



[cern.ch/allpix-squared](http://cern.ch/allpix-squared)



# Allpix Squared

Generic Pixel Detector Simulation Framework



Simon Spannagel  
6<sup>th</sup> BTTB Workshop  
Zurich, 18/01/2018

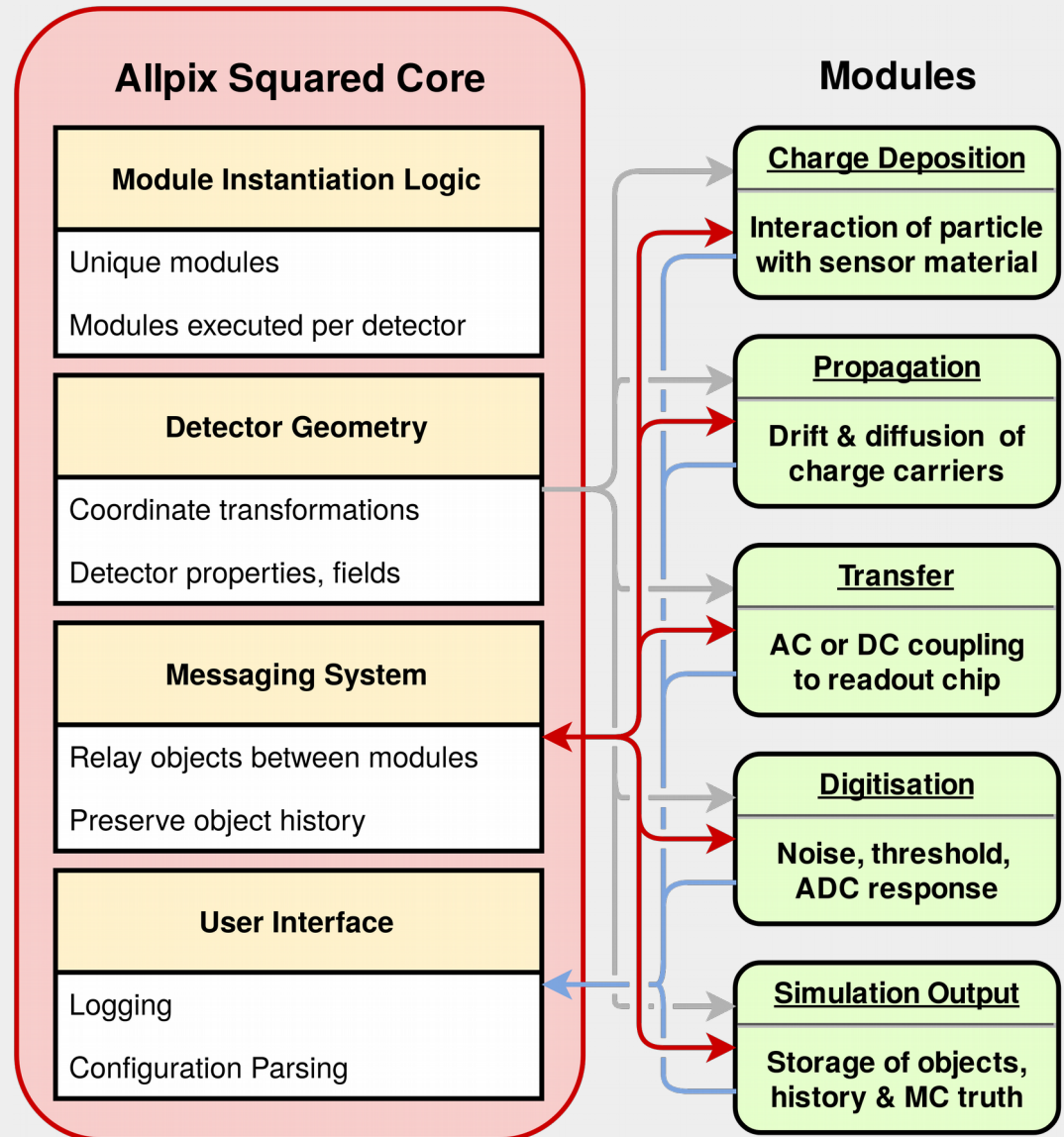
# Why?

- Wanted: simulation software, that was flexible and
  - ...allowed us to **test different** simulation models
  - ...wouldn't require huge efforts to support new detectors
  - ...would facilitate usage of **precise electric fields**
- Many existing frameworks which cover parts of this
- Developed new software within **CLICdp collaboration**
  - Understand prototypes, optimize designs for future sensors
  - Simulate **SOI, HVCMOS, capacitively coupled hybrids, ...**
- Decided to start from scratch, based on established ideas:
  - AllPix – Geant4 for deposition, parametrized detector models
  - PixelAV – Charge propagation with Runge-Kutta-Fehlberg



# The Allpix Squared Framework

- Written in modern C++
- Prov. central components
  - Convenient user interface
  - Logging, configuration
  - Geometry and transformations
- Implement physics in independent modules
  - Plug & play concept
  - IO using ROOT TTrees
- Loading lib, parallelization...



# Configuration

- Framework configured by one file
  - All desired modules listed in order of execution
  - Key-value pairs in **TOML-style**
    - Human readable
    - No overhead (as e.g. XML has)
    - Parsers for many languages
- Support for physical units
  - Never wonder again, what units are used – type them out!
  - Automatic conversion to internal units

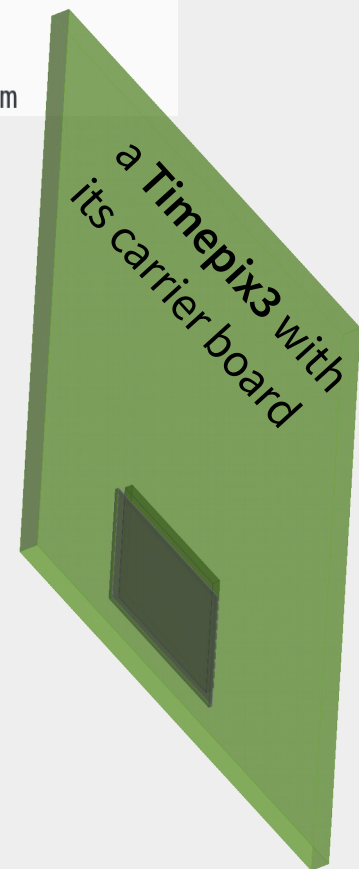
```
1 [AllPix]
2 log_level = "INFO"
3 number_of_events = 500000
4 detectors_file = "telescope.conf"
5
6 [GeometryBuilderGeant4]
7 world_material = "air"
8
9 [DepositionGeant4]
10 physics_list = FTFP_BERT_LIV
11 particle_type = "Pi+"
12 number_of_particles = 1
13 beam_energy = 120GeV
14 # ...
15
16 [ElectricFieldReader]
17 model="linear"
18 bias_voltage=150V
19 depletion_voltage=50V
20
21 [GenericPropagation]
22 temperature = 293K
23 charge_per_step = 10
24 spatial_precision = 0.0025um
25 timestep_max = 0.5ns
26
27 [SimpleTransfer]
28
29 [DefaultDigitizer]
30 adc_resolution = 4
31 adc_slope = 1000
32
33 [DetectorHistogrammer]
34 name = "telescope1"
35 file_name = "telescope1_histograms.root"
36
37 [ROOTObjectWriter]
38 file_name = "simulation_data.root"
```

# Detector Models

- Different detector types available
  - Monolithic detectors
  - Hybrid detectors w/ bump bonds
- Easy configuration through model files
  - Give it a name, decide on the type
  - Set detector parameters
- Some model files already shipped with the framework, at the moment:

ATLAS FE-I3, FE-I4, CMS PSI46/dig, Medipix3, Timepix3, CLICpix, CLICpix2, Mimosas23, Mimosas26

```
1 type = "hybrid"
2
3 number_of_pixels = 256 256
4 pixel_size = 55um 55um
5
6 sensor_thickness = 300um
7 chip_thickness = 700um
8
9 # ...
10
11 [support]
12 thickness = 1.76mm
```

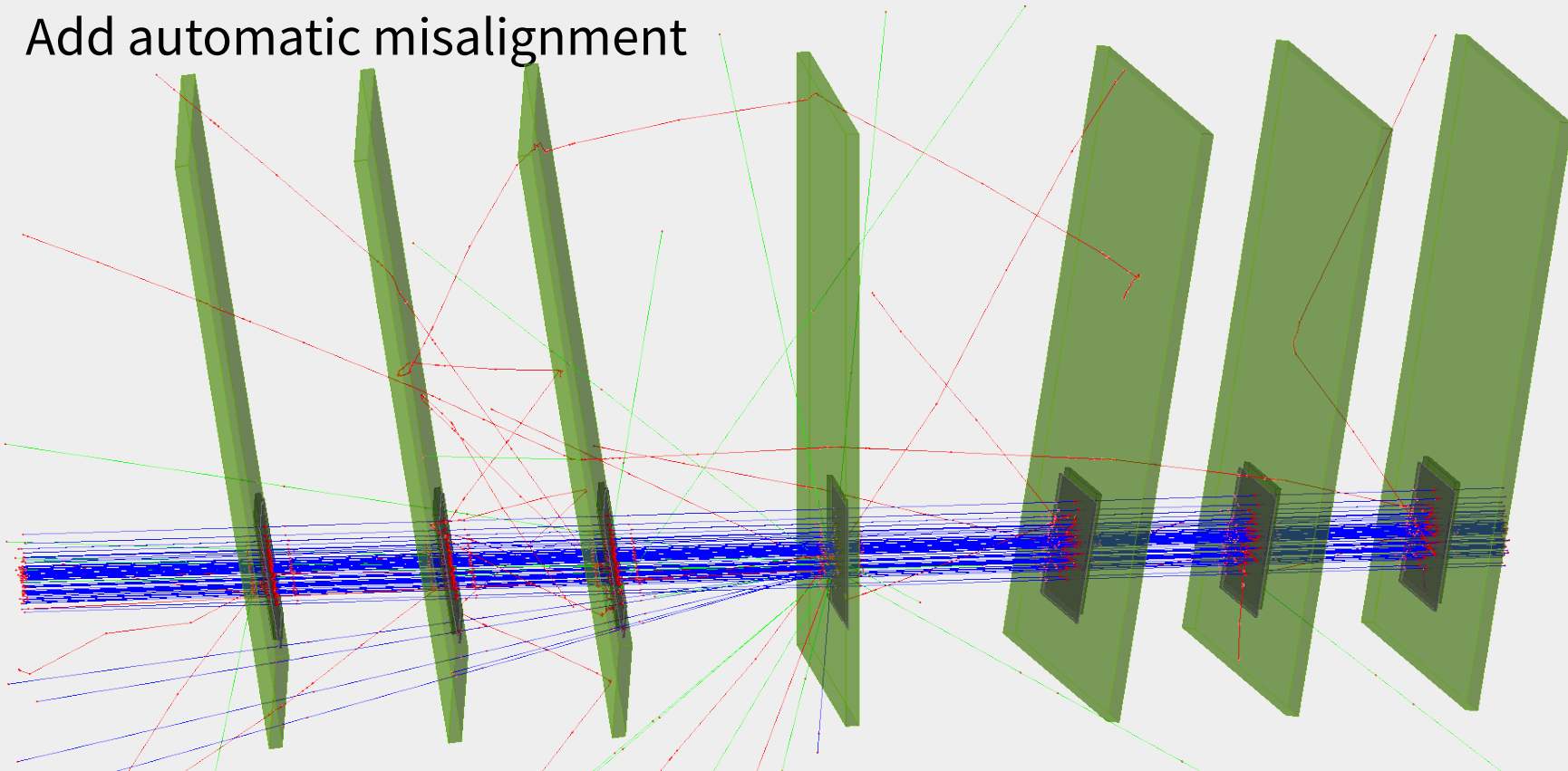


# Experimental Setup

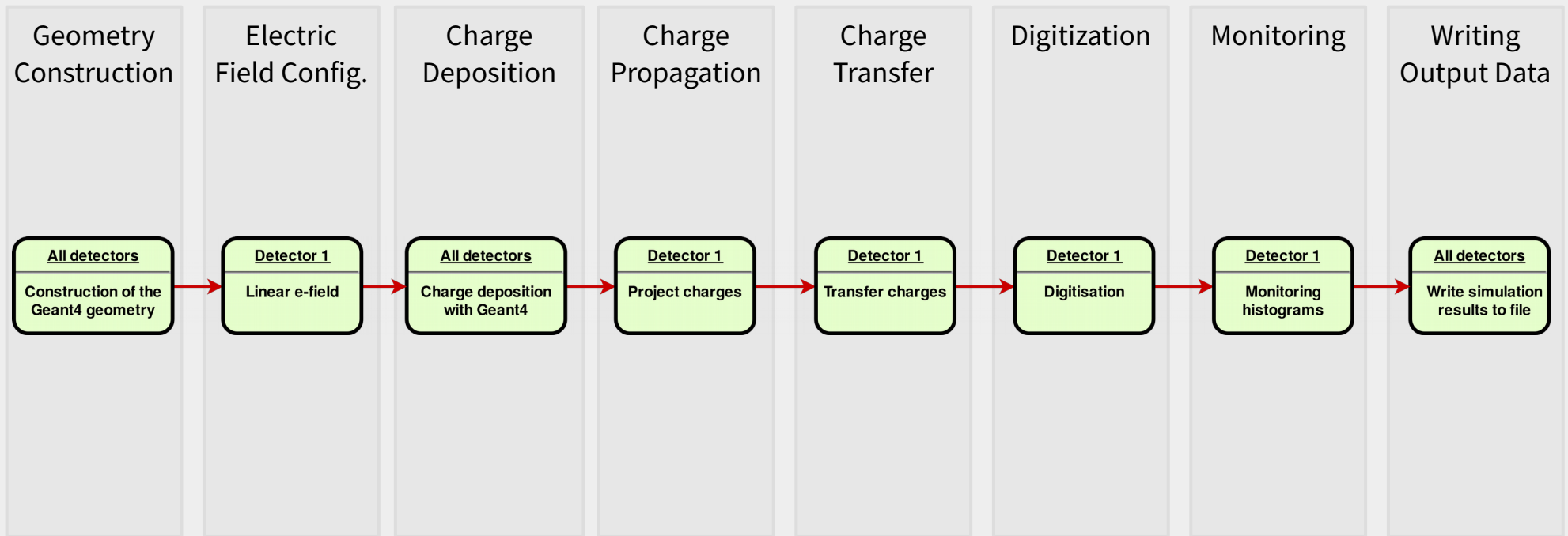
- Set of detectors in separate file
  - Using pre-defined detector models
  - Set position & orientation
- Possibility to overwrite model parameters
- Add automatic misalignment

```

1 [telescope0]
2 type = "timepix"
3 position = 0mm 0mm 10mm
4 orientation = -15deg 12deg 8deg
5
6 [telescope1]
7 type = "timepix"
8 position = 0mm 0mm 150mm
9 orientation = 9deg 180deg 0deg
10 alignment_precision_position = 50um 50um 1mm
  
```



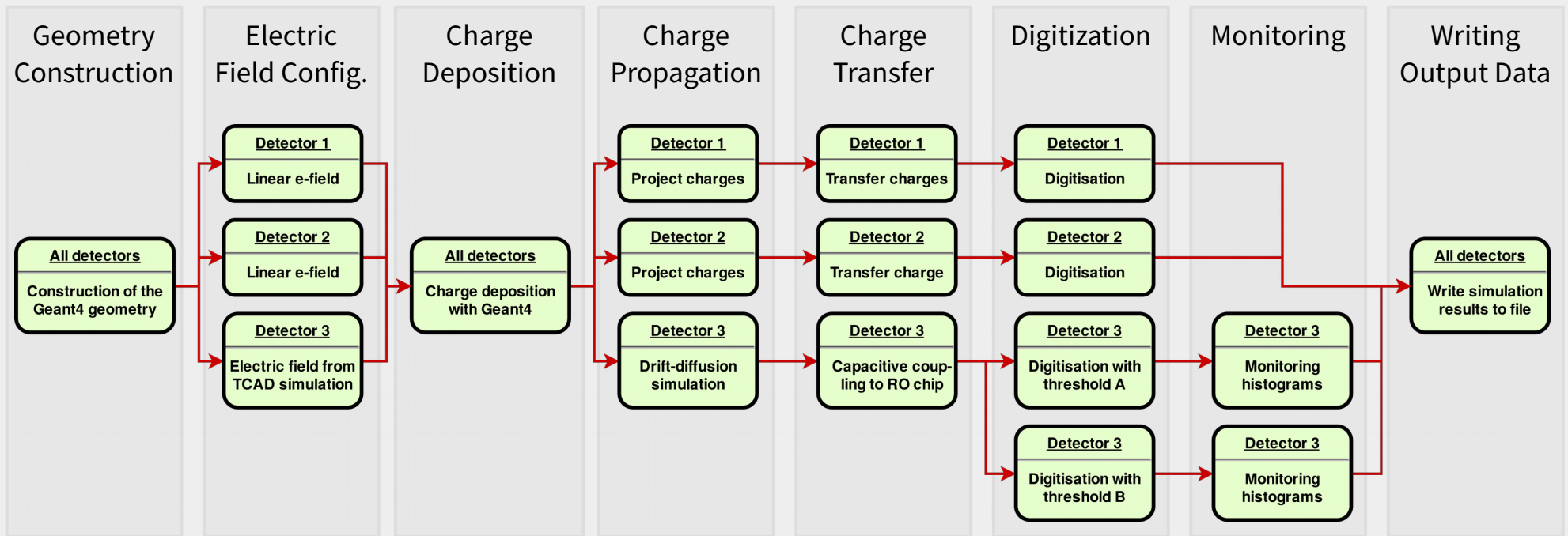
# Modular Approach



- Modules are placed in linear order in the configuration file
- Select suitable deposition, propagation, ... modules
- Approach allows to quickly plug together simple **simulations** (start from examples shipped in the repository)



# Modular Approach (II)



- ...as well as more involved setups
- E.g. have detailed simulation for **device under test** and quicker, less detailed simulations for **telescope planes**
- Apply different settings in the same simulation run

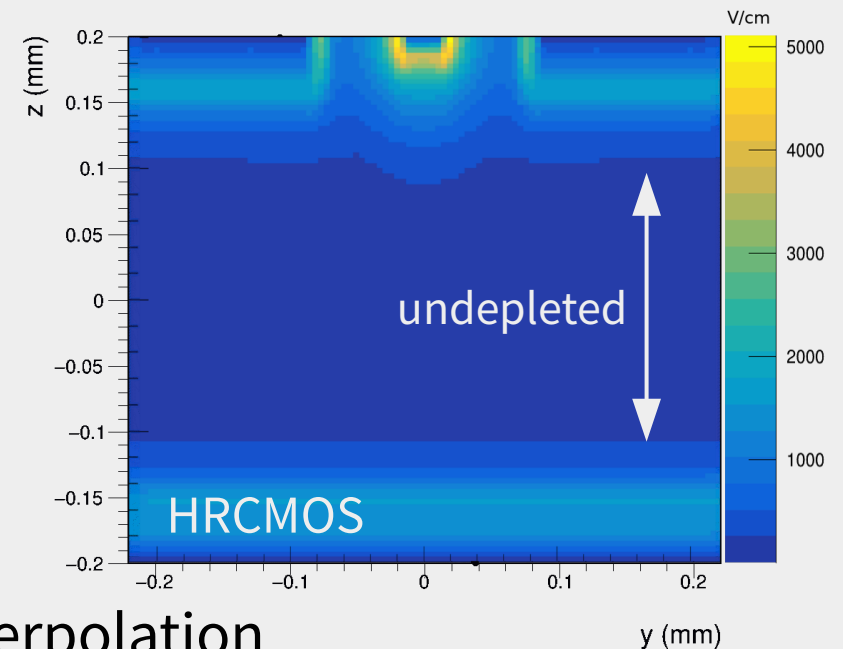
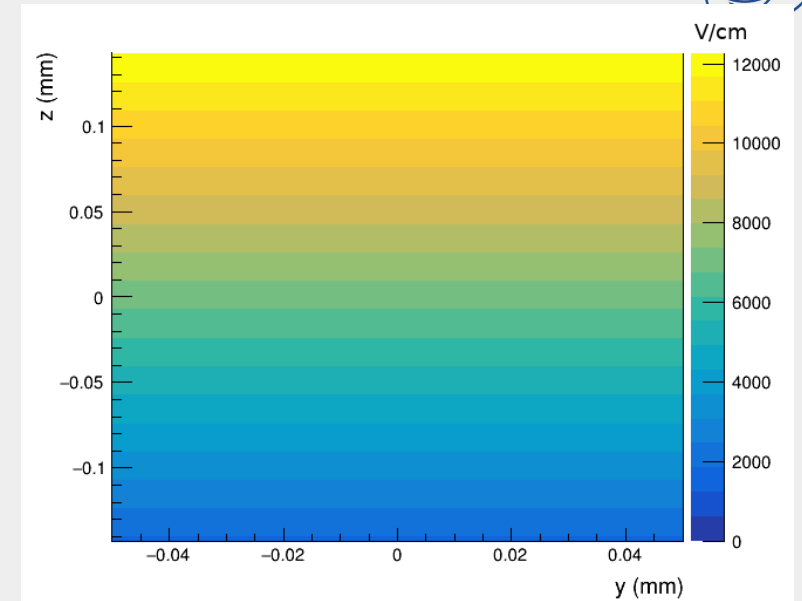
# Electric Fields

- **Option 1: Linear electric field**

- Good approximation for thick, planar sensors
- Defined by setting
  - Depletion + bias voltage

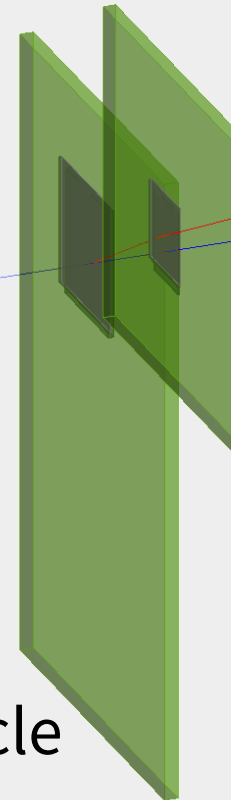
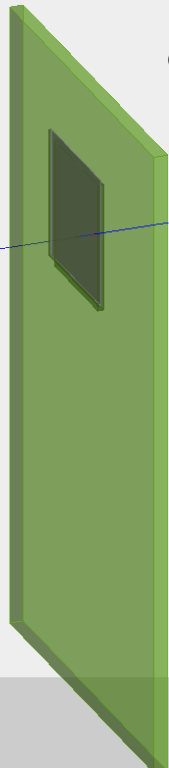
- **Option 2: Electric field from TCAD**

- Detailed e-static simulation, drift along field lines
- **TCAD DF-ISE Mesh Converter**
  - Converts adaptive to regular
  - 3D, 2D quantities, barycentric interpolation



# Charge Deposition: Geant4

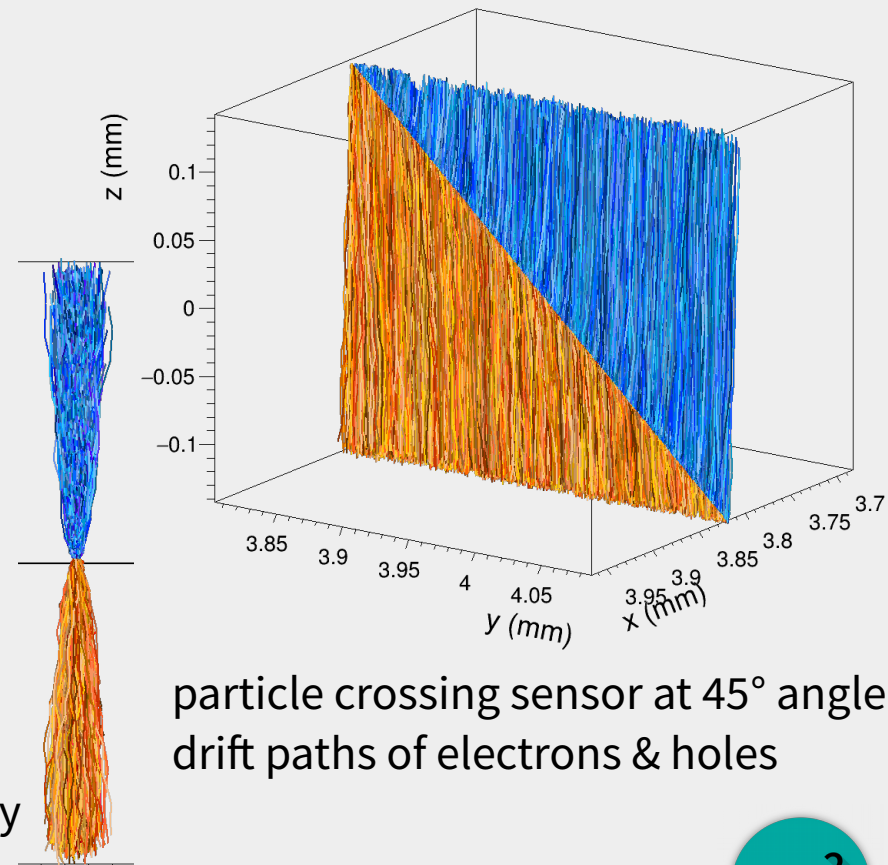
- Dependency on Geant4 only on module level
  - Framework can be built without Geant4
  - Geant4 interaction abstracted by module
- Define particle type (name or PDG code)
  - Beam energy, position, direction, size
    - # of particles / event
- Select physics list, possibly enable PAI
- BeamOn called for every particle



# Charge Propagation

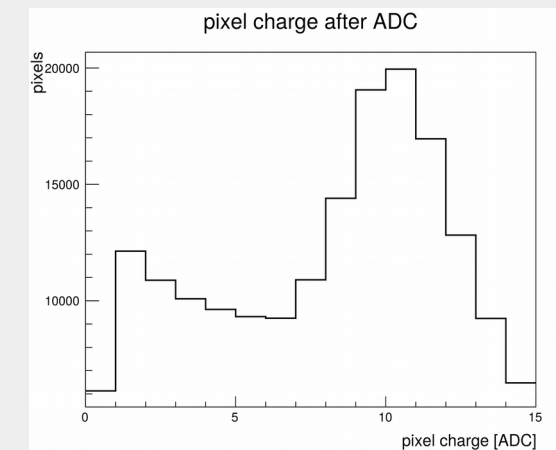
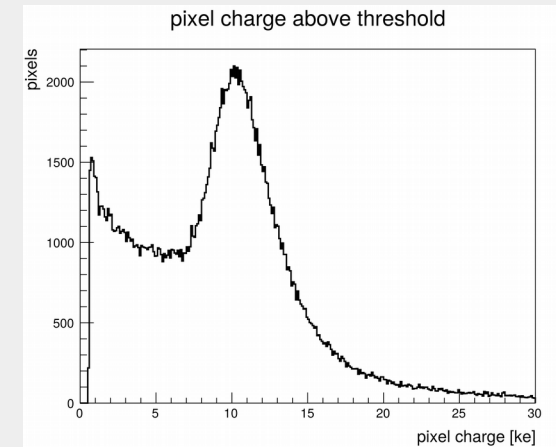
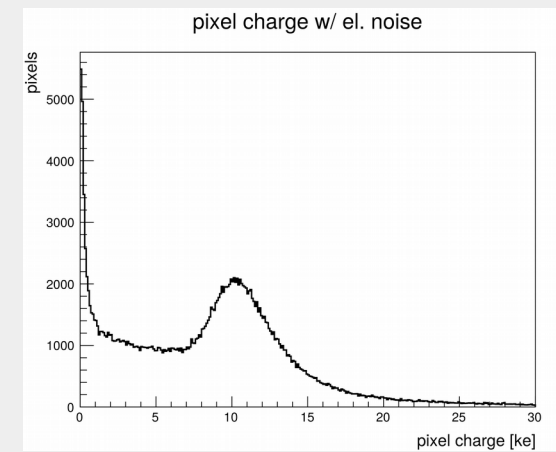
- Pick up deposited charge carriers, transport to implants
- **Option 1: Projection**
  - Project charges on implant side, adding diffusion
  - Only for linear fields, integrating
- **Option 2: Drift-Diffusion model**
  - Drift along field lines using 4<sup>th</sup> order Runge-Kutta-Fehlberg method
  - Transient information available
  - Allows to create drift animation
    - See [website](#)

projection along  
particle trajectory



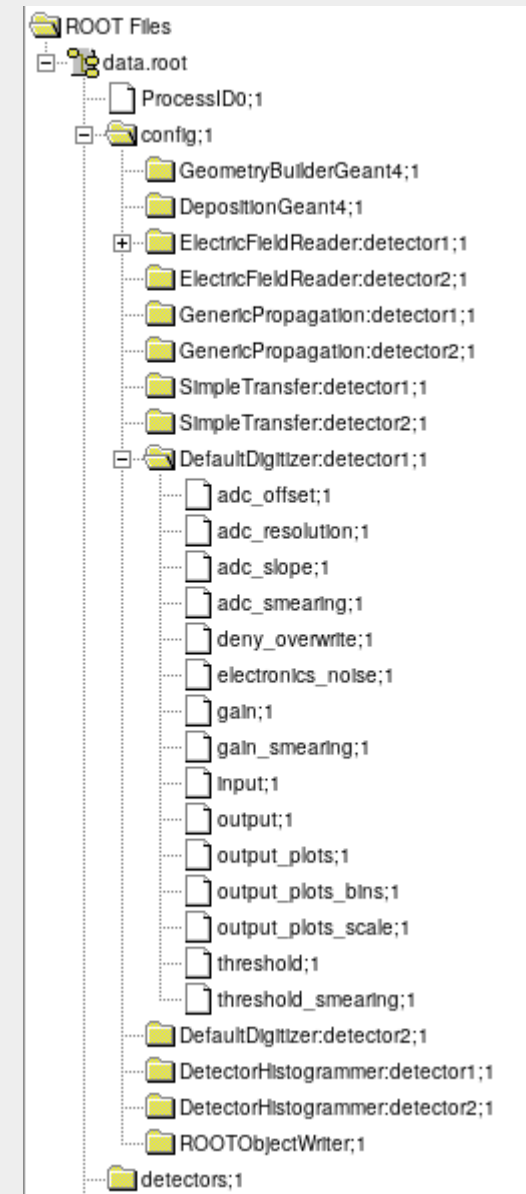
# Digitization

- Simulation of readout chip circuitry
- Different (optional) stages:
  - Electronics noise
  - Amplifier gain & gain smearing
  - Threshold & smearing
  - ADC resolution & smearing
  - ADC calibration (currently: linear)
- Monitoring plots for each step produced



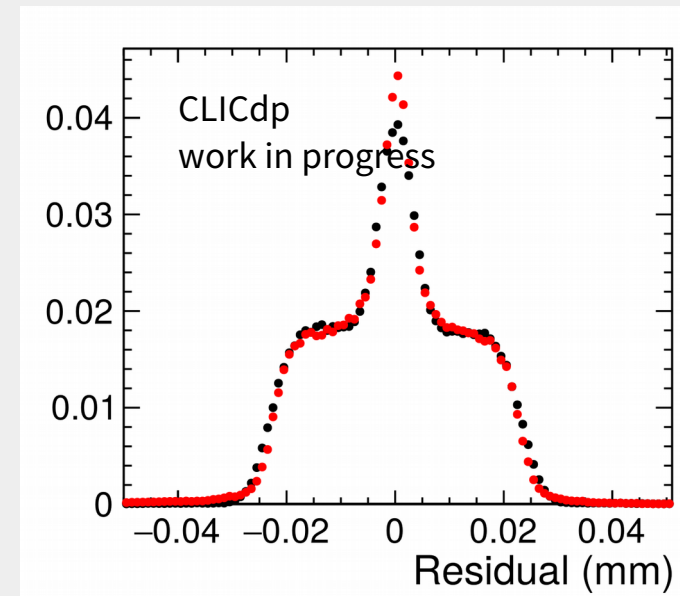
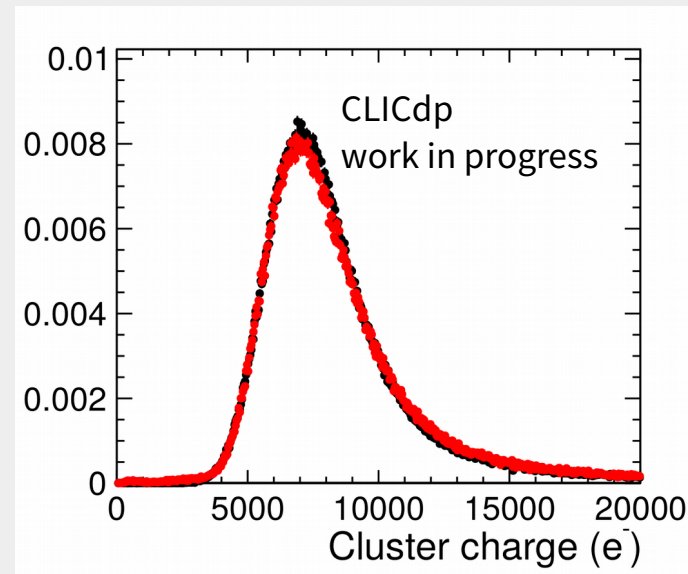
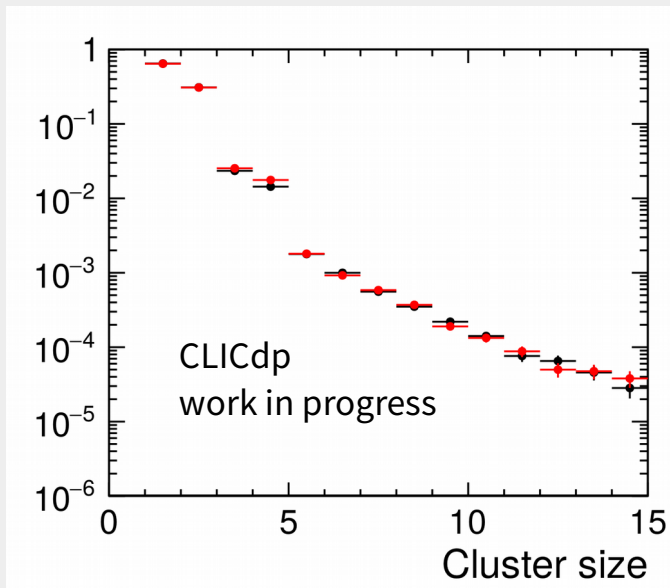
# Data Storage

- Different output writers available:
  - LCIO (EUTelescope), RCE (Proteus), ...
- Native format: ROOT files with all objects
  - Also contains detectors & sim. parameters
  - Full framework **configuration** can be **reconstructed from** single **data file**
- ROOTObject reader replays data from file
  - Simulate deposition & propagation once
  - Read data from file and **quickly repeat** final digitization **step** with different parameters



# Comparison – Work in Progress

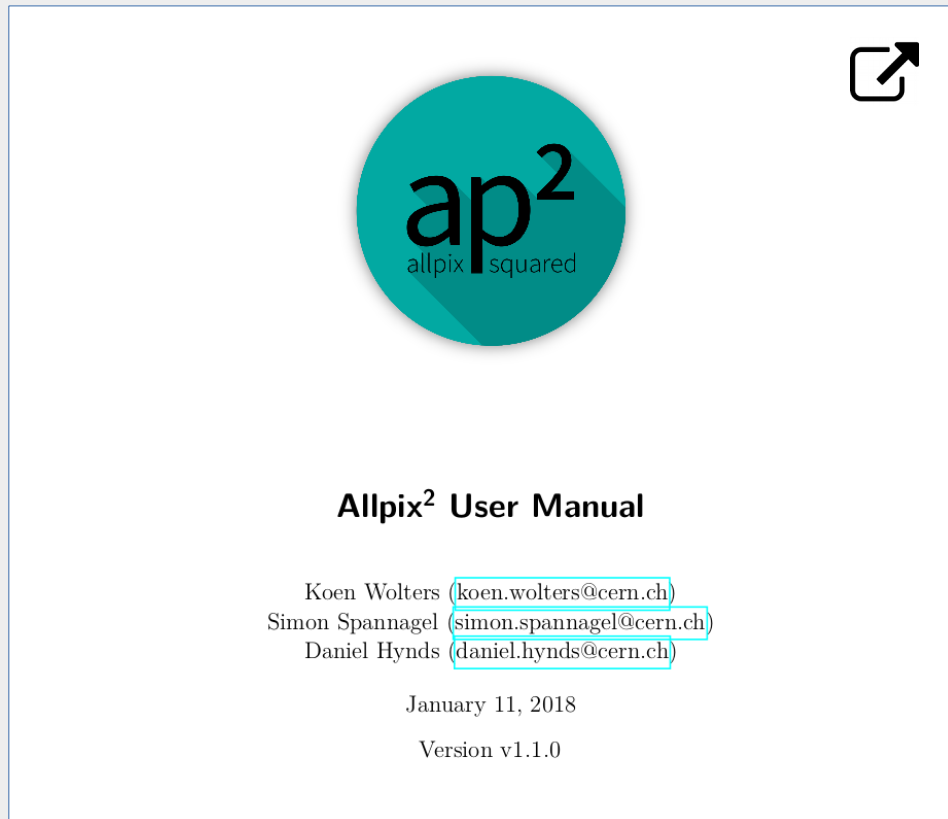
- **CLICdp Timepix3 Telescope + DUT: 100 $\mu$ m planar sensor**
- “simply” simulated device with parameters taken from data
  - Bias/depletion voltages, temperature, threshold
- Reconstruction with same cuts & corrections (e.g. eta-corr)



**Data** / **Allpix<sup>2</sup>**

# Documentation & Manuals

- Extensive User Manual ~115 pages (PDF/TeX)



- Well-documented code (Doxygen)
- Module documentation (Markdown)

## GenericPropagation

Maintainer: Koen Wolters (koen.wolters@cern.ch), Simon Spannagel (simon.spannagel@cern.ch)

Status: Functional  
Input: DepositedCharge  
Output: PropagatedCharge

### Description

Simulates the propagation of electrons and/or holes through the sensitive sensor volume of the detector. It allows to propagate sets of charge carriers together in order to speed up the simulation while maintaining the required accuracy. The propagation process for these sets is fully independent and no interaction is simulated. The maximum size of the set of propagated charges and thus the accuracy of the propagation can be controlled.

The propagation consists of a combination of drift and diffusion simulation. The drift is calculated using the charge carrier velocity derived from the charge carrier mobility parameterization by C. Jacoboni et al. @Jacoboni. The correct mobility for either electrons or holes is automatically chosen, based on the type of the charge carrier under consideration. Thus, also input with both electrons and holes is treated properly.

The two parameters `propagate_electrons` and `propagate_holes` allow to control which type of charge carrier is propagated to their respective electrodes. Either one of the carrier types can be selected, or both can be propagated. It should be noted that this will slow down the simulation considerably since twice as many carriers have to be handled and it should only be used where sensible. The direction of the propagation depends on the electric field configured, and it should be ensured that the carrier types selected are actually transported to the implant side. For linear electric fields, a warning is issued if a possible misconfiguration is detected.

A fourth-order Runge-Kutta-Fehlberg method with fifth-order error estimation is used to integrate the electric field. After every Runge-Kutta step, the diffusion is accounted for by applying an offset drawn from a Gaussian distribution calculated from the Einstein relation

$$\sigma = \sqrt{\frac{2k_B T}{q} \mu \Delta t}$$

using the carrier mobility  $\mu$ , the temperature  $T$  and the time step  $\Delta t$ . The propagation stops when the set of charges reaches any surface of the sensor.

The propagation module also produces a variety of output plots. These include a 3D line plot of the path of all separately propagated charge carrier sets from their point of deposition to the end of their drift, with nearby paths having different colors. In this coloring scheme, electrons are marked in blue colors, while holes are presented in different shades of orange. In addition, a 3D GIF animation for the drift of all individual sets of charges (with the size of the point proportional to the number of charges in the set) can be produced. Finally, the module produces 2D contour animations in all the planes normal to the X, Y and Z axis, showing the concentration flow in the sensor. It should be noted that generating the animations is very time-consuming and should be switched off even when investigating drift behavior.

### Dependencies

This module requires an installation of Eigen3.

### Parameters

- **temperature**: Temperature of the sensitive device, used to estimate the diffusion constant and therefore the strength of the diffusion. Defaults to room temperature (293.15K).
- **charge\_per\_step**: Maximum number of charge carriers to propagate together. Divides the total number of deposited charge carriers at a specific point into sets of this number of charge carriers and a set with the remaining charge carriers. A value of 10 charges per step is used by default if this value is not specified.
- **spatial\_precision**: Spatial precision to aim for. The timestep of the Runge-Kutta propagation is adjusted to reach this spatial precision after calculating the uncertainty from the fifth-order error method. Defaults to 0.1nm.
- **timestep\_start**: Timestep to initialize the Runge-Kutta integration with. Appropriate initialization of this parameter reduces the time to optimize the timestep to the spatial precision parameter. Default value is 0.07ns.
- **timestep\_min**: Minimum step in time to use for the Runge-Kutta integration regardless of the spatial precision. Defaults to 0.0ps.
- **timestep\_max**: Maximum step in time to use for the Runge-Kutta integration regardless of the spatial precision. Defaults to 0.1ns.
- **integration\_time**: Time within which charge carriers are propagated. After exceeding this time, no further propagation is performed for the respective carriers. Defaults to the LHC bunch crossing time of 25ns.
- **propagate\_electrons**: Select whether electron-type charge carriers should be propagated to the electrodes. Defaults to true.
- **propagate\_holes**: Select whether hole-type charge carriers should be propagated to the electrodes. Defaults to false.
- **output\_plots**: Determines if output plots should be generated for every event. This causes a significant slow down of the simulation, it is not recommended to enable this option for runs with more than a couple of events. Disabled by default.
- **output\_plots\_step**: Timestep to use between two points plotted. Indirectly determines the amount of points plotted. Defaults to timestep\_max if not explicitly specified.
- **output\_plots\_theta**: Viewpoint angle of the 3D animation and the 3D line graph around the world X-axis. Defaults to zero.
- **output\_plots\_phi**: Viewpoint angle of the 3D animation and the 3D line graph around the world Z-axis. Defaults to zero.
- **output\_plots\_use\_pixel\_units**: Determines if the plots should use pixels as unit instead of metric length scales. Defaults to false (thus using the metric system).
- **output\_plots\_use\_equal\_scaling**: Determines if the plots should be produced with equal distance scales on every axis (also if this implies that some points will fall out of the graph). Defaults to true.
- **output\_plots\_align\_pixels**: Determines if the plot should be aligned on pixels, defaults to false. If enabled the start and the end of the axis will be at the split point between pixels.
- **output\_animations**: In addition to the other output plots, also write a GIF animation of the charges drifting towards the electrodes. This is very slow and writing the animation takes a considerable amount of time, therefore defaults to false. This option also requires `output_plots` to be enabled.
- **output\_animations\_time\_scaling**: Scaling for the animation used to convert the actual simulation time to the time step in the animation. Defaults to 1.0e9, meaning that every nanosecond of the simulation is equal to an animation step of a single second.
- **output\_animations\_marker\_size**: Scaling for the markers on the animation, defaults to one. The markers are already internally scaled to the charge of their step, normalized to the maximum charge.
- **output\_animations\_contour\_max\_scaling**: Scaling to use for the contour color sets from the theoretical maximum charge at every single plot step. Default is 10, meaning that the maximum of the color scale axis is equal to the total amount of charges divided by ten (values above this are displayed in the same maximum color). Parameter can be used to improve the color scale of the contour plots.
- **output\_animations\_color\_markers**: Determines if colors should be for the markers in the animations, defaults to false.

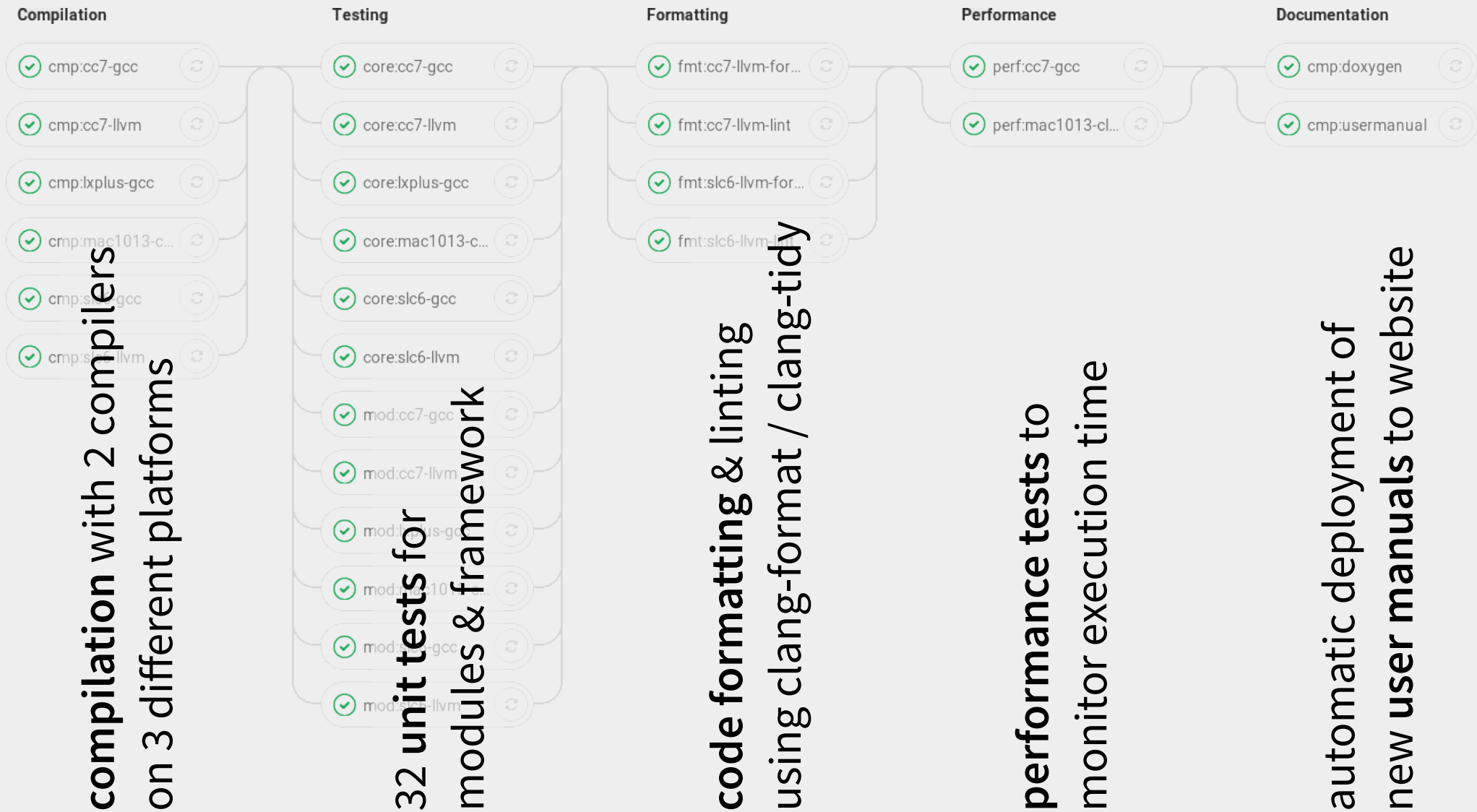
### Usage

An example of generic propagation for all sensors of type "Timepix" at room temperature using packets of 25 charges is the following

```
[GenericPropagation]
type = "Timepix"
temperature = 293K
charge_per_step = 25
```






# Continuous Integration & Testing

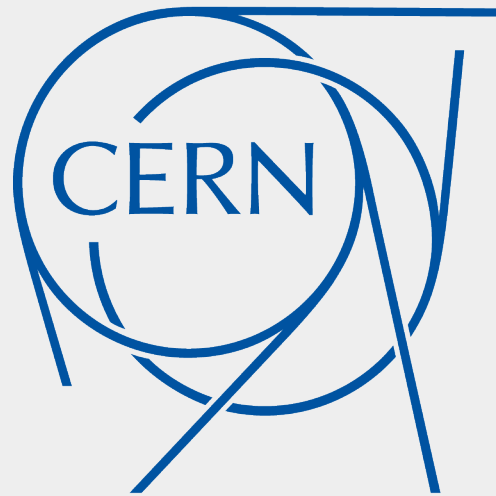


# Summary & Outlook

- Generic, modular framework to simulate pixel detectors
  - Reads TCAD e-fields, implements drift-diffusion model
  - Modern, documented code & extensive user manual
- First comparison with test beam data
  
- Version 1.1 released last week
  - New module, many improvements to framework
  - Check release notes for details  
<https://cern.ch/allpix-squared/post/2018-01-12-allpix-squared-1.1-released/>
- Welcoming **contributions from the community**
- **Tutorial** this afternoon

# Resources

-  Website  
<https://cern.ch/allpix-squared>
  - Release Notes, Information, User Manual, Code Reference
-  Repository  
<https://gitlab.cern.ch/simonspa/allpix-squared>
  - Source Code, Issue Tracker
-  Mailing Lists:
  - allpix-squared-users <https://e-groups.cern.ch/e-groups/Egroup.do?egroupId=10262858>
  - allpix-squared-developers <https://e-groups.cern.ch/e-groups/Egroup.do?egroupId=10273730>
- User Manual:  
<https://cern.ch/allpix-squared/usermanual/allpix-manual.pdf>



# Automatic Detector Misalignment

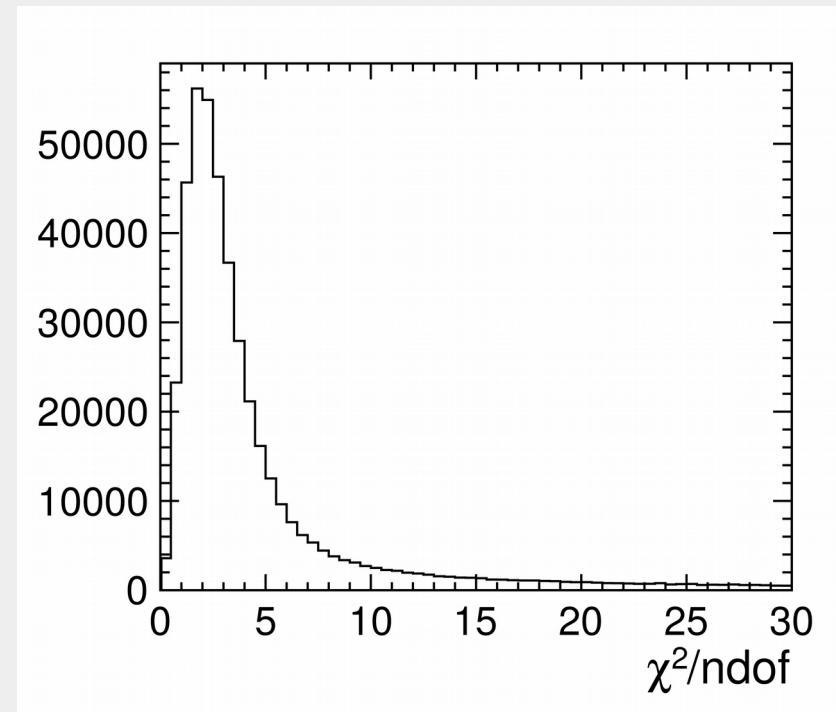
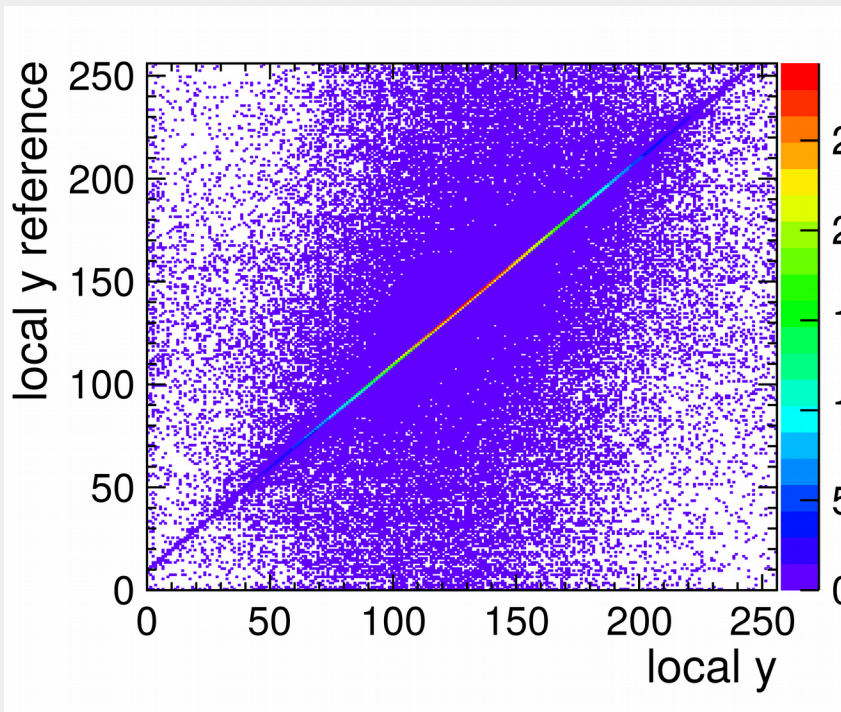
- Artifacts in reconstructed simulation residuals
  - From pixel-perfect alignment, cured by misaligning detectors
- New parameters for detectors:

```
25  
26 alignment_precision_position = 0.1mm 0.1mm 1mm  
27 alignment_precision_orientation = 0.1deg 1.3deg 2deg  
28
```

- To be added in detectors file for each detector individually
- Parameters define Gaussian width to draw shifts from
- Reproducible misalignments:
  - Set **random\_seed\_core** to known value

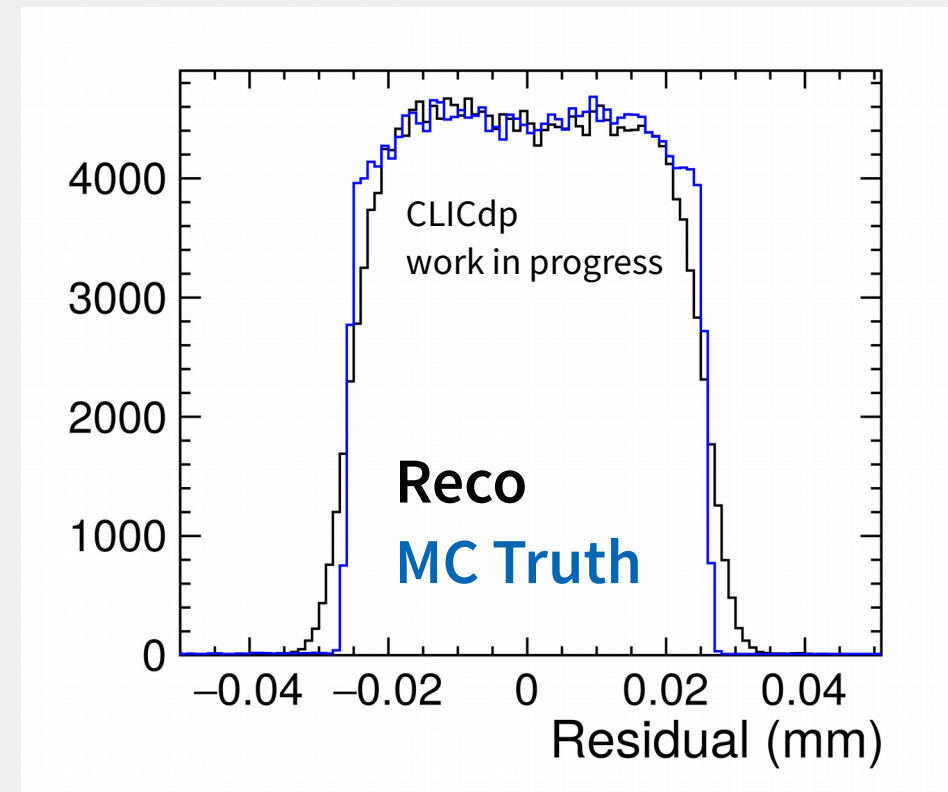
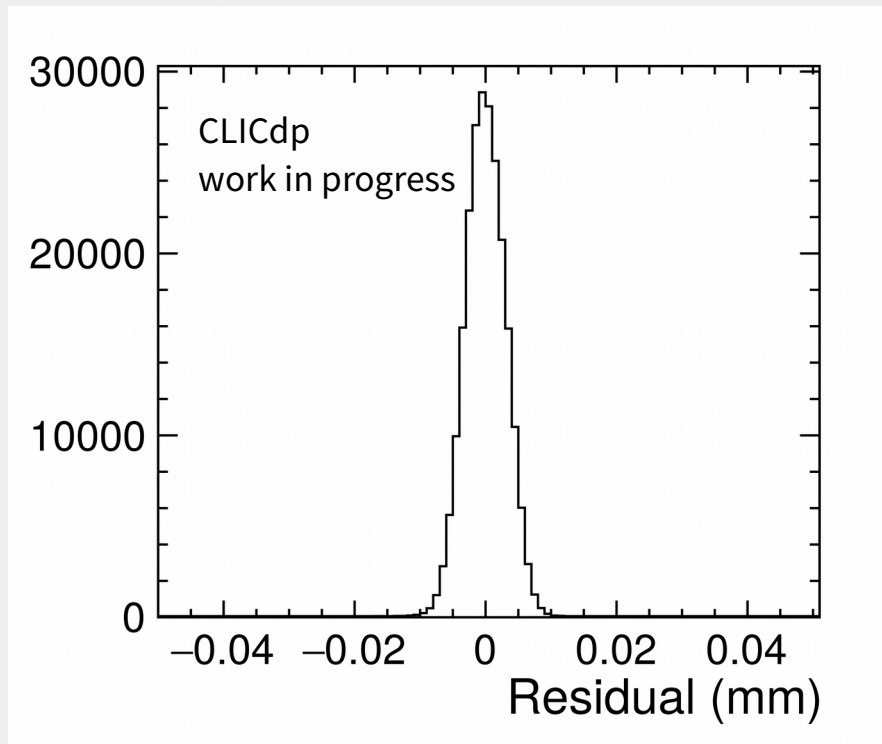
# Reconstruction

- Framework allows to simulate full setup, not only DUT
  - Beam telescopes, source measurements...
- Possible to use different methods
  - Precise simulation in DUT
  - More coarse simulation for telescope planes



# Monte Carlo Truth Information

- Monte Carlo Truth information available
  - Entry & exit point of primary and secondary particles
- Compare tracks reconstructed from telescope simulation with MC truth



# Module Tests

```

Test project /builds/simonspa/allpix-squared/build
  Start 15: test_modules/test_08-1_writer_root.conf
  Start 3: test_modules/test_01_geobuilder.conf
  Start 4: test_modules/test_02-1_electricfield_linear.conf
  Start 5: test_modules/test_02-2_electricfield_init.conf
1/18 Test #4: test_modules/test_02-1_electricfield_linear.conf ..... Passed    1.10 sec
  Start 6: test_modules/test_03-1_deposition.conf
2/18 Test #3: test_modules/test_01_geobuilder.conf ..... Passed    1.30 sec
3/18 Test #5: test_modules/test_02-2_electricfield_init.conf ..... Passed    1.40 sec
  Start 7: test_modules/test_03-2_deposition_mc.conf
  Start 8: test_modules/test_04-1_propagation_project.conf
4/18 Test #15: test_modules/test_08-1_writer_root.conf ..... Passed    4.67 sec
  Start 9: test_modules/test_04-2_propagation_generic.conf
5/18 Test #6: test_modules/test_03-1_deposition.conf ..... Passed    4.27 sec
6/18 Test #8: test_modules/test_04-1_propagation_project.conf ..... Passed    4.07 sec
7/18 Test #7: test_modules/test_03-2_deposition_mc.conf ..... Passed    4.11 sec
  Start 10: test_modules/test_05_transfer_simple.conf
  Start 11: test_modules/test_06-1_digitization_charge.conf
  Start 12: test_modules/test_06-2_digitization_adc.conf
8/18 Test #12: test_modules/test_06-2_digitization_adc.conf ..... Passed    3.65 sec
9/18 Test #9: test_modules/test_04-2_propagation_generic.conf ..... Passed    4.20 sec
10/18 Test #10: test_modules/test_05_transfer_simple.conf ..... Passed    3.68 sec
11/18 Test #11: test_modules/test_06-1_digitization_charge.conf ..... Passed    3.68 sec
  Start 13: test_modules/test_06-3_digitization_gain.conf
  Start 14: test_modules/test_07_histogramming.conf
  Start 16: test_modules/test_08-2_writer_rce.conf
  Start 17: test_modules/test_08-3_writer_lcio.conf
12/18 Test #13: test_modules/test_06-3_digitization_gain.conf ..... Passed    3.77 sec
13/18 Test #17: test_modules/test_08-3_writer_lcio.conf ..... Passed    3.77 sec
14/18 Test #14: test_modules/test_07_histogramming.conf ..... Passed    3.77 sec
  Start 18: test_modules/test_08-4_writer_corrvreckan.conf
  Start 19: test_modules/test_08-5_writer_corrvreckan_mc.conf
  Start 20: test_modules/test_09_reader_root.conf
15/18 Test #16: test_modules/test_08-2_writer_rce.conf ..... Passed    4.18 sec
16/18 Test #20: test_modules/test_09_reader_root.conf ..... Passed    0.92 sec
17/18 Test #18: test_modules/test_08-4_writer_corrvreckan.conf ..... Passed    3.69 sec
18/18 Test #19: test_modules/test_08-5_writer_corrvreckan_mc.conf ... Passed    3.69 sec

100% tests passed, 0 tests failed out of 18

```

- Tests for every module
- Fixed random seed
  - Reproduces same output
- Matching regular expressions
- Single change (1e difference) fails the CI
  - Invaluable for monitoring framework
  - Catching issues before merging code



# Adding New Unit Tests

- Very simple: **only requires configuration file!**
- Regular expression defines test condition
- More elaborate description to be found in the user manual

```
1  [Allpix]
2  detectors_file = "detector_rotate_misaligned.conf"
3  log_level = "TRACE"
4  number_of_events = 0
5  random_seed = 0
6  random_seed_core = 0
7
8  [GeometryBuilderGeant4]
9
10 ∨ #PASS (DEBUG)  misaligned: (8.72466deg,171.099deg,178.504deg)
11 #PASSOSX (DEBUG) misaligned: (9.04007deg,170.886deg,178.731deg)
```