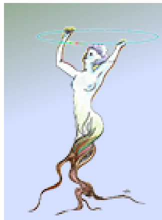
KDetSim
A root based 3D simulation tool for semiconductor detectors

Gregor Kramberger

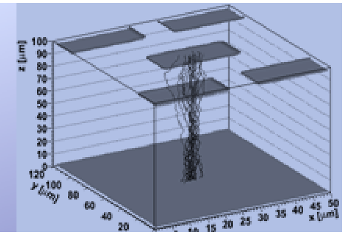
Jožef Stefan Institute, Ljubljana, Slovenia



KDetSim

a simple way to simulate detectors

A ROOT based library for calculation of fields and signals in semiconductor detectors



KDetSim | Examples | Manual/Tutorial | Downloads | Class Index | Presentations

KDetSim

KDetSim introduction

KDetSim is a shared library (.dll under Windows .sl under Linux) which is dedicated to solving Poisson/Laplace equation in 2D and 3D and monte-carlo simulation of the charge transport inside semiconductor detectors. It is based on ROOT in the sense that it rellys heavily on its class libraries for all aspects of operation (visualization, IO , user interface...). The class library can be used to built executable code, but the primary use is within the root interpreter (CINT), where programs are executed in the form of macros.

Manual/Tutorial

An introduction and explanation of useage can be found in the manual, which is in fact a tutorial. Different functionalities of the classes are demonstrated on several examples.

Examples

A repository of several examples which can server as a starting point for the simulations.

Downloads

The distribution package with instructions to install on different OS platforms.

Class Index

A complete list of all classes defined in KDetSim can be found at above link. A complete hierarchy graph of all classes (Class Hierarchy), showing each class's base and derived classes can be found here and a complete list of data types here.

Who are we?

Gregor Kramberger, Jozef Stefan Institute, Ljubljana

» Last changed: 2012-10-23 15:06 » Last generated: 2012-10-23 15:06
This page has been automatically generated. For comments or suggestions regarding the documentation please send a mail to KDetSim developers.

Outline

- Motivation
- Calculation of electric and weighting field
- Simulation of charge transport
- Structure of simulation library
- Examples of simulations
  - Pad detector
  - Strip detector, Edge-TCT
  - 3D detector
  - Pixel detector
  - LGAD
- Conclusions

Motivation

- Why? TCAD, Synopsis are excellent, but:
  - Drift simulation as solution of differential equations
    - Very demanding in terms of CPU and time (4D problem)
    - without fast step-wise approach to drift simulation is the Monte Carlo approach for studying detector properties crucial to HEP (charge sharing, Lorentz angle …) not possible
  - Not suited for large multi-electrode system.
  - Not easy to include data from other packages: GEANT …
  - Not so flexible as custom made code.

- The goal is a fast and easy root based package for simulation of signal in semiconductor detectors:
  - root interface allows for an easy and standard GUI/IO interface, well integrated with other HEP tools (GEANT…)
  - C++ code in forms of class library is very fast and kind to the computer resources
  - should compile on most OS (Mac, Linux ,Windows, Unix)
  - should be easily upgradable: e.g. adding new mobility model, impact ionization coefficients …
  - extensively used in TCT simulations

History and how to get it …

- Basic components of the simulation package done during my PhD. thesis. Over the years the package grew as the knowledge and requirements progressed (e.g. including multiplication, magnetic field, full 3D simulation …)
- Several publications were published using the software (by far not all…)
  - G. Kramberger . et al., Signals in non-irradiated and irradiated single sided silicon detectors, NIM A457 (2001) 550.
  - G. Kramberger, PhD. Thesis, University of Ljubljana, 2001.
  - G. Kramberger et al., Influence of trapping on silion strip detector design and performance, IEEE trans. nucl. sci., 2002, vol. 49(4), p. 1717 (PDF)
  - By: Mikuz, M; Studen, A; Cindro, V; et al., Timing in thick silicon detectors for a Compton camera, IEEE TRANSACTIONS ON NUCLEAR SCIENCE Volume: 49 Issue: 5 Pages: 2549-2557 Part: 2 Published: OCT 2002
  - D. Contarato, PhD Thesis, Universisty of Hamburg, 2005.
  - G. Kramberger and D. Contarato, Simulation of signal in irradiated silicon pixel detectors, NIMA 515 (2004)
  - G. Kramberger and D. Contarato, How to achieve highest charge collection efficiency in heavily irradiated position-sensitive silicon detector , NIM A 560 (2006) 98.
  - Kramberger, G.; Cindro, V.; Mandic, I.; et al. Modeling of electric field in silicon micro-strip detectors irradiated with neutrons and pions , JOURNAL OF INSTRUMENTATION Volume: 9 Article Number: P10016 Published: OCT 2014.
  - Mandic, Igor; Cindro, Vladimir; Gorisek, Andrej; et al. "TCT measurements with slim edge strip detectors" NUCLEAR INSTRUMENTS & METHODS IN PHYSICS RESEARCH SECTION A-ACCELERATORS SPECTROMETERS DETECTORS AND ASSOCIATED EQUIPMENT Volume: 751 Pages: 41-47 Published: JUL 1 2014 .

Link to the software:

http://www-f9.ijs.si/~gregor/KDetSim/

Basics – Calculation of Electric Field

- The package doesn't solve continuity/GR equations in silicon, **but takes *N***

$$\frac{\partial n}{\partial t} = \mu_e n \nabla \vec{E} + D_e \nabla^2 n + G_n - R_n$$
$$\frac{\partial p}{\partial t} = -\mu_h p \nabla \vec{E} + D_h \nabla^2 p + G_p - R_p$$
$$N_{eff} = p - n + N_D - N_A + \sum_{deep} Q_t \quad ,$$

Electric field

$$\nabla \vec{D} = e_0 N_{eff}(\vec{r}) \quad , \qquad \vec{D} = \varepsilon \varepsilon_0 \vec{E} \quad , \quad \vec{E} = -\nabla U(\vec{r}) \quad ,$$

$$\boxed{\nabla(\varepsilon(\vec{r}) \nabla U(\vec{r})) = -\frac{e_0 N_{eff}(\vec{r})}{\varepsilon_0}}$$

Boundary conditons:

$$\frac{\partial U}{\partial x} = 0 \quad , \frac{\partial U}{\partial y} = 0, \quad \frac{\partial U}{\partial z} = 0 \quad \text{at borders of simulated volume}$$

$U =$ voltage at electrodes

Weigthing field

$$\Delta U_w(\vec{r}) = 0 \qquad , \quad \vec{E}_w = \nabla U_w(\vec{r}) \quad ,$$

$$\frac{\partial U_w}{\partial x} = 0 \quad , \frac{\partial U_w}{\partial y} = 0, \quad \frac{\partial U_w}{\partial z} = 0 \quad \text{at borders of simulated volume}$$

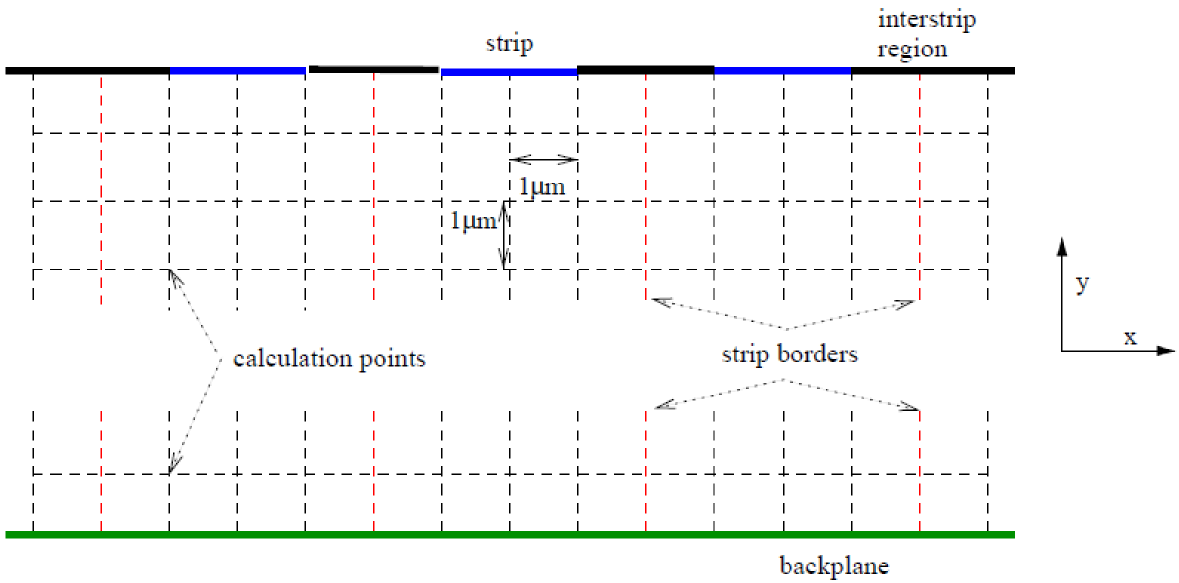
$U_w = 1$ at readout electrode and $U_w = 0$ at all other electrodes

Note that the boundary conditions are crucial and often the reason for miss-interpretation of the results.

Reflective boundary conditions at the detector surface are most common – also in TCAD and Synopsis, but …

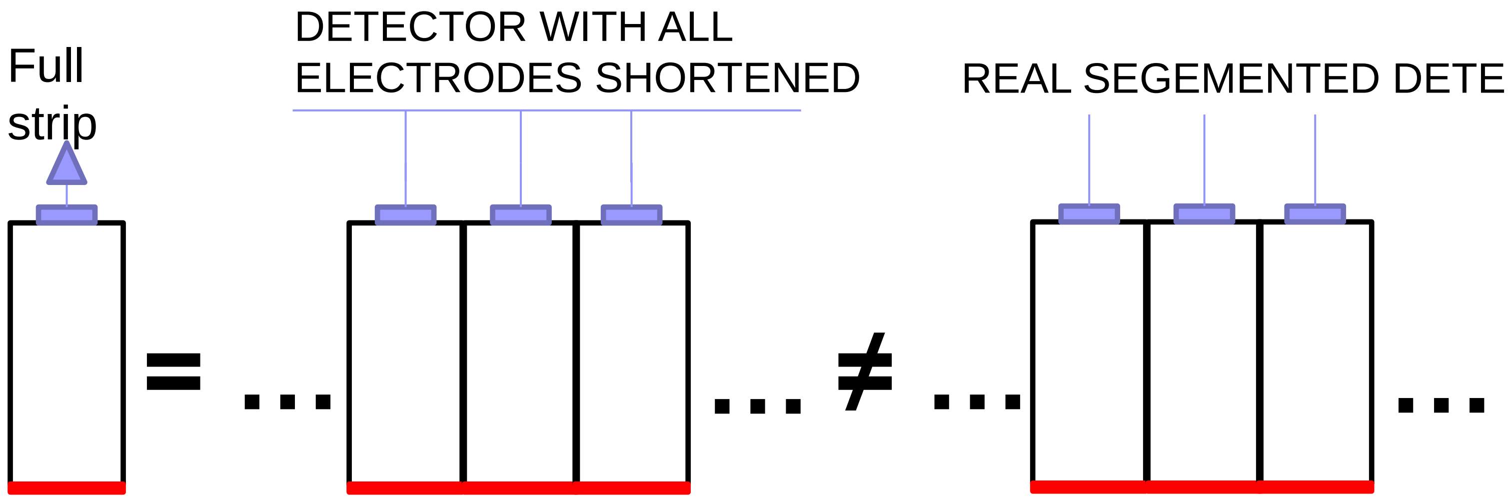
Basics – Calculation of Electric Field

The partial differential equation is solved numerically by using finite difference equation on the mesh (FEM approach). An example of the mesh in 2D is shown in figure below. The mesh can be defined in 3D with complex electrode arrangements/shapes (see examples). The mesh should be orthogonal but doesn't have to be equidistant.



The differential equation translates to solving the system of equations for U, where every node represents an equation. The boundary conditions are essential as they determine the solution of the equations. The system of equations is solved by inverting the matrix When simulating a 3D structure (Pixel detector) the system results in large number of equations (Nx*Ny*Nz). The matrix which should be inverted is sparse which significantly speeds up its inverse, so 10^6 node system is solved in the time scale of minutes on Sandy-bridge Core I7 portable CPU .

Basics – Calculation of Weighting Field



All neighboring electrodes also with U_w=1 –> all strips shortened

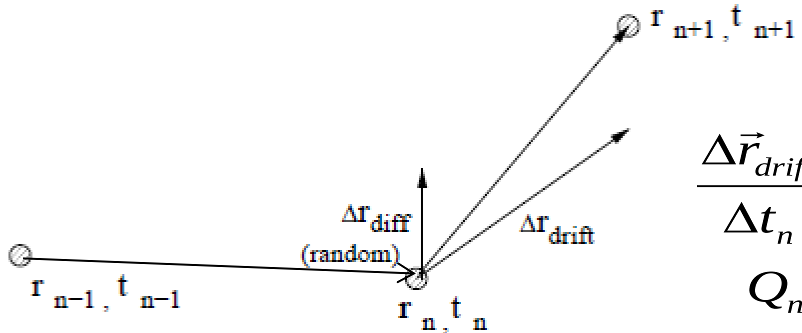
All neighboring electrodes at U_w=0 –> individual channels

Unlike for electric field where for the symmetry reasons only a half strip can be used to calculate the field one should simulate a much larger section for the weighting field. Often not done in TCAD simulations.

A lot of effects in irradiated silicon detectors – such as e.g. "trapping induced charge sharing" can not be simulated without proper weighting field.

Basics – Simulation of charge transport

- Charge/current induced by a point-like charge q



$$\vec{r}_{n+1} - \vec{r}_n = \Delta\vec{r}_{drift} + \Delta\vec{r}_{diff}$$

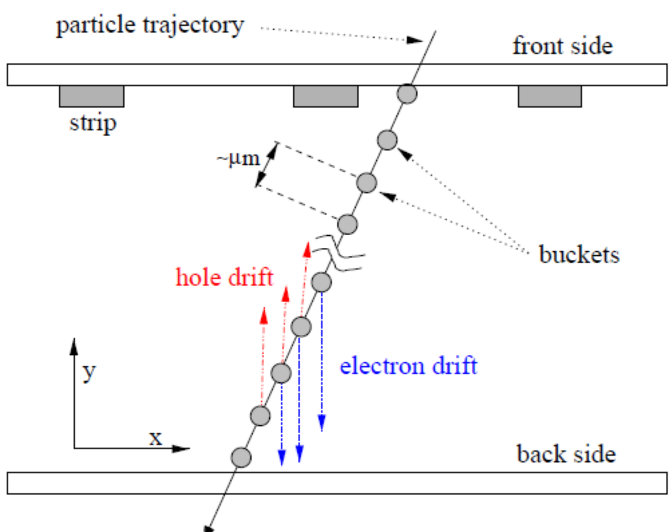
$$\vec{F} = \vec{E} + \mu_{H\,e,h}\,\vec{E} \times \vec{B}$$

$$\frac{\Delta\vec{r}_{drift}}{\Delta t_n} = \mu \cdot \vec{F}(\vec{r}_n(t_n)) \quad , \quad \Delta\vec{r}_{diff} = \vec{G}aus(\sqrt{2D\Delta t_n})$$

$$Q_{n+1} - Q_n = q \cdot \underbrace{P(t_n, \tau_{eff})}_{P=0 \text{ or } P=1 \text{ - trapping}} \cdot \underbrace{[U_w(\vec{r}_{n+1}) - U_w(\vec{r}_n)]}_{\text{difference in } U_w}$$

$$I = \frac{Q_{n+1} - Q_n}{t_{n+1} - t_n}$$

$$I(t) = \sum_{buckets} I_e(t) + I_h(t)$$

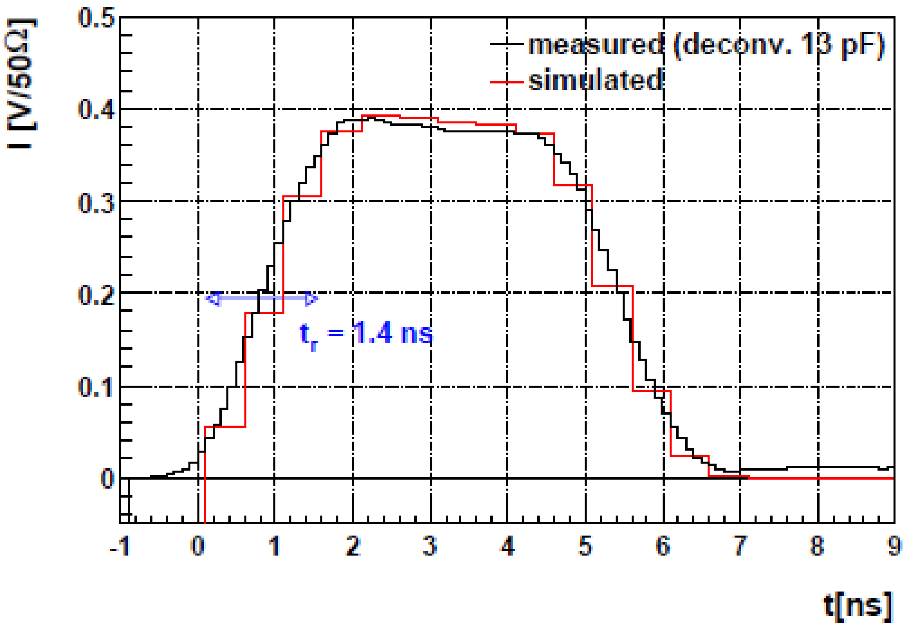


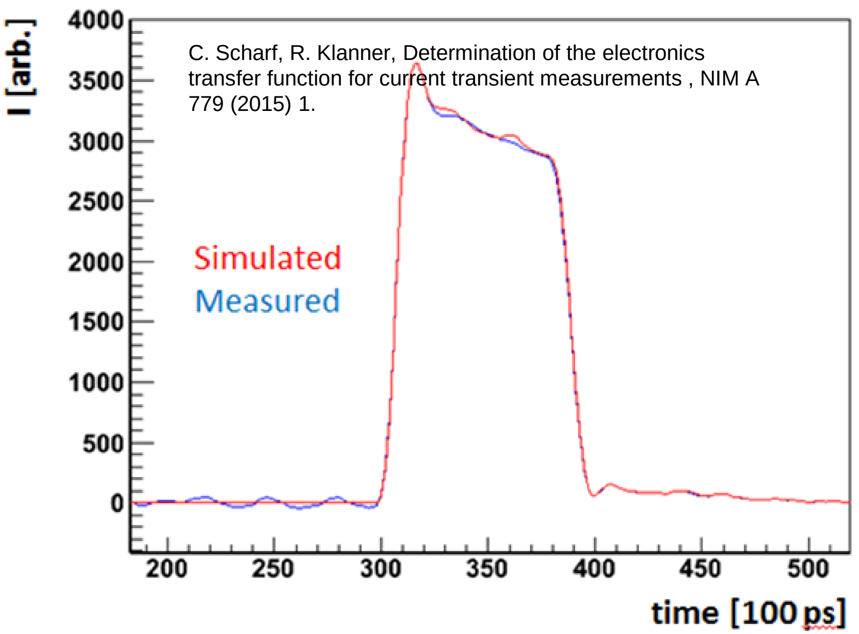
Different models are already included (Gaussian beam, exponential attenuation of beam, minimum ionizing particle), but you can easily make your own function which distributes buckets q around the sensor

Basics – electronics processing

- Basic electronics models are included:
  - preamp
  - CR, RC filtering / shaping
- FFT – is also included to convolute simulated signals with transfer function

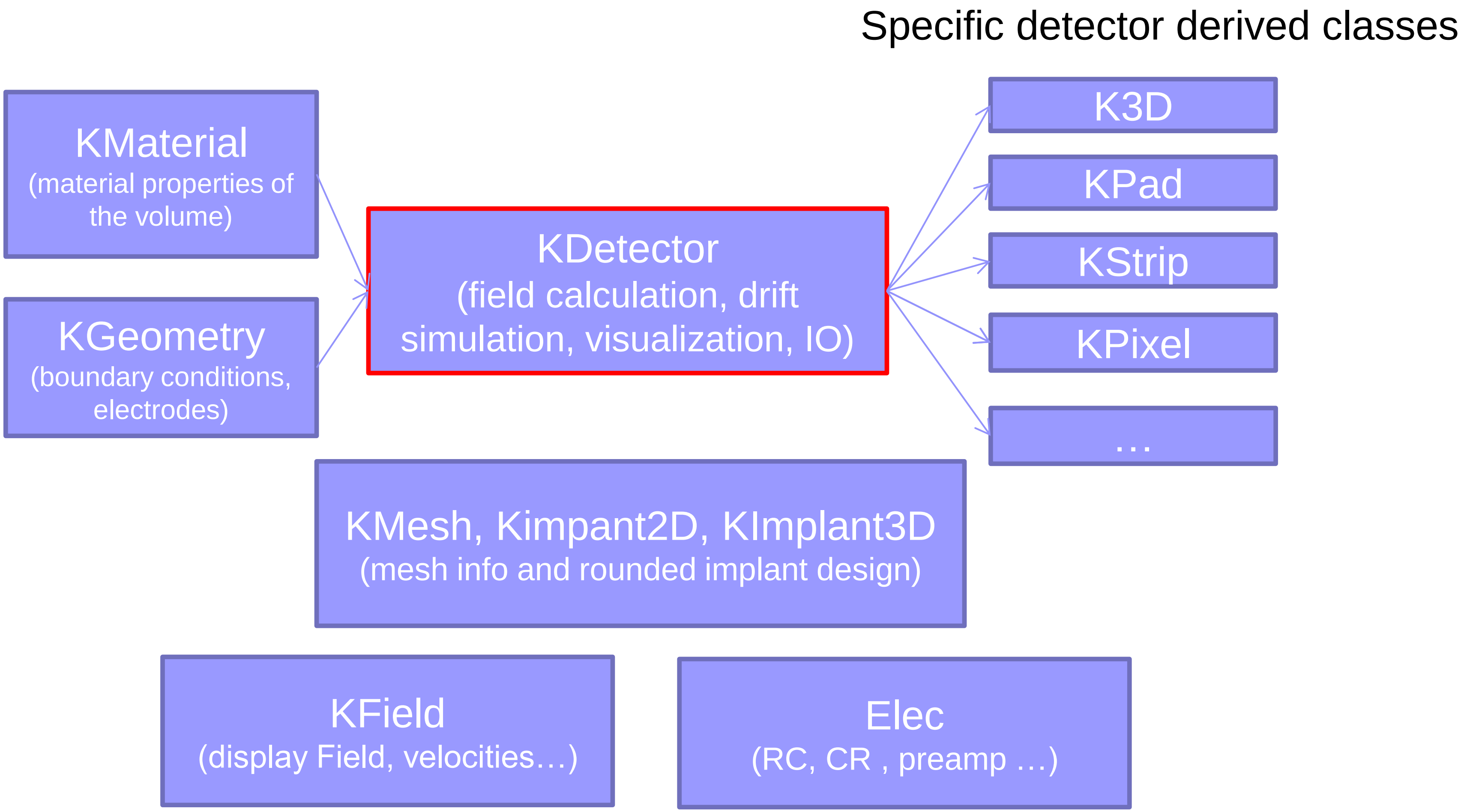
FZ-n diode , 15 kΩcm, V=100 V
Simulation used to extract transfer function





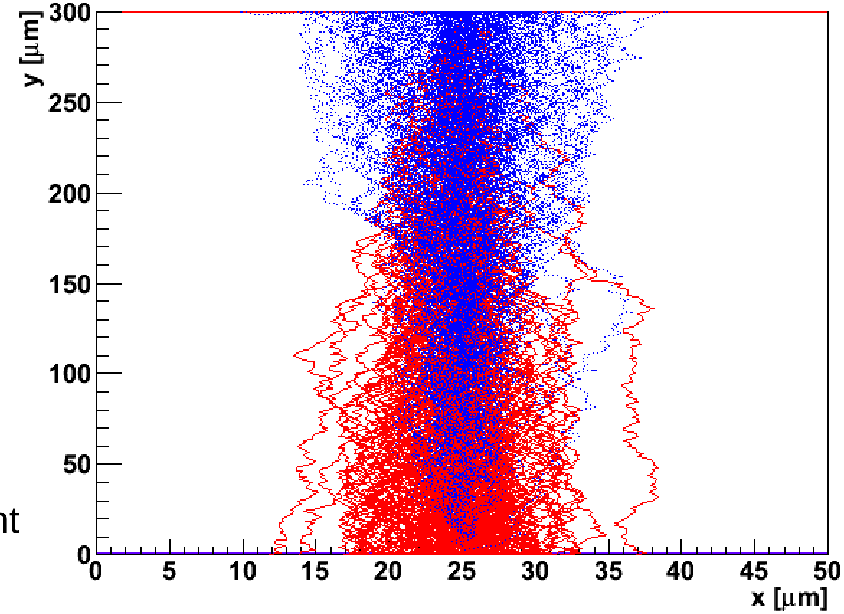
Structure of the simulation library

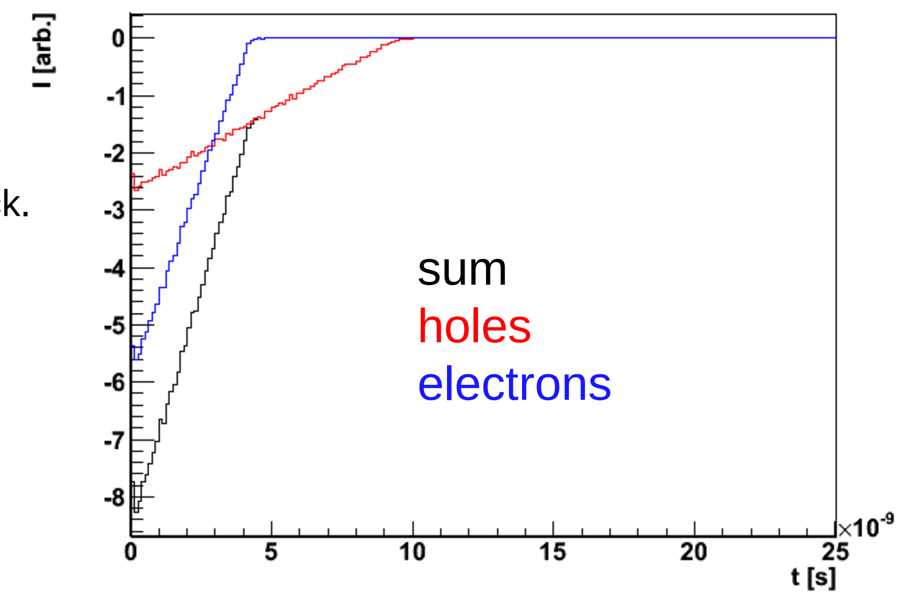
The library is a single .dll, .sl which is loaded in the root framework



Examples of simulation – pad detectors

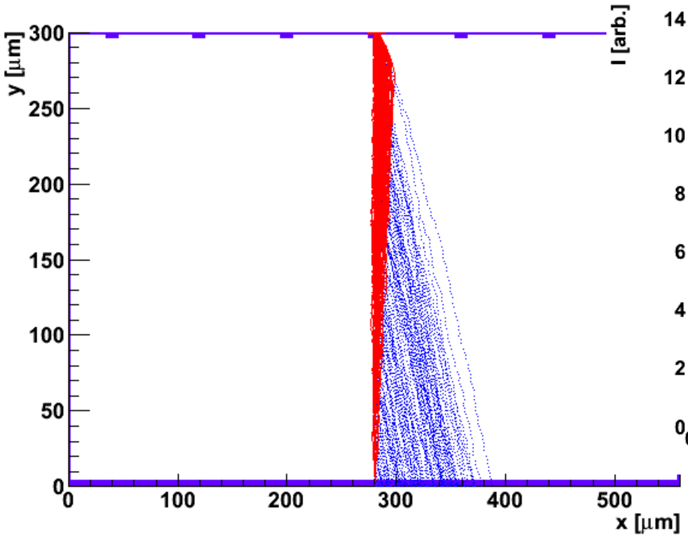
```
{
  TF1 *neff=new TF1("neff","[0]",0,1000);
  neff->SetParameter(0,1);    // set Neff [1e12 cm-3]
  KPad det(50,300);           // dimensions of the sample
  det->Neff=neff;
  det->Voltage=-200;          // Set voltage
  det->SetUpVolume(1);        // Setup electrodes
  det->SetUpElectrodes();

  TCanvas c1;
  det->SetEntryPoint(25,299.9,0.5); // define track entry point
  det->SetExitPoint(25,1.,0.5);         // define track exit point
  det->Temperature=253;                 // set temperature
  det->diff=1;                          // switch on diffusion
  det->ShowMipIR(200);                  // draw drift paths

  TCanvas c2;
  det->MipIR(200,1);                    // simulate mip 200 buck.
  det->sum->Draw();                     // draw current
  det->pos->Draw("SAME");
  det->neg->Draw("SAME");
}
```

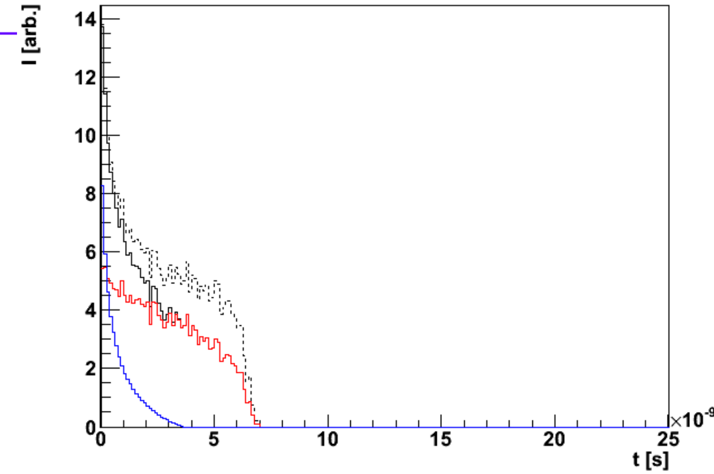


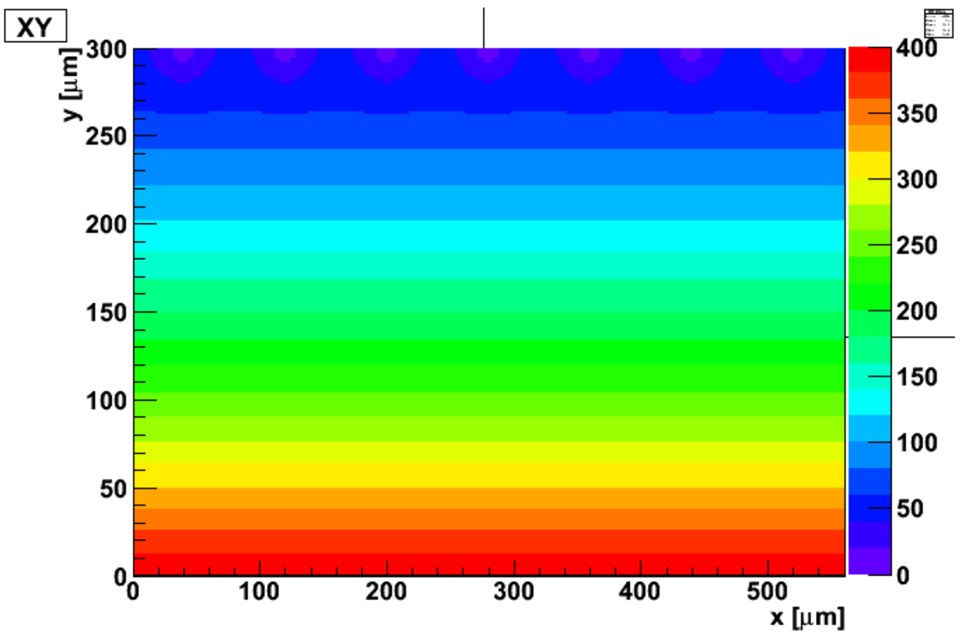


Examples of simulation – strip detectors

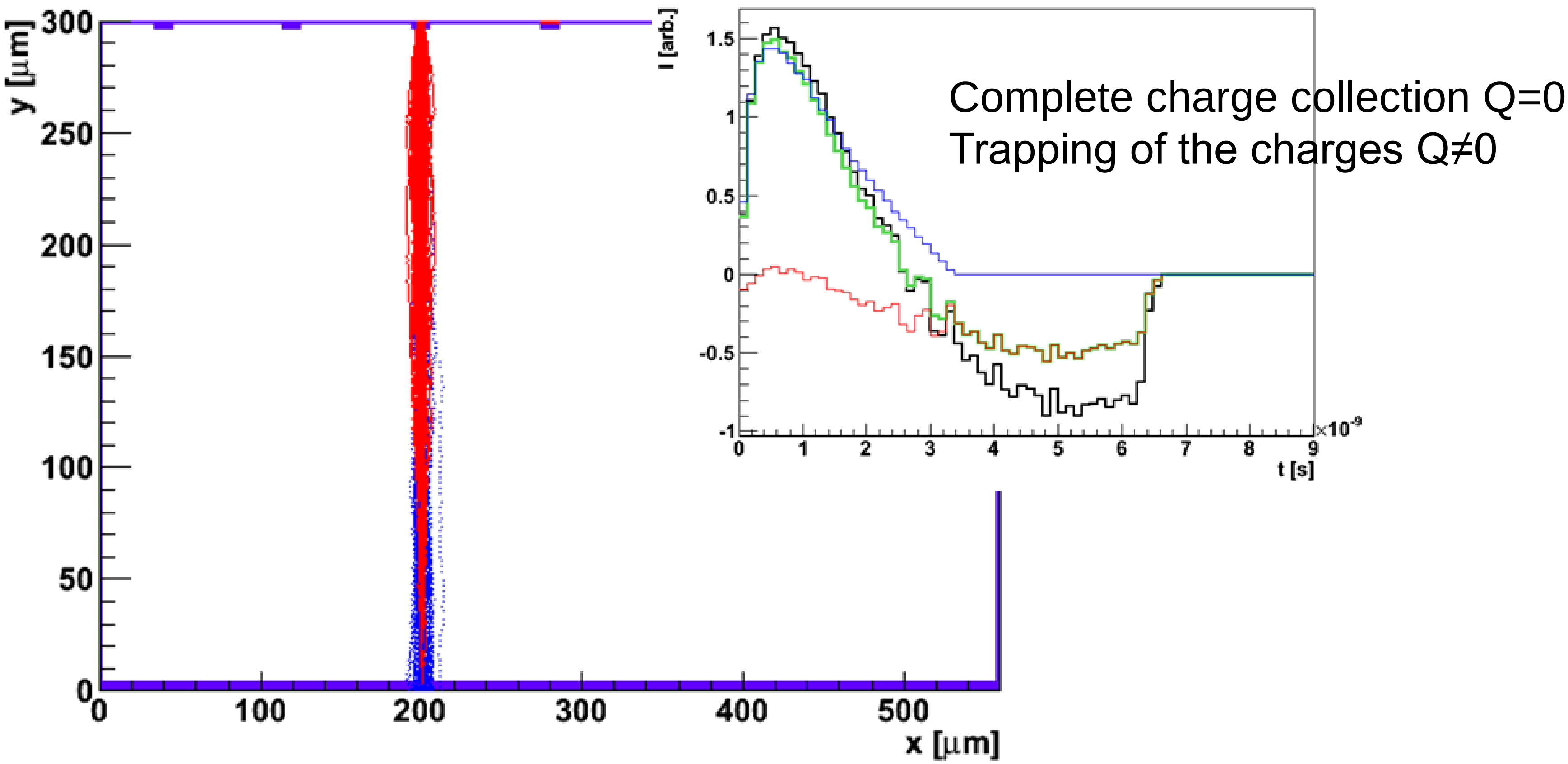
```
{
 TF3 *f2=new TF3("f2","[0]",0,3000,0,3000,0,3000);
 f2->SetParameter(0,-2);

 KStrip *det=new KStrip(80,10,2,7,300);
 det->Voltage=400;
 det->SetUpVolume(2);

 det->SetUpElectrodes();
 det->SetBoundaryConditions();
 det->SetUpMaterial(0); //Silicon

 det->NeffF=f2;
 det.SetDebug(0);
 det.CalField(0);
 det.CalField(1);
 det.diff=0;
 det.B[0]=0;
 det.B[1]=0;
 det.B[2]=1.8e-4;

 TCanvas c1;
  det->SetEntryPoint(200,299.9,0.5);
  det->SetExitPoint(200,1.,0.5);
  det->Temperature=253;
  det->diff=1;
  det->ShowMipIR(100);

 TCanvas c2;
  det->MipIR(300,1);
  det->sum->DrawCopy();
  SimpleTrap(det.pos,10e-9,0,25e-9);
SimpleTrap(det.neg,12e-9,0,25e-9);
  det->sum->Reset();
det->sum->Add(det.pos); det->sum->Add(det.neg);
det->sum.Draw("SAME");   det->pos->Draw("SAME");
det->neg->Draw("SAME");
}
```





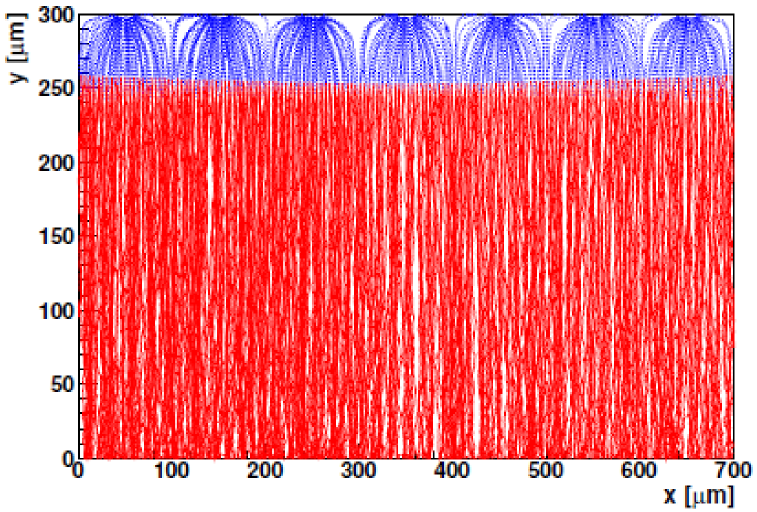


Examples of simulation – strip detectors

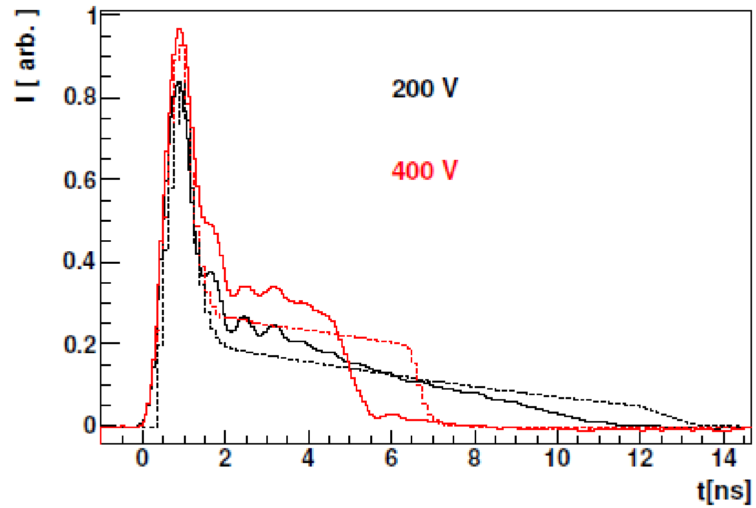


- Simulation of charge collection for the track through the neighboring strip (very difficult to simulate with TCAD in resources efficient way)

Examples of simulation – Edge-TCT

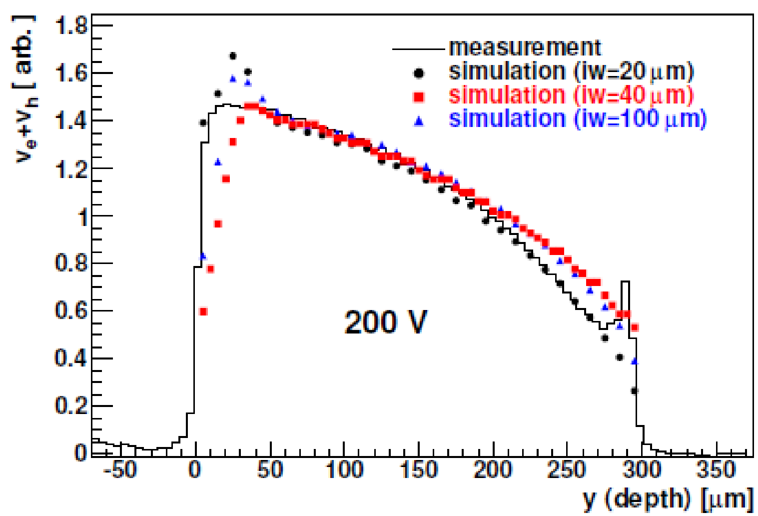


(a)



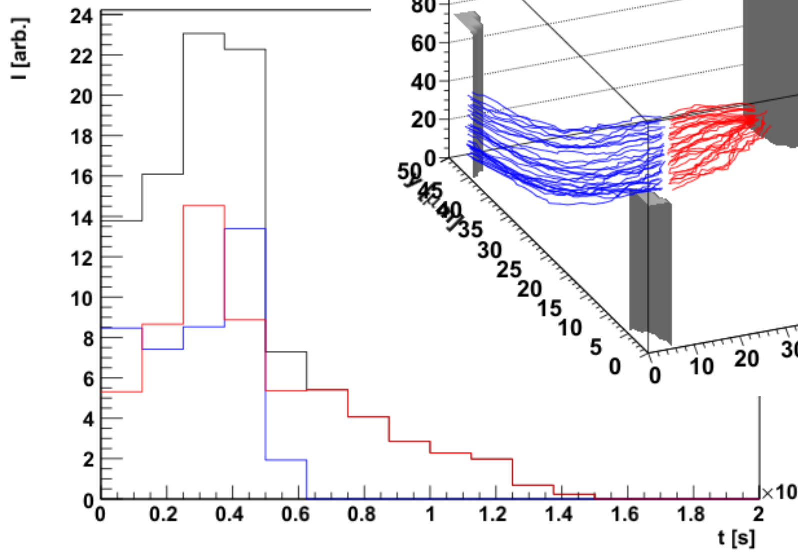
(b)

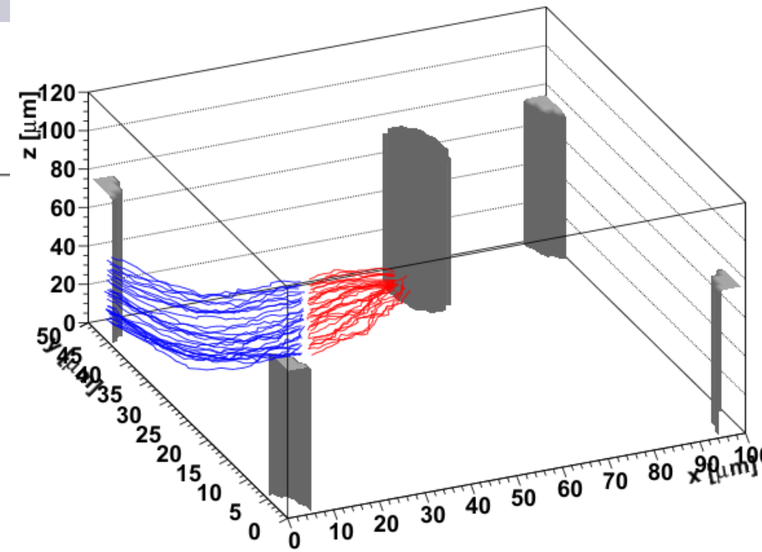
- Non-irradiated HPK sensor (width 80um - pitch 18 um-thickness 300 um)
- Gaussian beam
- Good agreement in velocity profiles – shown the significance of boundary condition

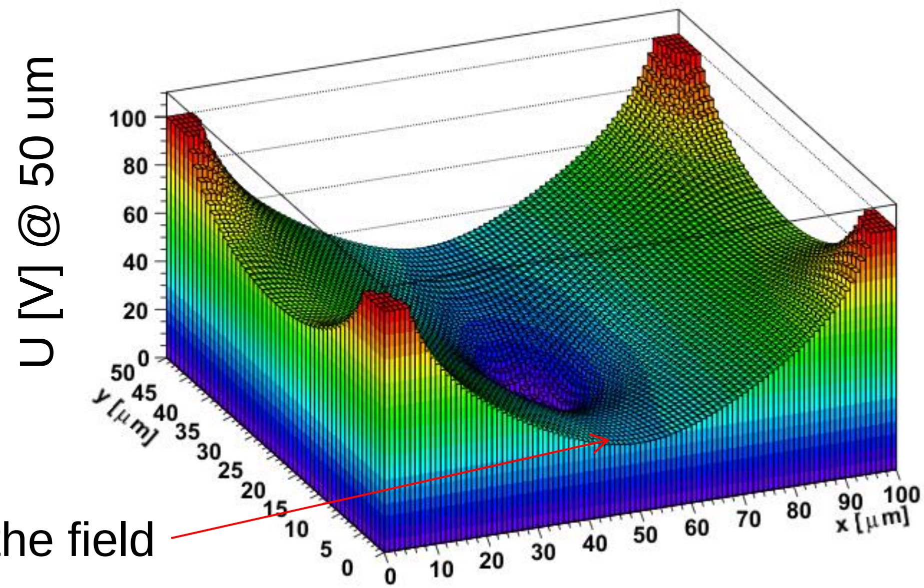


3D detectors

```
{
 gStyle->SetCanvasPreferGL(kTRUE);

  // define a 3D detector with 5 electrodes
  // x=100 , y is 50 and thickness 120
  K3D *det=new K3D(5,100,50,120);
  // define the voltage
  det->Voltage=100;
  // define the drift mesh size and simulation mesh size in microns
  det->SetUpVolume(1,1);
  // define  columns #, postions, weigthing factor 2=0 , material Al=1
  det->SetUpColumn(0,0,0,5,75,2,1);
  det->SetUpColumn(1,100,0,5,75,2,1);
  det->SetUpColumn(2,0,50,5,75,2,1);
  det->SetUpColumn(3,100,50,5,75,2,1);
  det->SetUpColumn(4,50,25,5,-75,16385,1);
  Float_t Pos[3]={100,50,1};
  Float_t Size[3]={100,50,2};
  det->ElRectangle(Pos,Size,0,20);
  det->SetUpElectrodes();
  det->SetBoundaryConditions();
  //define the space charge
  TF3 *f2=new TF3("f2","x[0]*x[1]*x[2]*0+[0]",0,3000,0,3000,0,3000);
  f2->SetParameter(0,-2);

  det->NeffF=f2;
  det->CalField(0); // calculate weigting field
  det->CalField(1); // calculate electric field

   // set entry points of the track
   det->enp[0]=30;   det->enp[1]=30;   det->enp[2]=50;
   det->exp[0]=30;   det->exp[1]=30;   det->exp[2]=10;

   // switch on the diffusion
   det->diff=1;
   // Show mip track
   TCanvas c1; c1.cd();
   det.ShowMipIR(30);
   // Show electric potential
   TCanvas c2; c2.cd();
   det.Draw("EPxy",60).Draw("COLZ");
   // calcualte induced current
   TCanvas c3; c3.cd();
   det.MipIR(100);
   det->sum.Draw();   det->neg.Draw("SAME");   det->pos.Draw("SAME");
}
```



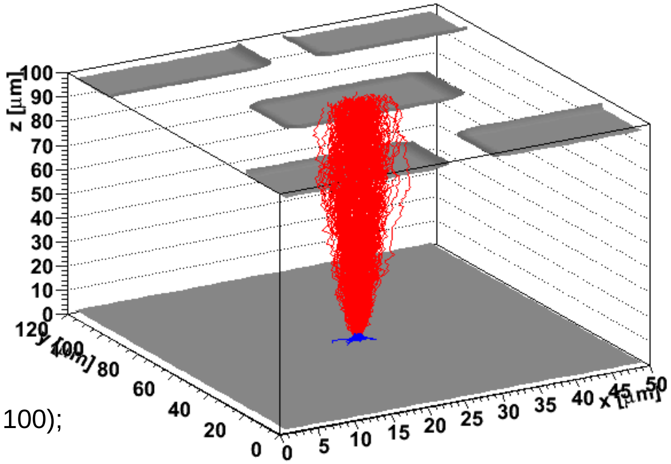


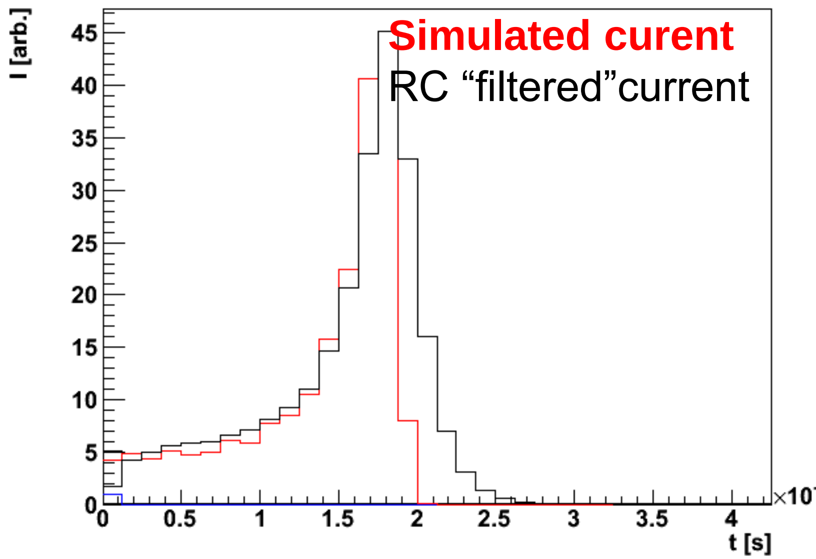


saddle in the field

Examples - Pixel sensor

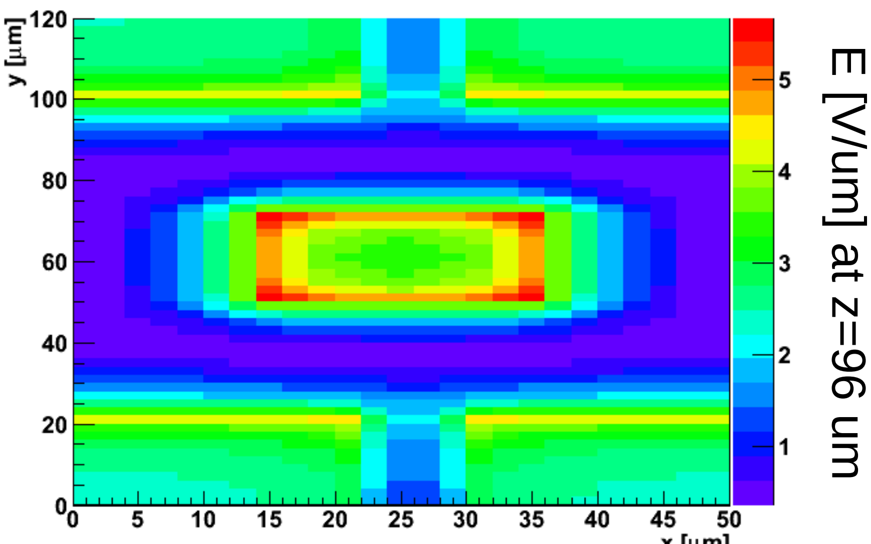
TCT generation of red light





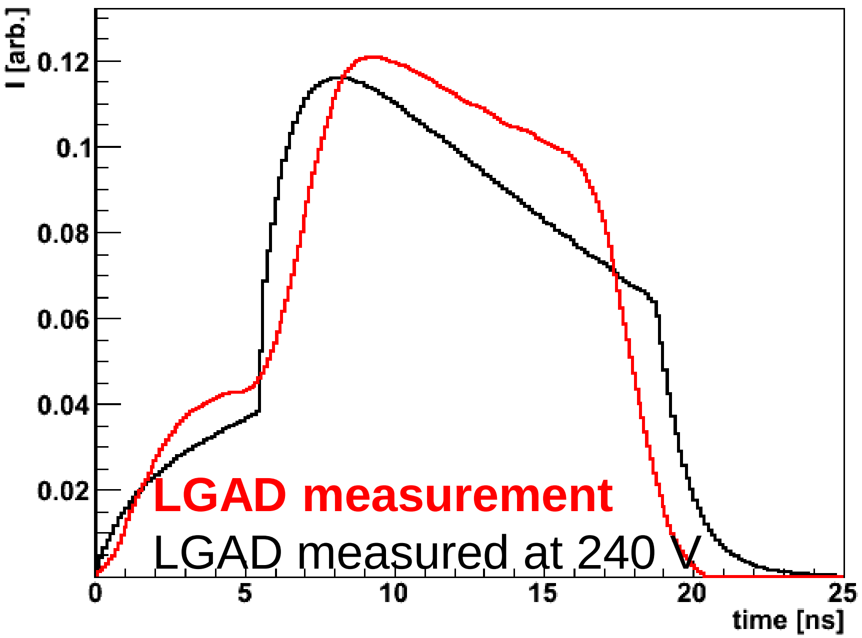

```
{
 KPixel *det=new KPixel(5,50,120,100);
 det->Voltage=200;
 det->SetUpVolume(2,2,1);
 det->SetUpPixel(0,25,60,10,10,2,16385);
 det->SetUpPixel(1,10,10,10,10,2,1);
 det->SetUpPixel(2,40,10,10,10,2,1);
 det->SetUpPixel(3,40,110,10,10,2,1);
 det->SetUpPixel(4,10,110,10,10,2,1);
 det->SetUpElectrodes();
 det->SetBoundaryConditions();
  TF3 *f2=new TF3("f2","x[0]*x[1]*x[2]*0+[0]",0,3000,0,3000,0,3000);
  f2->SetParameter(0,-2);
  det->NeffF=f2;
  det->CalField(0);  det->CalField(1);
  det->diff=1;
  det->enp[0]=25;   det->enp[1]=60;   det->enp[2]=1;
  det->exp[0]=25;   det->exp[1]=60;   det->exp[2]=3;
  det->MipIR(200,3);
  det.sum.DrawCopy(); det.neg.Draw("SAME"); det.pos.Draw("SAME");
  //electronics processing - RC filter
  Elec el(3e-12);
  el->preamp(det.sum);
  det->sum.Scale(det.pos.GetMaximum()/ det->sum.GetMaximum());
  det->sum.Draw("SAME");
}
```

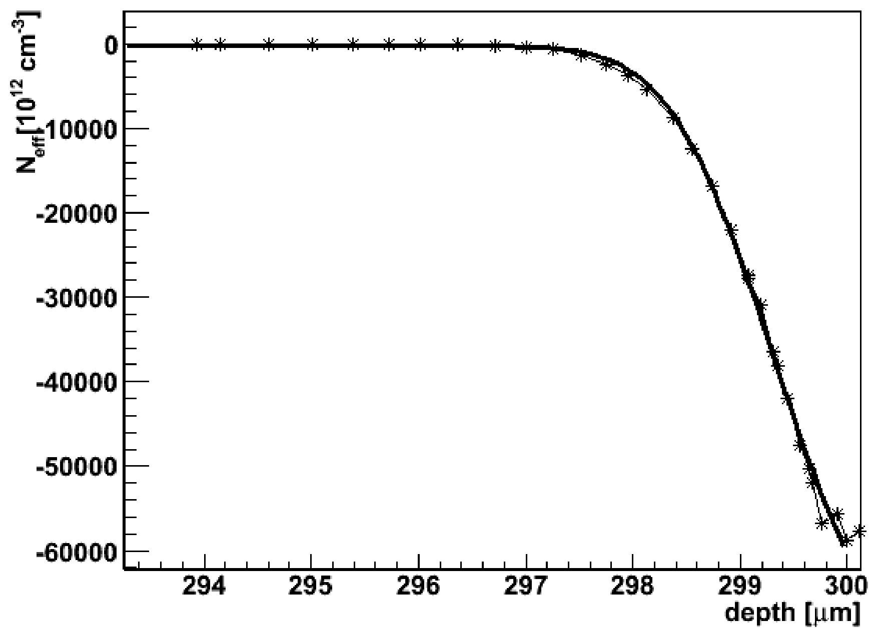
Electric field strength



Examples LGAD diodes

- Multiplication is treated in KDetSim in the same way as e.g. in GEANT – multiplied carriers (in accordance with impact ionization process and set thresholds) are treated in the same way as original charges.
- Problem of charge multiplication – diffusion hard to implement with impact ionization – impact ionization coefficients were measured with devices with the diffusion taken into account





Conclusions

- KDetSim is fast and highly portable root based code for simulation of signal in semiconductor detectors.
- It is well suited for TCT.
- Anyone interested is welcome to join in writing/debugging/improving code.