

Monte Carlo simulations - Allpix²

Daniel Hynds

Updated simulation configuration

```
[Allpix]
number_of_events = 1000
detectors_file = "tutorial-geometry.conf"
log_level = "Warning"
```

```
[GeometryBuilderGeant4]
```

```
[DepositionGeant4]
particle_type = "Pi+"
source_energy = 120GeV
source_type = "beam"
beam_size = 3mm
source_position = 0um 0um -200mm
beam_direction = 0 0 1
physics_list = FTFP_BERT_EMZ
```

```
[ElectricFieldReader]
model="linear"
bias_voltage=-50V
depletion_voltage=-30V
output_plots = 1
```

```
[ProjectionPropagation]
temperature = 293K
output_plots = 1
```

```
[SimpleTransfer]
output_plots = 1
```

```
[DefaultDigitizer]
threshold = 600e
output_plots = 1
```

```
[detector1]
type = "timepix"
position = 0mm 0mm 0mm
orientation = 0 0 0
```

```
[detector2]
type = "timepix"
position = 0mm 0mm 20mm
orientation = 0 0 0
```

```
[detector3]
type = "timepix"
position = 0mm 0mm 40mm
orientation = 0 0 0
```

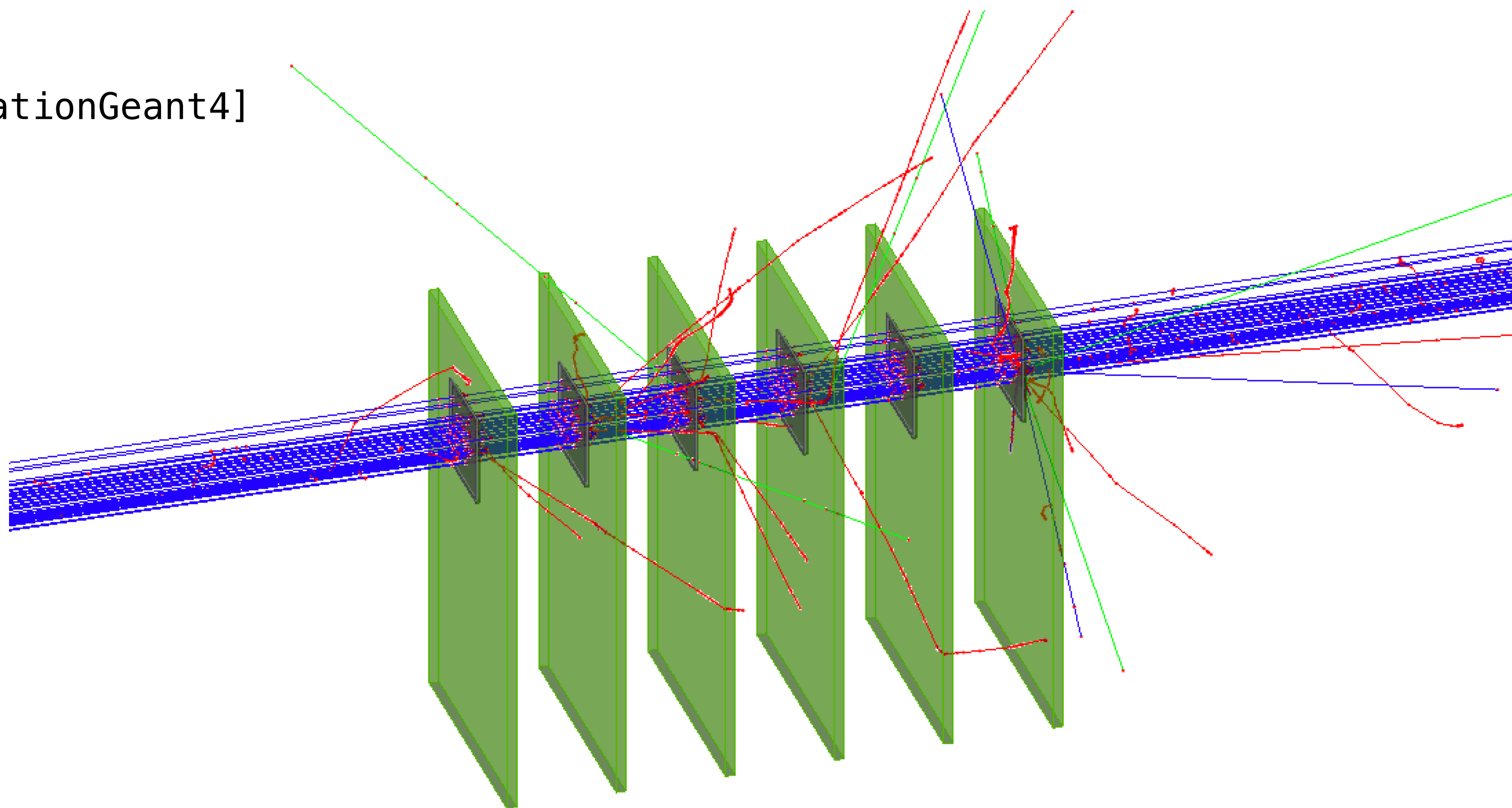
```
[detector4]
type = "timepix"
position = 0mm 0mm 60mm
orientation = 0 0 0
```

```
[detector5]
type = "timepix"
position = 0mm 0mm 80mm
orientation = 0 0 0
```

```
[detector6]
type = "timepix"
position = 0mm 0mm 100mm
orientation = 0 0 0
```

Visualising the setup

[VisualizationGeant4]



Corryvreckan - testbeam reconstruction software

```
[Allpix]
number_of_events = 1000
detectors_file = "tutorial-geometry.conf"
log_level = "Warning"
```

```
[GeometryBuilderGeant4]
```

```
[DepositionGeant4]
particle_type = "Pi+"
source_energy = 120GeV
source_type = "beam"
beam_size = 3mm
source_position = 0um 0um -200mm
beam_direction = 0 0 1
physics_list = FTFP_BERT_EMZ
```

```
[ElectricFieldReader]
model="linear"
bias_voltage=-50V
depletion_voltage=-30V
output_plots = 1
```

```
[ProjectionPropagation]
temperature = 293K
output_plots = 1
```

```
[SimpleTransfer]
output_plots = 1
```

```
[DefaultDigitizer]
threshold = 600e
output_plots = 1
```

```
[CorryvreckanWriter]
reference = detector1
dut = dut
```

```
[detector1]
type = "timepix"
position = 0mm 0mm 0mm
orientation = 0 0 0
```

```
[detector2]
type = "timepix"
position = 0mm 0mm 20mm
orientation = 0 0 0
```

```
[detector3]
type = "timepix"
position = 0mm 0mm 40mm
orientation = 0 0 0
```

```
[dut]
type = "timepix"
position = 0mm 0mm 90mm
orientation = 0 45deg 0
```

```
[detector4]
type = "timepix"
position = 0mm 0mm 140mm
orientation = 0 0 0
```

```
[detector5]
type = "timepix"
position = 0mm 0mm 160mm
orientation = 0 0 0
```

```
[detector6]
type = "timepix"
position = 0mm 0mm 180mm
orientation = 0 0 0
```


Processing detectors in different ways

When we added more detectors to the geometry file, everything took care of things under the hood

- No need to add additional information to the simulation configuration file

What is happening is that a separate instance of each module is created per detector

- This allows some measure of multithreading to be used to improve simulation times - all detectors can be run in parallel

This behaviour is controlled by the module type, either it is **unique** or **detector-specific**

A single detector chain

GeometryBuilderGeant4

DepositionGeant4

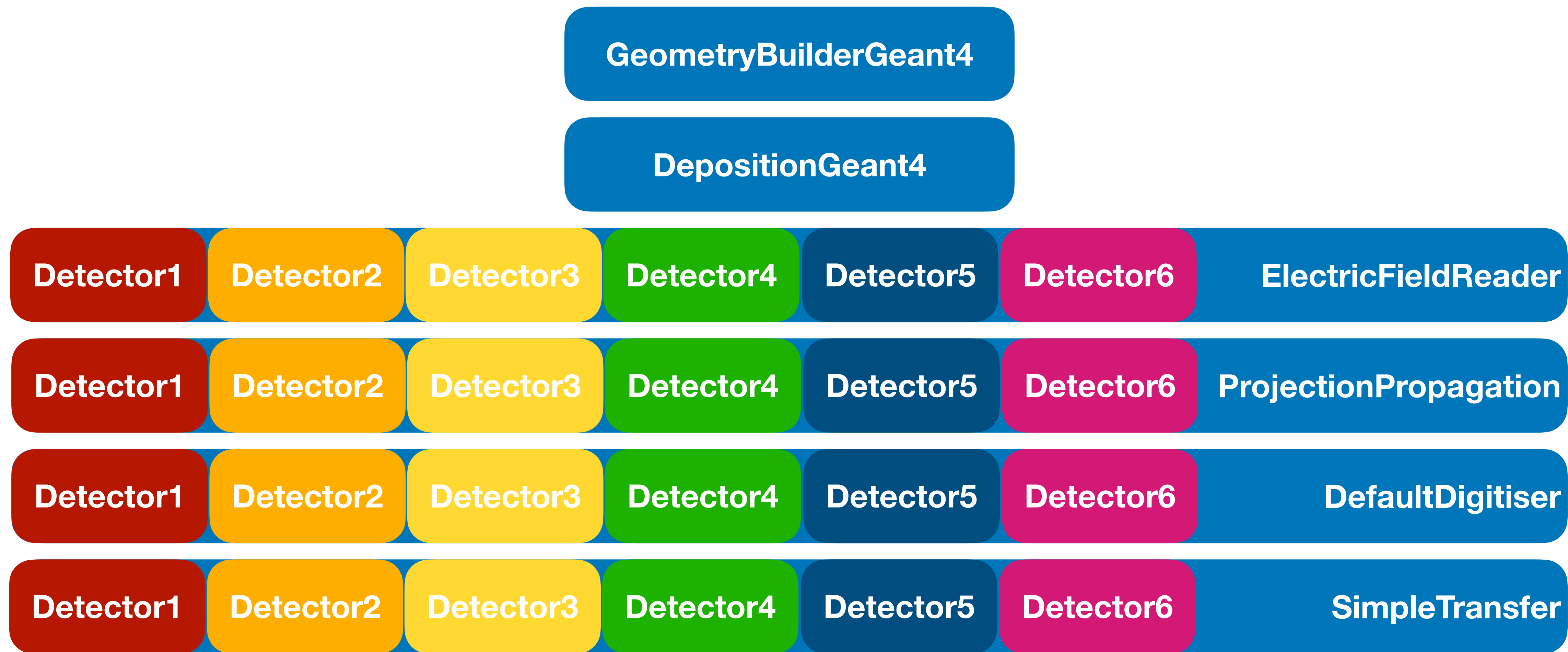
ElectricFieldReader

ProjectionPropagation

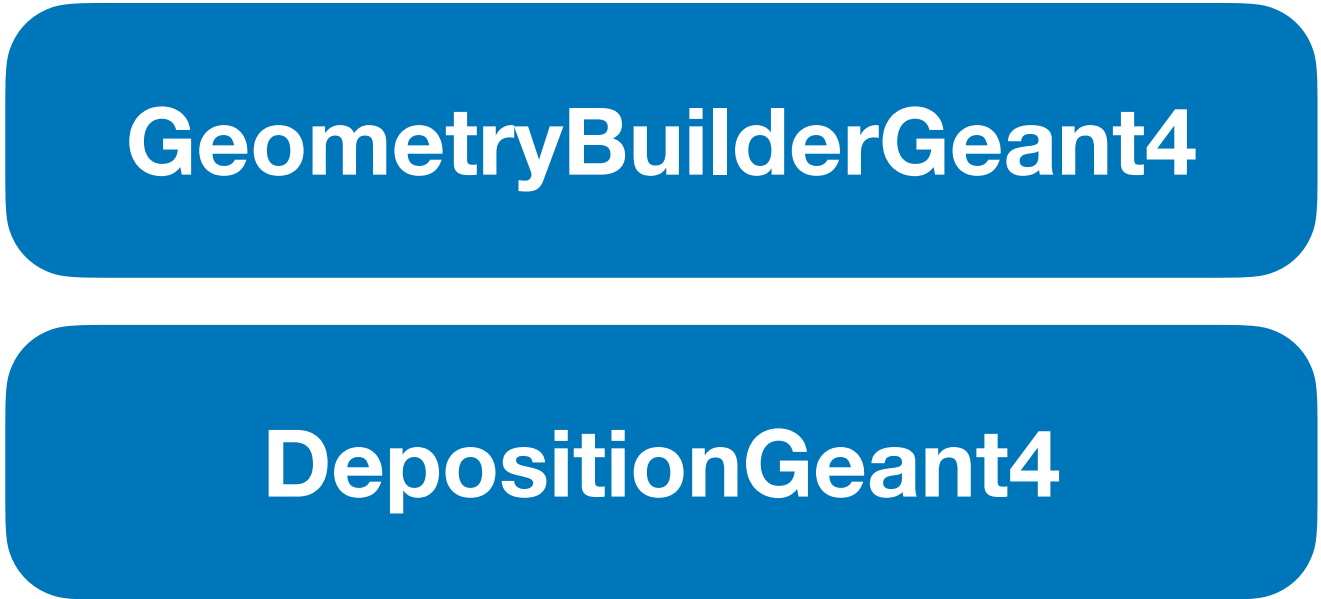
DefaultDigitiser

SimpleTransfer

A multi-detector chain



What if ?



Specifying detector type/name

When we added more detectors to the geometry file, everything took care of things under the hood

- No need to specify which detector each module was being applied to

By default, all modules will apply to all detectors. Can overwrite this behaviour by specifying either the **name** or **type** of detector to run over

- We can use this to either make a module have different parameters **per-detector**, or operate on a subset of detectors

Set to operate on all detectors



```
[ElectricFieldReader]
model="linear"
bias_voltage=-50V
depletion_voltage=-30V
```

```
[ElectricFieldReader]
model = "linear"
name = "detector1"
bias_voltage=-100V
depletion_voltage=-30V
```



Instantiation for detector1 will be overwritten by this one, since it is the same type of module and specified only for detector1

Specifying detector type/name

When we added more detectors to the geometry file, everything took care of things under the hood

- No need to specify which detector each module was being applied to

By default, all modules will apply to all detectors. Can overwrite this behaviour by specifying either the **name** or **type** of detector to run over

- We can use this to either make a module have different parameters per-detector, or operate on a **subset of detectors**

```
[ProjectionPropagation]
name = "detector2", "detector3", "detector4", "detector5", "detector6"
temperature = 293K
```

```
[GenericPropagation]
name = "detector1"
temperature = 293K
```


Specifying detector type/name

```
[Allpix]
number_of_events = 1000
detectors_file = "tutorial-geometry.conf"
log_level = "Warning"
```

```
[GeometryBuilderGeant4]
```

```
[DepositionGeant4]
particle_type = "Pi+"
source_energy = 120GeV
source_type = "beam"
beam_size = 3mm
source_position = 0um 0um -200mm
beam_direction = 0 0 1
physics_list = FTFP_BERT_EMZ
```

```
[ElectricFieldReader]
model="linear"
bias_voltage=-50V
depletion_voltage=-30V
output_plots = 1
```

```
[ElectricFieldReader]
model="linear"
name="detector1"
bias_voltage=-100V
depletion_voltage=-30V
output_plots = 1
```

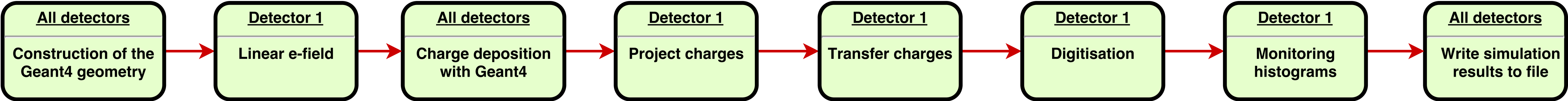
```
[ProjectionPropagation]
temperature = 293K
name = "detector2", "detector3", "detector4",
"detector5", "detector6"
output_plots = 1
```

```
[GenericPropagation]
name = "detector1"
temperature = 293K
output_plots = 1
```

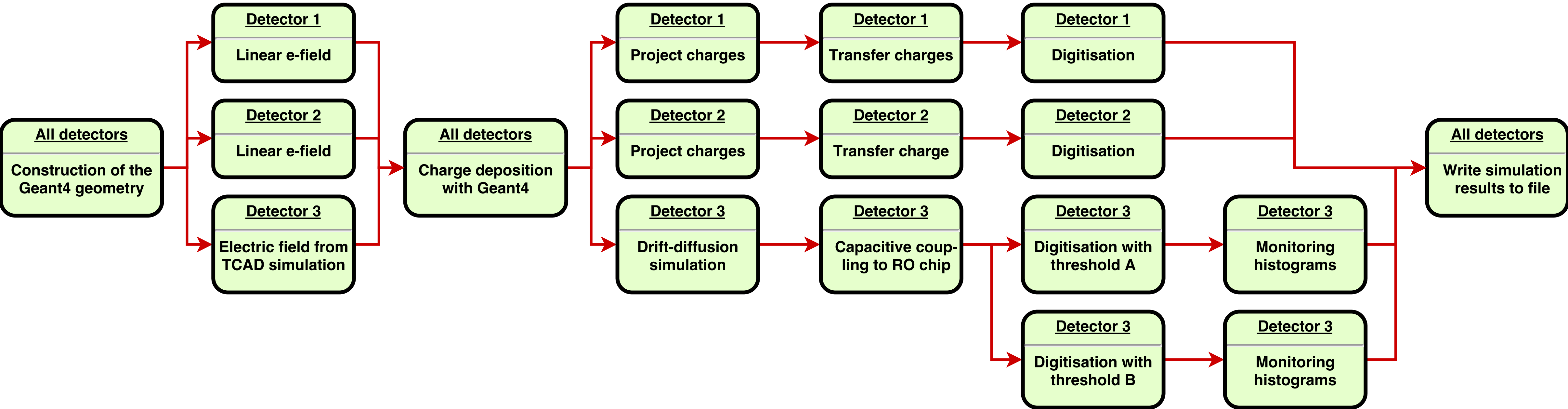
```
[SimpleTransfer]
output_plots = 1
```

```
[DefaultDigitizer]
threshold = 600e
output_plots = 1
```


Single detector chain



Multi-detector chain



Making your own module

Until now, setting up a simulation and configuring different modules for different detectors

- No need to touch c++ code, only config files

Next step is developing a custom module - keep in mind that modules may already be implemented/can be configured in a way that you need (cf. Digitisation is reasonable generic)

- Consider that making your module generic will benefit other users - no point in implementing 10 times a module to apply time walk effects to an ASIC

Useful script comes with the software to make it easy to develop new modules: `make_modules.sh`

- Define the name of the module
- Whether the module is unique or operates per-detector
- The type of message that the module accepts

Messages

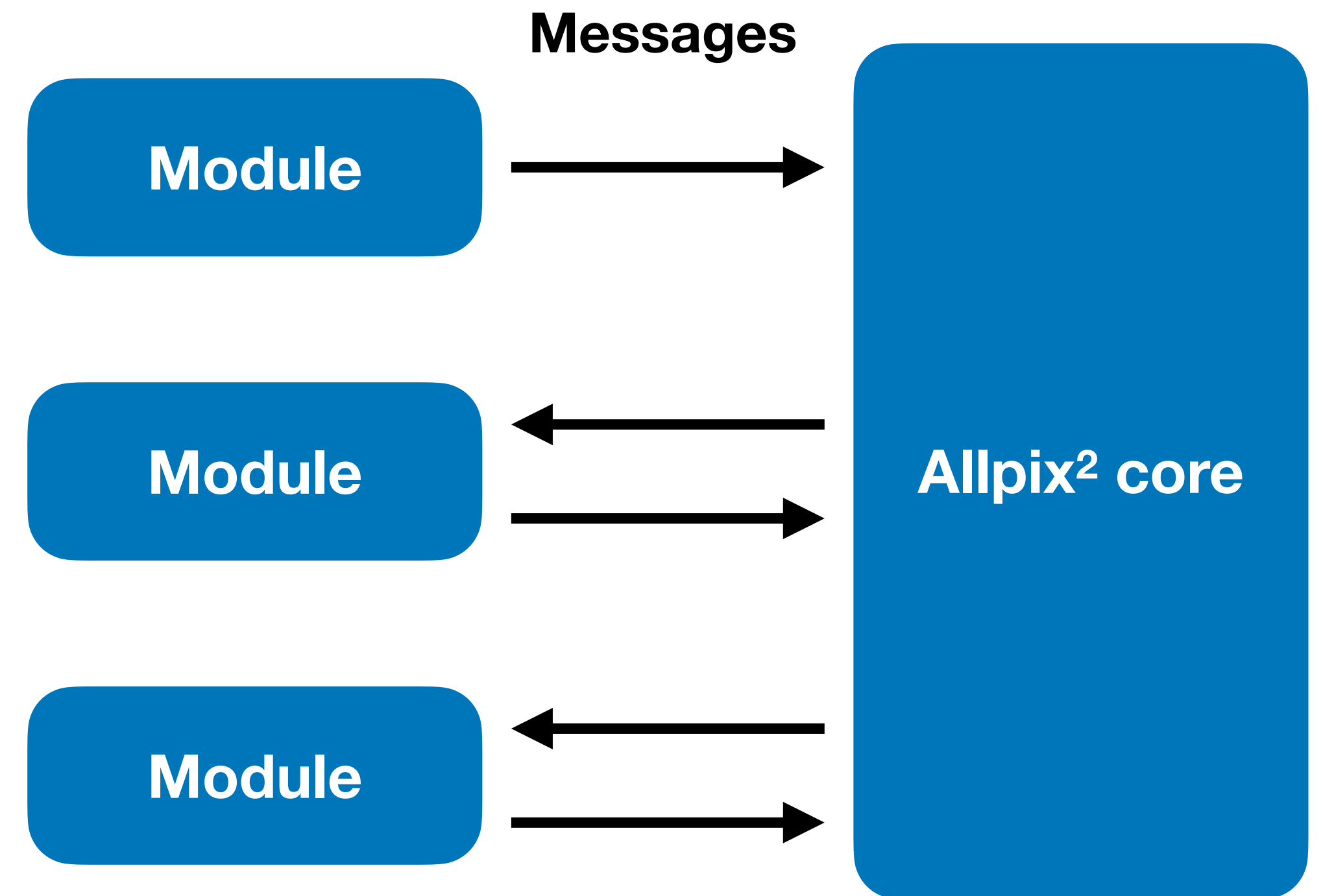
Modules exist entirely standalone in allpix-squared

- Information exchange by dispatching and receiving messages, via the core of the software
- Check performed on start-up for configuration errors - check with messages each module is waiting for and whether messages being dispatched are subsequently used

For per-detector modules, separate messages are dispatched for each detector, with the detector name used in the identification

New modules need to decide what objects to pick up

- DepositedCharges, PropagatedCharges, etc.



Message types

Inside “src/objects” the complete list of objects can be found, each of which has the message definition alongside it. Objects include:

- DepositedCharge
- PropagatedCharge
- Pulse
- PixelHit
- MCParticle
- Etc.

And are trivial to extend

```
#include "Pixel.hpp"

namespace allpix {
    /**
     * @ingroup Objects
     * @brief Pixel triggered in an event after digitization
     */
    class PixelHit : public Object {
    public:
        /**
         * @brief Construct a digitized pixel hit
         * @param pixel Object holding the information of the pixel
         * @param local_time Timing of the occurrence of the hit in local reference frame
         * @param global_time Timing of the occurrence of the hit in global reference frame
         * @param signal Signal data produced by the digitizer
         * @param pixel_charge Optional pointer to the related pixel charge
         */

        /**
         * @brief Typedef for message carrying pixel hits
         */
        using PixelHitMessage = Message<PixelHit>;
    };
};
```

Producing your own module

```
$ cd ../etc/scripts/  
$ ./make_module.sh
```

Preparing code basis for a new module:

```
Name of the module? TutorialExample  
Short description of the module? This module will demonstrate how to write a new module  
Type of the module?
```

```
1) unique  
2) detector  
#? 2  
Event order requirements?
```

```
1) Module  
2) SequentialModule  
#? 1
```

```
Input message type? DepositedCharge  
Creating directory and files...
```

```
./make_module.sh: line 162: realpath: command not found  
Name:      TutorialExample  
Description: This module will demonstrate how to write a new module  
Author:    Daniel Hynds (daniel.hynds@cern.ch)  
Path:        
This module listens to "DepositedCharge" messages from one detector
```

Re-run CMake in order to build your new module.

Writing your own module

C* TutorialExampleModule

h* TutorialExampleModule

C* TutorialExampleModule > No Selection

```
1 /**
2  * @file
3  * @brief Implementation of TutorialExample module
4  *
5  * @copyright Copyright (c) 2017-2023 CERN and the Allpix Squared authors.
6  * This software is distributed under the terms of the MIT License, copied verbatim in the file "LICENSE.md".
7  * In applying this license, CERN does not waive the privileges and immunities granted to it by virtue of its status as an
8  * Intergovernmental Organization or submit itself to any jurisdiction.
9  * SPDX-License-Identifier: MIT
10 */
11
12 #include "TutorialExampleModule.hpp"
13
14 #include <string>
15 #include <utility>
16
17 #include "core/utils/log.h"
18
19 using namespace allpix;
20
21 TutorialExampleModule::TutorialExampleModule(Configuration& config, Messenger* messenger, std::shared_ptr<Detector> detector)
22     : Module(config, detector), detector_(std::move(detector)), messenger_(messenger) {
23
24     // Allow multithreading of the simulation. Only enabled if this module is thread-safe. See manual for more details.
25     // allow_multithreading();
26
27     // Set a default for a configuration parameter, this will be used if no user configuration is provided:
28     config_.setDefault<int>("setting", 13);
29
30     // Parsing of the parameter "setting" into a member variable for later use:
31     setting_ = config_.get<int>("setting");
32
33     // Messages: register this module with the central messenger to request a certain type of input messages:
34     messenger_>bindSingle<DepositedChargeMessage>(this, MsgFlags::REQUIRED);
35 }
36
37 void TutorialExampleModule::initialize() {
38
39     // In this simple case we just print the name of this detector:
40     LOG(DEBUG) << "Detector with name " << detector_>getName();
41 }
42
43 void TutorialExampleModule::run(Event* event) {
44
45     // Messages: Fetch the (previously registered) messages for this event from the messenger:
46     auto message = messenger_>fetchMessage<DepositedChargeMessage>(this, event);
```

Initialise variables/
histograms etc.

Main code, executed
each event

Compiling and including your module

CMake set up to compile all modules in the corresponding directory

- Just need to rerun cmake from the build directory and compile

Module can then be added in the simulation configuration file in the same way as any other module

```
$ cd ../../build/  
$ cmake ..  
$ make install -j 4
```

```
$ cd ../examples/  
../../bin/allpix -c tutorial-simulation.conf
```

...

[TutorialExample]

...

A few other features - MC history

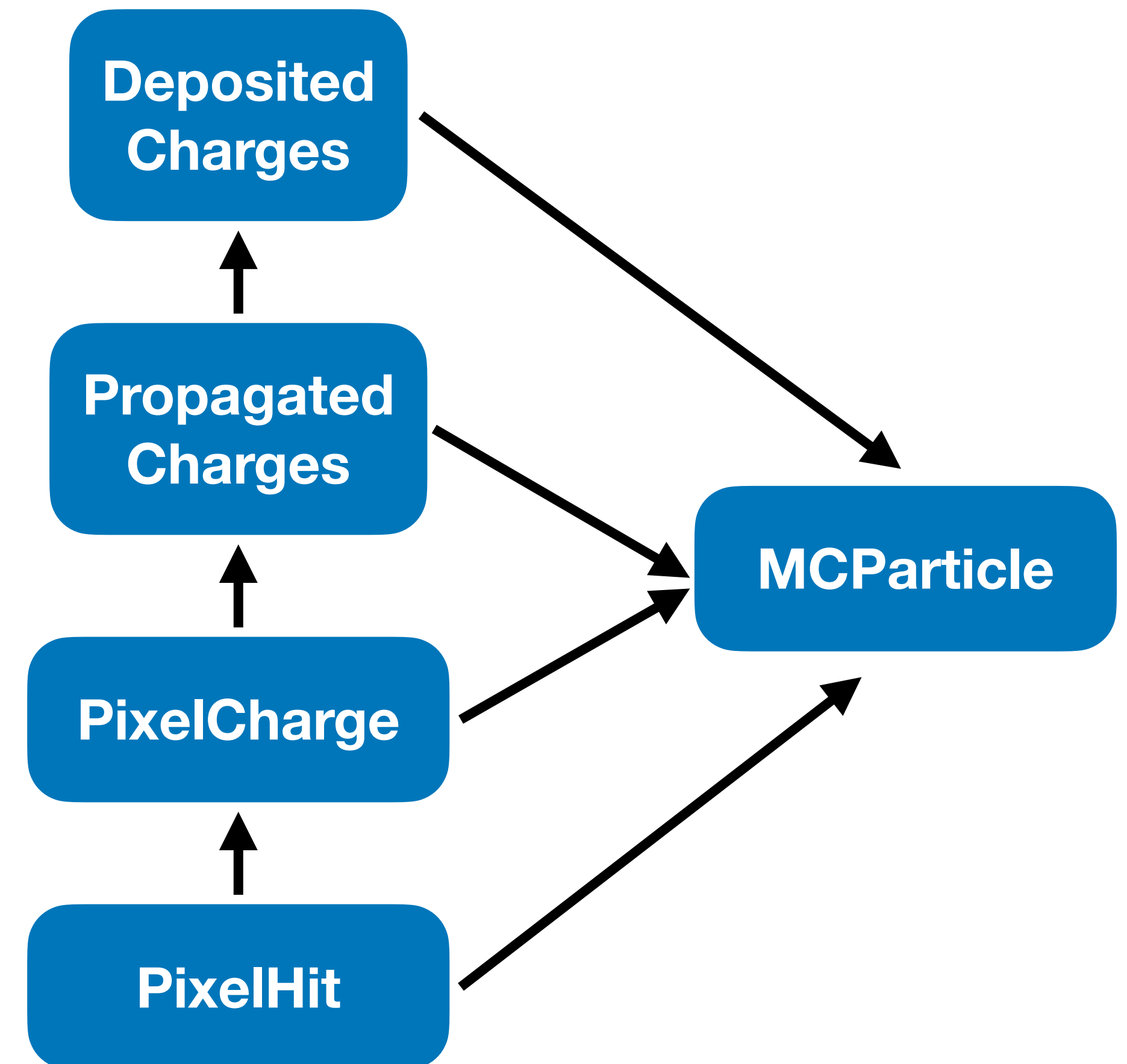
All objects contain information about where they come from

- Direct link to the preceding object
- All objects link back to original MC particle

Messages templated in the code, so adding a new object is straightforward

- Define the object, must inherit from Object
- Add a definition for the message

```
/**  
 * @brief Typedef for message carrying propagated charges  
 */  
using PropagatedChargeMessage = Message<PropagatedCharge>;
```



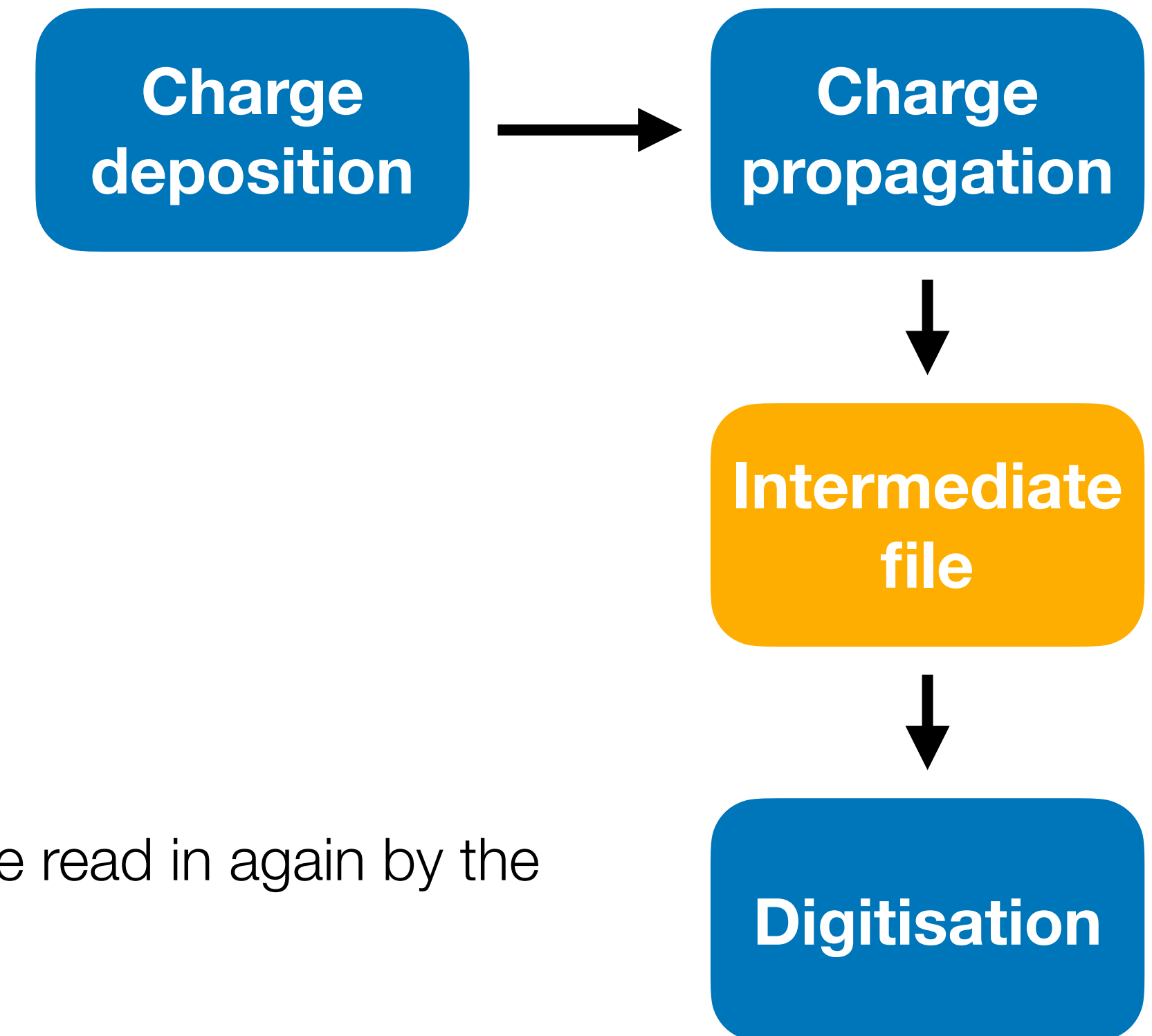
A few other features - output writing

Several output formats are already supported

- LCIO - format used in linear collider community
- RCE - ATLAS pixel group data format
- Corryvreckan - reconstruction framework developed in EP-LCD
- Text files - simple human-readable format
- RootObjects - allpix-squared data

This last class writes out native allpix-squared objects, such that they can be read in again by the framework

- Allows intermediate file-writing to avoid repeating CPU-intensive parts of the simulation
- eg. Write out propagated charge objects so that tuning of the digitisation parameters does not require re-running Geant4 and propagation code

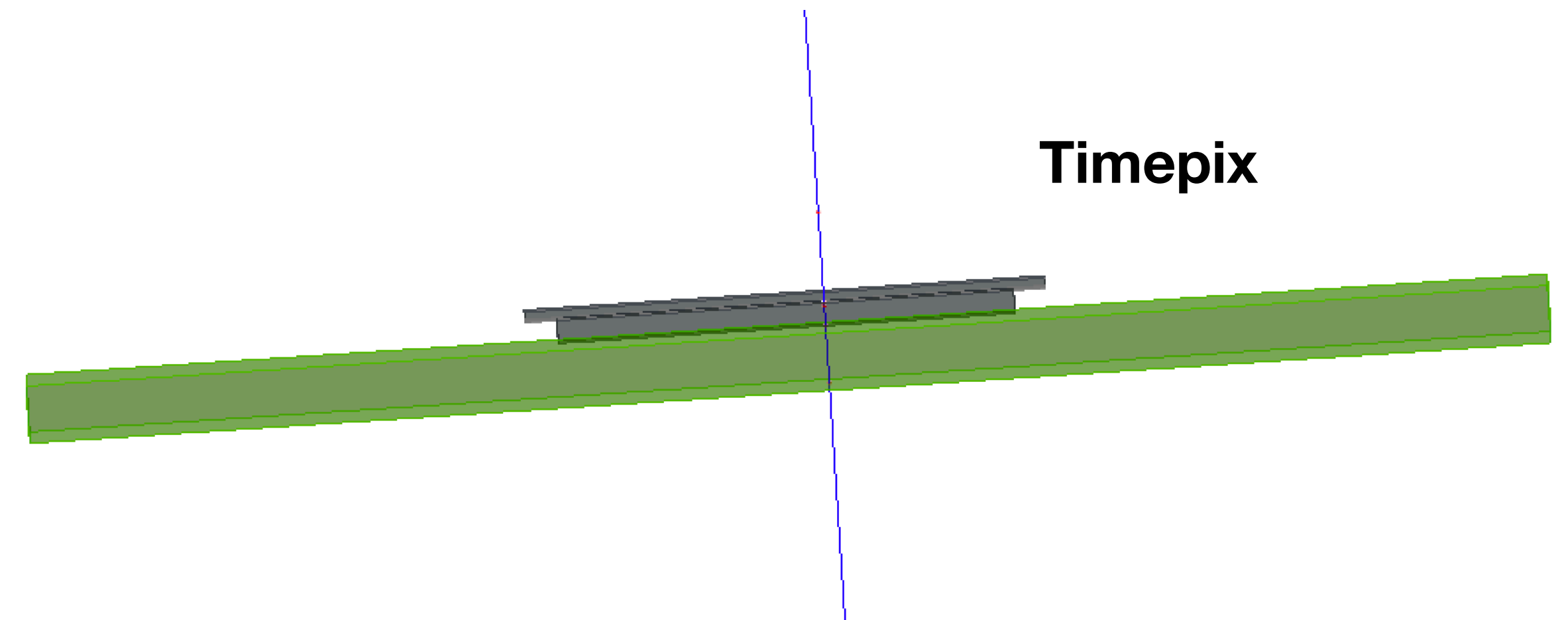
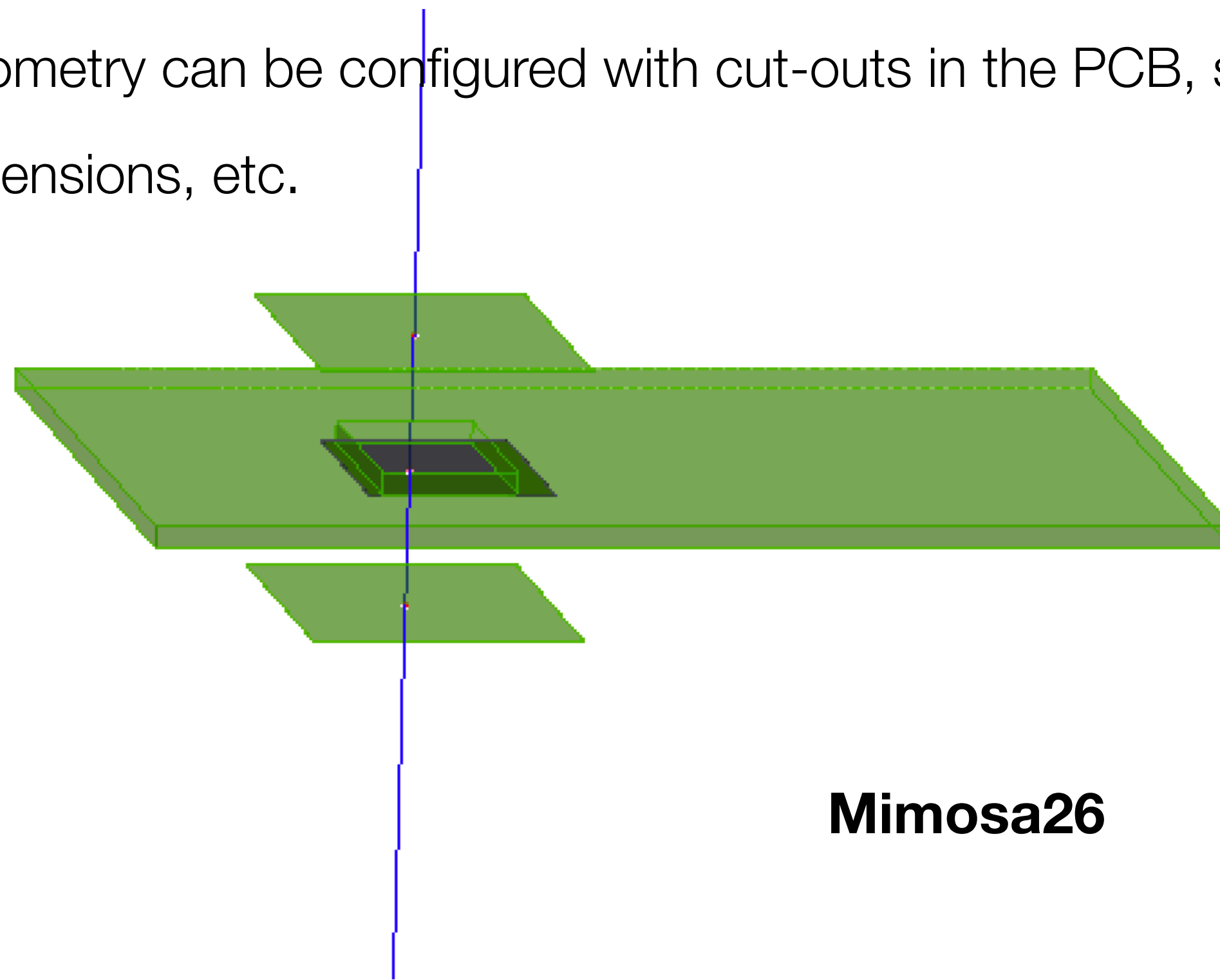


A few other features - geometry

Currently-implemented geometries are for hybrid and monolithic detectors

- Monolithic can be used for strip detectors, with 1 by n “pixels” of appropriate size

Geometry can be configured with cut-outs in the PCB, support materials (windows/physical supports), bump dimensions, etc.



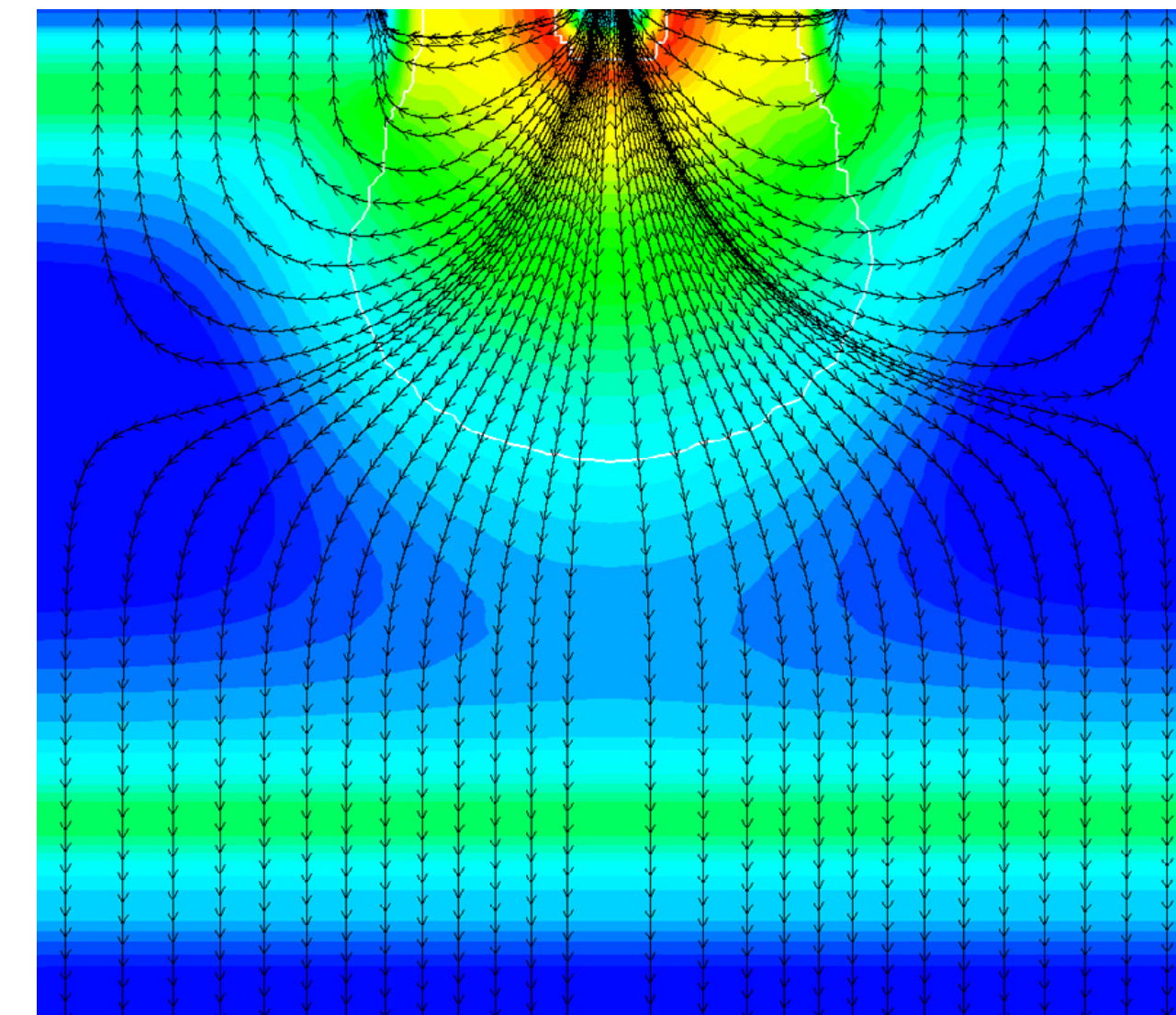
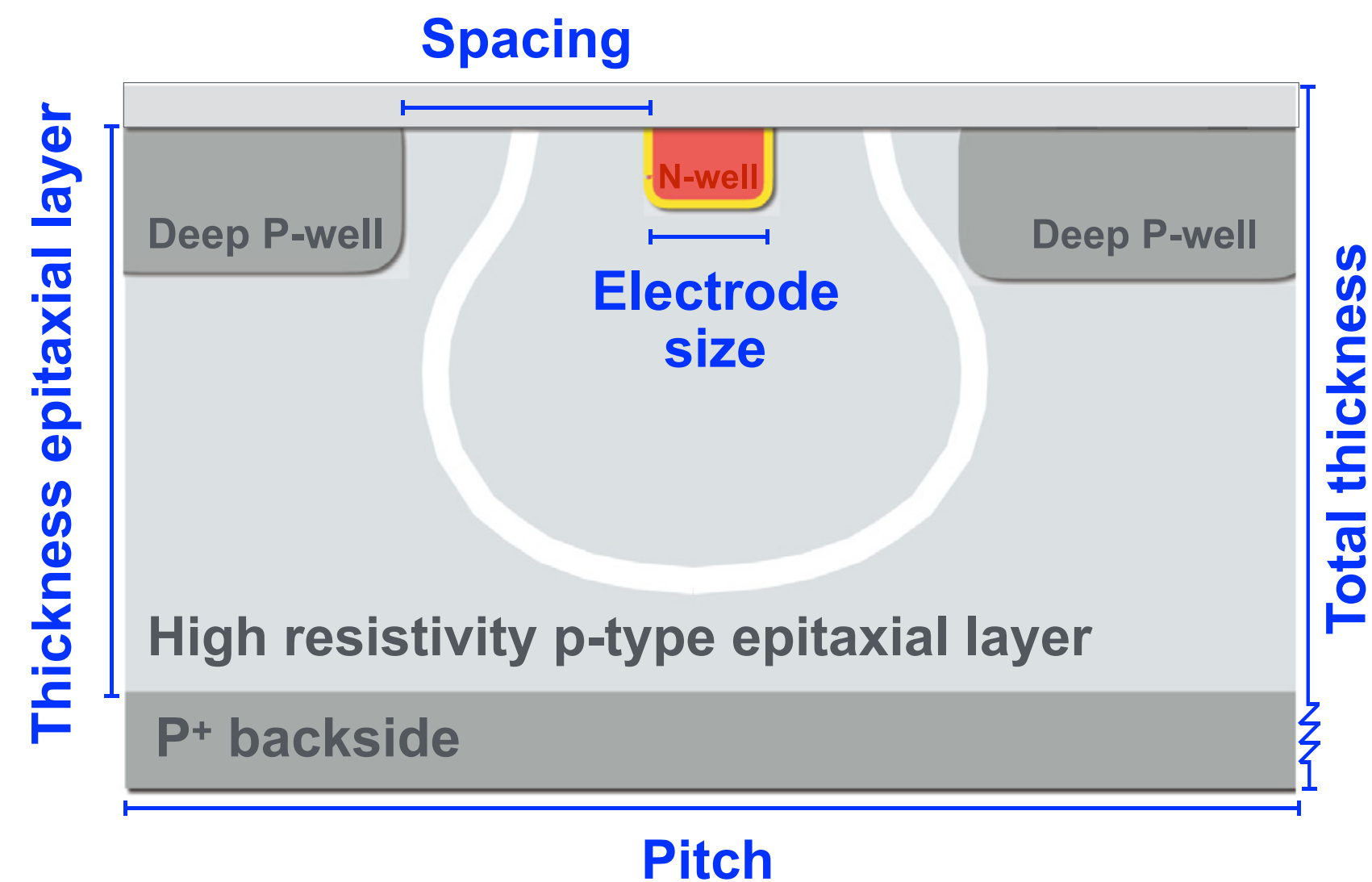
A few highlights of what is currently there

Electric and weighting field import from TCAD - 1

S.Spannagel et al, NIMA 964 (2020) 163784

TCAD field input is essential for more complex electric field configurations - particularly monolithic detectors

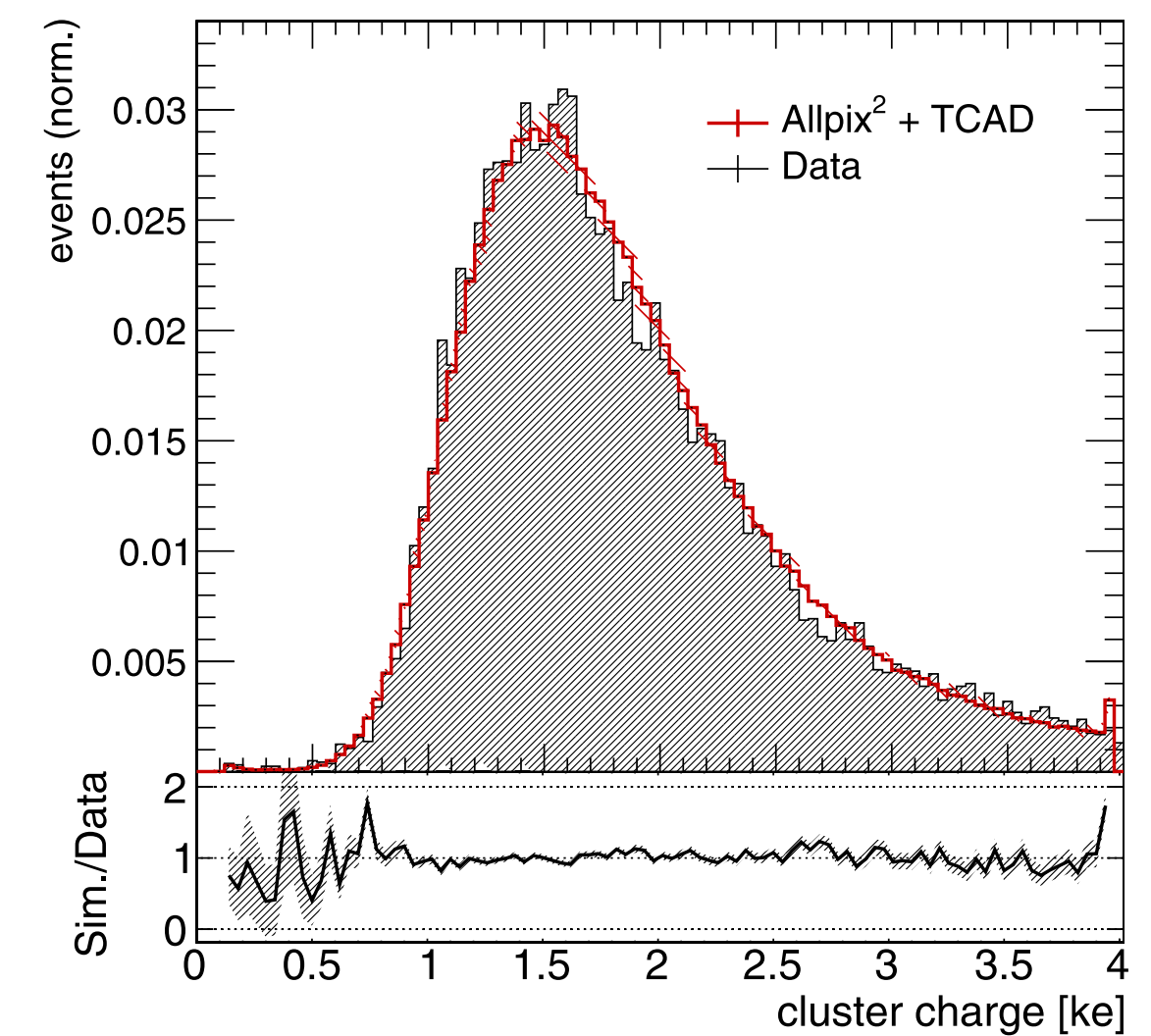
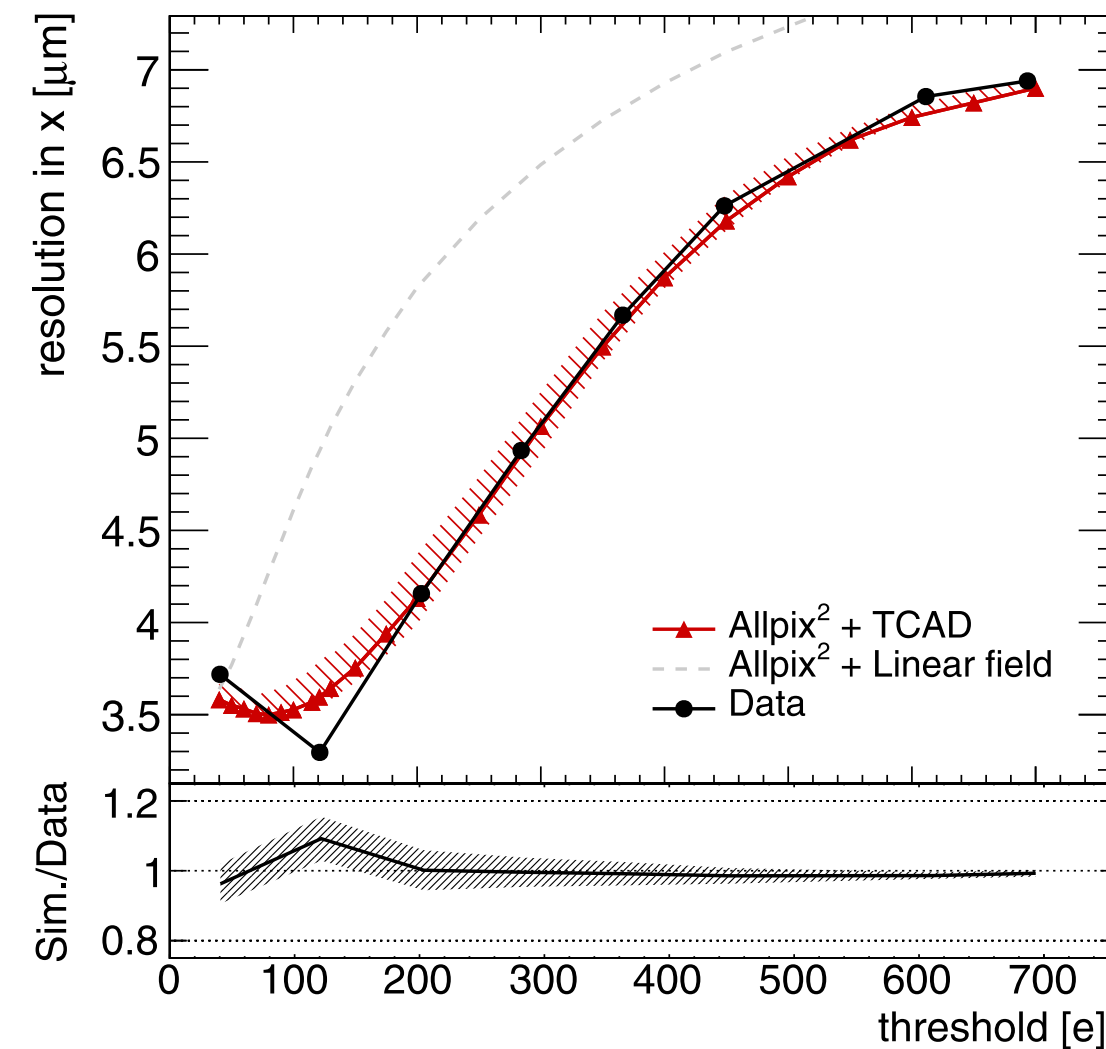
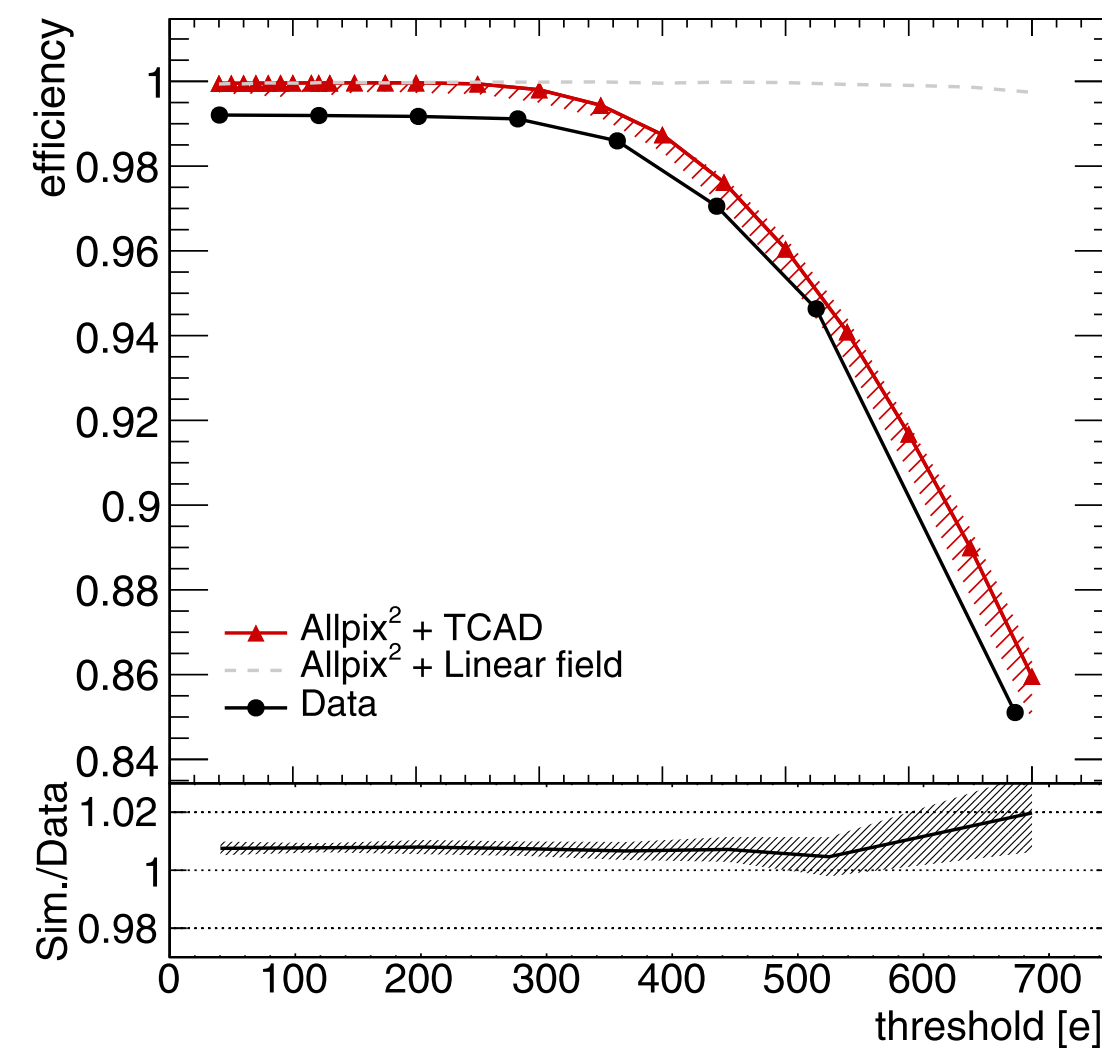
- Complicated field shape around the collection implants
- Undepleted bulk gives rise to carrier recombination



Currently implemented features - weighting field import from TCAD

TCAD field input is essential for more complex electric field configurations - particularly monolithic detectors

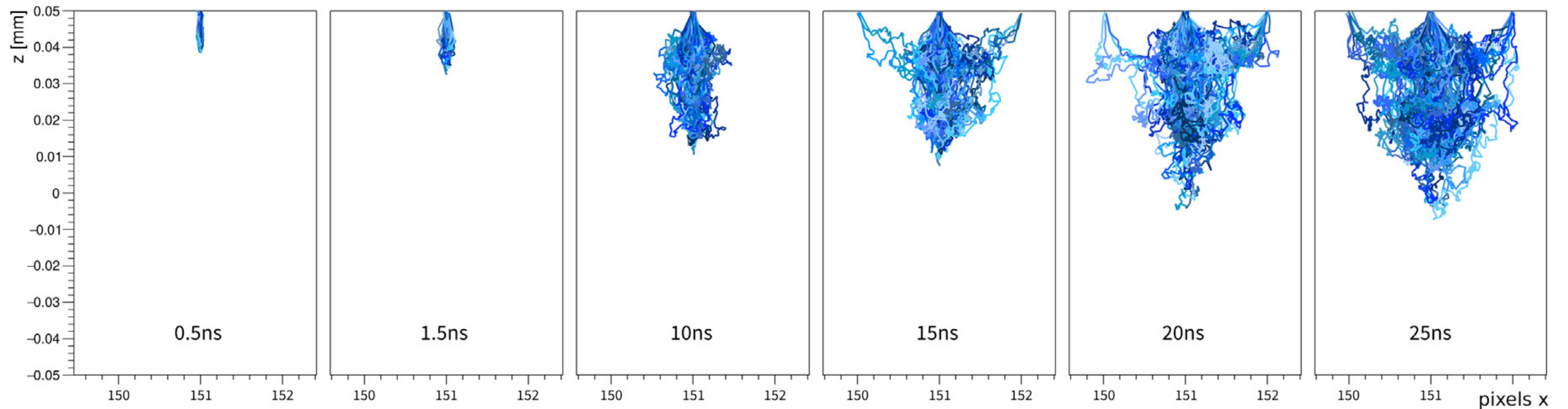
- Complicated field shape around the collection implants
- Undepleted bulk gives rise to carrier recombination



Currently implemented features - weighting field import from TCAD

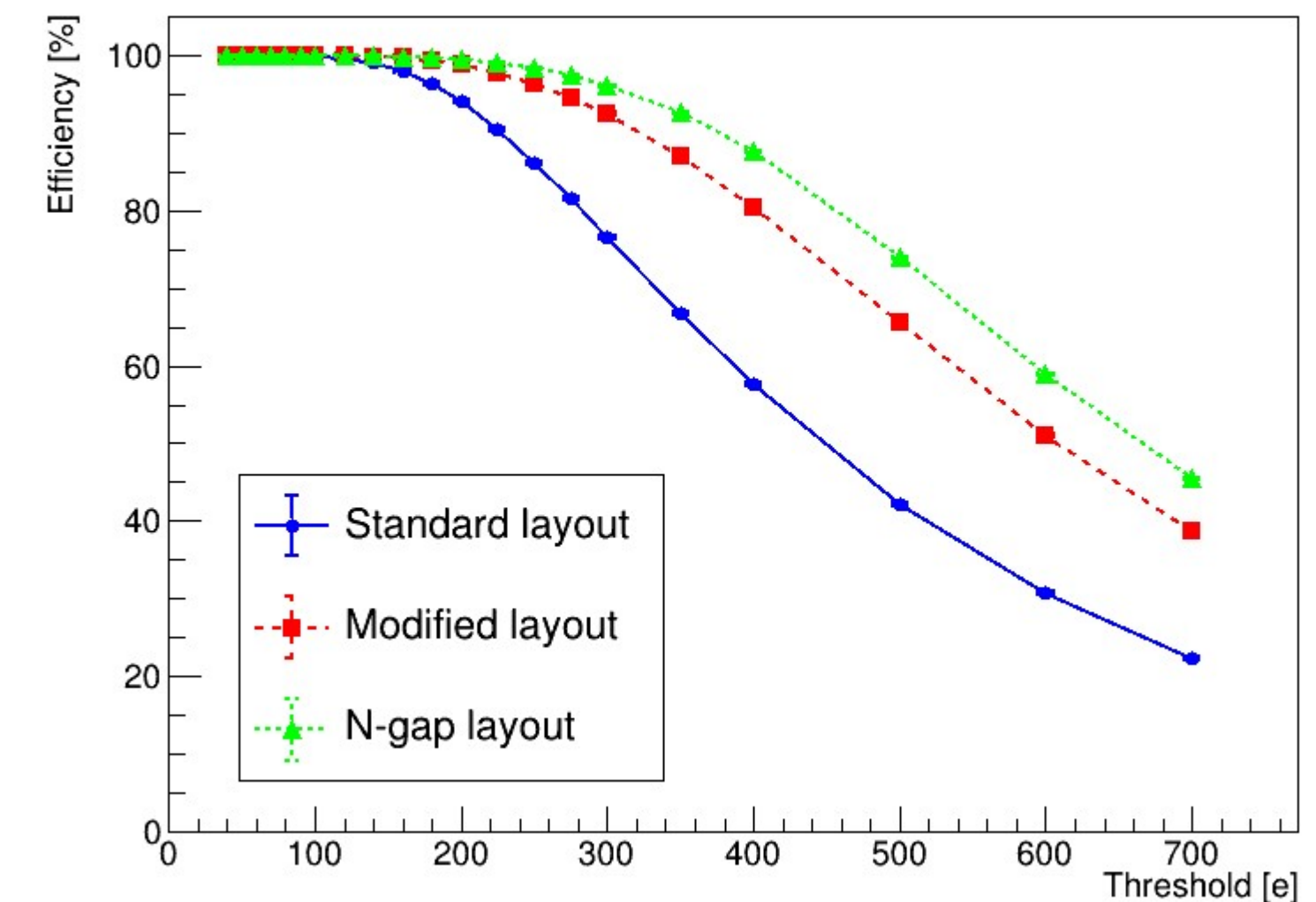
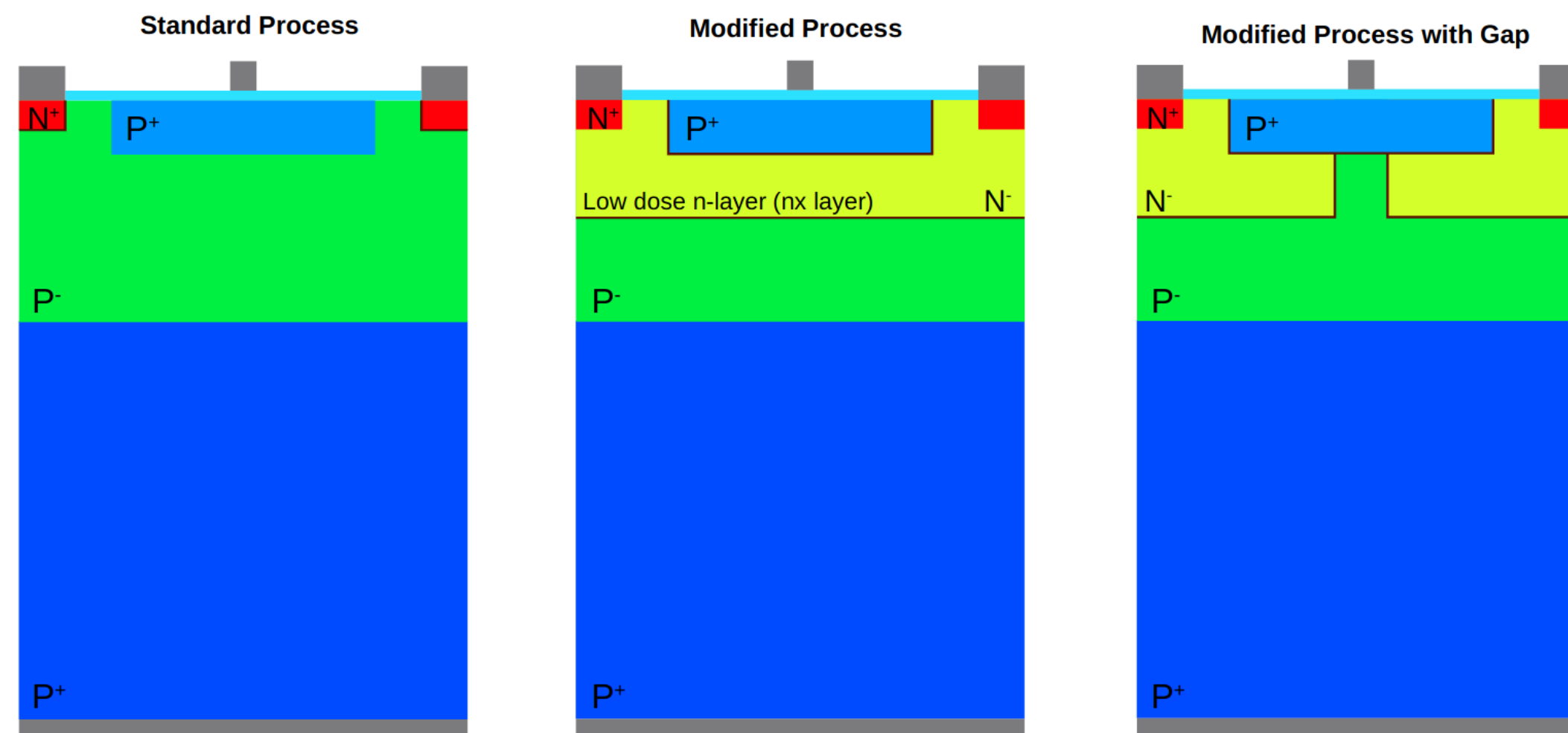
TCAD field input is essential for more complex electric field configurations - particularly monolithic detectors

- Complicated field shape around the collection implants
- Undepleted bulk gives rise to carrier recombination



Similar ongoing activities for 65 nm monolithic CMOS developments

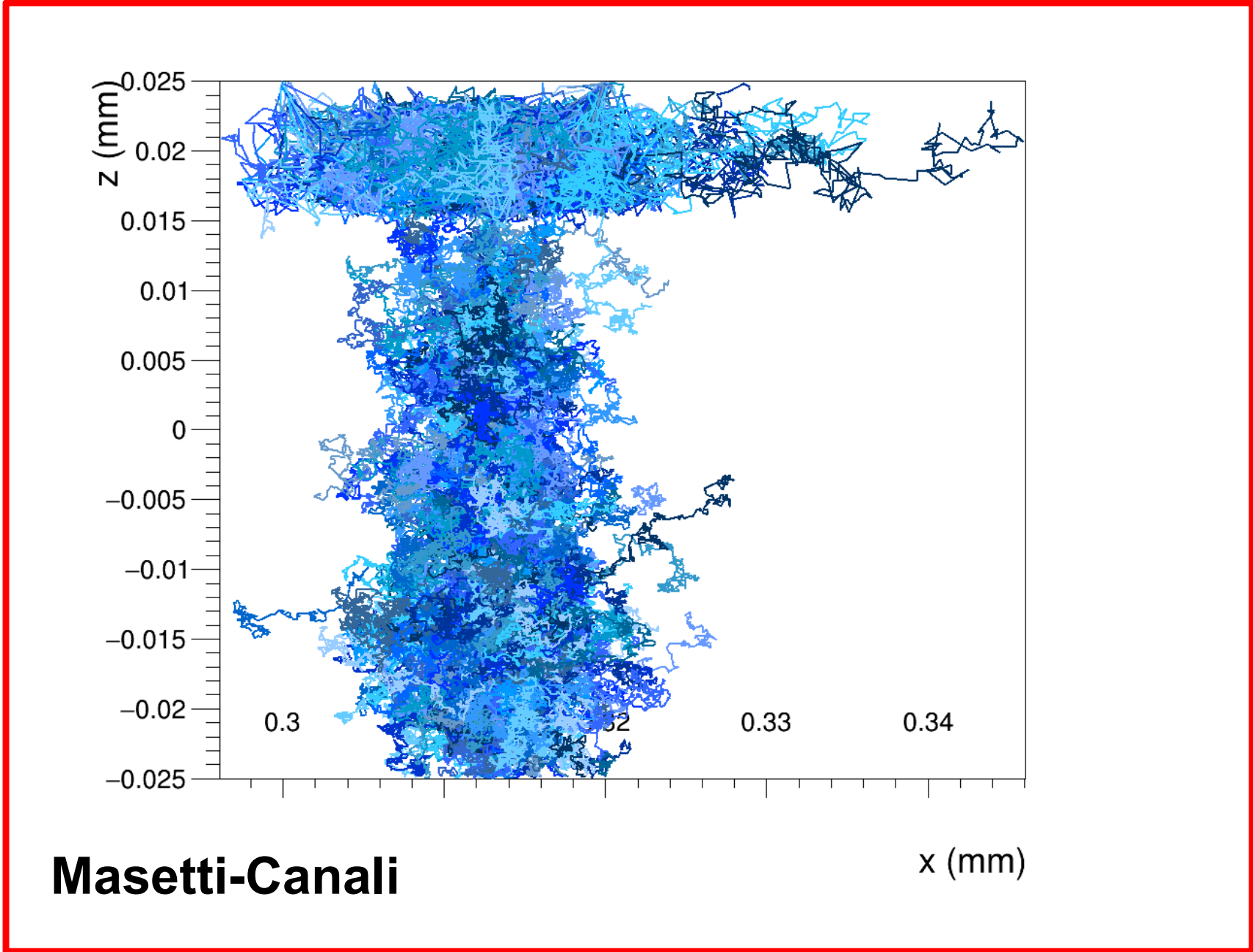
- Closely tied to ALICE developments for ITS3
- Similarly complex device where TCAD is combined with Monte Carlo in allpix²



Propagation models

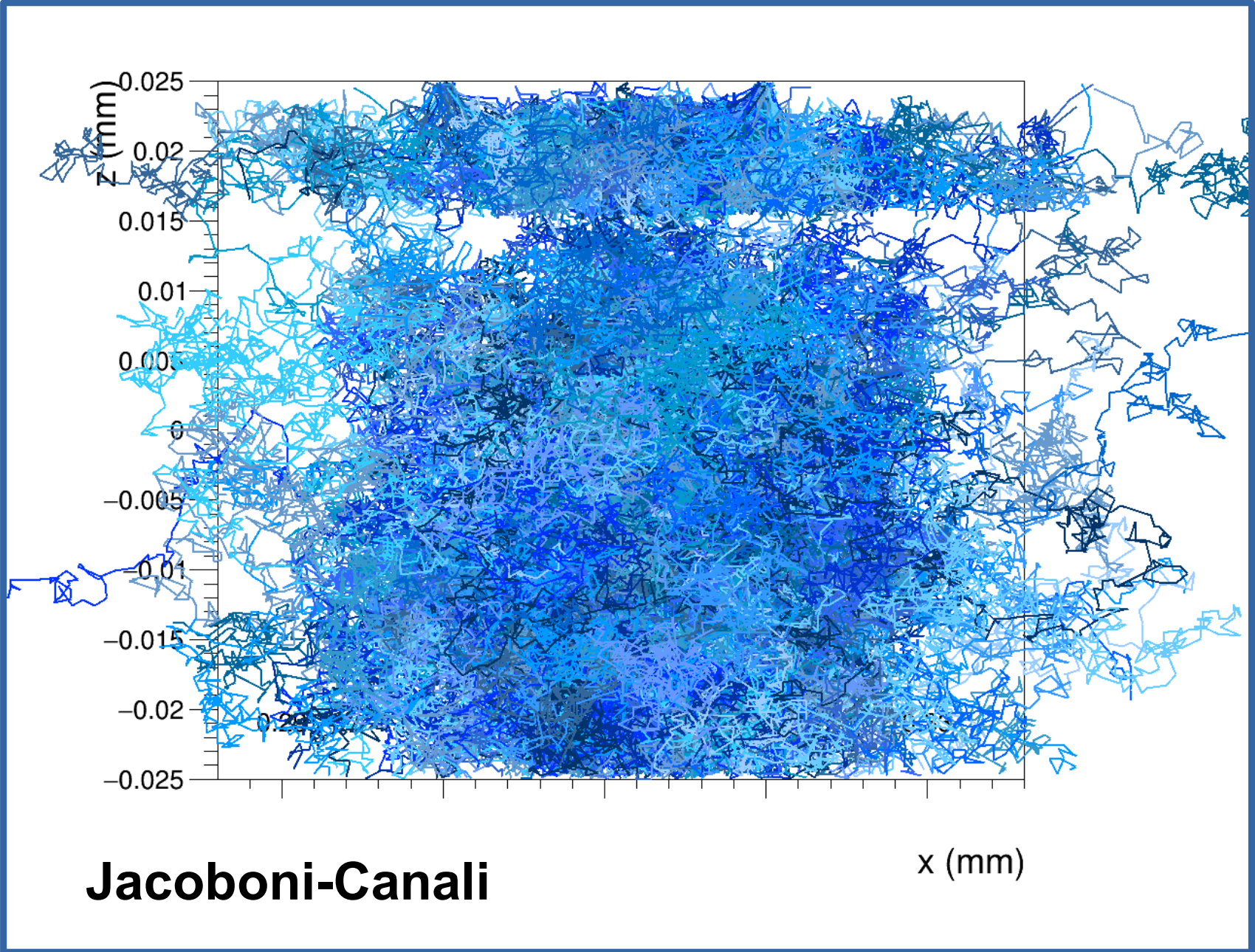
Masetti-Canali

$$\mu(E, N) = \frac{\mu_m(N)}{\left(1 + (\mu_m(N) \cdot E/v_m)^\beta\right)^{1/\beta}}$$

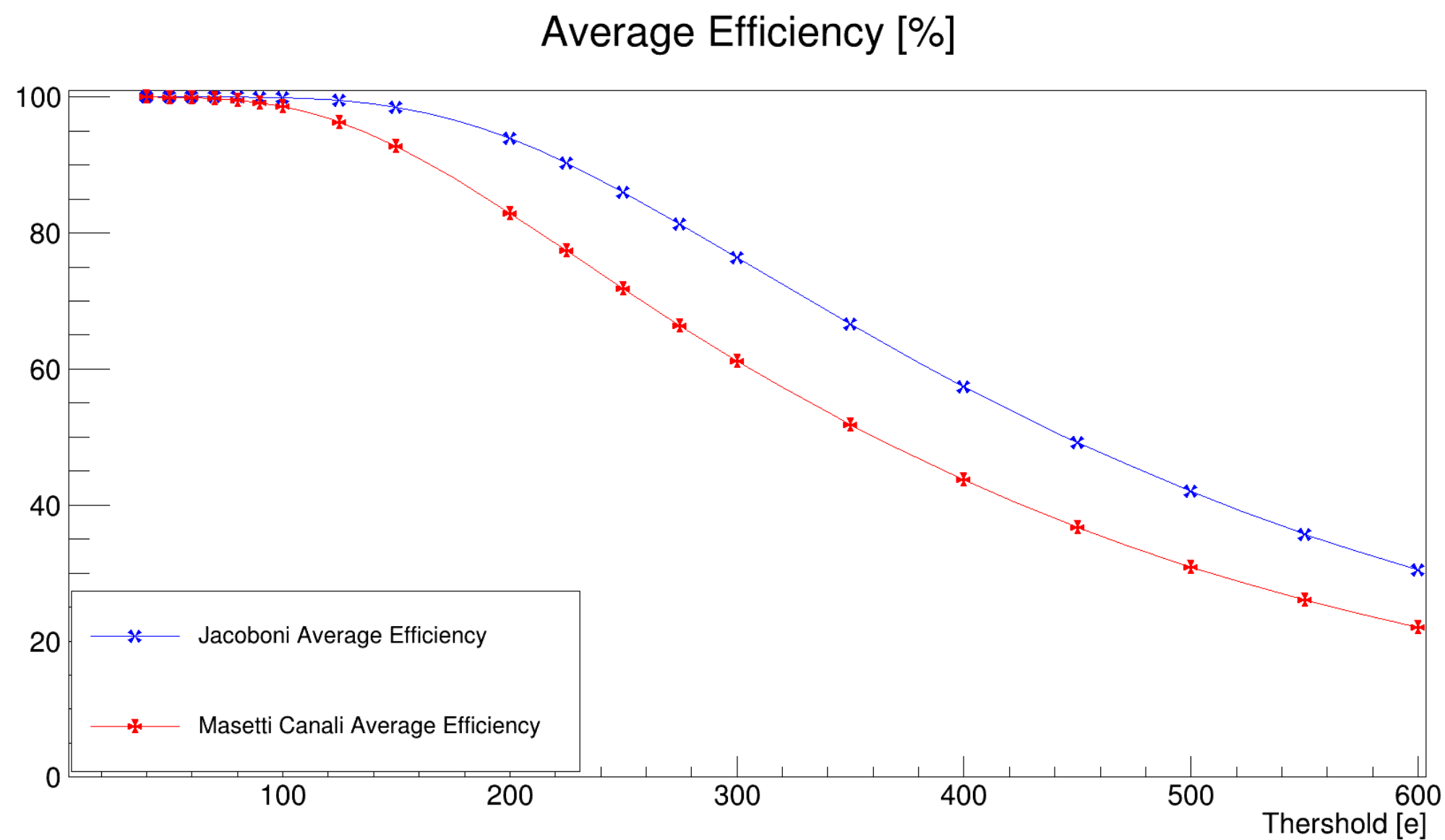
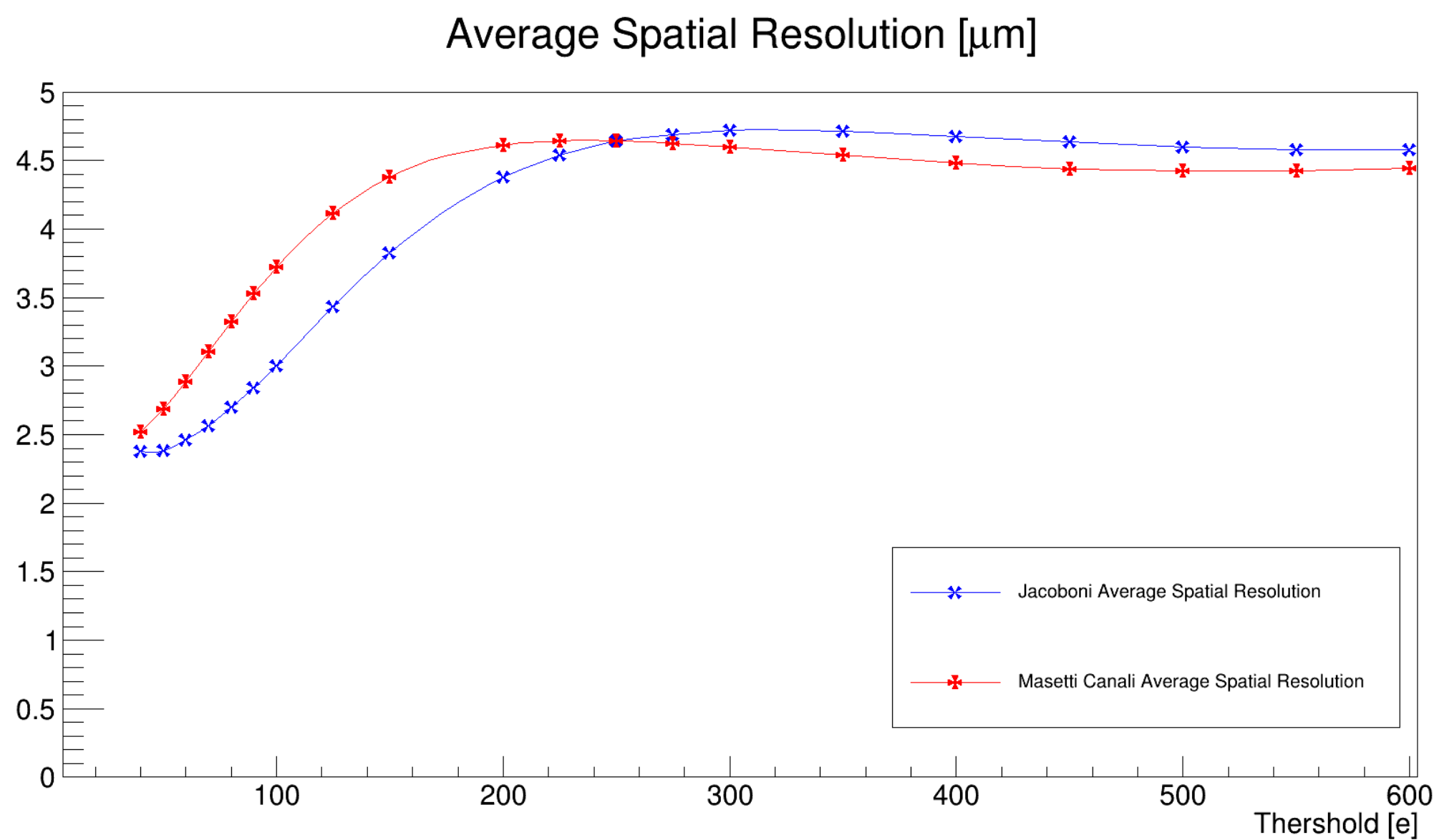


Jacoboni-Canali

$$\mu(E) = \frac{v_m}{E_c} \frac{1}{\left(1 + (E/E_c)^\beta\right)^{1/\beta}},$$

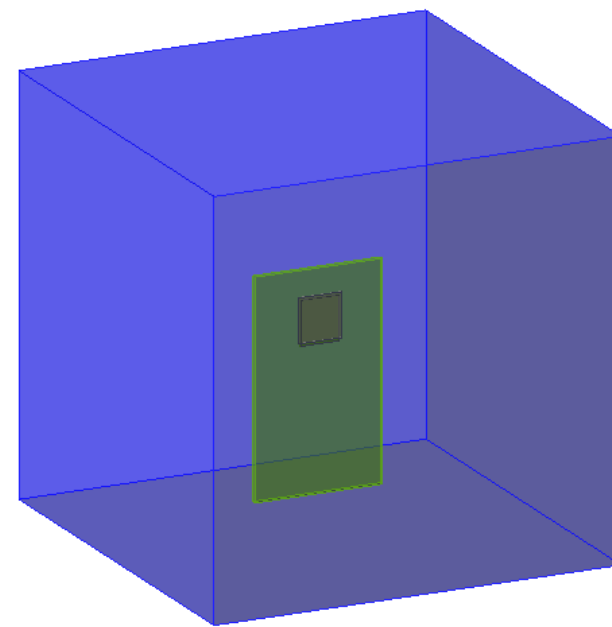



```
mobility_model = "custom"  
mobility_function_electrons = "[0]/[1]/pow(1.0+pow(x/[1],[2]),1.0/[2])"  
mobility_parameters_electrons = 1.0927393e7cm/s, 6729.24V/cm, 1.0916  
mobility_function_holes = "[0]/[1]/pow(1.0+pow(x/[1],[2]),1.0/[2])"  
mobility_parameters_holes = 8.447804e6cm/s, 17288.57V/cm, 1.2081
```

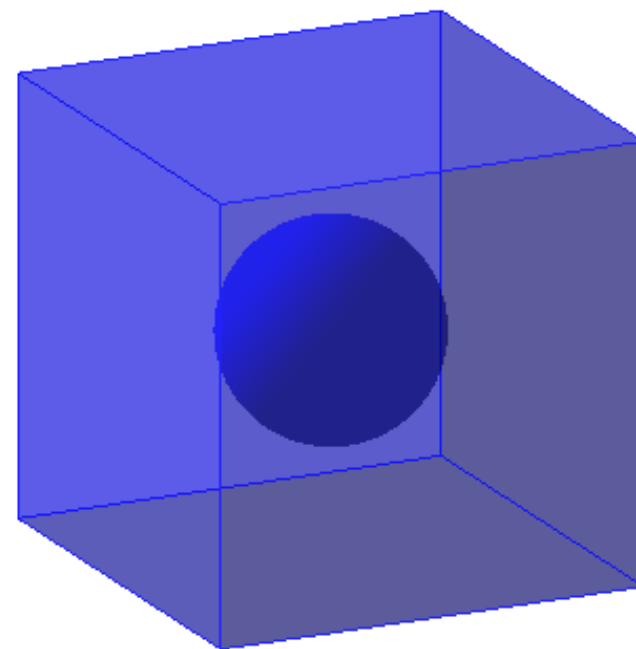


Allpix squared was not originally designed to work with passive materials

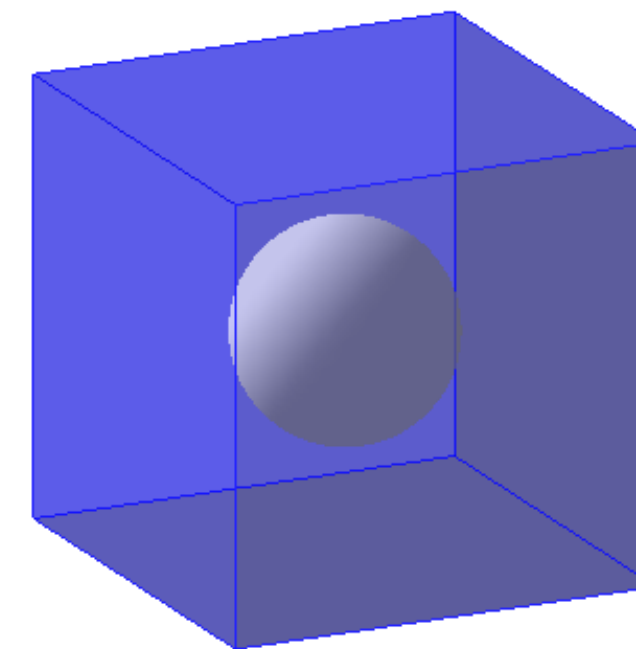
- everything was linked to detector models and geometry
 - MSc project a few years ago to extend the geometry description and give (easy) access to full Geant4 shape list



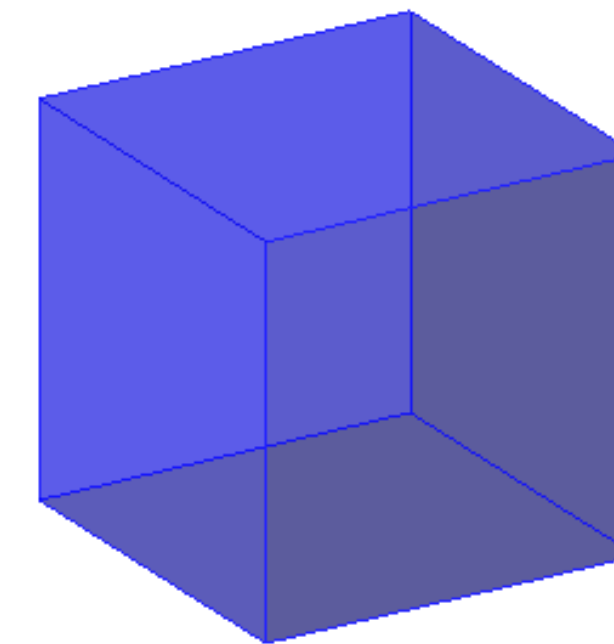
(a) Detector inside a passive material



(b) Passive material inside another passive material



(c) 4.4b with a sphere of World material



(d) 4.4b with a sphere of the same material as the cube

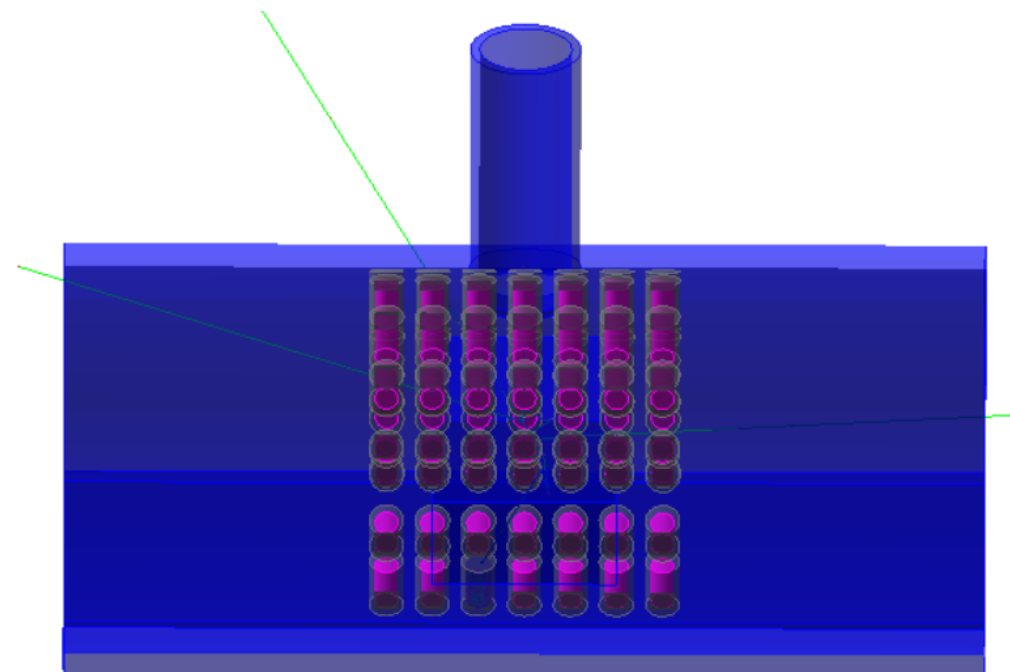
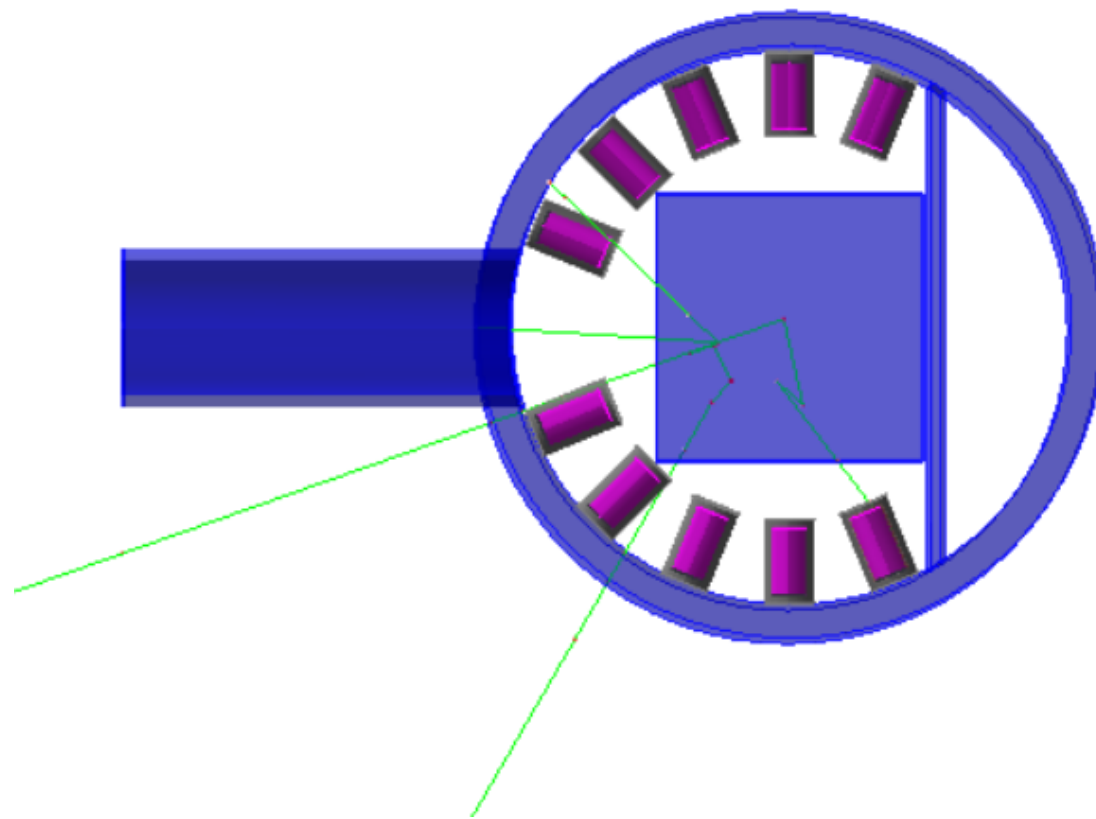
Passive materials

K.Van Den Brandt, Master Thesis, 2017

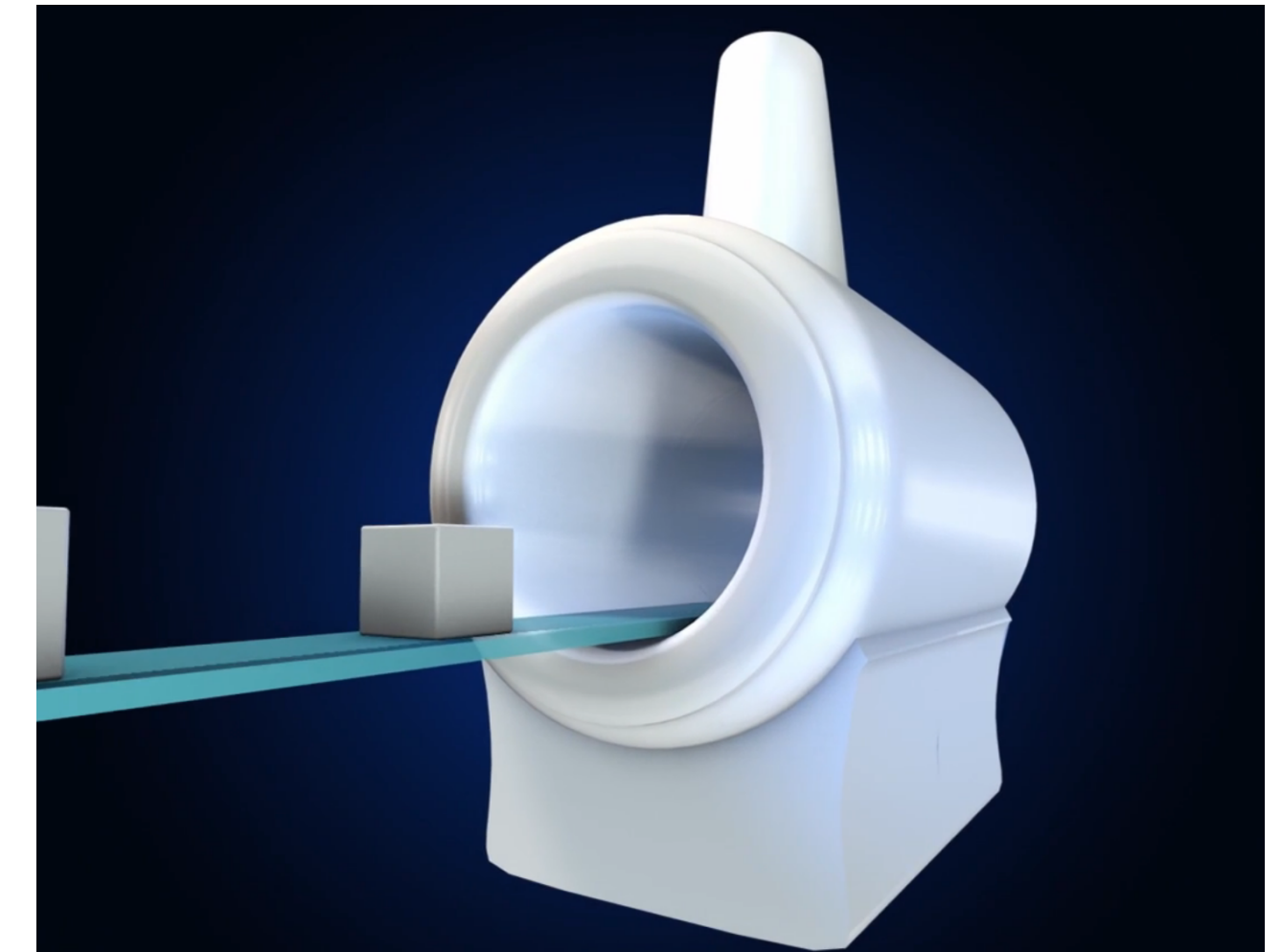
Allpix squared was not originally designed to work with passive materials

- everything was linked to detector models and geometry

- MSc project a few years ago to extend the geometry description and give (easy) access to full Geant4 shape list
- Use cases wildly outside of what we envisaged...



Neutron scanner simulation



Again, another area where we did not originally imagine going was outside of silicon as the sensor material

- Lots of interest from photon science community for additional materials - GaAs, CdTe, etc.
- Required some re-writing of the core to nicely abstract away the differences and allow relevant properties to be set: Geant4 material, mobility models, electric field calculations...

Table 6.1: List of default sensor material properties implemented in Allpix²

Material	Charge Creation Energy [eV]	Fano factor	Sources
Silicon	3.64	0.115	25 , 26
Gallium Arsenide	4.2	0.14	27
Cadmium Telluride	4.43	0.24	28 , 29
Cadmium Zinc Telluride $\text{Cd}_{0.8}\text{Zn}_{0.2}\text{Te}$	4.6	0.14	30 , 31
Diamond	13.1	0.382	32 , 32
Silicon Carbide (4H-SiC)	7.6	0.1	33 , 34

Non-silicon materials: GaAs

```
type = "hybrid"  
number_of_pixels = 256 256  
pixel_size = 55um 55um  
  
sensor_thickness = 500um  
sensor_excess = 1mm  
sensor_material = "GaAs"  
  
bump_sphere_radius = 9.0um  
bump_cylinder_radius = 7.0um  
bump_height = 20.0um  
chip_thickness = 300um  
chip_excess_left = 15um  
chip_excess_right = 15um  
chip_excess_bottom = 2040um  
  
[support]  
thickness = 1.76mm  
size = 47mm 79mm  
offset = 0 -22.25mm
```

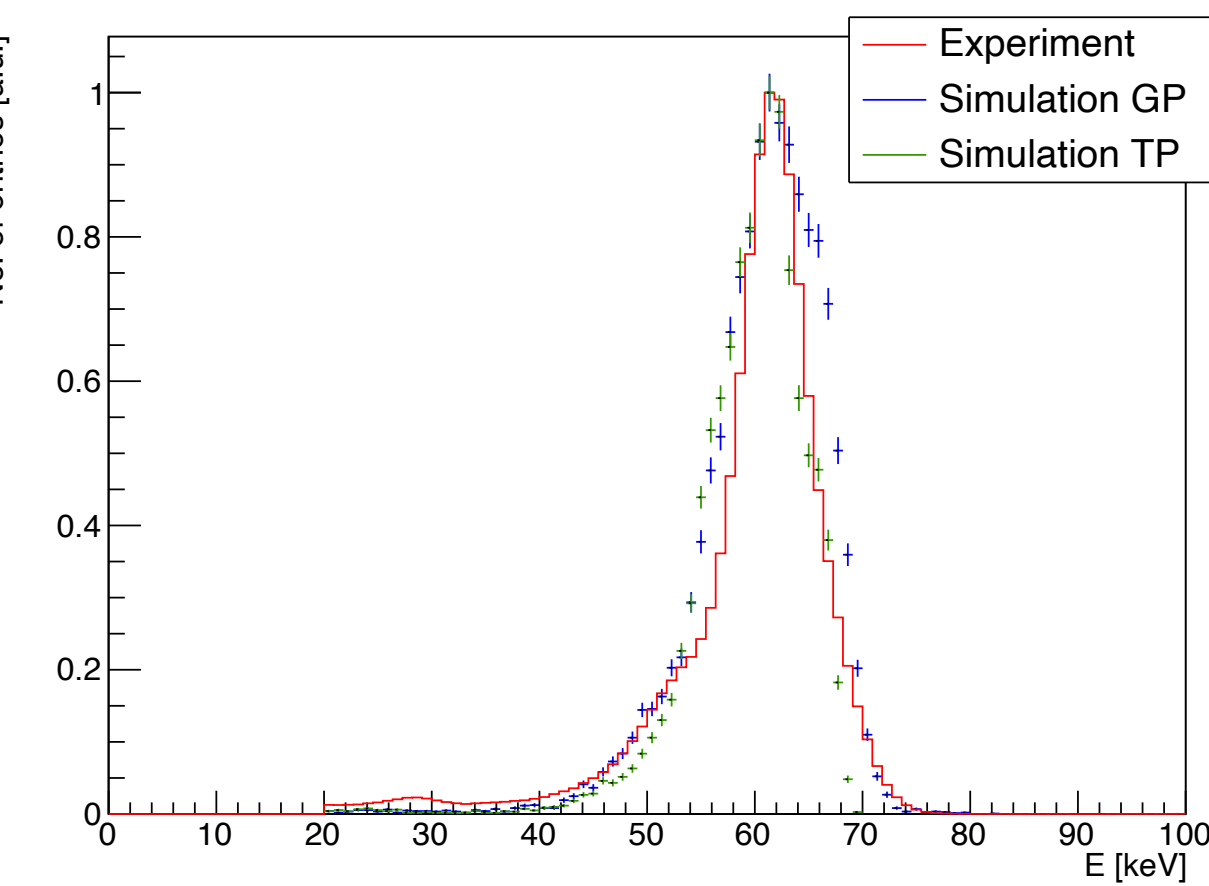
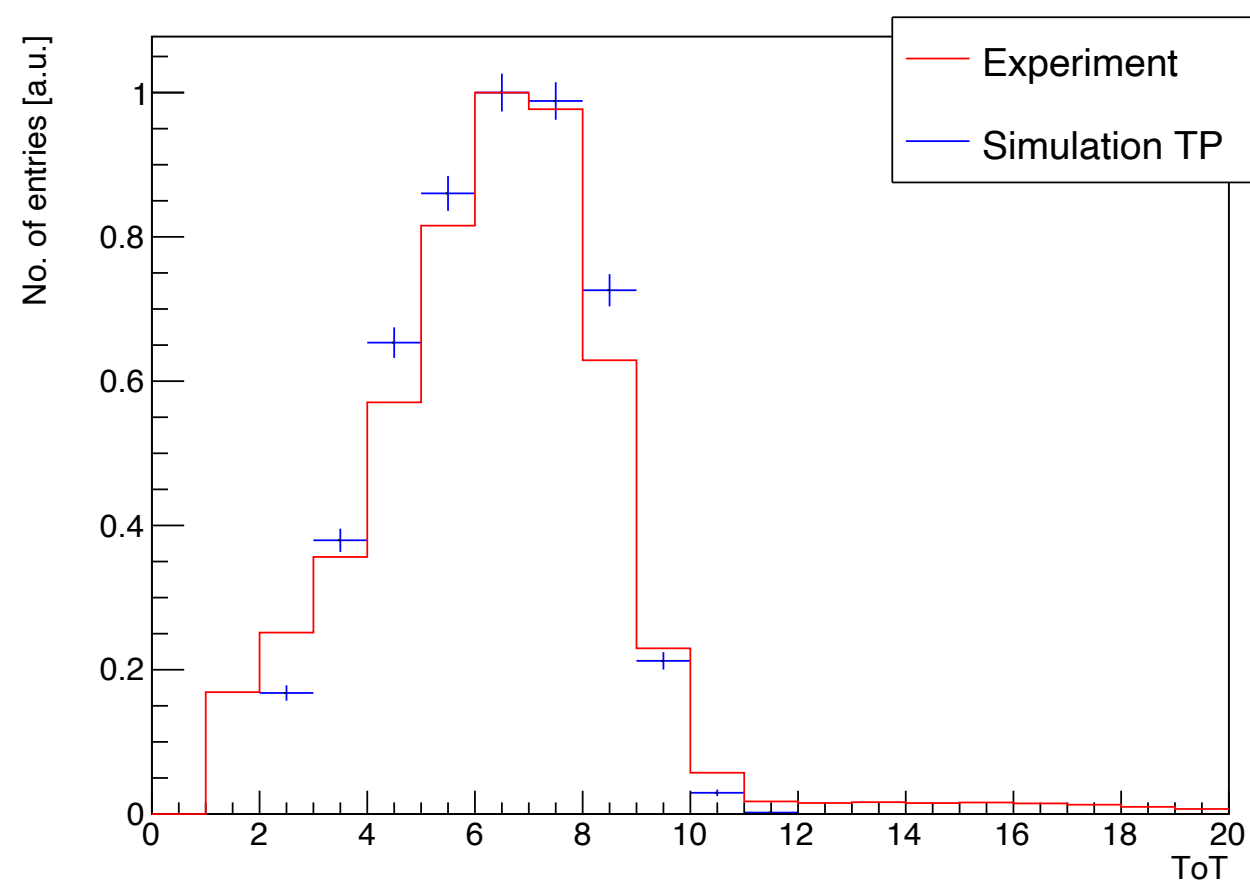
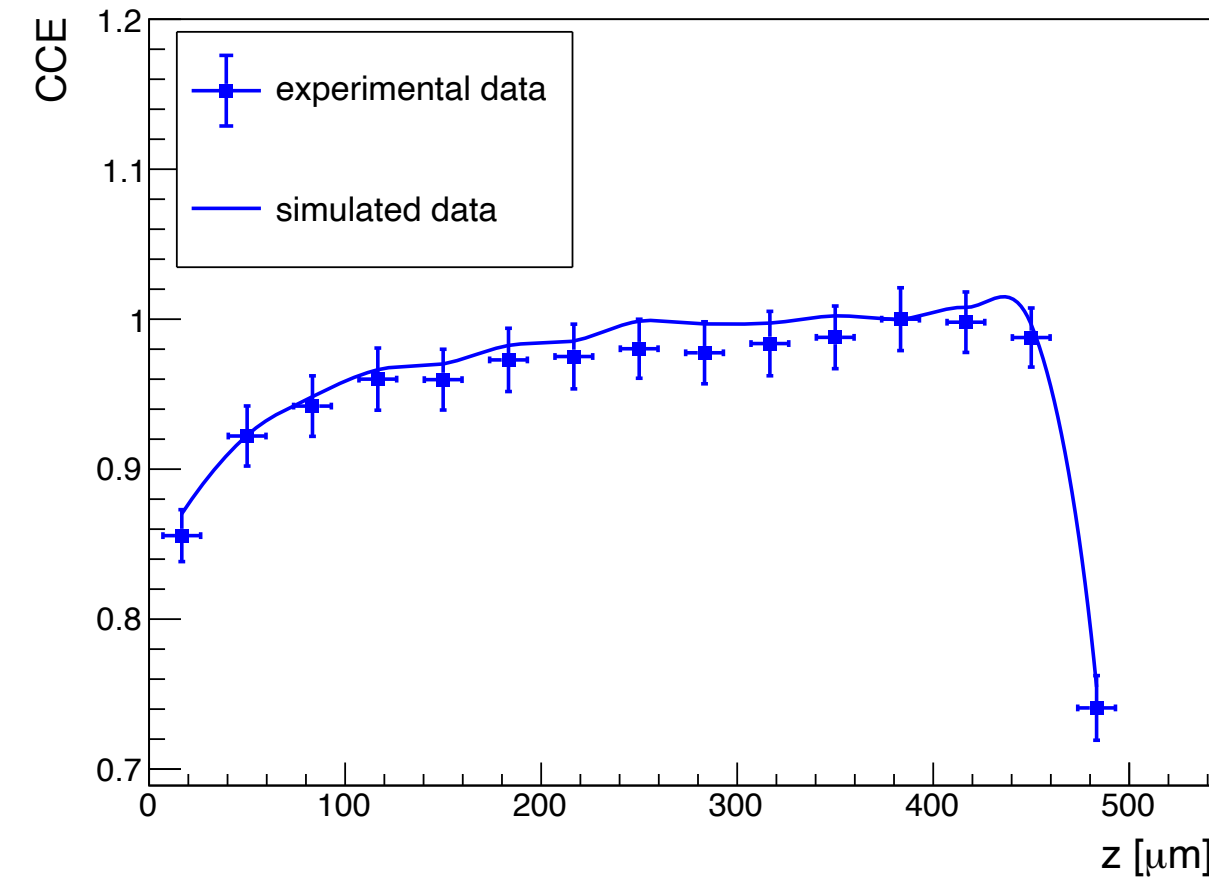
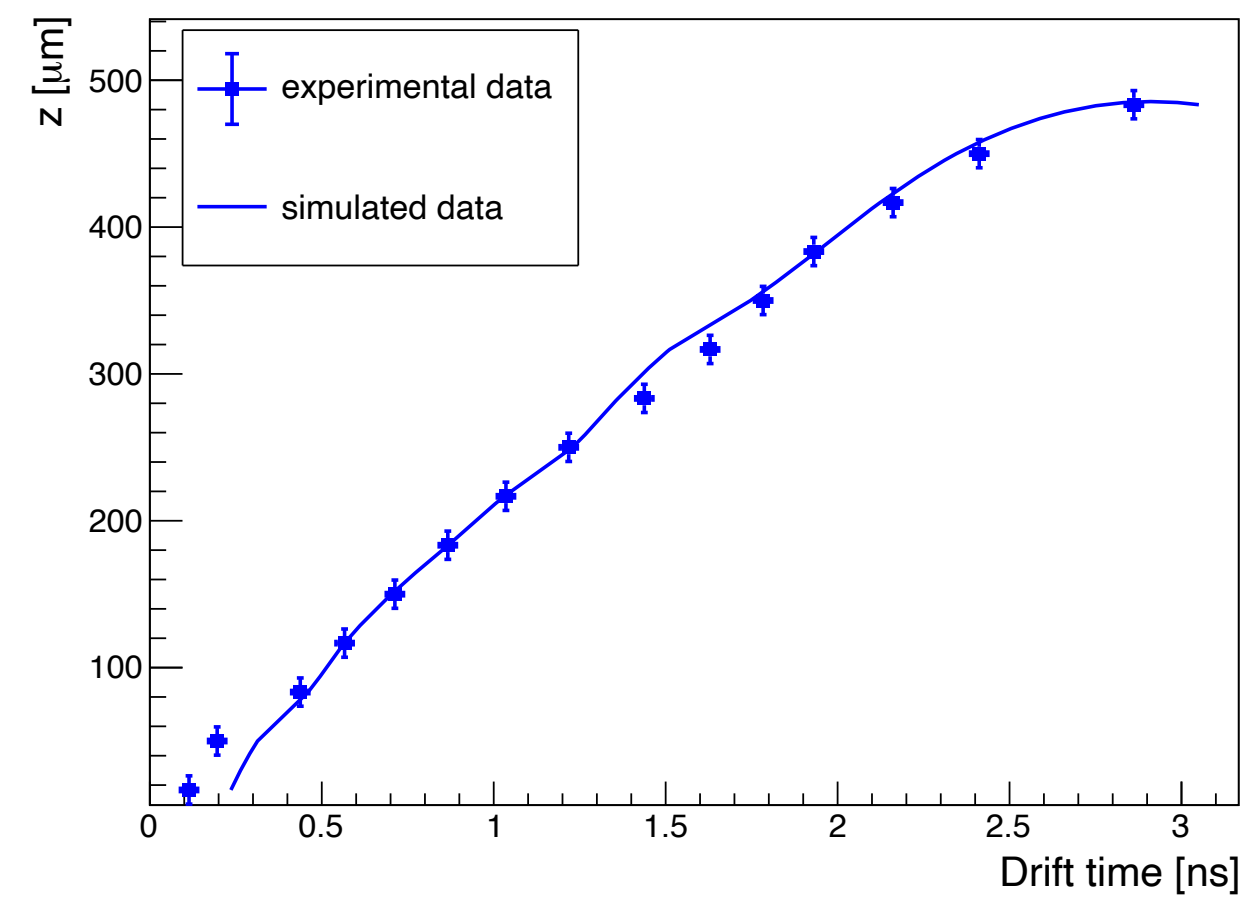
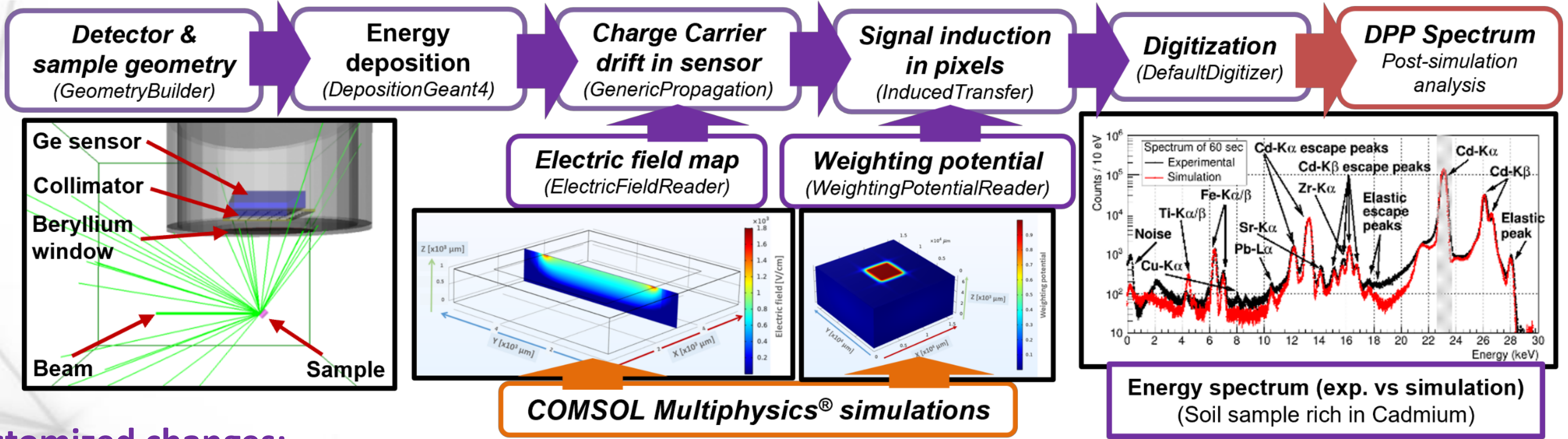


Figure: 8 keV photons spectra.

Figure: of ²⁴¹Am source spectra.



- ✓ The Caughey–Thomas approximation for the low-field carrier mobilities of 4H-SiC was implemented in the Allpix²

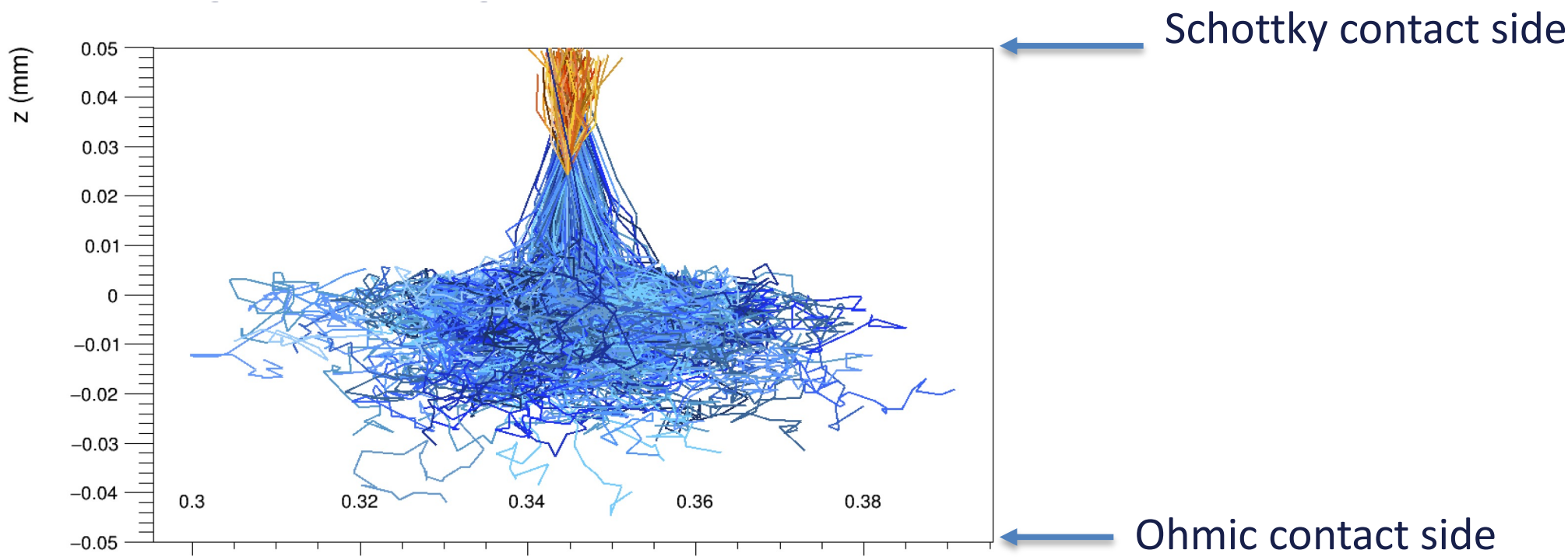
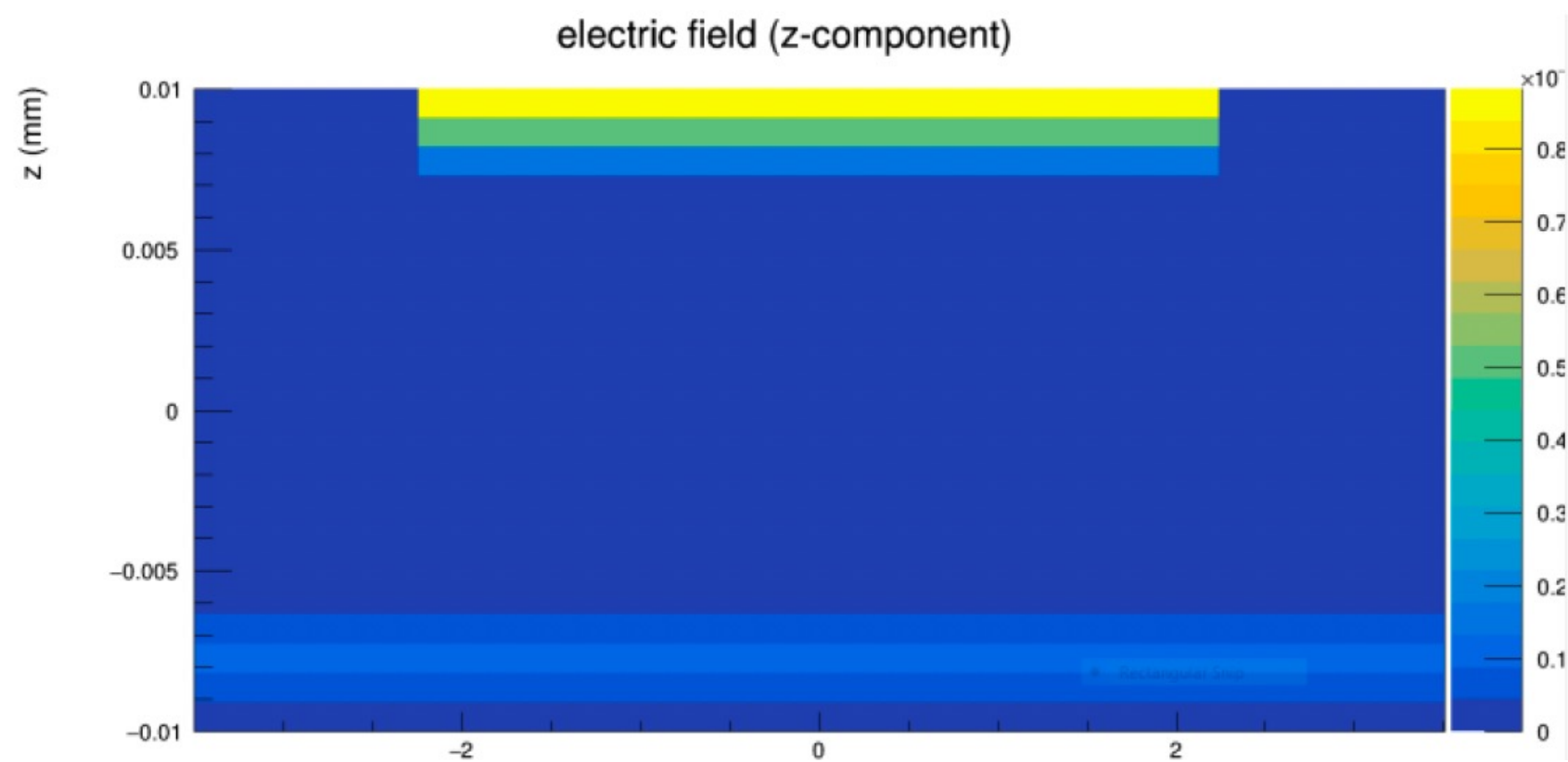
$$\mu_{n,p} = \mu_{n,p}^{min} + \frac{\mu_{n,p}^{max} - \mu_{n,p}^{min}}{1 + \left(\frac{N_i}{N_{n,p}^{crit}}\right)^{\delta_{n,p}}}$$

considering the temperature dependence of the parameters, the equation becomes

$$\mu_{n,p} = \mu_{n,p}^{min,0} \left(\frac{T}{300K}\right)^{\alpha_{n,p}} + \frac{\mu_{n,p}^{max,0} \left(\frac{T}{300K}\right)^{\alpha_{n,p}} - \mu_{n,p}^{min,0} \left(\frac{T}{300K}\right)^{\beta_{n,p}}}{1 + \left(\frac{T}{300K}\right)^{\gamma_{n,p}} \left(\frac{N_i}{N_{n,p}^{crit,0}}\right)^{\delta_{n,p}}}$$

Table 1. Caughey–Thomas fit parameters for carrier mobilities in 4H-SiC

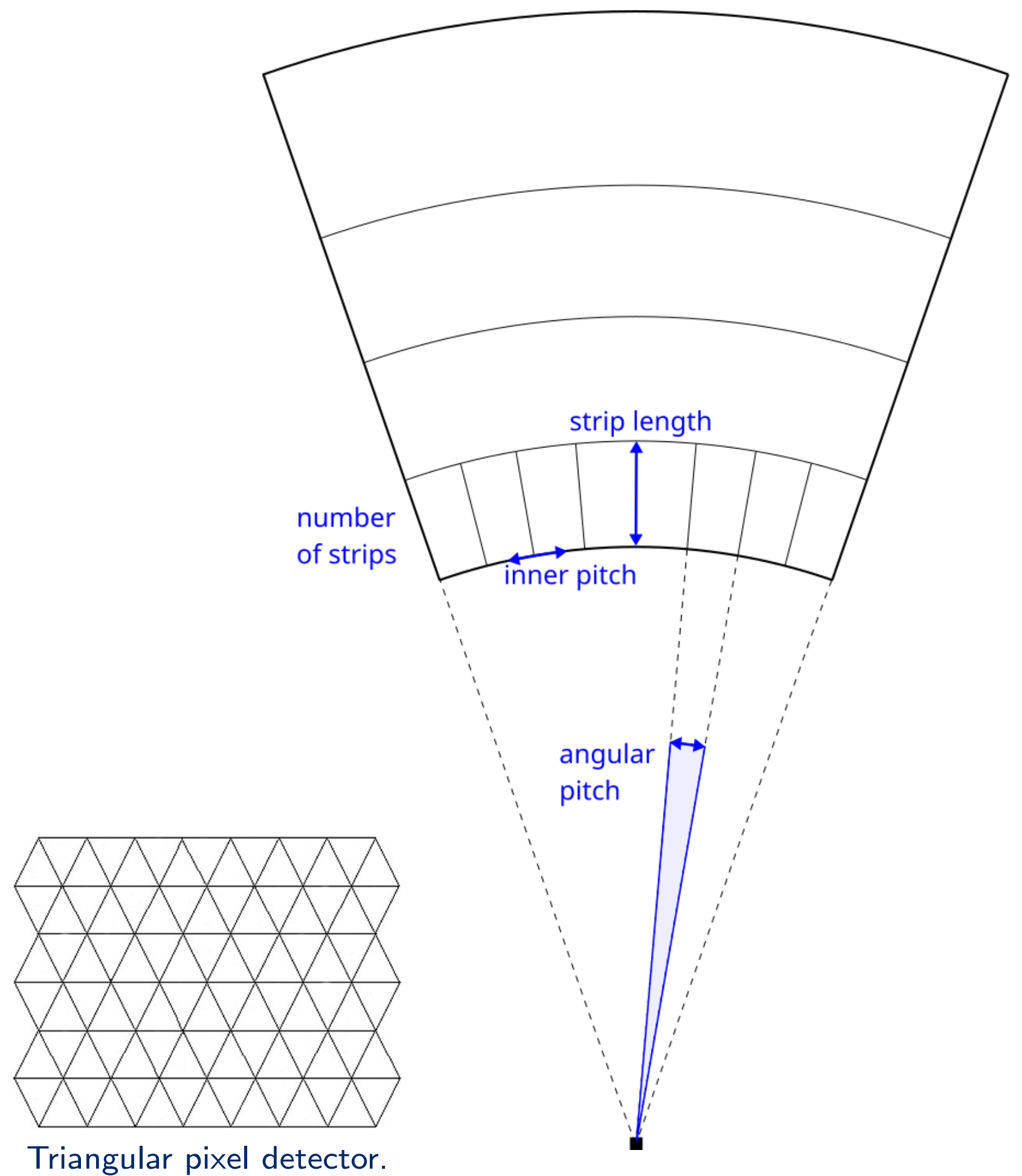
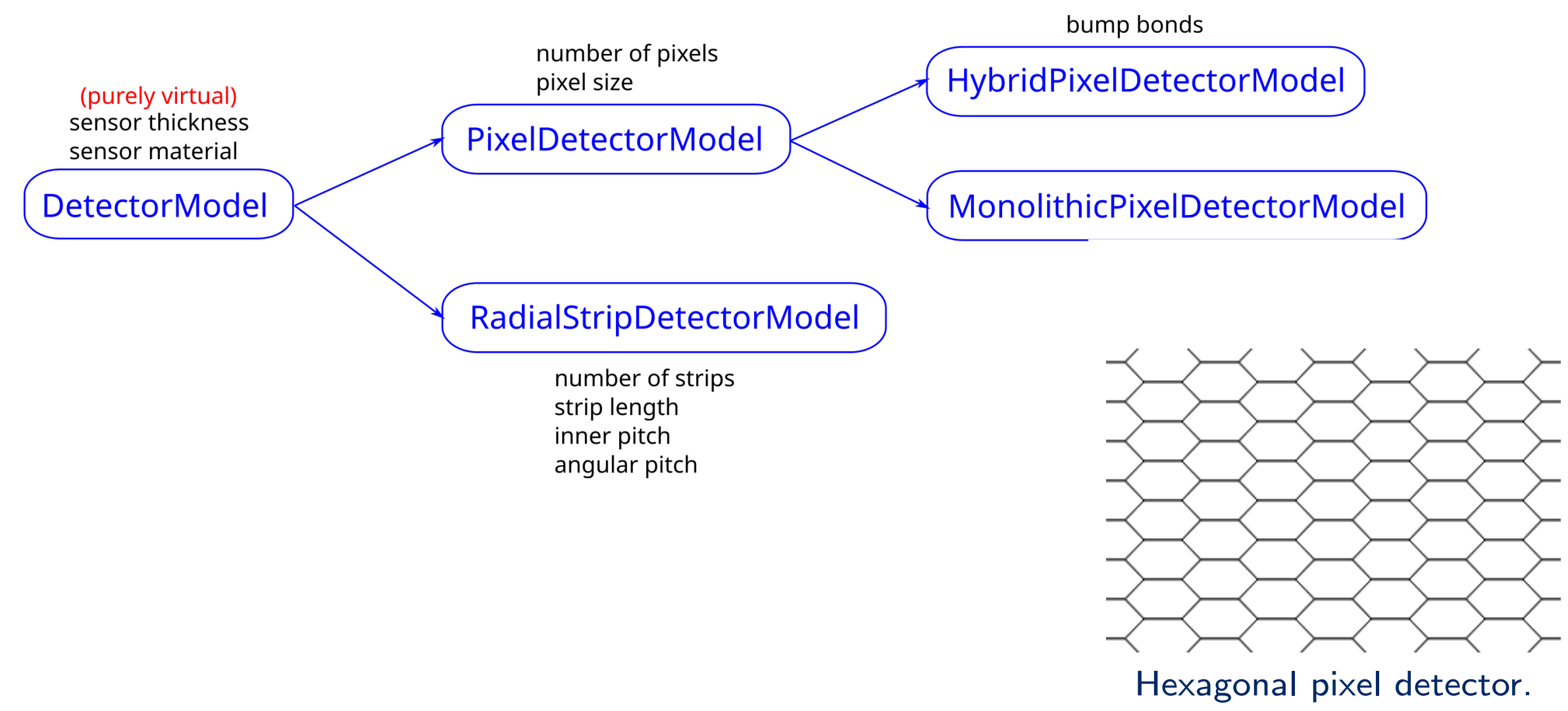
Carrier type	$\mu^{min,0}$ $\text{cm}^2\text{V}^{-1}\text{s}^{-1}$	$\mu^{max,0}$ $\text{cm}^2\text{V}^{-1}\text{s}^{-1}$	$N^{crit,0}$ cm^{-3}	α	β	γ	δ	Reference
Electrons	40	950	2×10^{17}	-0.5	-2.4	-0.76	0.76	[3]
Holes	15.9	124	1.76×10^{19}	0	-1.8	0.00	0.34	[4]



Weird detector geometries

Requirements from different groups have again led to changes in the core to make it flexible enough to cope with different sensor geometries

- This can be in the trivial sense of how the sensor is mapped onto the channel



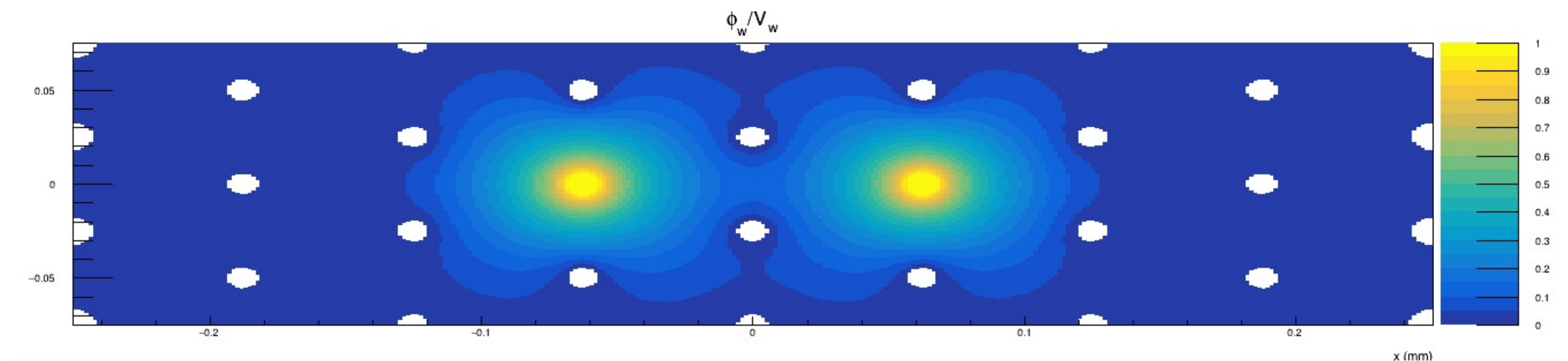
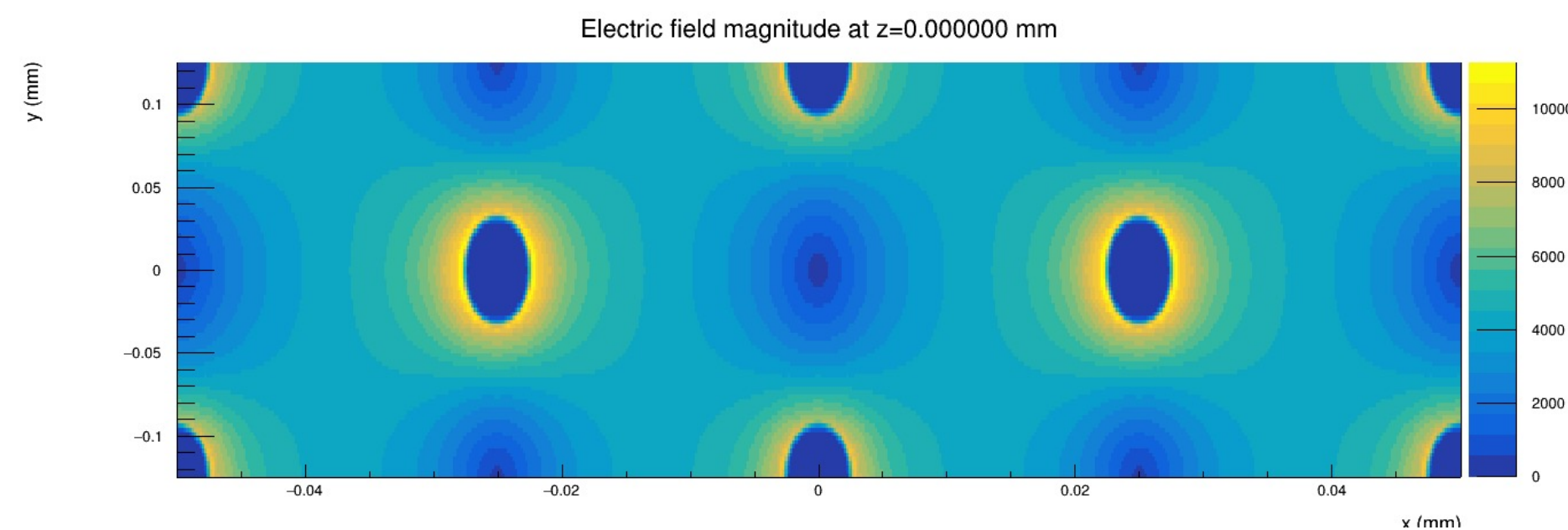
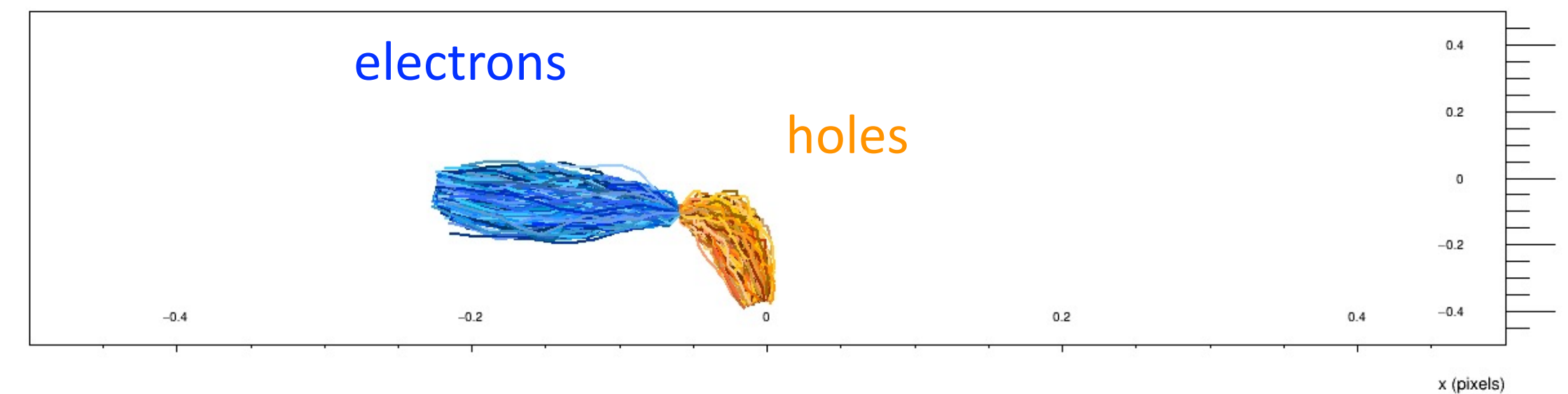
Weird detector geometries

3rd Alpix Squared User Workshop: <https://indico.cern.ch/event/1126306>

Requirements from different groups have again led to changes in the core to make it flexible enough to cope with different sensor geometries

- This can be in the trivial sense of how the sensor is mapped onto the channel
- Or more complicated implant shapes - cf. 3D!

Carriers drifting

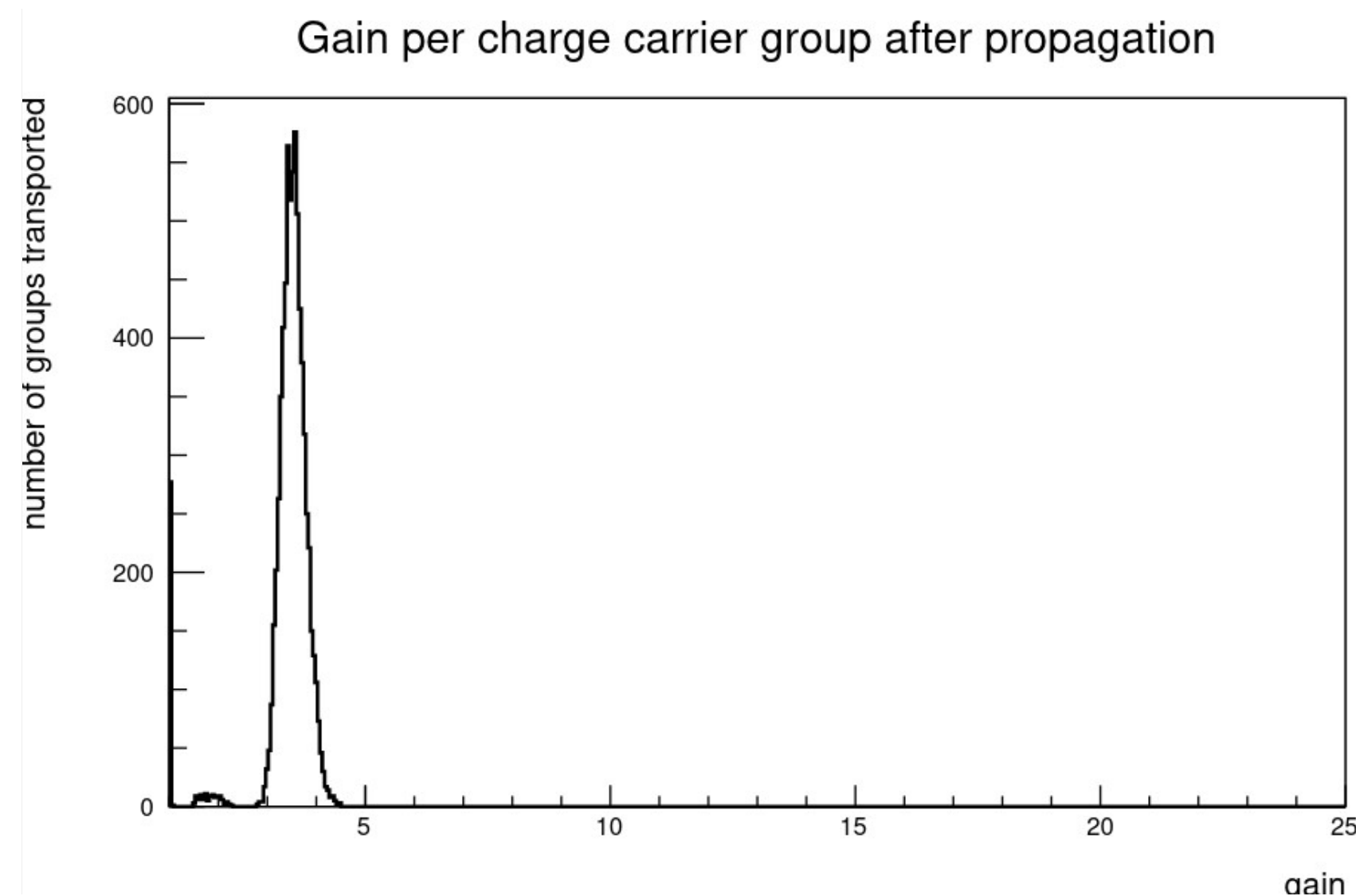


Avalanches in silicon

3rd Allpix Squared User Workshop: <https://indico.cern.ch/event/1126306>

Ongoing work in addressing one of the popular topics in detector development at the moment: charge multiplication

- Multiple models have been implemented and validation is underway



```
[GenericPropagation]  
temperature = 293K  
multiplication_model = "massey"  
multiplication_threshold = 100kV/cm
```

Implementation of charge multiplication through impact ionization underway

- Multiple models available, selection via configuration file:
 - Massey
 - van Overstraeten-de Man
 - Okuto-Crowell
 - Bologna
- Fully documented in user manual
- **Implementation in Allpix² completed**, undergoing testing, Comparison with Weightfield2 & TCAD simulations

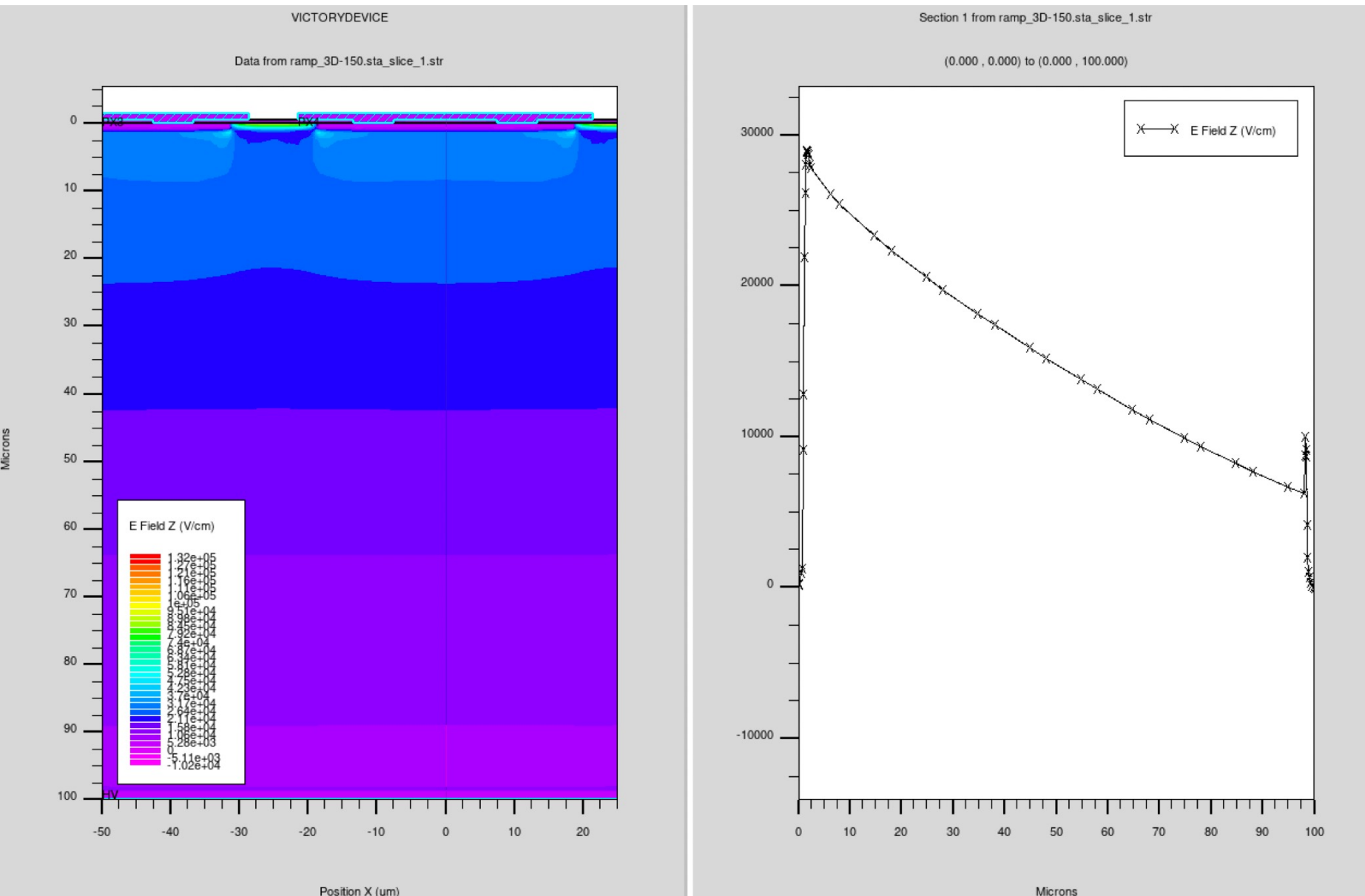
Similarly, implementation of radiation damage is required to study devices for hadron collider experiments

- Clearly contains two components:
 - Modified TCAD electric field profile (proper modelling of space charge effects)
 - Charge trapping implementation in allpix²

2016 JINST 11 P04023

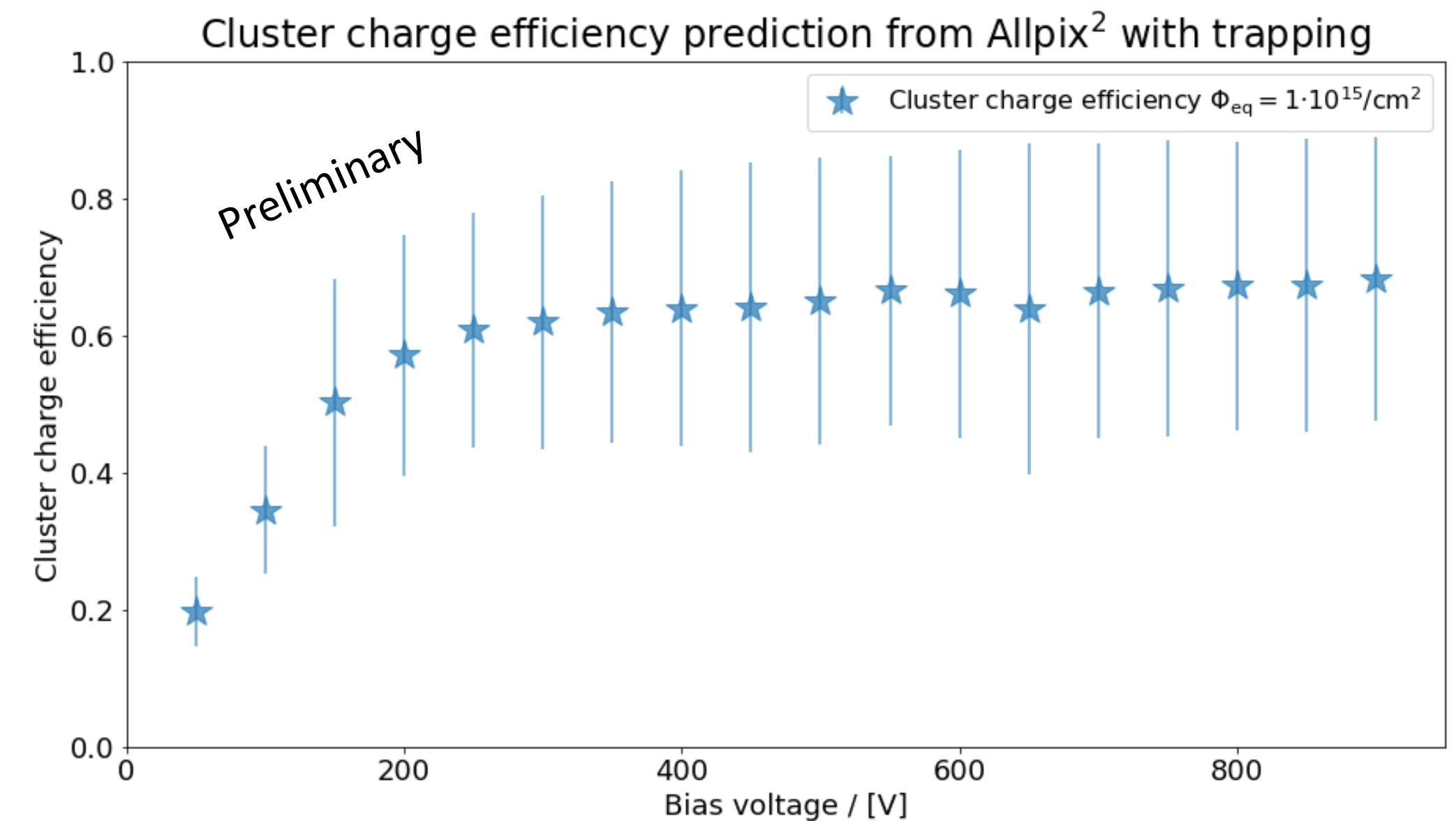
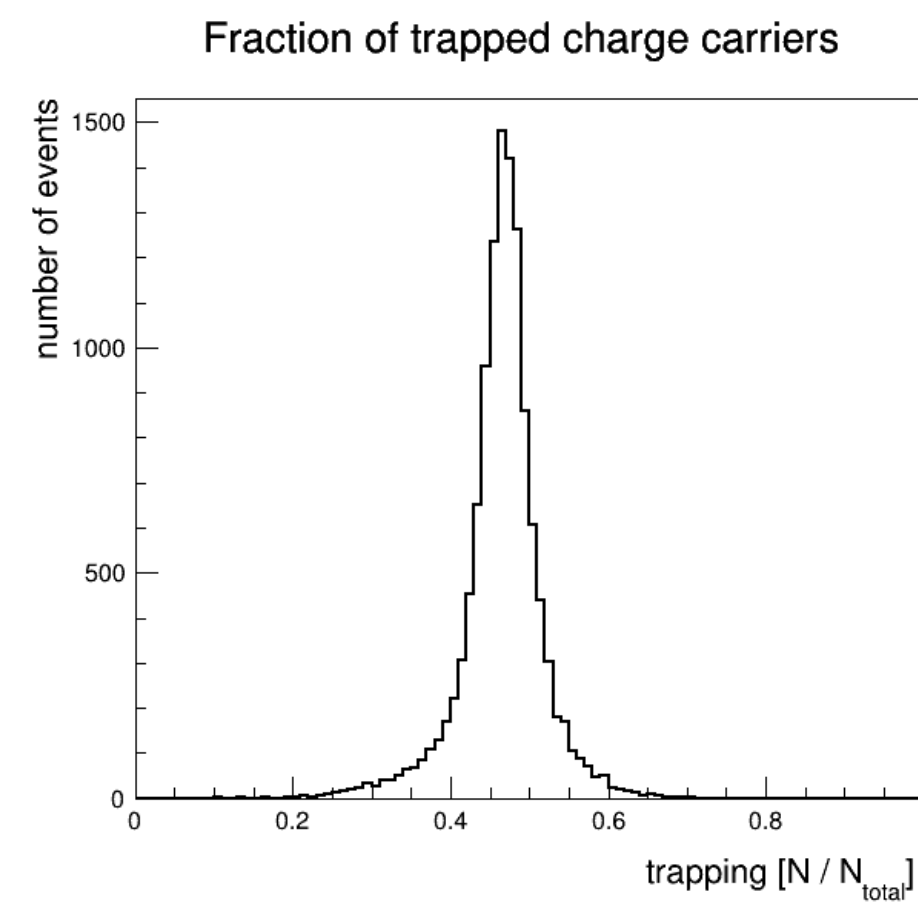
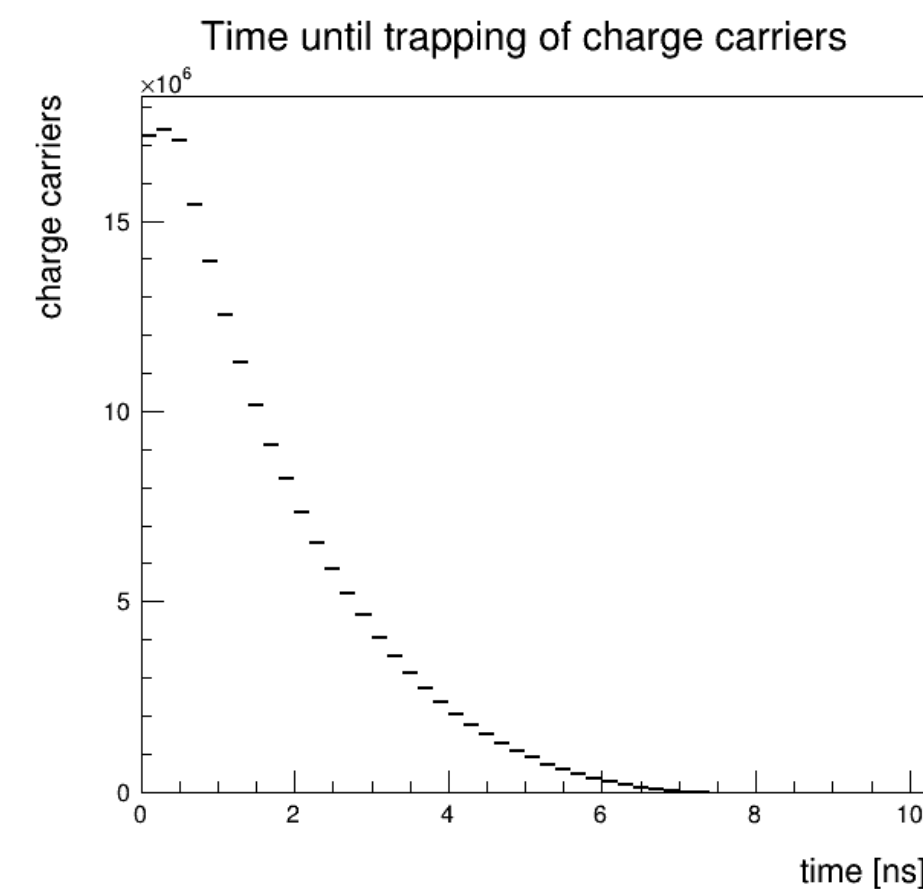
Table 1. The key parameter values used in the Synopsys device simulation. These include: donor and acceptor concentrations, N_D and N_A , and their electron and hole capture cross sections, $\sigma_{D,A}^{e,h}$, for silicon sensors after irradiation with 23 GeV protons (top rows) [11], and for sensors after irradiation with 23 MeV protons (bottom rows) [8].

ϕ_{neq} [10^{14}neq/cm^2]	N_A [10^{14}cm^{-3}]	N_D [10^{14}cm^{-3}]	σ_A^e [10^{-15}cm^2]	σ_D^e [10^{-15}cm^2]	σ_A^h [10^{-15}cm^2]	σ_D^h [10^{-15}cm^2]
2 (23 GeV) [11]	6.8	10	6.6	6.6	1.65	6.6
6 (23 GeV) [11]	16	40	6.6	6.6	1.65	1.65
12 (23 GeV) [11, 18] ⁴	30	69	3.8	3.8	0.94	0.94
24 (23 GeV) [11, 18] ⁴	61	138	3.8	3.8	0.94	0.94
5 (23 MeV) [8]	4.2	15	10	10	10	10
10 (23 MeV) [8]	12.5	52	10	10	10	10



Similarly, implementation of radiation damage is required to study devices for hadron collider experiments

- Clearly contains two components:
 - Modified TCAD electric field profile (proper modelling of space charge effects)
 - Charge trapping implementation in allpix²



A whole lot more

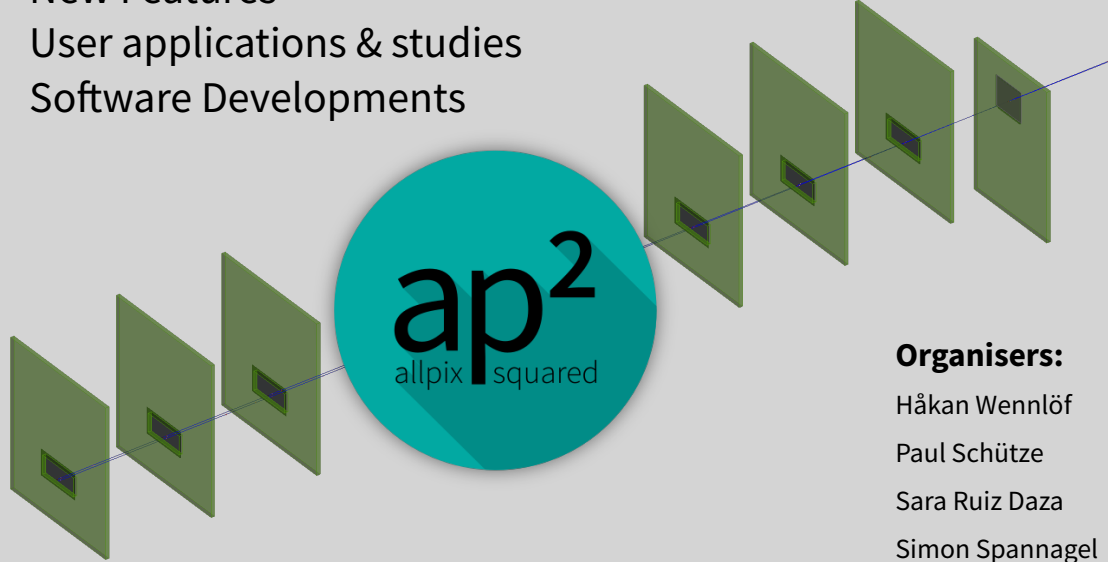
4th Allpix² User Workshop



22-23 May 2023

DESY, Hamburg, Germany

New Features
User applications & studies
Software Developments



Organisers:
Håkan Wennlöf
Paul Schütze
Sara Ruiz Daza
Simon Spannagel
Stephan Lachnit

Abstract deadline: **31 March 2023**
Registration deadline: **14 April 2023**

For more information, please scan the QR code
or go to: <https://indico.cern.ch/e/apsqws4>



Contact: allpix-squared-workshop@cern.ch



5th Allpix² User Workshop



22-24 May 2024

University of Oxford, UK



Organisers
Daniel Hynds
Paul Schütze
Simon Spannagel
Håkan Wennlöf



<https://indico.cern.ch/e/apsqws5>

Abstract deadline: **22 April**
Registration deadline: **4 May**

Allpix² User Workshop 6th edition



7-9 May 2025

Nikhef, Amsterdam, the Netherlands

Abstract deadline: **31 March 2025**
Registration deadline: **9 April 2025**

Organisers:
Håkan Wennlöf
Isis Hobus
Paul Schütze
Petja Skomina
Simon Spannagel

allpix-squared-workshop@cern.ch



<https://indico.cern.ch/e/apsqws6>



7th Allpix² User Workshop



28 - 30 April 2026

MBI-ÖAW, Vienna, Austria

Abstract Deadline: **23 March 2026**
Registration Deadline: **15 April 2026**

Organisers
Brigitte De Monte
Håkan Wennlöf
Paul Schütze
Roberto Russo
Sebastian Onder
Simon Spannagel
Thomas Bergauer

allpix-squared-workshop@cern.ch



<https://indico.cern.ch/e/apsqws7>



Austrian Academy of Sciences



16 June 2026 - <https://indico.global/e/ukinst2026>

daniel.hynds@physics.ox.ac.uk

41

Summary

A lot of functionality has been implemented in `allpix2`, after the initial work to set it up as a platform for semiconductor detector simulations

- A modern and self-contained framework that allows developers to work on solid-state physics

It is straightforward to get up-and-running on `cvmfs` or with a local installation for development work. The first port of call is always the (extensive) user manual:

- <https://allpix-squared.docs.cern.ch>

Support available via email on the dedicated mailing list, on Mattermost and on the dedicated forum:

- allpix-squared-users@cern.ch
- <https://mattermost.web.cern.ch/allpix2>
- <https://cern.ch/allpix-squared-forum>

Look out for the announcement of the Allpix Squared workshop 2027!

