Coding up LUXE's Central Software: Theia

Arka Santra, Noam Tal Hod
June 22, 2021
Weizmann Institute of Science

Goal

- Create a **central software structure** for the data analysis of LUXE
  - Inspired by the Athena setup of ATLAS.
  - Federico gave a nice overview of Athena recently.
  - This setup will include everything that comes after the GEANT4 simulation:
    - Digitization, Reconstruction, Analysis
  - May incorporate ptarmigan and the G4 simulation (but may remain external to the release)
- Modular structure of the setup
  - Parallel use by several sub-detectors
  - Modular: common modules are (e.g. B-field transport, config readers, etc) written centrally
  - Easy to understand, implement, debugging
- Modern CMake will be used to link the libraries:
  - Cross-platform usage
- Starting from the tracker's FastSim code
  - Many modules already implemented (fully/partially)
  - Need some cleaning and adapt to new structure.
  - Necessary for tracker.

Structural Outline: Theia

- "Theia is the goddess of sight and the shining light of the clear blue sky. She is the consort of Hyperion (the god of light), and mother of Helios (the sun), Selene (the moon), and Eos (the dawn)."
- The setup as we are thinking it (first pass):
  - Theia/<Module>/<Sub-Modules1>, <Sub-modules2> …
  - Each package will have basic structure like src/, header/, python/. Root/, share/, config/
    - A **header** (same name as the package) directory having all the header (*.h) files.
    - An **src** directory containing the source code (*cxx files).
    - A **python** directory containing python scripts.
    - A **Root** directory for Root macros
- Example
  - Theia/Common/package1, package2/…
    - utilized by all sub-detectors: Geometry, Logger, Particles, ConfigReader etc.
  - Each package has source, run, build structure
  - The source directory contains all the code setups.

```
Theia
├── BSMCalorimeter
├── Cherenkov
├── Common
│   ├── ConfigReader
│   │   ├── build
│   │   ├── run
│   │   └── source
│   │       ├── CMakeLists.txt
│   │       └── ConfigReader
│   │           ├── CMakeLists.txt
│   │           ├── ConfigReader
│   │           └── src
│   │               └── ConfigReader.cxx
│   ├── Geometry
│   │   ├── build
│   │   ├── run
│   │   └── source
│   │       ├── CMakeLists.txt
│   │       └── Geometry
│   ├── Logger
│   │   ├── build
│   │   ├── run
│   │   └── source
│   │       ├── CMakeLists.txt
│   │       └── Logger
│   │           ├── CMakeLists.txt
│   │           ├── Logger
│   │           │   └── LUXELog.h
│   │           └── src
│   │               └── LUXELog.cxx
```

```
├── Magnets
│   ├── build
│   ├── run
│   └── source
│       ├── CMakeLists.txt
│       └── Magnets
│           ├── CMakeLists.txt
│           ├── Magnets
│           │   └── Magnets.h
│           └── src
│               └── Magnets.cxx
├── Particles
│   ├── build
│   ├── run
│   └── source
│       ├── CMakeLists.txt
│       └── Particles
│           ├── CMakeLists.txt
│           ├── Particles
│           │   ├── LUXEUtility.h
│           │   └── Particle.h
│           └── src
│               └── Particle.cxx
└── Plotting
    ├── build
    ├── run
    └── source
        ├── CMakeLists.txt
```

The common module

- Contain the core modules like the **Magnets, Particles, Geometry, ConfigReader, Logger** and **Plotting**
  - **Magnets:** - dimension, field, position, type configurable
    - Can be used by all sub-detectors.
  - **Particles**: particle origin, momentum, id, charge
    - Good for cross-talk between sub-detectors.
  - **Geometry**: quick setup of all LUXE sub-detectors (under discussion).
  - **Plotting**: useful for making plots and do a quick check of detector quality parameters.
  - **ConfigReader:**
    - Each sub-detector will have its own config file - containing dimension, position, layout, magnetic field.
    - One config file that can be used as a master file.
      - Good for playing with several sub-detectors.
    - Some information can be read directly from the G4 output files.
  - **Logger:** useful for debugging

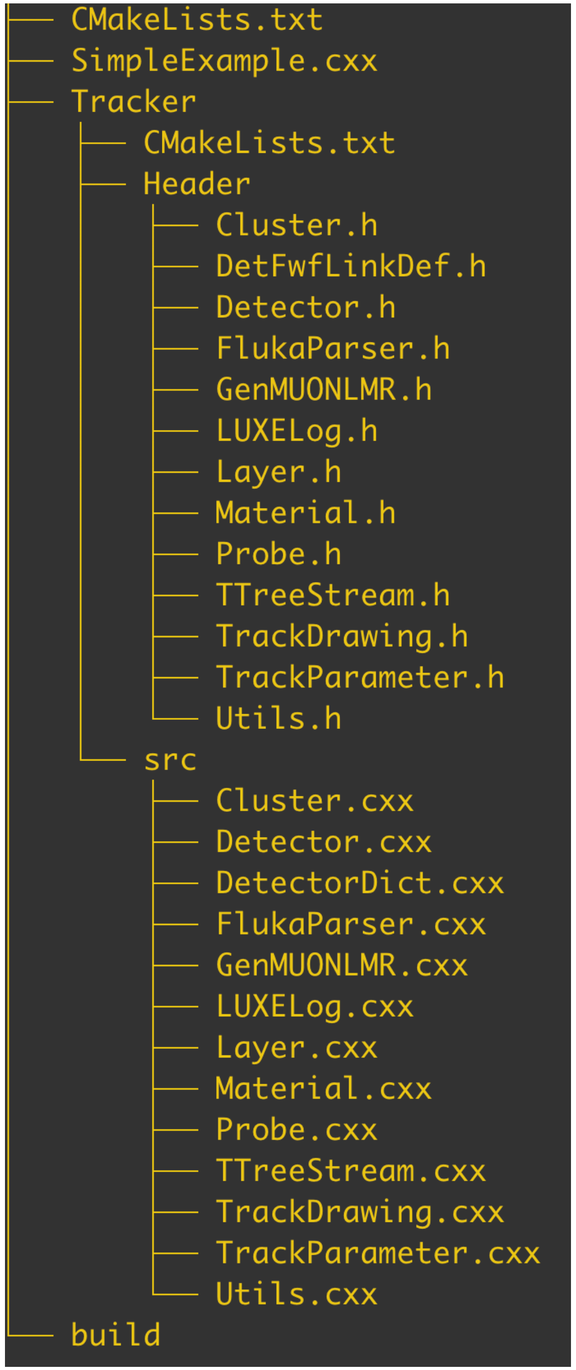
The header and src directory

Working now on the Tracker

- **header** directory: Class structure.
  - Class parameters.
  - Declaration of class functions, *setters and getters.*
  - Private variables
  - Inheritance.
- Thinking of several classes:
  - **Layer, Material, Track, TrackDiagram, Probe, Detector**…
  - Track and TrackDiagram -> inherit from Particle class
  - Detector -> inheritance from Geometry
  - …
- The **src** directory:
  - Definition of class functions.
  - Utilization of private variables.

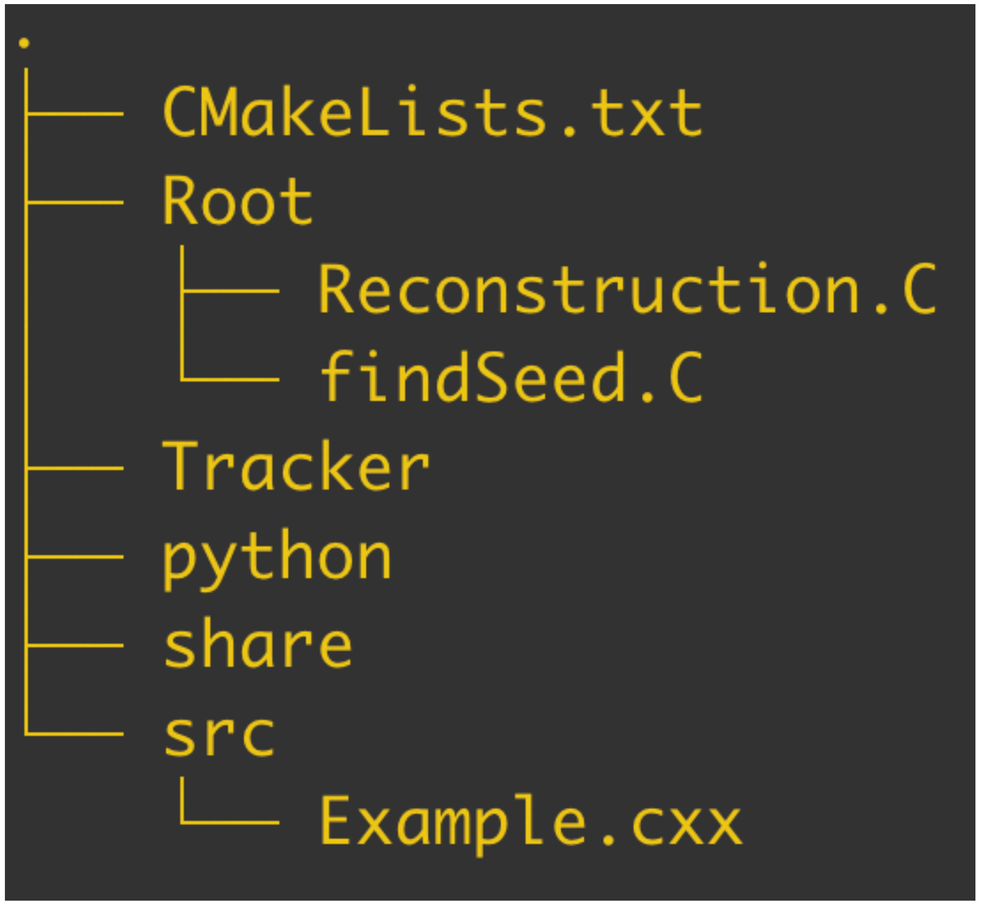
Present status (I)

- Some infrastructure from Tracker FastSim setup
  - Cleaning up process ongoing.
  - Class renaming/restructuring for our purpose.
  - Parameters will be used from ConfigReader module
  - Transition from makefile to CMake - completed.



Present status (II)

- A few new classes were written: sitting on my laptop
  - **Magnets, MakeHists, Particle** class
  - **ConfigReader:** reading from external config file
- They are for the /Common packages
- **Collaboration welcome:** Will put on the git after a few more skeleton files are prepared.
- More detailed description of classes, inheritances can be found here.



Summary

- **Theia**: the software structure should be in place for analysis of the collected data.
- Need to be as **modular** as possible.
  - Good for writing, debugging, understanding.
- Think of the future necessity: plan ahead.
- **It's a big project: collaboration is welcome.**
  - The skeleton setup will live on git very soon.