



# **PYXEL:** The collaborative detection simulation framework

Thibaut Prod'homme<sup>a</sup>, Matej Arko<sup>a</sup>, Frédéric Lemmel<sup>a</sup>, Benoit Serra<sup>b</sup>, Elizabeth George<sup>b</sup>, Enrico Biancalani<sup>c</sup>, Hans Smit<sup>a</sup>, David Lucsanyi<sup>d</sup>, Bradley Kelman<sup>e</sup> <sup>a</sup>ESA, <sup>b</sup>ESO, <sup>c</sup>Leiden Observatory, <sup>d</sup>CERN, <sup>e</sup>OU/CEI

Allpix Squared User Workshop, 17/08/2021

ESA UNCLASSIFIED - For ESA Official Use Only





## What is the common point between these ESA missions?







## What is the common point between these ESA missions?





Same instrument detector techno: CCDs

They all have an instrument simulator with similar objectives

They all built it from scratch using different programming languages, teams, models etc.





## What is the common point between these video games?





💳 💶 📕 🚼 🧰 🚍 🚼 📕 ╧═ 📟 📕 📕 🚍 ∺ 🗤 🖓 🔤 📲 🛃 🖓 トー 🖬 🖉 🗮 🛨 🚍 🚍 🙀 → THE EURO



## What is the common point between these video games?





## were built using the same framework: the UNREAL



engine!

20.00 ST 2 11

### **Unreal Engine**

Unreal Engine is a top game engine developed by Epic Games since 1998. The latest engine is Unreal Engine 4 (UE4). Unreal Engine 5 is expected in 2021.

Unreal Engine is well-documented and easy-to-use. You can use it to develop any type of game — from console to mobile (including Android and iOS). It's even used in industries beyond game development, including automotive.





## Pyxel: the unreal engine behind 2020s instrument simulators?



### **Features**

- Pyxel is a detector simulation framework
- It simulates detector effects on images from optics down to readout electronics
- It can host and pipeline any detector effect models (CIS, CCD, MCT, MKID)

## Collaboration

- Open source python package hosted on Gitlab soon to be released under MIT license
- Joint development effort between ESA and ESO
- First introduced to the community at SPIE 2018
- Beta released in 2019, now version 0.10, 70+ users -> growing community

### What is new?

- New features: Jupyter notebook interface, parallel computing on cluster/cloud, xarray output
- New models: persistency, IPC, cross-talk, CTI (ARCTIC) etc.



## What is a "detection (chain) simulation"?



IN





### OUT





## Why simulating detection chains?



### Instrument and detector simulations are needed each stage of a project

Technology and concept trade-off	
Performance estimate	
Instrument parameter optimization	
<b>Performance verification</b>	
Data processing pipeline validation	
Instrument optimization and calibration	
<b>Performance verification</b>	
Calibrating out instrumental effect	
Supporting data analysis	

Definition Design **Design Consolidation** Manufacture & Testing Launch/First light Commissioning **First science data Operation** 

Scientific exploitation



## Pyxel: motivations & key objectives



Every project develops from scratch its own instrument simulator..

## **Reusability**

Do not reinvent the wheel Spend time on what matters

> Flexible Multi-purpose

Python Based on popular, open-source Python packages Thousands of detector models are described in the literature >> Source code not always available/runnable >> cannot be pipelined

## **Knowledge Transfer**

Collaborative User-oriented Simple Cross-platform

*Python Gitlab Documentation*  Sometimes several implementations >> Not always validated

> **Reliability** Transparency Readability

*Open source Unit test Integrated testing Documentation YAML* 



## **Pyxel** architecture







## Input photon distribution



**Pipeline:** The **core algorithm**, hosting and running the **models**, which **are grouped** into different levels **imitating the working principles** of the detector/instrument

**Detector object:** A **bucket** containing all **properties and data** of the simulated instrument, which may be used or modified by the models

<u>Configuration file</u>: Pyxel's user unique entry point, based on YAML, structured, easy-to-read and understand

### Output image





11

# Configuration file

YAML> structured, easy to read and understand

Structured around 3 sections:

- 1. running mode (which also holds information about the outputs)
- 2. detector: environment, geometry, characteristics and material
- 3. detection pipeline: model groups and functions

.1	parametric:
.2	
.3	mode: sequential
4	parameters:
.5	- key: pipeline.photon_generation.illumination.arguments.level
6	<pre>values: numpy.unique(numpy.logspace(0, 6, 100, dtype=int))</pre>
.7	
8	outputs:
.9	output_folder: 'outputs'
20	
21	ccd_detector:
22	
23	geometry:
24	
25	row: 100 # pixel
26	col: 100 # pixel
27	total_thickness: 10. # um
28	pixel_vert_size: 10. # um
29	pixel_horz_size: 10. # um
30	
31	material:
32	material: 'silicon'
33	
\$4	environment:
55	
50	characteristics:
57	qe: U.S # -
00	eta: 1. # e/photon
19	
11	alle 100 # V/V
12	
13	fue: 100000 # e
14	fwc serial: 200000 # e
10	

```
46 pipeline:
47
48
     # -> photon
49
     photon generation:
       - name: illumination
50
          func: pyxel.models.photon_generation.illumination
51
52
          enabled: true
53
          arguments:
54
              level: 0
55
        - name: shot noise
56
          func: pyxel.models.photon generation.shot noise
57
          enabled: true
58
59
     # photon -> photon
60
      optics:
61
62
      # photon -> charge
63
      charge_generation:
64
        - name: photoelectrons
65
          func: pyxel.models.charge generation.simple conversion
66
          enabled: true
67
68
      # charge -> pixel
69
      charge collection:
70
       - name: simple collection
71
          func: pyxel.models.charge_collection.simple_collection
72
          enabled: true
73
        - name: fixed_pattern_noise
74
          func: pyxel.models.charge_collection.fix pattern noise
75
          enabled: true
76
          arguments:
77
            pixel non uniformity: data/pixel non uniformity.npy
78
        - name: full well
79
          func: pyxel.models.charge collection.simple full well
80
          enabled: true
81
82
      # pixel -> pixel
83
      charge_transfer:
84
      # pixel -> signal
85
      charge measurement:
86
87
       - name: simple measurement
88
          func: pyxel.models.charge_measurement.simple_measurement
          enabled: true
89
90
        - name: output node noise
91
          func: pyxel.models.charge_measurement.output_node_noise
92
          enabled: true
93
          arguments:
94
            std deviation: 1.e-6 # Volt
95
96
      # signal -> image
97
      readout electronics:
98
        - name: simple_amplifier
99
          func: pyxel.models.readout electronics.simple amplifier
100
          enabled: true
101
        - name: simple processing
102
          func: pyxel.models.readout_electronics.simple_processing
103
          enabled: true
```

### ■ 🔜 📲 🚍 💳 🛶 📲 🔚 🔚 🔜 📲 🚍 🛻 🚳 🛌 📲 🗮 📥 🖬 👫 📲 🛨



## 4 running modes

- Single run: one image in, one image out, single pipeline run
   > quick look/ health check
- **Parametric**: pipeline is run multiple times looping over a range of model or detector parameters > sensitivity analysis
- Calibration: optimize model or detector parameters to fit target data sets using a user-defined fitness function/figure of merit > model fitting, instrument optimization
- Dynamic: the pipeline is run N times incrementing time and saving detector attributes > simulation of non-destructive readout mode, and time-dependent effects (e.g. persistence)













#### **References:**

[1] M. D. Perrin et al.: "Simulating point spread functions for the James Webb Space Telescope with WebbPSF", Space Telescopes and Instrumentation 2012, SPIE Proc., Vol. 8442, pp. 11. (2012).
[2] Lucsanyi, D. and Prod'homme, T., "Simulating Charge Deposition by Cosmic Rays Inside AstronomicalImaging Detectors," IEEE Transactions on Nuclear Science67, 1623–1628 (July 2020)
[3] A. Short et al.: "An analytical model of radiation-induced Charge Transfer Inefficiency for CCD detectors", Monthly Notices of the Royal Astronomical Society 430(4), 3078{3085 (2013).
[4] B. J. Rauscher: "Teledyne H1RG, H2RG, and H4RG Noise Generator", Publications of the Astronomical Society of the Pacific 127(957), 1144 (2015).

14



(circular aperture)

(displacement damage in irradiated CCD)

(corr. & uncorr. pink, white, ACN, PCA0)

#### **References:**

[1] M. D. Perrin et al.: "Simulating point spread functions for the James Webb Space Telescope with WebbPSF", Space Telescopes and Instrumentation 2012, SPIE Proc., Vol. 8442, pp. 11. (2012). [2] Lucsanyi, D. and Prod'homme, T., "Simulating Charge Deposition by Cosmic Rays Inside AstronomicalImaging Detectors," IEEE Transactions on Nuclear Science67, 1623–1628 (July 2020) [3] A. Short et al.: "An analytical model of radiation-induced Charge Transfer Inefficiency for CCD detectors", Monthly Notices of the Royal Astronomical Society 430(4), 3078{3085 (2013). [4] B. J. Rauscher: "Teledyne H1RG, H2RG, and H4RG Noise Generator", Publications of the Astronomical Society of the Pacific 127(957), 1144 (2015).

![](_page_15_Picture_0.jpeg)

## Try Pyxel and join the collaboration!

![](_page_15_Picture_2.jpeg)

![](_page_15_Figure_3.jpeg)

### Check website: esa.gitlab.io/pyxel

- blog to share the latest news, developments, and advertise job ads
- Pyxel's improved and comprehensive documentation
- detailed contribution guide to help more advanced users contributing to Pyxel via Gitlab e.g. improving documentation, adding new models, reporting bugs
- Link to examples and tutorials (also available on Binder without prior installation)

### Request membership: pyxel@esa.int

 Gitlab + mailing list + chat platform: where users can create/fix issues, reach out to the community for technical support and share experience, tips, hacks

16

![](_page_16_Picture_0.jpeg)

![](_page_16_Picture_1.jpeg)

Detector modelling is key to achieve the demanding objectives of the current and next generation of instruments onboard scientific space missions and ground-based experiments.

We developed Pyxel as a collaborative tool to provide a common framework to promote reusability, knowledge transfer, and reliability in the instrumentation community.

Pyxel has now reached v0.10: in beta but stable, contains many models and new features and it is becoming ever easier to use and contribute to.

Next milestone for Pyxel is version 1.0 and open source release. Hot topics right now: addition of ArCTIc CTI model, calibration of persistence model, possibility of adding APD detectors...

Pyxel: the "unreal engine" behind 2020s instrument simulators and beyond!