# Geant 4 Simulations of Time Projection Chambers

## *Estimating Beamstrahlung Backgrounds*

Adrian Vogel
DESY FLC

# Background Sources at the ILC

$e^+e^-$ pairs are a main source of background

- beams have to be focused very strongly ($\sigma_y$ = 5 nm)
- beam-beam interaction creates beamstrahlung
- beamstrahlung photons scatter to $e^+e^-$ ($10^5$ / BX)
- $e^+e^-$ smash into forward calorimeters (BeamCal) and magnets of the beam delivery / extraction line
- lots of photons, neutrons, and charged particles

Other sources are supposed to be negligible (beam dump, synchrotron radiation,. . . ) or have to be studied in further detail (beam halo, extraction line losses)

# Problems with Background

Inner silicon trackers (VXD, SIT, FTD)

- hits from charged particles (direct / indirect)
- silicon bulk damage from neutron fluence

Main gaseous tracker (TPC)

- Compton scattering, photon conversion
- neutron-proton collisions (recoil) with hydrogen
- additional primary ionisation, field distortions

Calorimeters (ECAL, HCAL)

- more photons from nuclear reactions, neutron capture
- random low-energy hits, radiation damage (?)

# Simulation Tools – Guinea Pig (D. Schulte)

Input

- set of beam parameters ($E$, $\vec{\sigma}$, $\vec{\beta}$, $Q$, ...)

Output

- particles in the disrupted beams
- beamstrahlung photons
- $e^+e^-$ pair particles
- hadronic scattering products (“minijets”)

Existing simulation data

- TESLA beam parameters (500 GeV, 800 GeV)
- various ILC parameter sets (500 GeV, 1 TeV)

# Simulation Tools – Mokka

Mokka is a full detector simulation

- based on the Geant 4 framework
- written in C++, modular design
- main development at LLR, France
- now: contributions from many different users
- successor of Brahms (GEANT3, Fortran)

Mokka uses LCIO as a persistency framework

- predefined storage classes (particle, track, hit,. . . )
- lightweight and robust, cross-platform design
- supported by large parts of the ILC community

# Mokka – Analysis and Reconstruction

Mokka writes out “raw” hits

- idea: simulation takes much longer than digitisation, reconstruction, and analysis (presumably)
- write out simulation results as early as possible
- apply digitisation afterwards (with different settings)

Mokka output is processed by Marlin

- modular analysis and reconstruction framework
- set of processors read and write data collections from and to LCIO files
- also used for real data from prototypes

# Tracking – Behaviour of Geant 4

Geant 4 transports particles step by step

Step length is determined by the minimum of:

- the distance to the next physical volume boundary
- the free path to the next discrete physics process (for all applicable processes, randomised)
- the limit of the step length

Discrete processes (e. g. decay) cause a step to end

Continuous processes (e. g. energy loss by ionisation) are applied after a step has ended for some other reason

# Tracking – Problems with the TPC

The fundamental process in a TPC is ionisation

- TPC contains low-density material (gas)
- discrete processes are rare, steps are long
- small number of rather large energy deposits

A real TPC is read out by small anode pads

- large number of small energy deposits
- needed for tracking and $dE/dx$

Steps in the simulation need to be broken down

- introduction of artificial volume boundaries
- limitation of the step length

# Option 1 – Layers

Implementation

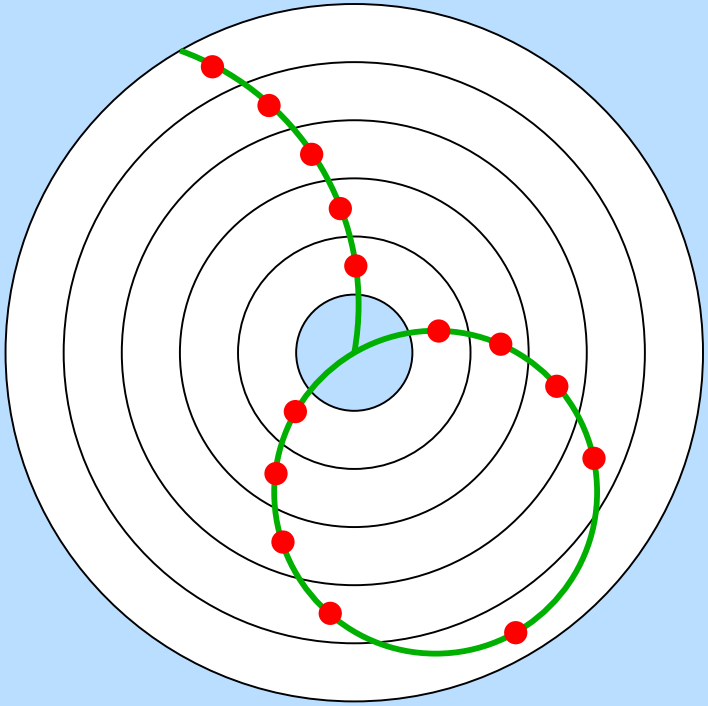
- segmentation in $\rho$
- divide the TPC volume into 200 “layers” of gas
- sum up the energy deposits in each layer $\longrightarrow$ hits

Pros

- simple and fast
- suitable for high-$p_\mathrm{t}$ tracks

Cons

- information loss for low-$p_\mathrm{t}$ tracks
- hard-coded readout geometry



# Option 2 – Voxels

Implementation

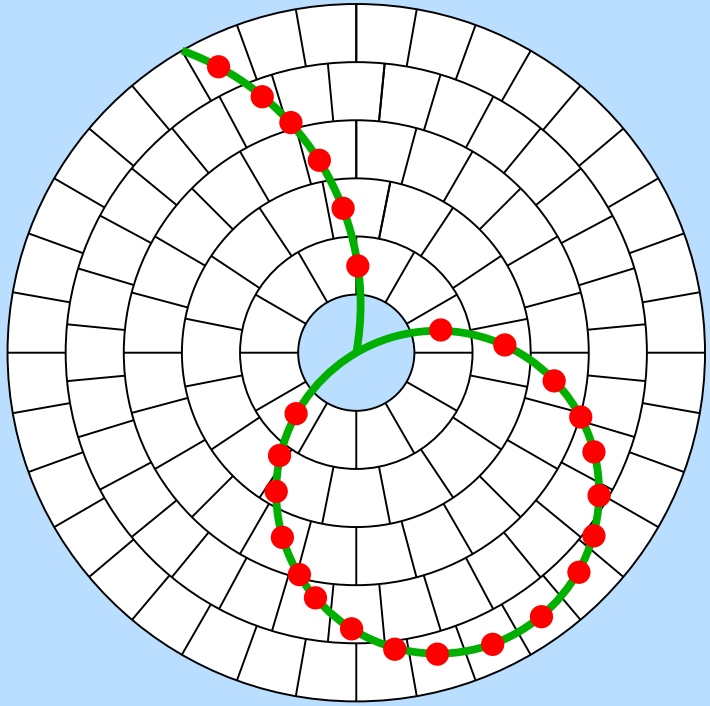
- segmentation in $\rho$, $\varphi$, and $z$
- divide the TPC volume into layers, wedges, and disks
- sum up the energy deposits in each voxel $\rightarrow$ hits

Pros

- realistic information for all tracks

Cons

- slow navigation
- hard-coded readout geometry
- many hits, large output files



# Option 3 – Step Limits

Implementation

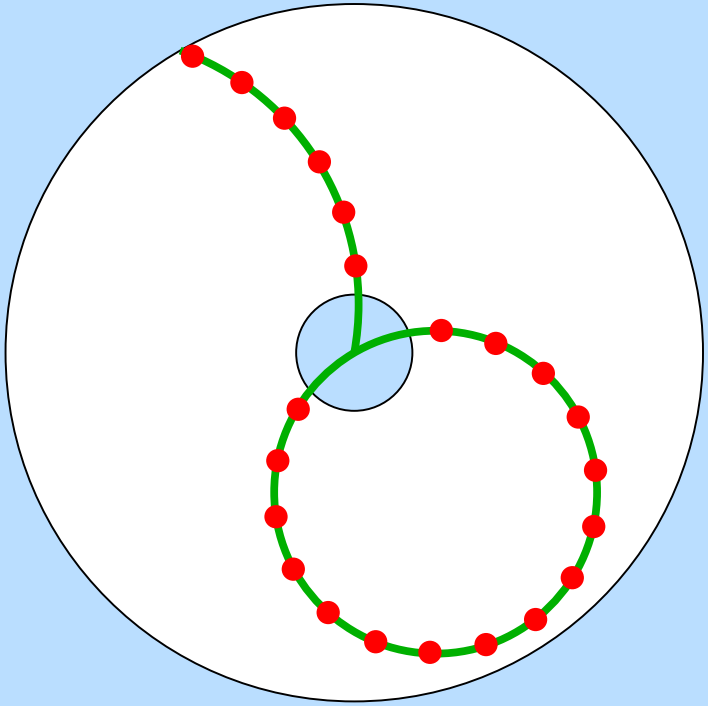
- segmentation in the direction of flight
- assign maximum step length to the TPC volume
- write out energy deposit for each step $\rightarrow$ hits

Pros

- realistic information for all tracks
- simple and fast

Cons

- binning effects possible
- very large output files possible



# Step Limits in Geant 4

In the physics description

- implemented as a “pseudo-process” `G4StepLimiter`
- not included in the built-in physics lists of Geant 4
- added to the selected physics list in Mokka at runtime (for all long-lived charged particles)

In the geometry description

- attach an object of the class `G4UserLimits` to a logical volume (the TPC gas, in this case)

# Cuts in the TPC

Minimum energy deposit of a step

- need at least $\Delta E = 32\,\mathrm{eV}$ for a hit (Argon ionisation)

Minimum kinetic energy of a track

- steering parameter (Mokka default is 10 MeV)
- particles with $E < 10\,\mathrm{MeV}$ curl on one pad
  (So what? They're nevertheless there!)
- what about delta electrons and background hits?
- don't make the simulation too friendly!

`G4UserSpecialCuts` $(\ell_{\mathrm{max}}, t_{\mathrm{max}}, E_{\mathrm{min}}, R_{\mathrm{min}})$

- available, but currently not used (and not needed)

# Implementations of the TPC in Mokka

Option 1 – Layers

- available since the first Mokka release
- only minor modifications over the years
- used in all currently predefined geometry models

Option 2 – Voxels

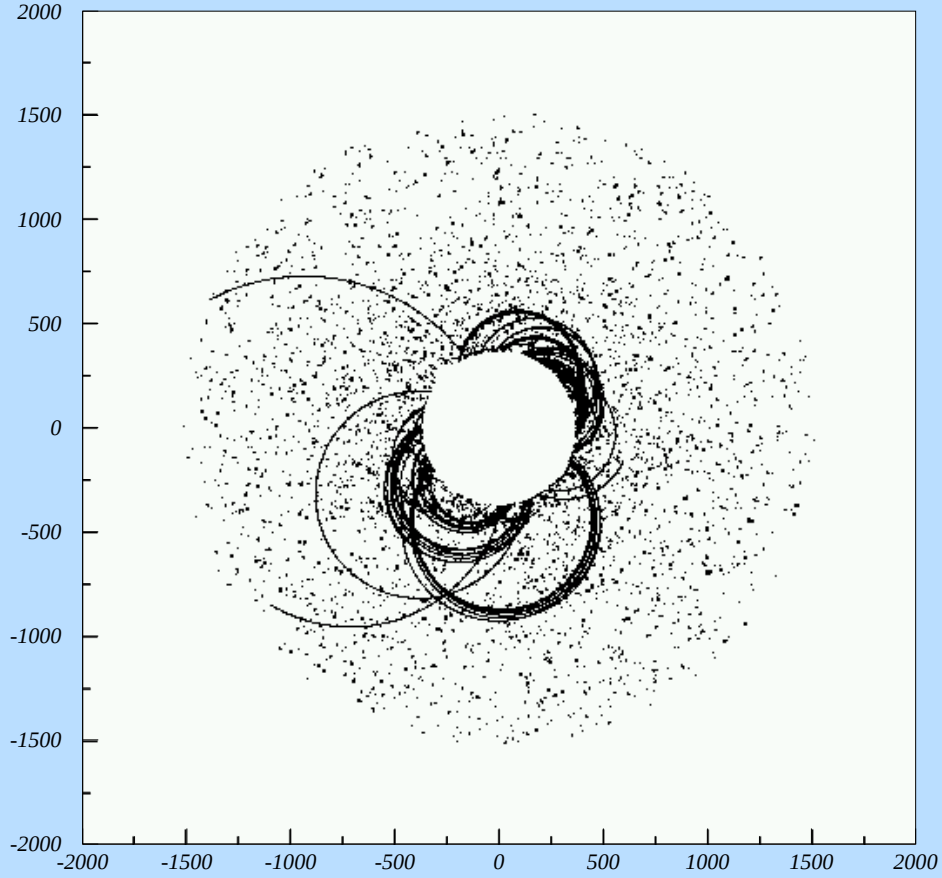
- proof of principle: it works (with $\mathcal{O}(10^9)$ voxels)
- not released to the public (significantly slower)

Option 3 – Step Limits

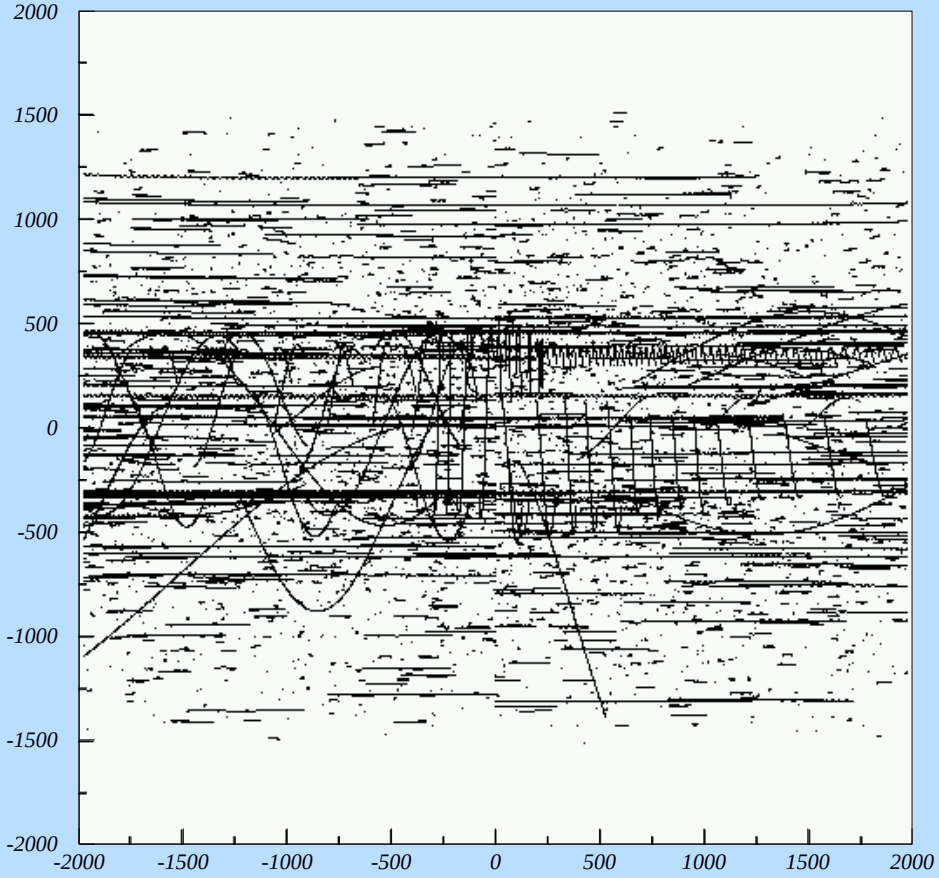
- available since Mokka 06-00 (drivers `tpc04` and up)
- used in the latest revisions of the LDC detector models

# TPC Hits – “Salt and Pepper”

Mokka hits in the TPC (overlay of 100 BX)



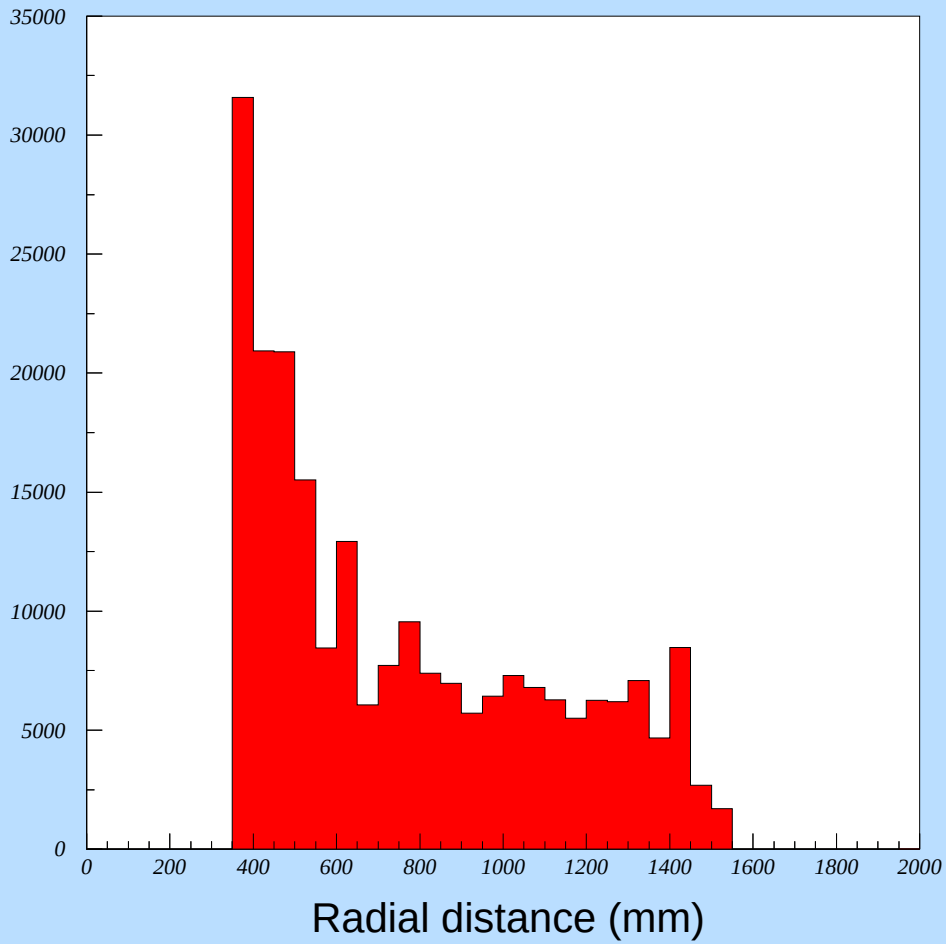

Front view



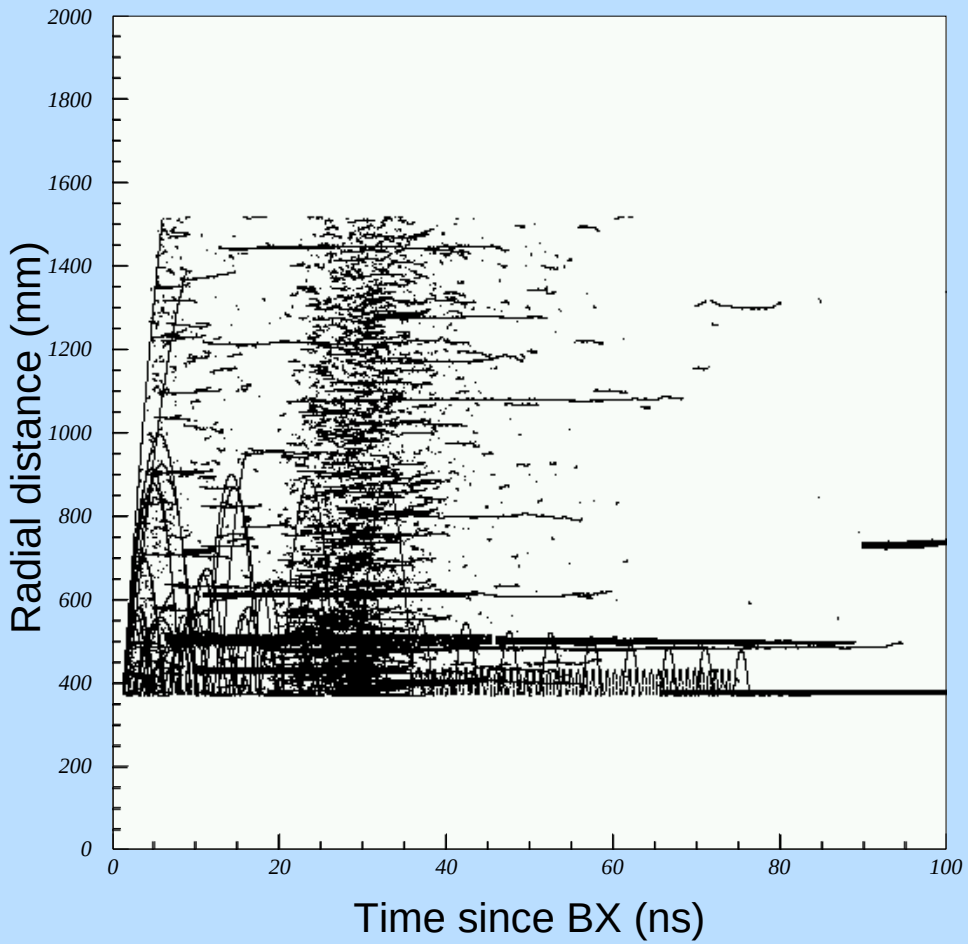

Side view

# TPC Hits – Distributions

Mokka hits in the TPC (overlay of 100 BX)




Radial distribution




Time structure

# Analysis – Occupancy

Transformation of “Mokka hits” to a realistic occupancy

- set up a map of discrete voxels
- fill in each energy deposit (with charge sharing)
- calculate the fraction of occupied voxels

Some simplified assumptions (for the beginning)

- perfect charge sharing, no diffusion effects yet
- no gain fluctuations, no electronics effects
- drift with endless “time loop” (modulus operation)
- scalable dependency on the voxel size

Work is currently (= this week) in progress!

# Estimation of Errors

Generators

- Guinea-Pig and others agree on the level of 10%
- pairs are supposed to be the main background source

Simulation

- Mokka is based on Geant 4 – the HEP standard
- full simulation contains e. g. backscattering
- detector geometries and magnetic fields can have subtle effects with major impact (also seen in Brahms)
- hadronic physics and esp. neutron modelling is always difficult! (compare different models)

We aim for a safety factor of 10 in order to rest easy

# Questions and Tasks

- Which design decisions affect the TPC, and how?
- Can we use a quencher which contains hydrogen?
- How large will the occupancy be at a given time? (with superposition of 160 bunch crossings)
- Provide a “background library” with ready-to-use events to be superimposed on “real” physics for analyses
- Set up a consistent software toolkit for the TPC: digitisation – tracking – reconstruction – analysis
- Will the background signals have an impact on pattern recognition, efficiencies, resolutions?