The DIRC photon propagation software within PandaRoot

- Short description
- Installation
- Documentation
- Visualization examples
- Code
  - surfaces, volumes, connections, managing....
- PandaRoot
- Framework history
- Summary

Short description

- Set of routines to play with photon propagation
  - C++, STL, root
  - Doxygen documentation
  - examples available
  - part of pandaroot
- Surfaces, volumes, materials
  - can be connected
  - accessible by a “manager”
- Examples exist
  - independent on pandaroot

```
drcprop> ls
CMakeLists.txt          DrcOptMirror.o          DrcSurfCyl.h
Doxyfile                DrcOptPixel.cxx         DrcSurfCyl.o
DrcOptDev.cxx           DrcOptPixel.h           DrcSurfPolyFlat.cxx
DrcOptDev.h             DrcOptPixel.o           DrcSurfPolyFlat.h
DrcOptDev.o             DrcOptReflAbs.h         DrcSurfPolyFlat.o
DrcOptDevManager.cxx    DrcOptReflNone.cxx      DrcSurfPolyPara.cxx
DrcOptDevManager.h      DrcOptReflNone.h        DrcSurfPolyPara.h
DrcOptDevManager.o      DrcOptReflNone.o        DrcSurfPolyPara.o
DrcOptDevSys.cxx        DrcOptReflPerfect.cxx   DrcSurfPolySphere.cxx
DrcOptDevSys.h          DrcOptReflPerfect.h     DrcSurfPolySphere.h
DrcOptDevSys.o          DrcOptReflPerfect.o     DrcSurfPolySphere.o
DrcOptLens.cxx          DrcOptReflSilver.cxx    DrcSurfPolySphere.odg
DrcOptLens.h            DrcOptReflSilver.h      DrcSurfQuadFlatDiff.cxx
DrcOptLens.o            DrcOptReflSilver.o      DrcSurfQuadFlatDiff.h
DrcOptMatAbs.cxx        DrcOptVol.cxx           DrcSurfQuadFlatDiff.o
DrcOptMatAbs.h          DrcOptVol.h             DrcUtil.cxx
DrcOptMatAbs.o          DrcOptVol.o             DrcUtil.h
DrcOptMatLithotecQ0.cxx DrcPhoton.cxx           DrcUtil.o
DrcOptMatLithotecQ0.h   DrcPhoton.h             Makefile.am
DrcOptMatLithotecQ0.o   DrcPhoton.o             Makefile.in
DrcOptMatVacuum.cxx     DrcPropLinkDef.h        Readme
DrcOptMatVacuum.h       DrcSurfAbs.cxx          Todo.txt
DrcOptMatVacuum.o       DrcSurfAbs.h            doc
DrcOptMirror.cxx        DrcSurfAbs.o            examples
DrcOptMirror.h          DrcSurfCyl.cxx          pics
drcprop>
```

```
examples> ls
Makefile            test_disk       test_lens.cc        test_simple_bar.o
test_absorption     test_disk.cc    test_lens.o         test_test
test_absorption.cc  test_disk.o     test_sheet          test_test.cc
test_absorption.o   test_focus      test_sheet.cc       test_test.o
test_cylinder       test_focus.cc   test_sheet.o
test_cylinder.cc    test_focus.o    test_simple_bar
test_cylinder.o     test_lens       test_simple_bar.cc
examples>
```

Installation

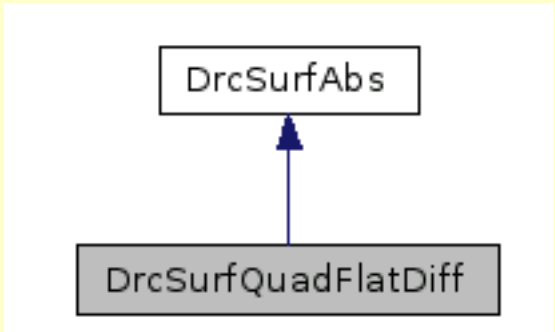
- get it from svn repository
  - svn co https://subversion.gsi.de/fairroot/pandaroot
- cd pandaroot/drc/drcprop/examples
- make (root needs to be present)
- run it
  - eg. test_simple_bar

Documentation

doxygen

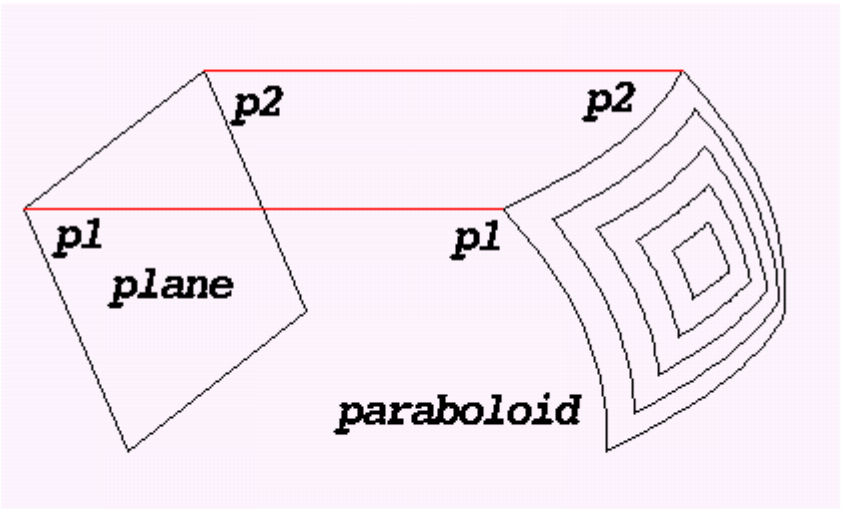
Public Member Functions

	DrcSurfQuadFlatDiff () *Empty constructor.*
virtual	**~DrcSurfQuadFlatDiff** () *Destructor.*
	DrcSurfQuadFlatDiff (const **DrcSurfQuadFlatDiff** &s) *Copy constructor.*
DrcSurfQuadFlatDiff &	**operator=** (const **DrcSurfQuadFlatDiff** &s) *Assignment operator.*
void	**addSurface** (const **DrcSurfAbs** &surf, TVector3 p1, TVector3 p2) *Add surface to connect to and 2 points.*
virtual **DrcSurfQuadFlatDiff** *	**clone** () const *Virtual copy constructor.*
TVector3	**normal** (const TVector3 &point) const *Normal vector.*
bool	**surfaceHit** (**DrcPhoton** &ph *Intersection vector.*
void	**print** (fstream &stream) con *Write surface coordinates to*
void	**shift** (const TVector3 &shift) *Perform a shift of the object*
void	**rotate** (const TRotation &rot *Perform a rotation of the obj*
virtual bool	**isFlat** () *Flag if surface is flat.*



Detailed Description

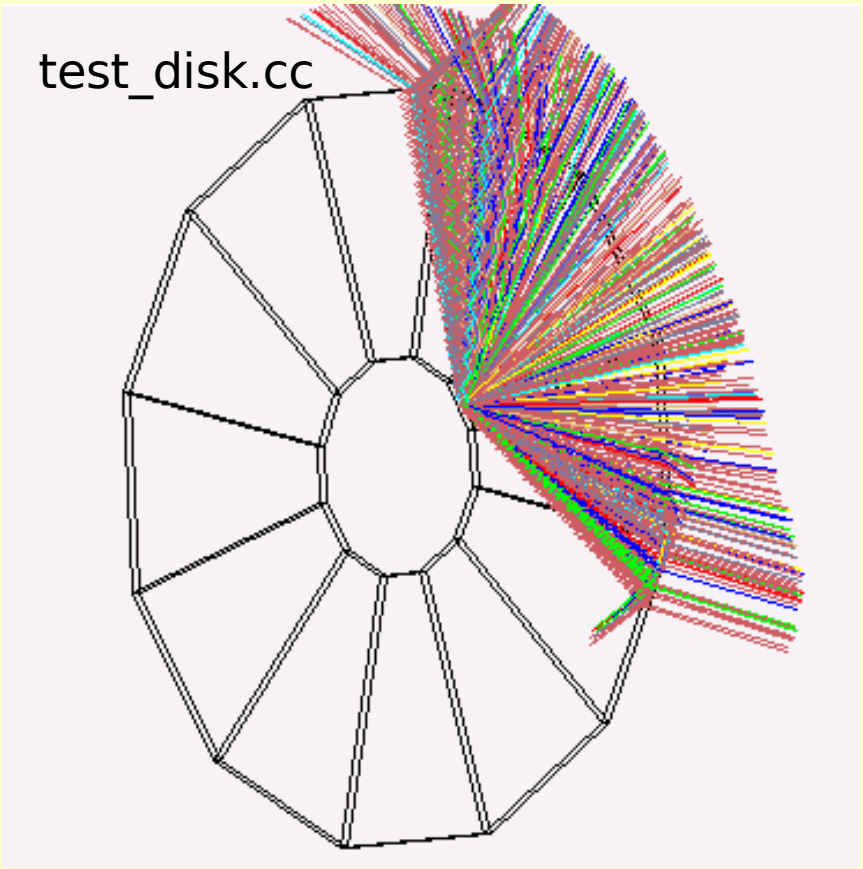
Abstract class for flat surface representation, defined by 2 points of one surface and 2 points of another surface. The 4 points have to be within a plane. The purpose of this class is to generate surfaces between one surface and a non flat surface. Eg. the surfaces between a flat surface (one side) and a spherical surface (the other side) to construct the sides of a lens.

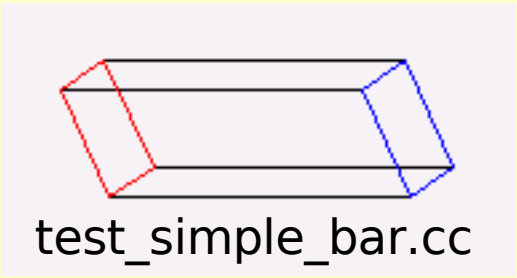


Since you connect to non planar surfaces which are created in an unshifted and unrotated coordinate system, you should generate the difference surface in the same system and rotate the whole construct after the generation.

Examples

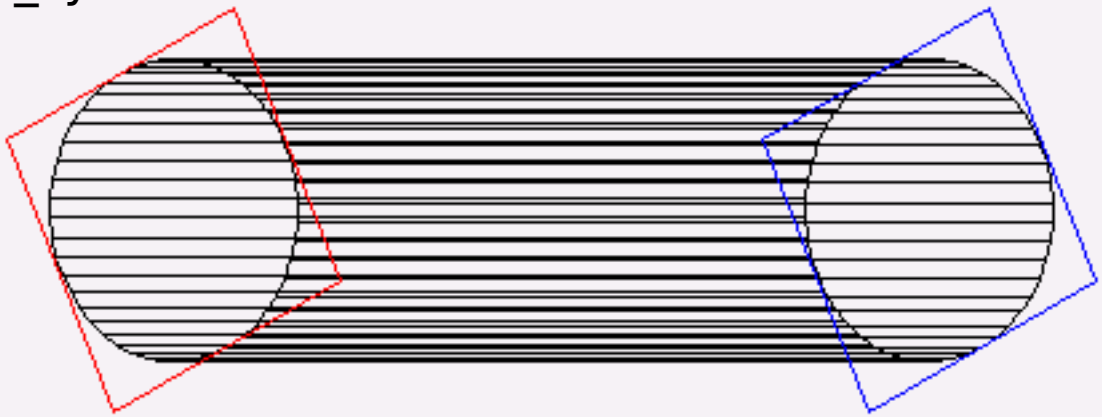
test_disk.cc



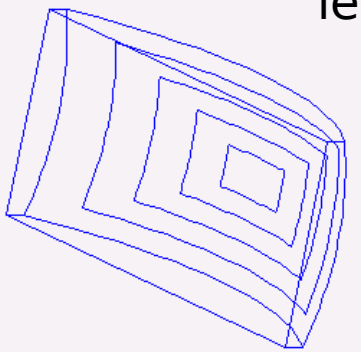


test_simple_bar.cc

test_cylinder.cc



lens



Code

- basic entity is a surface
  - flat or non flat surfaces available
  - reflectivity
- surfaces can be combined to volumes
  - material
- surfaces of different volumes can be connected and form a device system
- manager administrates
  - device systems, volumes, and surfaces
  - generates Cherenkov radiation
  - volume drawing (can be switched off)
  - photon track visualization (can be switched off)

Code - surfaces

```
// 8 points define a bar

TVector3 p1(-10.0, +5.0, 0);      //         p5----------p8
TVector3 p2(-10.0, -5.0, 0);      //        /|          /|
TVector3 p3(+10.0, -5.0, 0);      //       / |         / |
TVector3 p4(+10.0, +5.0, 0);      //      /  p6--------/--p7
//                                        /  /        /  /
//                                       /  /        /  /
//                                      /  /        /  /
//                                     /  /        /  /
TVector3 p5(-10.0, +5.0, -40); // p1----------p4 /
TVector3 p6(-10.0, -5.0, -40);  //  | /          | /
TVector3 p7(+10.0, -5.0, -40); //  |/           |/
TVector3 p8(+10.0, +5.0, -40);//p2----------p3

// Define from points 6 surfaces of the bar. The points have to be given in
// the sequence going around the surface, clock- or counterclock-wise.
// There are 2 additional surfaces, a mirror and a screen.

// How to produce surfaces by shift and rotate operation is for sake of clearness
// not shown here, but in one of the other examples.

// Declare flat surfaces with arbitrary number of points.

DrcSurfPolyFlat a1;

a1.addPoint(p1);
a1.addPoint(p2);
a1.addPoint(p3);
a1.addPoint(p4);
a1.setName("pfront");
```

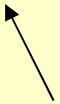
Code
- volumes

```
// create a volume consiting of surfaces
// create a material the bar will consist of

DrcOptVol bar;

DrcOptMatLithotecQ0 quartz;
bar.setOptMaterial(quartz);

bar.addSurface(a1);
bar.addSurface(a2);
bar.addSurface(a3);
bar.addSurface(a4);
bar.addSurface(a5);
bar.addSurface(a6);
bar.setName("bar");
```

deep copy of surface

Code -connections

```
// Build a optical system consisting out of several volumes,
// mirrors and screens.
// This layer has the advantage, that a device consisting out
// of many equal subsystems
// like a bar box, easily can be reproduced.

DrcOptDevSys opt_system;

opt_system.addDevice(bar);
opt_system.addDevice(screen);
opt_system.addDevice(mirror);

// couple surface 1 of device 1 with surface 2 of device 2
//                                    dev1   dev2    surf1     surf2
opt_system.coupleDevice("bar","screen","pback","screen_front");
opt_system.coupleDevice("bar","mirror","pfront","mirror_front");
```



names allow to identify objects

Code
- managing

```
// The manager must be created as pointer. It is created as singleton, that is only
// one manager can exist per application.

DrcOptDevManager* manager = new DrcOptDevManager();

manager->addDeviceSystem(opt_system);
```

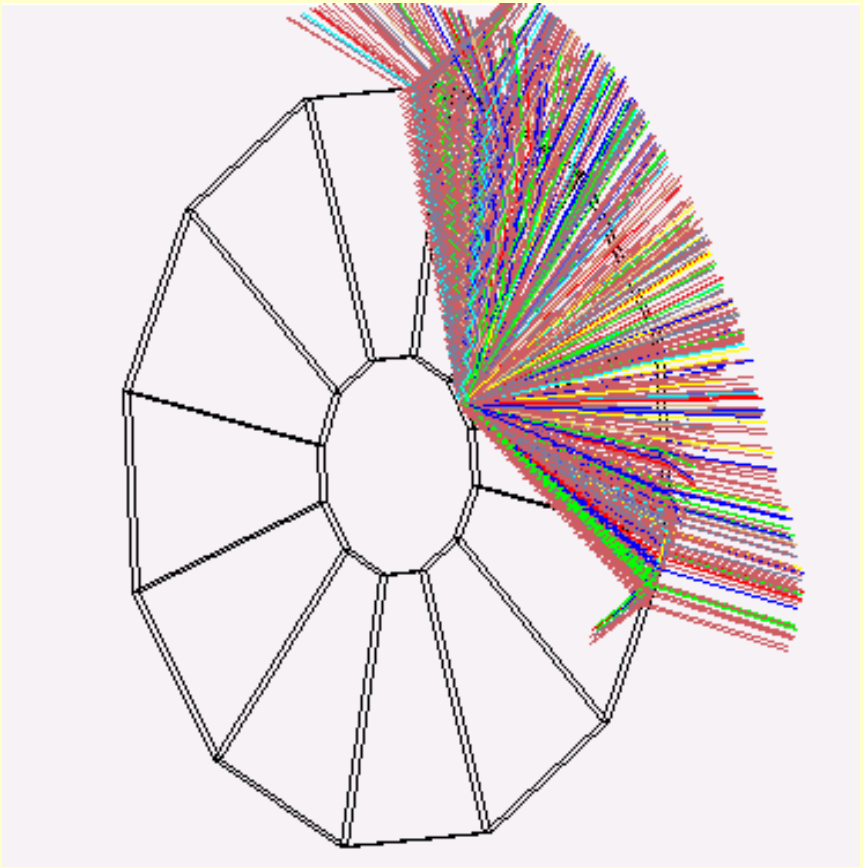
other program part:

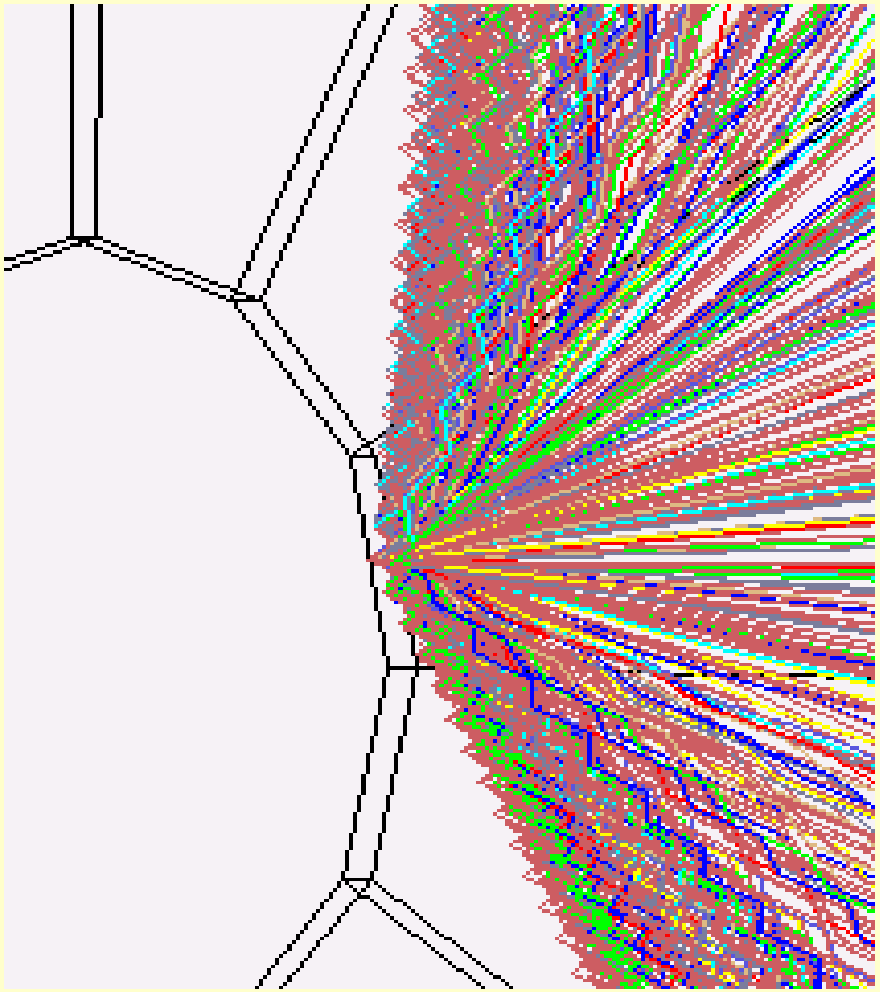
```
DrcOptDevManager manager   = DrcOptDevManager::instance();
```

Code
- run it

beta=0.99

Total photons: 844
measured: 521
absorbed: 4
lost: 319





root Geo.C

Code -photons

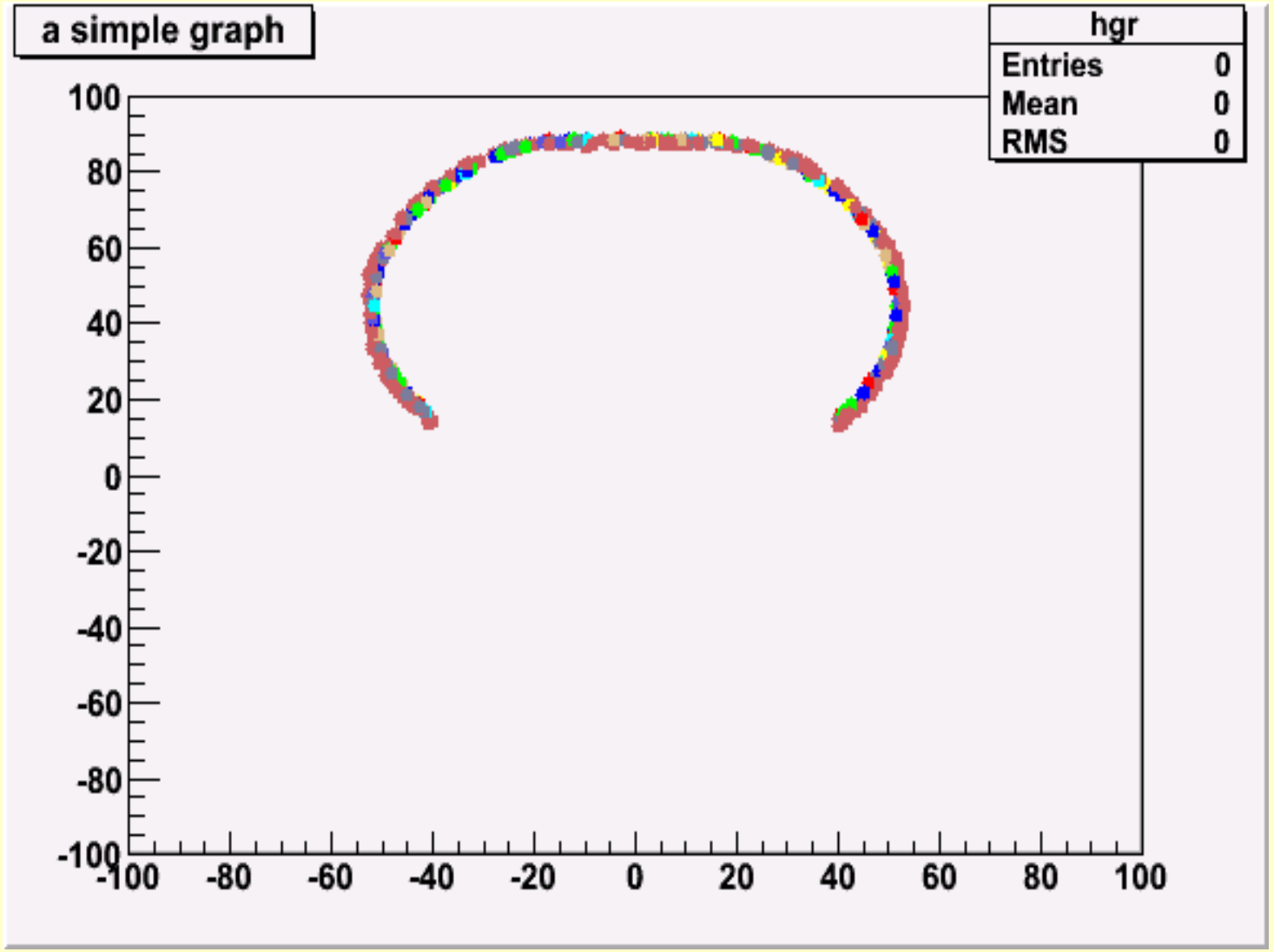
double m_lambda	Wavelength in nm.
TVector3 m_position	Actual position of photon.
TVector3 m_positionOld	Old position of photon.
TVector3 m_direction	Normalized direction of photon.
Drc::PhotonFate m_fate	The fate of the photon.
int m_reflections	Number of suffered reflections.
int m_verbosity	Verbosity level 0-5.
DrcOptDev * m_dev	Pointer to device where photon is.
double m_time	Time of flight.

photon knows about position, direction, device it is in, and fate

fate

- measured
- flying
- lost
- absorbed

Code
- results



root Screen.C

Code - extending

DrcOptMatAbs.h

```
virtual double refIndex(const double lambda) const = 0;
virtual double refIndexDeriv(const double lambda) const = 0;
virtual double refIndexDeriv2(const double lambda) const = 0;
virtual bool absorptionFlag(double lambda, double& length) const = 0;
```

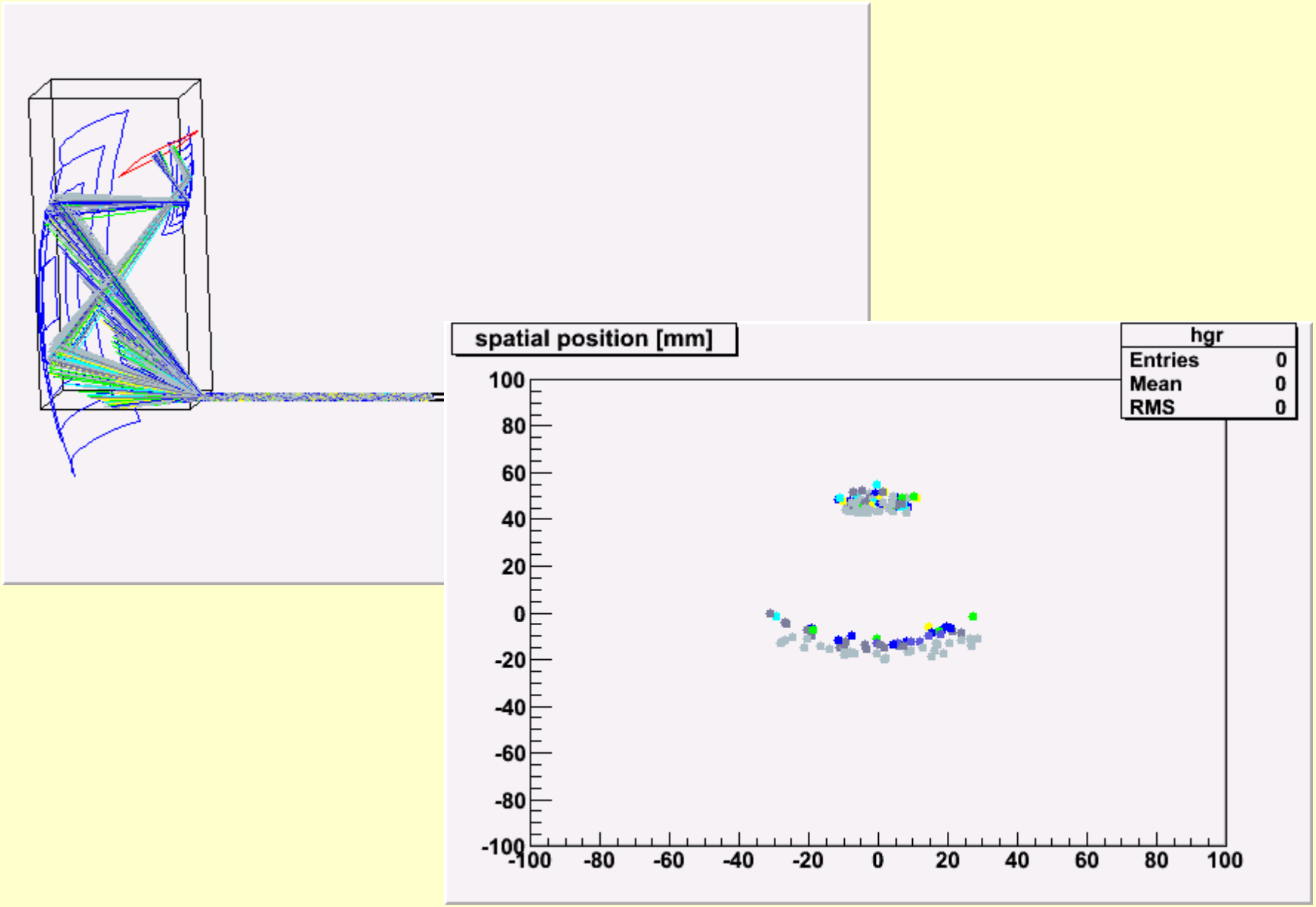
DrcOptMatLithotecQ0.h

```
class DrcOptMatLithotecQ0 : public DrcOptMatAbs
```

DrcOptMatLithotecQ0.cxx

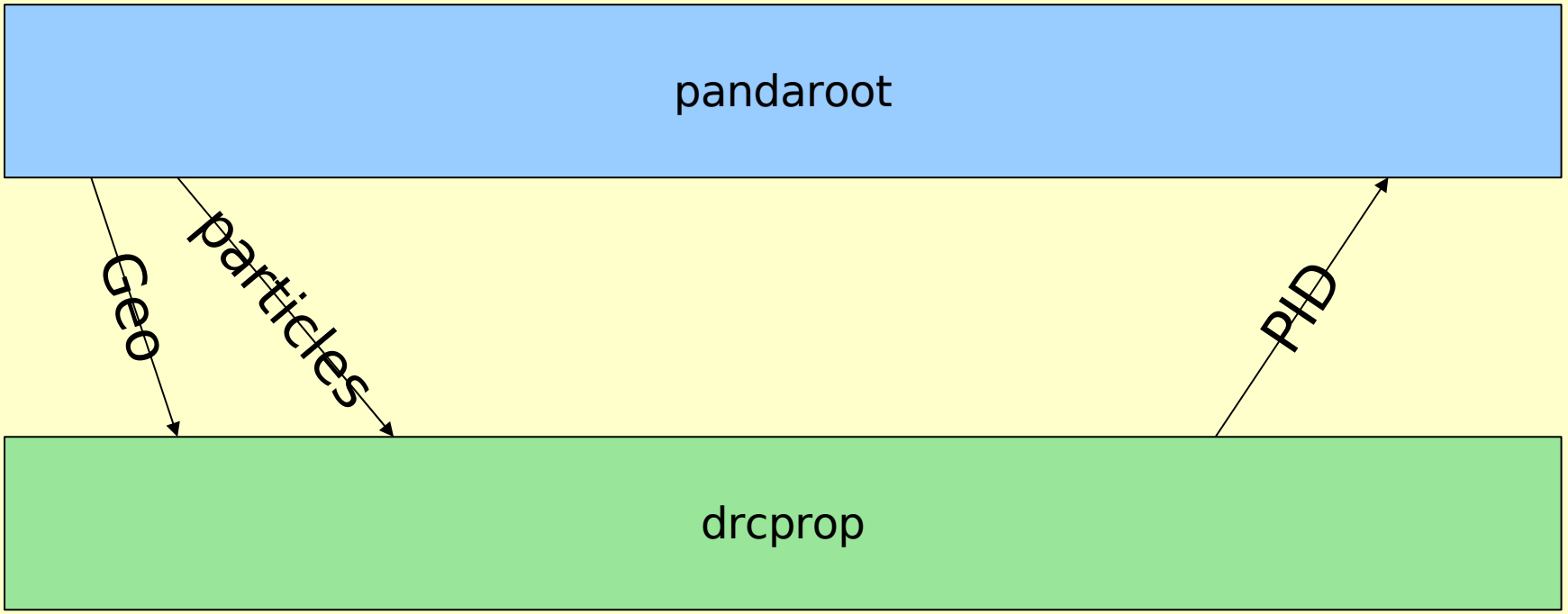
```
bool DrcOptMatLithotecQ0::absorptionFlag(double lambda, double& length) const
 { // Rayleigh scattering.
 static const double clarity = 2100*1000;      // @ 633 nm in mm (2100 m)

 double trans = exp(-(length)/(clarity*pow(lambda/663,4)));
 double cmp   = m_ran.Uniform(1.0);
 if (cmp>trans) {
 length *= trans; // eg. trans = 0.5 : 50% of way.
 return true;
 }
 return false; // no absorption.
 }
```

Code – complex geo. by embedding surfaces



PandaRoot

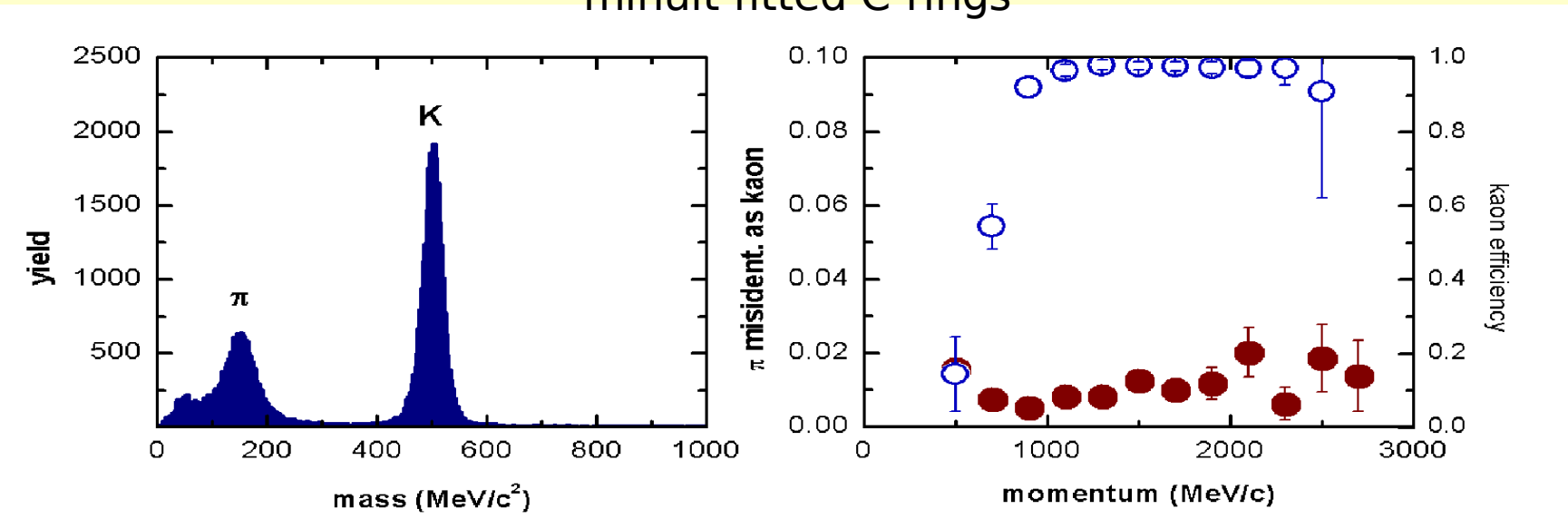
- no direct dependency
  - advantage: no installation of external packages necessary (except root)
  - disadvantage: pandaroot DIRC-geometry has to be reproduced within drcprop and results (photons, fitted rings) put back.



Framework history

- drcprop routines have a long history
  - Jim++ (reco)
  - Sergey (reco)
  - PandaBoc V0 (reco)
  - PandaBoc V1 (geo)
  - pandaroot (geo)

minuit fitted C-rings



Summary

- optical volumes defined by surfaces can be coupled and generate optical device system
- manager holds one or several device system
- manager generates cherenkov photons and propagates them
- manager holds the propagated photon list
- easy to include new materials, surfaces...