OLD DOMINION
UNIVERSITY

DIRC Tagger simulation

Carlos Ayerbe Gayoso

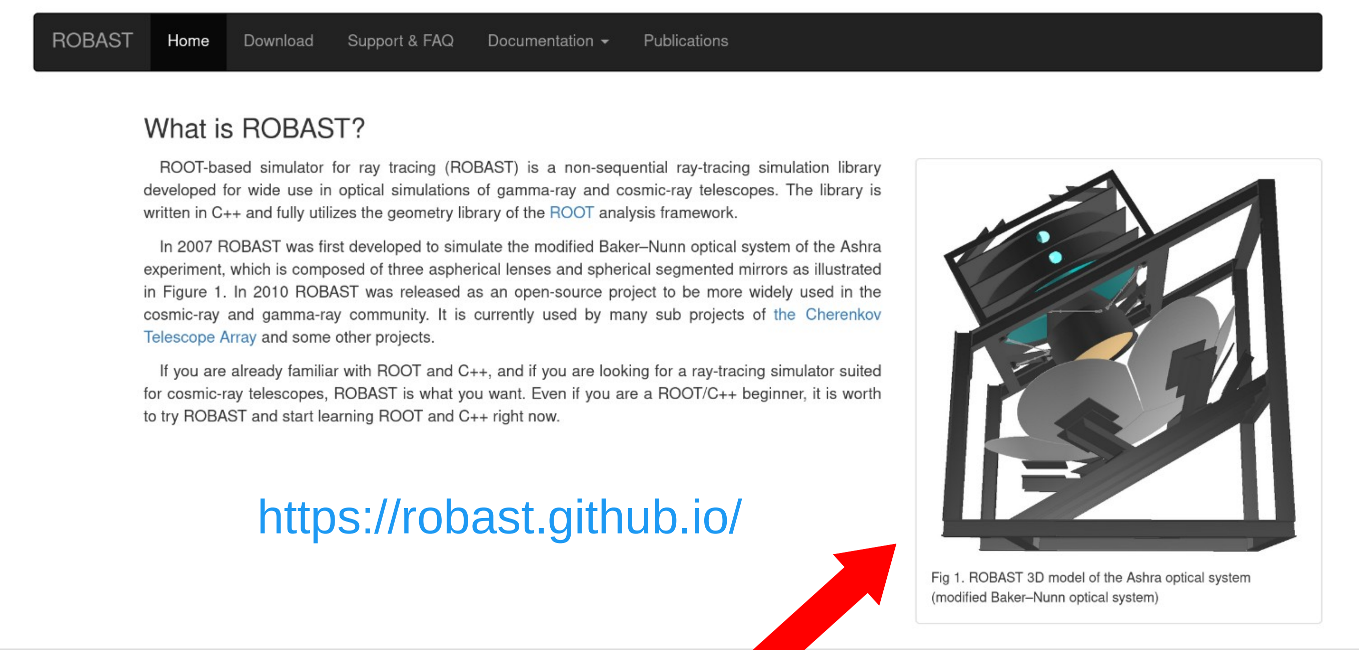
in collaboration with

Charles Hyde (ODU) and Grzegorz Kalicy (CUA)

The name of the tool

The simulation was done with a set of ROOT-based libraries called ROBAST

Simple program for ray tracing allowing to incorporate optics elements as mirrors, or focal elements which allow to analyze the information directly trough ROOT



What is ROBAST?

ROOT-based simulator for ray tracing (ROBAST) is a non-sequential ray-tracing simulation library developed for wide use in optical simulations of gamma-ray and cosmic-ray telescopes. The library is written in C++ and fully utilizes the geometry library of the ROOT analysis framework.

In 2007 ROBAST was first developed to simulate the modified Baker–Nunn optical system of the Ashra experiment, which is composed of three aspherical lenses and spherical segmented mirrors as illustrated in Figure 1. In 2010 ROBAST was released as an open-source project to be more widely used in the cosmic-ray and gamma-ray community. It is currently used by many sub projects of the Cherenkov Telescope Array and some other projects.

If you are already familiar with ROOT and C++, and if you are looking for a ray-tracing simulator suited for cosmic-ray telescopes, ROBAST is what you want. Even if you are a ROOT/C++ beginner, it is worth to try ROBAST and start learning ROOT and C++ right now.

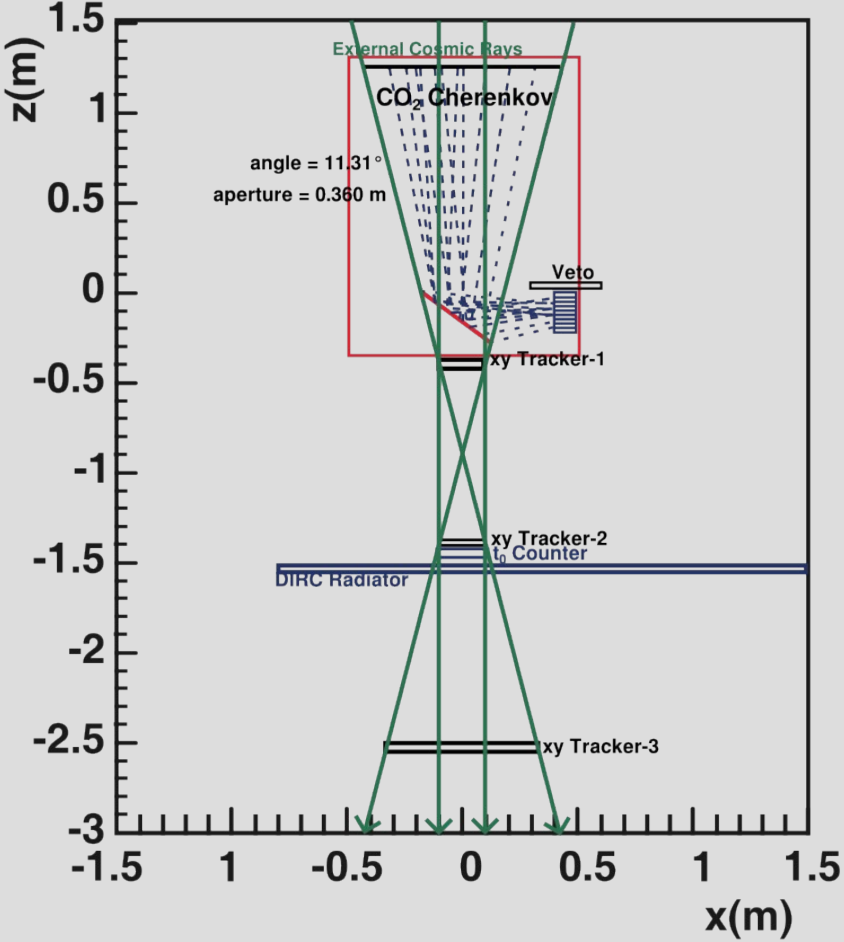
Fig 1. ROBAST 3D model of the Ashra optical system (modified Baker–Nunn optical system)

https://robast.github.io/

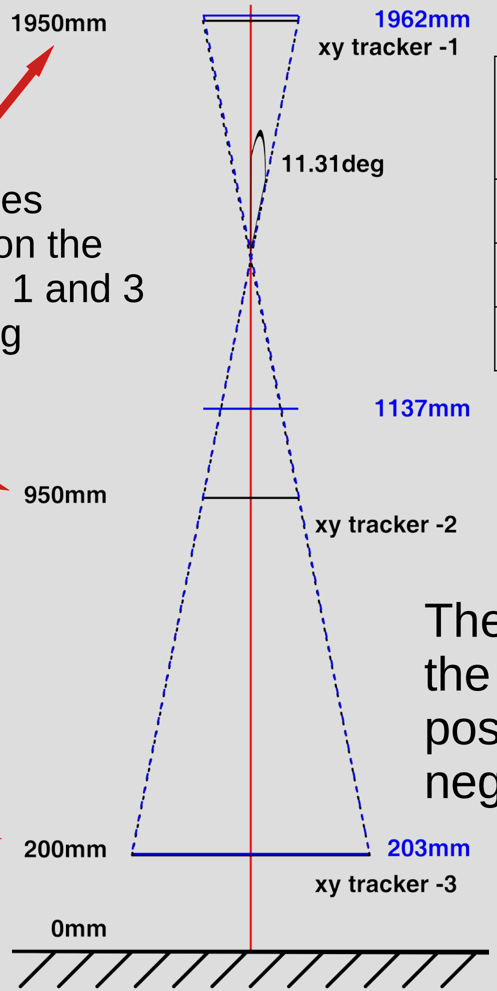
It is simple, but with some patience and ROOT-fu, mostly TGeometry, you can design complicated structures as this

The DIRC test set-up scheme

Original scheme (redone)



Geometric values (round) based on the size of trackers 1 and 3 with a 11.31 deg aperture



	Nathan's value 11/05/2023
Tracker 1	**1962 mm**
Tracker 2	**1137 mm**
Tracker 3	**203 mm**

The difference between the geometry and CAD positions are practically negligible

Placing the mirror

Originally, I started all the measurements from the focal point of the 11.31deg aperture. Perhaps was **not the clever** idea but, I kept that point as the the origin of my system of coordinates.

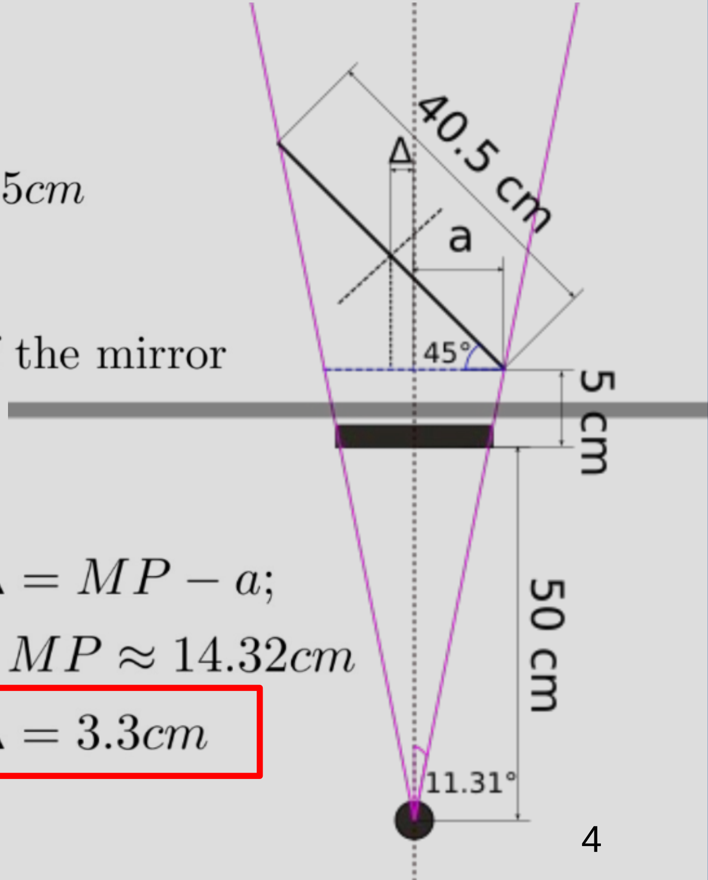
From now on, all the positions are referred to the focal point (0,0,0), which places T1 about 50 cm above it (0,0,50cm)

I have to assume certain distance between T1 and the lowest point of the mirror, 5 cm

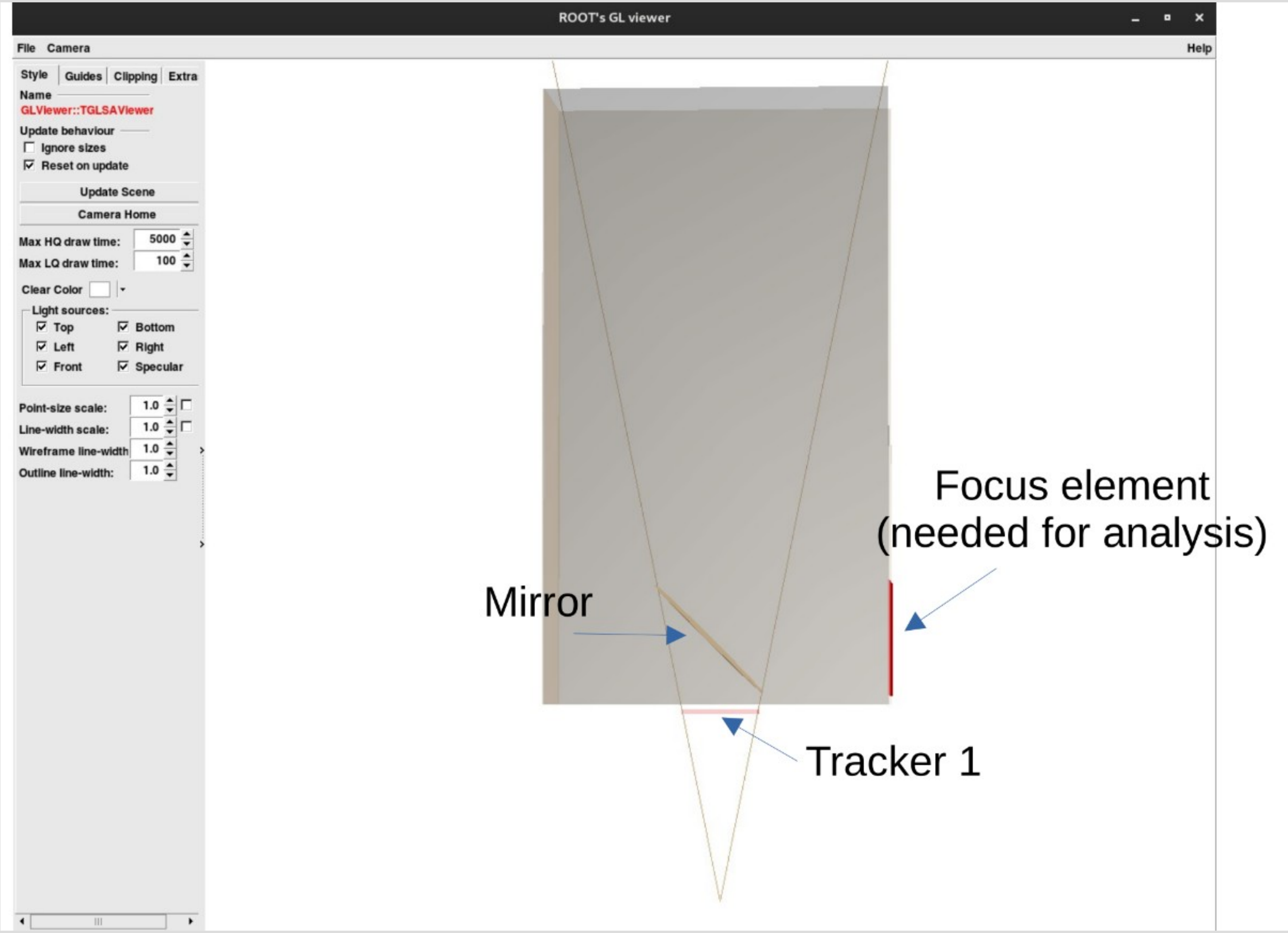
$$a = d\tan(\theta)$$
$$d = 50cm + 5cm$$
$$\theta = 11.31^\circ$$

$$MP = (ML/2)\cos(\alpha)\ \text{center of the mirror}$$
$$ML = 40.5cm\ \text{(mirror length)}$$
$$\alpha = 45^\circ$$

$$\Delta = MP - a;$$
$$a \approx 11cm;\ MP \approx 14.32cm$$
$$\boxed{\Delta = 3.3cm}$$



Building the geometry



Name
GLViewer::TGLSAViewer

Update behaviour
□ Ignore sizes
☑ Reset on update

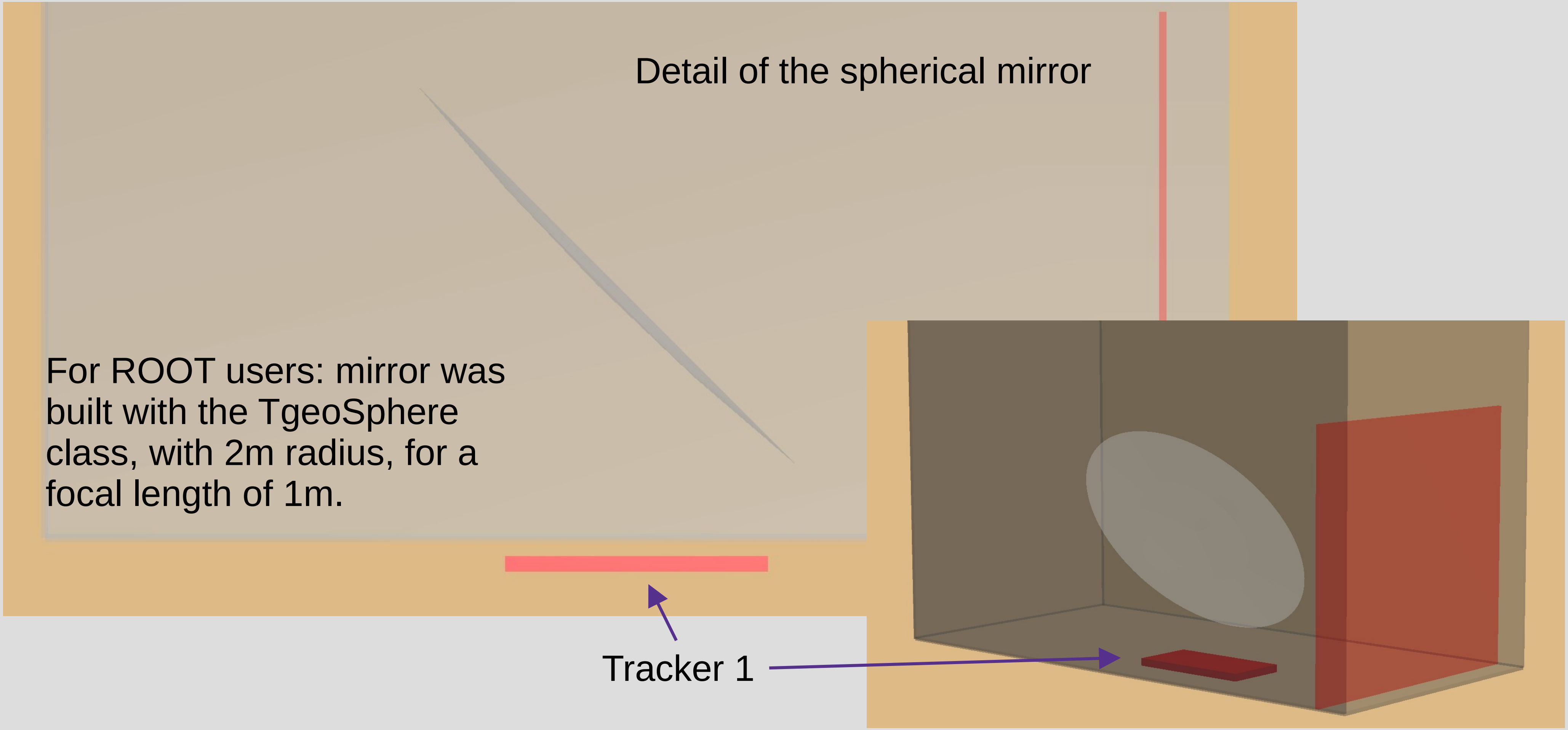
Max HQ draw time: 5000
Max LQ draw time: 100

Clear Color □

Light sources:
☑ Top ☑ Bottom
☑ Left ☑ Right
☑ Front ☑ Specular

Point-size scale: 1.0 □
Line-width scale: 1.0 □
Wireframe line-width 1.0
Outline line-width: 1.0

Building the geometry



Detail of the spherical mirror

For ROOT users: mirror was built with the TgeoSphere class, with 2m radius, for a focal length of 1m.

Tracker 1

Generating rays (muons and Cherenkov)

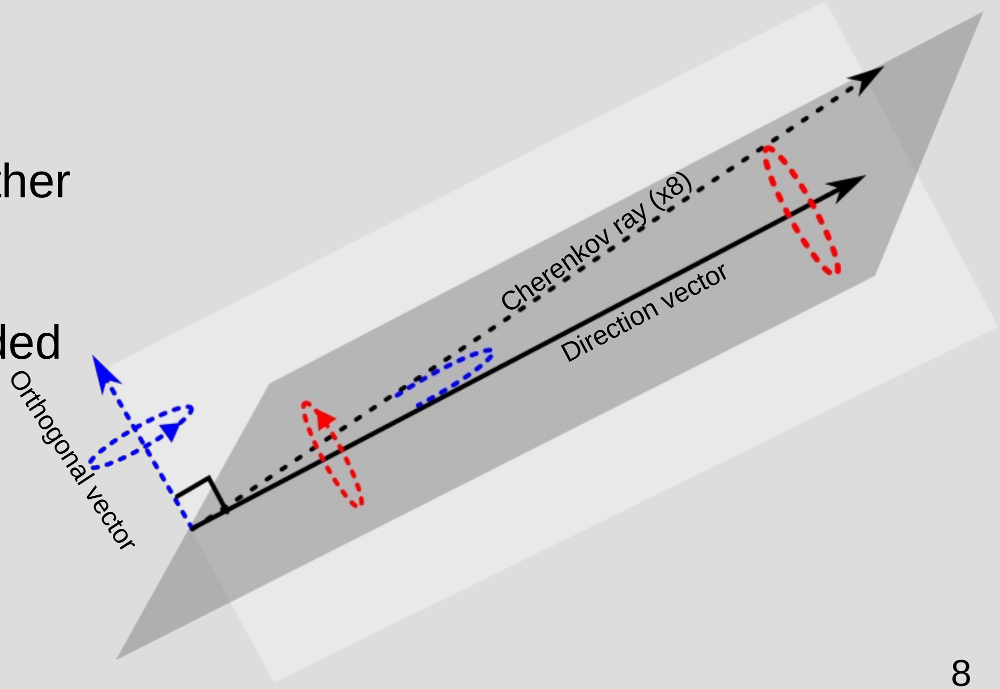
- The correct way to generate the rays (muons) should be:
  - Uniform in 10x20 cm^2 (size of T1)
  - Uniform between ~Cos(15) and 1
  - Uniform in Phi: 0, 360 deg
  - Select rays which intersect T1 and T3

This will cover the whole phase-space. **But it has the problem** of time consuming in this program, since it is a running root script and not a compiled code.

- The ‘solution’ I applied is to generate different (3) distribution of rays which always intersect T1 and T3 with the mirror in place as calculated before.
- The Cherenkov is emitted at a random position along of the ray estimated for given distribution.

Generating rays

- Initially the trajectory is determined, random the angle and the position from the top of the box.
- From that **direction vector**, an **orthogonal vector** is calculated
- A copy of the **direction vector** is created, and rotated the Cerenkov angle (1.7185deg) in the medium, with the **orthogonal vector** as rotation axis.
- This new vector, is copied 8 times.
- Each of the copied vector is rotated, with the **direction vector** as axis, separated 45deg+random phase per event, each of other (0, 45, 90, 135..), representing a cone of Cerenkov photons.
  - The random phase between 0:45deg is added to the rotation angle to avoid same photon structure at the end.

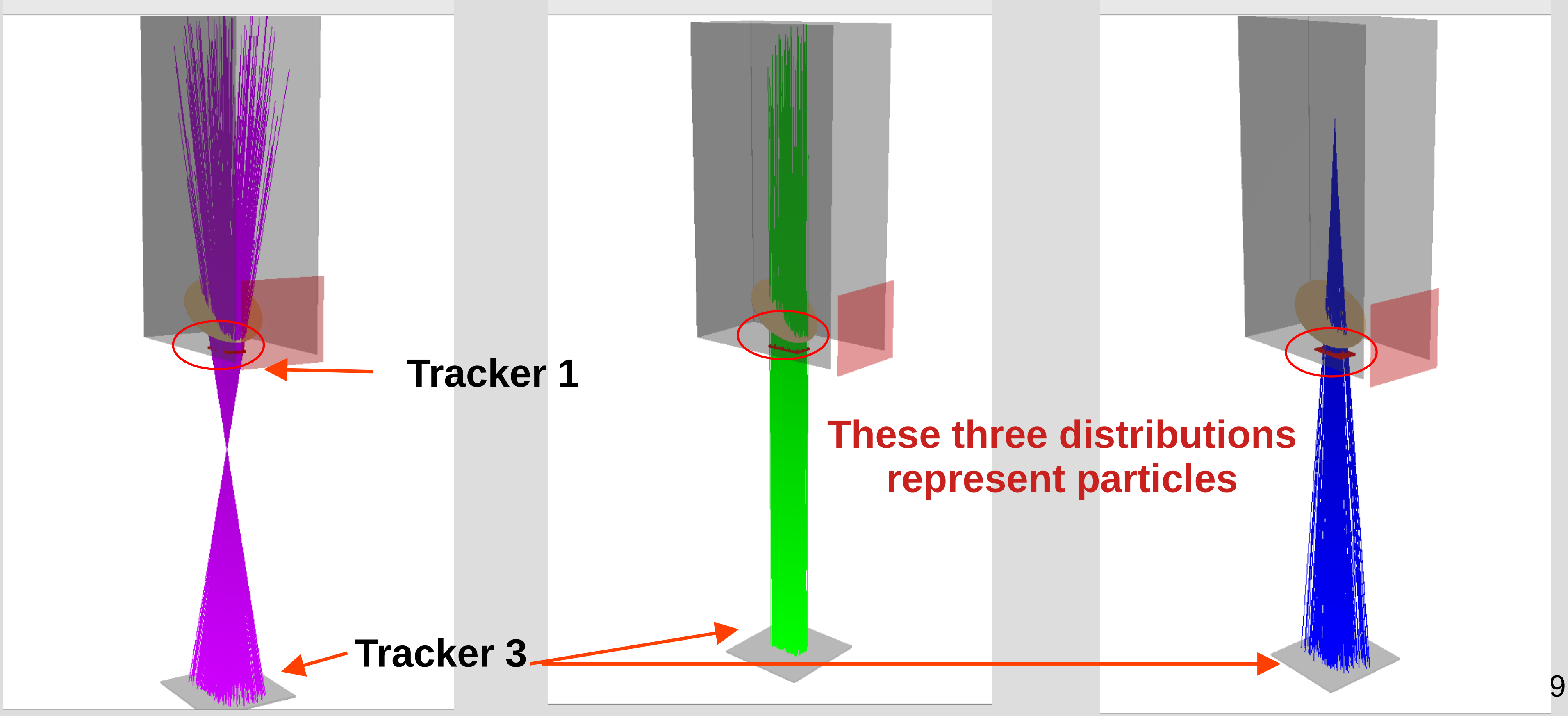


Three distributions

An elliptical cone within the 11.31 deg acceptance

Perpendicular rays covering the area of tracker 1

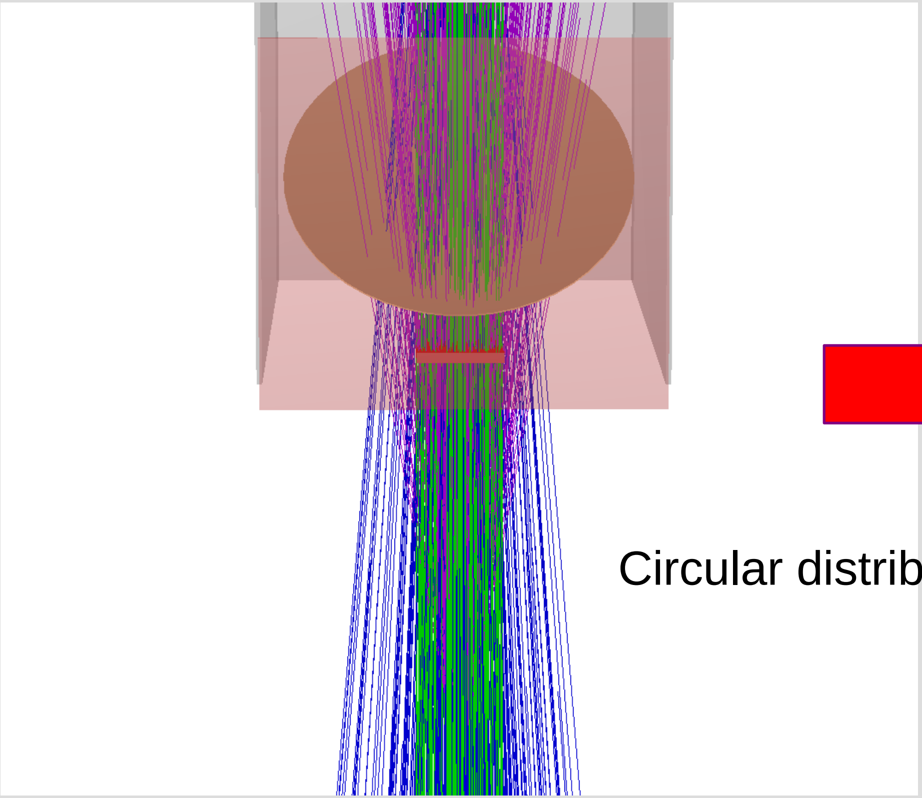
An elliptical cone with rays(particles) triggering tracker 1 and 3 but complementary to the first cone



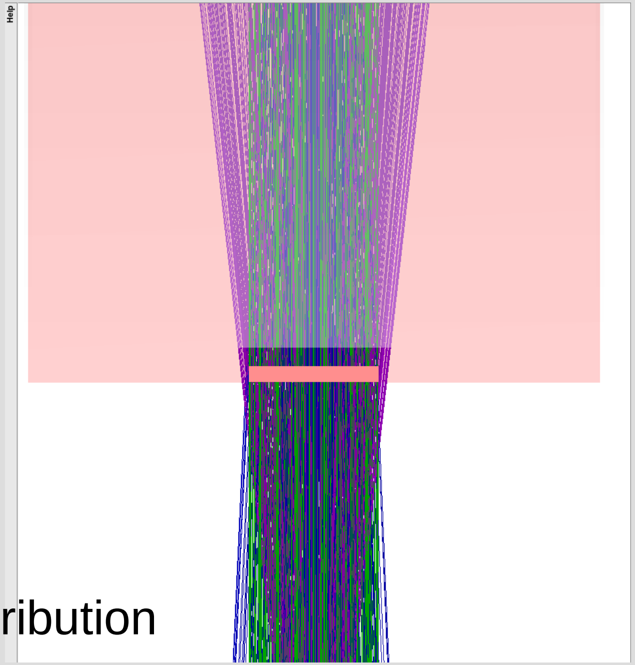
These three distributions represent particles

Details of the distributions

- The distributions **1** and **3**, are elliptical, not circular



Circular distribution



Elliptical distribution

Generating Cherenkov

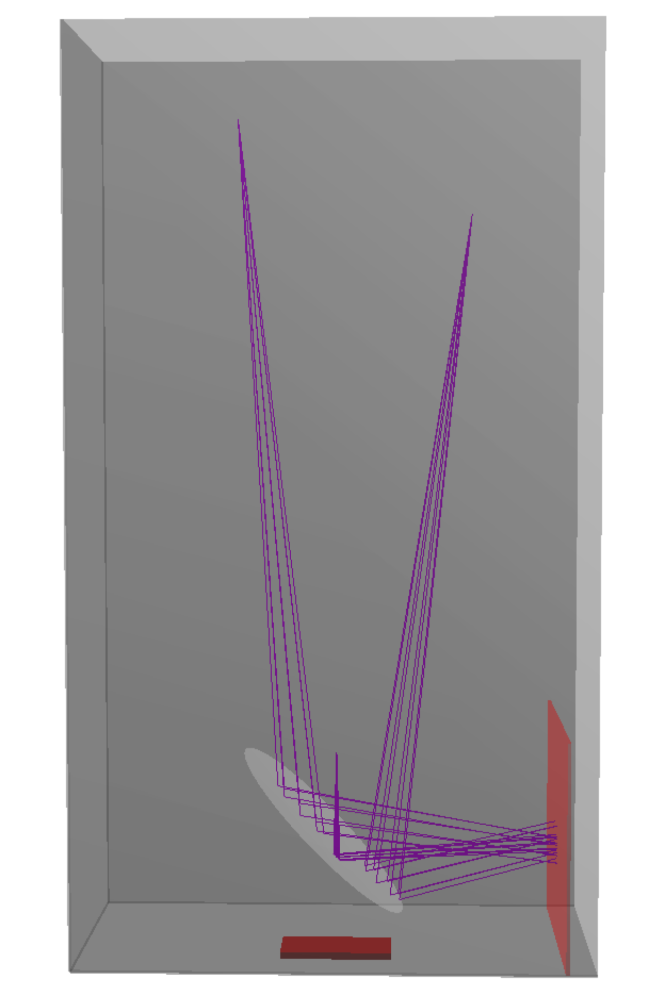
- For the elliptical cones
  - The path of the ray is determined by a random position and phi angle, inside the tagger box, which determines a radius.
  - Given the distance from top of the box to the (0,0) point, for the case **1**, or the bottom of the box and the calculated focal point inside the box, for the case **3,** the angle generated is determined (atan(r/distance))
  - The distance between the previous point and the physical limits for Cherenkov production (the tagger box) are used to randomize the point where the Cherenkov could be emitted

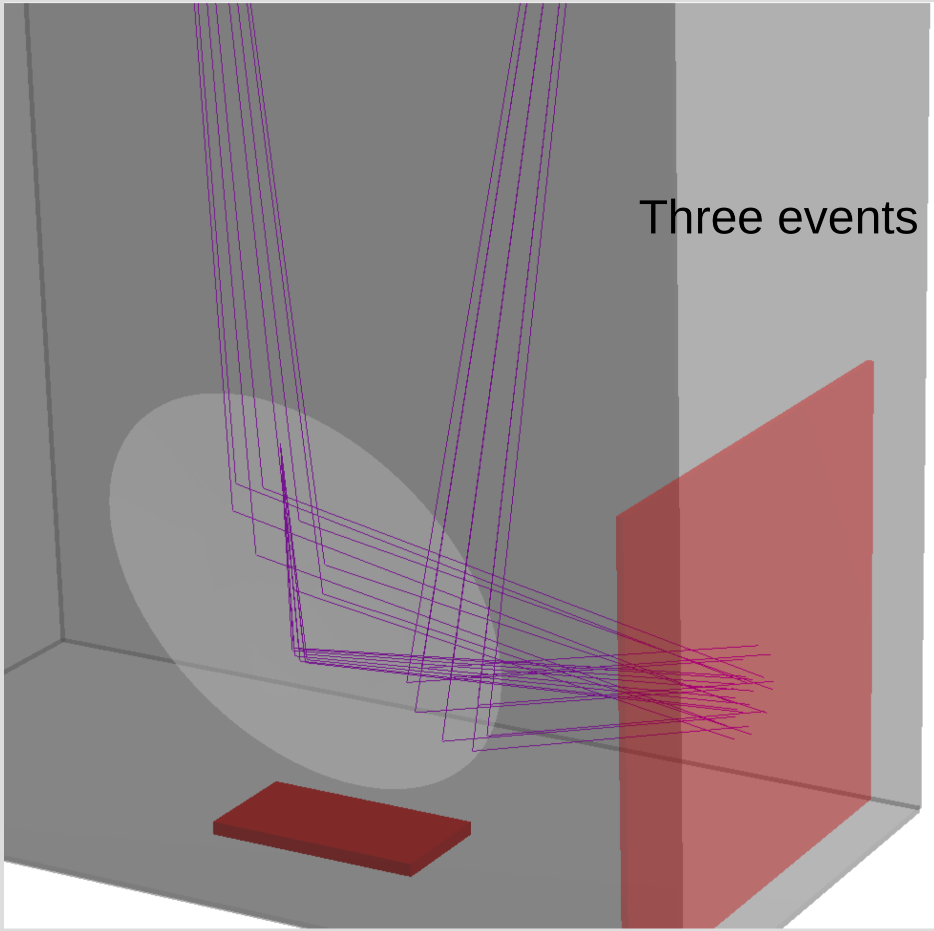
```
TRandom3 *randis  = new TRandom3();
randis->SetSeed(rseed);
Double_t disran = randis->Uniform( (out_d), d_line);
```

- For the perpendicular rays
  - Simple uniform distribution in X and Y of the size of the T1
  - Same calculation of the random point between the top and the bottom of the box.

- More sophisticated estimation of the emission point could be up to the mirror surface. Since the mirror is tilted 45deg, it makes calculus more complicated.

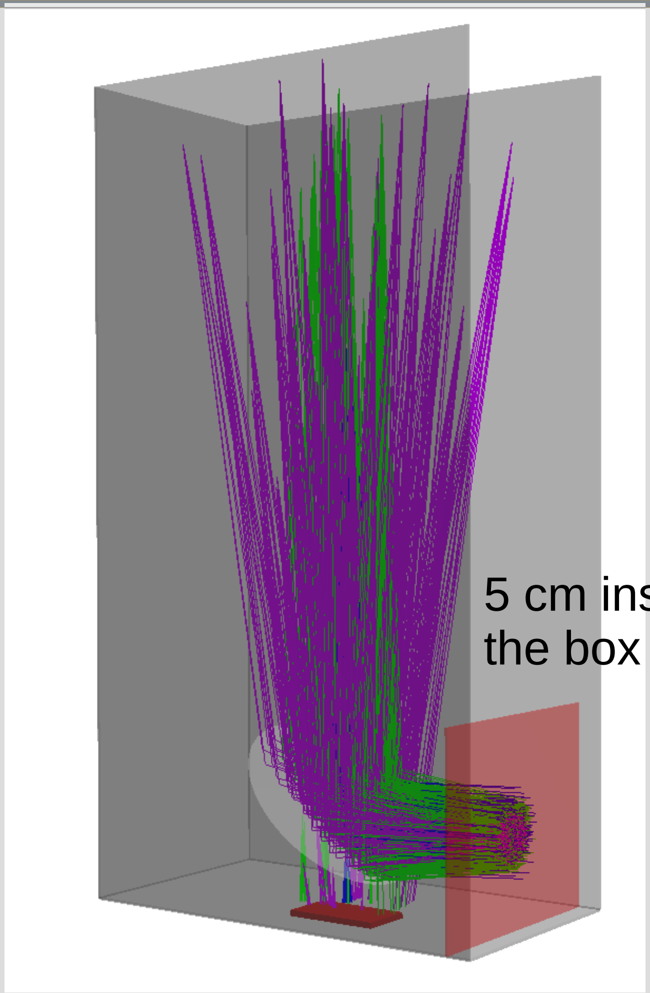
Details of the Cherenkov



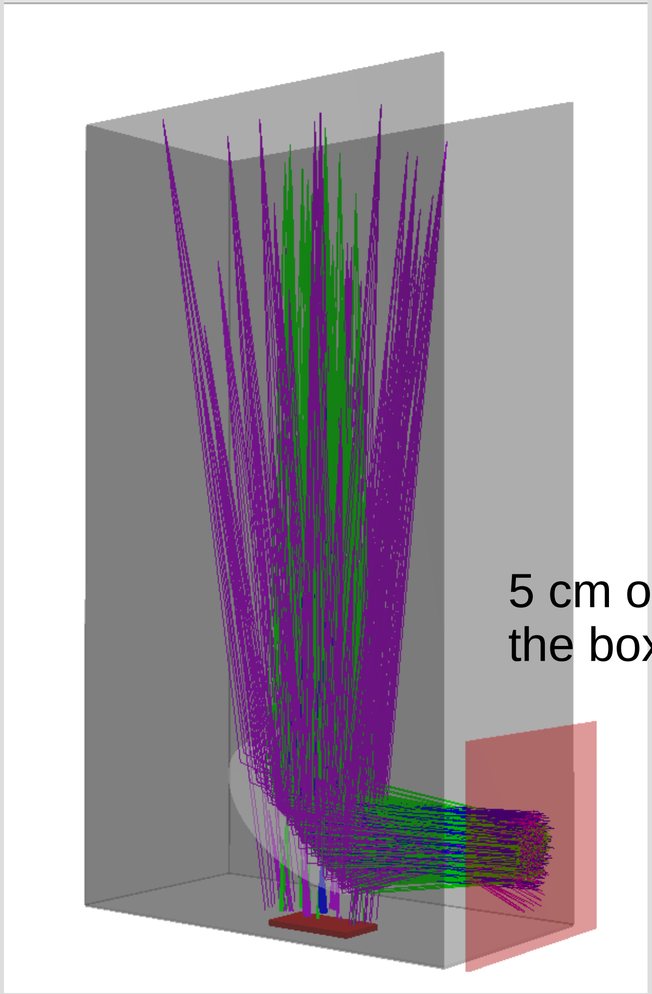


Three events

Analysis of the reflected photons



5 cm inside
the box wrt the wall



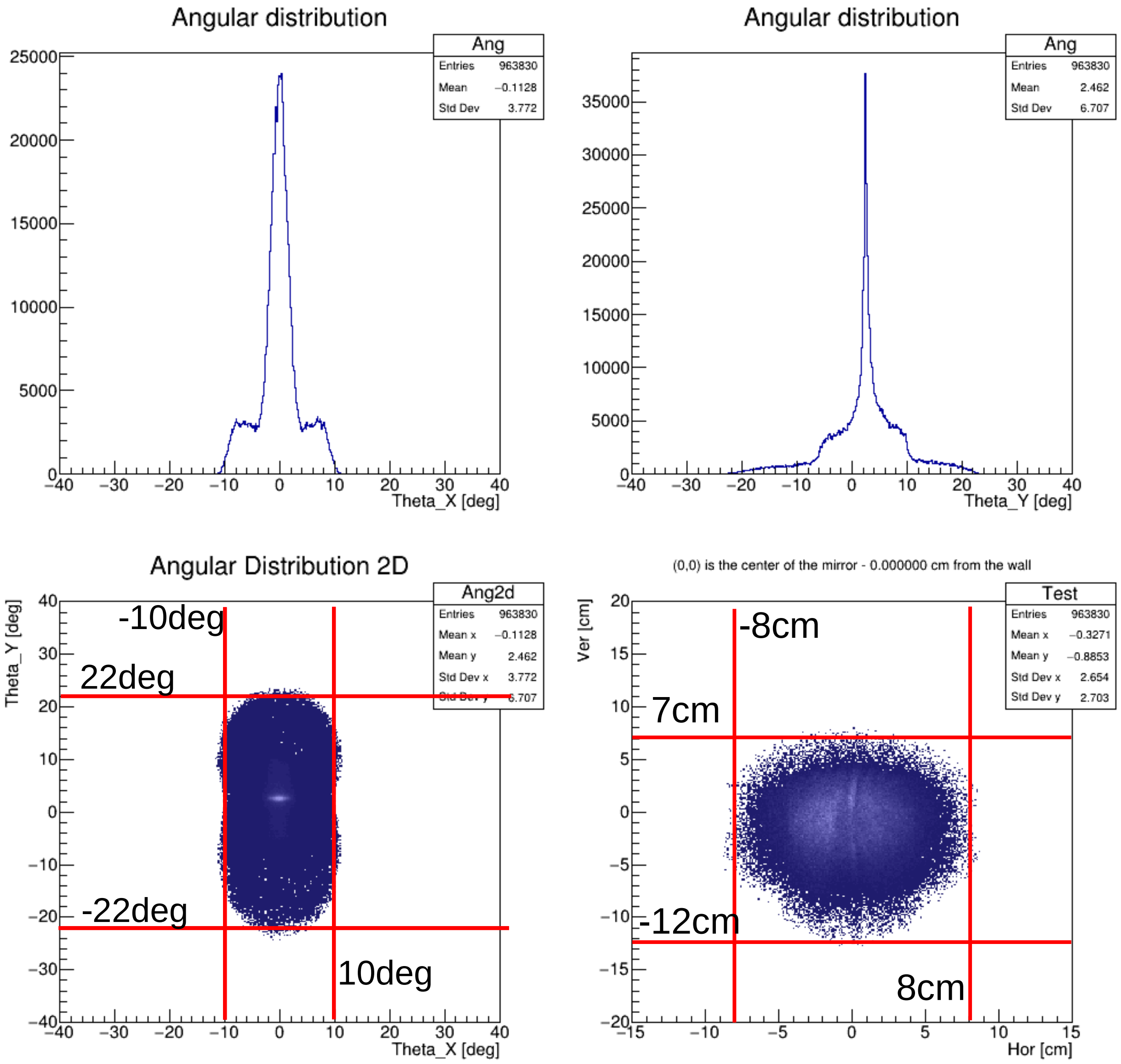
5 cm outside
the box wrt the wall

Analysis of the reflected photons

- For every event, which distribution is played is randomly choose
- 30k events were generated
  - Only 1000 events are shown in the geometry
- The focal/analyzer plane is placed, with respect the lateral wall of the box, at 0, 5, 10, -5, -10 cm

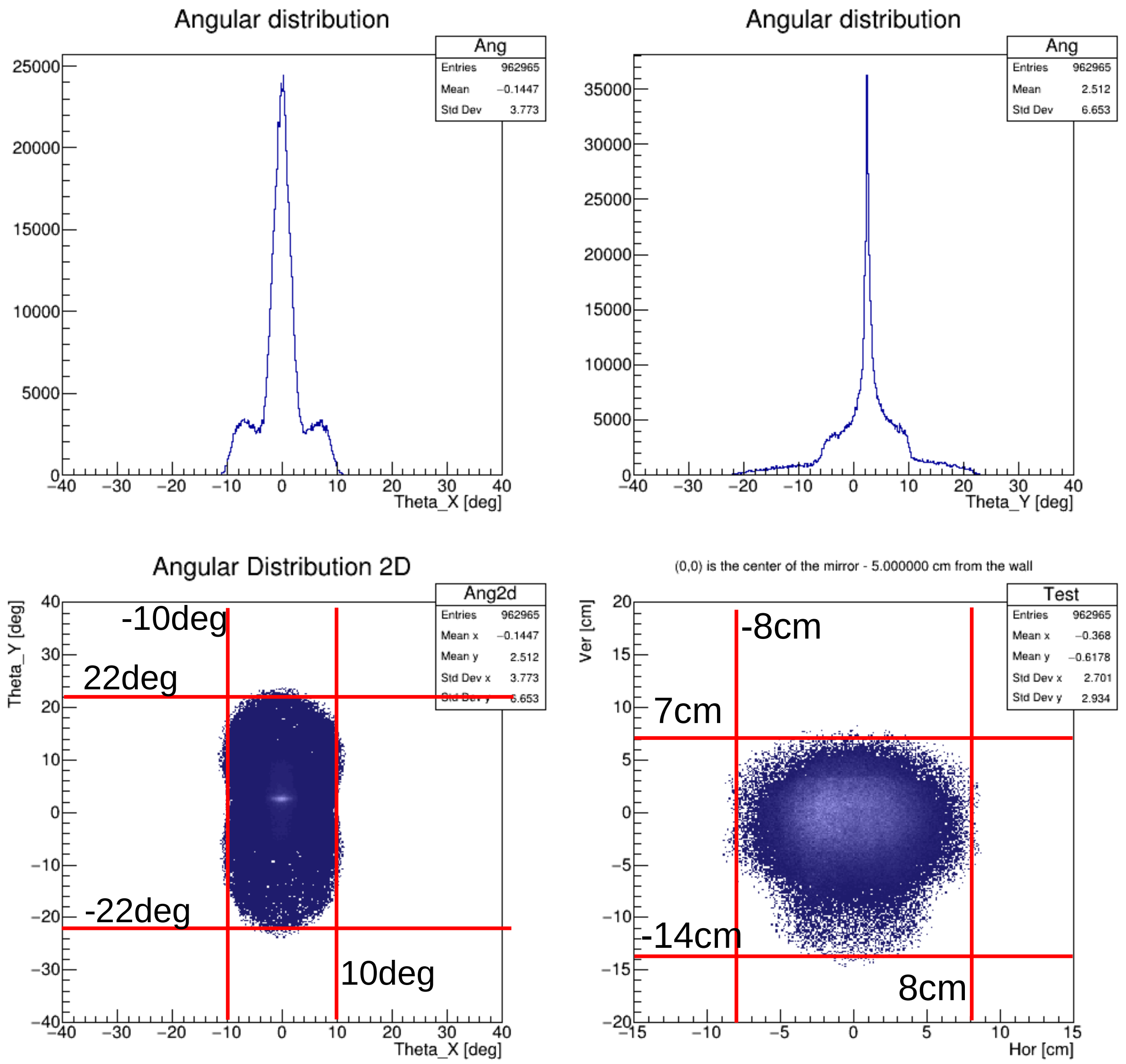
DISTANCE FROM THE SIDE WALL:

0 cm



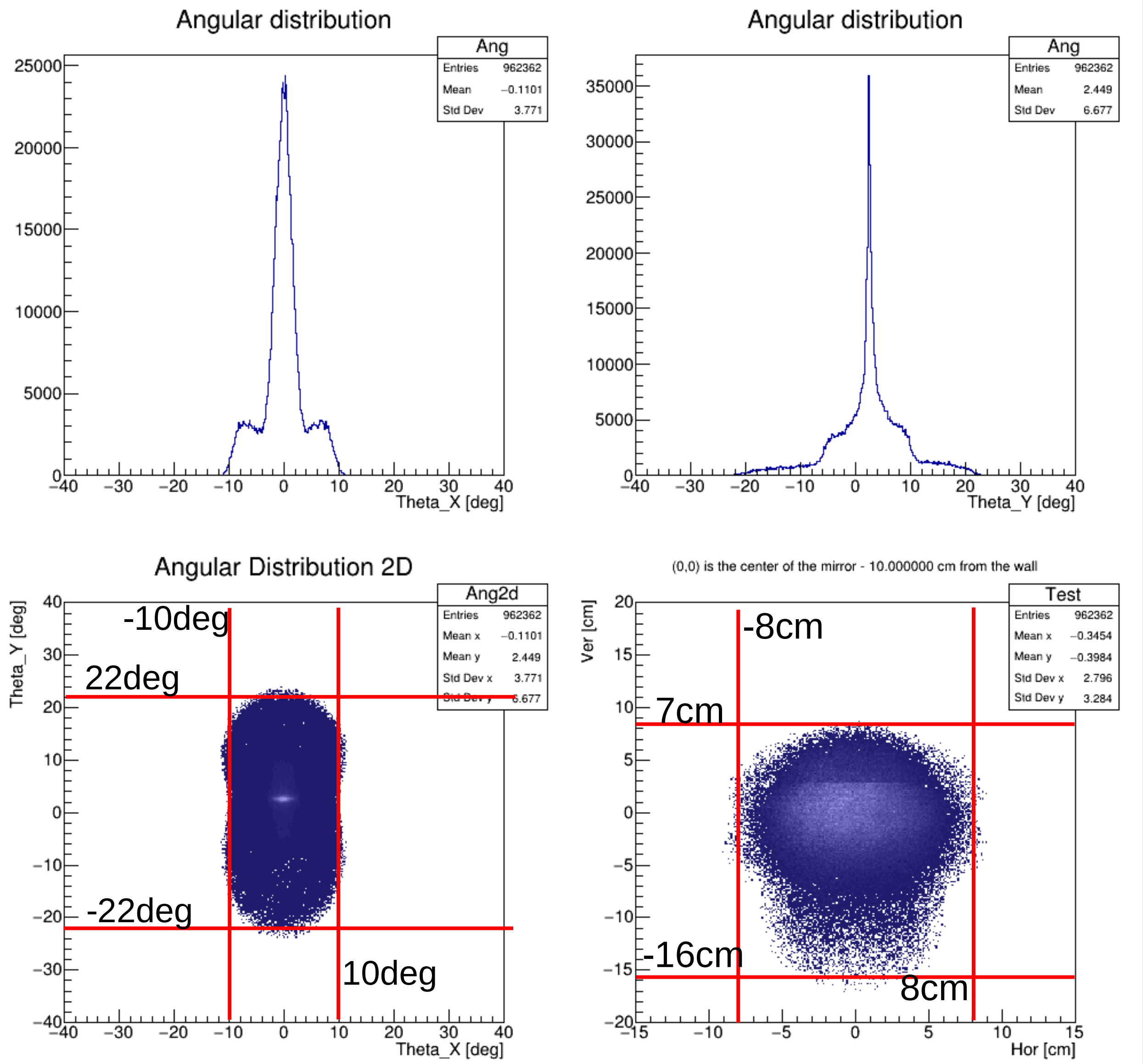
DISTANCE FROM THE SIDE WALL:

5 cm



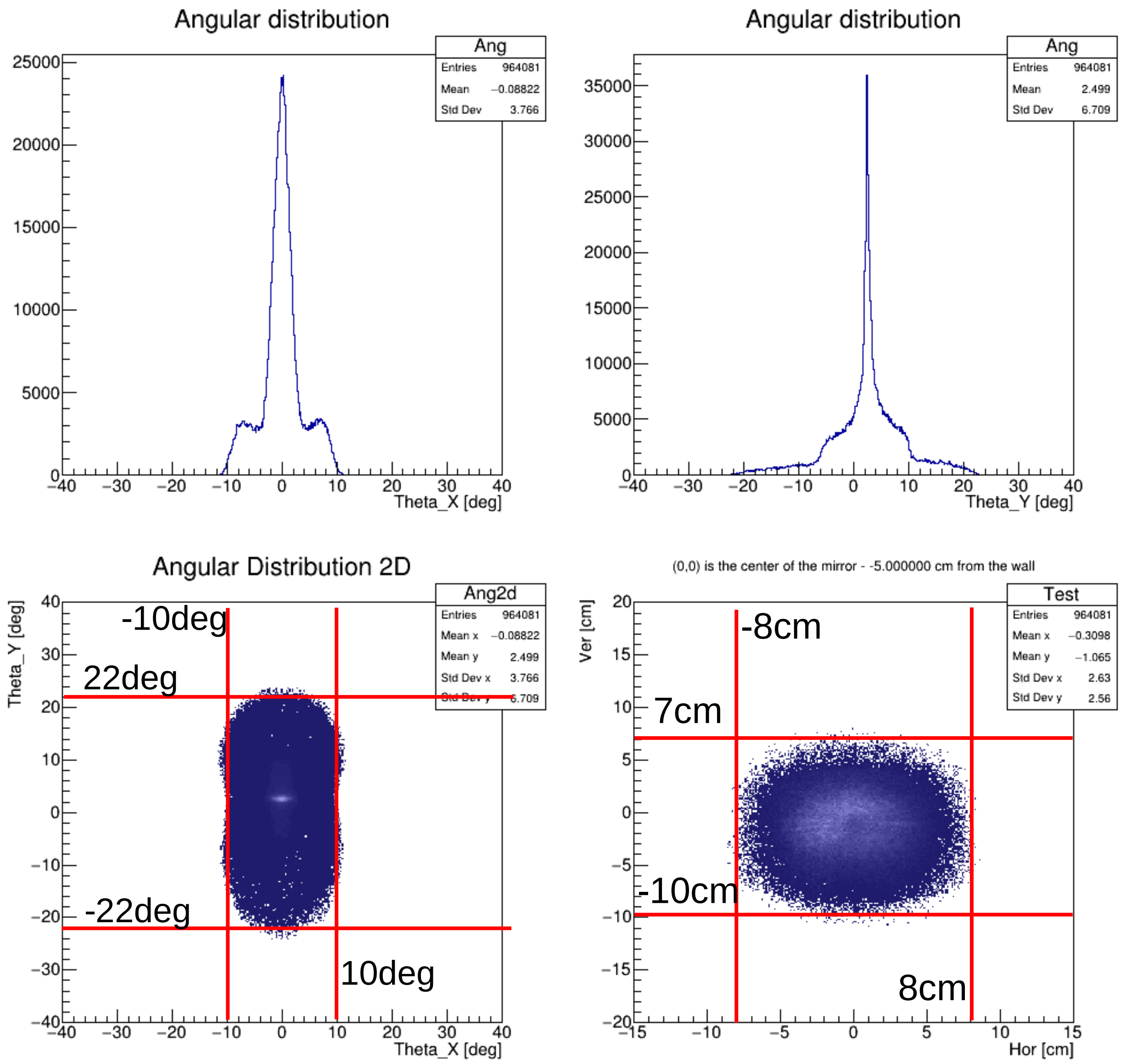
DISTANCE FROM THE SIDE WALL:

10 cm



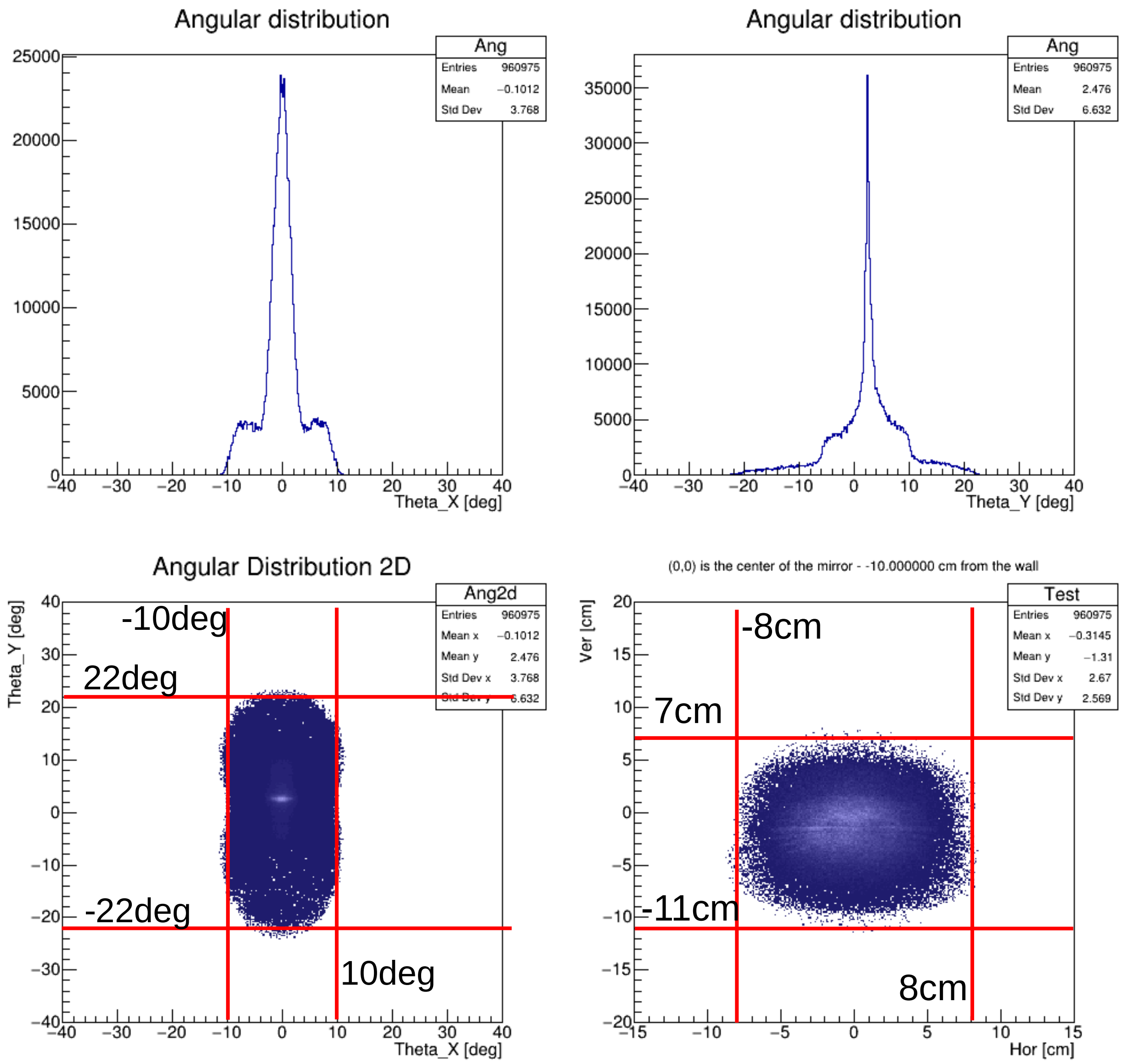
DISTANCE FROM THE SIDE WALL:

-5 cm (inside)



DISTANCE FROM THE SIDE WALL:

-10 cm (inside)



Conclusion

- The analyzer plane position shows the smaller size spot at -5cm
- The angular distribution goes from -22 to 22 deg in the vertical direction, and -10 to 10 deg in the horizontal direction.
- With these numbers, Charles designed a Winston cone (under production)
- Several numbers/distances, could have small variations, but should not affect greatly the performance.