# Visualization and (G)UI Parallel Sessions

Session 1: Monday: 15:30-16:45
10 - /vis/viewer/set/background for selection of background color – John
10 - Adding window location to view parameters - John
10 - Non-uniform scaling - John
10 - <sstream> - Koichi
15 - Boolean Operations still problems for some users + need for a boolean “cut” rather than “section” - John
15 - Enhanced Trajectory drawing - Joseph
10 - Generic cuts on attributes (make Trajectories/Hits invisible and/or culled based on physics atts) – Joseph

Session 2: Tuesday: 14:00-15:15
15 - Web services of Geant4 - Hajime
05 - DAWN web service - Joseph
05 - Integrated visualization of field-lines (electric and magnetic) - Joseph
10 - Simplified switching among multiple visualization drivers at event-level - Joseph
10 - Visualization for new Scorers - Joseph
05 - Issues around static objects CLHEP Transform3D::Identity and G4VisAttributes::Invisible when using DLLs on Windows - Guy
15 - Python interfaces - Koichi
05 - Some Future Considerations for Python Interface - Hajime
05 - Visualization for Parallel Worlds - Tsukasa

# /vis/viewer/set/background for selection of background color

John Allison introduced the topic

- Background
  - –/vis/viewer/set/background green
  - /vis/viewer/set/background 1 .5 .5
  - /vis/viewer/refresh (if not auto-refresh)
  - Implemented in SetView() of OpenGL, RayTracer[X]
  - Add 4th parameter (opacity)?

- All thought this was OK
- OpenInventor also has it
- Color Map generally useful (in graphics reps, could use its methods in detector construction for example).
- Color Map could also include wavelengths?
- Opacity as 4th param, OK
- Eventually have in HepRep too

# Adding window location to view parameters

John Allison introduced the topic

- Window location

```
/vis/open OGLIX 600+100+100
/vis/viewer/create 800-0+0
```

Implemented in OGL*X, RayTracerX; needed in OGL*Xm, Win32, OI…

- All thought this was OK
- Note that /vis/open above is compound command that includes /vis/viewer/create. That is, the two commands above are two different options
- Eventually have in HepRep too

# Non-uniform scaling

John Allison introduced the topic

- Non-uniform scaling

```
/vis/viewer/scale .1 .1 1
/vis/viewer/scaleTo .1 .1 1
```

  Implemented in OpenGL; needed elsewhere

- SceneHandler's job
- Objects will appear distorted, for example, sphere will become ellipsoid
- Example use case is drawing of accelerator

# From Koichi on <sstream>

## String Stream Shift to "sstream"

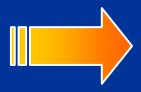
- <*strstream*> header is now obsolete. We should shift from <*strstream*> to <*sstream*>.
  - Shift from *ostrstream / istrsteam* to *ostringstream / istringstream*
  - Currently, warning message is suppressed by *`globals/management/include/G4Types.hh`*
- There are many files affected over categories (*see the next slide*).
  - The migration is expected to be done by each category.

*obsolete*

```
#include <strstream>

char strg[100];
std::ostrstream os(strg,100);
os << "hoge hoge" << '\0';
G4String astr= strg;

char* astring= "1 2 3";
std::istrstream is(astring);
is >> vx >> vy >> vz;
```



*new*

```
#include <sstream>

std::ostringstream os;
os << "hoge hoge";
G4String astr= os.str();


char* astring= "1 2 3";
std::istringstream is(astring);
is >> vx >> vy >> vz;
```

# From Koichi on <sstream>
## Rough Estimation of Impact

| Category | # of affected codes | notes |
|---|---|---|
| Run | 4 | |
| Interfaces | 5 | *done* |
| Persistency | 3 | |
| Particle/management | 2 | |
| Event | 1 | |
| Graphics_rep | 1 | |
| Geometry | 8 | |
| G3tog4 | 2 | |
| Global | - | *done* |
| Visualization | 39 | |
| Process/cut | 0 | *still remain unnecessary #include <strstream>-s* |
| Process/management | 4 | |
| Process/hadronic | 22 | |
| Process/electromagnetic | 11 | |
| Process/transportation | 2 | |
| Parameterisations | 1 | |
| Intercoms | 32 | *done* |
| Tracking | 3 | |

In addition, there still remains unnecessary <strstream> inclusions.

# From Koichi on <sstream>
## How to shift in practice?

1. Grep “strstream” in your whole codes.
   - `> grep –r strstream .`
2. Brush up your codes…
   - Check if your “`#include <strstream>`” statements in .cc/.hh files are really necessary.
     - unnecessary inclusion? (never used, but included!)
     - Inclusion in header files should be moved into .cc files in most cases.
3. Special Notes for implementations of *G4XXXMessenger*
   - *G4UImessenger.hh* will NOT include <strstream>(<sstream>), so that the inclusion is placed directly in your .cc files *if needed*.
4. #include <strstream> is replaced with #include <sstream>.
5. ostrstream / istrstream are shifted to ostringstream / istringstream.

# <sstream>

- Interfaces, Intercoms, Global, done by Koichi already
- Koichi has given us specific recipe for how to fix the code.
- Koichi has tested on Linux.  Tests OK.
- Guy has tested on Windows.  Tests OK.
- graphics_reps: 1 file to fix - Turns out John Allison has already done it
- vis: 39 files to fix - Turns out John Allison has already done it (thanks John!)

# Boolean Operations

John Allison introduced the topic

- Trouble if two surfaces are shared between operand and operator or nearly share surface or if have a whole hierarchy of boolean operations.
- Existing code is by Evgueni and requires deep expert knowledge.
- Evgueni is no longer supported for Geant4, and the problems to be solved are non-trivial, so he can't work on them as a brief, spare-time project, as he has helped us on other minor issues.
- John Allison and Evgueni have prepared a detailed proposal, a major re-implementation. Among the ideas, if two surfaces are very close, within a declared tolerance, declare them to be the same surface.
- Proposal includes not only fixing bugs, but implementing some geometries that are currently simply not supported. Will also provide new boolean operation of great use to visualization, the "cutaway".
- Among the users who have been affected by existing problems with boolean operations are LHCb.
- Vis-GIU group endorse the proposal. Question is how to fund the work.

# Boolean Operations

## G4Polyhedron Discussion Document (Draft 26/10/05)

### 1 Background

The original G4Polyhedron was written by Evgeni Tchemaev. At some stage, it became HepPolyhedron, with a view to incorporating in CLHEP; G4Polyhedron became a simple wrapper.

HepPolyhedron defines a three dimensional shape in terms of a set of facets. The facets are triangles or planar quadrilaterals. The vertices of the facets are specified by indices into a vector of vertices. Several specialised sub-classes implement common shapes, such as box and tube, that are used by common Geant4 solids. Note that not all Geant4 shapes are able to provide G4Polyhedron representations at present – see Appendix A. Access methods are provided to retrieve information for the visualisation scene handlers (Appendix B summarises the current interface).

A significant part of HepPolyhedron is the Boolean processor, allowing union (addition), intersection and subtraction. The algorithms for these operations are complex and it is difficult to avoid numerical problems. There are also logical problems well known in solid geometry. Unfortunately, this gives rise to one of the most common problem reports – see User Forum: Visualization threads 5, 51, 79, 87, 142, 168, 232 and Geometry threads 19, 48, 55, 65, 73, 130, 152, 281, 348, 430. Although this is not a geometry problem, if often gets reported there. Some of the earlier problems are fixed, but a residue remains.

### 2 Proposal

The problems of Boolean processing are complex and difficult. The algorithms need a thorough rethink. They were written many years ago. Expertise and familiarity with the code have faded away. It is not simply a matter of refining the algorithms, but of redesigning them using best current practice. In particular, we think that robustness can only be guaranteed by the introduction of the concept of tolerance, much as in Geant4 geometry tracking. We think that the only way forward is a rewrite.

A simple glance at the missing or incorrect representations (see Appendix A) or at the features of the existing code (see Appendix B) will convince you that this is no small job. Based on Evgeni's experience, we estimate that a rewrite will take at least 1 man-year of effort. From that we hope to get robust more efficient and complete code.

# Boolean Operations

- Proposal also includes two useful appendices

Complete document reproduced at end of this presentation

## 3 Appendix A – Geant4 shapes currently without proper polyhedron representation

### *3.1 Returning an incorrect polyhedron*

All BREPS return a simple bounding box by inheritance of G4BREPSolids – except G4BREPSolid, G4BREPSolidPCone and G4BREPSolidPolyhedra, which are correctly implemented.

G4Hype returns a tube

G4Polyhedra when constructed using "generic parameters" gives "HepPolyhedronPgon: error"

G4TwistedBox returns a simple box

G4TwistedTrap returns a simple trapezoid

G4TwistedTrd returns a simple Trd

G4TwistedTubs returns a simple Tubs

### *3.2 Others simply not implemented*

G4ClippablePolygon.cc

G4Polycone (when constructed using "generic parameters")

## 4 Appendix B – Current HepPolyhedron interface

### *4.1 Current specialised sub-classes*

(In cryptic notation)

HepPolyhedronBox (dx,dy,dz)

HepPolyhedronTrd1 (dx1,dx2,dy,dz)

HepPolyhedronTrd2 (dx1,dx2,dy1,dy2,dz)

# Enhanced Trajectories

Joseph introduced the topic

- Already discussed during Visualization Summary talk
- OK with Vis group to proceed

# Generic Cuts on Attributes

Joseph introduced the topic

- Develop a scheme to make trajectories or hits invisible and/or culled based on physics attributes
- Use the G4AttValues that are already set for trajectories or hits (at least in some of the existing examples such as A01 and RE01).

- Enhanced trajectories will be able to have color, line width, etc. changed based on some attributes
- The additional requirement is to make something invisible based on some attribute

- Invisible is not the same as setting a color
  - Objects that are set invisible may later be made visible by some interactive graphics systems (such as HepRep)
  - At that point, they still need a color

- What about culling?
  - In geometry, we find some cases where we want invisible objects culled, other cases where we do not want these culled
  - Same could apply to trajectories
- Future Work Item
- Guy pointed out that this may benefit from a “filter” design

# Web Services of Geant4

Hajime introduced the topic

- See Hajime's other talk today on this topic

# DAWN Web Service

Joseph introduced the topic

- Accept that some users are not going to have DAWN on their own machine
- So one could set up a web based service.
  - You fill out a form to tell it where to find your .prim file
  - You fill out some other parameters that are the equivalent to running the DAWN setup GUI on your local machine
  - You tell it your email address
  - You then hit the submit button and walk away
  - Some time later, you get an email that tells you where to pick up your completed eps file
- Could also have options to process the file through DAWNCUT or DAVID.
- Could even generate more views that you had asked for after the first one (if it had spare CPU capacity)
- Well defined-project in need of an enthusiastic web-services developer
  - Volunteers?

# Integrated visualization of field-lines (electric and magnetic)

Joseph introduced the topic

- Demonstration work was done by Laurent Desorgher
- More general implementation requires separation of kernel and vis driver parts

# Simplified switching among multiple vis drivers at event-level

Joseph introduced the topic

- Current solution is for user to explicitly save and reuse random seed
- Would be better to somehow automate this process
- Watch out for case of accumulating multiple events or multiple runs
- Will need to transfer scene information from one scene handler to another
- Transfer already works for some of the drivers, not for others
- It is a problem of “replaying” the view parameters

# Visualization for new Scorers

Joseph introduced the topic

- Useful new classes from Tsukasa and colleagues
- Vis Group should provide solutions to easily show these hits
- What is to be visualized is a surface
  - plane, surface of sphere, etc., not a 3D field
- For each hit on this surface, just need to represent a single scaler

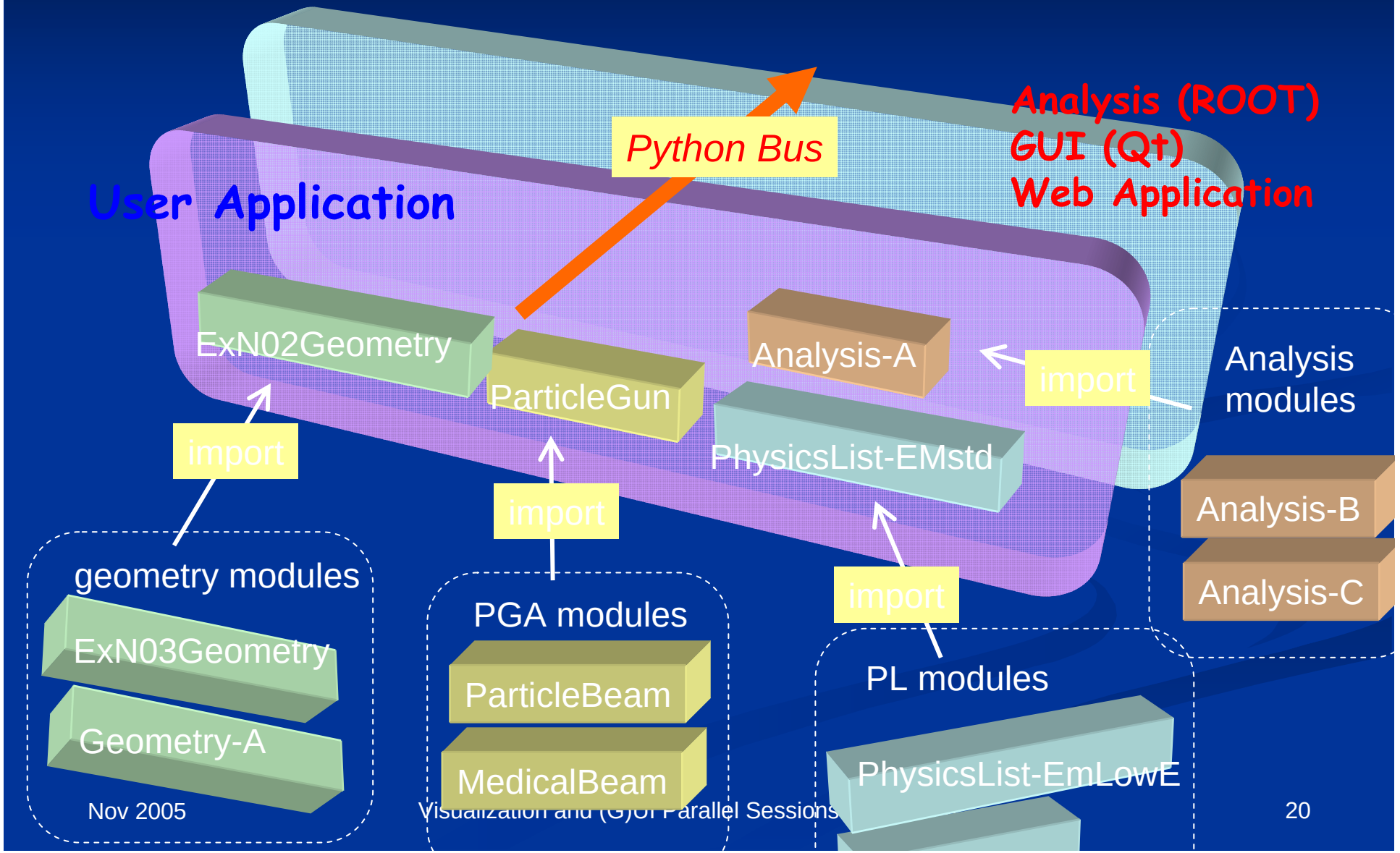
# Issues around static objects

Guy introduced the topic

- static objects cause problems for DLLs on Windows
  - CLHEP Transform3D::Identity
  - G4VisAttributes::Invisible
- John Allison will handle Invisible
  - replacing the static with getInvisible()
- Guy or John will handle Transform3D::Identity
  - Cannot change this in CLHEP itself, but can just instead use the default constructor of Transform3D

# Python Interfaces

See Koichi's talk from this morning's Plenary



# Some Future Considerations for Python Interfaces

Hajime introduced the topic

- Geant4Py gives possibility to touch not only visualization and interfaces, but to touch some part of Geant4 kernel
- A next step might be to expose all of the Vis objects to Python
- A later step might entirely separate vis from intercoms, making all vis communication go through python

# Visualization for Parallel Geometries

Tsukasa introduced the topic

- Just something we must keep in mind as we move forward with the work on parallel geometries
- Vis group will keep an eye on this
- Relationship to new scoring classes

# Appendix: Complete Text of Boolean Operations Proposal

## G4Polyhedron Discussion Document (Draft 26/10/05)

### 1 Background

The original G4Polyhedron was written by Evgeni Tchernaev. At some stage, it became HepPolyhedron, with a view to incorporating in CLHEP; G4Polyhedron became a simple wrapper.

HepPolyhedron defines a three dimensional shape in terms of a set of facets. The facets are triangles or planar quadrilaterals. The vertices of the facets are specified by indices into a vector of vertices. Several specialised sub-classes implement common shapes, such as box and tube, that are used by common Geant4 solids. Note that not all Geant4 shapes are able to provide G4Polyhedron representations at present – see Appendix A. Access methods are provided to retrieve information for the visualisation scene handlers (Appendix B summarises the current interface).

A significant part of HepPolyhedron is the Boolean processor, allowing union (addition), intersection and subtraction. The algorithms for these operations are complex and it is difficult to avoid numerical problems. There are also logical problems well known in solid geometry. Unfortunately, this gives rise to one of the most common problem reports – see User Forum: Visualization threads 5, 51, 79, 87, 142, 168, 232 and Geometry threads 19, 48, 55, 65, 73, 130, 152, 281, 348, 430. Although this is not a geometry problem, if often gets reported there. Some of the earlier problems are fixed, but a residue remains.

### 2 Proposal

The problems of Boolean processing are complex and difficult. The algorithms need a thorough rethink. They were written many years ago. Expertise and familiarity with the code have faded away. It is not simply a matter of refining the algorithms, but of redesigning them using best current practice. In particular, we think that robustness can only be guaranteed by the introduction of the concept of tolerance, much as in Geant4 geometry tracking. We think that the only way forward is a rewrite.

A simple glance at the missing or incorrect representations (see Appendix A) or at the features of the existing code (see Appendix B) will convince you that this is no small job. Based on Evgeni's experience, we estimate that a rewrite will take at least 1 man-year of effort. From that we hope to get robust more efficient and complete code.

The proposal is:

- Reimplement existing functionality, including:
  - Improved algorithms for Boolean processing;
  - Caching of normals (for speed);
- Offer the option of user-supplied normals (for space saving and speed);
- Add a new Boolean operation – cut – that creates open polyhedra for cutaway views;
- Complete the implementation of polyhedron representations of all Geant4 shapes.

John Allison
Evgeni Tchernaev
26th October 2005

# Appendix: Complete Text of Boolean Operations Proposal

## 3 Appendix A – Geant4 shapes currently without proper polyhedron representation

### *3.1 Returning an incorrect polyhedron*

All BREPS return a simple bounding box by inheritance of G4BREPSolids – except G4BREPSolid, G4BREPSolidPCone and G4BREPSolidPolyhedra, which are correctly implemented.

G4Hype returns a tube

G4Polyhedra when constructed using "generic parameters" gives "HepPolyhedronPgon: error"

G4TwistedBox returns a simple box

G4TwistedTrap returns a simple trapezoid

G4TwistedTrd returns a simple Trd

G4TwistedTubs returns a simple Tubs

### *3.2 Others simply not implemented*

G4ClippablePolygon.cc

G4Polycone (when constructed using "generic parameters")

## 4 Appendix B – Current HepPolyhedron interface

### *4.1 Current specialised sub-classes*

(In cryptic notation)

```
HepPolyhedronBox (dx,dy,dz)
HepPolyhedronTrd1 (dx1,dx2,dy,dz)
HepPolyhedronTrd2 (dx1,dx2,dy1,dy2,dz)
HepPolyhedronTrap
 (dz,theta,phi,h1,bl1,tl1,alp1,h2,bl2,tl2,alp2)
HepPolyhedronPara (dx,dy,dz,alpha,theta,phi)
HepPolyhedronTube (rmin,rmax,dz)
HepPolyhedronTubs (rmin,rmax,dz,phi1,dphi)
HepPolyhedronCone (rmin1,rmax1,rmin2,rmax2,dz)
HepPolyhedronCons
 (rmin1,rmax1,rmin2,rmax2,dz,phi1,dphi)
HepPolyhedronPgon
 (phi,dphi,npdv,nz,z(*),rmin(*),rmax(*))
HepPolyhedronPcon (phi,dphi,nz,z(*),rmin(*),rmax(*))
HepPolyhedronSphere (rmin,rmax,phi,dphi,the,dthe)
HepPolyhedronTorus (rmin,rmax,rtor,phi,dphi)
```

# Appendix: Complete Text of Boolean Operations Proposal

```
HepPolyhedronEllipsoid (dx,dy,dz,zcut1,zcut2)

HepPolyhedronEllipticalCone(dx,dy,z,zcut1)
```

### 4.2 Other creation methods

```
int createTwistedTrap(double Dz,
                      const double xy1[][2],
                      const double xy2[][2]);
```

### 4.3 General creation method

```
int createPolyhedron(int Nnodes, int Nfaces,
                     const double xyz[][3],
                     const int faces[][4]);
```

### 4.4 Boolean operations

```
HepPolyhedron add(const HepPolyhedron &p) const;

HepPolyhedron subtract(const HepPolyhedron &p) const;

HepPolyhedron intersect(const HepPolyhedron &p) const;
```

### 4.5 Surface and Volume

```
double GetSurfaceArea() const;

double GetVolume() const;
```

### 4.6 Access methods

```
int GetNoVertices() const { return nvert; }

int GetNoFacets() const { return nface; }

HepPolyhedron & Transform(const HepTransform3D & t);

bool GetNextVertexIndex
 (int & index, int & edgeFlag) const;

HepPoint3D GetVertex(int index) const;

bool GetNextVertex
 (HepPoint3D & vertex, int & edgeFlag) const;

bool GetNextVertex
 (HepPoint3D & vertex, int & edgeFlag,
  HepNormal3D & normal) const;

bool GetNextEdgeIndeces
 (int & i1, int & i2, int & edgeFlag,
  int & iface1, int & iface2) const;

bool GetNextEdgeIndeces
 (int & i1, int & i2, int & edgeFlag) const;

bool GetNextEdge
 (HepPoint3D &p1, HepPoint3D &p2, int &edgeFlag) const;

bool GetNextEdge
 (HepPoint3D &p1, HepPoint3D &p2, int &edgeFlag,
  int &iface1, int &iface2) const;
```

# Appendix: Complete Text of Boolean Operations Proposal

```
void GetFacet
 (int iFace, int &n, int *iNodes,
  int *edgeFlags = 0, int *iFaces = 0) const;

void GetFacet
 (int iFace, int &n, HepPoint3D *nodes,
  int *edgeFlags = 0, HepNormal3D *normals = 0) const;

bool GetNextFacet
 (int &n, HepPoint3D *nodes,
 int *edgeFlags=0, HepNormal3D *normals=0) const;

HepNormal3D GetNormal(int iFace) const;

HepNormal3D GetUnitNormal(int iFace) const;

bool GetNextNormal(HepNormal3D &normal) const;

bool GetNextUnitNormal(HepNormal3D &normal) const;
```