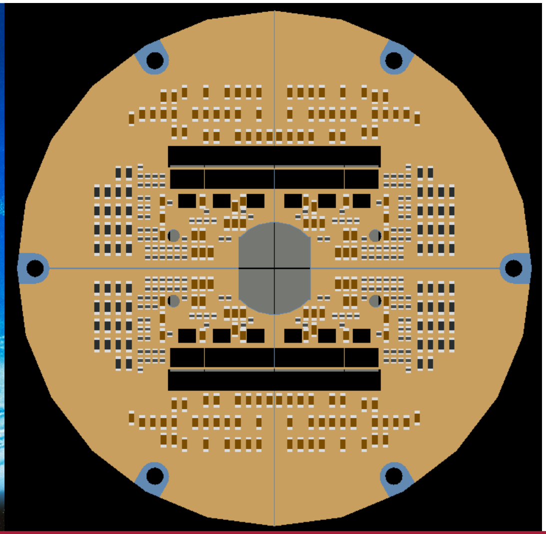
EBERHARD KARLS UNIVERSITÄT TÜBINGEN

FACULTY OF SCIENCE

Institute of Astronomy and Astrophysics



Geant4Py for X-ray Detector Background Simulations

Christian Pommranz

S. Diebold, C. Tenzer, A. Santangelo

22.10.19, 14th Geant4 Space Users Workshop

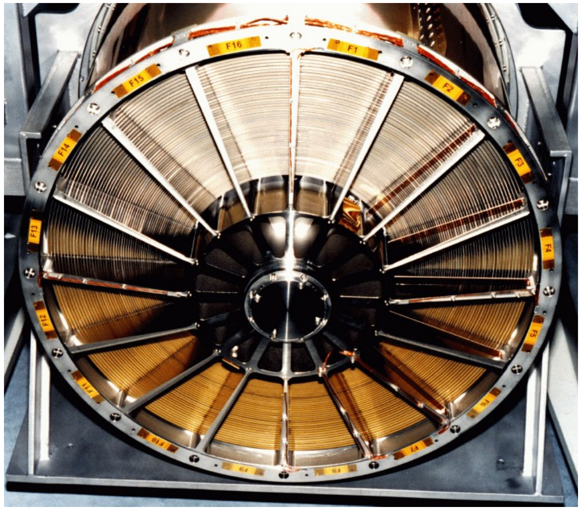
Outline

- XMM-Newton EPIC-pn Background Simulations
- Geant4Py
- Use Case: X-ray Detector Background Simulation with Geant4Py

XMM-Newton

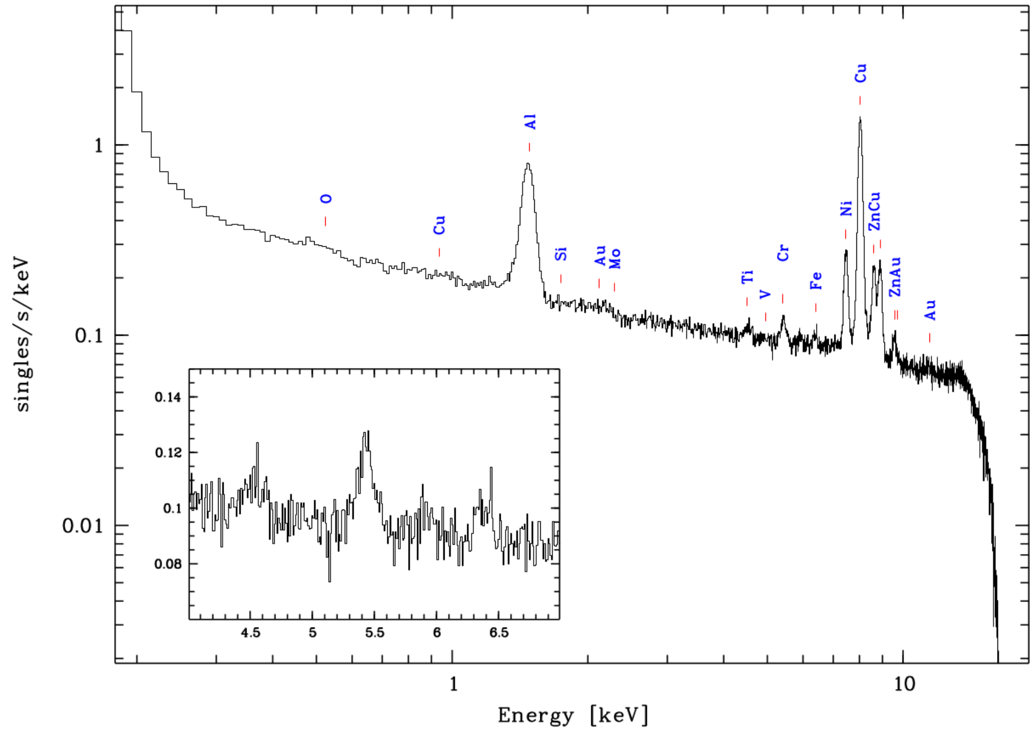
- Launched by ESA in December 1999 with an Ariane 5 rocket
- Still an active mission
- 3 EPIC (European Photon Imaging Camera) CCD cameras
  - 2x EPIC MOS (Metal Oxide Semiconductors), 0.1 - 12 keV
  - 1x EPIC-pn, 0.15 - 15 keV

Images: ESA



EPIC-pn Background

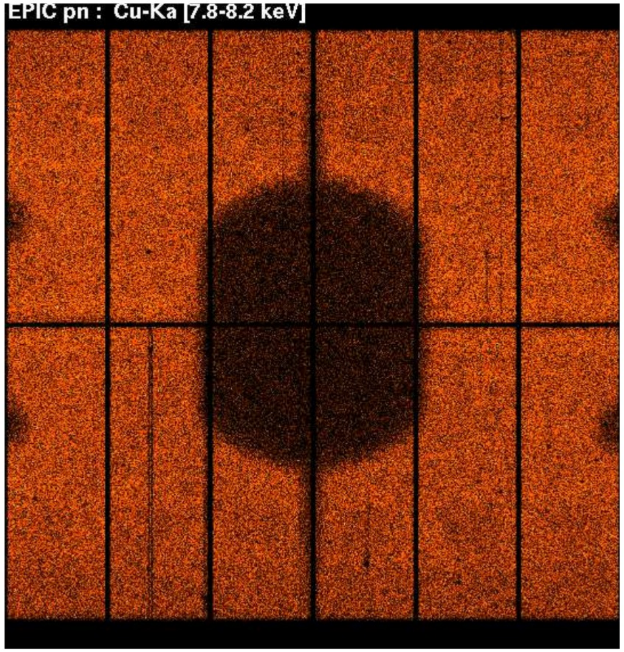
- EPIC pn-CCD background spectrum obtained with „closed" filter observations
- Several prominent features (e.g. Al-Kα, Ni-Kα, Cu-Kα)
- Fluorescent X-ray emission induced by high-energy particles



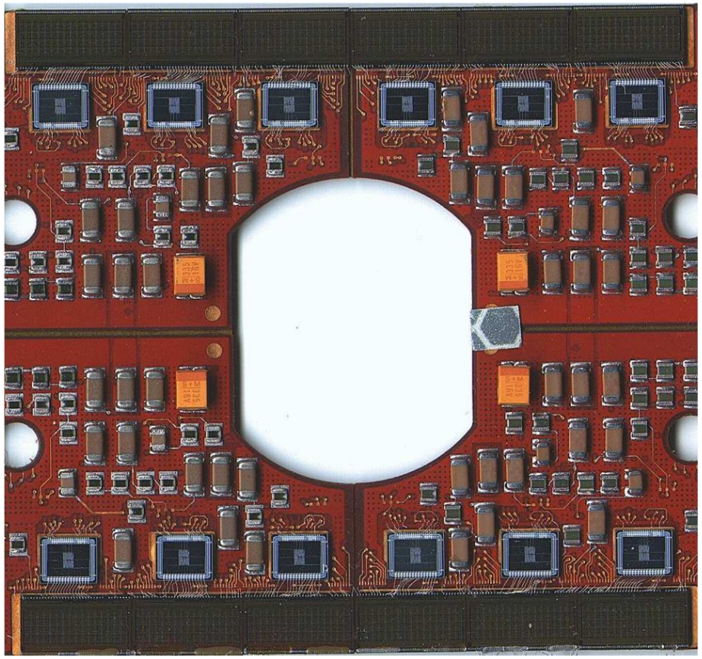
(Freyberg et al., 2004)

Spatial Distribution

- Prominent lines in background spectrum show strong spatial inhomogeneity
- In correspondence with electronics mounted on PCB board



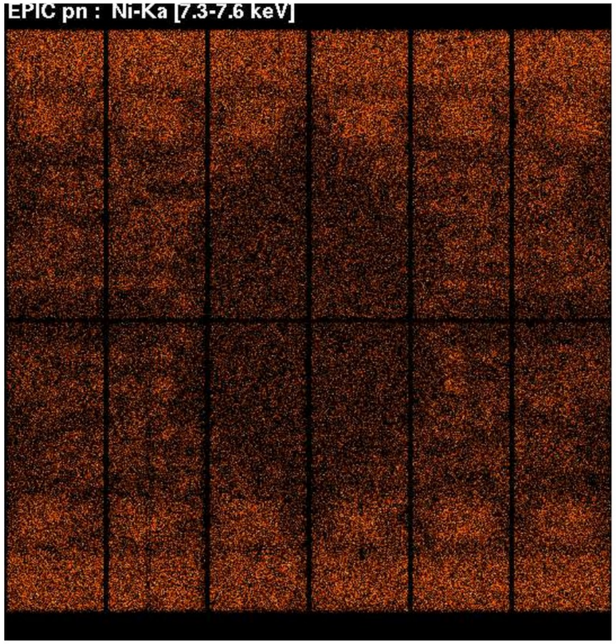
(Freyberg et al., 2004)



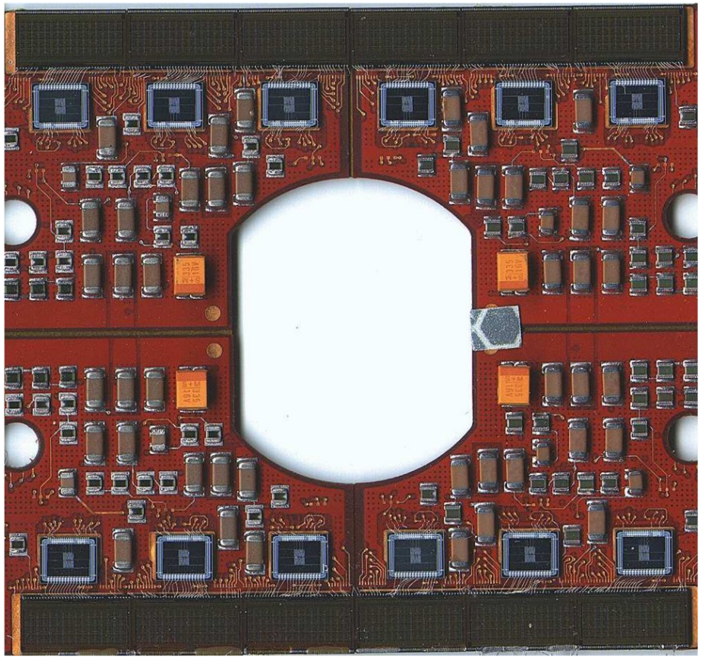
Spatial Distribution

- Prominent lines in background spectrum show strong spatial inhomogeneity

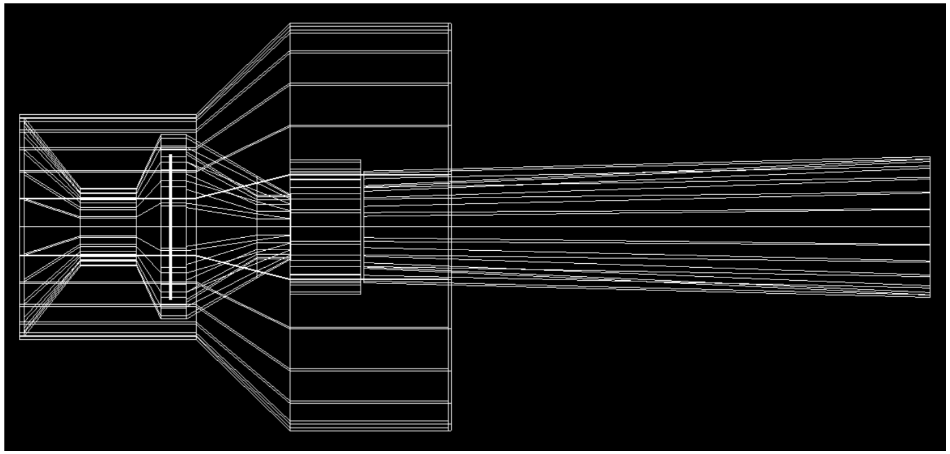
- In correspondence with electronics mounted on PCB board



(Freyberg et al., 2004)



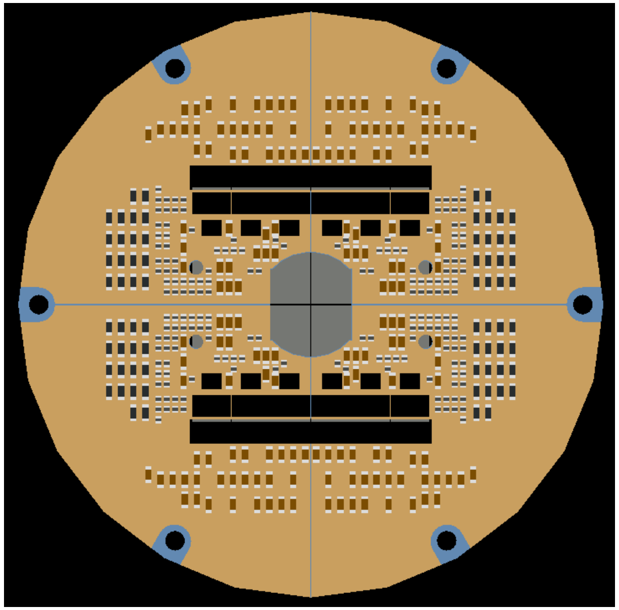
Geant4 Simulation



- Geometry described with GDML

- CCDs defined as sensitive detectors using GDML auxiliary information

→ Geometry separated from application source code



Simulate Diffuse Radiation in Space

- Produce isotropic radiation in space with Geant4 GPS cosine-law angular emission
- Attention: Normalization! (G. Santin, 2007)

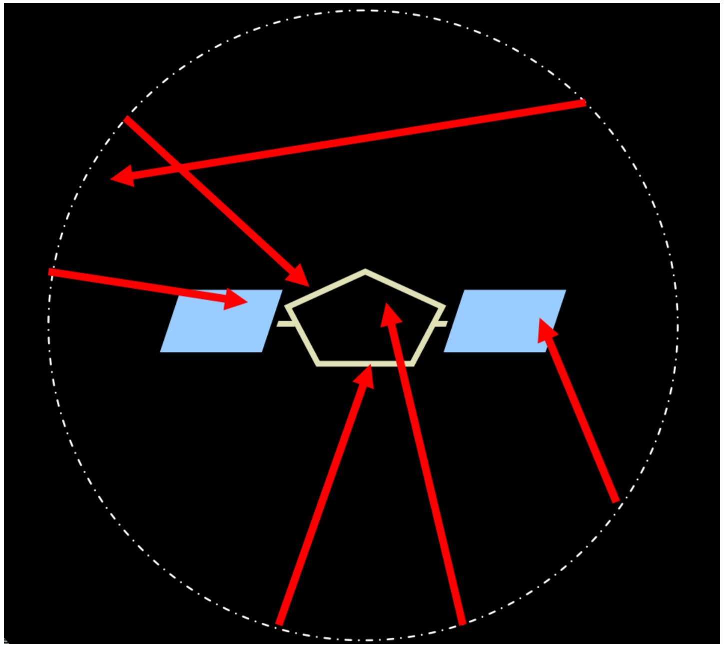


Image: Santin, 2007

```
/gps/pos/type      Surface
/gps/pos/shape     Sphere
/gps/pos/centre    0. 0. 0. cm
/gps/pos/radius    5. m
/gps/ang/type      cos
/gps/ang/mintheta  0. deg
/gps/ang/maxtheta  9. deg
```

Parts in Geant4 Application (C++)

- Remaining parts in Geant4 Application:
  - Read/setup geometry from GDML
  - Code plugging everything together
  - Detector implementation
  - Output to ROOT

→ Most of the code is executed only once at initialization or on interactions between particles and detector (rare cases in this simulation).

→ Geant4 application written in Python?

Hybrid Approach

In many cases execution speed and development speed are opposed

Execution Speed ⟷ **Development Speed**

- Heavily optimized code
- Platform specific code
- Compiled languages

- Rapid prototyping
- Dynamical scripting languages

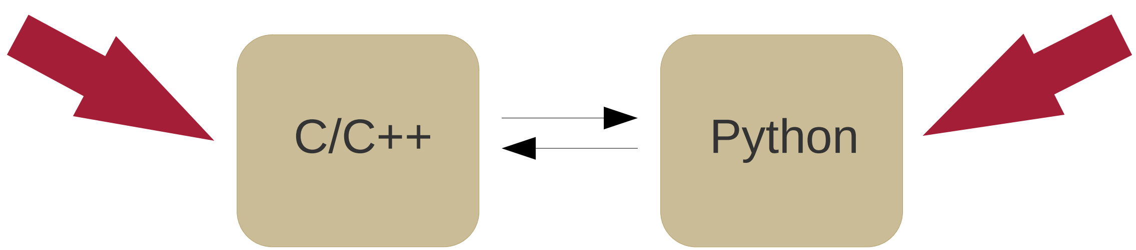
Hybrid Approach

In many cases execution speed and development speed are opposed

Execution Speed **Development Speed**

- Heavily optimized code
- Platform specific code
- Compiled languages

- Rapid prototyping
- Dynamical scripting languages



Geant4Py

- Geant4 Python interface
- Included in the Geant4 distributions since version 8.1
- Interfacing done with Boost-Python
  - Calls Python → C++ and C++ → Python possible
- Natural Pythonization approach, direct mapping
- Various levels of pythonization in Geant4 applications possible

(Murakami and Yoshida, 2006)

C++

```
class MySteppingAction :
  public G4UserSteppingAction {
// My Stepping Action

void SteppingAction
      (const G4Step* step) {
  G4StepPoint* preStepPoint=
        step->GetPreStepPoint();
  G4Track* track=
        step->GetTrack();
  G4VTouchable* touchable=
        track->GetTouchable();

  if(preStepPoint->GetCharge()==0)
     return;

  G4ThreeVector pos=
    preStepPoint->GetPosition()
  G4int id=
    touchable->GetReplicaNumber()
  G4double dedx=
      step->GetTotalEnergyDeposit();
  }
};
```

Python

```
class MySteppingAction
   (G4UserSteppingAction):
   "My Stepping Action"

  def SteppingAction(self, step):

    preStepPoint=
      step.GetPreStepPoint()
    track= step.GetTrack()

    touchable= track.GetTouchable()


if(preStepPoint.GetCharge()==0):
      return

     pos= preStepPoint.GetPosition()

     id=
touchable.GetReplicaNumber()

     dedx=
       step.GetTotalEnergyDeposit()
```

Exposing C++ Classes

```cpp
using namespace pyG4Event;

// ============================================================================
// module definition
// ============================================================================
void export_G4Event()
{
  class_<G4Event, G4Event*, boost::noncopyable>("G4Event", "event class")
    .def(init<G4int>())
    // ---
    .def("Print",              &G4Event::Print)
    .def("Draw",               &G4Event::Draw)
    .def("SetEventID",         &G4Event::SetEventID)
    .def("GetEventID",         &G4Event::GetEventID)
    .def("SetEventAborted",    &G4Event::SetEventAborted)
    .def("IsAborted",          &G4Event::IsAborted)
    // ---
    .def("AddPrimaryVertex",   &G4Event::AddPrimaryVertex)
    .def("GetNumberOfPrimaryVertex", &G4Event::GetNumberOfPrimaryVertex)
    .def("GetPrimaryVertex",   &G4Event::GetPrimaryVertex,
     f_GetPrimaryVertex()[return_internal_reference<>()])
    // ---
    .def("GetTrajectoryContainer", &G4Event::GetTrajectoryContainer,
     return_internal_reference<>())
    .def("SetUserInformation", &G4Event::SetUserInformation)
    .def("GetUserInformation", &G4Event::GetUserInformation,
     return_internal_reference<>())
    ;
```

Geant4Py Modifications

- Expose new Geant4 classes in Python
  - G4GeneralParticleSource
  - G4PhysListFactory
  - G4SDManager
  - …
- Enable support for Python 3.2+
- Unified Python2 + Python3 source base

  → Allows to use Python2 and Python3 Geant4Py applications on the same system

- Extending Geant4Py is straightforward, but …

Challenges in Extending Geant4Py

- Complexity in Geant4 translates into complexity in bindings

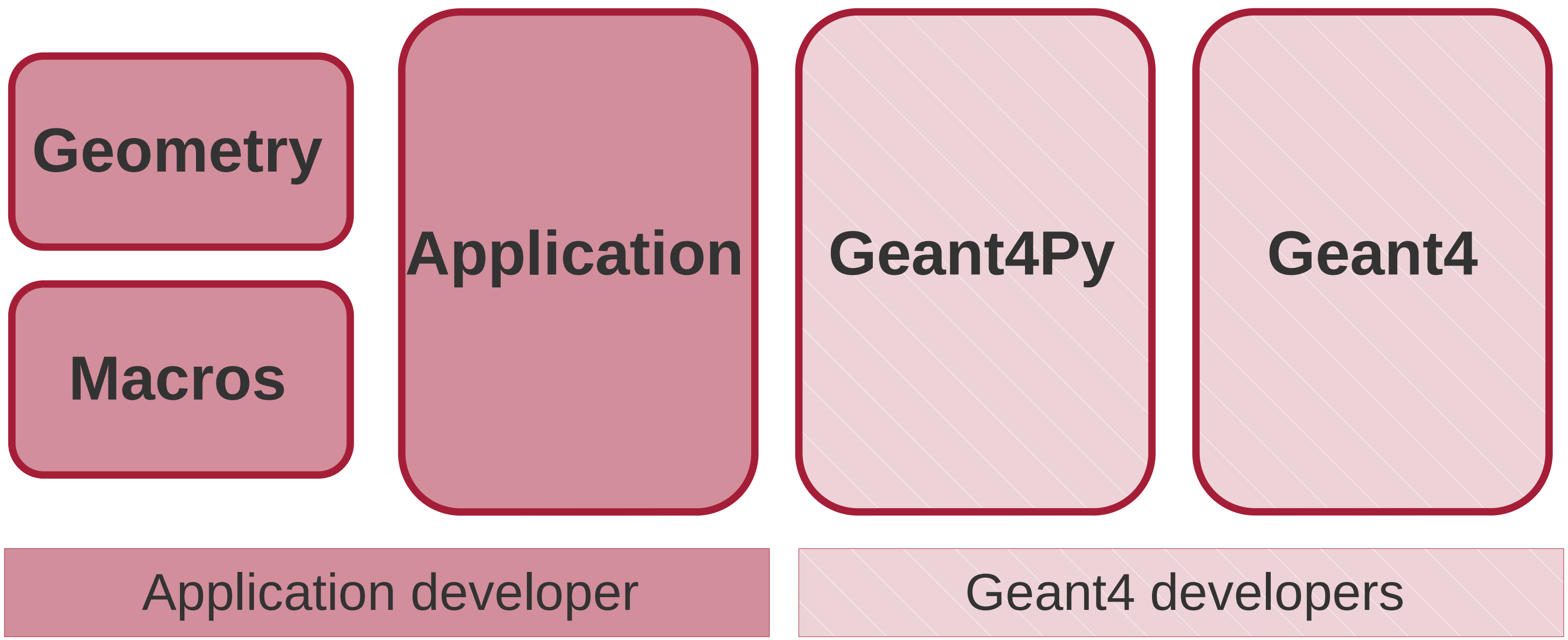
Simple is better than complex.
Complex is better than complicated.
– Zen of Python –

```
struct G4GDMLAuxStructType
{
   G4String type;
   G4String value;
   G4String unit;
   std::vector<G4GDMLAuxStructType>* auxList;
};

typedef std::vector<G4GDMLAuxStructType> G4GDMLAuxListType;

typedef std::map<G4LogicalVolume*,G4GDMLAuxListType> G4GDMLAuxMapType;
```

- Debugging can be time consuming (Interplay CPython, Geant4, Boost)

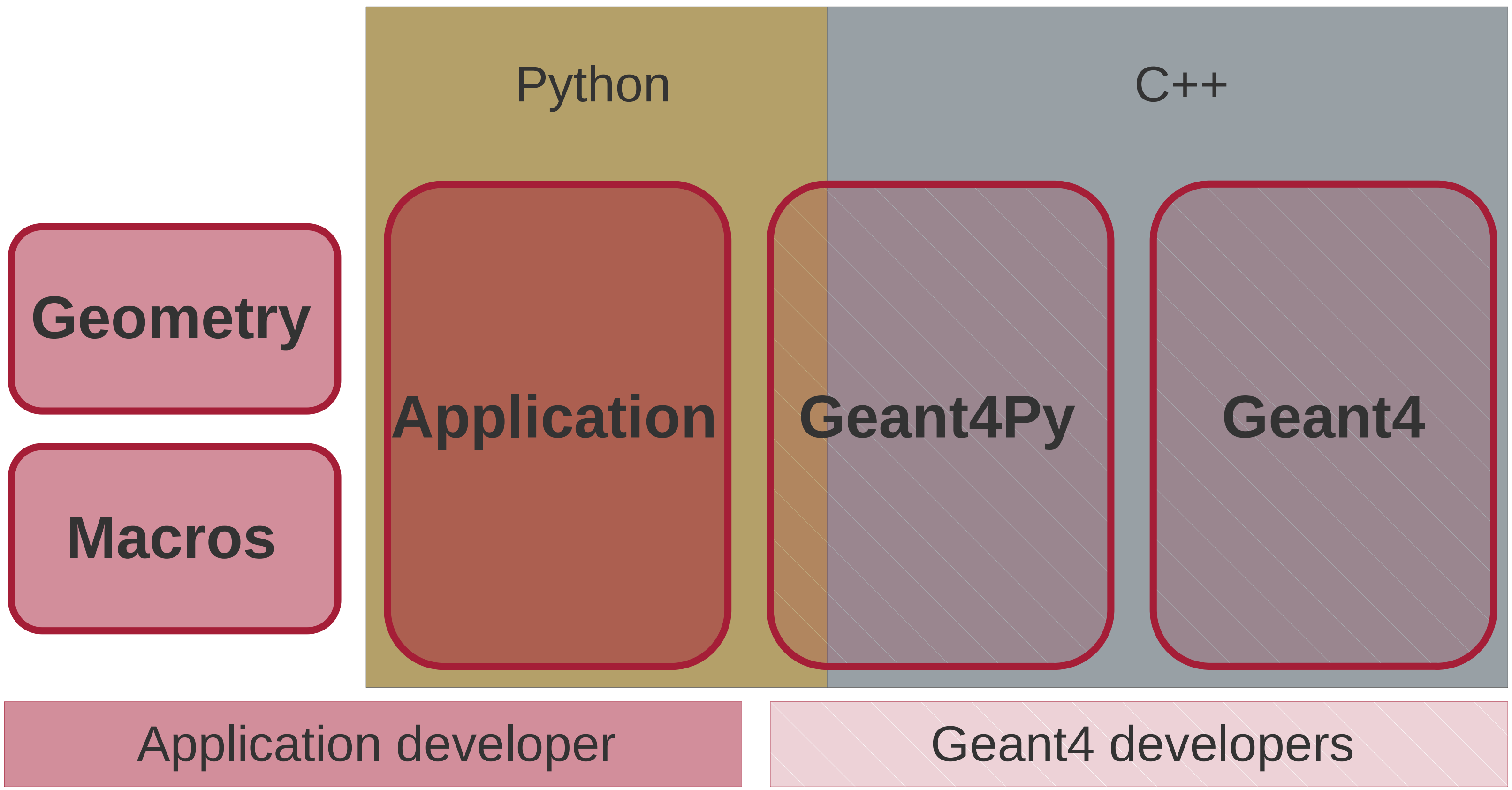
Code

- Application completely in Python + GDML + Geant4 Macros



Code

- Application completely in Python + GDML + Geant4 Macros



```python
class Detector(G4VSensitiveDetector):
    def __init__(self, name, tree):
        super().__init__(name)
        [...]
        self.hit_list = []
        self.tree = tree

    def Initialize(self, hce):
        self.hit_list = []

    def ProcessHits(self, step, rohist):
        if step.GetTotalEnergyDeposit() > 0.:
            hit = {'edep': step.GetTotalEnergyDeposit(),
                   'vol_name': name,
                   }
            self.hit_list.append(hit)
        return True

    def EndOfEvent(self, hce):
        if len(self.hit_list) == 0:
            return
        [...]
```



```
[...]

from Geant4 import gApplyUICommand, gRunManager, G4PhysListFactory,
RanecuEngine, HepRandom
import g4py.GeneralParticleSource

engine = RanecuEngine()
HepRandom.setTheEngine(engine)

# gdml.DetectorConstruction is derived from G4VUserDetectorConstruction
dc = gdml.DetectorConstruction('gdml/XMM-Newton/XMM.gdml')
gRunManager.SetUserInitialization(dc)

plf = G4PhysListFactory()
pl = plf.GetReferencePhysList("QGSP_BIC_HP_PEN")
gRunManager.SetUserInitialization(pl)

gps = g4py.GeneralParticleSource.Construct()

gApplyUICommand("/control/execute simulation.mac")
```


Runtime Comparison

- Numbers only valid for the specific application!

- Runtime:
  - 10^7 primary photons following the Gruber spectrum between 4.5 MeV and 1 GeV
  - 10 runs with each application

	C++	Python
Average Runtime in s	96.0	118.0

→ ~23 % runtime overhead for Geant4Py

Profiling

10^5 primary particles

Name	Call Count	Time (ms)		Own Time (ms)	
xmm_background.py	1	19006	100,0 %	16815	88,5 %
<built-in method _imp.create_dynamic>	21	435	2,3 %	435	2,3 %
__init__	1	350	1,8 %	350	1,8 %
<built-in method libPyROOT.LookupCppEntity>	101	118	0,6 %	118	0,6 %
root_open	1	235	1,2 %	102	0,5 %
__getattr__	96	179	0,9 %	88	0,5 %
__finalSetup	1	118	0,6 %	80	0,4 %
Write	1	70	0,4 %	70	0,4 %
EndOfEvent	100000	94	0,5 %	68	0,4 %
<method 'load_module' of 'zipimport.zipimporte	16	241	1,3 %	55	0,3 %
snake_case_methods	34	235	1,2 %	51	0,3 %
Initialize	100000	48	0,3 %	48	0,3 %
__init__.py	1	132	0,7 %	47	0,2 %
<method 'sub' of 're.Pattern' objects>	17458	94	0,5 %	41	0,2 %
<built-in method builtins.getattr>	15472	188	1,0 %	41	0,2 %
<built-in method marshal.loads>	125	25	0,1 %	25	0,1 %
<built-in method builtins.len>	112703	22	0,1 %	22	0,1 %
Close	1	19	0,1 %	19	0,1 %

Outlook

- Possible things to do:
  - Expose more features/classes from Geant4 to Geant4Py
  - Introduce Geant4Py compile switch to CMake configuration
    - This could make Geant4Py more visible to the users
  - Replace Boost-python with Pybind11?
  - More pythonic interface: rootpy vs. Pyroot?

- Get presented changes into upstream repository

EBERHARD KARLS UNIVERSITÄT TÜBINGEN

Thank you.

Contact:

Christian Pommranz
Sand 1
72074 Tübingen · Germany
Phone: +49 7071 29-75463
pommranz@astro.uni-tuebingen.de