Monte Carlo simulations - Allpix^2

Daniel Hynds

Updated simulation configuration

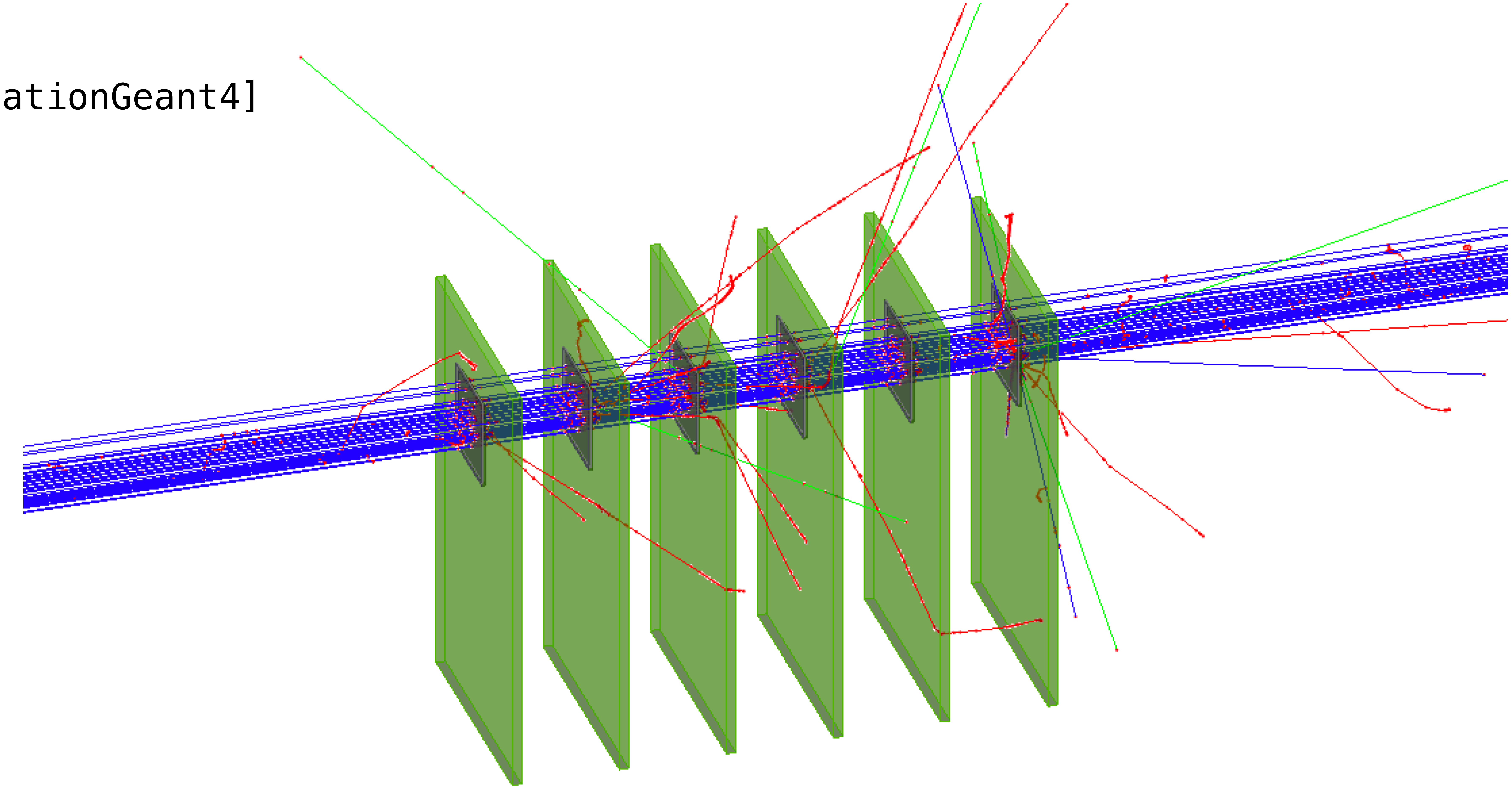
```
[Allpix]
number_of_events = 1000
detectors_file = "tutorial-geometry.conf"
log_level = "Warning"

[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
```


daniel.hynds@physics.ox.ac.uk

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

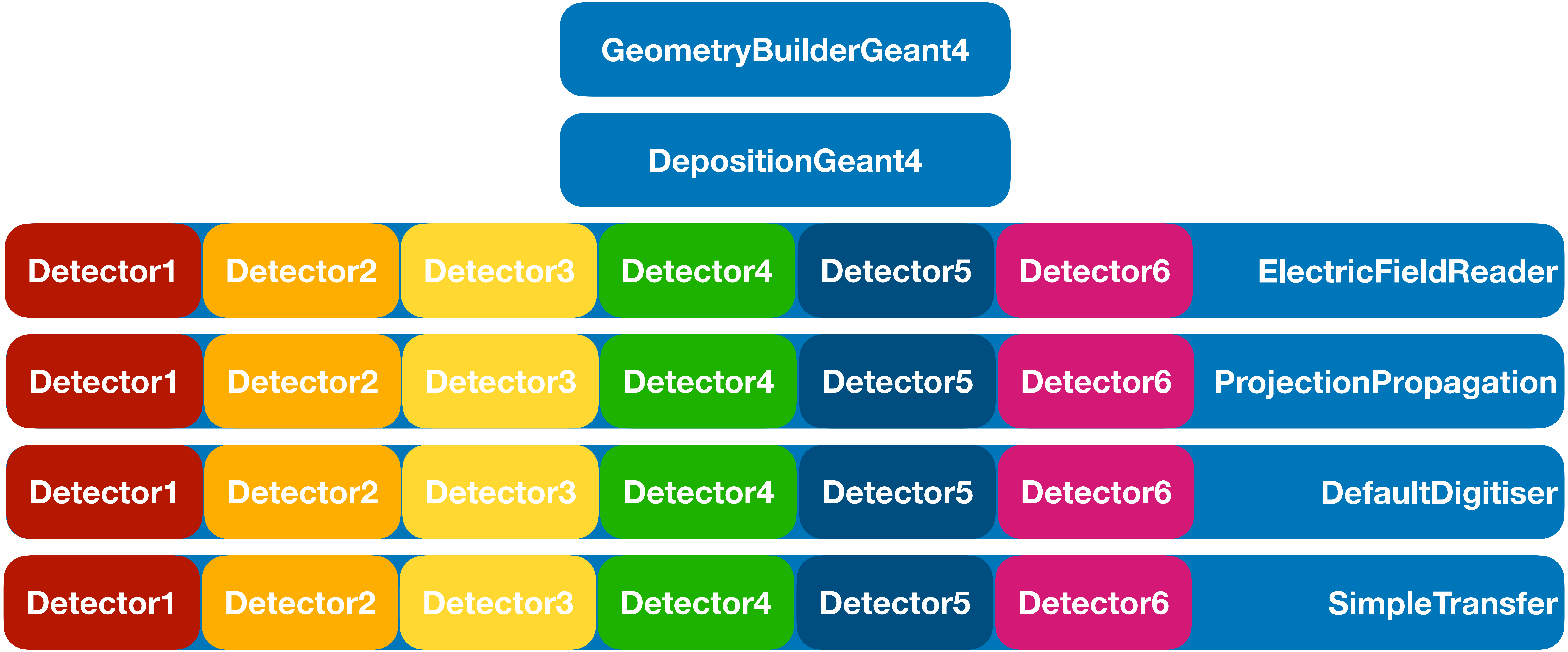
- This allows some measure of multithreading to be used to improve simulation times - all detectors can be run in parallel

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

A single detector chain

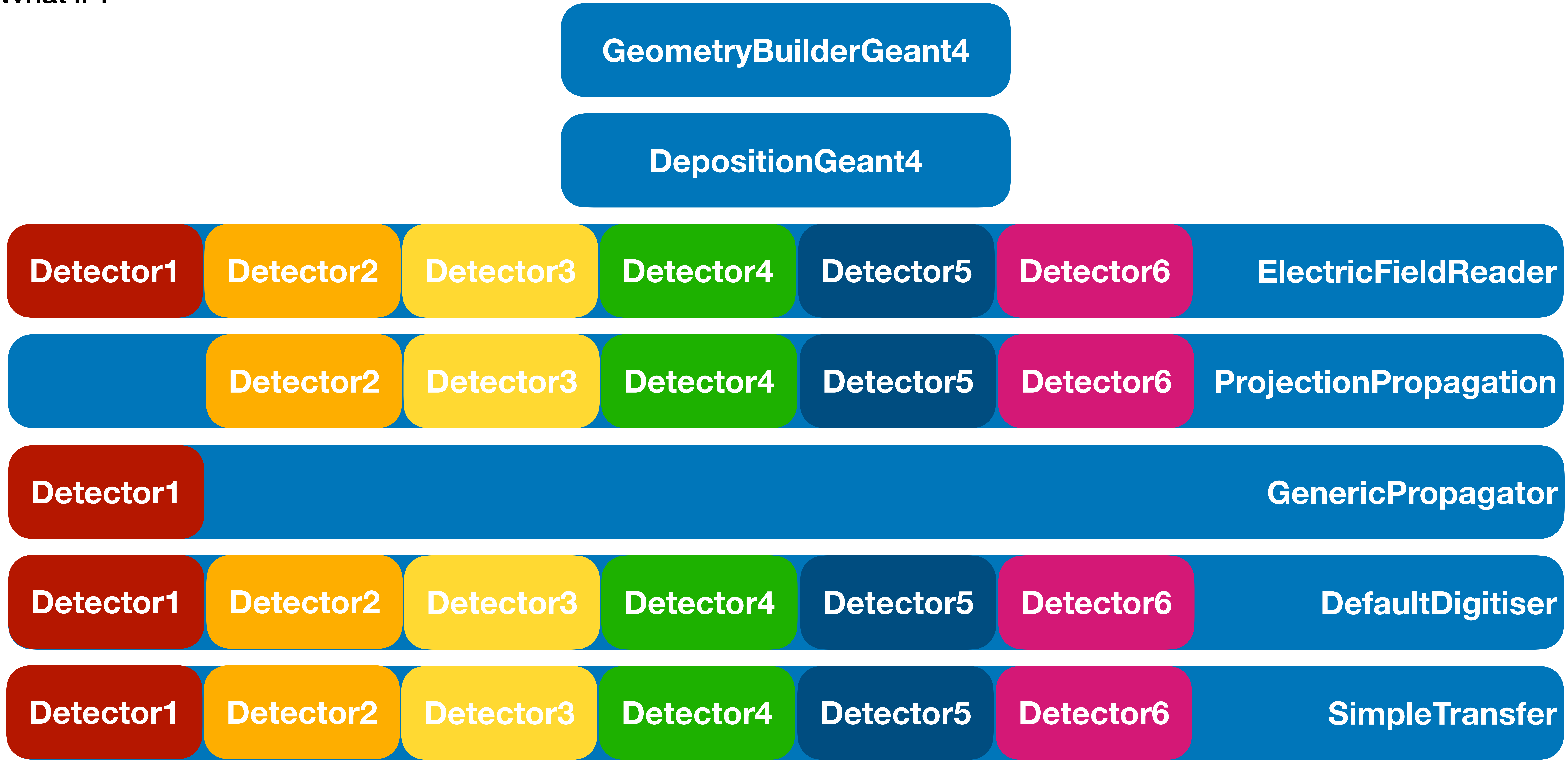
- GeometryBuilderGeant4
- DepositionGeant4
- ElectricFieldReader
- ProjectionPropagation
- DefaultDigitiser
- SimpleTransfer

A multi-detector chain



daniel.hynds@physics.ox.ac.uk

What if ?


GeometryBuilderGeant4
DepositionGeant4
Detector1
Detector2
Detector3
Detector4
Detector5
Detector6
ElectricFieldReader
Detector2
Detector3
Detector4
Detector5
Detector6
ProjectionPropagation
Detector1
GenericPropagator
Detector1
Detector2
Detector3
Detector4
Detector5
Detector6
DefaultDigitiser
Detector1
Detector2
Detector3
Detector4
Detector5
Detector6
SimpleTransfer

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
```

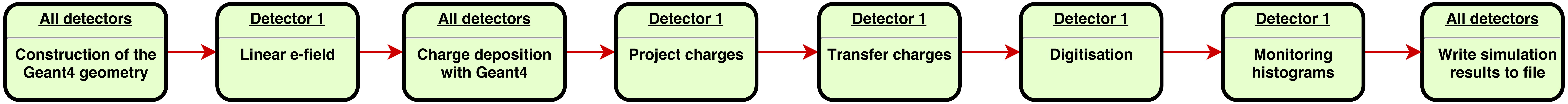

daniel.hynds@physics.ox.ac.uk

Specifying detector type/name

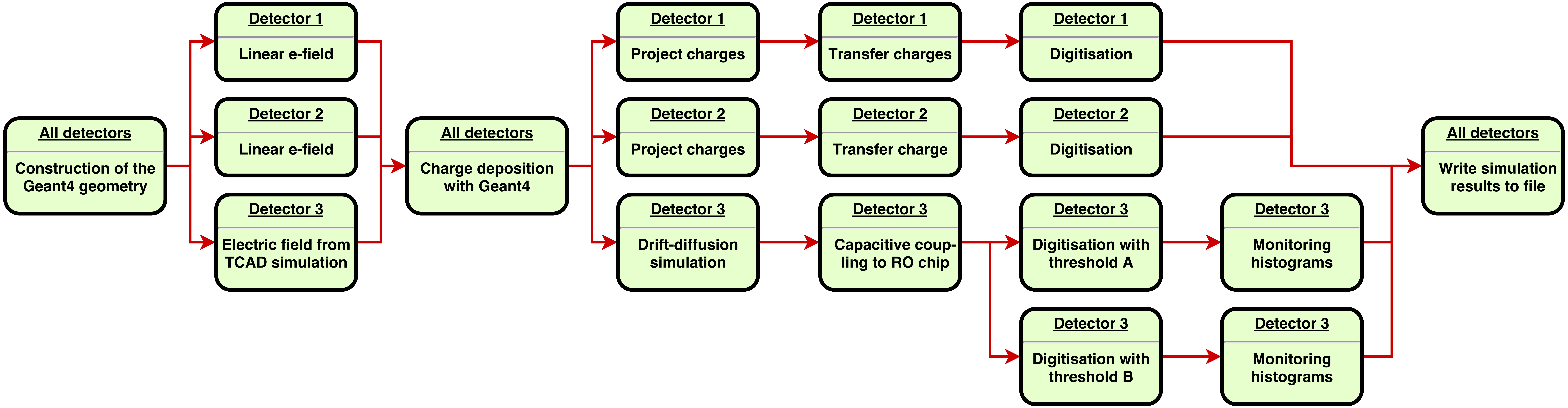
```
[Allpix]
number_of_events = 1000
detectors_file = "tutorial-geometry.conf"
log_level = "Warning"

[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



daniel.hynds@physics.ox.ac.uk

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

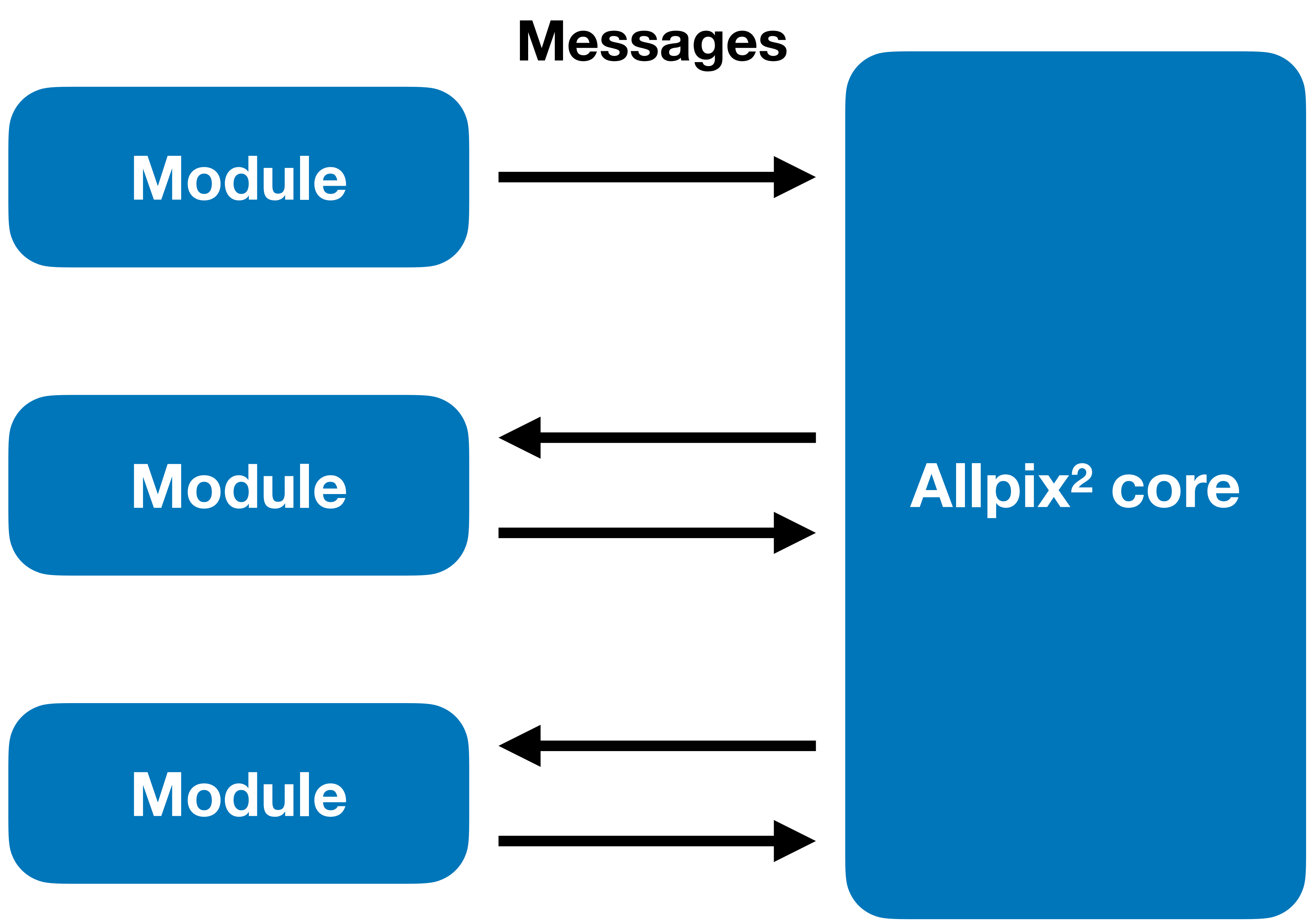
Modules exist entirely standalone in allpix-squared

- 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.



daniel.hynds@physics.ox.ac.uk

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.

```
#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>;
```

And are trivial to extend

daniel.hynds@physics.ox.ac.uk

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

Initialise variables/ histograms etc.

Main code, executed each event

TutorialExampleModule.cpp

```cpp
/**
 * @file
 * @brief Implementation of TutorialExample module
 *
 * @copyright Copyright (c) 2017-2023 CERN and the Allpix Squared authors.
 * This software is distributed under the terms of the MIT License, copied verbatim in the file "LICENSE.md".
 * In applying this license, CERN does not waive the privileges and immunities granted to it by virtue of its status as an
 * Intergovernmental Organization or submit itself to any jurisdiction.
 * SPDX-License-Identifier: MIT
 */

#include "TutorialExampleModule.hpp"

#include <string>
#include <utility>

#include "core/utils/log.h"

using namespace allpix;

TutorialExampleModule::TutorialExampleModule(Configuration& config, Messenger* messenger, std::shared_ptr<Detector> detector)
    : Module(config, detector), detector_(std::move(detector)), messenger_(messenger) {

    // Allow multithreading of the simulation. Only enabled if this module is thread-safe. See manual for more details.
    // allow_multithreading();

    // Set a default for a configuration parameter, this will be used if no user configuration is provided:
    config_.setDefault<int>("setting", 13);

    // Parsing of the parameter "setting" into a member variable for later use:
    setting_ = config_.get<int>("setting");

    // Messages: register this module with the central messenger to request a certaintype of input messages:
    messenger_->bindSingle<DepositedChargeMessage>(this, MsgFlags::REQUIRED);
}

void TutorialExampleModule::initialize() {

        // In this simple case we just print the name of this detector:
        LOG(DEBUG) << "Detector with name " << detector_->getName();
}

void TutorialExampleModule::run(Event* event) {

    // Messages: Fetch the (previously registered) messages for this event from the messenger:
    auto message = messenger_->fetchMessage<DepositedChargeMessage>(this, 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]
...
```


daniel.hynds@physics.ox.ac.uk

A few other features - MC history

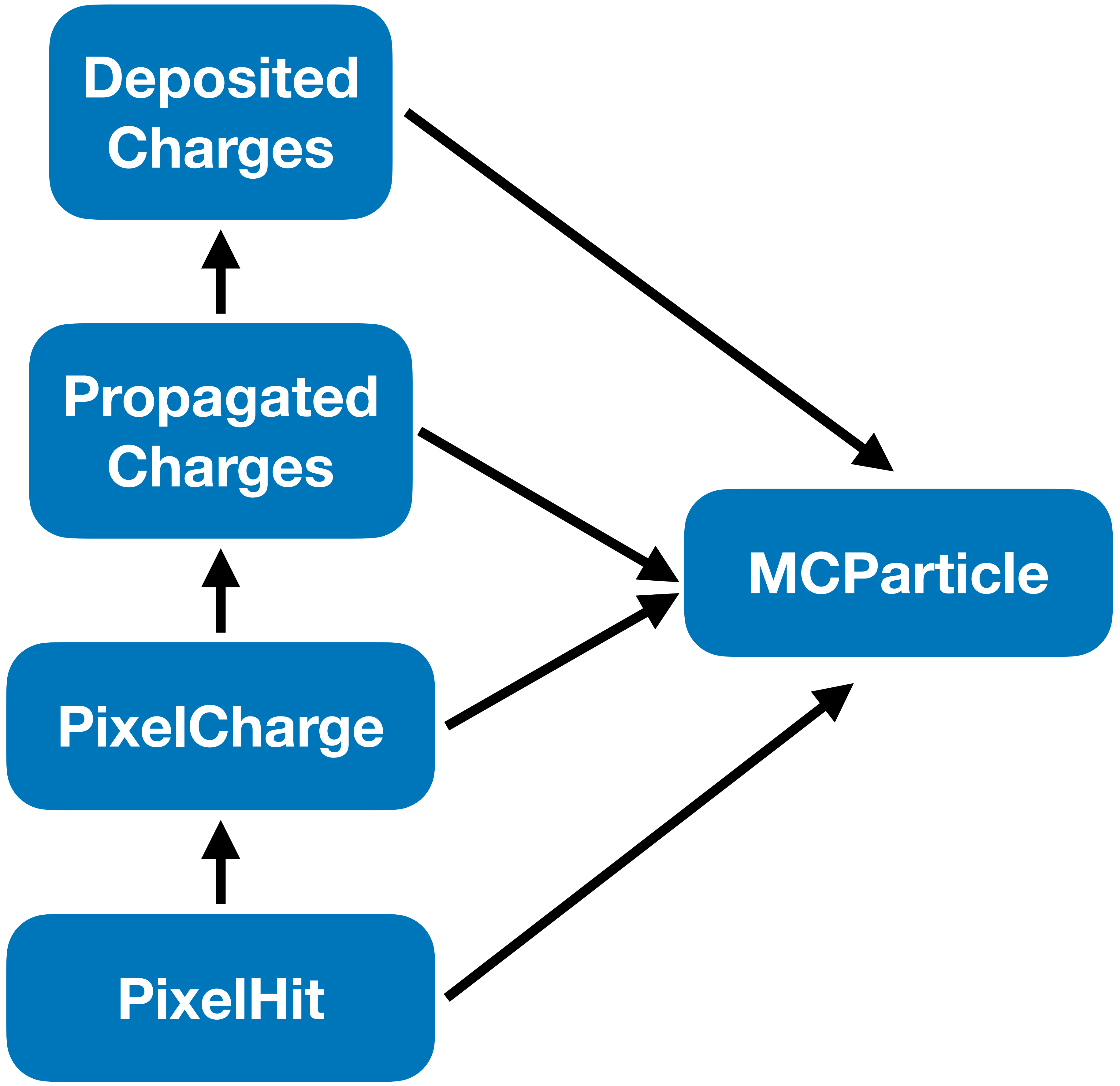
All objects contain information about where they come from

- 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

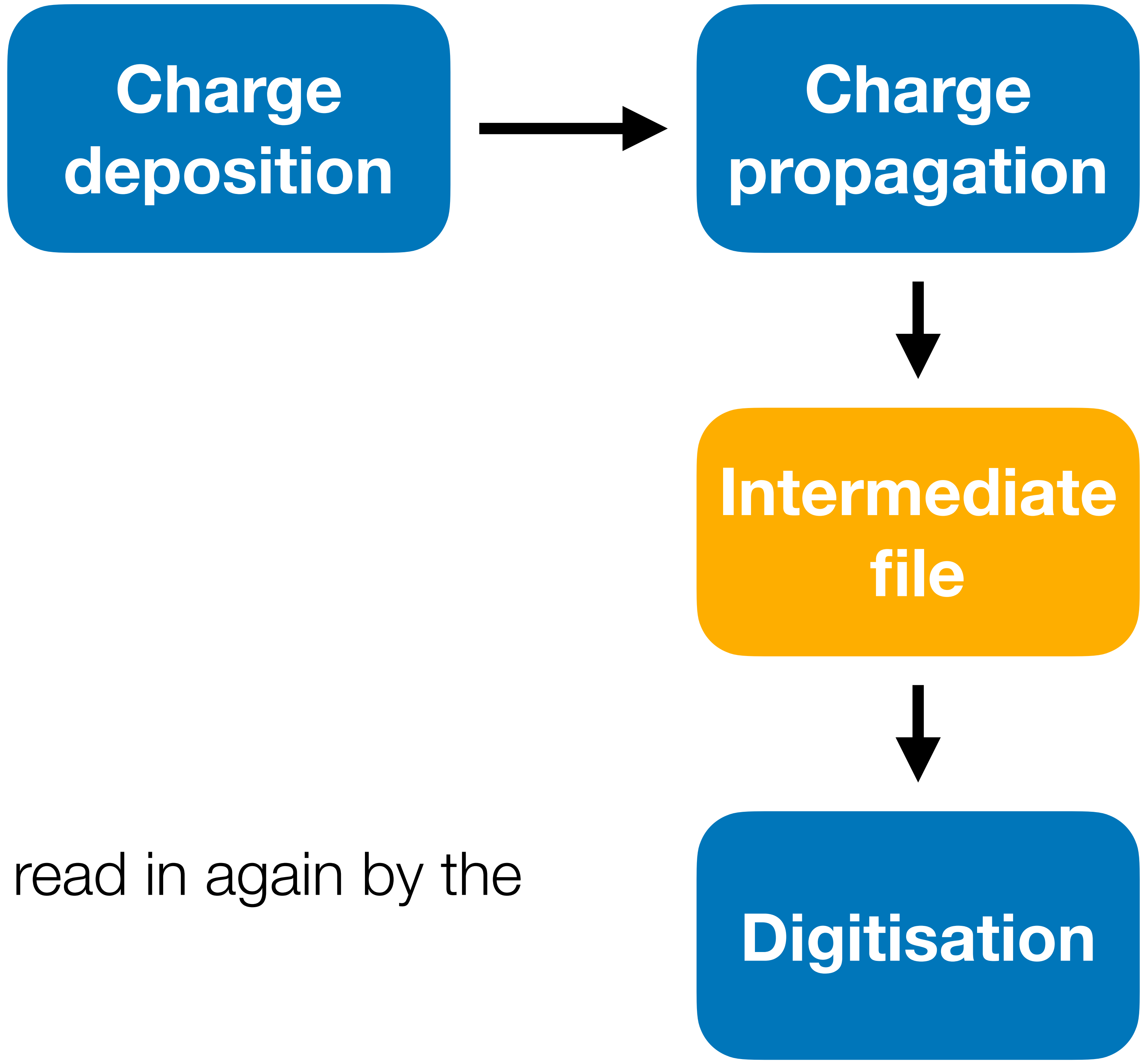
```
/**
 * @brief Typedef for message carrying propagated charges
 */
using PropagatedChargeMessage = Message<PropagatedCharge>;
```



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

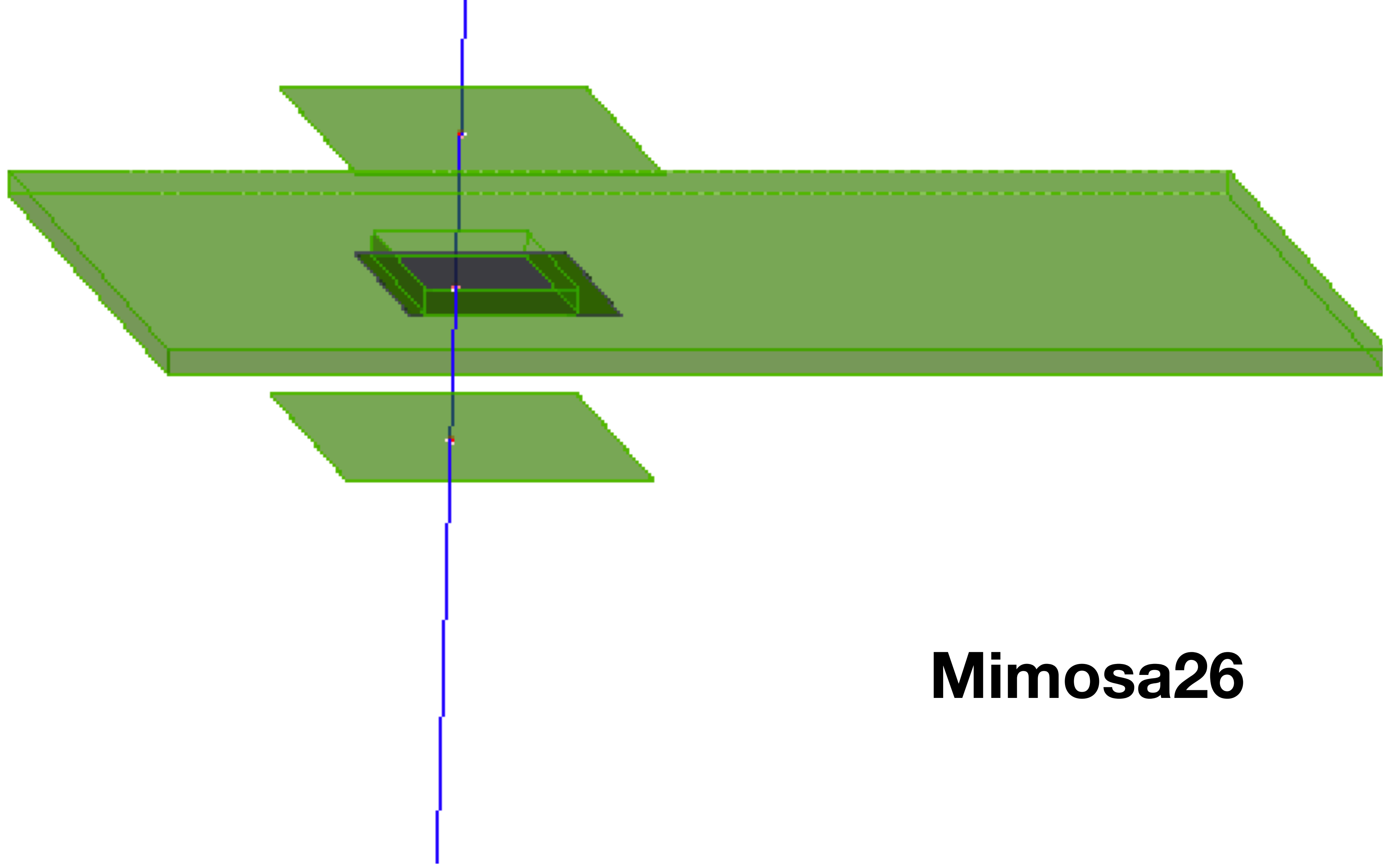
- Allows intermediate file-writing to avoid repeating CPU-intensive parts of the simulation
- eg. Write out propagated charge objects so that tuning of the digitisation parameters does not require re-running Geant4 and propagation code

A few other features - geometry

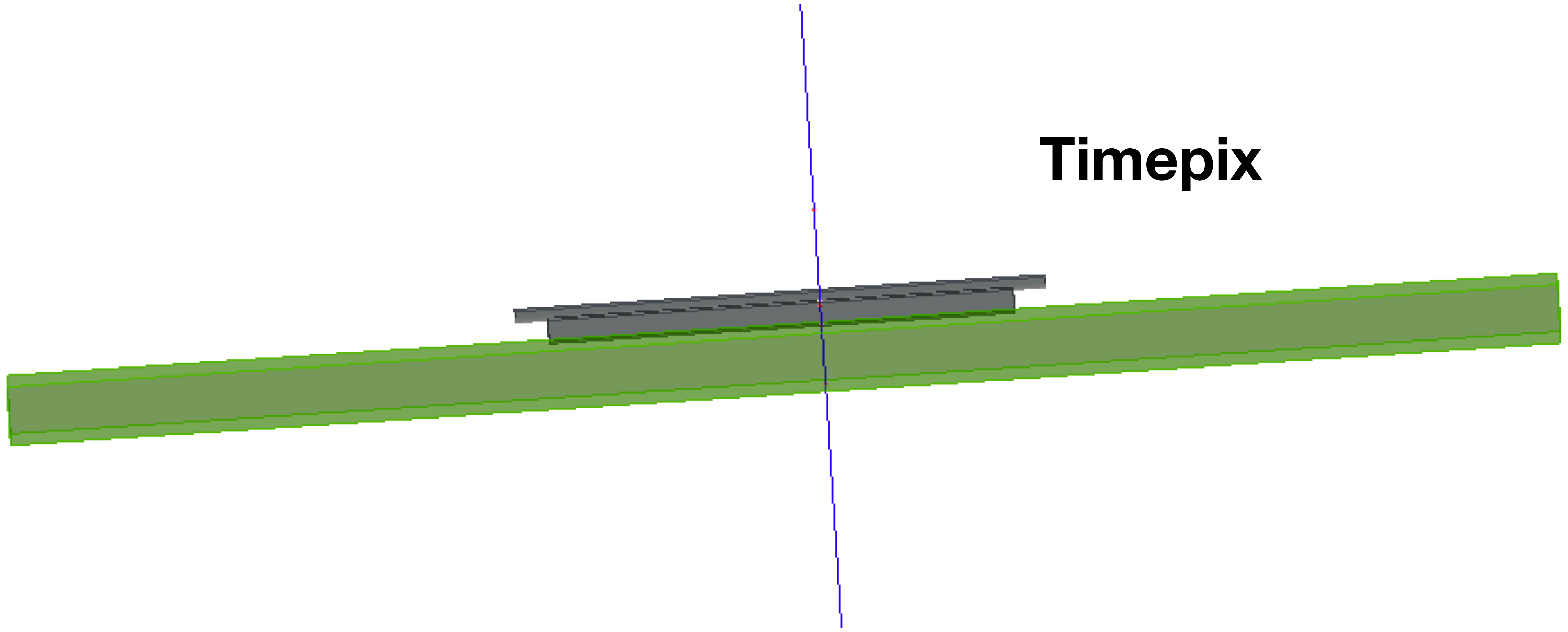
Currently-implemented geometries are for hybrid and monolithic detectors

- 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.



Mimosa26



Timepix

A few highlights of what is currently there

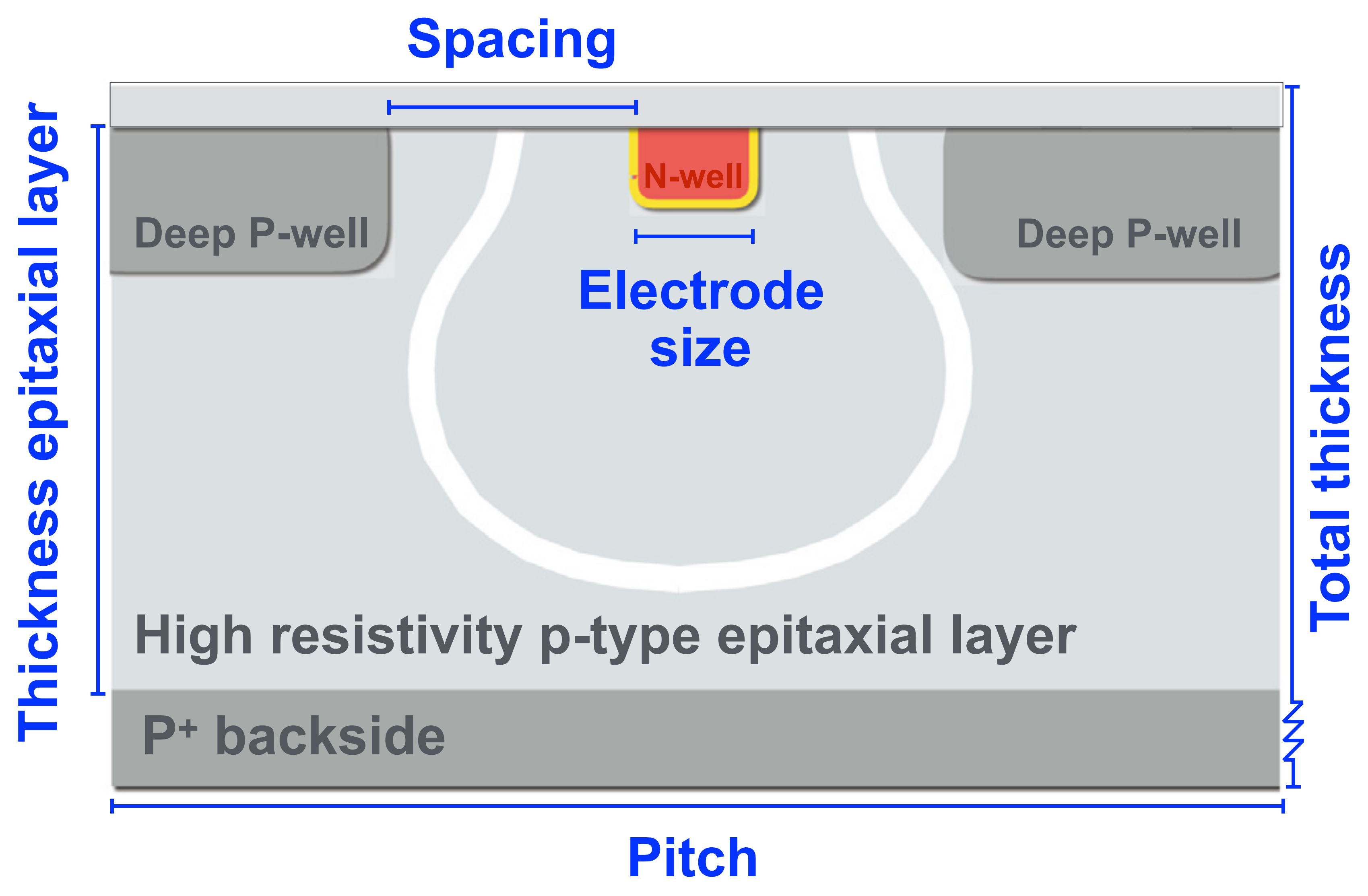
daniel.hynds@physics.ox.ac.uk

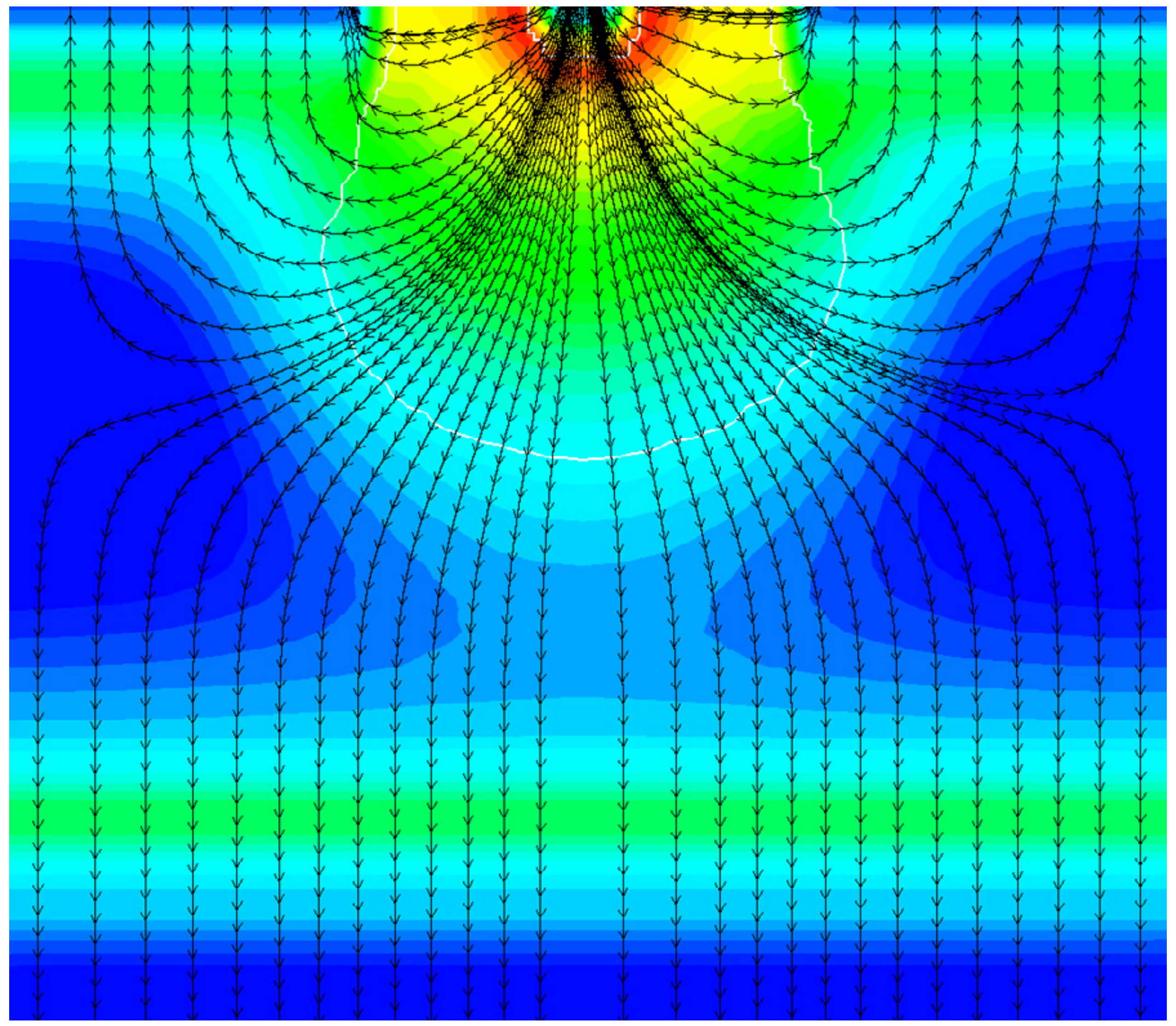
Electric and weighting field import from TCAD - 1

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

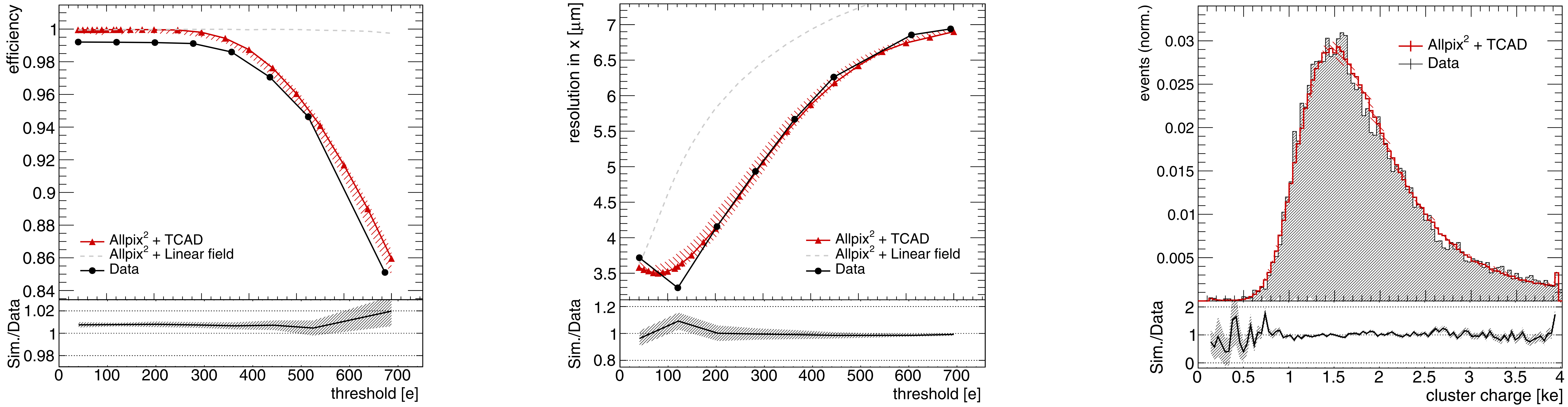




Currently implemented features - weighting field import from TCAD

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

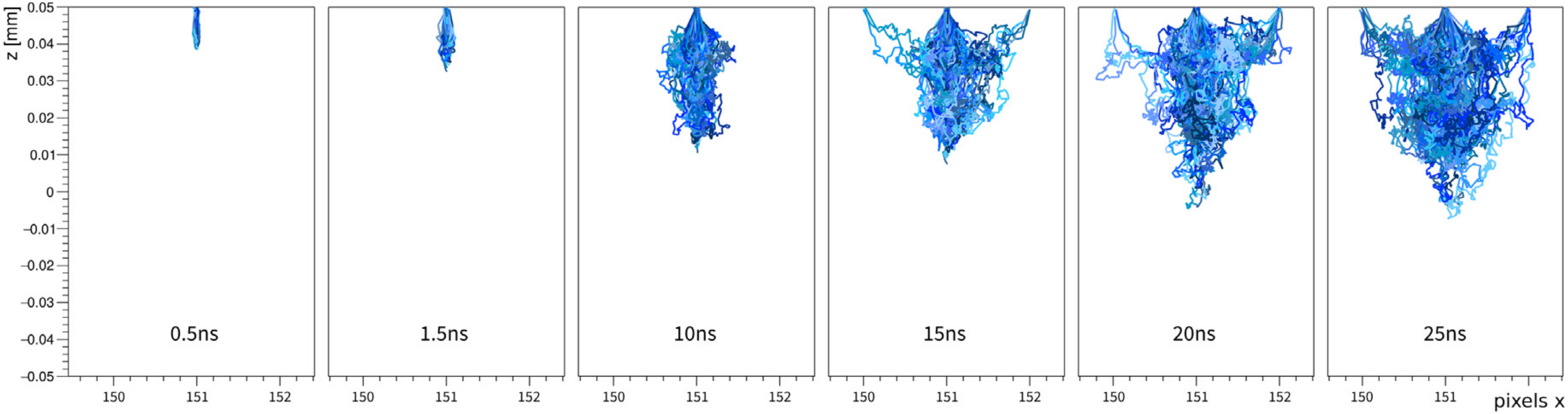
- Complicated field shape around the collection implants
- Undepleted bulk gives rise to carrier recombination



Currently implemented features - weighting field import from TCAD

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

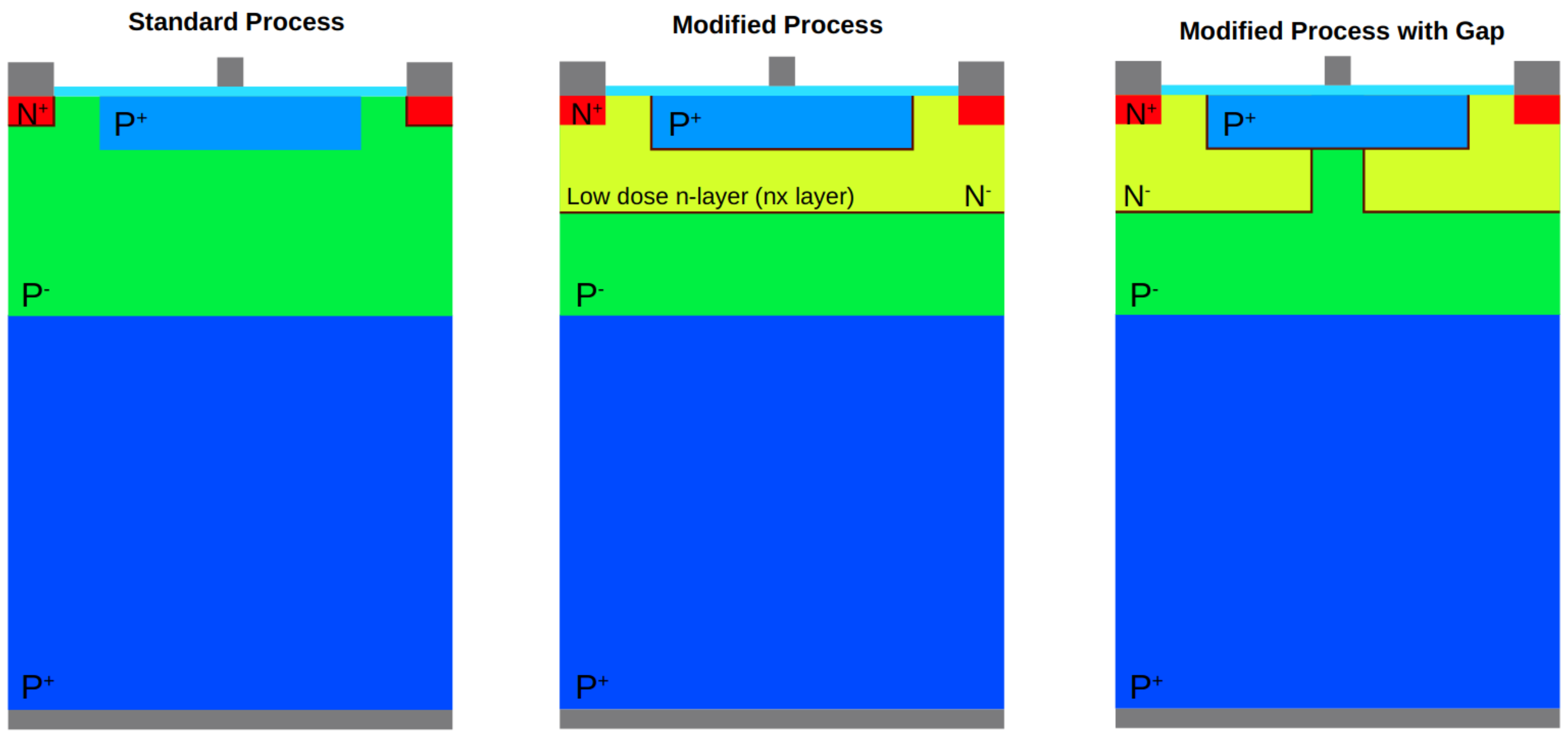
- Complicated field shape around the collection implants
- Undepleted bulk gives rise to carrier recombination

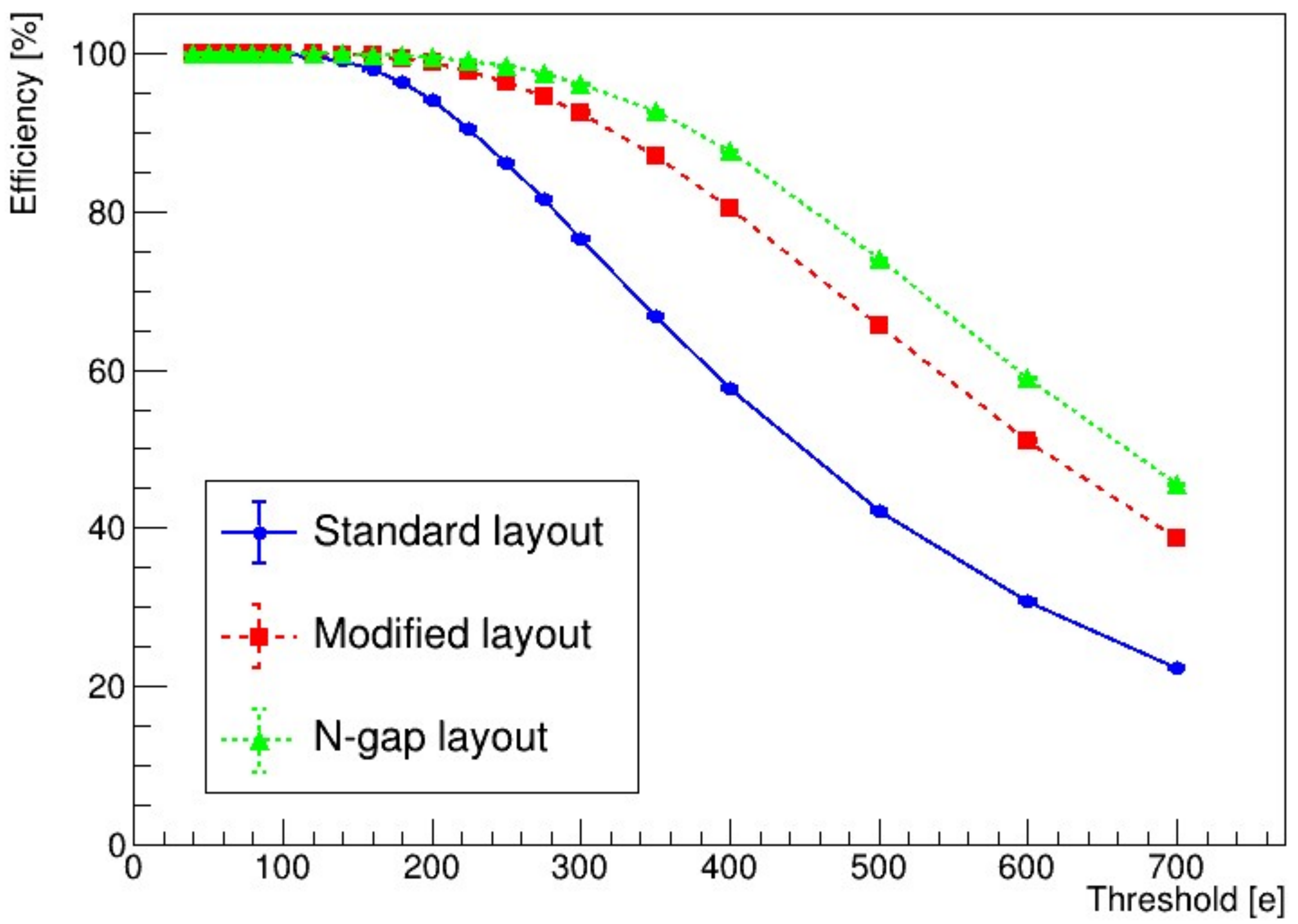


daniel.hynds@physics.ox.ac.uk

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





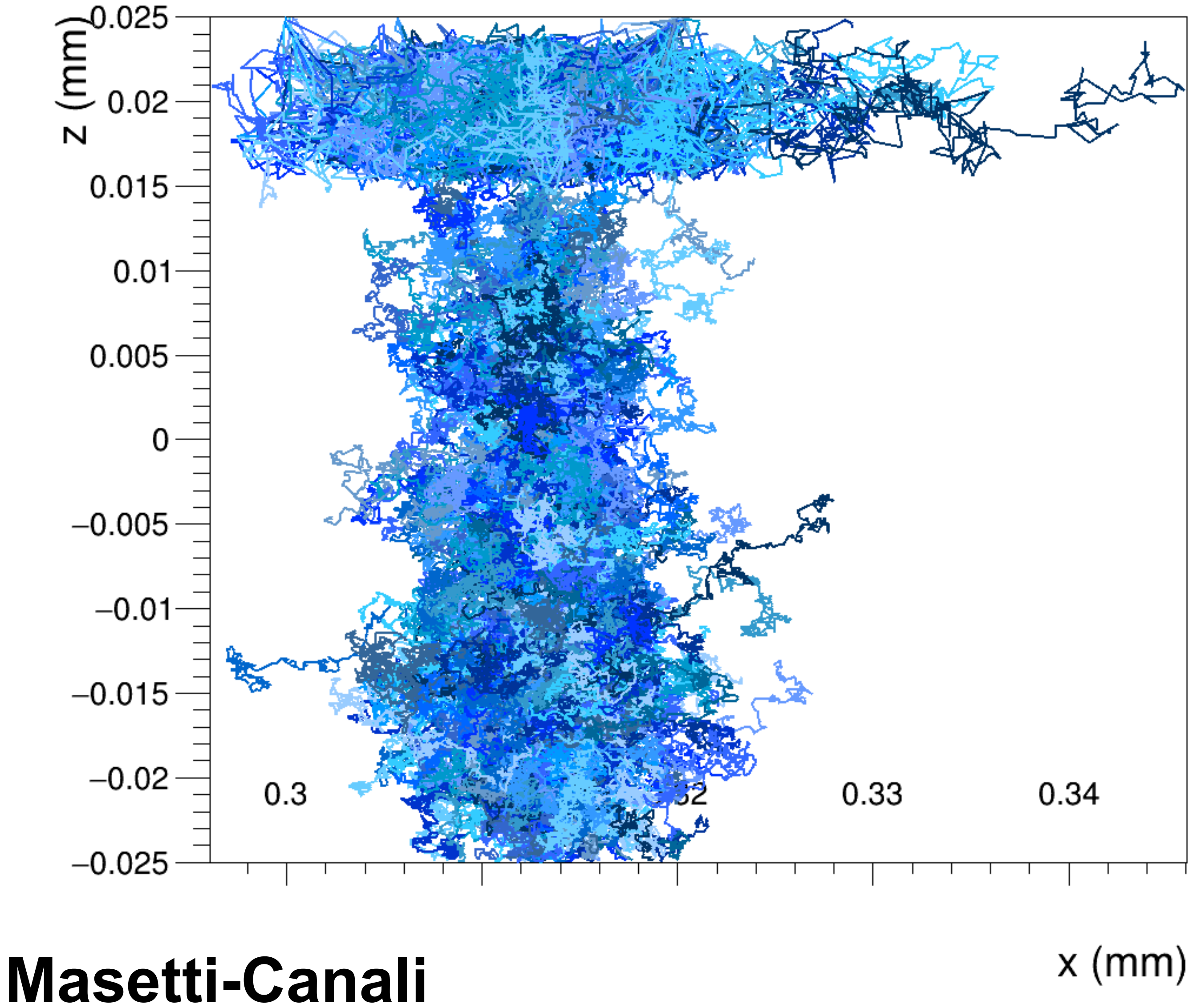
Propagation models

Masetti-Canali

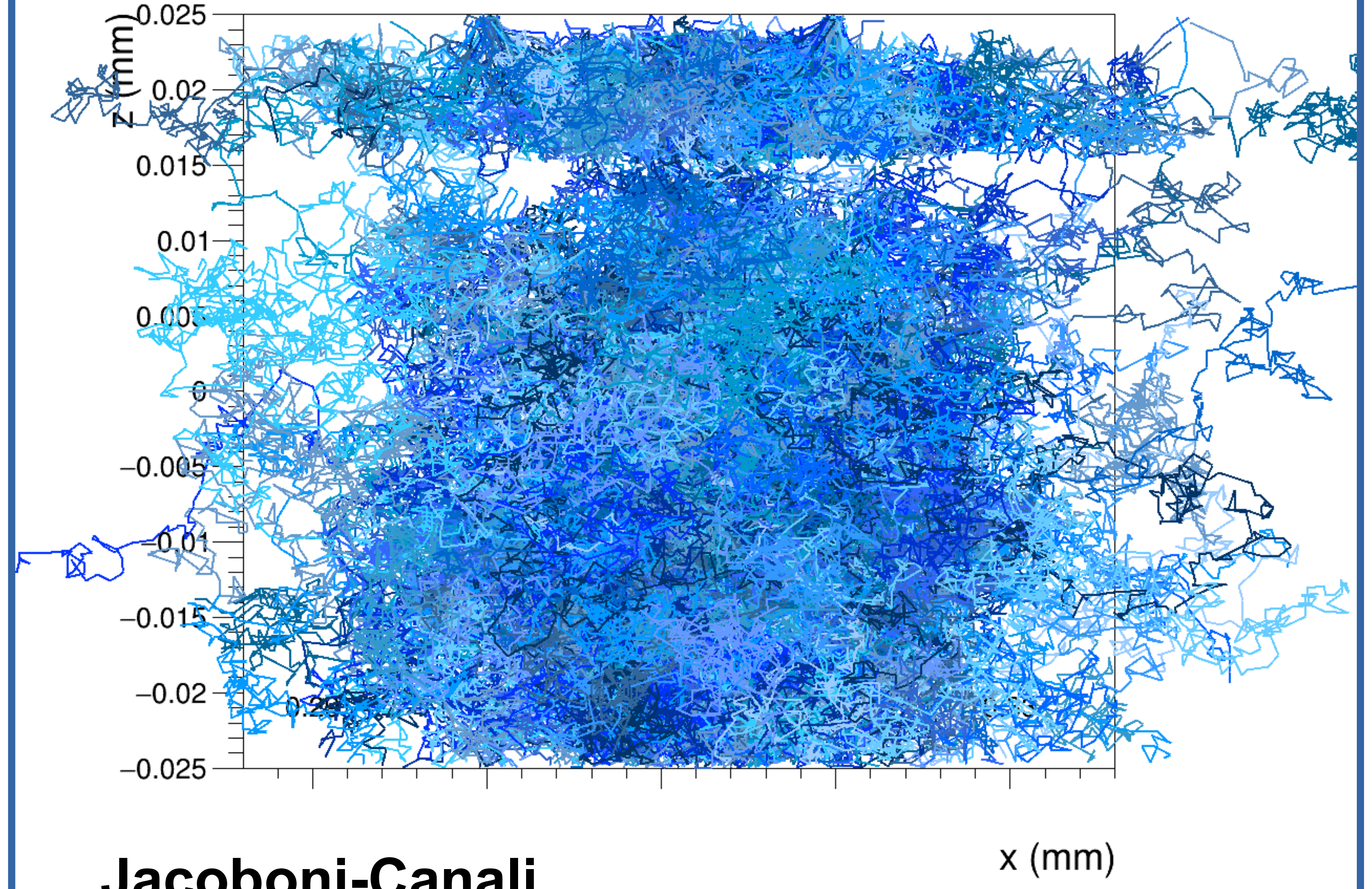
$$\mu(E,N) = \frac{\mu_m(N)}{\left(1 + (\mu_m(N)\cdot E/v_m)^\beta\right)^{1/\beta}}$$

Jacoboni-Canali

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



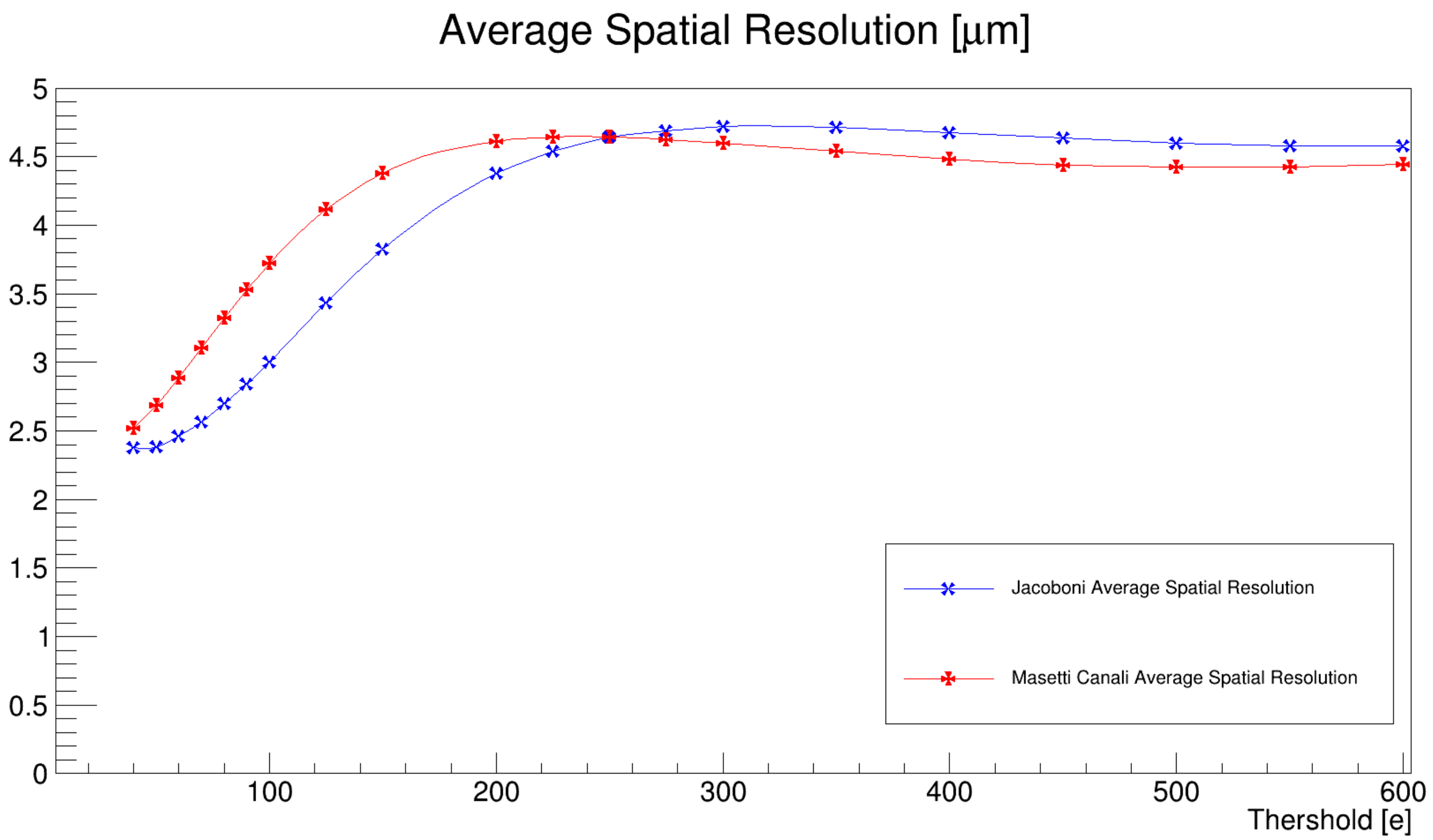
Masetti-Canali

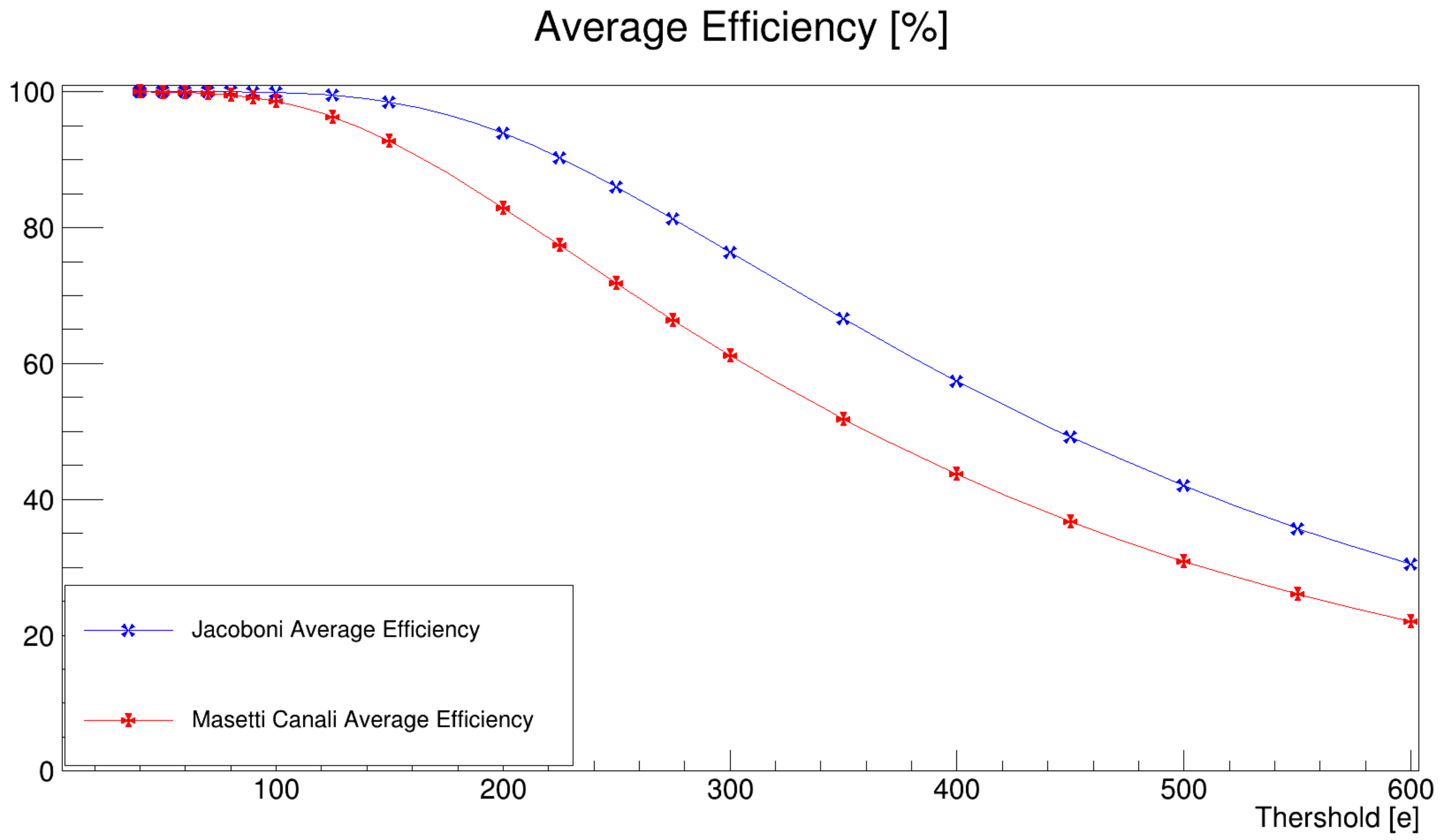


Jacoboni-Canali

daniel.hynds@physics.ox.ac.uk

```
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
```



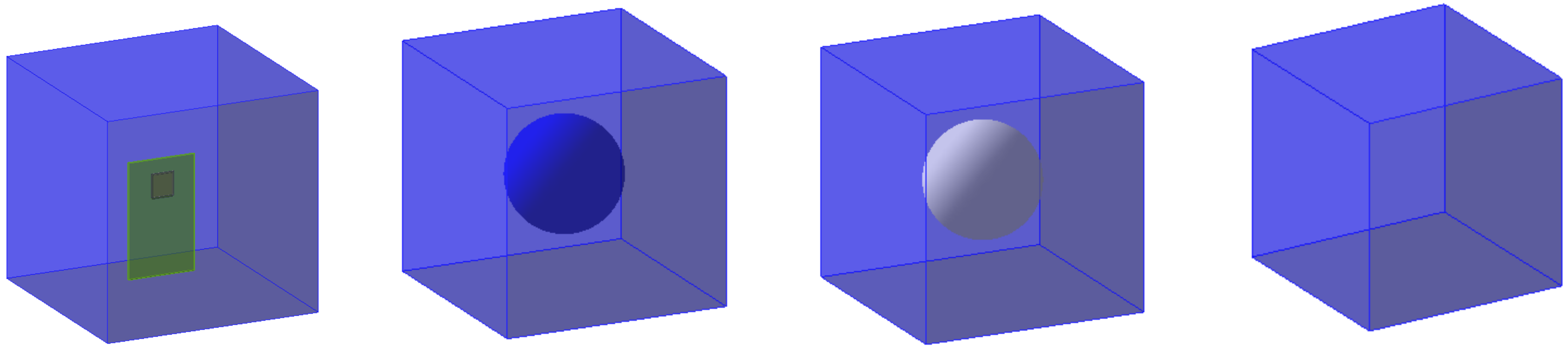


Passive materials

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

daniel.hynds@physics.ox.ac.uk

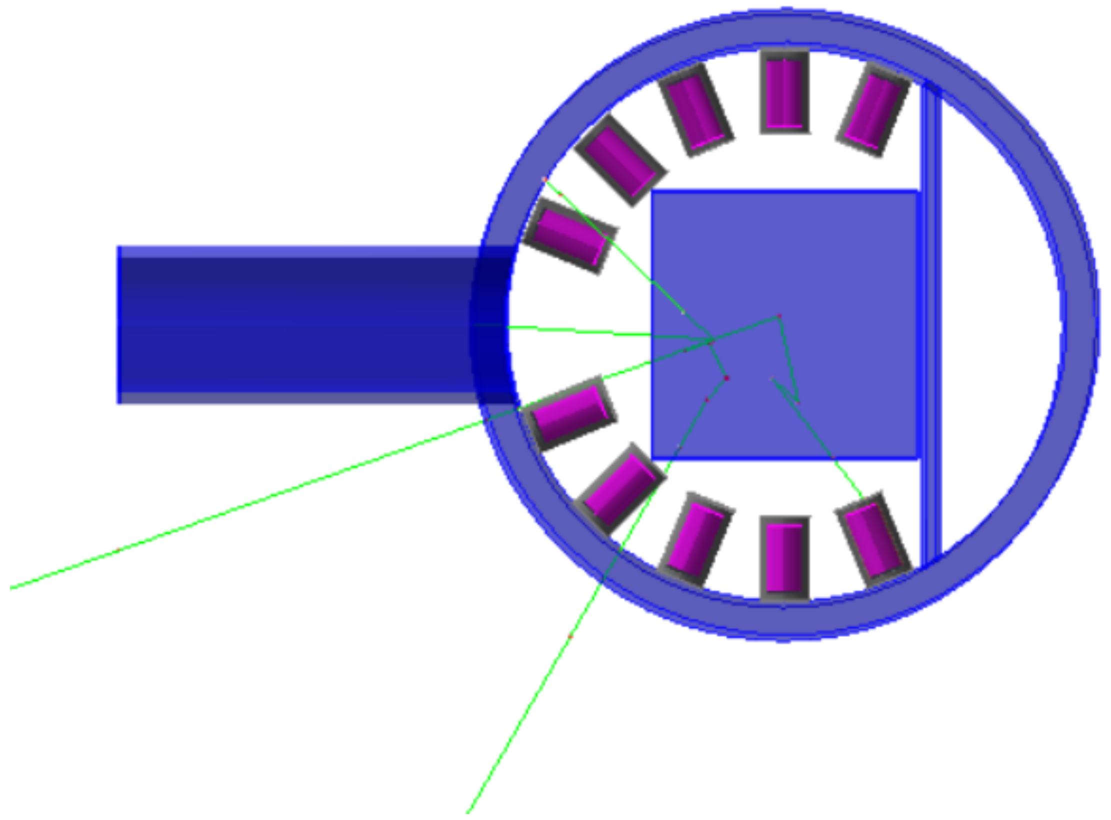
Passive materials

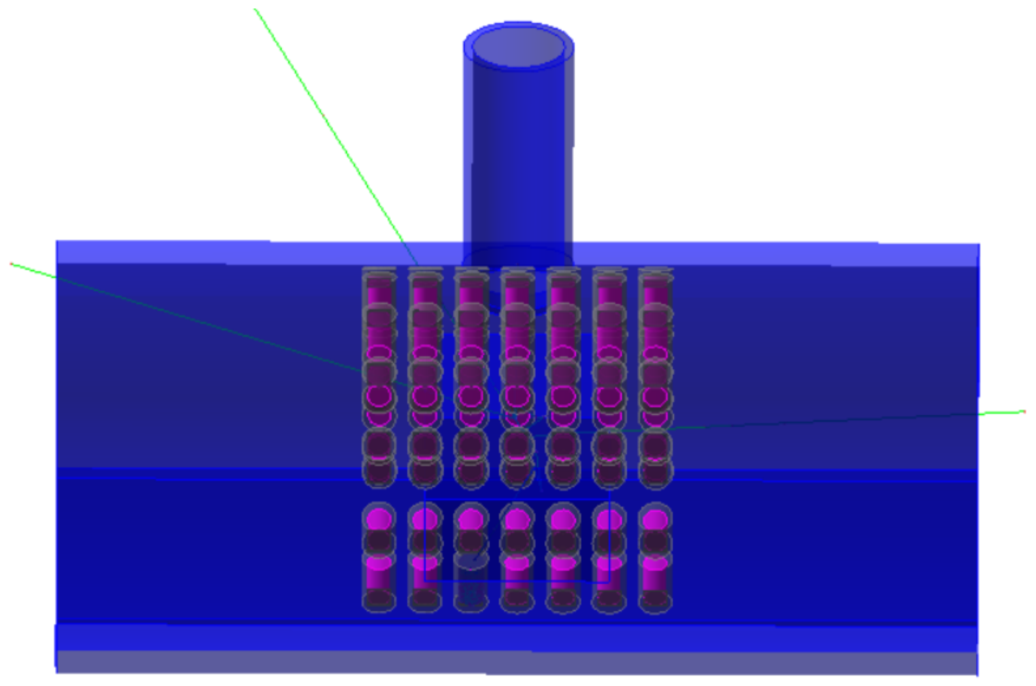
K.Van Den Brandt, Master Thesis, 2017

Allpix squared was not originally designed to work with passive materials

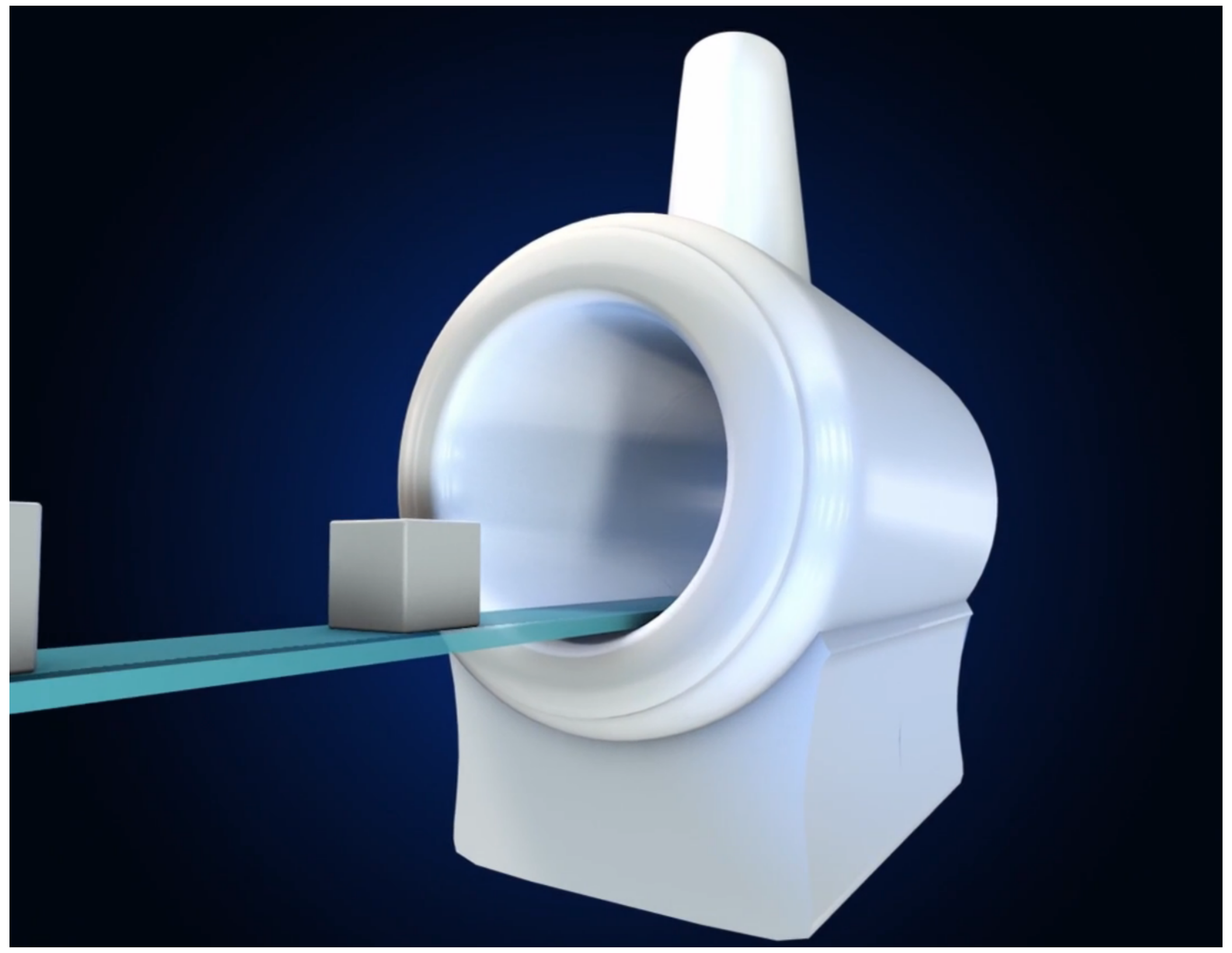
- everything was linked to detector models and geometry

- 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



daniel.hynds@physics.ox.ac.uk

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…

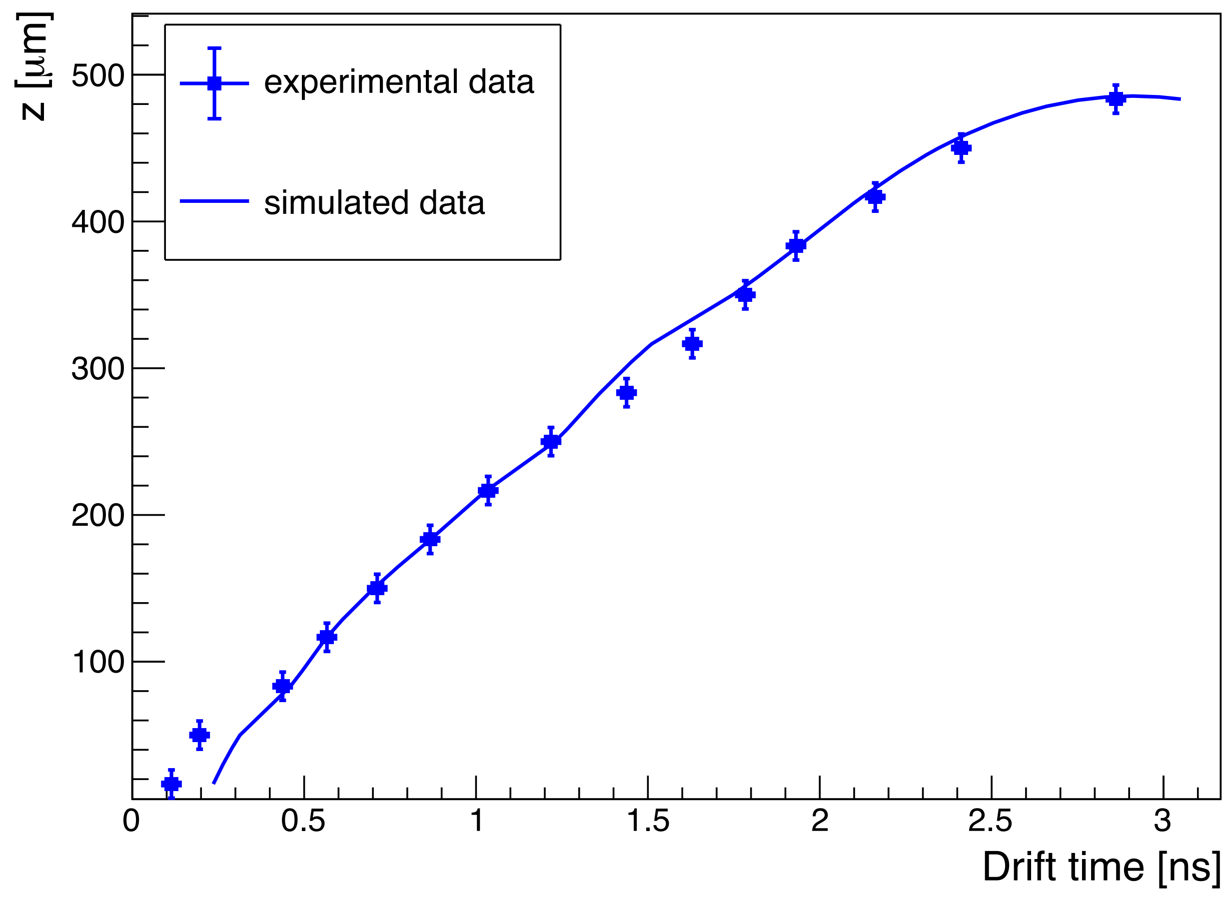
Table 6.1: List of default sensor material properties implemented in $Allpix^2$

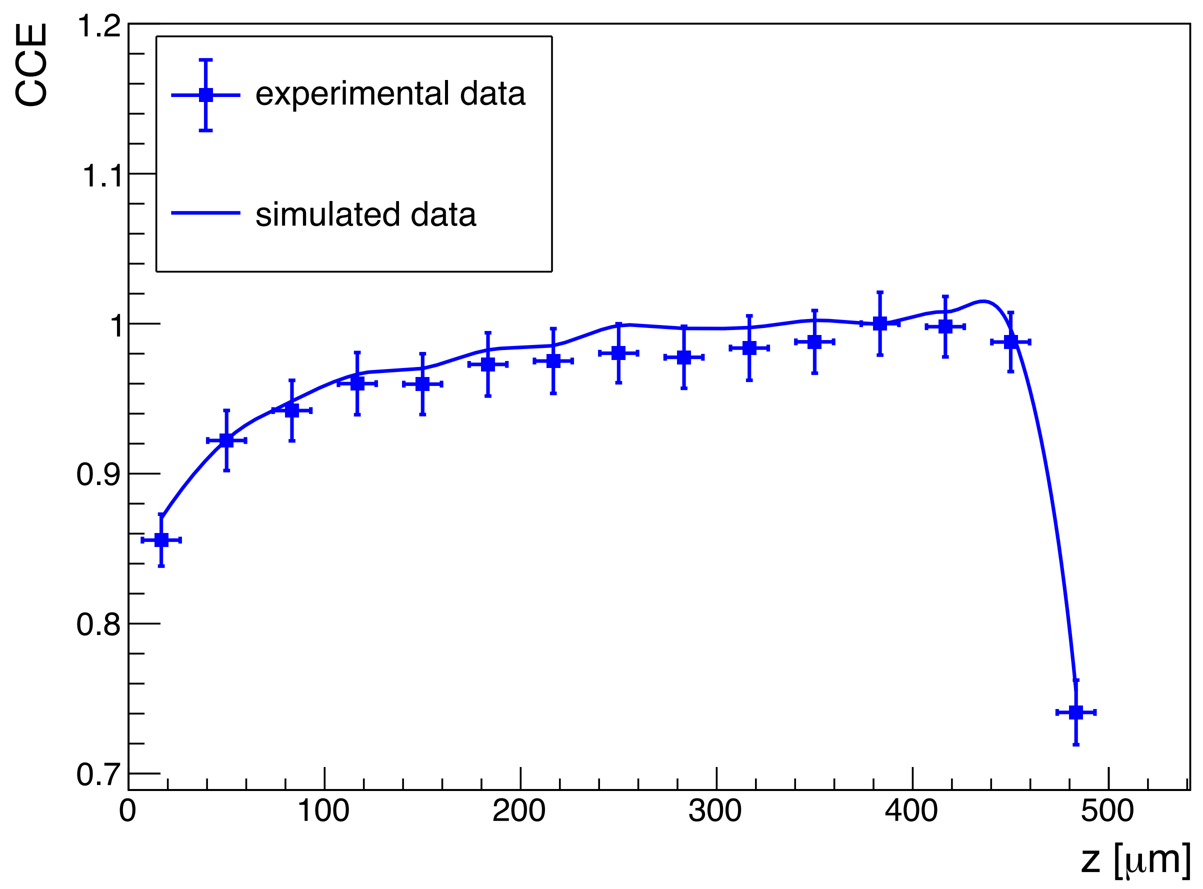
Material	**Charge Creation Energy [eV]**	**Fano factor**	**Sources**
Silicon	3.64	0.115	[25], [26]
Gallium Arsenide	4.2	0.14	[27]
Cadmium Telluride	4.43	0.24	[28], [29]
Cadmium Zinc Telluride $Cd_{0.8}Zn_{0.2}Te$	4.6	0.14	[30], [31]
Diamond	13.1	0.382	[32], [32]
Silicon Carbide (4H-SiC)	7.6	0.1	[33],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

daniel.hynds@physics.ox.ac.uk

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
```





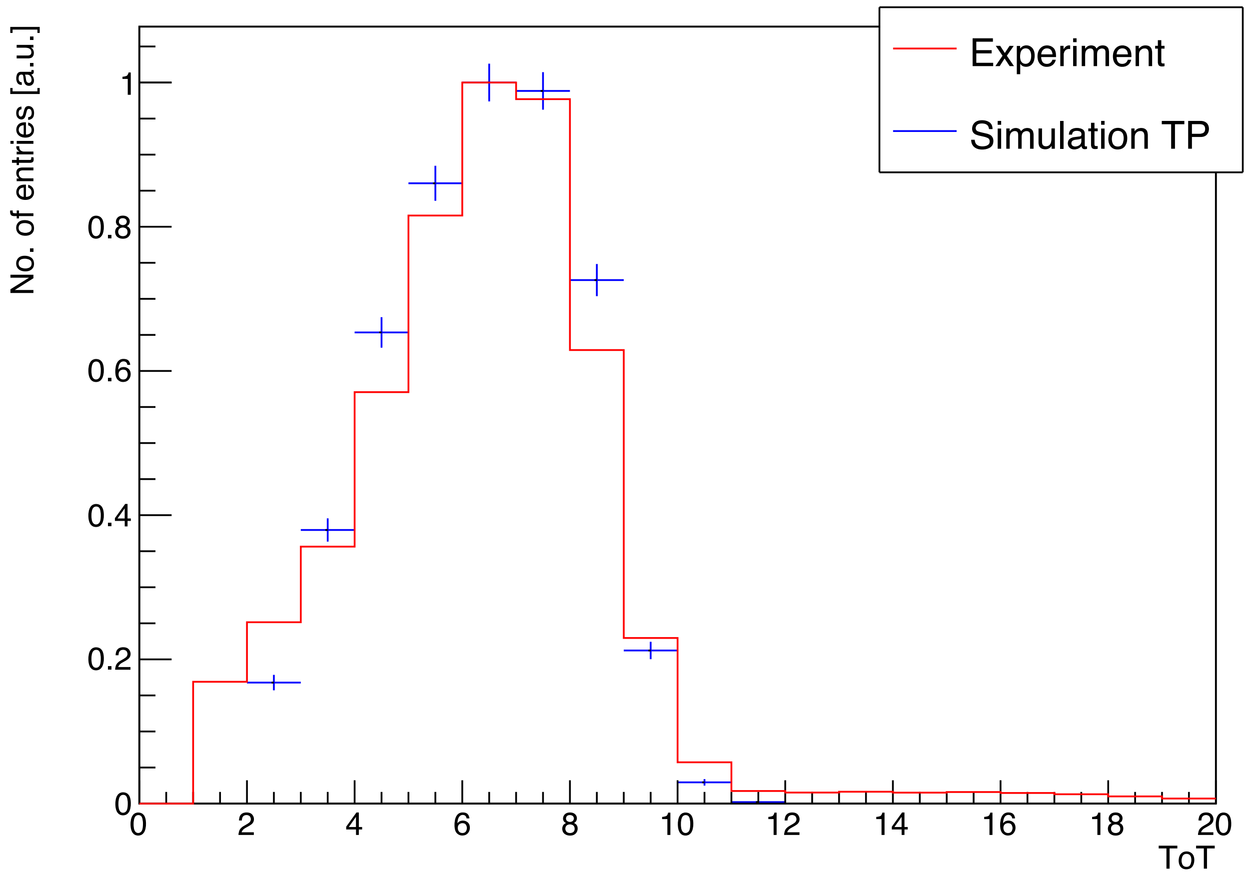


Figure: 8 keV photons spectra.

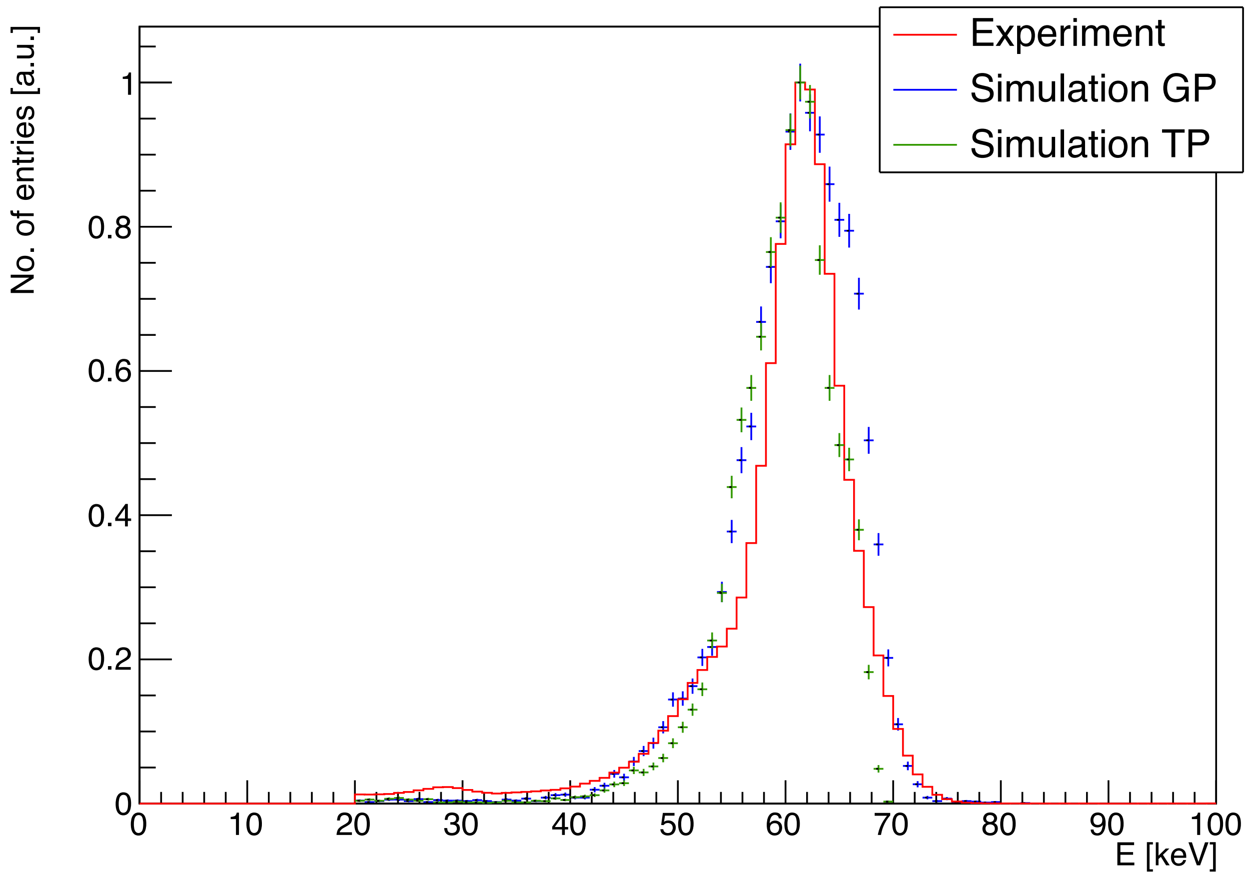
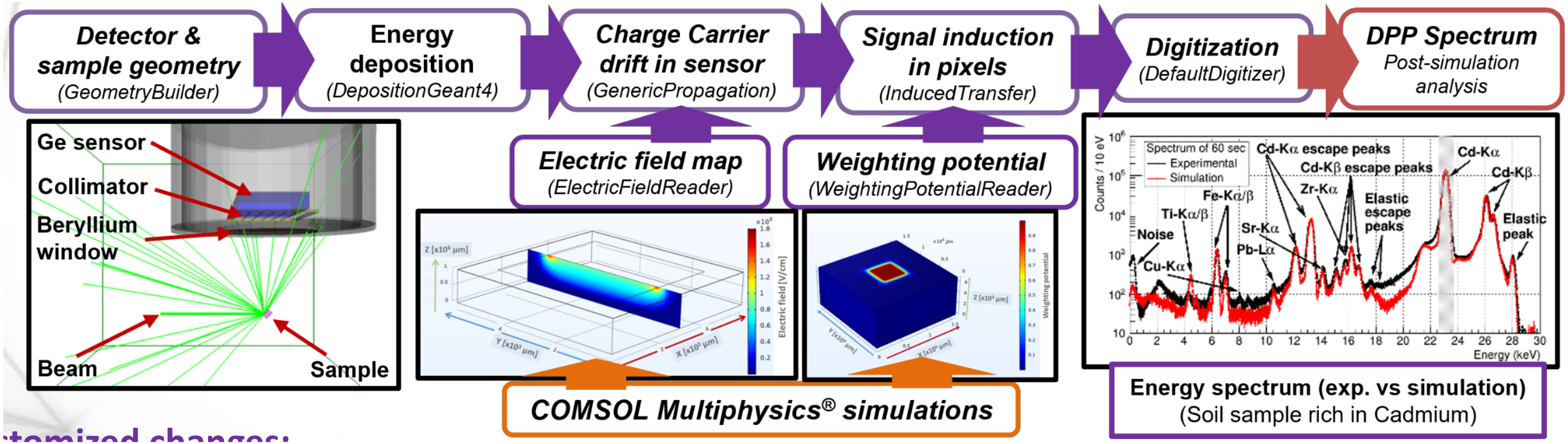


Figure: of ^{241}Am source spectra.

daniel.hynds@physics.ox.ac.uk

Non-silicon materials: Ge



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

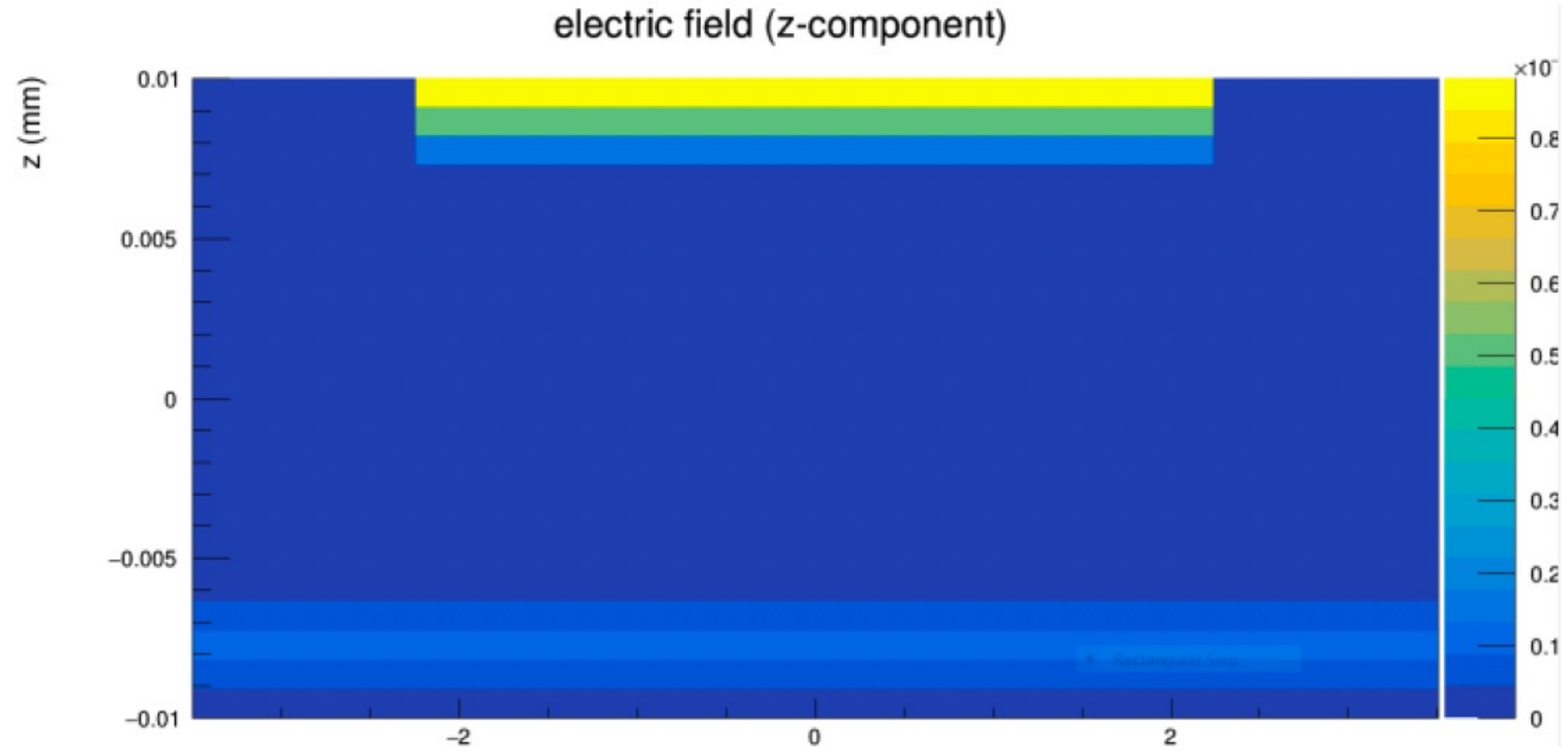
$$\mu_{n,p} = \mu_{n,p}^{min} + \frac{\mu_{n,p}^{max} - \mu_{n,p}^{min}}{1 + \left(\frac{N_i}{N_{n,p}^{crit}}\right)^{\delta_{n,p}}}$$

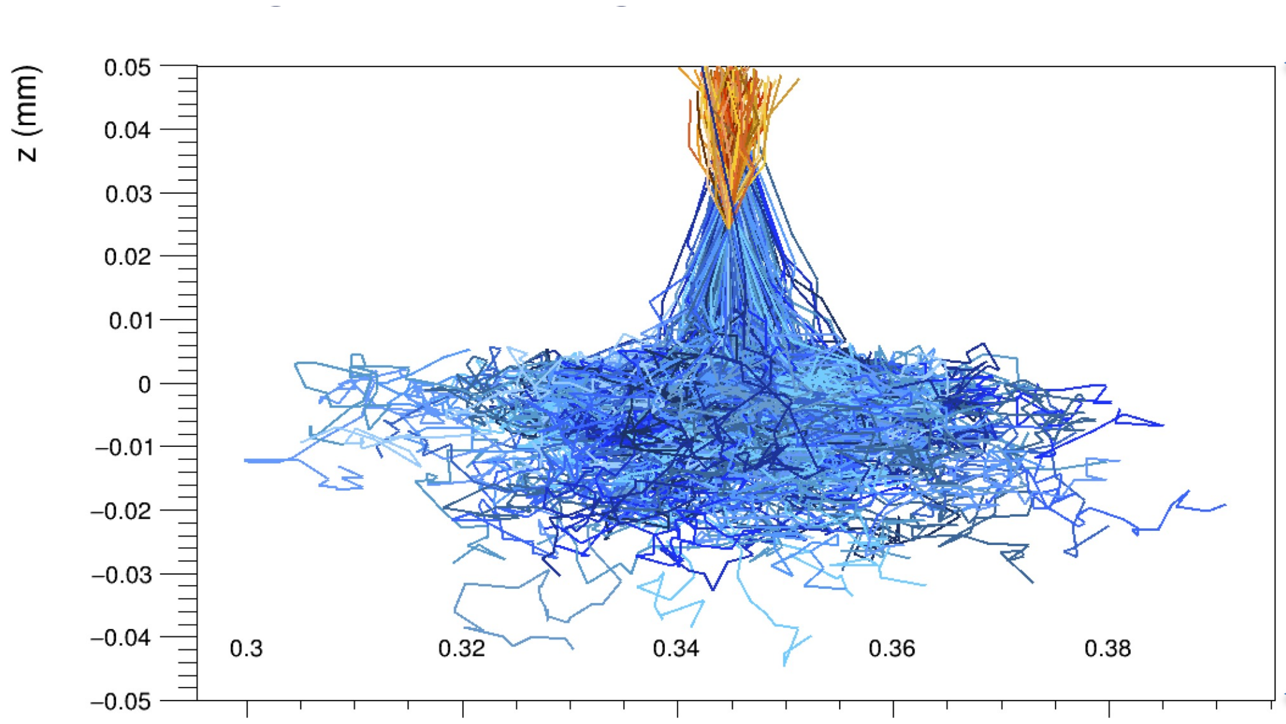
considering the temperature dependence of the parameters, the equation becomes

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

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

Carrier type	$\mu^{min,0}$ $cm^2V^{-1}s^{-1}$	$\mu^{max,0}$ $cm^2V^{-1}s^{-1}$	$N^{crit,0}$ 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]

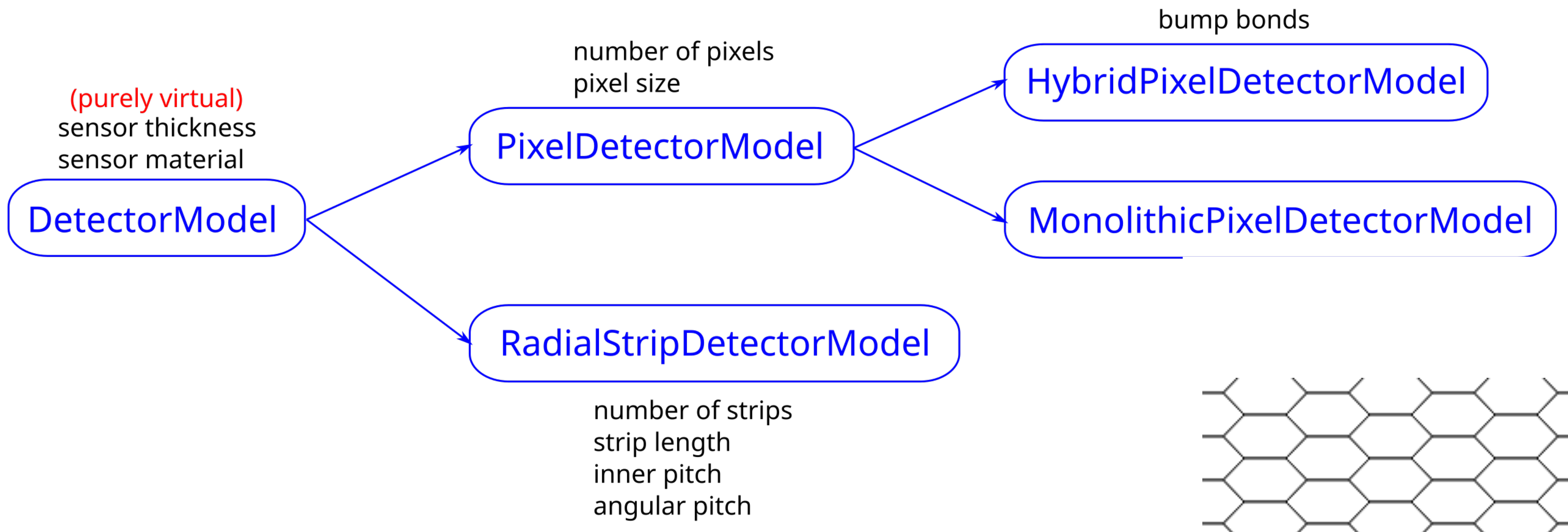


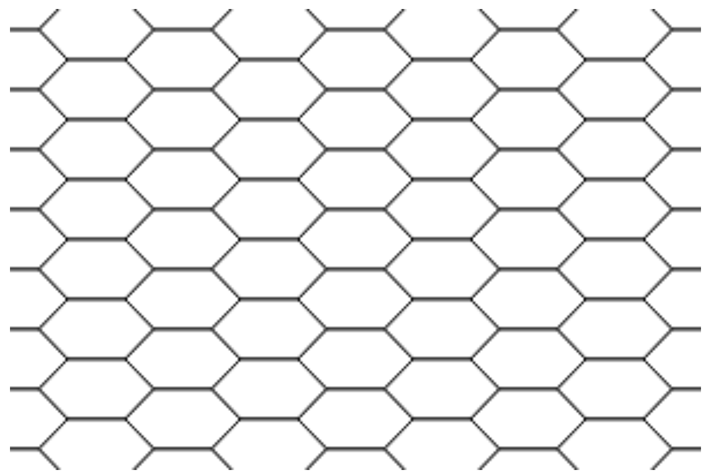


daniel.hynds@physics.ox.ac.uk

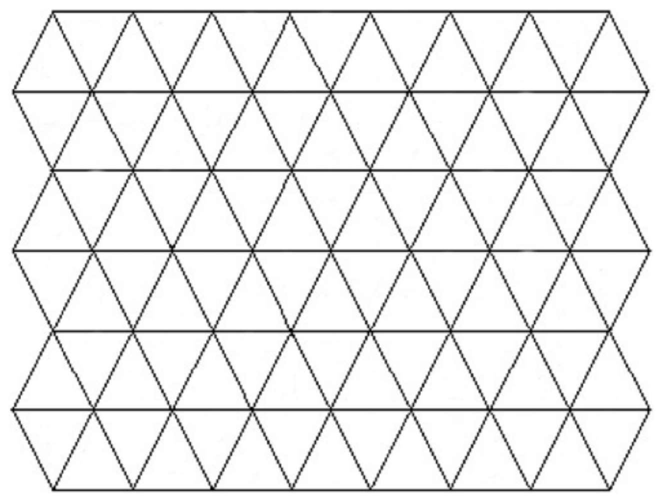
Requirements from different groups have again led to changes in the core to make it flexible enough to cope with different sensor geometries

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

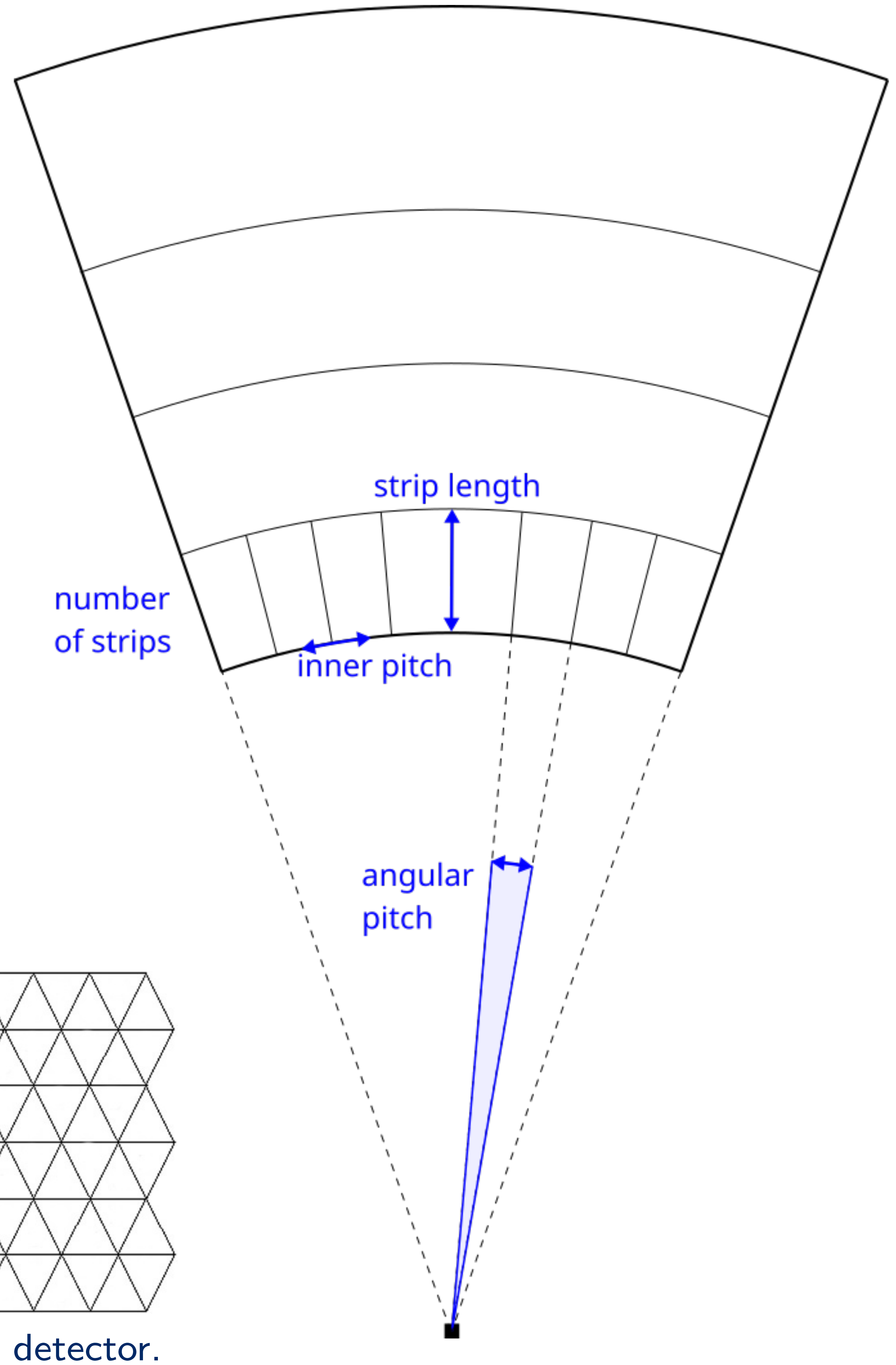




Hexagonal pixel detector.

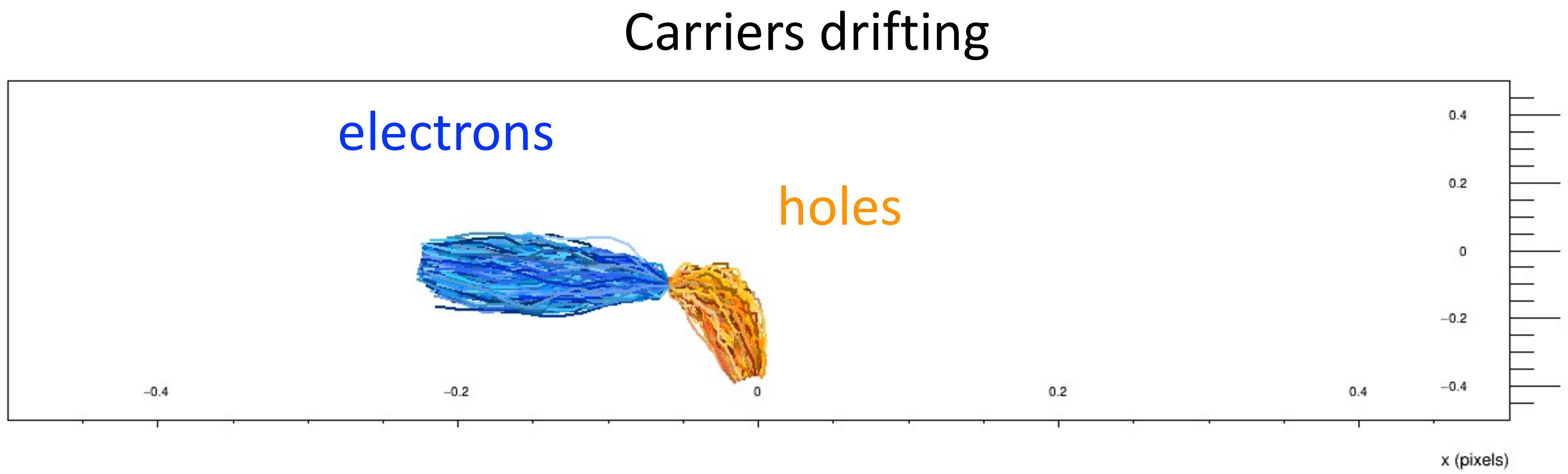


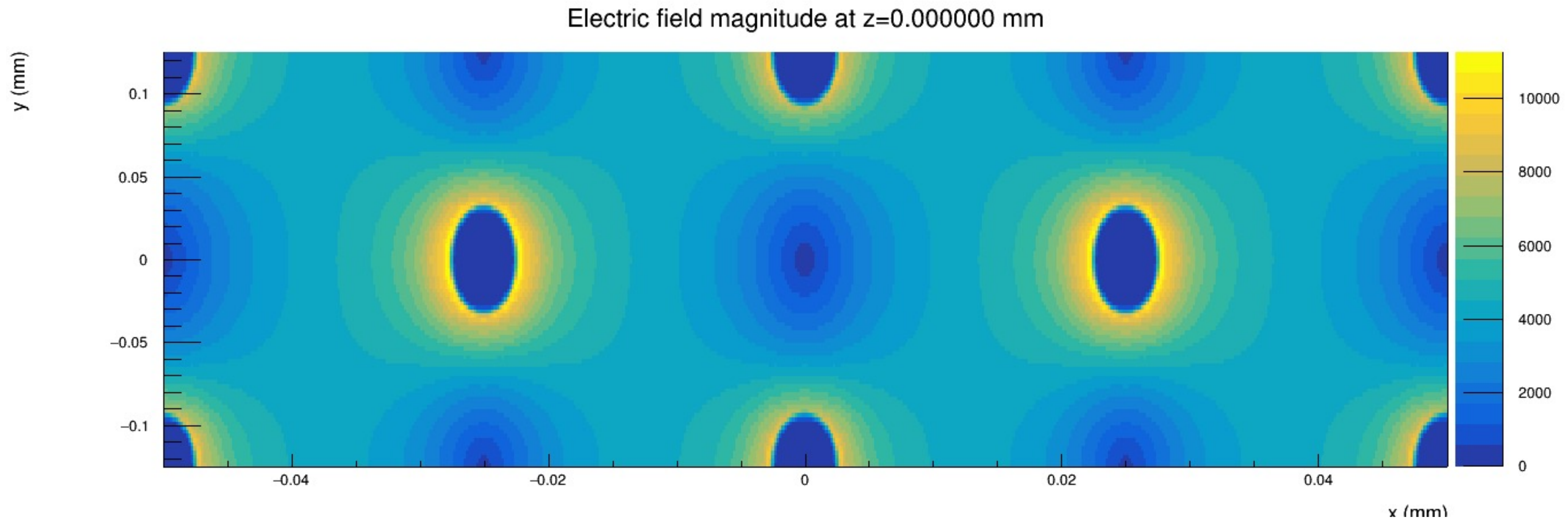
Triangular pixel detector.

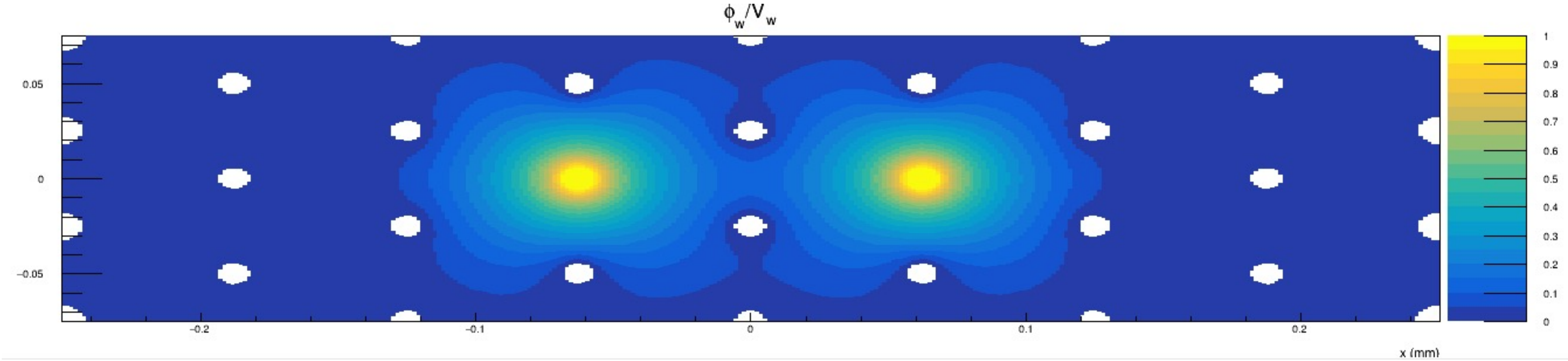


Requirements from different groups have again led to changes in the core to make it flexible enough to cope with different sensor geometries

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



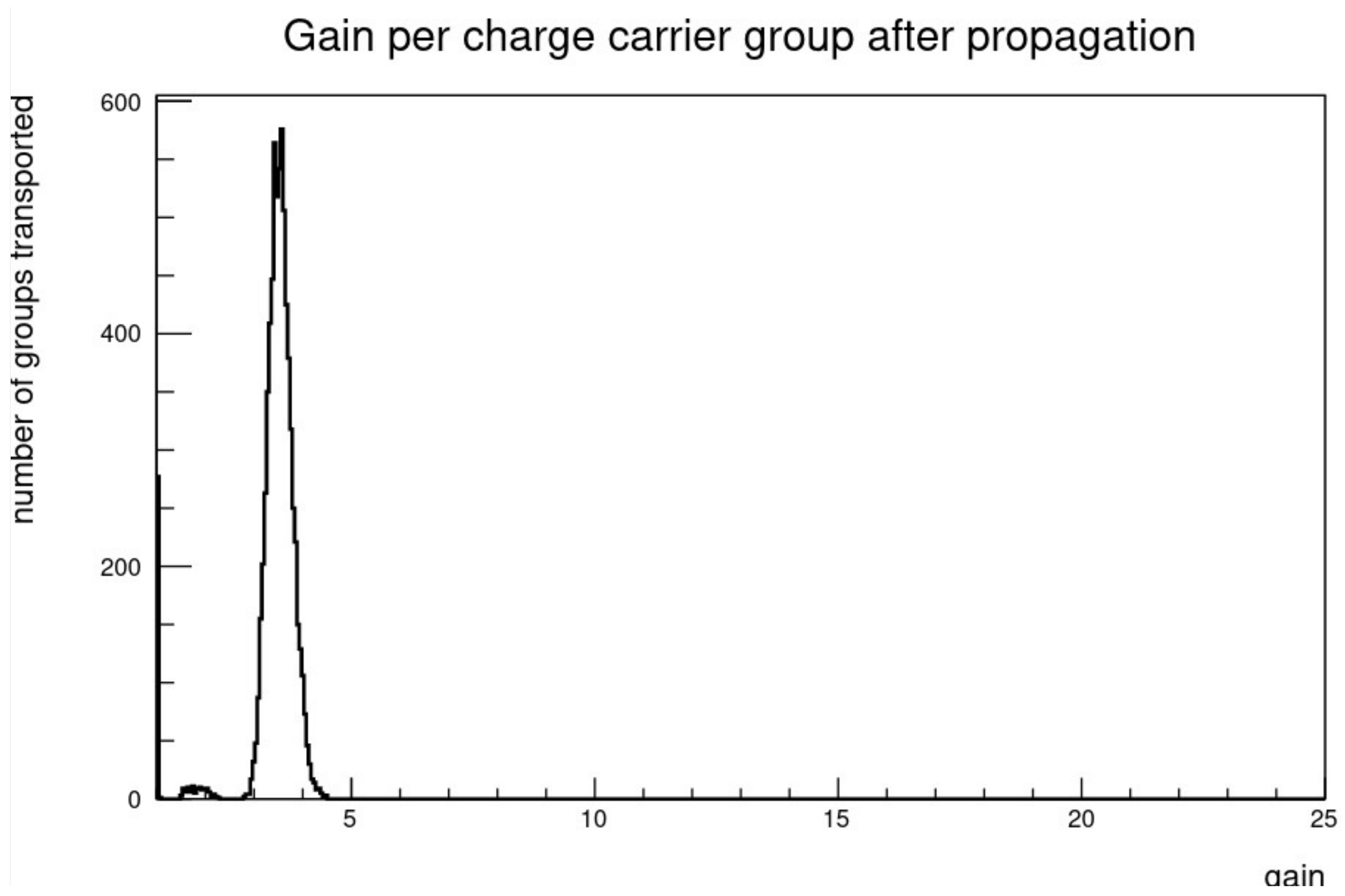




Ongoing work in addressing one of the popular topics in detector development at the moment: charge multiplication

- 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 Allpix^2 completed**, undergoing testing, Comparison with Weightfield2 & TCAD simulations

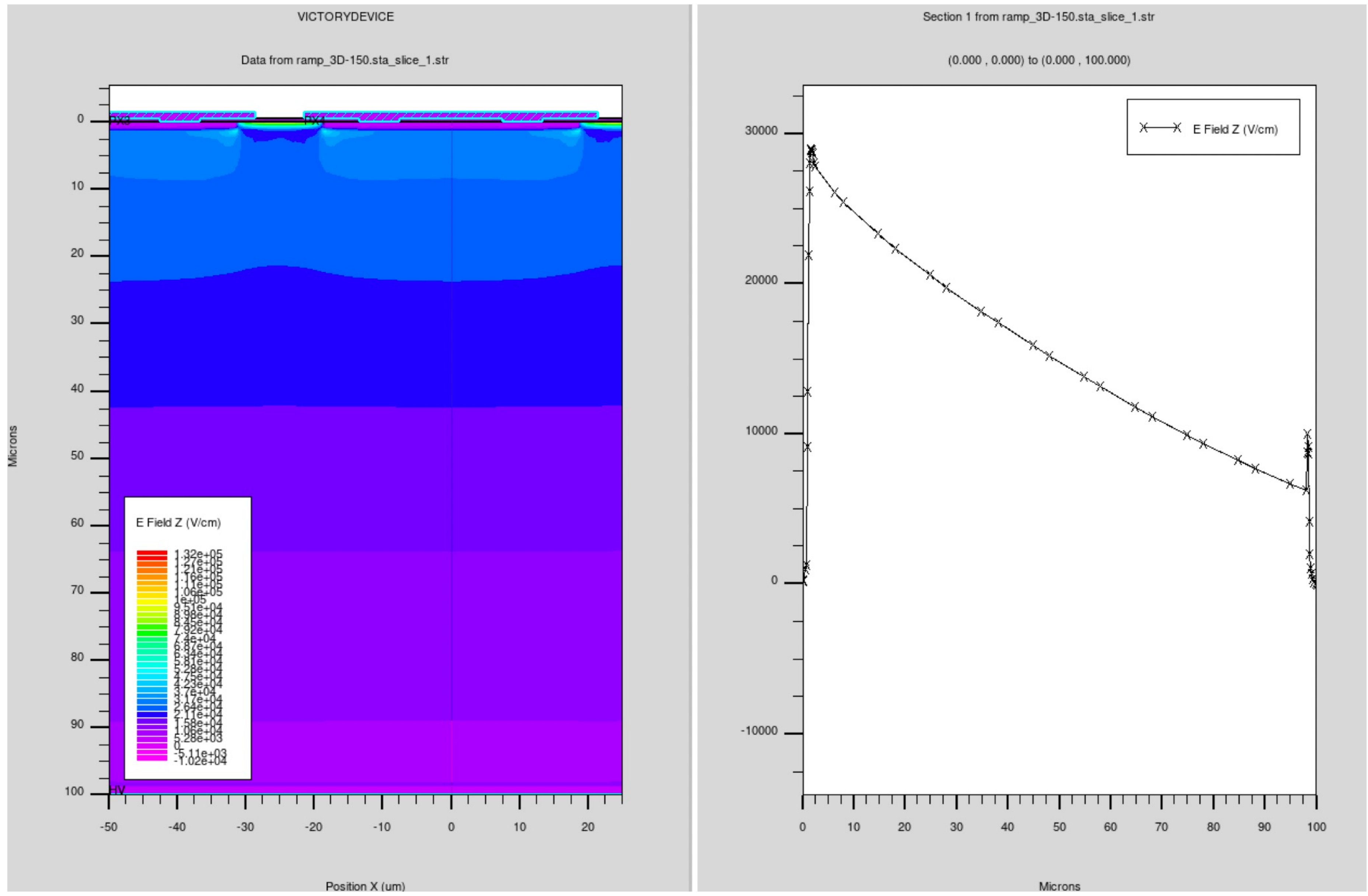
Similarly, implementation of radiation damage is required to study devices for hadron collider experiments

- Clearly contains two components:
  - Modified TCAD electric field profile (proper modelling of space charge effects
  - Charge trapping implementation in allpix^2

2016 JINST **11** P04023

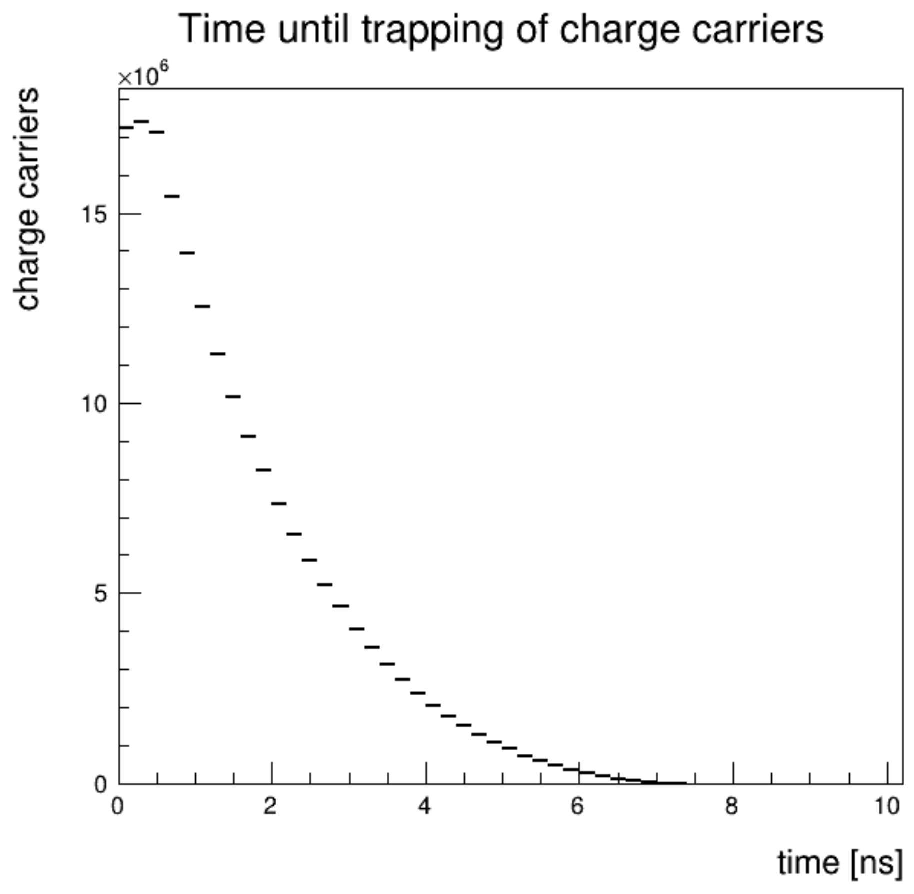
Table 1. The key parameter values used in the Synopsys device simulation. These include: donor and acceptor concentrations, N_D and N_A, and their electron and hole capture cross sections, $\sigma^{e,h}_{D,A}$, for silicon sensors after irradiation with 23 GeV protons (top rows) [11], and for sensors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

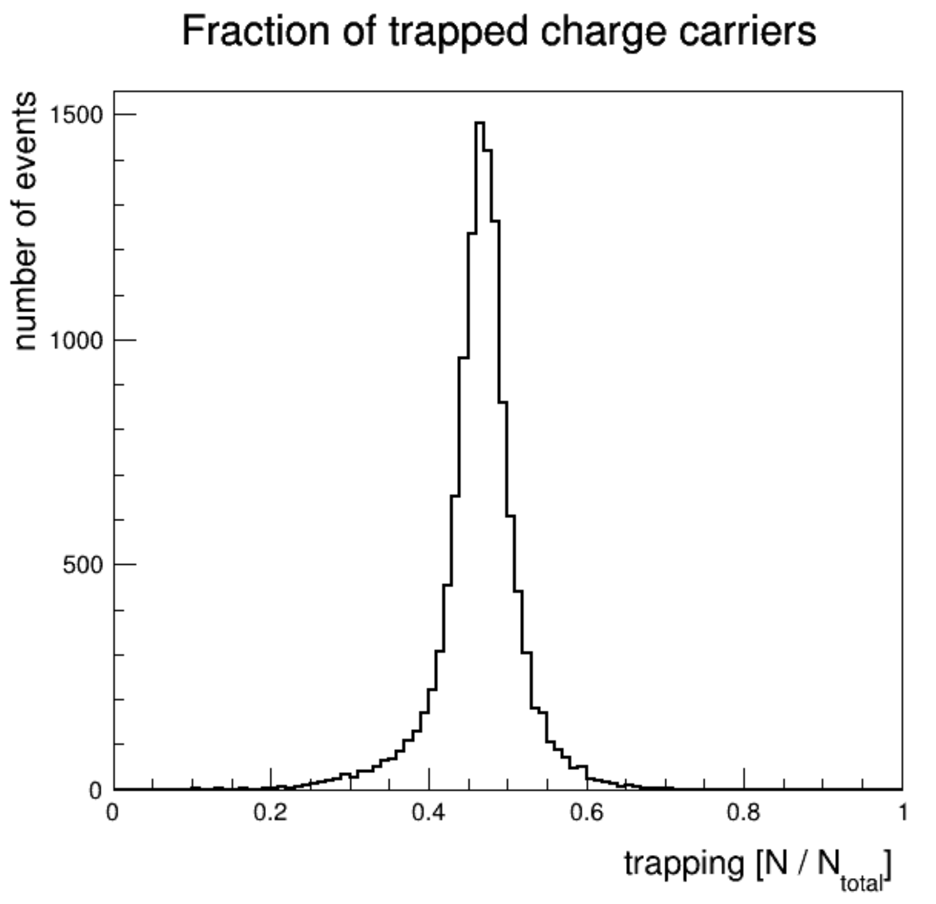
$\phi_{\rm neq}$ [10^{14}neq/cm^2]	N_A [10^{14}cm^{-3}]	N_D [10^{14}cm^{-3}]	σ^e_A [10^{-15}cm^2]	σ^e_D [10^{-15}cm^2]	σ^h_A [10^{-15}cm^2]	σ^h_D [10^{-15}cm^2]
2 (23 GeV) [11]	6.8	10	6.6	6.6	1.65	6.6
6 (23 GeV) [11]	16	40	6.6	6.6	1.65	1.65
12 (23 GeV) [11, 18][4]	30	69	3.8	3.8	0.94	0.94
24 (23 GeV) [11, 18][4]	61	138	3.8	3.8	0.94	0.94
5 (23 MeV) [8]	4.2	13	10	10	10	10
10 (23 MeV) [8]	12.5	52	10	10	10	10

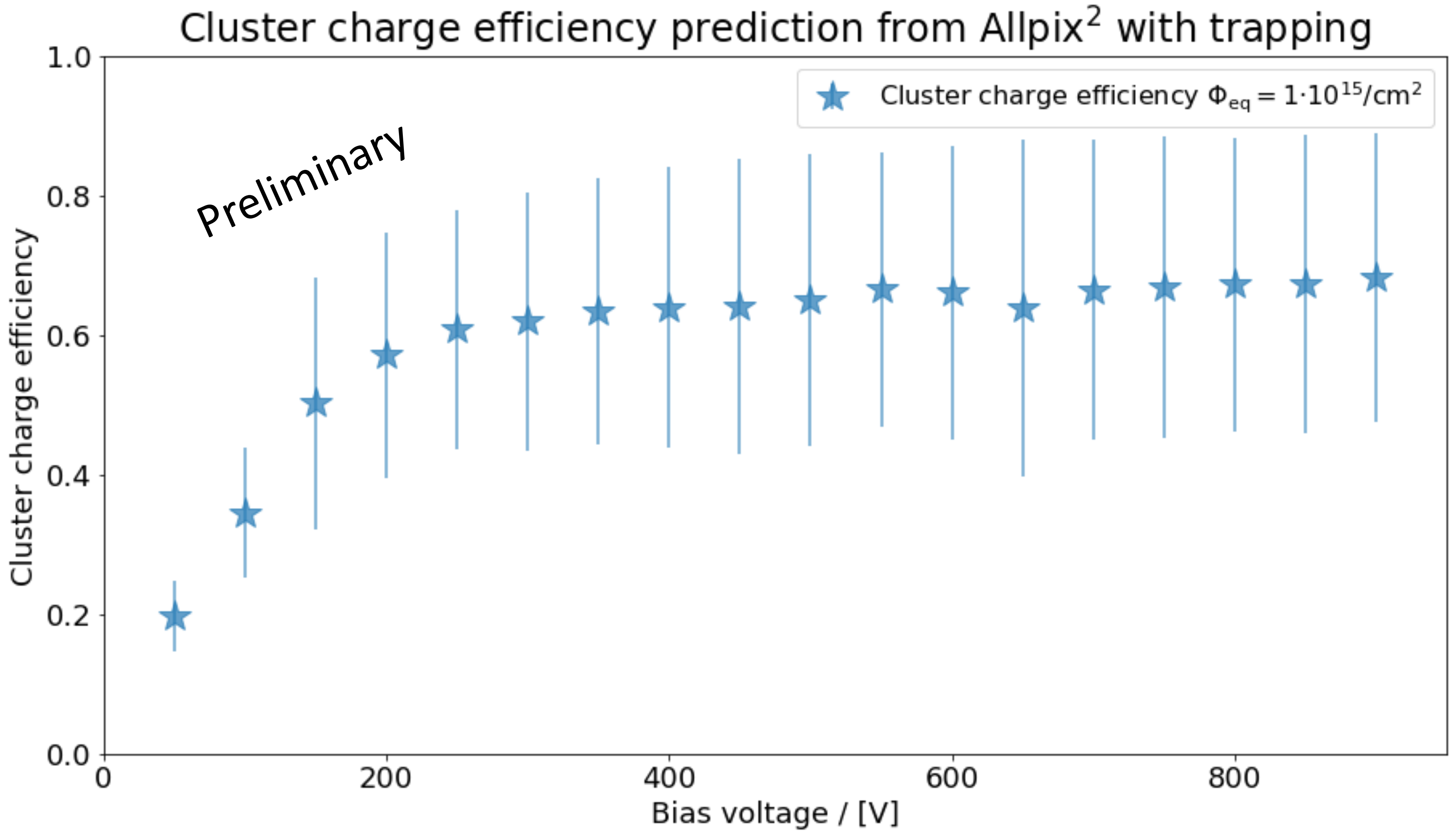


Similarly, implementation of radiation damage is required to study devices for hadron collider experiments

- Clearly contains two components:
  - Modified TCAD electric field profile (proper modelling of space charge effects
  - Charge trapping implementation in allpix^2

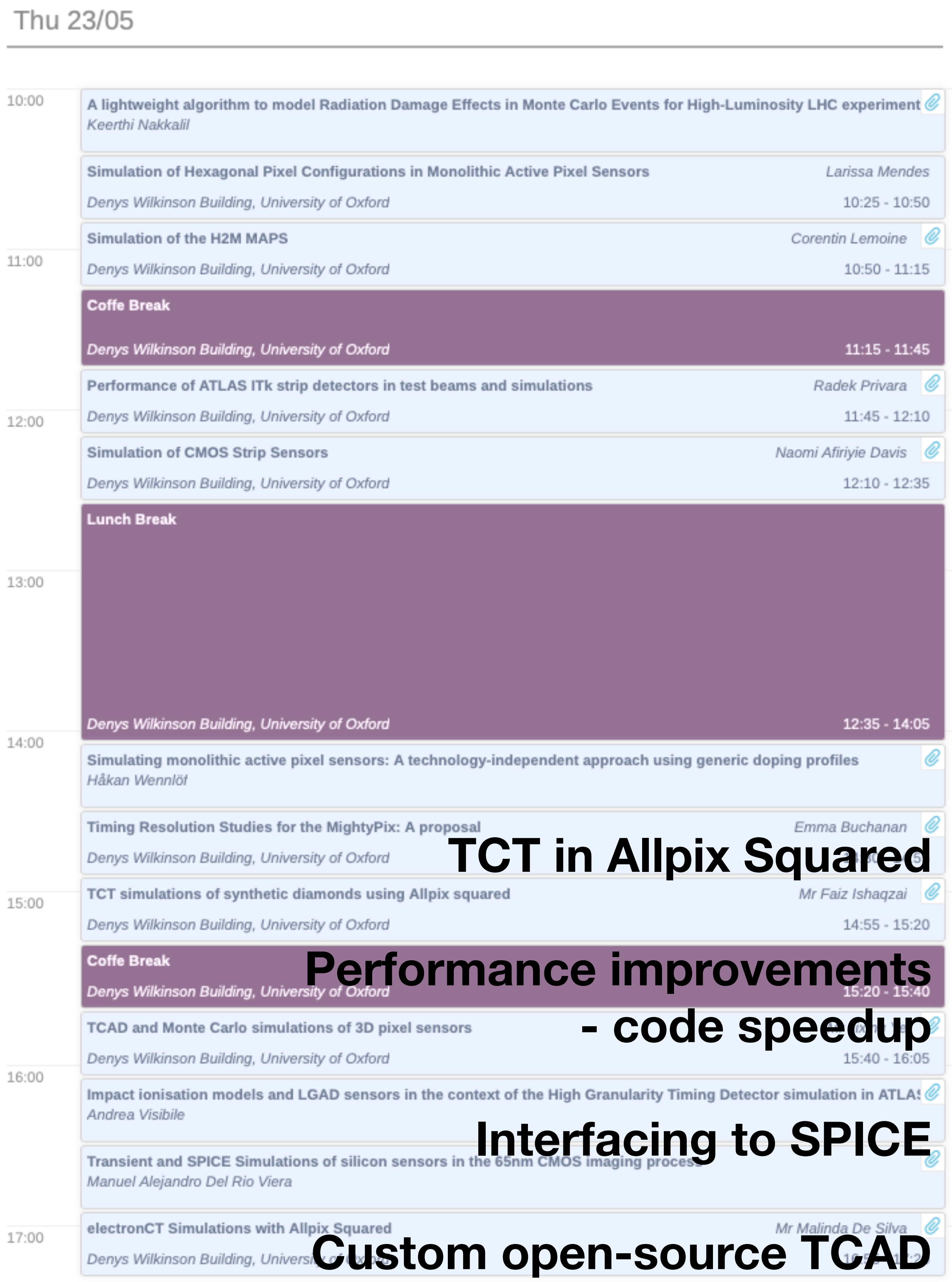






daniel.hynds@physics.ox.ac.uk

A whole lot more



TCT in Allpix Squared

Performance improvements - code speedup

Interfacing to SPICE

Custom open-source TCAD

Summary

A lot of functionality has been implemented in allpix^2, 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 2025!