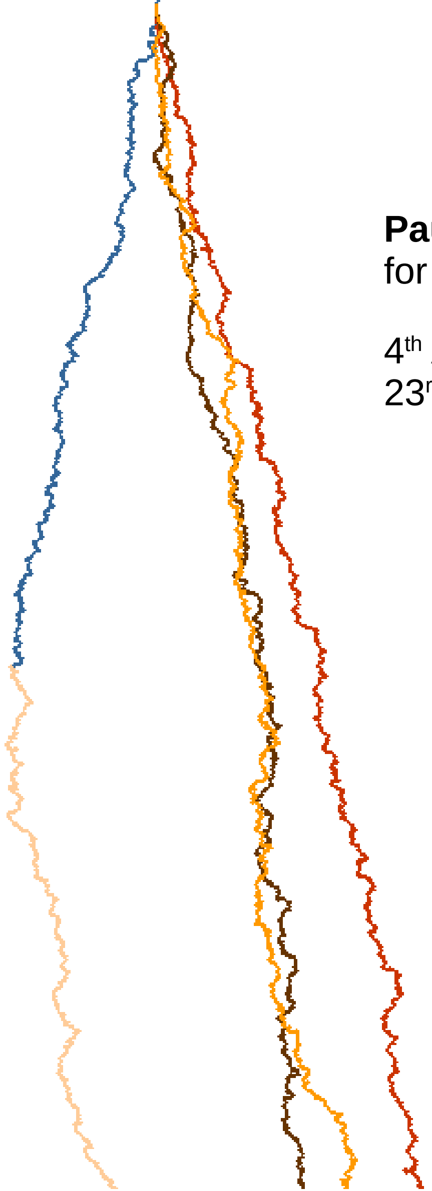
Allpix Squared 3.0



Paul Schütze
for the Allpix Squared Authors

4th Allpix Squared User Workshop
23rd May 2023

Allpix Squared 3.0 Released

Thursday, May 04, 2023

We are more than just happy, one might say even a bit proud, to announce the release of the next major version of the Allpix Squared framework: **Allpix Squared 3.0**. The new release comes with a large number of new features of which you can read below and contains more than 2100 commits over the last feature release, version 2.4.

A direct link to the source code can be found here:

https://gitlab.cern.ch/allpix-squared/allpix-squared/tags/v3.0.0

The new version is also available as docker image in the project's docker registry and as read-to-use version on CVMFS.

We would like to thank everyone who contributes to this project. To date, we count 52 contributors - thank you for your input and your will to share your work with the community! In addition, we express gratitude to all those who contributed via fruitful discussions and thorough testing of new as well as existing features.

We would also like to advertise our **fourth User Workshop**, which will be held at DESY from 22 May 2023 to 23 May 2023. Although the in-person registration deadline has passed, there is still the chance to register for a remote participation and learn about news around the development and applications of Allpix Squared.

In the following, we will briefly present some of the more prominent new features and changes to the framework:

- Website & Documentation
- Detector Geometries
- Impact Ionization
- New Simulation Objects
- Additional Features & Changes
- Development Visualization

Website and Documentation

For Allpix Squared 3.0, the documentation and the entire website has been reworked. A major part of the rework has been the transition from LaTeX as main documentation format to Markdown. Using Markdown means the documentation can be read directly in GitLab/GitHub and in many IDEs. Using Markdown also fixes many of the display issues that existed on the old online documentation, so we suggest you to give it a try! Besides the improved online documentation, the PDF manual also continues to exist.

The new website is now hosted on allpix-squared.docs.cern.ch. Some of the new features are responsive design and a site-wide search function. The new website now also hosts the API reference in the same design. But again, we recommend to just try it out.

Detector Geometries

In Allpix Squared 3.0, the detector geometry subsystem has seen a major overhaul, and now features multiple different sensor geometries and more flexibility in combining sensor and readout ASIC.

Detector Assemblies

From Allpix 3.0 onwards, the sensor geometry and the sensor assembly type can be specified separately in the detector model file, e.g.

```
type = "hybrid"
geometry = "hexagon"
```

to combine a hexagonal pixel sensor with a separate readout ASIC connected via bump bonds, or

```
type = "monolithic"
geometry = "pixel"
```

for a monolithic active pixel sensor with rectangular pixels.

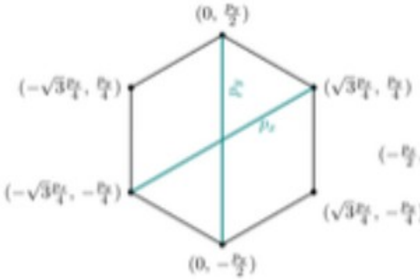
Hexagonal Pixel Detectors

One of the newly implemented sensor geometries are hexagonal pixel grids as used in many X-ray detectors or photon science applications. This implementation has been around for quite some time already and has been tested by multiple projects.

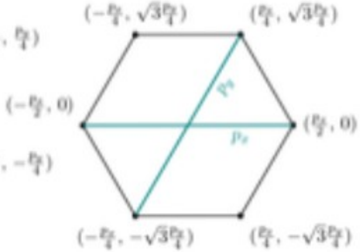
Allpix Squared implements hexagons using **axial coordinates**, since they

- make many calculations quite simple and fast since they derive from a projection of cubic Cartesian coordinates
- require only two coordinates and therefore fit well into the framework and the indices in Cartesian coordinates already available.
- minimize required memory footprint by storing only two and not the third (redundant) cubic coordinate. The latter can always be reconstructed from the first two as $z = -x - y$.

These two orientations of hexagons are referred to as **pointy** (with sides parallel to the y axis of the Cartesian coordinate system and corners at the top and bottom) and **flat** (with sides parallel to the Cartesian x axis and corners to the left and right) in Allpix Squared. The pitches p_x and p_y have been chosen as the diameter of the hexagon at two adjacent corners. They align with the axial coordinate system and are oriented differently with respect to the Cartesian system for the two variants, respectively.

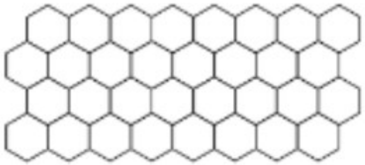


(a) Hexagon in *pointy* orientation



(b) Hexagon in *flat* orientation

These hexagons can be assembled to grids in two different orientations:



(a) 8x4 grid with pointy hexagons

(b) 8x4 grid with *flat* hexagons

It should be noted that with two pitches defined, irregular (or stretched) hexagons can be used as well, simply by providing different sizes for pitch p_x and p_y.

As always, a more detailed description can be found in the user manual.

Barsky line clipping algorithm is implemented to provide fast and efficient intersection calculations. For ellipsoid implants, intersections with the cylinder and its caps are considered.

More details on sensor implants are provided in the user manual.

Radial strips

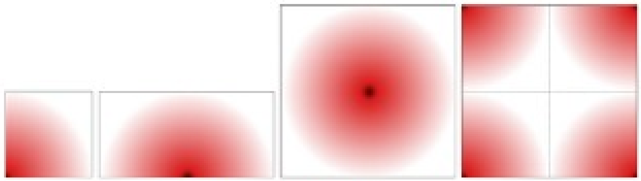
Both in the past and the future, particle physics experiments implement radial silicon strip detectors, and a new detector model caters to these needs. The implementation includes many parameters such as the possibility to define stereo angles.

This new detector model is described in the user manual and in addition, there are examples demonstrating the use of the model with a single sensor as well as a full petal of the ATLAS ITk Upgrade.

Detector Fields

With new possibilities of defining sensor geometries, also new ways of mapping electric fields from finite element simulations into the sensor plane have been devised. Since this has drastically extended the possibilities of mapping pixel fields, a new section has been added to the user manual, describing the different options.

The following types of fields can be loaded and mapped to individual pixels with Allpix Squared 3.0:



Examples for pixel geometries in field maps. The dark spot represents the pixel center, the red extent the electric field. Pixel boundaries are indicated with a dotted line where applicable.

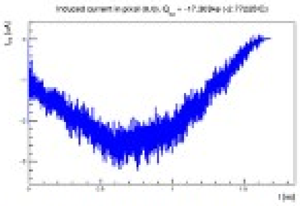
Fields are always expected to be provided as rectangular maps, irrespective of the actual pixel shape. Maps are loaded once and assigned on a per-pixel basis. Depending on the symmetries of the pixel unit cell and the pixel grid, different geometries are supported as indicated in the figure above. The field for a quarter of the pixel plane, for half planes (see figures below) as well as for full planes (see figure above). The size of the field is not limited to a single pixel cell, however, for some quantities such as the electric field only the volume within the pixel cell is used and periodic boundary conditions are assumed and expected. Larger fields are for example useful for the weighting potential, where also potential differences to neighboring pixels are of interest.

A special case is the field presented in the right panel of the figure above. Here, the field is not centered at the pixel unit cell center, but at the corner of four adjacent rectangular pixels.

In addition, the manual now also contains a section detailing weighting potential lookups and the mechanism used to calculate induced currents with the Shockley-Ramo theorem.

Impact Ionization

Many particle detectors rely on the effect of impact ionization. Allpix Squared has been extended by several models for impact ionization, which is now available in the `GenericPropagation` and `TransientPropagation` modules. This allows among others to study the behaviour of Low Gain Avalanche Detectors (LGAD) and alike, among others simulating the induced current over time as shown below:

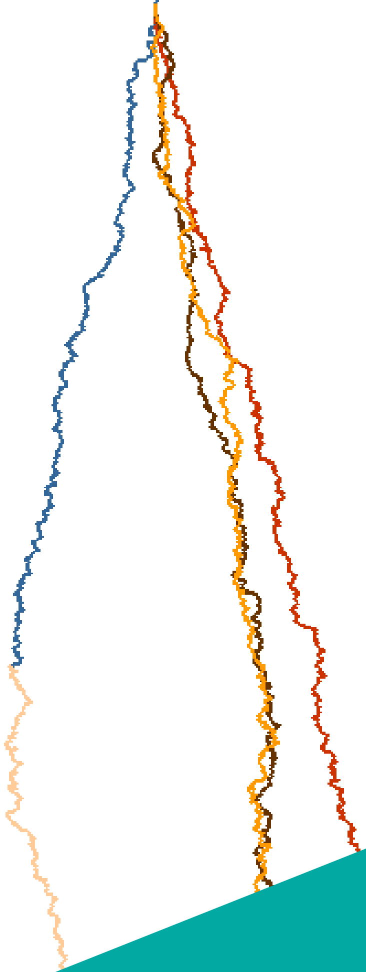


The implementation of impact ionization in the framework is two-fold: one part is modeling the gain factor, the other part

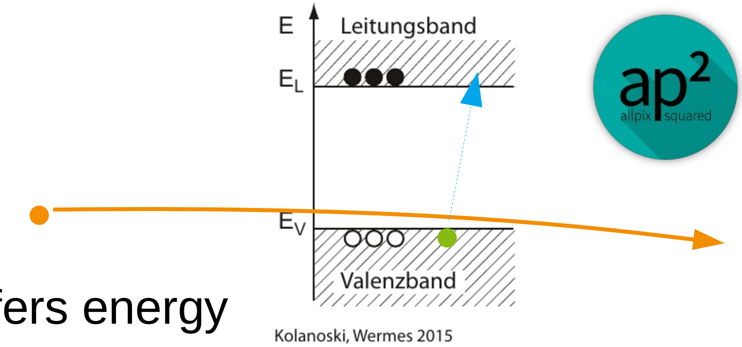
Allpix Squared 3.0 – What's New?

- Website & Documentation
  - ➔ Talk by Stephan just before
- Detector Geometries
  - ➔ Covered briefly, see also Hexagon Håkan, Radial Radek & Max, 3D Me
- Impact Ionization
- Simulation Objects
- More, more, more …

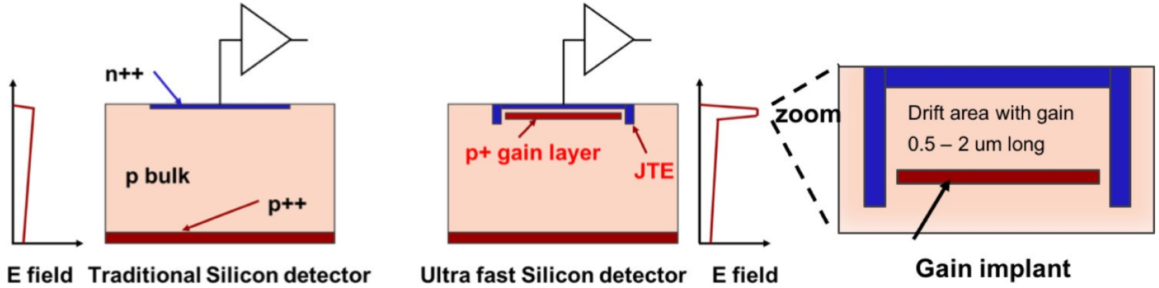
Impact Ionization



Impact Ionization



- Impact Ionization: highly energetic charge carrier transfers energy to electron → generation of secondary charge carrier
  - Possible at high electric fields
- Important mechanism for several types of detectors, e.g. UFSDs / LGADs
  → Multiplication limited to a small region in depth by additional p-n-junction
- Implementation started by F. Pitters (ÖAW) two years ago
- Implemented in !472
- Fully reworked in !964 and !972
- Implemented in *GenericPropagation* and *TransientPropagation*

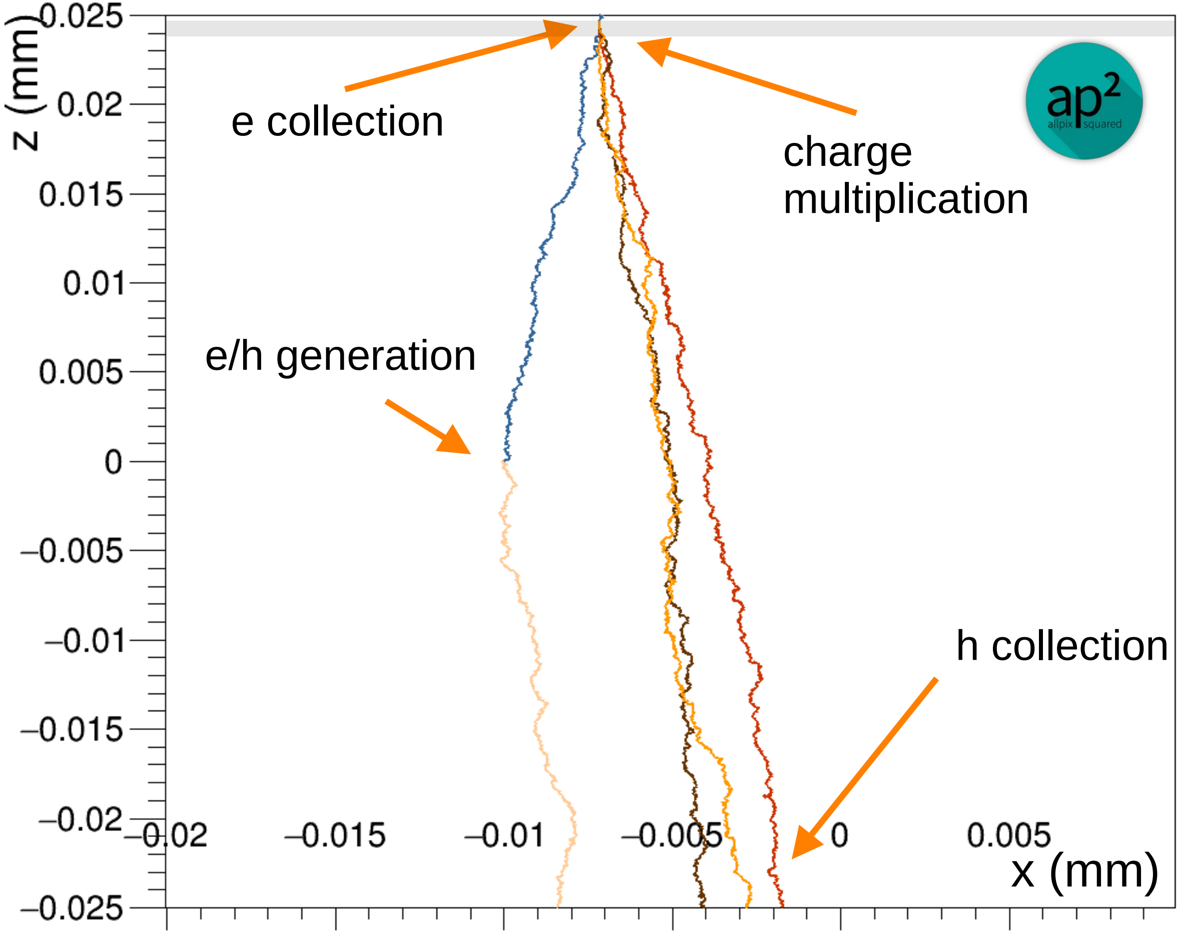


Impact Ionization

- Implemented mechanism:
  - Per step of the propagation, calculate …
    - local gain as a function of electric field
    - number of generated charge carriers stochastically per carrier in a group of carriers
  - Generate opposite-type charge carriers and place at the end of the step
    (for *GenericPropagation* do this only if corresponding type should be propagated)
  - Add same-type charge carriers to the group of charge carriers
- Local gain:
  - Several models implemented (van Overstraeten - De Man, Massey, Okuto-Crowell, Bologna, Optimized (RD50) models, custom model)

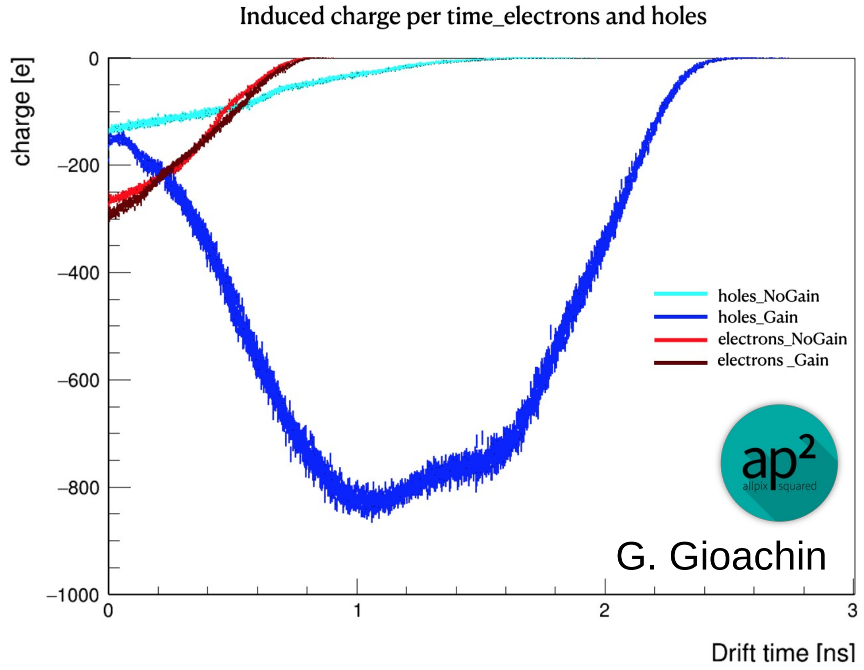
Impact Ionization

- Exemplary linegraph
- Single e/h pair generated
- Gain layer close to the sensor surface
- Propagation in magnetic field for better visibility

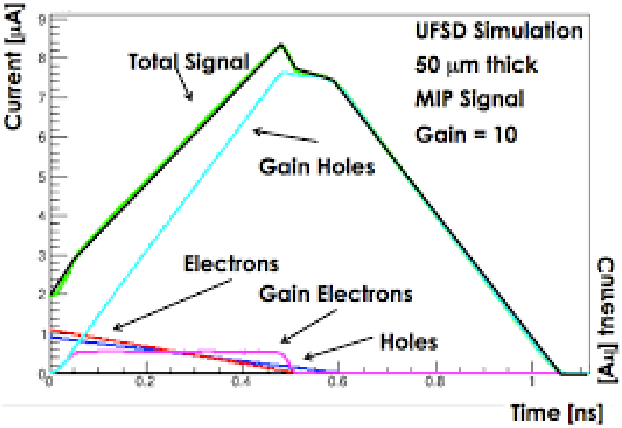


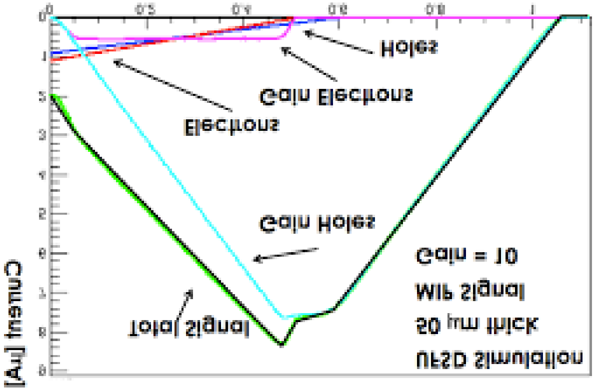
Impact Ionization

- Transient propagation including impact ionisation
  - ➔ Expected pulse shape well reproduced
- Gain highly dependent on selected impact ionisation model
- Many thanks to G. Gioachin, C. Ferrero, N. Cartiglia, F. De Wit and many more for discussion, testing and reporting!

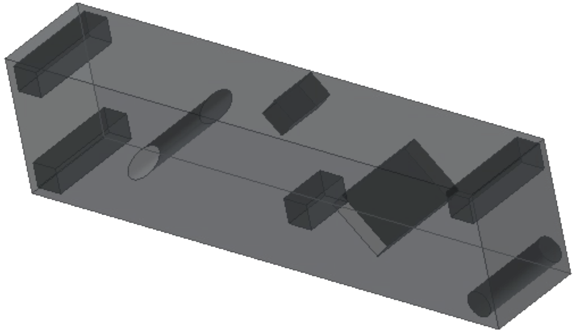


G. Gioachin





N. Cartiglia



Detector Geometry

Detector Assemblies

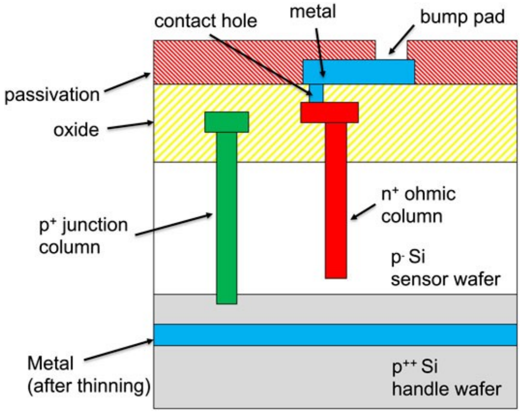
- Sensor type and assembly type can be specified separately
  - Assembly (*type*):
    - *monolithic*
    - *hybrid* (expects information on chip and bump bonds)
  - Sensor type (*geometry*):
    - *pixel*
    - *radial_strip*
    - *hexagon*
- Enables all possible combinations without code replication

```
type = "monolithic"
geometry = "pixel"
```

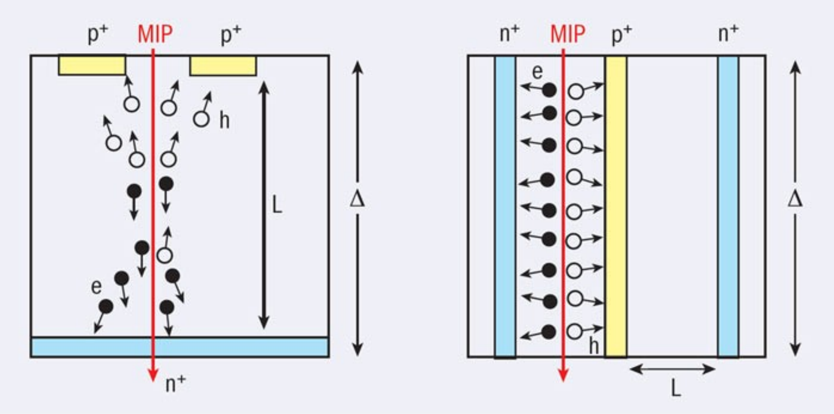
```
type = "hybrid"
geometry = "hexagon"
```

3D Sensors

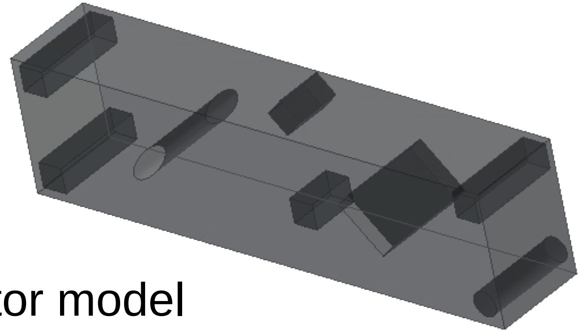
- p- and n-implants implemented as columns through the sensor volume
  - ➔ Generation of a horizontal pn-junction
- Can be applied in harsh radiation environments – e.g. HL-LHC (ATLAS)
- Implemented via !672
  - General principle established
  - Few issues still open



doi:10.3389/fphy.2021.624668



3D Sensors



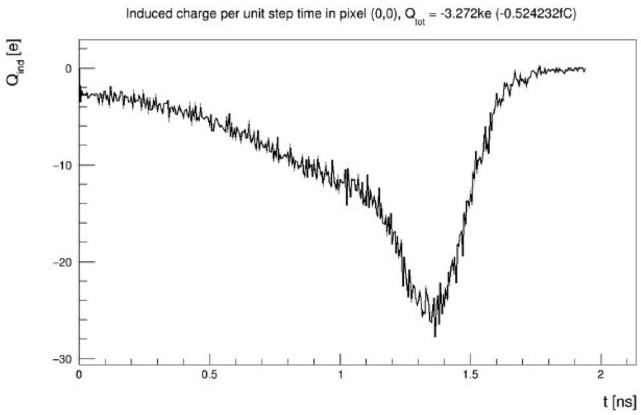
- Definition of per-pixel implants via detector model
  - Position with respect to pixel center
  - Shape & orientation
  - Front/backside
- Implants are replicated over the pixel matrix
- Add as many implants as required,
  syntax similar to support layers (PCB etc)
  - Requires matching electric field map
- Collision detection of charge carriers with implants – motion stops immediately at implant border

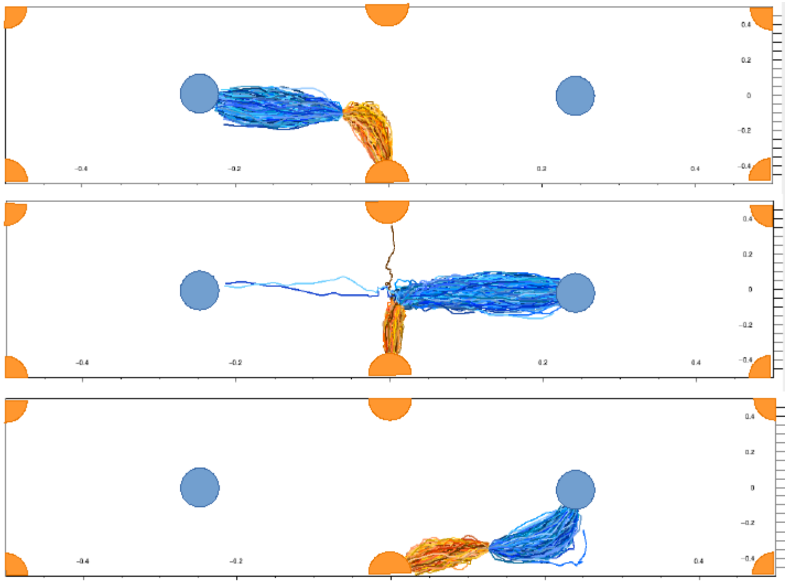
! For now: only sensor-material implants possible

```
type = "monolithic"
number_of_pixels = 3 3
pixel_size = 250um 50um
sensor_thickness = 50um

[implant]
type = frontside
shape = ellipse
size = 13um 13um 50um
offset = 62.5um 0

[implant]
type = backside
shape = rectangle
orientation = 45deg
size = 10um 40um 25um
offset = -62.5um 0
```

3D Sensors

- First simulations with ATLAS 3D sensor geometry
  - Two central front-side columns (collect charge)
  - Six ohmic backside contact columns
- Charge collection & sharing as expected
- Pulses from transient simulation





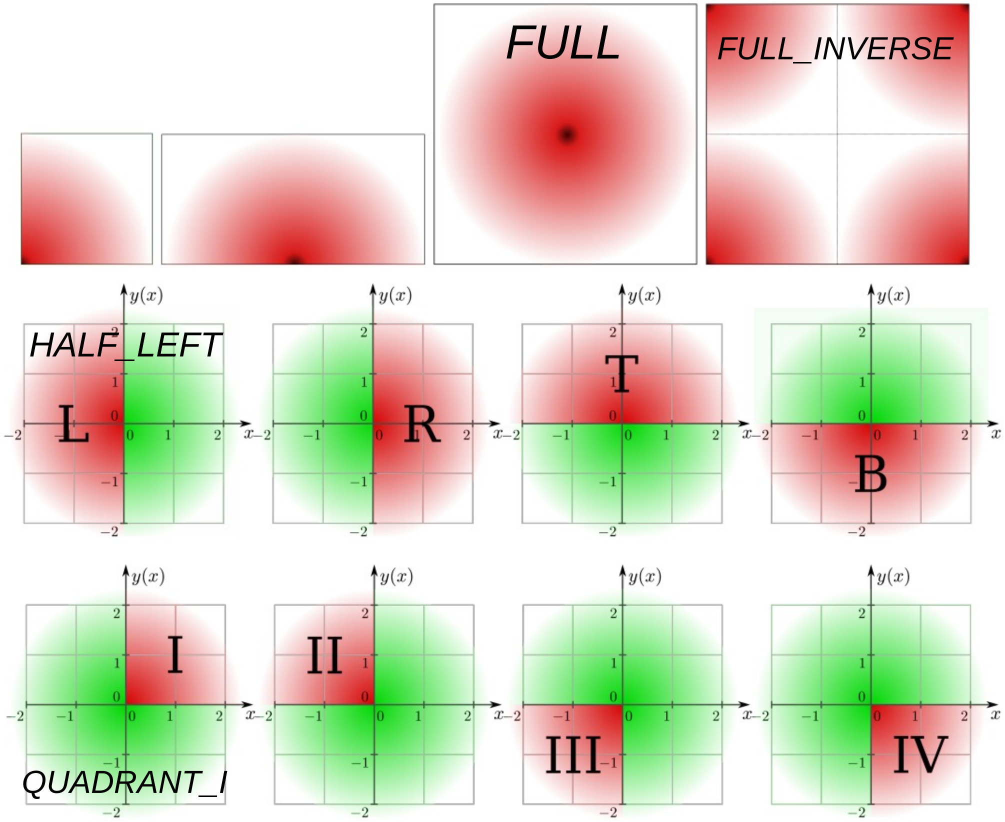
Fields from: Marco Bomben, Gilberto Giuliarelli, Gian-Franco Dalla Betta

Detector Fields

- Lookup of field maps (electric field, doping profile, weighting potential) changed:
- **Before** (< v3.0): provide field of a small region, replicate over sensor area
  - Issue: this only works for rectangular pixel geometries
- **After**: Define field mapped to individual pixels
  - Field is not limited to single pixels (e.g. important for weighting potentials)
  - Plus: possibility to define the *before* solution to support multi-pixel fields
- Implemented in !560

Detector Fields

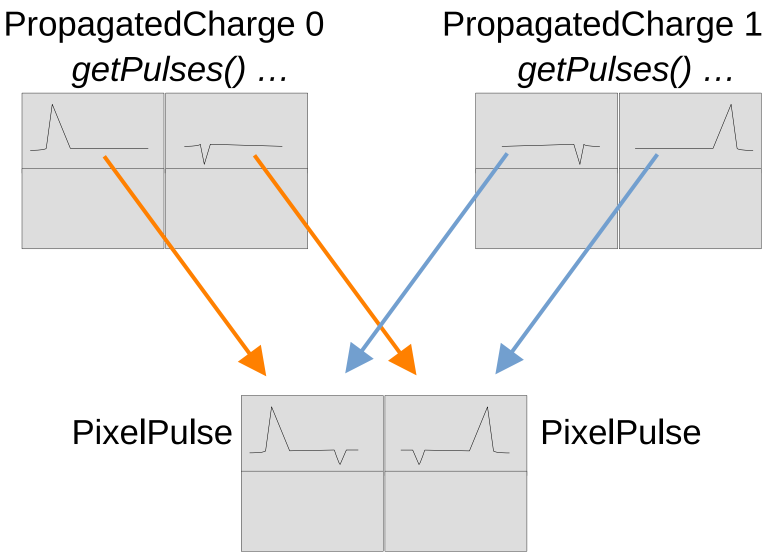
- Provide *field_mapping* parameter in field reader module
- Many options …
    - See right → *PIXEL_…*
    - *SENSOR*:
        - Provide field starting from pixel (0,0)
        - Replicated periodically until opposite sensor edge
        - Replicates *v2.x* behaviour



Simulation Objects

Allpix::PixelPulse

- New object: *PixelPulse*
  - Carries the pulse of a single pixel for a full event
  - Represents the output of a detector front-end, e.g. an amplifier
  - Currently used in *CSADigitizer* module
  - Inherits from *Pulse* (*std::vector<double>*)
    - ➔ Easy to access in analysis
- Implemented in !759




```
PixelPulse mypulse;
for(const auto& bin : mypulse) {
    // do something
}
```

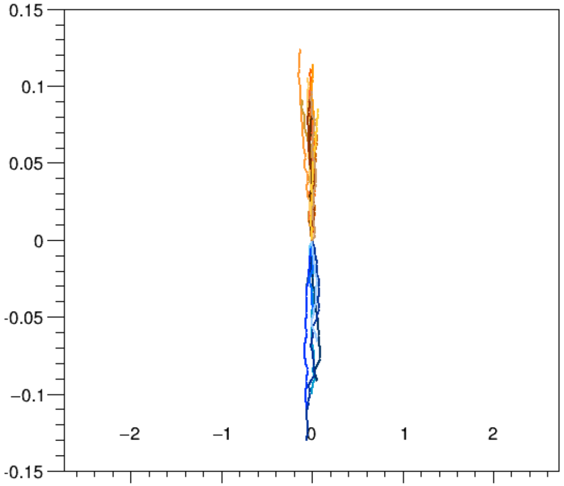
Carrier Status

- Introduced property *CarrierState*
- Indicates status of charge carriers:
  - *MOTION*, *RECOMBINED*, *TRAPPED, HALTED*
  - *HALTED*: Stopped propagation, e.g. reaching sensor surface or implant
- Used during propagation and passed on to *PropagatedCharge* objects
- Implemented in !591
- Benefit:
  - Simplifies filtering in analysis
  - Output linegraphs filtered by status

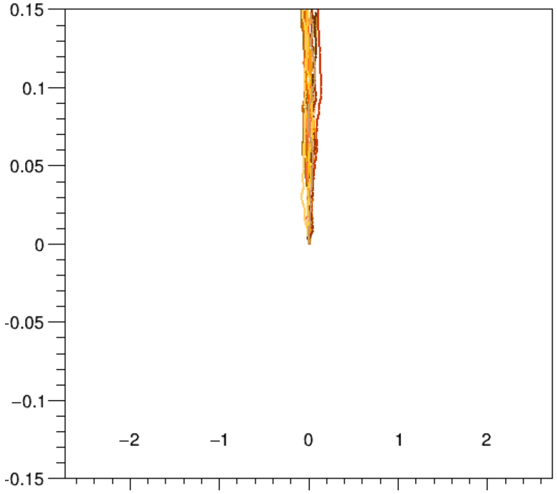
Linegraphs by CarrierState

- Implemented in !592

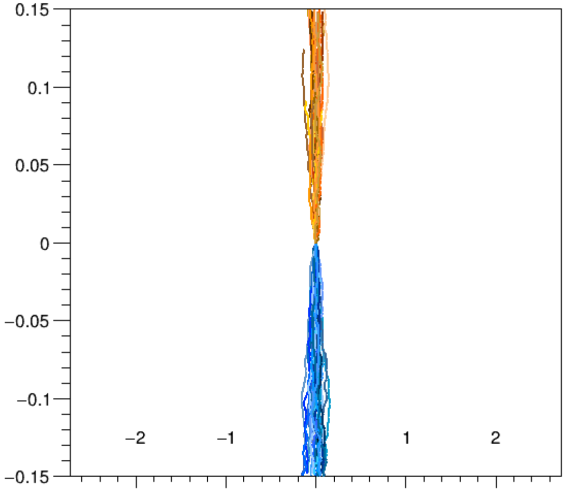
output_linegraphs_recombined



output_linegraphs_collected



output_linegraphs



Summary

Summary

- Allpix Squared has seen a major rework with **v3.0**
- Presented here only the **biggest new features**:
    - Impact Ionisation
    - Detector Geometries & Fields
    - Simulation Objects
- There's **many, many more**, such as …
    - Check mobility selection against detector material
    - New physics models & active/passive materials
    - Possibility to abort individual events
    - Liang-Barsky method: interpolation of senosr edge
    - Geant4 improvements
    - Update CI
    - Improve documentation
    - …

Allpix Squared 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