Extensions to the Mokka Detector Simulation Framework

The Hunt for Neutron Background

Adrian Vogel
DESY FLC

What It's All About

Why do simulations with neutrons?

- ILC has the novel problem of beamstrahlung
- beamstrahlung scatters and produces e^+e^- pairs
- pairs crash into the detector and produce neutrons
- neutrons are a source of background, e. g. in the TPC

Why use Mokka?

- the main tool for detector simulations in Europe
- push the transition from Geant 3 (Brahms) to Geant 4

Why modify and extend Mokka?

- mostly inevitable, partly for convenience

Mokka Particle Generator

Guinea Pig is used as the primary particle generator

- simulates beam-beam interaction
- generates (among others) e^+e^- pair particles
- has its own output format ($E, \vec{\beta}, \vec{x}_0$ in raw text)

Extensions to Mokka

- modification of `PrimaryGeneratorAction` to recognise Guinea Pig files with pair particles
- new `GuineaPigInterface` to read them
- new auxiliary command `/generator/path` to set a default path to generator files

Mokka Physics Lists

`PhysicsList` (Mokka built-in)

- doesn’t support neutron production at all

`LCPhys` (dedicated Linear Collider physics list)

- made and maintained by Dennis Wright from SLAC
- old version (included in current Mokka release)
  doesn’t produce neutrons either
- newer versions support neutron production,
  but use poor models for low energies (up to now)
- keep an eye on new releases to come!
  (can be retrieved via CVS from `freehep.org`)

Mokka Physics Lists with Neutrons

`PhysicsListNeutrons` (derived from `PhysicsList`)

- enables gamma-nuclear processes (`EM_GNPhysics`)
- uses high-precision neutron models (`QGSP_HP`)
- features scattering, moderation, interactions with gas

Extensions to Mokka

- add the modified list to the Mokka kernel
- adapt the `PhysicsListFactory` to access it

Mokka Geometry Drivers

Existing drivers described the forward region poorly

- only elementary beam tube, mask, and calorimeters
- quadrupole fields are missing completely
- no support for a crossing angle

New geometry drivers for the forward region

- generalised beam tube (straight or X-shaped)
- more detailed mask (absorbers, support, magnets,. . .)
- simplified calorimeters (homogeneous, non-sensitive)
- better magnet fields (solenoid, DID, quadrupoles)

Further work is being done by A. Elagin and B. Pawlik!

Mokka Geometry Data

Three datasets for the forward region have been implemented as MySQL source files

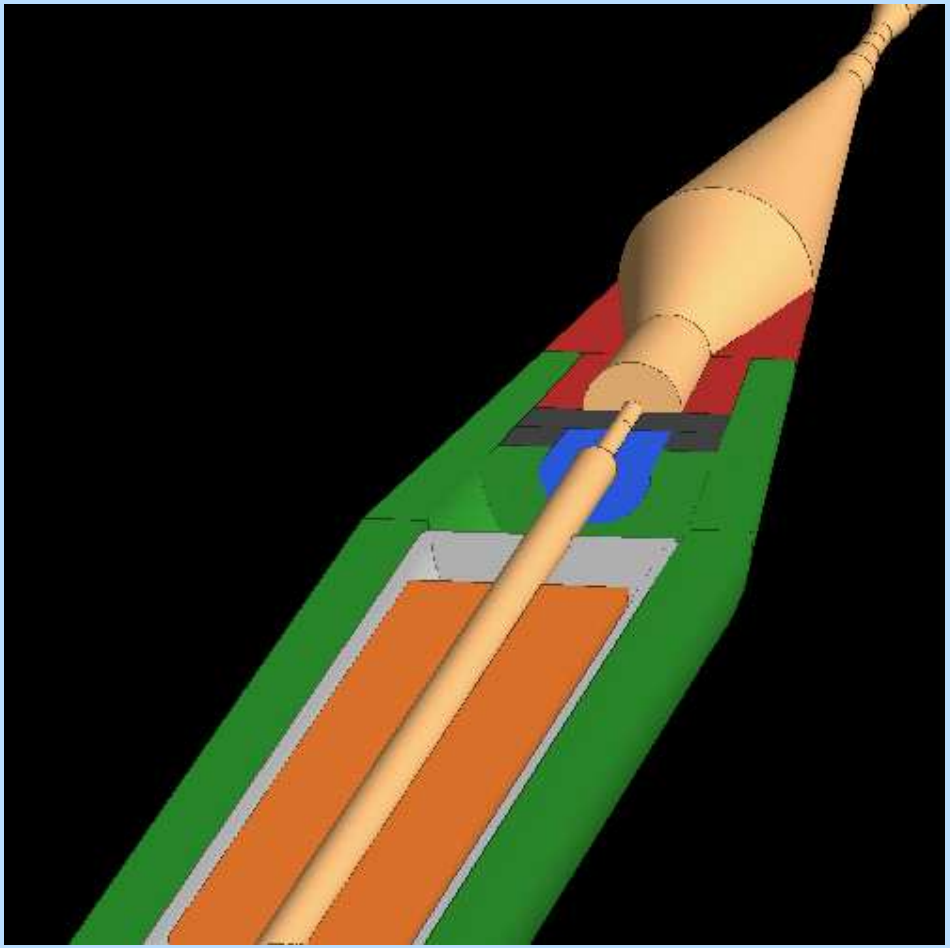
- TDR layout (LAT and LCAL, $L^* = 3.00\,\mathrm{m}$)
- Stahl proposal (LumCal and BeamCal, $L^* = 4.05\,\mathrm{m}$)
- modified Stahl proposal with 20 mrad crossing angle

New geometry model based on “D10” (TDR-like)

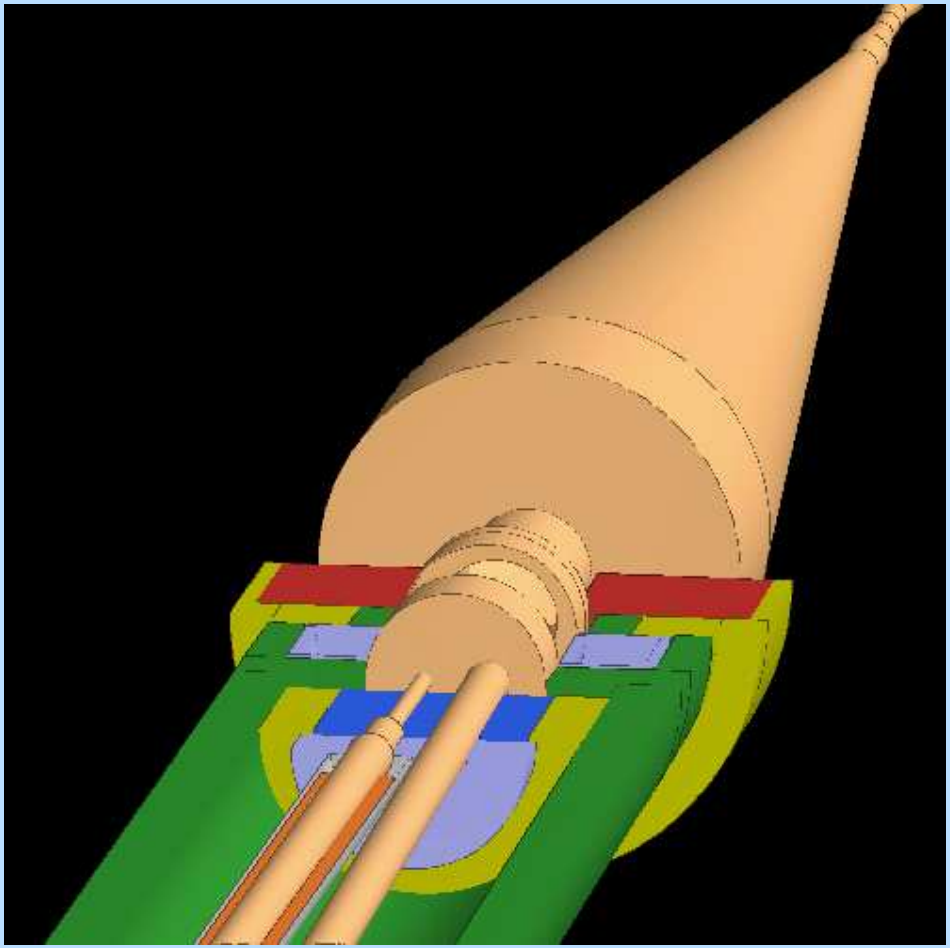
- some subdetectors are replaced
- the rest of “D10” is kept as-is
- needs a new “detector concept” with a bigger world box

Databases contain geometries and visualisation attributes

Mokka Geometry Views



TDR layout



modified Stahl proposal
(20 mrad crossing angle)

Mokka Shared Parameters

The crossing angle is a shared parameter

- three possible detector setups (0, 2, 20 mrad)
- known to beam tube, mask, and fields

Geometry drivers are universal

- pieces can be placed on-axis or up/downstream
- pieces can have two inner holes

Problem with the primary generator

- pairs could be boosted into the right frame
- but: `PrimaryGeneratorAction` cannot see the chosen `CGAGeometryEnvironment`!

Mokka Materials

TPC should be filled with gases other than Argon

- modified TPC driver reads gas from its database

Additional materials in the materials database

- simple compounds: CO_2, CH_4, CF_4, H_2
- gas mixtures: P 5, P 10, TDR gas

Problem with the beam tube vacuum

- pressure and density are way too high! ($10^{-2} \gg 10^{-10}$)
- database fields are limited fixed-point numbers
- suggestion: interpret negative values as logarithms when constructing the materials ($x \mathrel{\widehat{=}} 10^x$ if $x < 0$)

Mokka Plugins: Little Ones

`MaterialPlugin`

- introduces `/Mokka/printMaterialTable` (derived from `G4UIcmdWithoutParameter`)
- prints the current table of `G4Materials`

`MagPlugin`

- introduces `/Mokka/getMagneticField` *x y z* (derived from `G4UIcmdWith3VectorAndUnit`)
- prints the magnetic field vector at a given point

These two need not be registered!
(Everything happens in the constructor)

Mokka Plugins: The Bureaucrat

`LogPlugin`

- writes time of Mokka startup and exit to a log file
- logs start and end time of each run
- keeps track of the number of processed events (useful for remote checking of job progress)

Problem with initialisation

- plugin behaviour should be configurable via `/Mokka/init/userInit...` commands
- but: `UserInit` is not ready at the time the plugin constructor is called! (global object)

Mokka Plugins: The Aesthete

`VisPlugin`

- can project trajectories and hits onto a given plane
- can assign custom colours to certain particles
- can prevent certain particles from being drawn

Implementation problems

- switches Mokka's internal drawing completely off
- redefines Mokka's drawing methods (modified) for steps, trajectories, and hits
- defies principles of object-oriented programming

Could be implemented more cleanly as part of the kernel

Mokka Plugins: The Worker

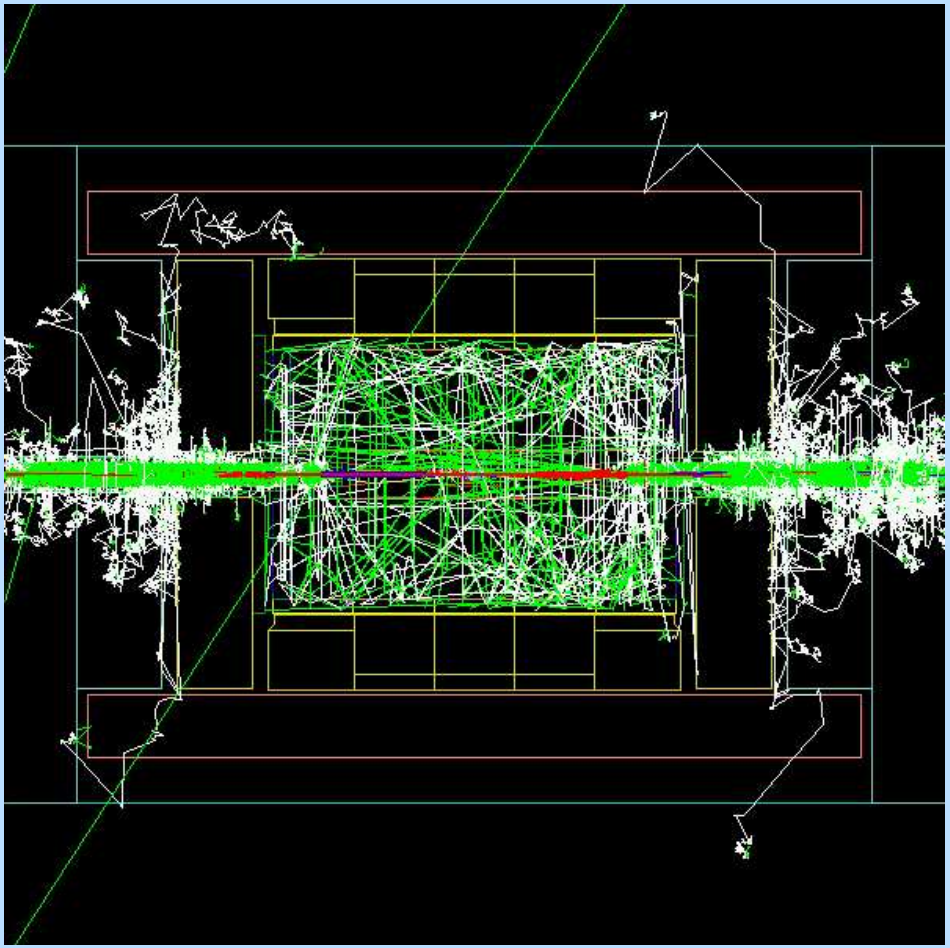
`InvaderPlugin`

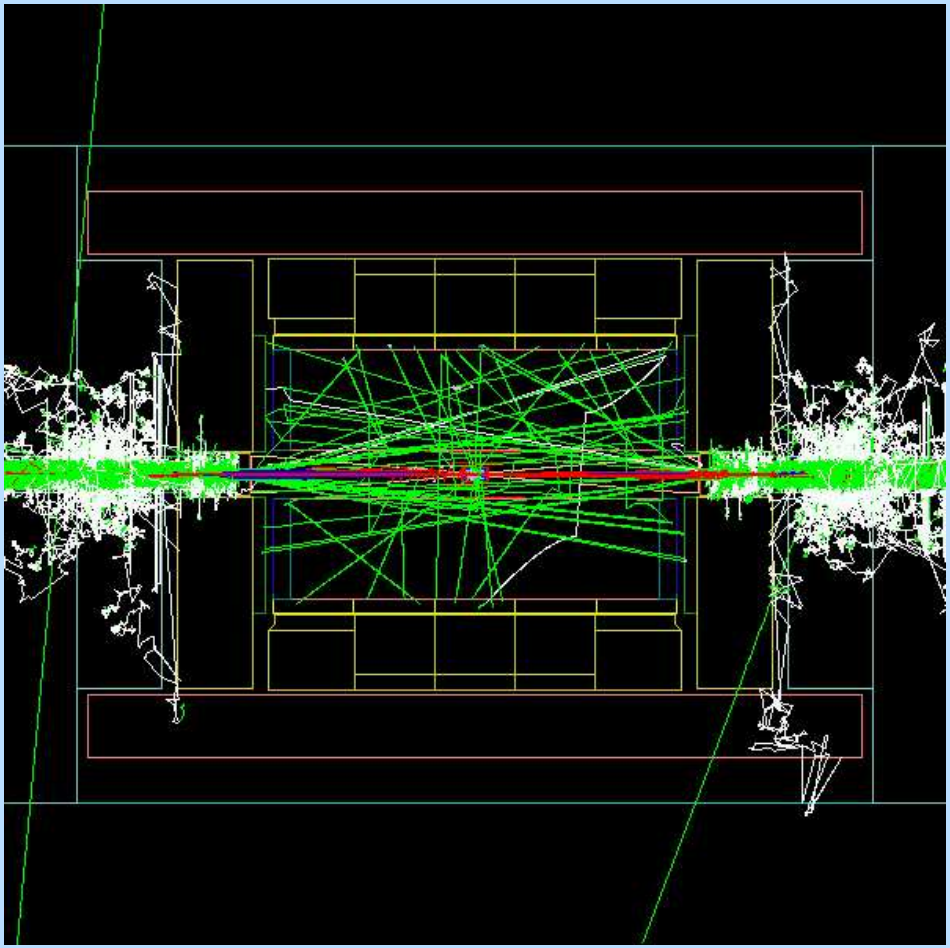
- logs origin and energy of each neutron
- notices every neutron which enters the TPC, logs current energy, counts the entries
- dumps positions of all TPC hits

Implemetation problems

- needs to store information from `G4Track` which Mokka's `Trajectory` doesn't provide
- always needs `SetStoreTrajectory(true)` or would have to do a complete "shadow tracking"

What It Comes Down To





TDR layout

Stahl proposal

Tracks created by 1000 pair particles (≈ 1/100 BX) in *rz*

Some Further Suggestions

Keep your directories clean!

- do not set `G4LIB` and `G4LIBDIR` in the makefiles
- let all dependencies and object code be placed in `$G4WORKDIR/tmp/$G4SYSTEM` by Geant 4

Simplify command line parameters

- let everything be controlled only by the steering file
- do away with the *B*-field tuner altogether

Make LCIO a permanent component of Mokka

- would further strenghten LCIO as a common base
- stop support for non-LCIO output data?