some form of database, a description of the detector geometry. The database has several conflicting requirements:

- To store or provide access to elements of the detector.

- To consume a minimal amount of memory. For example, symmetry or otherwise duplicated volumes[1] of parts of the detector can be stored.
- To have an efficient search/lookup mechanism for the tracking, particularly for when querying areas of the detector close to a known position.

  This is the most common case at tracking time in GEANT 3 - computations are performed on parts of the detector immediately surrounding a particle's position, although particles can traverse the full detector before intersecting a volume.
- Continuing from the previous point, the database must be tunable, either manually or automatically to improve the above search. For instance, although GEANT 3's virtual divisions are not stored in the detector description, the database and its contents must have sufficient functionality to support it.

  A further example from GEANT 3 is the user's[2] ability to add dummy bounding volumes to the hierarchical detector description.
- GEANT 4 must be capable of receiving detector geometries from GEANT 3 and from CAD systems.

Numerous services are required of the 'database' - in particular, transformations between different coordination systems in the case of a hierarchically structured database. It is also necessary to ascribe attributes such as materials and sensitivity to volumes, in addition to their pure geometric description.

3.1 Hierarchical Structure

A hierarchical approach is adopted by GEANT 3 for the description the detector geometry. This is illustrated in figure 1.

Users develop a detector description by positioning *daughter* volumes inside *mother* volumes. Except where boolean operations are performed, the volume boundaries are strictly non-intersecting.

A coordinate system is defined relative to the center of each mother volume, with the exception of the two poly shapes. Positioning a volume inside a mother requires the specification of a translation and a rotation matrix relative to the current mother volume. Hence, determining the coordinates of a point relative to the centre of a given volume requires the application of possibly many rotations and translations.

A well known advantage of the hierarchical structure is that of its tracking time performance - a search for the next volume volume intersected on a specific trajectory can be simply limited to a single mother and its daughters. Judicious use of dummy volumes can improve the search, even when there is no obvious physical hierarchy. A hierarchical structure with local coordinate systems simplifies user positioning of volumes relative to others.

A disadvantage arises when transforming from a global or 'master' reference system to a given local reference system. Many rotation matrices and translations may need to be performed in series. Most commonly, at tracking time,

[1] Volume - a geometrical entity positioned in space. ie. It has at least an ascribed shape and position.

[2] User - Someone who uses/programs GEANT to describe a detector.

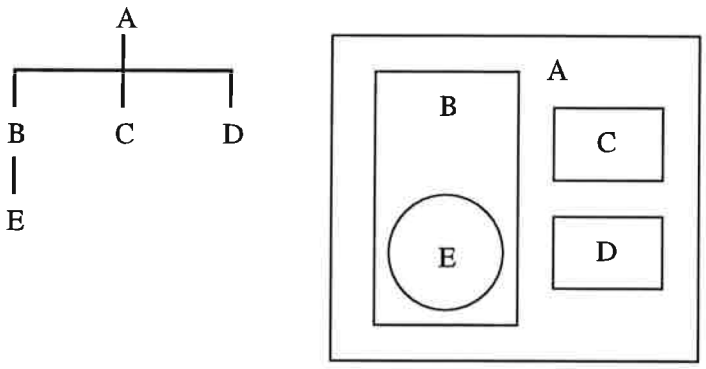

Figure 1: A Hierarchical Detector Geometry

transformations are only required between a coordinate system either one level up (mother of current volume) or one level down (to a daughter of the current volume). This minimises the performance impact. Global magnetic or electric field maps require the global coordinates of a particle to be known, so the complete transformation can still be necessary.

3.2 Flat Structure

A flat geometry is one where there is no user or machine generated hierarchy. Additionally, the geometry is entirely unambiguous - if a particle can only ever be inside one entry of the database at a time, whereas with a hierarchy, a particle may be within several. Flat geometries are effectively hierarchical geometries with only one 'level' - figure 2 illustrates this. The hierarchical geometry is easier to model, because the need to make a hole in the outer volume in removed.

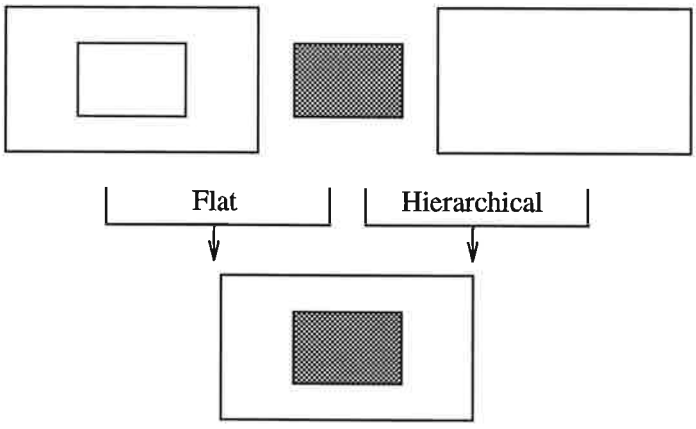

Figure 2: Differences between Flat and Hierarchical Geometries. Voids or Holes must be created inside volumes in the hierarchical case.

A completely flat structure has both advantages and disadvantages over a

hierarchical structure:

- CAD geometries are flat, simplifying import and export to the simulation.
- At most, only one coordinate transformation is required between the global coordinate system and a part of the detector.
- Dummy volumes cannot be as used improve tracking as naturally as in a hierarchical structure. By definition the dummy volumes introduce a hierarchy, and so require an additional special representation.
- A flat geometry does not automatically simplify the tracking time search, and cannot be improved without the use of dummy volumes. Many optimisations are possible (see Section 4), but these may have to manage numbers of entities several orders of magnitude greater than the hierarchical case.
- There are fewer possibilities for tracking optimisation, and those that remain are less effective than for a hierarchical geometry (see Section 4).

3.3 GEANT 3 and CAD Geometries

A requirement of GEANT 4 is a bidirectional interface to CAD systems, utilising the STEP standard. A conversion tool or import facility for GEANT 3 geometries is also planned.

A hierarchical geometry with good performance, even if the geometry is only on one level, would allow both geometries to be supported. The performance of flat geometries would be highly dependent on any optimisation techniques used for tracking.

Sub-sections of the detector must be also exchangeable with CAD systems. This minimises the file size and memory requirements for CAD, speeds up the process, and corresponds with typical working practices - typically an engineer will be working on only a small subset of the entire detector. It should be possible to not export dummy volumes added for optimisation. This implies some form of tagging mechanism.

On import, or re-import of all or part of a detector from a CAD system, any hierarchy will have been lost, and the entities will probably be boundary representations (BREPs). This creates two problems whose potential solutions are discussed in the remainder of this section:

- A single-leveled CAD hierarchy will be less performant than a properly structured hierarchy.
- The entities, or primitives, used is the description will be boundary representations. Although tracking is possible in BREPs, it is significantly slower than in GEANT 3's solid primitives.

3.3.1 Hierarchy Generation

As explained previously, hierarchical geometries are more performant than flat geometries because of the help they give to the intersection search at tracking time. Consequently the amount of non-structured geometry in a detector must be kept to a minimum. Several possibilities were examined for managing this:

Do Nothing Tracking remains possible in a single-leveled geometry. Although it will be slower than with a structured geometry, how much slower would depend on the detector. If only a small part of the detector is unstructured, the performance penalty will be slight if particles spend only a small fraction of their lifetime in those areas. If any tracking optimisations work well in those areas, then the penalty will be smaller still.

While not in production, this penalty may be acceptable.

Manual Generation As a contrast to the previous option, the geometry could be structured by hand. This requires a good UI, but uses the same functionality as is required to input and structure the geometry by hand initially, so the functionality will always be present.

It may be possible to check for illegal overlaps between entities, by making use of the calculated bounding volumes that are required by most tracking optimisation schemes.

Automatic Generation Several algorithms exist for automatically generating a hierarchy, principally from the field of ray tracing. Although not as performant as a human created structure, they represent an improvement over an unstructured geometry.

Their significant disadvantage is that the structure created would not follow logical or functional parts of the detector, and consequently would not be humanly maintainable.

Out of the three options, the first two appear most realistic. Neither of the first two options require adding any functionality to the simulation beyond what what already predicted. If in the future, CAD exchange became significant, the third option could be investigated. Modular design should allow this option to be added at a later date, without significant modifications.

3.3.2 Conversion of Imported BREPs

Typical current CAD systems manipulate and exchange boundary representations. Tracking is less efficient in BREPs than in constructive solid geometry (CSG) based entities such as GEANT 3's solid primitives.

Conversion from BREP to CSG in all but the simplest cases is generally regarded as impossible, because of the many different ways a given BREP can be made from CSG.

Several conversion methods were considered:

Automatic Conversion could be performed on very simple cases - essentially hexahedral BREPs only. These are a common case in detectors, which works both for and against this method: conversion of a significant percentage of the BREPs would greatly improve tracking performance, but it is unlikely that any automatic method could efficiently identify and reuse rotation matrices, translation and transformation operators resulting in significant memory use. The CSG representation would use less memory than the BREP, but would probably introduce many more rotation matrices than necessary.

Manual Conversion is possible because the a user knows what an entity represents, and can put the necessary CSG primitives together in an optimised way.

Assisted Conversion Although automatic conversion is not practical, it would be possible to suggest to the user that a given entity 'looks like a BOX', and offer to do the conversion. This would speed up the manual conversion, while allowing the user to check each conversion.

If a particular entity was exchanged regularly, it would be possible to add a recognition function, and so speed up the conversion of irregular entities.

Of the three options presented above, only manual conversion looks practicable in the short term. Assisted conversion could be added at a later date. An interface should be specified for this and for user defined conversions when STEP I/O is designed and implemented.

4 Tracking Optimisation

Tracking optimisations, as discussed here, as those that reduce the number and/or cost of the 'next intersection' search at tracking time.

Pure tracking, that is to say tracking of geantinos theoretically only requires one service from the geometry sub-system - *"Which volume is next intersected along a given direction from a given point?"*. Although the current implementation frequently has to ask *"where am I?"*, this should be minimised as much as possible in future by using logical techniques.

Clearly, the structure of the geometry in the 'database' can be of significant help, particularly if a well structured hierarchical description is used. Tracking optimisations aim to further reduce the number of candidate volumes for which potential intersections are computed, typically by further dividing the detector volumes.

From experience with GEANT 3.21, it is known that these techniques can at least double the tracking performance of the simulation. However, even the technique of virtual divisions used in 3.21 is not suited to unstructured geometries, such as those GEANT 4 will import from CAD systems, unless the user take some action. One method for improving performance in this case is to collect volumes with similar symmetries (ex. along x,y and z), and overlap them using the MANY option. This method requires some skill on behalf of the user, is less maintainable, and takes time to implement.

The remainder of this section discusses four ways of optimising tracking in GEANT 4.

4.1 Virtual Divisions

Virtual divisions were implemented in GEANT 3.21, and reduce the number of intersection computations required at tracking time.

The virtual division technique both reduces the number of intersection calculations, and ensures that the search is localised around the particle. Figure 3 illustrates the method.

Each mother volume is sliced into a series of strips, along one axis. The strip width is determined by a heuristic based on the extent of the daughters

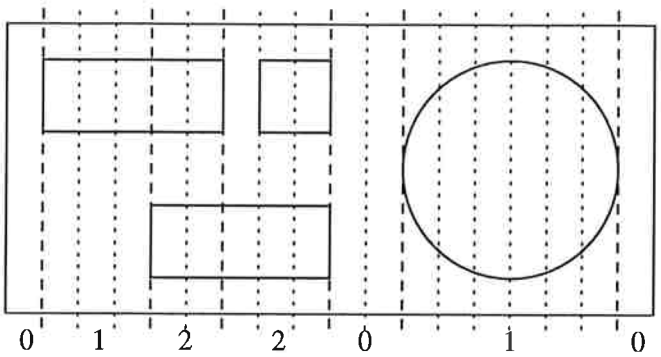

Figure 3: Virtual Divisions. The dashed lines represent divisions that remain after collecting adjacent divisions with the same contents.

along the given axis. The supported axes are the three Cartesian axes and the radial and phi axes from the cylindrical polar coordinate system. Usually, one of the axes is better because of the hierarchical structure of the geometry. For example, a circular array of crystals is best optimised by slicing into phi segments.

Based on the extent of the daughters along the given axis, the contents of each division is stored. The regular divisions enable a fast lookup of candidate volumes for the *"where am I?"* search. A further optimisation is to gather adjacent division with the same contents. Storing both gathered and ungathered slices increases memory usage, but the gathered divisions improve the intersection search.

At tracking time, the division that a particle is in is determined and intersections to its known contents and to its boundaries are computed. If none of its contents are intersected, the next division to be examined is determined, and the search is repeated. If the divisions are along a Cartesian axis the next division can be determined completely logically because of the known ordering of the divisions.

Virtual divisions are most performant only when the number of daughters in each division is small. This is not the case if the axis used for the mother volume does not correspond well with its contents. For example, a barrel array of crystals, positioned off-axis and rotated arbitrarily would not benefit greatly from radial divisions. A user could however position a dummy volume around the barrel, at the correct rotation, and then phi or z axis based division would be significantly more performant. An extension of this example to a global coordinate system with an unstructured, unsymmetrical, flat hierarchy demonstrates two points:

- 1D Virtual divisions are not *automatically* performant in this case. If the user intervenes by either creating a partial hierarchy with dummy volumes, or converts the three-dimensional structure into several one-dimensional, overlapping structures, then the performance is recovered.

- Division along the radial or phi axes is only beneficial if they have the correct origin - such as the origin of a circular array. This will rarely be

the case in an unstructured hierarchy.

One dimensional virtual divisions have the additional advantage of being entirely setup at initialisation time and are very cheap to step through at tracking time because of their known ordering.

4.2 Grid Based Methods

One of the main reasons for the success of the virtual division technique is its ability to effectively localise the search for intersected volumes around a particle. Another method for localising the search would be to effectively construct a 3D map of the entire detector. This could be accomplished by placing a grid over the detector, and storing the volumes inside each cell (figure 4).

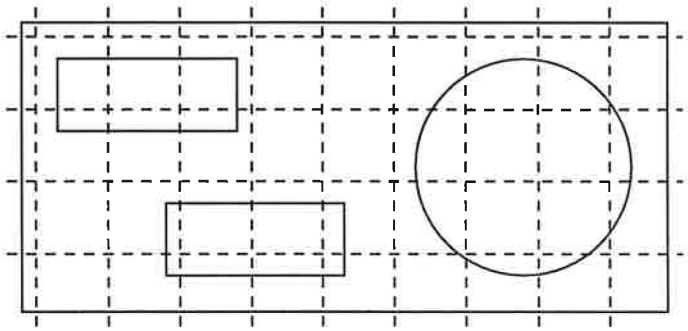

Figure 4: A Fixed Grid

A grid with a fixed, regular, spacing or one with variable spacings would enable fast determination of the cell a particle is in, and hence localise the tracking time search. Determining the contents of each cell could be done entirely at initialisation time, based on the bounding box of each volume.

4.2.1 Fixed Grids

Fixed grids have the advantage of simplicity over variable grids. Determining within which cell a particle is inside is trivial, as is the generation of the contents of each cell.

Their effectiveness would not depend on user optimisations, or on a well structured geometry - performance would be identical for either many-leveled or single-leveled hierarchies. Although advantageous for CAD imported geometries, the technique does not take advantage of hierarchies when they are present.

The main problem with the technique is that of memory consumption versus grid granularity. If the grid is not fine enough, too many volumes will be present inside each cell, and the tracking performance will be poor. Alternatively, if the grid is too fine, the memory requirements will be too great. For example, a small detector that is $10 \times 4 \times 4$ metres contains 16000 $0.10 \times 0.10 \times 0.10$ metre cubes. The CMS detector would contain more than one million similarly sized cubes. Consequently, the storage of the grid and its contents could easily consume more than 10 MB.

To reduce the memory requirements, cells with common contents could make use of the same contents list. For this to occur frequently, the grid's granularity has to match that of the detector's smaller components.

An additional disadvantage is that unlike virtual divisions, the grid's cells cannot be gathered together, to facilitate large steps - the gathering in 3D is dependent on the direction in which the particle is traveling. Intersection with the boundaries of each cell remains computationally inexpensive, but every cell intersected by the particle requires checking. Fortunately HEP detectors are usually very densely packed, and so provided the grid size was suitable, on average, few cells would be tested. Figure 5 illustrates this.

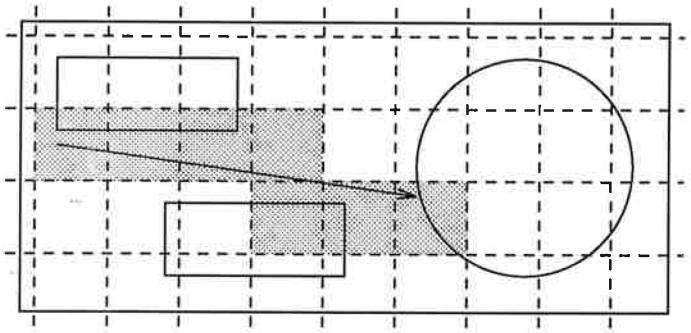

Figure 5: Intersection of Cells. The contents of cells intersected along a given trajectory must be tested for intersection. A grid-based approach does not allow 'gathering' of cells and the intersection's with cell boundaries cannot be avoided,unlike with virtual divisions.

Volumes are more likely to be in many cells than virtual division. To avoid retesting the same volume, a record of each tested volume could be maintained during the search. This would favour using a fine grid, to minimise the number of volumes in each cell, but would be too costly in terms of memory use.

4.2.2 Variable Grids

A possible way of overcoming the conflict between memory use and grid granularity would be to have a variable sized grid, varying across the width, length and breadth of the detector.

This technique would reduce memory requirements by taking account of the granularity of the detector along each Cartesian axis. This method could still not fully adjust to the varied size of detector elements, because detectors are most often radial, which does not correspond well with the Cartesian axes.

A variable sized grid complicates the intersection tests, and also complicates determining within which cell a particle is inside. Although efficient, even if applied at each node, a well-done tree hierarchy will be more efficient and hence faster.

4.3 Voxel Based Methods

To better exploit the varied granularity of detectors, a tree based map could be used in the same manner as the grid-based methods. A tree based map could

be created by recursively dividing the detector into octants, for example, and storing the volumes of the detector inside each end node. Hence the detector would be mapped with a set of varied sized voxels (see figure 6).

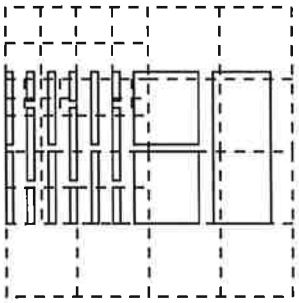

Figure 6: Progressive Refinement of Voxels. Voxels containing more than 3 volumes have been subdivided.

The voxel a particle is in can be determined by descending through the tree structure from the head node. If a particle is known to have been in a given node, the search can be improved by commencing from that node, traversing up the structure to less refined branches until a contained voxel is found, and then traversing back down the structure towards the leaf nodes.

Memory usage would be minimised either by using progressive refinement, subdividing until only a given maximum number of volumes were in each end node, or by subdividing to a great depth and then collecting similar nodes. This second approach would have to make use of subsets to collect nodes with few contents.

Voxel based methods retain the disadvantage of grid based methods in that every voxel intersected along the particle's trajectory must be tested for intersection. Intersection calculations cannot be avoided by logical means alone, as is the case with virtual divisions, although the variable level refinement helps to minimise the number of calculations.

The varied depth of refinement allow memory use also to be minimised, but complicates intersection calculations. A search is less complicated and more efficient than with non-uniform grids, because the search is binary with only a variable depth. A particle, on leaving a very refined node, could enter a node on a branch of the tree originating many levels up. An extreme case would be changing branches descended from the topmost node from a leaf node. This reduces the performance at tracking time because of the more costly lookup procedure - determining adjacent voxels involves traversing the tree.

4.4 Smart Voxels

The smart voxel technique contains features from both virtual division and the voxel based methods. The technique can use hierarchical geometries, and in that case will be as efficient as virtual division.

The method works by, for each mother volume (level in hierarchy), performing one dimensional virtual division, using a heuristic to find the best axis. This is illustrated in figure 7. The divisions are gathered as per the one-dimensional

GEANT 3 case. Each division containing too many volumes is then refined by applying virtual division again, but using a second Cartesian axis. If the resultant sub-divisions contain too many volumes, each subdivision is further refined by dividing again along a third Cartesian axis. Note that the refinement is performed on a per division basis, independently from other divisions.

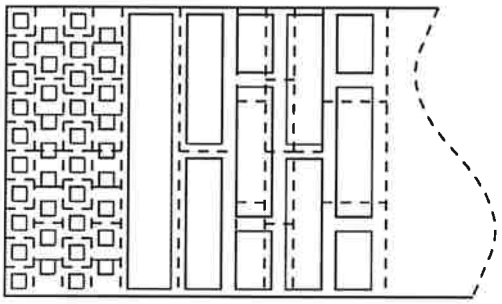

Figure 7: Smart Voxels. In this diagram, x divisions and their respective sets of independent y divisions have been drawn.

Instead of forming a tree structure in memory, the smart voxel technique forms a flat forest in the case of a flat CAD geometry, or in the case of a well structured hierarchical geometry reverts to one dimensional virtual divisions. The technique takes advantage of local coordinate systems if they are present and the refinement compensates where they are not, or where many volumes are defined at one level. This reduces the need to add dummy volumes, or for overlapping many volumes as used in GEANT4, to ensure that symmetries are exploited.

At tracking time, the search proceeds similarly to the virtual division technique. The volume containing the particle is known, and the most refined division containing the particle is found. Intersections to volumes inside this division are computed, and to the bounds of any containing divisions. If the division is at the first level of refinement, equivalent to a virtual division, no boundary intersection computations need be made. If the closest intersection is to one of the boundaries, the relevant adjacent division is examined. The adjacent division can be determined entirely logically, based on the intersected boundary and/or the components of the particle's momentum.

The smart voxels can be computed entirely at initialisation time and share, in common with virtual division, low memory requirements and computationally cheap intersections with the divisions. Additional functionality is required to find the extent of primitives in a restricted coordinate space - for example, the y range where x is between 15 and 16, and z is unrestricted.

5 Magnetic Field Tracking

Tracking in a magnetic field is intrinsically more difficult for simulations than for the uncharged or no magnetic field case. Particles in the field have a helical path, and so only a careful use must be made of the linear path tracking algorithms, so that the tracking both correct and accurate. Several different algorithms have

been used within the lifetime of GEANT 3, each placing different emphasis on performance and safety/accuracy.

This section discusses the algorithms the have or could be used for a performant and accurate tracking in magnetic fields. None of the methods discussed solve the problem of varying magnetic fields. In all cases the step size must be limited to ensure enough resolution is obtained.

5.1 Safety Based Tracking

The magnetic field tracking in GEANT 3.15 was inherently safe, and made no use of the linear intersection routines.

A tracking time, the isotropic distances to candidate volumes are computed, and the particle moved the minimum safety distance along it's true helical path. Intersection is checked only by detecting that the volume containing the particle has changed.

The performance of this tracking is severely limited by the fact that particles can travel close to a volumes, but not actually intersect. Consequently, the step size is limited when large steps could be performed. For example, particle's with high energy may travel along an almost linear path (see figure 8). Additionally the isotropic distances, or 'safety' distances as implemented in GEANT, are usually under-approximations, further reducing the tracking efficiency.

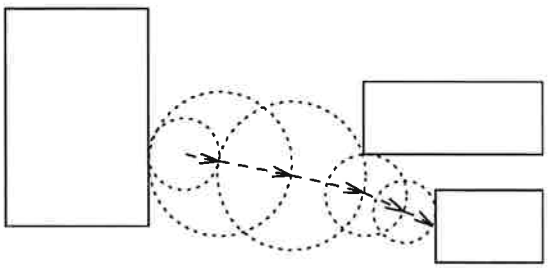

Figure 8: Magnetic Field Tracking based on Isotropic Safety. In this example, the particle has still to intersect the target volume after 5 steps.

5.2 Approximate Tracking

GEANT 3.21 uses a faster magnetic field tracking algorithm that makes use the linear intersection routines. Several parameters can be varied to favour either the tracking accuracy or tracking speed. This algorithm was added to GEANT3.15 to speed up the previous 'safety based' tracking, and is also used in the VENUS simulation code at TRISTAN.

At each tracking step, linear intersections are computed, based on the tangent to the particle's path. The minimum intersection distance is then steped along the particle's helical path. This is inherently unsafe, but the degree of safety can be varied by setting the maximum deflection angle through which the particle can deviate from the linear path. If the step along the helical path ends in another volume, the step distance is halved repeatedly, until the step is found to end within the same volume as the particle.

With this scheme, it is possible for volumes to be missed (figure 9), and many steps are required before the current volume is changed. The step distance must reduce to a small value before linear steps are made, followed by an attempt to 'push' the particle into a new volume is performed.

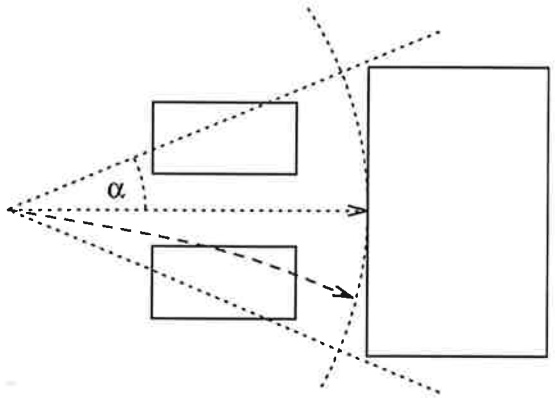

Figure 9: Current Magnetic Field Tracking. The helical step is limited by the linear intersection distance and deviation angle α. This is unsafe, and can result in intersections being skipped.

5.3 Back Checking

This approximate method is used by FLUKA, and relies on two enquiries to the geometry and a user parameter for safety.

As per the approximate method outlined in the previous section, a linear intersection is computed, and used to step along the particle's helical path. If this is inside a new volume, the step length is reduced.

As a further check, an intersection test is made from the proposed endpoint back through the particle's starting point. If this intersects a volume, then then step length is cut.

Although unsafe, this algorithm become increasingly accurate as the particle's maximum deflection angle is reduced, but performance will always be less than simpler algorithms because of the two sets of intersection calculations.

5.4 Exact Helical Tracking

Exact magnetic tracking can be performed using a similar algorithm to the non-magnetic case by computing intersections to volumes along helices. The has the advantage of safety and accuracy over most approximate methods, and ensures that multiple steps are not required to change volumes. Helical tracking is employed in GISMO's surface based geometry, the DELPHI simulation, and was used in old versions of GEANT.

Although theoretically sound, three main implementation difficulties arise:

- The intersection of a helix with many surfaces can only be determined numerically, implying a form of iterative root calculation. This has significant impacts on performance.

- Helical tracking conflicts with multiple-scattering, the data for which has been determined experimentally, and is provided only for linear steps.
- A varying magnetic field cannot easily be included in the helix calculations.

5.5 Cylindrical Safety Based Tracking

This tracking algorithm is an extension of the current (3.21) tracking algorithm to ensure that it is safe, and remove or reduce the need for user setable parameters.

The current algorithm (see Section 5.2) is unsafe because it can miss volume intersections. To ensure that volumes are not skipped, volumes intersecting a cylinder along a linear path tangent to its helical path are examined. The step size is determined by the smallest of the linear intersection distances to the volumes, and the isotropic safety. Consequently, volumes can never be skipped (figure 10), and by restricting candidate volumes to those in the cylinder, the step will not be limited when traveling parallel to volumes as per safety based tracking (section 5.1.

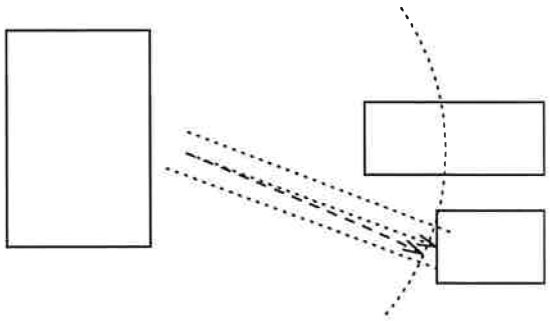

Figure 10: Magnetic Field Tracking using a Cylindical Safety. The step size is limited only by the linear intersection distances and safety distances of volumes intersecting the cylinder.

The radius of the cylinder can be determined based on the particle's momentum, the magnitude of the magnetic field and a user parameter. The user parameter can be used to tune the performance of the tracking only - if the radius is either too small or too large, at no time will the correctness of the tracking be affected, only its performance.

The difficulty in implementing a cylindrical safety lies is generating the candidate volume intersection list. Their intersection with a cone must be computed, and the list must be generated taking the smart voxel technique into account (figure 11). In effect, the geometry in inquired twice. This may have an unacceptable performance impact.

If these difficulties can be solved, the technique is promising because of its inherent safety, ability to adapt with different field strengths and particle momenta, and the possibility safely factoring-in multiple-scattering.

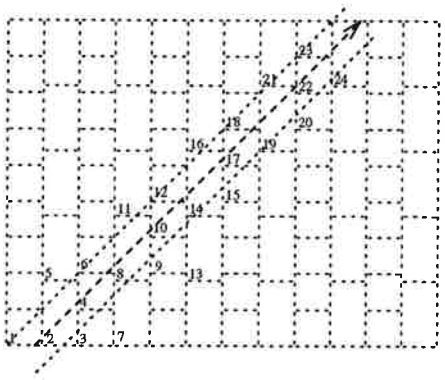

Figure 11: Generation of Candidate Volumes with Cylindrical Safety. Voxels intersected along the length of the cylinder must be checked in the order of intersection.

5.6 Improved Approximate Tracking

This algorithm is improves on the current algorithm of GEANT 3.21 by reducing the possibility of skipping volumes.

The closest linear intersection is computed, given the tangent to the particles path. Isotropic safety is also calculated. A Runga-Kutta based algorithm used to calculate the particle's potential end point for the step. The step is shortened by any of the following, potentially during the Runga-Kutta process for efficiency:

- The magnetic field gradient is too great.
- The opening angle between the target point and the tangent to the particle's initial trajectory becomes greater than a defined maximum value. In this becomes true, the step is cut such that the opening angle equals the maximum.

  The maximum value can come from a systemwide default, and have local overrides. This enables the angle to be tuned where the magnetic field changes rapidly.
- The distance (lateral displacement) between the target end point and the tangent to the particle's initial trajectory exceeds the isotropic safety. If this becomes true, the step is cut such that the distance equals the isotropic safety.

The step limitation based on lateral displacement helps ensure that volumes are not skipped, while still permitting large steps for tracks with a small deviation from linear.

6 Conclusions

The different geometrical requirements for GEANT 4 requires increased functionality to GEANT 3. In particularly, the geometry subsystem must have good performance with unstructured geometries.

A hierarchically structured geometry is known from experience to be performant at tracking time, because of the reduction in candidate volumes for intersections. A hierarchical geometry can also be made very compact.

The virtual division technique of GEANT 3 is not sufficient for completely flat or partially structured geometries, unless users actively create a hierarchy or overlap several one-dimensional structures when creating the geometry.

An extension of virtual division to three dimensions - 'smart voxels' - appears promising for its computational cheapness and low memory costs, and its ability to reduce to virtual divisions in the case of a hierarchical geometry. It significantly reduces the need to structure or modify flat or poorly structured geometries by hand.