Praca nad projektem TPCReco i reorganizacja kodu

Jakub Żak Wojciech Kos Agata Bijak
Michał Pasiowiec

Opiekun: dr hab. Artur Kalinowski

1 Wstęp

Program *TPCReco* służy do analizy danych z detektora *ELITPC*, który bada reakcje fotonuklearne metodą monochromatycznych promieni gamma. Został on napisany w języku `C++` i jest aktywnie rozwijany na Wydziale Fizyki UW. Czerpie w dużej mierze z genewskiej biblioteki do analizy danych *ROOT*[1].

2 Cel projektu

Celem Projektu jest reorganizacja istniejącego kodu do analizy danych z eksperymentu Warsaw Active target TPC (dawniej ELITPC). Projekt podizelony został na trzy części:

- Stworzenie wzoru testów funkcjonalnych klass aplikacji,
- Reorganizacja konfiguracji aplikacji,
- Reorganizacja klasy EventSource,

Projekt wykonano przy pomocy stemu wersjonowania kodu (*GIT*[2]) oraz oprogramowania służącego do konteneryzacji projektu (*Docker*[3])

3 Kod projektu

Repozytorium projektu znajduje się pod adresem `https://github.com/Kinexity/TPCReco`. Tam wprowadzano wszelkie zmiany za pomocą programu *GIT*, które na koniec włączono do głównego repozytorium aplikacji pod adresem, `https://github.com/akalinow/TPCReco`. Uczestnicy pracowali na następujących gałęziach projektu:

1. `develJZ`
2. `Zmiiokmoesker`.

4 Wprowadzone zmiany

Poniżej wymieniono główne zmiany w kodzie:

4.1 Test funkcjonalny kalsy `EventTPC.cpp`

W pliku testu `EventTPC_test.cpp` znajdują się funkcje typu void, które przyjmują następujące argumenty:

- std::shared_ptr EventPtr = myEventSource->loadDataFile(dataFileName)->getCurrentEvent()
- std::map<std::string, double> Test_Reference - zdefiniowany w dataEventTPC.h
- std::map<std::string, std::string> Test_Reference_Titles - zdefiniowany w dataEventTPC.h

Funkcje testujące to:

- get1DProjection_Titles_Test
- get2DProjection_Titles_Test
- get1DProjection_Test
- get2DProjection_Test
- GetTotalCharge_Test
- GetMaxCharge_Test
- GetMaxChargePos_Test
- GetSignalRange_Test
- GetMultiplicity_Test

Powyrzsze funkcje porównuje wyniki wywołanych metod klasy EventTPC z danymi porównawczymi, które znajdują się w pliku dataEventTPC.h. Dane

porównawcze przechowywane są w formanie map, gdzie kuczem są wywoływane metody z argumentami, w formacie std::string.

4.2 Klasa ConfigManager

Powstała klasa służąca do wczytywania parametrów z linii poleceń oraz plików konfiguracyjnych, a następnie konfiguracji obiektu boost::property_tree::ptree i zwracania obiektu skonfigurowanego. Klasa zawiera następujące metody:

- boost::property_tree::ptree getConfig(int, char**) - metoda zwracająca skonfigurowane drzewo
- `boost::program_options::variables_map parseCmdLineArgs(int,char)` - metoda do parsowania argumentów z linii poleceń
- `boost::program_options::options_description parseAllowedArgs(std::string)` - metoda do parsowania listy dozwolonych argumentów linii poleceń z pliku konfiguracyjnego

Sposób formatowania plików konfiguracyjnych zawierających listę dozwolonych argumentów linii poleceń został zawarty w TPCReco/Utilities/config/README.md. Przykładowy plik wygląda następująco:

```
{   "Options":
    {
        "pressure": {
            "type": "float",
            "defaultValue": 60,
            "description": "float - CO2 pressure [mbar]",
            "isRequired": true
        },
        "no-type": {
            "type": "bool",
            "defaultValue": false,
            "description": "Skip comparing event type",
            "isRequired": false
        },
        "files": {
            "type": "std::vector<std::string>",
            "defaultValue": "PLACEHOLDER_FOR_VALUE",
            "description": "strings - list of files to browse. Mutually exclusive with 'dire
            "isRequired": false
        }
    }
}
```

4.3 Reorganizacja klasy EventSource

Wyseparowano system rozpoznawania typu danych z klasy MainFrame. Dodano namespace EventSourceFactory z funkcją EventSourceFactory::makeEventSourceObject która przyjmuje przez referencję obiekt typu boost::property_tree::ptree z konfiguracją i zwraca std::shared_ptr¡EventSourceBase¿ będący wskaźnikiem na obiekt klasy pochodnej do EventSourceBase. Typ tego obiektu zależy od nazw plików z danymi zawartych w konfiguracji. W trakcie tworzenia tego obiektu konfiguracja zostaje rozszerzona o zmienną zapisującą typ obiektu ("eventType") w formacie enum event_type (wartości odpowiadające nazwom klas EventSource[...]) oraz o flagę ("onlineFlag"), która mówi czy pliki mają być na żywo ładowane z folderu (aplikuje się tylko do plików typu GRAW). Zmieniono obsługę systemu plików z biblioteki ROOT na bibliotekę boost::filesystem. Funkcja jest w stanie rozpoznać i załadować następujące formaty plików:

- pliki w formacie .root wygenerowane przez bibliotekę ROOT - klasa EventSourceROOT
- pliki z symulacji Monte Carlo - klasa EventSourceMC
- pliki w formacie GRAW (zarówno pojedyncze, mnogie jak i z możliwość aktywnego ładowania nowo dodanych plików) - klasy EventSourceGRAW oraz EventSourceMultiGRAW

W przypadku nieprawidłowej konfiguracji zostaje zwrócony pusty wskaźnik.

5 Podsumowanie

Projekt wykonano podczas semestru zimowego orza letniego roku akademickiego 2022/2023 na Wydziale Fizyki UW. W wyniku prac, dodano test kalsy EventTPC, zmienionu został sposób konfiguracji aplikacji oraz poprawione zostało rozpoznawanie obiektów EventSourceBase. Dzięki temu, uczestnicy zrealizowali ogólne założenia projektu.

Literatura

[1] The ROOT Project, https://root.cern.ch/

[2] GIT - The Stupid Content Manager, https://git-scm.com/

[3] Docker, https://www.docker.com/

[4] JSON syntax - https://en.wikipedia.org/wiki/JSON#Syntax