



cern.ch/allpix-squared



Allpix Squared

A Generic Pixel Detector Simulation Framework

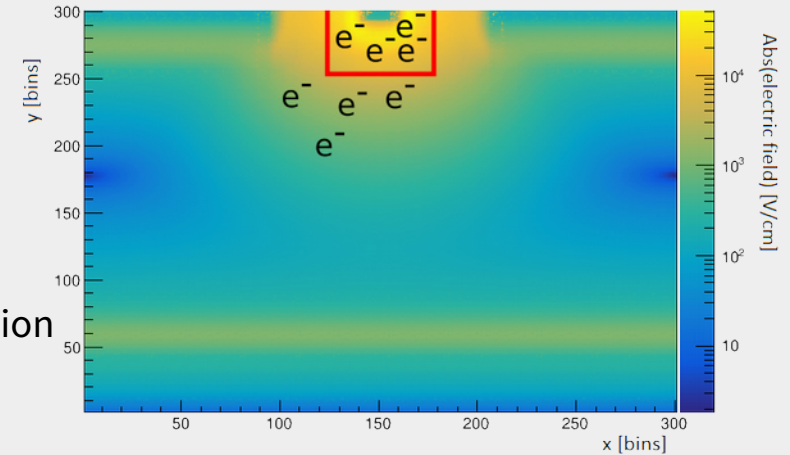
Simon Spannagel, CERN

EP-SFT Group Meeting

CERN, 08 April 2019

The Challenge

- Silicon detectors widely used in HEP – vertex & tracking, timing detectors, calorimeters
- New commercial CMOS technologies entering the market
 - Strong drive towards monolithic detectors
 - Entails complex sensor layouts and charge collection
- **Wanted:** flexible simulation software, that
 - ...allowed to **test different** simulation **models** for signal formation
 - ...allow parametrized detector models
 - ...would facilitate usage of **precise electric fields**
- Developed new framework within **CLICdp collaboration**, based on established ideas:
 - **AllPix** – Geant4 user classes for digitizer development of large simulations: parametrized detector models
 - **PixelAV** – Charge propagation with Runge-Kutta-Fehlberg



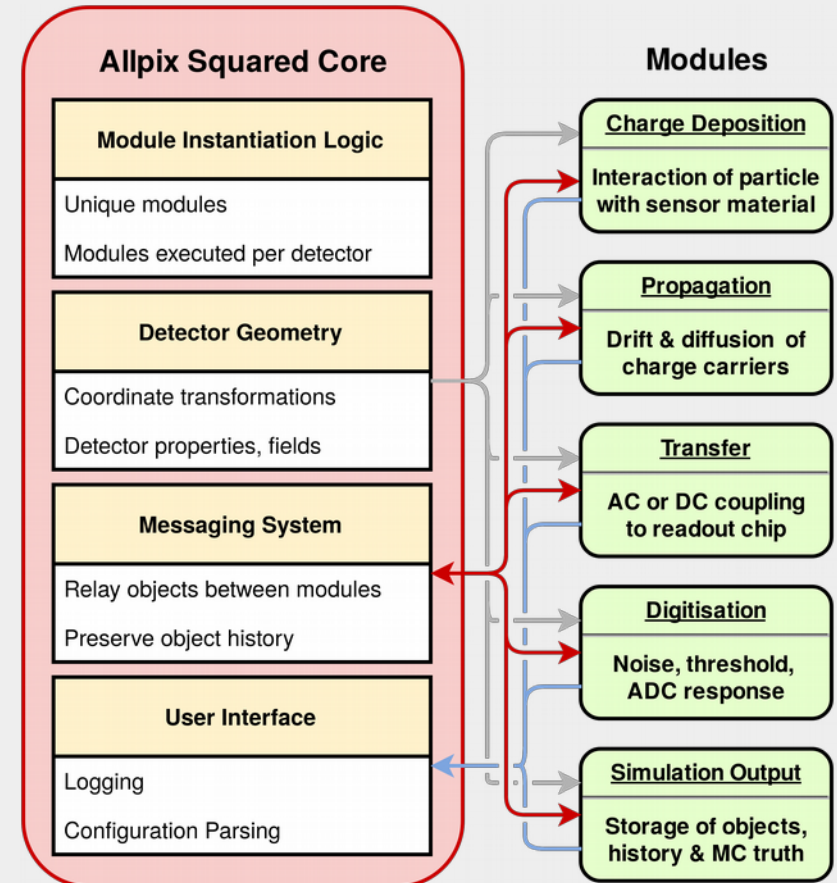
Two-Sentence-Summary

“Allpix² is a generic, open-source software framework for the simulation of silicon pixel detectors. Its goal is to ease the implementation of detailed simulations for both single detectors and more complex setups such as beam telescopes from incident radiation to the digitised detector response.”

Nucl. Instr. Meth. A 901 (2018) 164 – 172
[doi:10.1016/j.nima.2018.06.020](https://doi.org/10.1016/j.nima.2018.06.020)

Basic Design Principle

- Separate infrastructure (core) from the physics (modules)
- Make life easy for the user
 - Allow plugging together the simulation chain from individual modules
 - Auto-generate Geant4 models
 - Convenient configuration, with units
 - Provide Monte Carlo truth information
- Implement physics in independent modules
 - Plug & play concept
 - Offer a selection of different algorithms



Framework Configuration

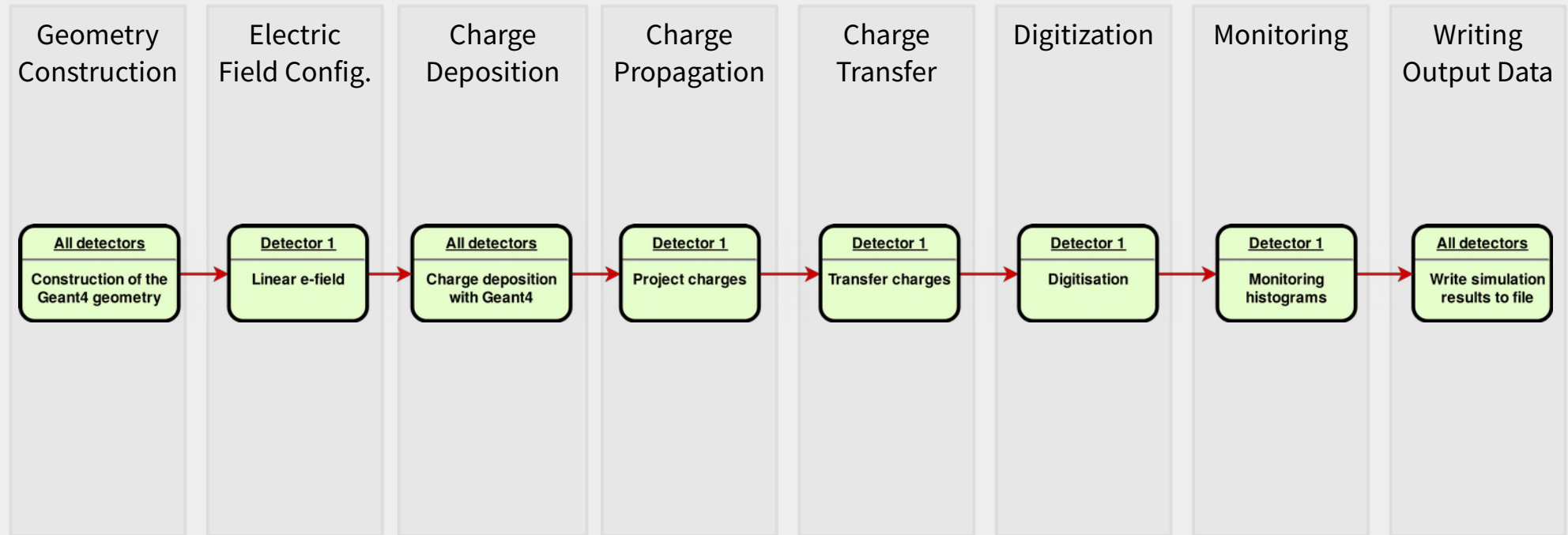
- Framework configured by one file
 - All desired modules listed in order of execution
 - Key-value pairs in **TOML-style** (extended)
 - Human readable
 - Little overhead (e.g. compared to XML)
- Support for physical units
 - Never ask what units are used – type them out!
 - Automatic conversion to internal units
 - No need for manual conversions in C++

```
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]
```



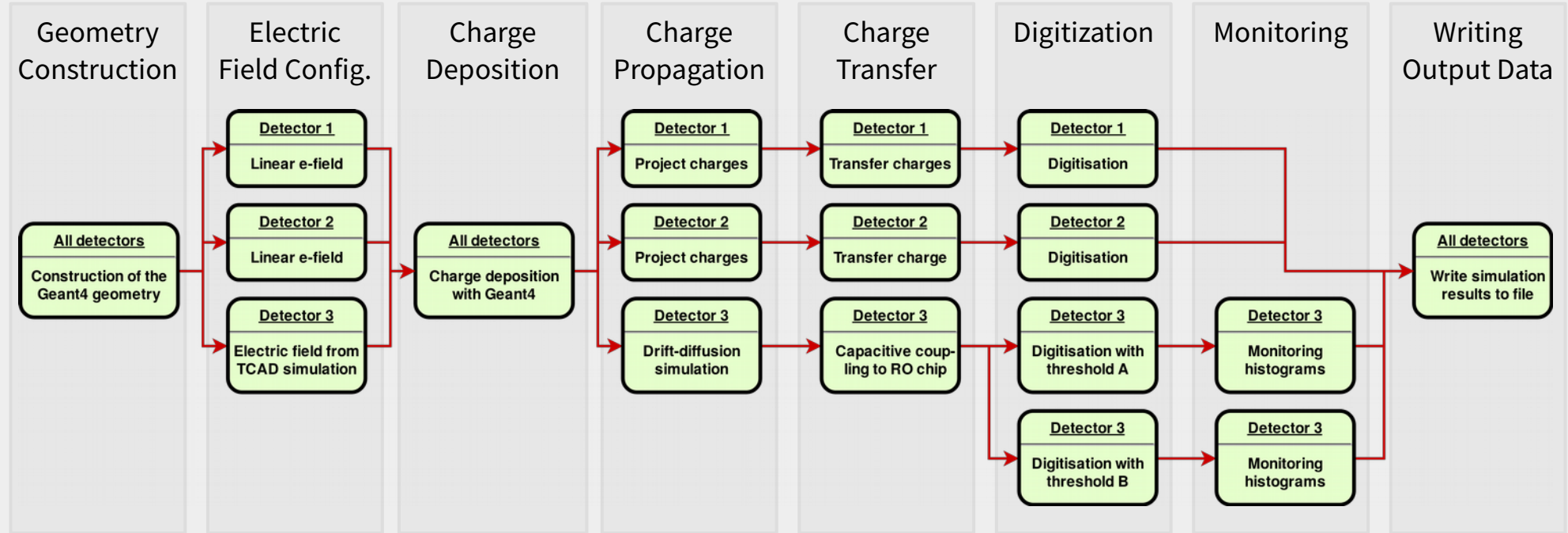
Plugging together the Simulation Chain

- Allows to quickly plug together simple simulations...
(start from examples shipped in the repository)



Plugging together the Simulation Chain

- ...as well as more involved simulations



Messages & Object History

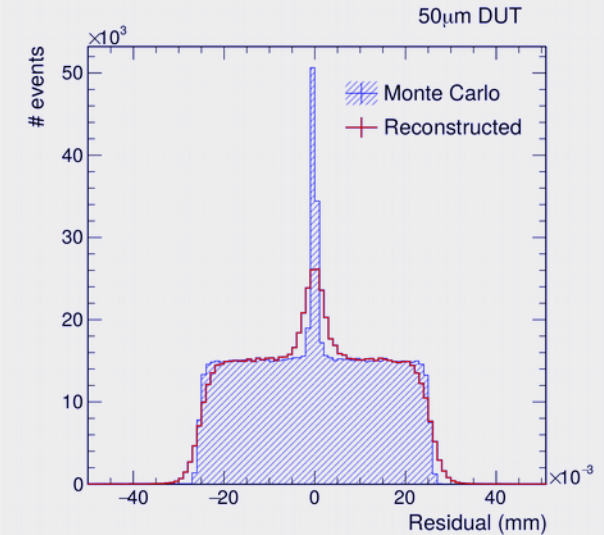
- Data exchanged between modules by means of messages
 - Message holds specific data type and has an origin (detector name, stream name...)
 - Modules bind to specific message (or message type) via central messenger

- Object history

- For each object the full provenance is recorded using TRef objects

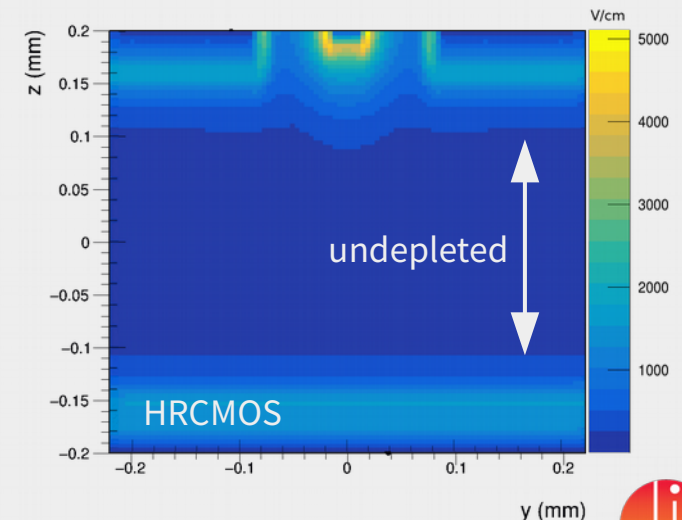
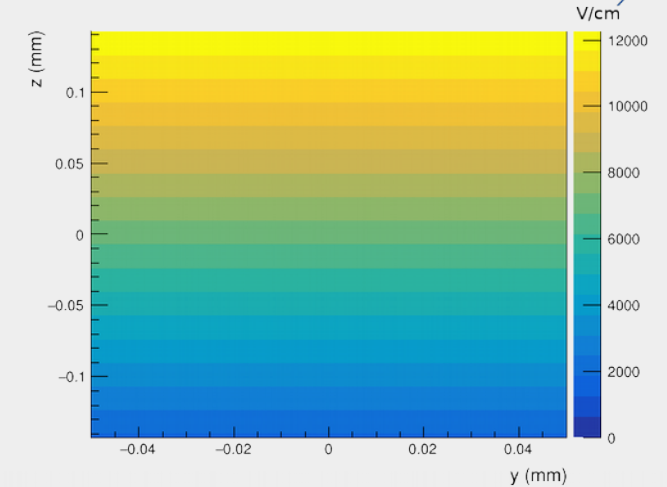
$\text{PixelHit} \leftarrow \text{PixelCharge} \leftarrow \text{PropagatedCharge(s)} \leftarrow \text{DepositedCharge(s)} \leftarrow \text{MCParticle(s)} \leftarrow \text{MCTrack}$

- Allows direct relation of each pixel hit from front-end to initial particle(s)
- Relation between MCParticles enables sorting between primaries (entered sensor from outside) and secondaries (produced within sensor)
- Answers questions like “where in the detector did the charge carriers of this hit originate from?”



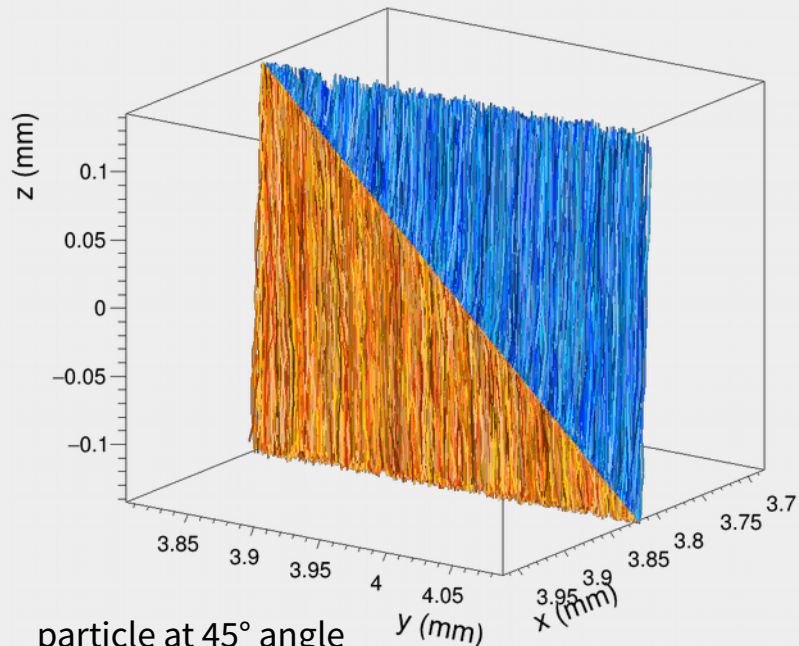
Adjusting the Levels of Detail

- Modular design allows to choose algorithms. Examples:
 - Energy deposition:
 - Geant4 particle tracking
 - Point charge at defined position
 - Electric field:
 - built-in linear approximation
 - loading finite-element field maps (TCAD)
 - Charge transport:
 - calculate total drift time, project onto surface
 - RKF integration, stepping through field
 - Shockley-Ramo calculation: full current pulse (validation ongoing, to be merged)
- Can be adjusted for each detector individually

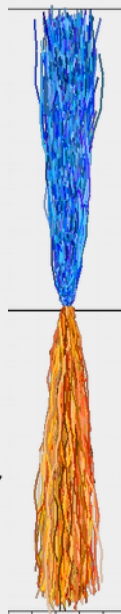


Visualizations of Charge Transport

- Propagation modules can produce line graphs of charge carriers
- Helpful to understand detector behavior

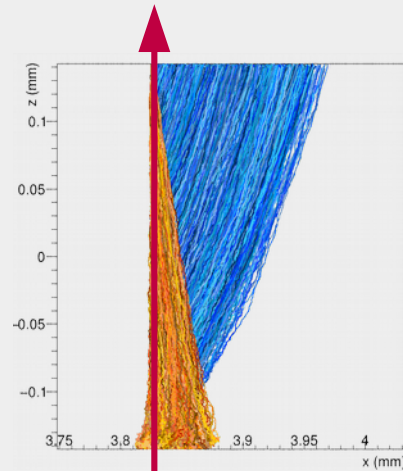


particle at 45° angle
drift paths of electrons & holes

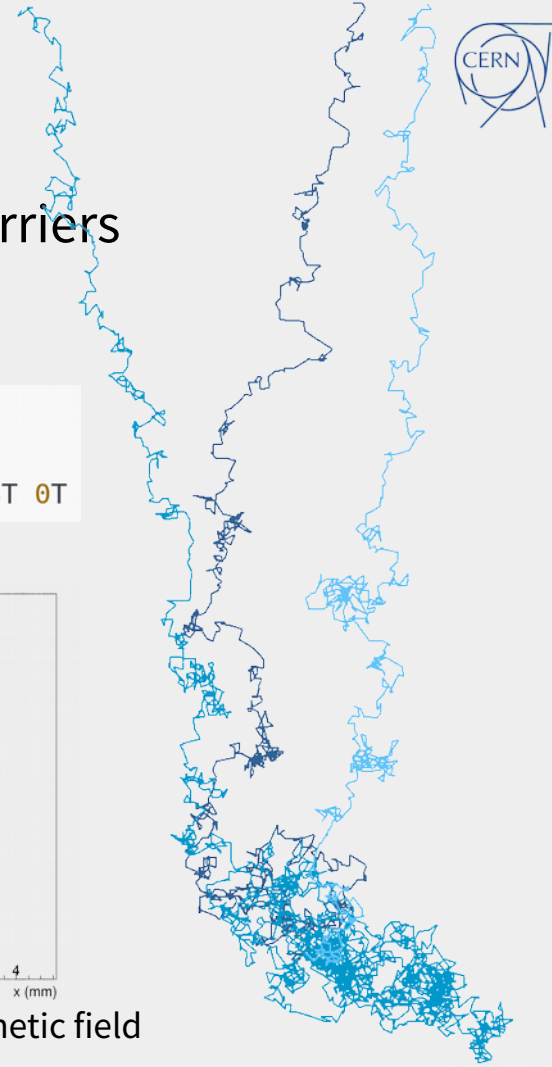


projection along trajectory

```
[MagneticFieldReader]  
model = "const"  
magnetic_field = 0mT 3.8T 0T
```

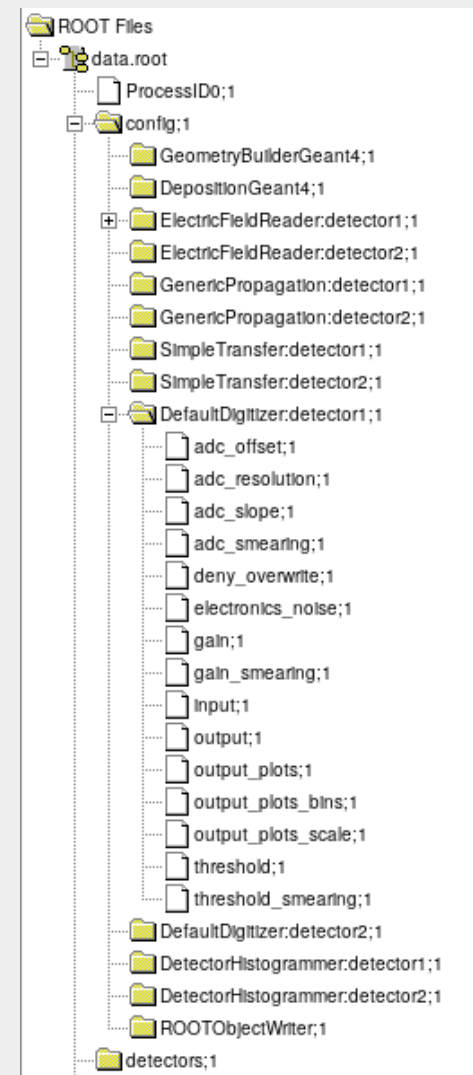


Lorentz drift in magnetic field



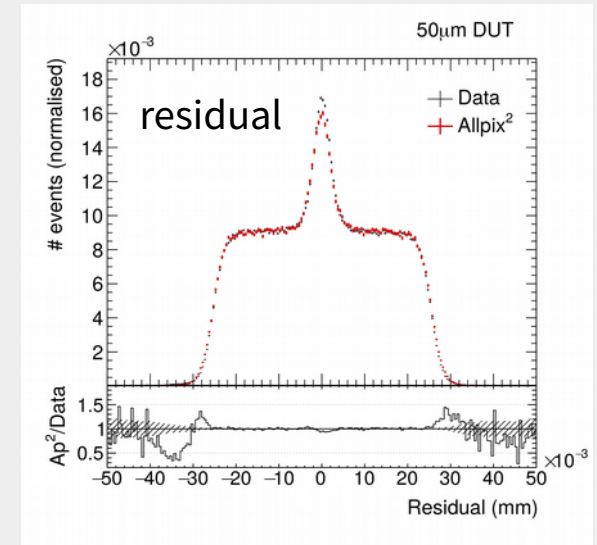
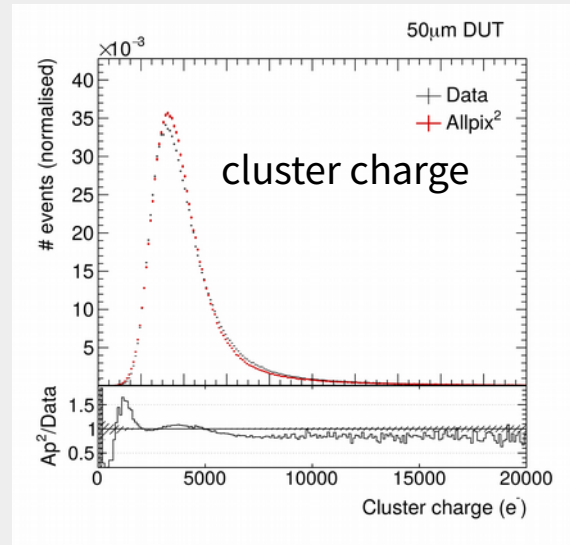
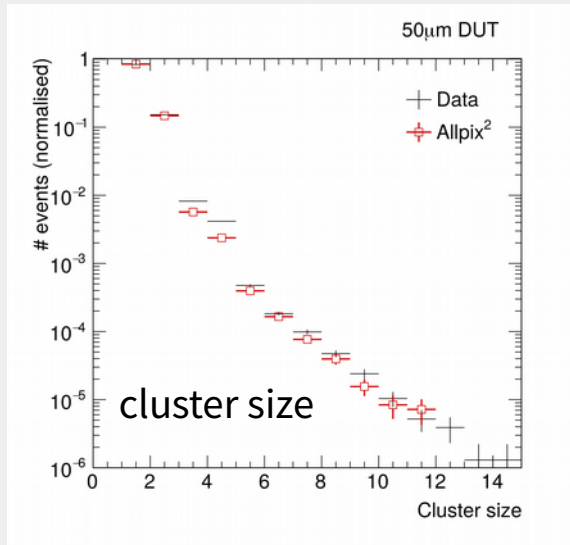
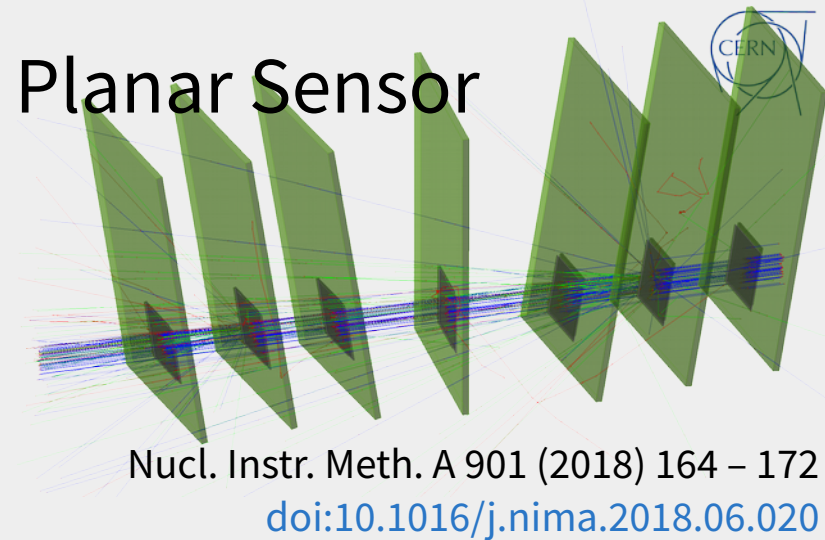
I/O

- Different output writers available
- 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



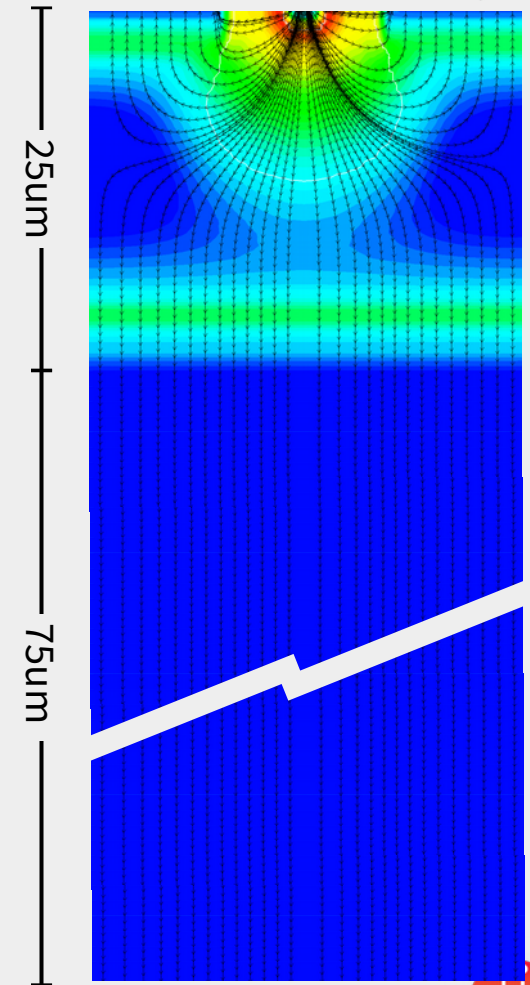
Simulation of a Timepix3 with 50 μ m Planar Sensor

- Full telescope: 6 planes Timepix3 + DUT
- Linear electric fields
- Full reco: clustering, eta correction, tracking
- Matches very well with data



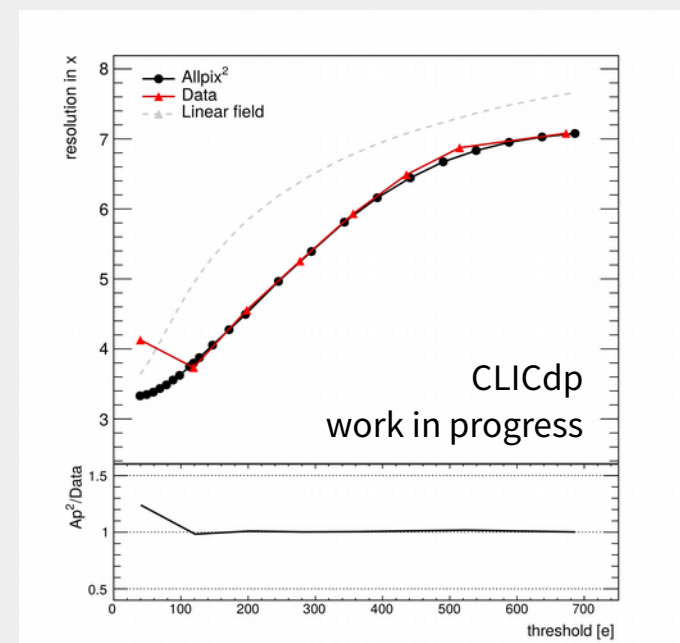
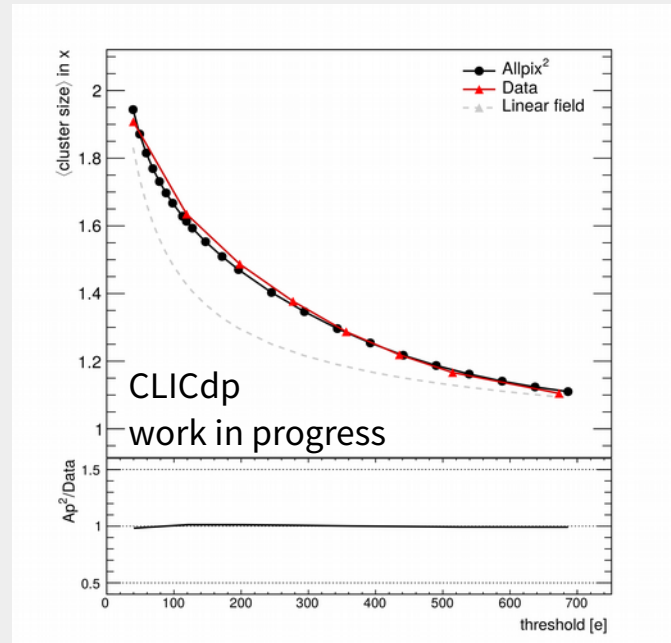
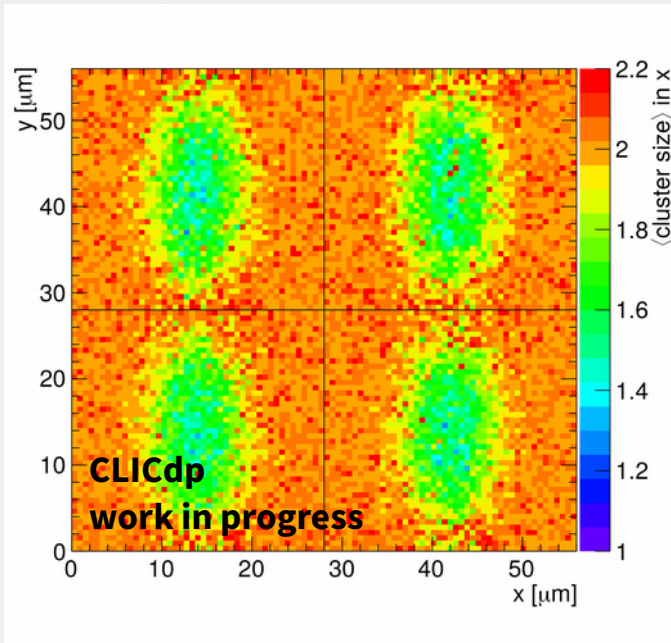
Monolithic CMOS in High-Resistivity Silicon

- Simulation of ALICE Investigator-like chip, 28x28um pitch
 - Field in top 25um (high-resistivity) silicon
 - Undepleted in 75um silicon substrate
- SPS beam: 120 GeV Pions, only one detector simulated
 - Using Monte Carlo truth information as reference
 - Smeared with telescope resolution obtained from data
- Import electric field from TCAD simulations
- Using Geant4's photoabsorption ionization model (PAI) for thin sensors
- Challenges: life time / recombination, influence of charge cloud on field...
 - Trade-off between accuracy and necessary simplifications



Monolithic CMOS in High-Resistivity Silicon

- Manage to reproduce x-y-correlation features in cluster size
- Data and simulation matches very well: **cluster size & resolution vs. threshold**
- Comparison: linear field simulation does not describe data



Development of Allpix Squared

- Meant as community-driven project
 - Still most contributions from “core team”
 - Increasing number of external contributions
- Fully GitLab-centered development
 - Issue tracking, merge requests, continuous integration
- All development is performed “in the open”
 - Full repository public, including issues
 - Subscribers receive information on all actions
- Semantic versioning:
Major.Minor.Patch = Framework.Features.Bugfixes

v1.3.2	2019-02-21
v1.3.1	2018-12-17
v1.3	2018-11-21
v1.2.3	2018-11-13
v1.2.2	2018-09-07
v1.2.1	2018-08-02
v1.2	2018-06-13
v1.1.2	2018-04-25
v1.1.1	2018-03-08
v1.1	2018-01-11
v1.0	2017-08-29

Warnings, Strict Formatting & Code Linting

- In order to maintain a high code quality, we
 - Enabled many compiler warnings, with **-Werror**
 - Require strict adherence to code formatting (indentation, brace positions...)
 - Clang-tidy is used to spot e.g. missing `&`, `std::move()` or NamingConventionViolations
- A bit painful for newcomers
 - CI fails many times, users need to be taught how to read failed job logs
- Particularly useful: `/etc/git-hooks`
 - **pre-commit-clang-format-hook** – checks formatting before committing
 - **pre-push-tag-version-hook** – check for pre-release things (update version)
- CMake suggests installing hooks (suggests update on changes)

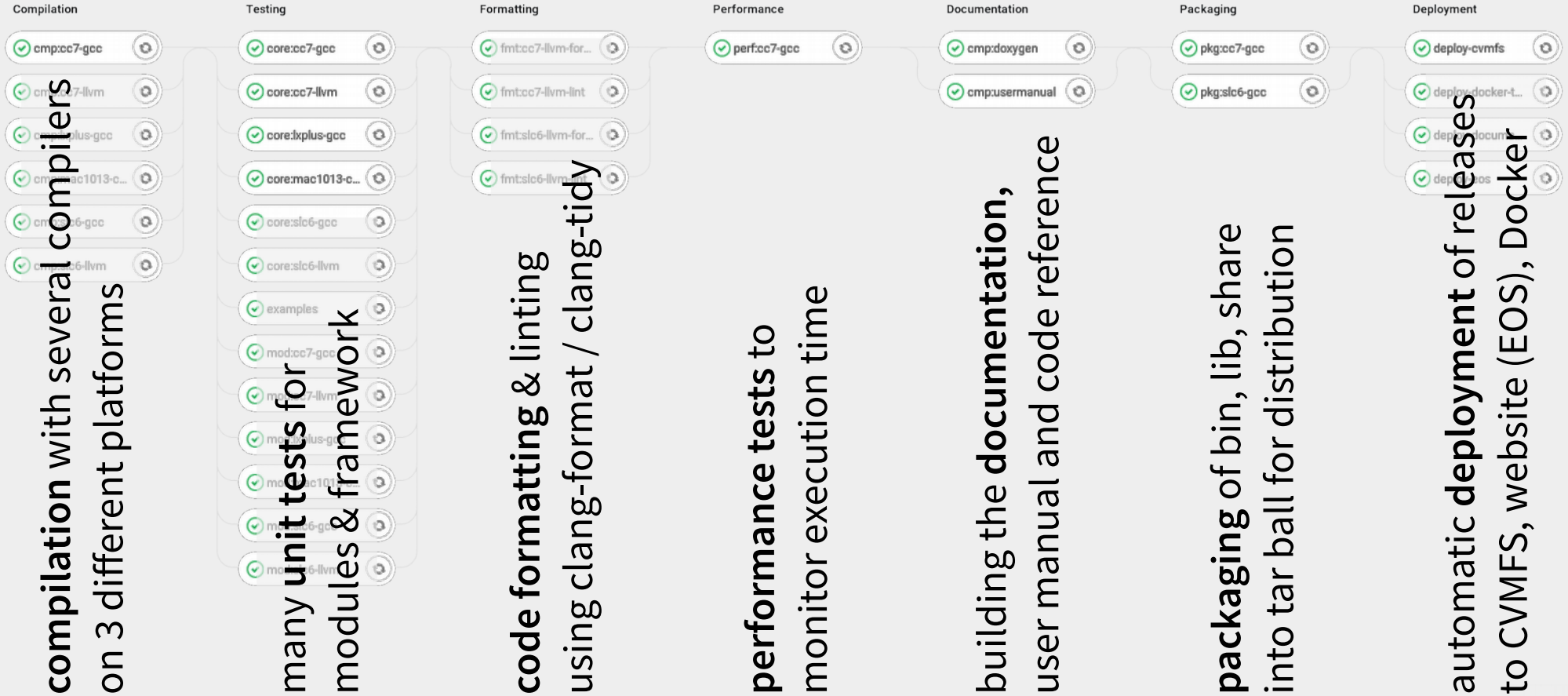
Code Review via Merge Requests

- No new code lands in master without review by another party
 - Using GitLab's approval feature
 - Extensive discussions about code, but also style, naming schemes
- Proven to be **very effective**
 - Several bugs found before the merge
 - New users appreciate guidance
- Proven to be **labor-intensive**
 - Read (and understanding) every change
 - Always be supportive, positive

...just some of them

Python macro to read output objects TTree !191 · opened 2 months ago by Sebastien Murphy	MERGED ✓ 18 updated 1 week ago
Revamp MeshConverter: Change interpolation & improve performance !200 · opened 3 weeks ago by Simon Spannagel	MERGED ✓ 5 updated 1 week ago
Write full Proteus configuration in RCEWriter !203 · opened 2 weeks ago by Moritz Kiehn	MERGED ✓ 12 updated 1 week ago
Invert Detector Rotations !164 · opened 6 months ago by Simon Spannagel documentation detector models bug	MERGED ✓ 8 updated 2 weeks ago
RCEWriter: fix Proteus geometry output !202 · opened 2 weeks ago by Moritz Kiehn	MERGED ✓ 3 updated 2 weeks ago
FieldParser: be more careful about units !201 · opened 3 weeks ago by Simon Spannagel	MERGED ✓ 1 updated 2 weeks ago
Add option for a depletion from the backplane !198 · opened 3 weeks ago by Paul Schutze physics improvement	MERGED ✓ 11 updated 3 weeks ago
New Field File Format APF & common FieldParser/FieldWriter !197 · opened 1 month ago by Simon Spannagel	MERGED ✓ 11 updated 3 weeks ago
New Module: DepositionPointCharge !194 · opened 1 month ago by Simon Spannagel	MERGED ✓ 12 updated 1 month ago

Continuous Integration & Testing



Automated Testing

```
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 v #PASS (DEBUG) misaligned: (8.72466deg,171.099deg,178.504deg)
11 #PASSOSX (DEBUG) misaligned: (9.04007deg,170.886deg,178.731deg)
```

```
Test project /builds/allpix-squared/allpix-squared/build
  Start 53: test_core/test_01-1_globalconfig_detectors.conf
  Start 54: test_core/test_01-2_globalconfig_modelpaths.conf
  Start 55: test_core/test_01-3_globalconfig_log_format.conf
  Start 56: test_core/test_01-4_globalconfig_log_level.conf
1/22 Test #53: test_core/test_01-1_globalconfig_detectors.conf ..... Passed    0.81 sec
  Start 57: test_core/test_01-5_globalconfig_log_file.conf
2/22 Test #56: test_core/test_01-4_globalconfig_log_level.conf ..... Passed    2.11 sec
3/22 Test #55: test_core/test_01-3_globalconfig_log_format.conf ..... Passed    2.11 sec
  Start 58: test_core/test_01-6_globalconfig_missing_model.conf
  Start 59: test_core/test_01-7_globalconfig_random_seed.conf
4/22 Test #57: test_core/test_01-5_globalconfig_log_file.conf ..... Passed    1.11 sec
```

[...]

100% tests passed, 0 tests failed out of 22

- No real “unit testing” up till now, more of a “system test”
- Framework & module tests
- Prepare a plethora of configuration files
 - Run single event with fixed seed
 - Reproduces same output
 - Matching regular expressions
- Single change (1e difference) fails CI
- Invaluable for monitoring framework
→ catching issues before merging code

User Manual & Code Documentation

- Focus from the very beginning on well-documented framework
- Source code documentation for every class, method
 - Doxygen markup for code reference
 - Deployed to the website for tags
- Extensive User Manual in LaTeX
 - Automatically compiled by CI
 - Module documentation as Markdown
 - Document module parameters, algorithms
 - Included in manual via Pandoc

```
namespace allpix {  
  
  /**  
   * @brief Instantiation of a detector mode  
   *  
   * Contains the detector in the world with  
   * (like the electric field). All model sp  
   * properties are stored in its DetectorMo  
   */  
  class Detector {  
  friend class GeometryManager;  
  
  public:  
    /**  
     * @brief Constructs a detector in the  
     * @param name Unique name of the dete  
     * @param model Model of the detector  
     * @param position Position in the wor  
     * @param orientation Rotation matrix  
     */  
    Detector(std::string name,  
             std::shared_ptr<DetectorModel>  
             ROOT::Math::XYZPoint position  
             const ROOT::Math::Rotation3D&  
  
    /**  
     * @brief Get name of the detector  
     * @return Detector name  
     */  
    std::string getName() const;
```

GenericPropagation

Maintainer: Koen Vanders (k.vanders@cern.ch), Simon Spannagel (s.spannagel@cern.ch)

Status: Functional
Input: DepositedCharge
Output: PropagatedCharge

Description

Simulates the propagation of electrons and 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 precision. The propagation process for these sets is fully independent and no interactions are 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 per environmental condition. Acceleration at electrodes. The carrier mobility for other electrons or holes is automatically chosen based on the type of the charge carrier under consideration. Thus, also type 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. Other 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 inside should only be used where sensible. The direction of the propagation depends on the electric field configuration, and should be ensured that the carrier types selected are actually transported to the intended side for slow 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:

$$r = \sqrt{\frac{2D \Delta t}{\mu}}$$

using the carrier mobility μ , the temperature T and the time step. The propagation stops when the set of charges reaches any of the electrodes.

The propagation module also produces a variety of output plots. These include a 2D line plot of the path of all separate propagated charge carrier sets from their point of deposition to the end of their drift, with nearby paths having different colors. In the coloring scheme, electrons are marked in blue colors, while holes are presented in different shades of orange. In addition, a 2D GC simulation 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 debugging drift behavior.

Dependencies

This module requires an installation of Digesit.

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. Controls the total number of deposited charge carriers at a specific point in time and the number of charge carriers used, with the remaining charge carriers. A value of 10 charges per steps usually default if this value is not specified.
- `spatial_precision`: Spatial precision to use for. The timestep of the Runge-Kutta propagation is adjusted to reach this spatial precision after calculating the necessary from the fifth order error method. Defaults to 0.1 cm.
- `time_step_start`: Timestep to initiate the Runge-Kutta integration with. A progressive estimation of the parameter reduces the time to start the timestep to the spatial precision parameter. Default value 0.01 ns.
- `time_step_min`: Minimum step time to use for the Runge-Kutta integration regardless of the spatial precision. Defaults to 1 ns.
- `time_step_max`: Maximum step time to use for the Runge-Kutta integration regardless of the spatial precision. Defaults to 0.1 ns.
- `propagation_size`: Time with 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 false.
- `propagate_holes`: Select whether hole-type charge carriers should be propagated to the electrodes. Defaults to false.
- `output_plots`: Determine 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. Default is false.
- `output_plots_line`: Timestep to use for the line plots. Indirectly determines the amount of points plotted. Defaults to inverse value if not explicitly specified.
- `output_plots_theta`: Viewport angle of the 3D animation and the 2D line graph around the world Z-axis. Defaults to zero.
- `output_plots_phi`: Viewport angle of the 3D animation and the 2D line graph around the world X-axis. Defaults to zero.
- `output_plots_use_pixel_size`: Determine if the plots should use pixels as unit instead of metric length scales. Defaults to false (this using the metric system).
- `output_plots_use_equal_scaling`: Determine if the plots should be produced with equal distance scales on every axis (which implies that some points will fall out of the graph). Defaults to true.
- `output_plots_align_plots`: Determine 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 right place between pixels.
- `output_animation_line`: In addition to the other output plots, also write a GC 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_animation_line_scaling`: Scaling for the animation used to convert the actual animation time to the time step the animation. Defaults to 1.0, meaning that every conversion of the animation requires an animation of a single second.
- `output_animation_marker_size`: Scaling for the markers on the animation. Defaults to one. The markers are already internally scaled to the charge of the step normalized to the maximum charge.
- `output_animation_color_equal_scaling`: Scaling to use for the color scale axis 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 (independent on the value above this, are displayed the same maximum color). Parameter can be used to improve the color scale of the output plots.
- `output_animation_color_markers`: Determine if colors should be for the markers on the animations. Defaults to false.

Usage

An example of generic propagation for all sensors of type "Trapez" at room temperature using packets of 25 charges in the following:

```
{GenericPropagation  
  type = "Trapez"  
  temperature = 293K  
  charge_per_step = 25
```

Automated Deployment of Tagged Versions

- Using CERN GitLab CI Tools
<https://gitlab.cern.ch/ci-tools>
- Deployment to website
 - Binary tarball packages placed on EOS
 - Using “ci-web-deployer” image
- Deployment as Docker image
 - Using “docker-image-builder”
 - DOCKER_FILE specified as variable
 - Build on top of Docker image w/ dependencies (Geant4, ROOT6)

```
variables:  
  EOS_PATH: "/eos/project/a/allpix-squared/www/"  
  DOCKER_FILE: etc/docker/Dockerfile
```

```
deploy-eos:  
  stage: deployment  
  tags:  
    - docker  
  variables:  
    GIT_STRATEGY: none  
  # Only run for new tags:  
  only:  
    - tags  
    - schedules # Only execute this on scheduled "nightly" pipelines  
  dependencies:  
    - pkg:cc7-gcc  
    - pkg:slc6-gcc  
  # Docker image with tools to deploy to EOS  
  image: gitlab-registry.cern.ch/ci-tools/ci-web-deployer:latest  
  script:  
    - deploy-eos
```

```
deploy-docker-tag:  
  stage: deployment  
  tags:  
    - docker-image-build  
  dependencies: []  
  only:  
    - tags  
  script:  
    - "echo" # unused but this line is required by GitLab CI  
  variables:  
    T0: gitlab-registry.cern.ch/allpix-squared/allpix-squared:${CI_COMMIT_TAG}
```

Automated Deployment: CVMFS

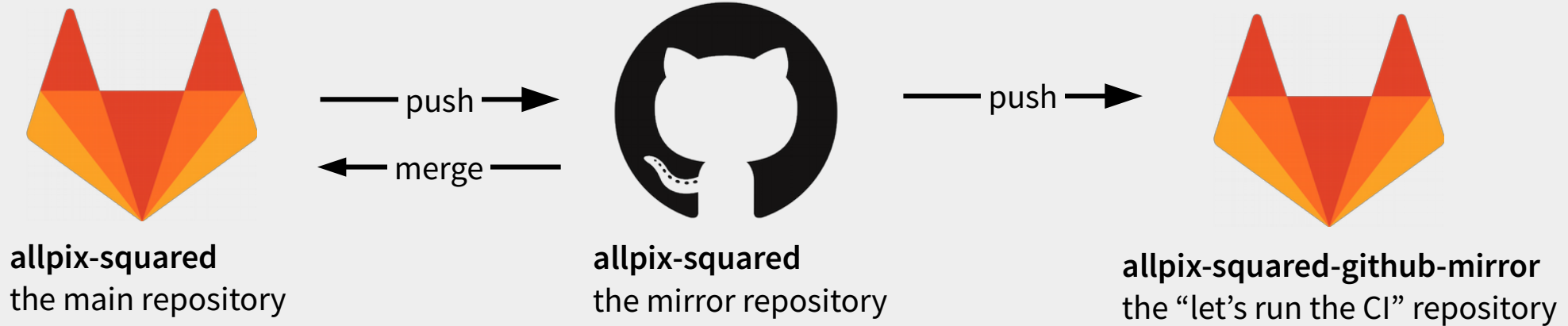
- A bit trickier... (see [talk by Marco Petric](#))
 - Runner (selected via “cvmfs-deploy” tag) opens shell on CVMFS submission server
 - No direct access to artifacts → use script to download & deploy
- Deployed version ready for use on LXPlus or HTCondor submission

```
deploy-cvmfs:
  stage: deployment
  dependencies:
    - pkg:cc7-gcc
    - pkg:slc6-gcc
  tags:
    - cvmfs-deploy
  only:
    - tags
    - schedules # Only execute this on scheduled "nightly" pipelines
  script:
    - ./gitlab/ci/download_artifacts.py $API_TOKEN $CI_PROJECT_ID $CI_PIPELINE_ID
    - export RUNNER_LOCATION=$(pwd)
    - if [ -z ${CI_COMMIT_TAG} ]; then export BUILD_PATH='latest'; else export BUILD_PATH=${CI_COMMIT_TAG}; fi
    - sudo -u cvclidp -i $RUNNER_LOCATION/.gitlab/ci/gitlab_deploy.sh $RUNNER_LOCATION $BUILD_PATH
    - rm -f allpix-squared-*.tar.gz
  retry: 1
```

```
$ source /cvmfs/clicdp.cern.ch/software/allpix-squared/1.3.2/x86_64-centos7-gcc7-opt/setup.sh
$ allpix --version
Allpix Squared version v1.3.2
built on 2019-02-21, 19:43:54 UTC
```

External Contributions via GitHub

- We rely on CERN's GitLab CI (and our own runners)
 - Allows us to run extensive CI and adjust to our needs
 - Requires full CERN account for write access
- Our solution: ping-pong mirror to GitHub

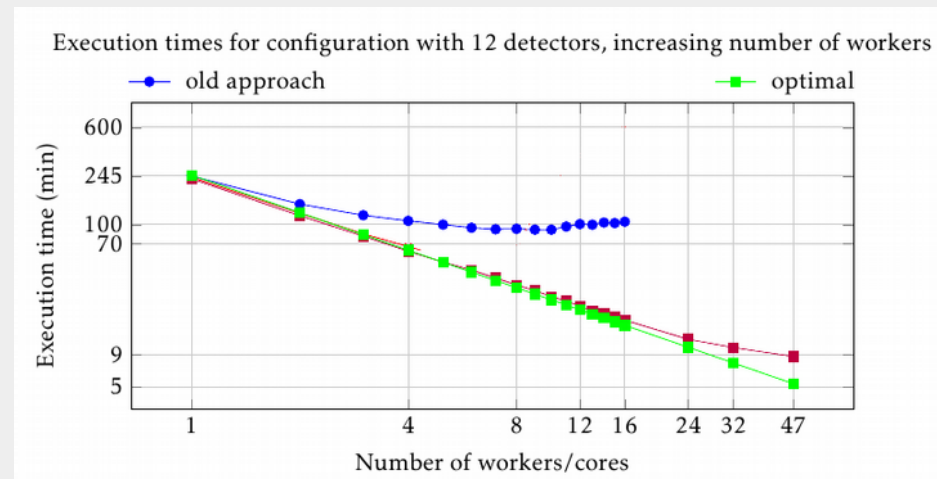




Google Summer of Code



- Participated for the first time in GSoC last year through HSF/CERN
 - Proposed project: Event-based Multi-Threading for Allpix Squared
 - Quite intricate due to seeding of PRNGs & potential race conditions
- Student was very active and motivated
 - Restructured parts of core framework, implemented first working version
 - Some road blocks encountered (interfaces with dependencies)
- Interesting observation:
 - Student was expecting direct instructions, we expected scientific collaboration → worked out well after discussing!



Summary

- Allpix Squared is a framework for MC simulation of silicon pixel detectors
 - Flexible framework combining Geant4, electrostatic TCAD simulations and charge transport algorithms
 - Continuously developed and extended
 - Modular design, strict coding rules, collaborative workflow
- Continuous integration one of the backbones
 - Automated compilation and testing, ensuring compliance with rules
 - Deployment of compiled versions to CVMFS, Docker, website
- Participation in GSoC 2018 very useful

- Will participate again in GSoC 2019, have very promising candidate already
- Interested in **Google Season of Docs** to further improve documentation, separate coding/physics
- Many ideas for extensions
(transient current, high-Z materials, multi-threading, charge multiplication, lifetime)

Resources



Website

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



Repository

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



Docker Images

https://gitlab.cern.ch/allpix-squared/allpix-squared/container_registry



User Forum:

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



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>



Users & Contributors

Disclaimer: these are just some users we have been in contact with – there probably are some more.

ONERA Aerospace Lab, Toulouse		CLICdp @ CERN		ATLAS @ DESY
Georg-August-Universität Göttingen		CMS Pixel @ CERN		CMS Lorentz Angle @ DESY
University of Birmingham		ATLAS Strips @ CERN		ELAD @ DESY
University of California, Berkeley		LHCb VeloPix @ CERN		University of Liverpool
NIKHEF, Amsterdam	University of Glasgow	ATLAS Monolithic @ CERN		ATLAS SCT @ KEK
	Czech Techn. University, Prague			Dortmund University
Rutherford Lab, STFC	ETH Zurich	IHEP Beijing	Freiburg University	Université de Genève
	Université de Montréal	Charles University, Prague	Utrecht University	AGH University Krakau

- First **user workshop** held
26-27 November 2018 @ CERN
- Tutorials, discussions, feedback
- Very successful, to be continued

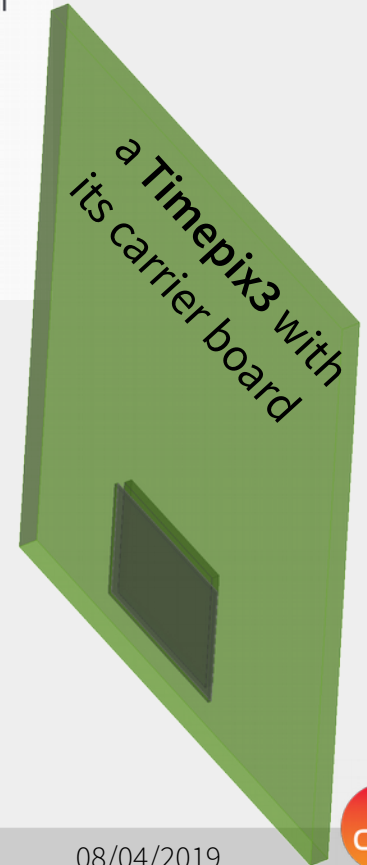


Detector Models

- 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
```

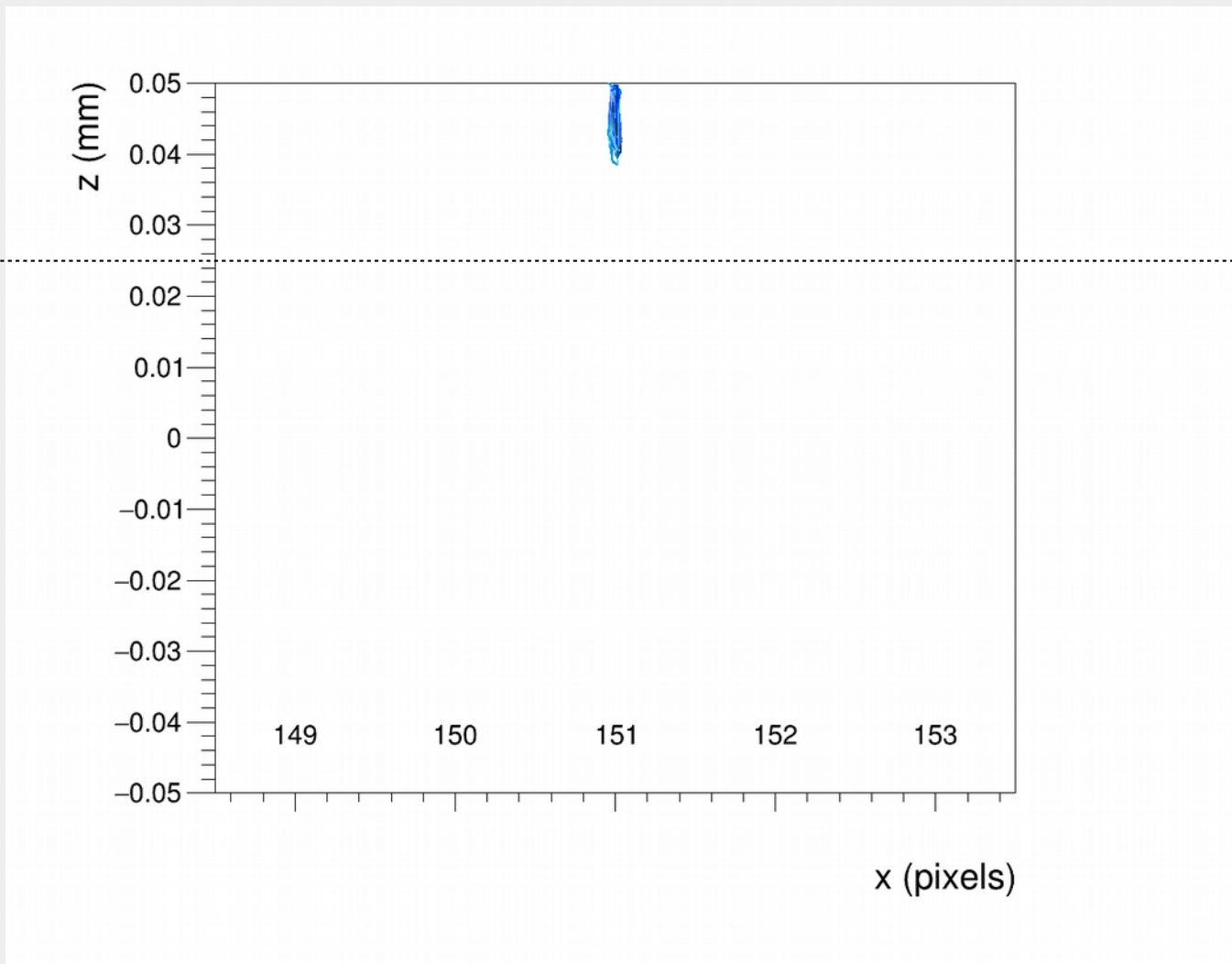


epitaxial
substrate



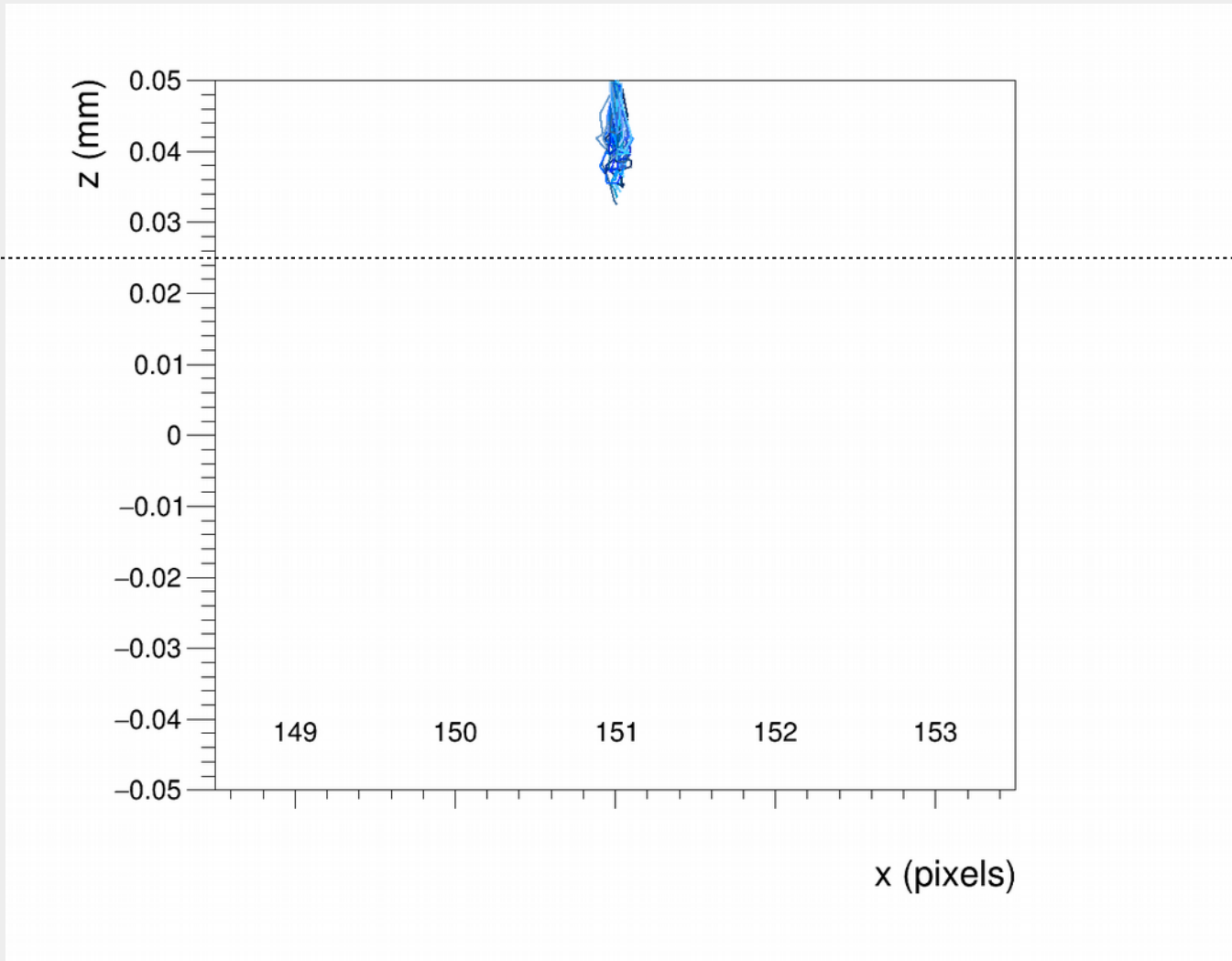
0.5ns

epitaxial
substrate



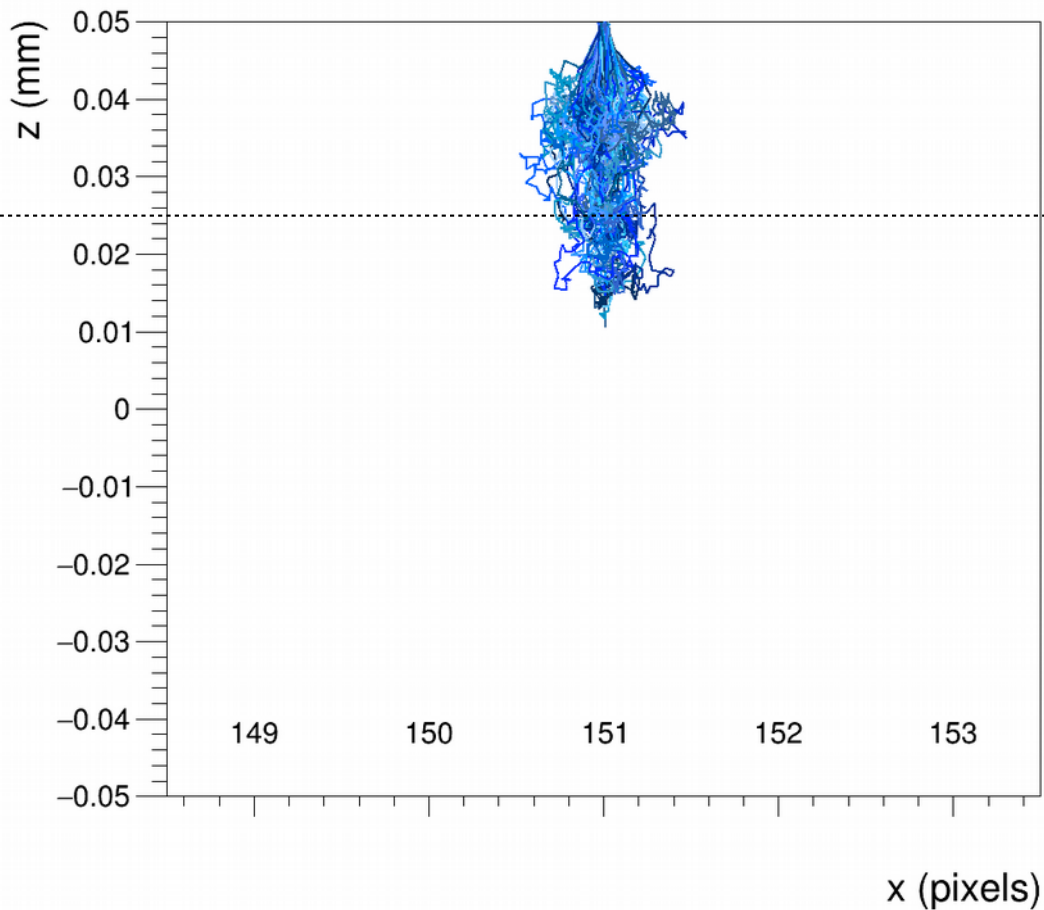
1.5ns

epitaxial
substrate



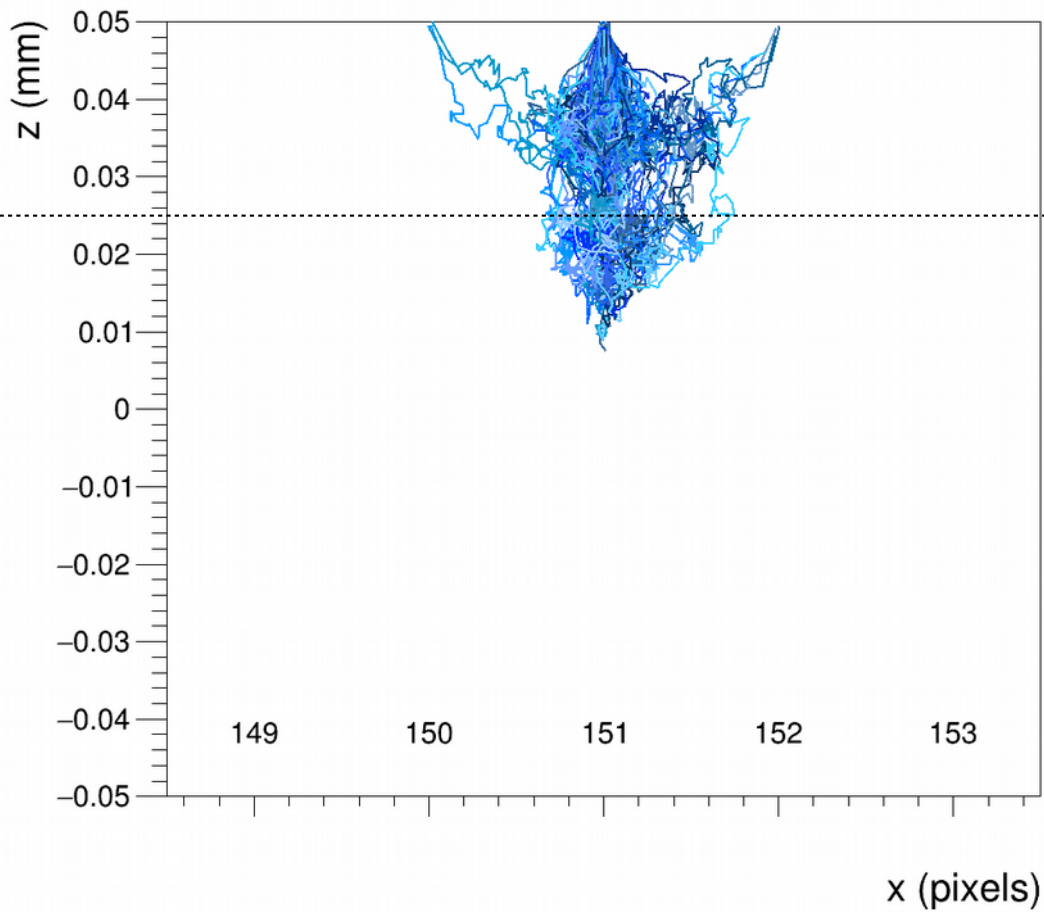
10ns

epitaxial
substrate



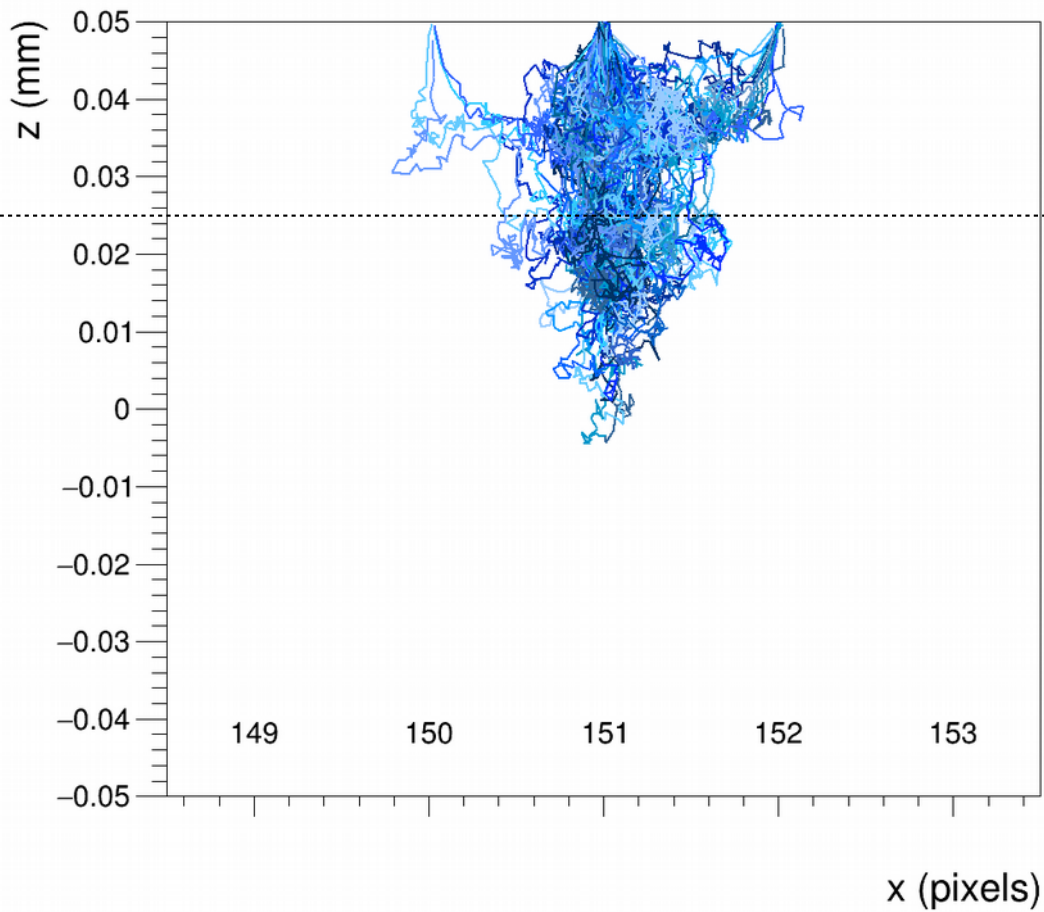
15ns

epitaxial
substrate



20ns

epitaxial
substrate



25ns

epitaxial
substrate

