New developments in the Geant4 geometry

Oliver Link, CERN / PH-SFT

Applications Area meeting

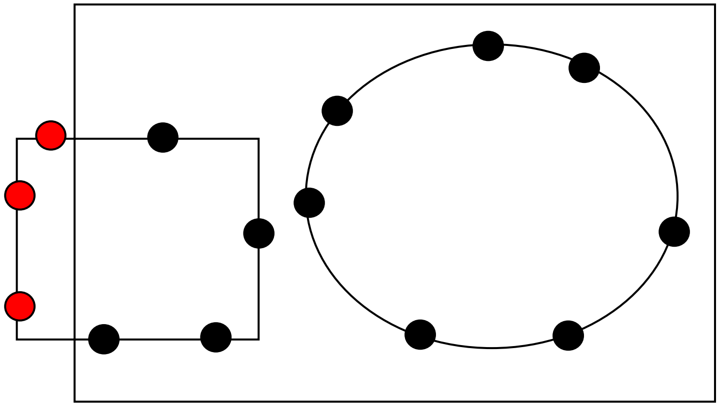
7. December 2005

Development mainly oriented to

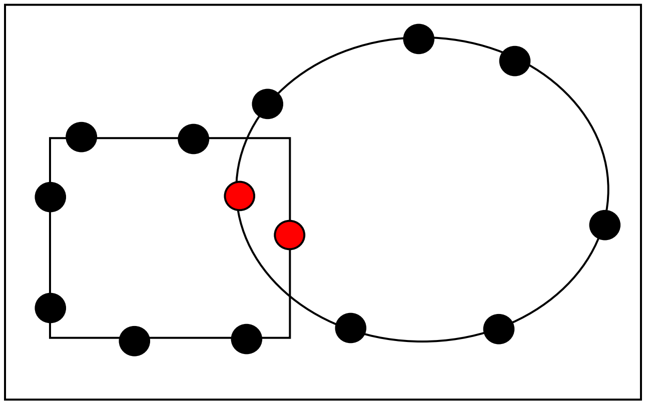
- A new Overlap checking facility
- Added 7 New Solids
- A new polynom root finder (release 7.1):
  - The Jenkins-Traub Algorithm
- Accuracy test for Solids: SurfaceChecker
- Review of normal vectors for CSG Solids

A new Overlap checking facility

- Implementation of `GetPointOnSurface()` method for all CSG and most specific solids.



Overlap with mother



Overlap of daughters

Activation at volume construction for placements and parameterized volumes.

7 new solids

- G4TwistedBox
- G4TwistedTrd
- G4TwistedTrap
- G4TwistedTubs
- G4Ellipsoid
- G4EllipticalCone
- G4Tet

G4Tet and G4Para

G4Tet(Name,p1,p2,p3,p4,flag)

G4Para(Name, fDx, fDy, fDz, α, θ, ϕ)

Written by Marcus H. Mendenhall and Robert A. Weller. (Vanderbilt Univ.)

G4Box and G4TwistedBox

G4Box(Name,fDx,fDy,fDz)

G4TwistedBox(Name,$\Delta\varphi$,fDx,fDy,fDz)

G4Trd and G4TwistedTrd

G4TwistedTrd(Name,fDx_1,fDx_2,fDy_1,fDy_2,fDz,$\Delta\varphi$)

G4Trd(Name,fDx_1,fDx_2,fDy_1,fDy_2,fDz)

G4Trap and G4TwistedTrap

G4TwistedTrap(Name,$\Delta\varphi$,fDz,θ,φ,fDx$_1$,fDx$_2$,fDy$_1$,fDx$_3$,fDx$_4$,fDy$_2$,α)

G4Trap(Name,fDz,θ,φ,fDx$_1$,fDx$_2$,fDy$_1$, α,fDx$_3$,fDx$_4$,fDy$_2$,α)

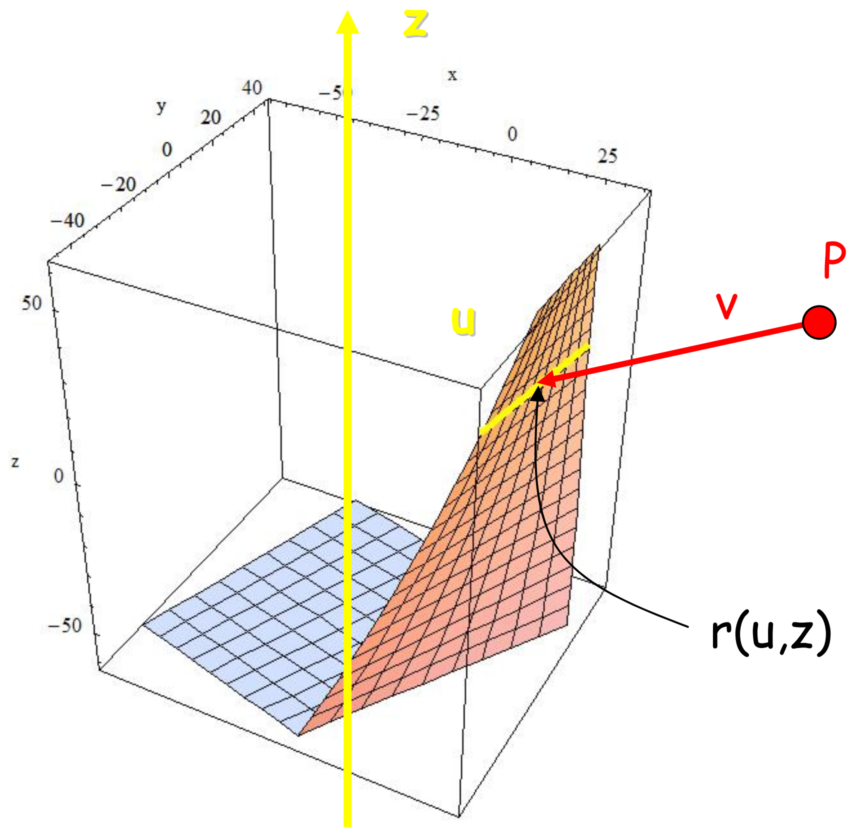
A Twisted Face

Surface Equation

$$\vec{r}(u,z) = \vec{P} + t \cdot \vec{v}$$

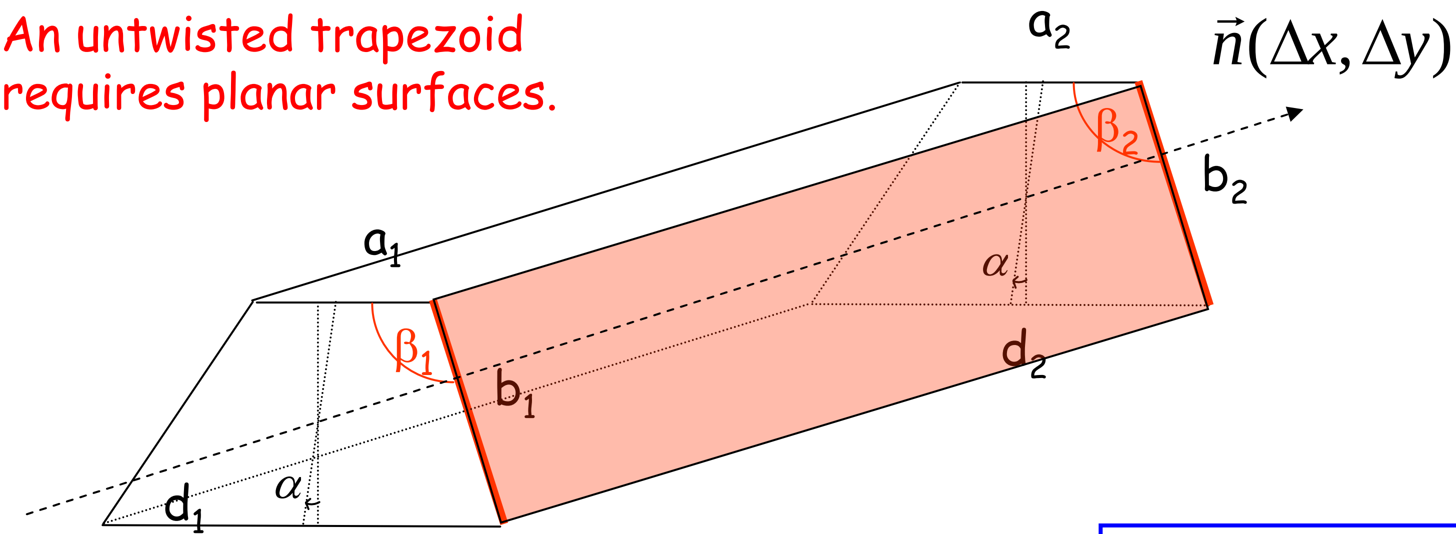
The solution contains trigonometric terms which are approximated with Padé expansions.
Polynom: 7th order.
Solution: Jenkins-Traub

+ Planarity condition



Planarity condition

An untwisted trapezoid requires planar surfaces.



Planarity is equivalent to

$$\beta_1 = \beta_2$$

$$\Leftrightarrow$$

$$\alpha_1 = \alpha_2$$

$$\frac{d_1 - a_1}{b_1} = \frac{d_2 - a_2}{b_2}$$

Polynomial Coefficients

c0 = 7200 b1 dphi (-4 (a1-d1) (pz vy-py vz)
+b1 (8 pz (vx-fTAlph vy)+(a1+a2+d1+d2-8 px+8 fTAlph py) vz))

c1 = -14400 b1 (-2 (a1-d1) (dphi pz vx+L vy-dy vz-dphi px vz)
+b1 (4 L (vx-fTAlph vy)+(a1-a2+d1-d2-4 dx+4 dy fTAlph) vz
-4 dphi (fTAlph pz vx+pz vy-fTAlph px vz-py vz)))

c2 = 600 b1 (a1 (-48 L vx+48 dx vz+dphi (20 pz vy+b1 vz-20 py vz))
+d1 (48 L vx-48 dx vz+dphi (-20 pz vy+b1 vz+20 py vz))
+b1 (96 (-L vy+dy vz)+dphi (-40 pz vx+(a2+d2+40 px) vz)
-8 fTAlph (12 L vx-5 dphi pz vy-12 dx vz+5 dphi py vz)))

c3 = 1200 b1 (-2 (a1-d1) (dphi pz vx+5 L vy-5 dy vz-dphi px vz)
+b1 (20 L (vx-fTAlph vy)+(-a1+a2-d1+d2-20 dx+20 dy fTAlph) vz
-4 dphi (fTAlph pz vx+pz vy-fTAlph px vz-py vz)))

c4 = 12 b1 (d1 (-200 L vx+200 dx vz+dphi (4 pz vy+b1 vz-4 py vz))
+a1 (200 L vx-200 dx vz+dphi (-4 pz vy+b1 vz+4 py vz))
+b1 (400 (L vy-dy vz)+dphi (8 pz vx+(a2+d2-8 px) vz)
+8 fTAlph (50 L vx-dphi pz vy-50 dx vz+dphi py vz)))

c5 = 4 b1 (-4 (a1-d1) (7 dphi pz vx-3 L vy+3 dy vz-7 dphi px vz)
+b1 (-24 L (vx-fTAlph vy)-6 (a1-a2+d1-d2-4 dx+4 dy fTAlph) vz
-56 dphi (fTAlph pz vx+pz vy-fTAlph px vz-py vz)))

c6 = 4 b1 (d1 (-28 L vx+9 dphi pz vy+28 dx vz-9 dphi py vz)
+a1 (28 L vx-9 dphi pz vy-28 dx vz+9 dphi py vz)
+2 b1 (9 dphi pz vx+28 L vy-28 dy vz-9 dphi px vz
+fTAlph (28 L vx-9 dphi pz vy-28 dx vz+9 dphi py vz)))

c7 = -36 b1 (-(a1-d1) (L vy-dy vz)+2 b1 (L (vx-fTAlph vy)-dx vz+dy fTAlph vz))

A New Polynom Root Finder: Jenkins-Traub Algorithm

The Jenkins-Traub Algorithm

- Converges for any distribution of zeros
- Works for any real polynom
- Is fast compared to other algorithms
- Is used by Mathematica
- Available in HEPNumerics G4 module (G4JTPolynomialSolver)

Based on M.A.Jenkins and J.F. Traub. "*A three-stage algorithm for real polynoms using quadratic iteration.*" . SIAM Journal on Numerical Analysis, 7 (545-566)

G4Tubs and G4TwistedTubs

G4Tubs(Name,ri,ra,fDz, ϕ_s,ϕ_d)

G4TwistedTubs(Name, $\Delta\varphi$, ri, ra, fDz, ϕ_d)

Based on „*Stereo Mini-jet Cells in a Cylindrical Drift Chamber*“ (hep-ex/0303014v1, K. Hoshina et al.)

G4Orb and G4Sphere

G4Orb(Name,r)

G4Sphere(Name, ri, ra, ϕ_s, ϕ_d, θ_s, θ_d)

G4Cons and G4EllipticalCone

G4EllipticalCone(Name,px,py,h,zcut)

G4Cons(Name, $r1_{min}$, $r1_{max}$, $r2_{min}$, $r2_{max}$,fDz, ϕ_s, ϕ_d)

D. Anninos (CERN, Cornell Univ.)

G4Hype and G4Ellipsoid

G4Hype(Name,ri,ra,α_i,α_a,fDz)

G4Ellipsoid(Name,px,py,pz,zcut1,zcut2)

G.Horton-Smith (Caltech) ,
G.Guerrieri (INFN Genova),
D. Anninos (CERN, Cornell Univ.)

G4BREPSolidPolyhedra/PCone

G4BREPSolidPolyhedra(Name, ϕ_s, ϕ_d, n_{sides}, n_{planes}, z_{start}, $\underline{z}_{values}$, $\underline{r}_{min}$, $\underline{r}_{max}$)

G4BREPSolidPCone(Name, ϕ_s, ϕ_d, n_{planes}, z_{start}, $\underline{z}_{values}$, $\underline{r}_{min}$, $\underline{r}_{max}$)

n_{planes}=7, n_{sides}=5

G4Torus and G4EllipticalTube

G4Torus(Name,ri,ra,R, ϕ_s, ϕ_d)

G4EllipticalTube(Name,px,py,fDz)

Testing the Solids

With traditional tools

- test10: Optical photons geometry test
- SBT, fred: tests behavior of CSG solids and specific
- SolidsChecker: Optical photons geometry test

Direct call of `DistanceToIn()`, `Inside()`, `SurfaceNormal()` etc.

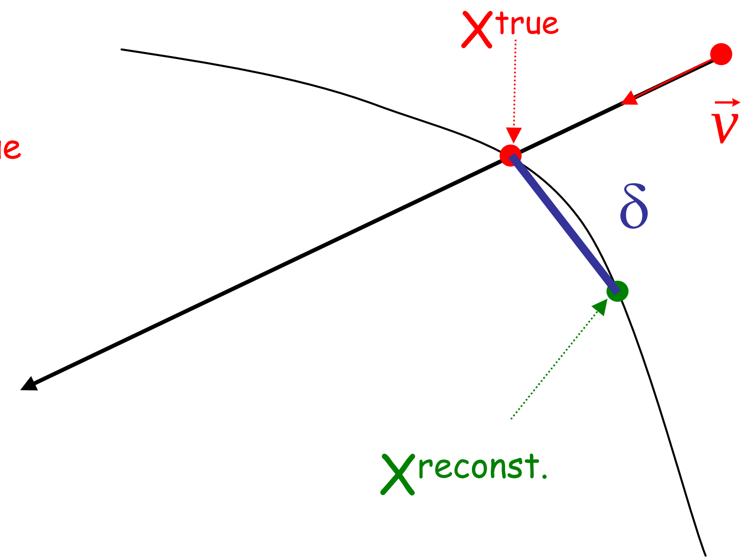
... and with a new testing tool

- SurfaceChecker: **Full tracking and Analysis**

The SurfaceChecker

1. Generate a random particle position P
2. Generate a random point X^{true} on the surface of the solid.
3. Ask G4 tracking for the intersection point given the point P and its direction v

➡ $X^{reconst.}$



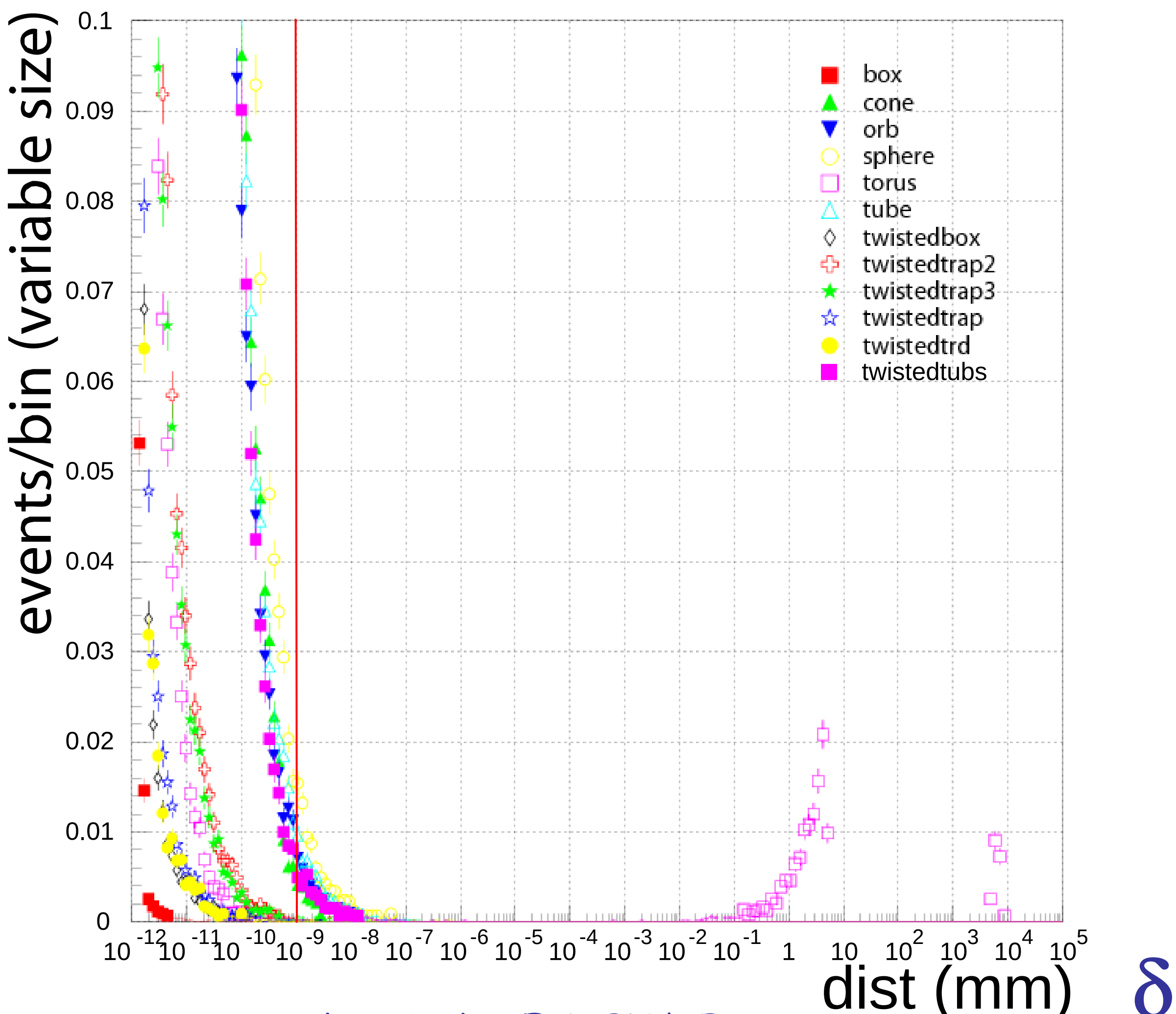
4. Automatic analysis: Select the intersection closest to Xtrue. This gives the **distance** δ

The **distance** δ between X^{true} and $X^{reconst.}$ gives us information about the goodness of the reconstruction.

Results I

World size
$10 \times 10 \times 10 m^3$

Solid sizes
L =10-50cm



Results II

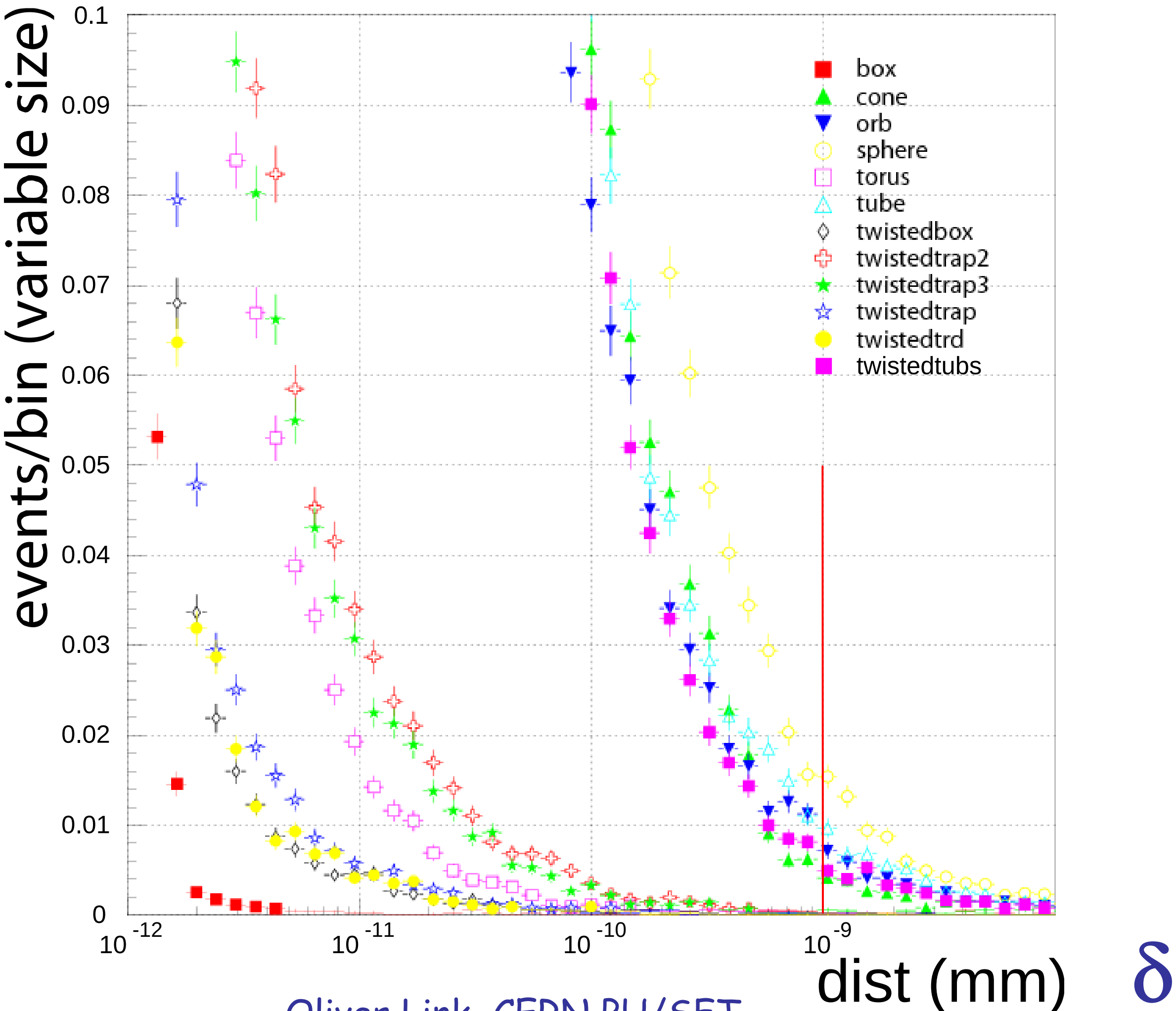
- The δ-plot revealed a severe problem with G4Torus. (1% wrong)
- It was due to numerical inaccuracy of the method used in DistanceToIn/Out
- The G4Torus DistanceToIn/Out algorithm was rewritten with the Jenkins-Traub Root finder.

Results III

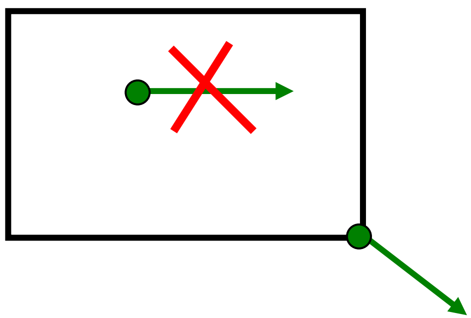
- With the new test suite we discovered a bug in G4Sphere
  - Only triggered in shells, **not** in half-spheres or full spheres.
  - Was dependent on the opening angle θ
- The cause was the conical surface.

Results IV

kCarTolerance
10^{-9}mm



Review of Surface Normal



- Particle position on surface
- Take corner and edges into account

Important for **optical** processes

- Implementation of `SurfaceNormal()` has been done for most CSG solids.
  - Now included in 7.1
- Still remaining for specific solids and BREPS
- Performance impact?
  - Today loss of 10% per-call measured for method `SurfaceNormal()` of G4Box.

Development currently going on …

- Design iteration for multiple/parallel navigators and propagation in field undergoing
- Importance Biasing and scoring
  - Coupled with multiple/parallel navigator
- Study to make tolerance tunable
  - Define formula for kCarTolerance dependent on the world size or make it explicitly tunable.
  - Define mechanism for checking and warning about unphysically small dimensions of solids.
- Optimized navigation in regular geometry structures (phantoms)

Summary

- New Overlap checking facility
- Added 7 new solids
- Added Jenkins-Traub Algorithm to solve high order polynoms
- New test SurfaceChecker for solids.
- Revision of surface normal

Padé Expansion

- The Padé approximation can be thought of as a generalization of a Taylor polynomial.
- More precisely, a Padé approximation of degree (*m*,*k*) to a function *f*(*x*) at a point x_0 is the rational function p(x)/q(x) where *p*(*x*) is a polynomial of degree *m*, *q*(*x*) is a polynomial of degree *k*
- Here is the Padé approximation of degree (2,4) to cos(x) at x=0:

$$\frac{1-\frac{61x^2}{150}}{1+\frac{7x^2}{75}+\frac{x^4}{200}}$$