ПРАВИТЕЛЬСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ
«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
(СПбГУ)

Кафедра вычислительной физики

Направление «Прикладные математика и физика»

Разработка интерфейса TShield между транспортным кодом SHIELD и программным комплексом MpdRoot

Магистерская диссертация студента

Соснова Дмитрия Евгеньевича

Научный руководитель:

к.ф.-м.н., доц.. **Немнюгин С.А.**

Рецензент:

к.ф.-м.н., н.с.. **Мерц С.П.**

Санкт-Петербург

2015

Содержание

Введение **3**

1 Описание используемых программных систем **6**

1.1 Описание системы Root 6

1.2 Описание системы FairRoot 6

1.3 Описание системы MpdRoot 7

1.4 Описание системы SHIELD 7

1.5 Первоначальное состояние проекта TShield 8

2 Создание библиотек SHIELD и HADGEN **9**

2.1 Изменение системы компиляции 9

2.2 Процедура создания динамической библиотеки 9

2.3 Создание динамической библиотеки HADGEN 10

2.4 Создание динамической библиотеки SHIELD 10

2.4.1 Входной файл for023.dat 11

2.4.2 Входной файл for022.dat 11

2.4.3 Входной файл pasin.dat 13

2.5 Выходные файлы системы SHIELD 16

2.6 Выводы по главе 2 17

3 Интерфейс THadgen **18**

3.1 Классы генераторов событий систем Root и FairRoot 18

3.2 Класс THadgen 18

3.3 Класс MpdGeneralGenerator 19

3.4 Использование класса THadgen в системе MpdRoot 20

3.5 Выводы по главе 3 20

4 Интерфейс TShield **21**

4.1 Класс TShield 21

4.2 Система геометрических данных системы Root 21

4.3 Преобразование геометрических данных 21

4.4 Выходные данные системы TShield 25

4.5 Выводы по главе 4 25

5 Использование системы TShield в системе MpdRoot **26**

5.1 Интерфейсы к классу TShield 26

5.2 Класс TShieldGenerator 26

5.3 Класс TShieldMC 26

5.4 Выводы по главе 5 27

6 Методы проведения апробации **28**

Выводы **29**

Список литературы **30**

Приложения **32**

1 Пример входного файла for023.dat системы SHIELD 32

2 Пример входного файла for022.dat системы SHIELD 33

3 Пример входного файла pasin.dat системы SHIELD 34

4 Листинг части файла исходного кода shield_geo.c 35

5 Листинг части файла исходного кода THadgen.cxx 41

6 Листинг части файла исходного кода MpdGeneralGenerator.cxx 44

7 Файл изменений THadgen_runMC.patch 45

8 Листинг файла исходного кода TShieldGeometryBool.cxx 46

9 Листинг части файла исходного кода TShieldGeometryConvert.cxx 48

10 Файл изменений TShield_MpdRoot.patch 54

Введение

Изучение экстремально плотной и горячей ядерной материи является актуальной задачей современной физики. Особый интерес связан с изучением нового состояния материи, позволяющим пролить свет на наиболее фундаментальные проблемы физики - кварк-глюонной плазмы (КГП), существование которой было предсказано современной теорией сильного взаимодействия [1]. Особое внимание уделялось изучению свойств КГП при энергиях столкновений 20-100 ГэВ на нуклон [2, 3].

Экспериментально доказано существование нового состояния вещества [1]. Вместе с тем недостаточно исследованы свойства КГП при энергиях от 2 до 10 ГэВ. Для получения подобных экспериментальных данных проектируются и строятся несколько ускорителей, в том числе и ускорительный комплекс NICA (Nuclotron-based Ion Collider fAcility) на базе Нуклотрона, рассчитанный на изучение столкновений ионов с энергиями до 11ГэВ и располагающийся в Объединённом институте ядерных исследований (ОИЯИ) в г. Дубна [4]. Схема ускорительного комплекса представлена на рисунке 1.

Рис. 1 – Схема ускорительного комплекса NICA

На коллайдере предусмотрена возможность размещения двух детекторов для проведения двух экспериментов одновременно. Один из детекторов - Многоцелевой детектор (Multi-Purpose Detector, MPD) планируется для изучения свойств горячей и плотной

ядерной материи, образованной при соударении тяжёлых ионов высоких энергий [5].

Для компьютерной обработки экспериментальных результатов с помощью программных пакетов Root [6] и FairRoot [7], разрабатываемых в научных центрах CERN [8] и GSI (FAIR) [9], в ОИЯИ ведётся разработка программного комплекса MpdRoot [10]. Для апробации алгоритмов компьютерной обработки экспериментальных данных необходимо проведение моделирования столкновения в детекторе, также производимое при помощи системы MpdRoot с использованием программных генераторов событий.

Отдельным по важности вопросом на экспериментах подобного уровня стоит контроль за безопасностью работников, в частности контроль радиационной обстановки, производимый как путём экспериментальных замеров, так и методами математического моделирования в случае планируемых экспериментов. Одним из вариантов увеличения качества моделирования является использование проверенной системы транспорта частиц. На текущий момент системой, используемой для моделирования радиационной защиты является система FLUKA [11], качество моделирования которой в случае использования для расчётов защиты от ионизирующего излучения уступают аналогичным системам моделирования прохождения частиц через вещество [12, 13], например проекту SHIELD[14], разрабатываемому в Институте Ядерных Исследований Российской Академии наук (ИЯИ РАН) в г. Москва[15].

В то же время использование системы SHIELD осложняется ввиду кардинального отличия системы от используемых в ОИЯИ систем, что не позволяло использовать пакет SHIELD в проекте MpdRoot. Потому было решено создать интерфейс к системе SHIELD и генератору событий системы SHIELD HADGEN для возможности использования системы SHIELD в составе комплекса MpdRoot. Также ввиду специализации системы SHIELD для расчётов, связанных с адронной физикой, и интересом Лаборатории Радиационной Биологии ОИЯИ, было решено предусмотреть возможность независимой от системы MpdRoot компиляции.

Таким образом, целью работы являлось разработка интерфейса между генератором HADGEN и программным комплексом MpdRoot для использования в качестве генератора столкновений и интерфейса между транспортным кодом SHIELD и программным комплексом MpdRoot для расчёта радиационной обстановки в зоне детектора.

Ввиду использования в системах MpdRoot и SHIELD разных языков программирования с невозможностью прямой работы между ними (C++ и FORTRAN), и минимизации зависимостей системы TShield от программных комплексов, для достижения поставленной цели решались следующие задачи:

- Сборка и отладка проекта TShield.
- Создание динамической библиотеки на основе исходных кодов приложения HADGEN.
- Создание динамической библиотеки на основе исходных кодов приложения SHIELD.
- Создание высокоуровневого интерфейса генератора событий THadgen.
- Создание высокоуровневого интерфейса транспортного кода TShield.
- Подключение систем THadgen и TShield к системе MpdRoot.

1 Описание используемых программных систем

1.1 Описание системы Root

Программный комплекс Root - набор пакетов и инструментов, разработанный в Европейском Центре Ядерных Исследований (ЦЕРН) на языке C++, включающий в себя следующие библиотеки [16, 17]:

- Библиотека встроенного интерпретатора C++ Cling libCling.
- Библиотека набора базовых классов libCore.
- Библиотека интерфейсных классов генераторов событий libEG.
- Библиотека для работы с матрицами и векторами libMatrix.
- Библиотека базовых классов работы с математическими и физическими векторами libMathCore.
- Библиотека командной строки системы ROOT libRint.
- Библиотека контейнера объектов TTree libTree.
- и другие.

Благодаря специализации комплекса для работы в области физики высоких энергий, система Root также используется в качестве платформы для разработки программного обеспечения для различных проектов в данной области.

1.2 Описание системы FairRoot

Программно-аппаратная платформа FairRoot - программный комплекс, основанный на системе Root и написанный на языке высокого уровня C++, разрабатывается в Институте Тяжёлых Ионов (FAIR) и предоставляет базовые классы для возможности создания программного обеспечения для моделирования и анализа физических процессов внутри детектора элементарных частиц. Программный комплекс включает в себя 413 классов и предоставляет обобщённые интерфейсы для подключения внешних библиотек.

Благодаря удобству использования системы, FairRoot является основой для разработки программного обеспечения детекторов различных лабораторий физики высокой энергии.

1.3 Описание системы MpdRoot

Программный комплекс MpdRoot - система, разрабатываемая на основе систем Root и FairRoot для моделирования и расчётов, связанных с многоцелевым детектором коллайдера NICA, планируемого к запуску в 2018 году.

Моделирование событий в детекторе в системе MpdRoot происходит следующим образом:

- Первичным генератором событий создаётся набор частиц, отвечающих первичному столкновению пучков частиц;
- Системой моделирования переноса частиц через вещество рассчитывается траектория движения, столкновения с другими частицами и системами детектора;
- Для моделирования взаимодействий, получаемых при столкновении переносимых транспортной системой частиц используется встроенный в транспортный код генератор событий.

Необходимо отметить, что с начала разработки в составе системы MpdRoot содержится генератор событий *FairShieldGenerator*, служащий для работы с данными системы SHIELD и обладающий следующими недостатками:

- Работа генератора основана на специальным образом подготовленных результатах, полученных при независимом запуске системы SHIELD.
- Генератор не допускает изменение входных параметров из системы MpdRoot
- Результаты работы системы SHIELD используются только в качестве первичного генератора событий, не позволяя использовать систему SHIELD в качестве системы моделирования переноса частиц.
- Невозможность независимого использования встроенного в SHIELD генератора HADGEN.

1.4 Описание системы SHIELD

Первоначальная версия системы SHIELD была разработана в ОИЯИ в 1967-72 годах на языке FORTRAN [18], после чего неоднократно модернизировалась в ИЯИ РАН.

В 1997 году в коде SHIELD был реализован алгоритм моделирования переноса атомных ядер произвольного заряда и массы , что позволило в дальнейшем использовать код, в частности, в задачах, связанных с проектирование ускорителей тяжёлых ионов.

Одной из особенностей системы SHIELD, накладывающей ограничения на спектр задач, решаемых с использованием пакета, является отсутствие в моделировании возможности задания и использования электромагнитных полей, что автоматически исключает возможность

использования SHIELD для моделирования процессов в активной зоне детектора.

Также транспортный код SHIELD включает в себя генератор событий HADGEN (HADron GENerator), также иногда называемый MSDM-генератором (Multi Stage Dynamical Model), либо как генератор событий SHIELD.

Системы HADGEN и SHIELD реализованы в виде прикладного программного обеспечения, исходный код систем основывается на коде, написанном на языке FORTRAN согласно стандарту FORTRAN 77 с добавлением некоторых нововведений из стандарта FORTRAN 90. Для сборки проекта использовался компилятор g77 [19].

1.5 Первоначальное состояние проекта TShield

На начало создания системы TShield над кодом систем HADGEN и SHIELD была проделана немалая работа: была создана динамическая библиотека SHIELD на основе кода проекта HADGEN, были созданы заготовки динамической библиотеки SHIELD и классов *THadgen* и *TShield*.

2 Создание библиотек SHIELD и HADGEN

2.1 Изменение системы компиляции

Ввиду использования для компиляции проектов HADGEN и SHIELD компилятора g77, поддержка и разработка которого была прекращена в 2008 году, невозможности сборки компилятора g77 на современных операционных системах и необходимости подключения проекта в качестве части более крупного проекта, использующего систему автоматизации сборки CMake, возникла необходимость в изменении компилятора на современный компилятор gfortran [20], входящий в стандартную поставку популярного набора компиляторов gcc [21] и создании набора файлов макросов для системы CMake [22].

Для обеспечения возможности независимого использования проекта была создана независимая система макросов языка системы автоматизации сборки CMake с возможностью автоматического включения в проект MpdRoot. Для определения существования родительского проекта (проекта, в рамках которого выполняется сборка) была использована проверка на существование переменной *PROJECT_NAME*, необходимой системе CMake. В случае компиляции проекта в качестве независимой системы для создания библиотек с использованием программы rootcint системы Root используется дополнительный макрос *ROOT_GENERATE_DICTIONARY*, находящийся в файле *cmake/TShieldConfig.cmake*.

Ввиду использования системой CMake раздельной компиляции отдельных файлов исходного кода в объектные файлы и зависимости в языке FORTRAN ожидаемого выходного значения неизвестной функции от названия, для корректной сборки и работы были переименованы часть функций, в числе которых и функция создания случайных чисел *RAN* проекта HADGEN, заменяющая стандартную функцию *RAN*, которая была переименована в функцию *LRAN*.

2.2 Процедура создания динамической библиотеки

Основной проблемой создания динамической библиотеки из исходного кода приложения, написанного на языке FORTRAN, является невозможность доступа к функциям, подпрограммам и переменным языка FORTRAN извне библиотеки. Таким образом, для доступа к данным и функциям необходимо создание дополнительных интерфейсов на языке C, с учётом следующих особенностей:

- Функции, подпрограммы и переменные языка FORTRAN при компиляции получают символ подчёркивания к имени.
- По причине того, что язык FORTRAN является регистронезависимым, при компиляции

имена функций, подпрограмм и переменных производных типов переводятся в нижний регистр (записываются строчными символами), а запись переменных стандартных типов производится прописными символами.

- Доступ к функциям и подпрограммам языка FORTRAN из языка C может быть предоставлен путём указания программе-компоновщику об их существовании созданием прототипа функции языка C с использованием ключевого слова *extern*.
- Для доступа к переменным производных типов языка FORTRAN из языка C необходимо задание структуры (объекта типа *struct*) языка C, аналогичной по содержанию исходной переменной, и указания программе-компоновщику о её внешнем задании использованием ключевого слова *extern*.

Также для доступа к функциям динамической библиотеки необходимо в заголовочный файл, используемый для подключения библиотеки, добавить указание программе-компоновщику о внешнем задании функции указанием ключевого слова *extern* совместно со строковым литералом *"C"*.

2.3 Создание динамической библиотеки HADGEN

Хотя динамическая библиотека и была создана на момент начала работы, для работы системы SHIELD и отключению дополнительных выходных файлов было решено закомментировать 277 вызовов функции записи в текстовые файлы и командную строку. Также, поскольку для идентификации частиц проект HADGEN использует систему нумерации частиц, предложенную коллаборацией Particle Data Group (PDG-коды) [23, 24], было решено добавить набор функций *hadgen_get_particle_type* и *hadgen_get_nuclei_parameters_by_pdg_code* для определения параметров частиц по переданному PDG-коду.

2.4 Создание динамической библиотеки SHIELD

Как было указано в параграфе 2.2, для создания динамической библиотеки SHIELD из исходного кода независимого приложения на языке FORTRAN необходимо создание промежуточного интерфейса на языке C, включающего в себя интерфейс для вызова функции, аналогичной главной программе исходного приложения, и функции для доступа к внутренним данным программы для возможности прямого ввода/вывода данных. Главной программой приложения SHIELD являлась программа *SHIELD*, находящаяся в исходном файле *Shield_so.f*. В программе производились следующие действия:

1. Открывались входные и выходные файлы.
2. Производилась загрузка данных из входного файла *for023.dat*.

3. Вызывались функции ввода параметров используемых химических веществ и описания геометрии эксперимента из файлов *for022.dat* и *pasin.dat* соответственно.

4. Производилась инициализация не определенных пользователем величин..

5. В цикле производился запуск функции *GENTRE*, отвечающей за запуск отдельнойй частицы из пучка частиц..

6. Вызывалась функция *SAVES* записи данных в выходные файлы.

В качестве функции динамической библиотеки, заменяющей главную программу *SHIELD* была создана функция *shield_ run*, выполняющая пункты 4 и 5 действий программы *SHIELD*. Для действий ввода-вывода данных были написаны функции, заменяющие работу с файлами на работу с методами доступа ко внутренним данным библиотеки.

2.4.1 Входной файл for023.dat

Данные во входном файле for023.dat [25] отвечают за множество различных параметров моделирования. Основными являются:

- BEAMDIR – Параметры (углы) направления пучка частиц (°).
- BEAMPOS – Позиция начала пучка частиц (см).
- HIPROJ – Заряд и число нуклонов в частице в случае использования тяжёлых ионов.
- JPART0 – Тип частиц в пучке. Возможные значения описаны в таблице 1.
- NSTAT – Число частиц в пучке.
- RNDSEED – Значение затравки генератора случайных величин.
- TMAX0 – Начальная энергия частицы, MeV/нуклон.

Пример входного файла for023.dat представлен в приложении 1.

По причине того, что при обработке входных параметров файла for023.dat в процедуре *SHIELD* изменению подвергались отдельные переменные стандартных типов данных, для ввода параметров в библиотеку был написан набор функций для доступа к отдельным переменным, перечисленный в таблице 2 и описанный в файле исходного кода *shield_ setget.c*.

2.4.2 Входной файл for022.dat

Входной файл системы SHIELD for022.dat [25], описывающий химический состав материалов (количеством не более 24), используемых для моделирования, описывается следующим образом:

- Для каждого из веществ указывается тип (чистое вещество, химическая смесь, смесь различных изотопов или химическая смесь с содержанием изотопов) и плотность.

Таблица 1 – Возможные типы частиц в пучке

JPART	Частица	JPART	Частица
-1	Любая частица	13	Электрон
1	Нейтрон	14	Позитрон
2	Протон	15	Мюон μ^-
3	Пион π^-	16	Мюон μ^-
4	Пион π^+	17	Электронное нейтрино
5	Пион π^0	18	Электронное антинейтрино
6	Антинейтрон	19	Мюонное нейтрино
7	Антипротон	20	Мюонное антинейтрино
8	Каон k^-	21	Дейтрон
9	Каон k^+	22	Тритий
10	Каон k^0	23	He-3
11	Каон k	24	He-4
12	Гамма лучи	25	Тяжёлый ион

Таблица 2 – Функции доступа к переменным, аналогичным параметрам файла for023.dat

Параметр входного файла	Функция для задания значения (сеттер)	Функция получения значения (геттер)
BEAMDIR	*shield_set_rundirection*	отсутствует
BEAMPOS	*shield_set_runpoint*	отсутствует
HIPROJ	*shield_set_aproj*	*shield_get_aproj*
	shield_set_zproj	*shield_get_zproj*
JPART0	*shield_set_incidentparticle*	*shield_get_incidentparticle*
NSTAT	*shield_set_statistics*	*shield_get_statistics*
RNDSEED	*shield_set_randomseed*	*shield_get_randomseed*
TMAX0	*shield_set_energy*	*shield_get_energy*

- Последовательно для каждого из химических элементов вещества указывается номер изотопа и концентрация в составе вещества.
- Опционально могут указываться иные свойства: плотность, массовое число, зарядовое

число, плотность чистого вещества и энергию ионизации.

Пример входного файла for022.dat представлен в приложении 2.

Функция *INMACR* исходного кода программы SHIELD обрабатывает данные следующим образом:

- В двумерный массив *NELEMD* структуры *MACRIN* записываются тип вещества и количество содержащихся элементов.
- В массив *RHOMED* структуры *MACRIN* записывается плотность вещества.
- В трехмерный массив *ELMIX* структуры *MACRIN* для каждого химического элемента записываются заданные параметры. При отсутствии значений опциональных параметров свойства элемента берутся из заранее заданных в исходном коде системы SHIELD массивов.

Для удобства задания химических веществ получение данных и запись параметров в массивы структуры *MACRIN* была разнесена в различные функции - функции *shield_add_medium* и *shield_send_medium*.

Функция *shield_add_medium* принимает на вход структуру типа *MediumData*, описанную в файле *shield_media.h* и отвечающую описанию химического вещества, записывает вещество в глобальную переменную *SMedium* с областью видимости файла *shield_media.c* в случае первого задания данного вещества и возвращает номер вещества в структуре.

Функция *shield_send_medium* производит запись данных о химических веществах, сохранённые в переменной *SMedium*, в массивы структуры *MACRIN* аналогично функции *INMACR*.

2.4.3 Входной файл pasin.dat

Для описания геометрии эксперимента система SHIELD использует подход комбинаторной геометрии: геометрические данные определяются некоторым количеством сложных объектов, называемых зонами, составленными применением логических операций к простым объектам, называемым телами; химический состав зоны определяется индексом используемого вещества. При описании зоны операция отрицания имеет наивысший приоритет из используемых, а дизъюнкция - наименьший. Для записи зоны в файле pasin.dat для обозначения операций конъюнкции, отрицания и дизъюнкции использовались символы '+', '−', 'OR' соответственно. Пример применения логических операций к телам указан на рисунке 2, описания различных типов тел представлены в таблице 3. Пример входного файла pasin.dat представлен в приложении 3. Также при задании геометрического описания имеется необходимость в однозначном определении зоны по координатам точки, то есть отсутствие пересечений зон.

Подпрограмма *GEOINI*, отвечающая за считывание геометрических параметров моделирования производила загрузку считанных параметров следующим образом:

Рис. 2 – Пример применения логических операций к телам 1 (круг) и 2 (квадрат). Символами "+ "– "OR"обозначены конъюнкция, отрицание и дизъюнкция соответственно.

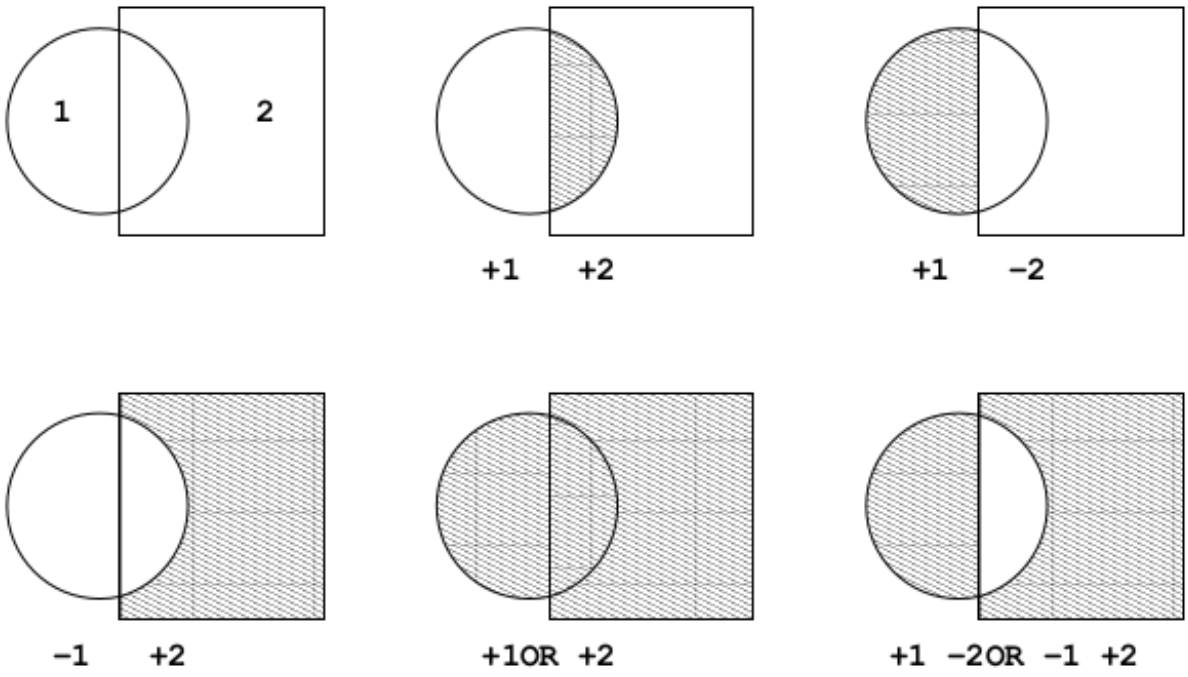


Таблица 3 – Описание тел системы SHIELD

Название SHIELD	Описание	Число параметров
SPH	Сфера	4
WED	Клин	12
ARB	Произвольный выпуклый восьмигранник	24
BOX	Прямоугольный параллелепипед	12
RCC	Цилиндр	7
REC	Эллиптический цилиндр	12
TRC	Прямой круговой конус	8
ELL	Эллипс	12

- Каждое тело переводилось в заранее заданный набор из плоскостей, определённых в системе SHIELD, при этом каждая плоскость определяется своим индексом и параметрами
- Полученный набор данных записывался последовательно в массив *BODYDB* структуры *GDATA2*.
- Индекс начала описания каждого тела в массиве *BODYDB* записывался в массив

BODYDA структуры *GDATA2*.

- Зоны описывались числом используемых тел, индексом химического вещества, свойством зоны (присутствие/отсутствие операций дизъюнкция и отрицание) и набором целочисленных значений, получаемых из определения зоны заменой дизъюнкции числом 0 и записывались последовательно в массив *ZONEDB* структуры *GDATA3*.
- Индекс начала описания каждой зоны в массиве *ZONEDB* записывался в массив *ZONEDA* структуры *GDATA3*.
- Общее количество тел и зон записывалось в переменные *NUMBOD* и *NUMZON* структуры *GDATA0*.

Для удобства использования системы представления геометрических данных были созданы следующие структуры и функции:

- Структура *SGeoBody* представляет описание тела и включает в себя тип тела и целочисленный массив параметров.
- Структура *SGeoZone* представляет описание зоны и включает в себя индекс химического вещества, приписанного зоне, целочисленный массив, используемый для определения зоны и число используемых в определении тел.
- Функция *shield_ add_ body* принимает на вход тип тела и указатель на массив параметров, выполняет перевод тела в набор плоскостей, записывает тело в массив *bodies* глобальной структуры *SGeometry* с областью видимости файла *shield_ geo.c* в случае первого задания данного тела и возвращает номер тела в структуре.
- Функция *shield_ add_ zone* принимает на вход определение зоны, указатель на массив структур *SGeoBody*, структуру типа *MediumData*, отвечающую химическому веществу и число используемых тел, проверяет корректность задания тел и химического вещества. В случае отсутствия уже созданной аналогичной зоны создаёт и записывает данные зоны в массив *zones* глобальной структуры *SGeometry* с областью видимости файла *shield_ geo.c*.
- Функция *shield_ init_ geometry* производит запись данных из структуры *SGeometry* в типы данных системы SHIELD аналогично подпрограмме *GEOINI*.
- Функция *shield_ clean_ geometry* производит очистку структуры *SGeometry* и вызывает функцию *shield_ clean_ medium*, очищающую структуру, содержащую данные об используемых химических веществах.

Для возможности использования системы SHIELD на сложной геометрии без опасения переполнения массивов структур *GDATA0* и *GDATA2* была добавлена функция обратного вызова *shield_ set_ gnextz_ callback*, позволяющая использовать внешние функции для определения зоны, через которую осуществляется транспорт частицы. Благодаря возможности вызывать функции работы с геометрией извне, можно сильно сократить число одновременно используе-

мых системой SHIELD зон, тел и химических веществ, динамически изменяя их для каждого нового положения частицы. Однако, в связи с возможностью получения расхождения в расчётах системы SHIELD и систем, использующих динамическую библиотеку, для предотвращения потери частицы было решено оставить совместную систему определения текущей зоны - после вызова внешней функции осуществляется вызов функции определения зоны транспорта частицы системы SHIELD *GCURZL* для контроля потери транспортируемой частицы.

Также в ходе работы была обнаружена и исправлена ошибка в функции *GCURZL*, приводящая к некорректному определению зоны в случае задания зоны с применением операции дизъюнкции и попадания частицы в первый набор тел, связанных операцией конъюнкция.

Часть текста файла исходного кода *shield_ geo.c*, содержащего код работы с геометрическим представлением данных системы SHIELD, представлена в приложении 4. Текст остальных исходных файлов доступен на электронном приложении.

2.5 Выходные файлы системы SHIELD

В системе SHIELD присутствует 11 выходных файлов различного назначения, представленные в таблице 4. Как указано в главе 2.2, для корректной работы динамической библиотеки необходимо изменение системы вывода данных, а именно: отключения вывода в выходные файлы и создание функций для доступа к данным библиотеки.

Таблица 4 – Описание выходных файлов системы SHIELD

Имя файла	Тип	Описание
for017	Текстовый	Дублирование входных геометрических данных
for024	Текстовый	Входные/выходные данные запуска
for025	Текстовый	Диагностический файл
for026	Текстовый	Информация о частицах и материалах
for027	Текстовый	Информация о пересечении с химическими элементами
for028	Текстовый	Может содержать информацию о вторичных нейтронах с энергией менее 14.5 МэВ
for029	Бинарный	Информация о длине трека частицы
for030	Бинарный	Информация о фрагментах частицы
for031	Бинарный	Информация о распределении энергии
for032	Бинарный	Информация о распределении энергии

Поскольку большая часть выходных файлов использовалась только для вывода данных,

для отключения вывода в выходные файлы был отключён вызов подпрограммы для вывода выходных данных *PRINTR* и закомментирован 281 вызов стандартной функции языка FORTRAN *WRITE*, используемой для вывода информации в файл.

В то же время, в случае выходного файла for017 по причине его использования для тестирования входных данных, было решено осуществить подмену используемого файла "виртуальным файлом переменной типа строка с возможностью чтения и записи стандартными методами, используемыми для работами с файлами.

Для передачи выходных файлов из динамической библиотеки было решено использовать функцию *TREEDEC*, используемую для сбора информации о текущей частице. Для этого была создана следующая система структур и функций:

- Структура *shield_tree_node* представляет информацию о частице, включающую в себя тип частицы, PDG-код частицы, местоположение, энергию, направление движения.
- Функция *shield_set_tree_callback*, принимающая указатель на функцию, принимающую переменную типа *shield_tree_node*, сохраняет указатель на функцию для последующего использования.
- Функция *shield_tree_new_branch* заменяет функцию *TREEDEC* системы SHIELD, по данным системы SHIELD конструирует переменную типа *shield_tree_node* и передаёт в функцию, полученную функцией *shield_set_tree_callback*.

Таким образом, функция, переданная в динамическую библиотеку с использованием функции *shield_set_tree_callback*, будет автоматически вызываться при выполнении моделирования.

2.6 Выводы по главе 2

- Была создана система сборки, позволяющая использовать проект как независимо, так и в виде части проекта MpdRoot.
- Была создана динамическая библиотека SHIELD на основе исходных данных независимого приложения на языке FORTRAN.

3 Интерфейс THadgen

3.1 Классы генераторов событий систем Root и FairRoot

Для создания генераторов событий в системах Root и FairRoot имеются независимые базовые классы генераторов - классы *TGenerator* и *FairGenerator* соответственно, сравнение которых представлен на таблице 5. В системе MpdRoot базовым классом генераторов событий является класс *FairGenerator* системы FairRoot

Таблица 5 – Сравнение TGenerator и FairGenerator

Система	Root	FairRoot
		MpdRoot
Класс	*TGenerator*	*FairGenerator*
Родительский класс	*TNamed*	*TNamed*
Внутреннее хранилище	*TObject**	Отсутствует
Выходные данные	*TClonesArray**	*FairPrimaryGenerator**

3.2 Класс THadgen

Для возможности независимой от систем MpdRoot и FairRoot сборки и компиляции *THadgen* было решено класс генератора событий *THadgen* наследовать от класса генератора событий системы Root *TGenerator*. Таким образом, реализация класса *THadgen* включает в себя:

- Конструктор *THadgen*, создающий объект *fParticles* типа *TClonesArray* из системы Root, используемый для хранения выходных данных, и запускающий установку стандартных начальных значений системы HADGEN.
- Методы для доступа к следующим методам библиотеки HADGEN:
  - *hadgen_set_randomseed*
  - *hadgen_set_incidentparticle*
  - *hadgen_set_nuclid*
  - *hadgen_set_energy*
  - *hadgen_set_statisticsnum*
  - *hadgen_set_aprojectile*
  - *hadgen_set_zprojectile*
  - и аналогичным методам получения данных.

- Методы *SetParticleFromPdgCode* для задания параметров частиц используя систему кодов PDG.
- Метод *GenerateEvent*, служащий для запуска инициализации системы HADGEN функцией *hadgen_ initialize*, выполнения генерации методом *hadgen_ generate* и завершение работы системы HADGEN по запуску системы *hadgen_ terminate*.
- Метод *CopyToInternalArray*, получающий выходные данные системы HADGEN и записывающий их в *fParticles*.
- Методы *ImportParticles*, служащие для вывода данных в объекты типа *TClonesArray*.

Часть текста файла исходного кода *THadgen.cxx*, содержащего код класса *THadgen* представлена в приложении 5. Полный текст класса представлен в электронном приложении.

3.3 Класс MpdGeneralGenerator

Для использования генератора *THadgen* ввиду малого различия между классами генераторов систем Root и FairRoot было решено создать интерфейс между классами *TGenerator FairGenerator*. Для этого был создан класс MpdGeneralGenerator, являющийся наследником класса *FairGenerator* и включающий в себя:

- Указатель на объект типа *TGenerator*.
- Методы *SetGenerator* и *GetGenerator* доступа к объекту типа *TGenerator*.
- Метод *ReadEvent*, запускающий работу генератора, получающий выходные данные и записывающие входные данные в объект типа *FairPrimaryGenerator* стандартным образом.

Таким образом, для использования объектов класса *THadgen* в системе MpdRoot производятся следующие действия:

- Создание объекта класса *THadgen*.
- Задание входных параметров в объект класса *THadgen*.
- Создание объекта класса *MpdGeneralGenerator*.
- Передача объекта класса *THadgen* в объект класса *MpdGeneralGenerator* методом *SetGenerator*.
- При необходимости изменения параметров доступно задание параметров в объекте типа *THadgen*, получаемом методом *GetGenerator*.
- Использование *MpdGeneralGenerator* как стандартного генератора событий системы *MpdRoot*.

Исходные коды метода *ReadEvent* представлены в приложении 6, полный листинг находится в электронном приложении к работе.

3.4 Использование класса THadgen в системе MpdRoot

Для использования класса THadgen в составе системы MpdRoot требуется изменение файла сценария *runMC.C* системы MpdRoot. Листинг файла изменений *THadgen_runMC.patch* представлен в приложении 7.

3.5 Выводы по главе 3

- Был создан интерфейсный класс генератора событий *THadgen*.
- Создан интерфейс *MpdGeneralGenerator* между базовыми классами генераторов событий систем Root и MpdRoot.

4 Интерфейс TShield

4.1 Класс TShield

Для возможности использования системы, основывающейся на динамической библиотеке SHIELD, как отдельно, так и в рамках другого проекта, для доступа к функциям динамической библиотеки SHIELD должен быть написан высокоуровневый класс - интерфейс для системы SHIELD, реализация которого является первоочередной задачей проекта TShield.

Класс TShield представляет собой высокоуровневый интерфейс, позволяющий производить запуск моделирования транспорта частиц.

Для получения входных данных параметров частиц и характеристик пучка в TShield используются методы доступа, вызывающие соответствующие функции, перечисленные в параграфе 2.4.2. В то же время, для получения геометрических данных эксперимента необходимо осуществление перевода данных из формата данных системы Root, используемой в проектах MpdRoot, FairRoot и Root.

4.2 Система геометрических данных системы Root

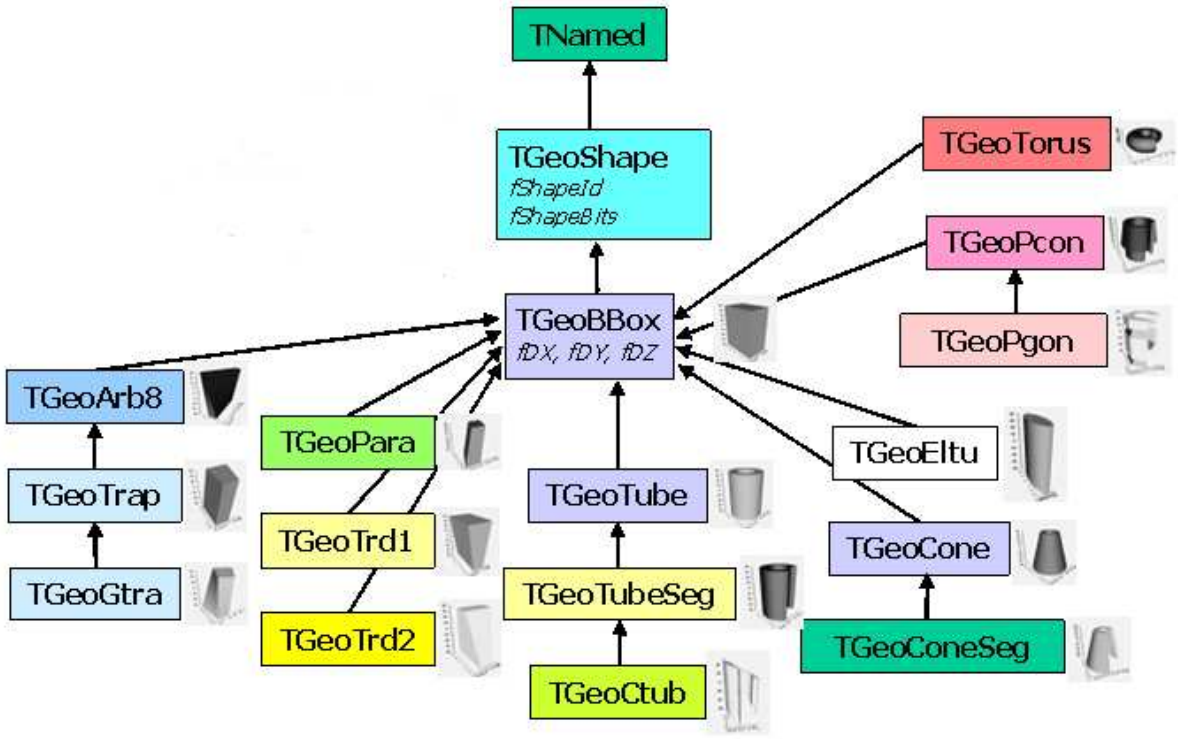
Для описания геометрии экспериментов в системах MpdRoot, FairRoot и Root используется система классов, принятая в Root:

- Класс *TGeoShape* описывает геометрического примитива в понятиях системы Root. Система классов геометрических примитивов представлена на рисунке 3 и в таблице 6.
- Класс *TGeoMedium* описывает химический состав вещества.
- Класс *TGeoMatrix* описывает матрицы преобразования координат.
- Класс *TGeoNode* описывает геометрическое положение объекта относительно родительского и хранит в себе указатели на объекты типов TGeoMatrix и TGeoVolume
- Класс *TGeoVolume* описывает характеристики объекта и имеют указатели на объекты типов TGeoMedium и TGeoShape и хранят список объектов типа TGeoNode, отвечающим дочерним объектам.
- Класс *TGeoNavigator* позволяет перемещаться по полученному дереву объектов.
- Класс *TGeoManager* хранит указатели на экземпляр класса TGeoNode, отвечающему главному объекту, и на экземпляр класса TGeoNavigator.

4.3 Преобразование геометрических данных

Для возможности использования системы TShield необходимо преобразование геометрических данных из формата Root, описанного в параграфе 4.2 в формат данных системы SHIELD,

Рис. 3 – Система классов геометрических примитивов



описанный в параграфе 2.4.3. Для этого в систему TShield был добавлен набор файлов, используемый для работы с геометрией:

- *TShieldGeometry.h* - Заголовочный файл системы преобразования геометрических данных.
- *TShieldGeometryBool.cxx* - Исходный файл, содержащий функции логических операций над телами и зонами.
- *TShieldGeometryConvert.cxx* - Исходный файл, содержащий код преобразования объектов системы Root в зоны в понятиях SHIELD.
- *TShieldGeometry.cxx* - Исходный файл, содержащий главные вызываемые функции.
- *TShieldGeometryOperators.cxx* - Исходный файл, содержащий функции операций над объектами типа TGeoHMatrix.
- *TShieldGeometryPrint.cxx* - Исходный файл, содержащий функции отладочной печати используемых объектов.

Для проведения логических операций с использованием зон и тел в понятиях SHIELD, были созданы функции логических операторов над телами, представленными парой (тип *std::pair*) объекта структурного типа *SGeoBody* и целого числа, отвечающего за операцию, производимую над телом. В качестве типа, отражающего содержание зоны, был использован вектор (тип *std::vector*) векторов типа описания тела (тип *zoneData*).

Таблица 6 – Описание классов геометрических примитивов

Класс геометрических примитивов	Описание класса
TGeoBBox	Прямоугольный параллелепипед
TGeoPara	Параллелепипед
TGeoTrd1	Трапецоид с изменением сторон только вдоль одного измерения
TGeoTrd2	Трапецоид с двумя параллельными основаниями
TGeoTrap	Обобщенный трапецоид
TGeoGtra	Скрученный трапецоид
TGeoArb8	Обобщенный восьмигранник
TGeoTube	Цилиндрическая труба
TGeoTubeSeg	Сегмент цилиндрической трубы
TGeoCtub	Сегмент цилиндрической трубы, обрезанной двумя плоскостями
TGeoEltu	Эллиптическая труба
TGeoHype	Гиперболоид
TGeoCone	Конус
TGeoConeSeg	Сегмент конуса
TGeoSphere	Сфера
TGeoTorus	Тор
TGeoParaboloid	Параболоид
TGeoPcon	Объект, состоящий из соединённых цилиндрических труб/конусов
TGeoPgon	Объект, аналогичный TGeoPcon, но содержащий многоугольники в основании
TGeoCompositeShape	Объект, получаемый применением логических операций над другими примитивами

Поскольку в определениях SHIELD операция дизъюнкции имеет наименьший приоритет, при выполнении функции операции дизъюнкции над зоной выполняется объединение описаний зон, а в случае конъюнкции - дополнение, вычисленное по правилам логики с учётом приори-

тета операций. Текст файла исходного кода *TShieldGeometryBool.cxx*, содержащего описание функций логических операций представлен в приложении 8.

Для выполнения преобразования геометрических данных система TShield использованием функции *shield_set_gnextz_callback* устанавливает метод *GetNextVolume* в качестве функции определения зоны текущего местоположения частицы, метод *GetNextVolume* вызывает главную функцию системы преобразования данных *getDataAtPoint*, принимающую в качестве входных данных объект типа *TGeoManager* и параметры частицы - местоположение и направление движения. Дальнейший алгоритм преобразования выглядит следующим образом:

- Функция *getDataAtPoint* получает указатель на экземпляр класса *TGeoNavigator*, приписанный объекту *TGeoManager*.
- Функция *getDataAtPoint* вызывает функцию *getZonesAtPoint*, которая проверяет нахождение частицы в области эксперимента, возвращает данные, соответствующие "внешнему вакууму"в определениях SHIELD или данные зоны, полученные из функции *getCurrentZone*.
- Для нахождения зоны в функции *getCurrentZone* вычисляются описания зоны необходимого объекта и производится логическое вычитание из его зоны зон, отвечающим внешним границам вложенных дочерних объектов.
- Для конвертирования заданного объектом *TGeoNode* тела в зону, в функции *getZoneFromBody* находится класс геометрического примитива и использованием метода *getZoneFromShape* находится определение зоны.
- В функции *getZoneFromShape* использованием оператора *dynamic_cast*, являющегося частью механизма динамической идентификации типа данных, находится корректный класс геометрического примитива, для которого специальными функциями, находящимися в файле исходных данных *TShieldGeometryConvert.cxx* и имеющими окончание "ToZones"в названии, осуществляется перевод в определение зоны. Для минимизации определения данных после использования логических операций перевод отдельного примитива системы Root осуществляется с минимизацией числа операций конъюнкции.
- Для нивелирования различий в точности систем SHIELD и Root в случае близкого нахождения точки с границей объектов, в качестве выходных данных передаётся две ближайшие зоны.
- После определения выходных данных система TShield при помощи функции *zoneDataToShield* производит перевод данных из типа *zoneData* в структуру, записывающую выходные данные в пригодном для использования в функции *shield_add_zone* виде.

После выхода из системы перевода геометрических данных система TShield в методе *GetNextVolume* производит очистку геометрических данных системы SHIELD и загрузку в неё актуальных

геометрических данных.

Часть текста файла исходного кода *TShieldGeometryConvert.cxx*, содержащего код преобразования объектов системы Root в зоны в понятиях SHIELD представлен в приложении 9. Текст остальных исходных файлов доступен на электронном приложении.

4.4 Выходные данные системы TShield

Для возможности полного сохранения и анализа результатов моделирования в класс TShield был добавлен объект типа системы хранения данных системы Root *TTree* с методом получения указателя на него (метод *GetTree*).

Для заполнения объекта хранения результатами используется система, описанная в параграфе 2.5: функция *AddTreeParticle*, отвечающая за заполнения дерева *TTree* вызывается функцией *TreeCallback*, переданной в систему SHIELD методом *shield_set_tree_callback* для получения выходных данных.

4.5 Выводы по главе 4

- Был создан интерфейсный класс *TShield*.
- Была создана система для преобразования геометрических данных из системы Root в систему SHIELD.

5 Использование системы TShield в системе MpdRoot

5.1 Интерфейсы к классу TShield

Для использования системы TShield в составе проекта MpdRoot необходимо создание интерфейса для возможности использования результатов моделирования. При этом для создания интерфейса могут быть использованы два различных подхода:

- Разработка класса интерфейса, наследуемый от класса-генератора событий и являющийся генератором событий системы MpdRoot.
- Разработка класса интерфейса, наследуемый от класса, обеспечивающего интерфейс для моделирования методами Монте-Карло, и являющийся классом системы транспорта частиц.

5.2 Класс TShieldGenerator

Для возможности использования системы TShield в качестве генератора событий был добавлен класс *TShieldGenerator* аналогично классу *THadgen*. В качестве метода доступа для передачи параметров работы системы используется метод, возвращающий указатель на используемый классом *TShieldGenerator* объект *TShield*.

Главным достоинством данного подхода является простота его использования и реализации, в то же время ввиду ограниченной применимости системы транспорта частиц в качестве генератора событий данная реализация является дополняющей реализацию второго подхода.

5.3 Класс TShieldMC

В качестве базового класса системы Root для моделирования методами Монте-Карло является чисто виртуальный класс *TVirtualMC*. Реализация класса интерфейса *TShieldMC*, наследованного от класса *TVirtualMC* обладает возможностью использования системы в качестве дополнительной системы переноса частиц в моделировании эксперимента MpdRoot. В то же время, ввиду изначальной реализации *TVirtualMC* под систему транспорта частиц Geant3 в описании класса *TVirtualMC* существует множество специфичных методов, обязательных к реализации в классах-наследниках.

Также в случае использования класса *TShieldMC* в системе MpdRoot необходимо создание файлов сценариев системы Root *shieldConfig.C* и *shieldlibs.C* в папке *gconfig* с правилами загрузки необходимых библиотек и создания объекта типа *TShieldMC* и изменение классов *FairRunSim* и *FairMCApplication*, находящиеся в каталоге *base* для корректной загрузки созданных фалов сценариев системы Root. Листинг файла изменений *TShield_MpdRoot.patch*

находится в приложении 10.

5.4 Выводы по главе 5

- Был создан интерфейсный класс *TShieldGenerator*, позволяющий использовать объекты класса *TShield* как генератор событий.
- Был создан интерфейсный класс *TShieldMC*, позволяющий использовать объекты класса *TShield* в качестве системы транспорта частиц системы MpdRoot.

6 Методы проведения апробации

Ввиду необходимости контроля корректности работы программы отдельное внимание было уделено контролю выполнения работы программы, основным методом которого является сверка результатов работы системы на наборе тестовых данных при одинаковом затравочном числе генератора случайных величин и контролю за работой функции *RAND*. Для определения корректности перевода геометрии производилось сравнение получаемого перевода тестовой геометрии с теоретическим.

Результаты тестирования показали отсутствие изменений, связанных с некорректностью созданных интерфейсов или изменений в коде систем HADGEN и SHIELD.

Выводы

В данной работе получены следующие результаты:

- Был изучена система работы программного комплекса MpdRoot
- Создана динамическая библиотека SHIELD
- Создан интерфейс генератора событий THadgen.
- Генератор событий THadgen подключён к системе MpdRoot с использованием класса интерфейса MpdGeneralGenerator.
- Создан и реализован алгоритм и преобразования геометрического представления из представления системы Root в представления системы SHIELD
- Создан интерфейс транспортного кода TShield.
- Созданы классы-интерфейсы для подключения класса TShield к системе MpdRoot в качестве различных типов классов.

Результаты данной работы были добавлены в программный комплекс MpdRoot в качестве отдельного пакета.

Также результаты данной работы могут быть применены в следующих областях:

- Система BmnRoot - программный комплекс для расчётов связанных с планируемым экспериментом BMN в ОИЯИ.
- Системы, основанные на системе FairRoot.
- Программный комплекс FairRoot.
- Адронная медицина.

Список литературы

1. *Gyulassy M.* The QGP Discovered at RHIC [Электронный ресурс].— Режим доступа: http://arxiv.org/pdf/nucl-th/0403032.

2. *Wilson T.* Super Proton Synchrotron marks its 25th birthday [Электронный ресурс].— Режим доступа: http://cerncourier.com/cws/article/cern/28470.

3. *Collaboration, RHIC.* Hunting the Quark Gluon Plasma (Formal Report) / RHIC Collaboration, New York:BNL, 2005 - 361 pp.

4. *Sissakian, A. N.* Design and Construction of Nuclotron-based Ion Collider fAcility (NICA), Conceptual design report / A. N. Sissakian [et al.]. — Dubna: JINR, 2008. — 149 pp.

5. *MPD, Коллаборация.* Многоцелевой детектор MPD для изучения столкновений тяжелых ионов на ускорителе NICA (Концептуальный дизайн-проект) / Коллаборация MPD. — Дубна: ОИЯИ, 2010. — 224 с.

6. Root [Электронный ресурс]. — Режим доступа: http://root.cern.ch.

7. FairRoot [Электронный ресурс]. — Режим доступа: http://fairroot.gsi.de.

8. CERN [Электронный ресурс]. — Режим доступа: http://home.web.cern.ch.

9. FAIR – Facility for Antiproton and Ion Research in Europe GmbH [Электронный ресурс]. — Режим доступа: http://www.fair-center.eu.

10. MpdRoot [Электронный ресурс]. — Режим доступа: http://mpd.jinr.ru.

11. FLUKA [Электронный ресурс]. — Режим доступа: http://www.fluka.org.

12. *Timoshenko G., Paraipan M., Florko B., Zaitsev L.* Monte-Carlo simulations for estimation of the radiation environment around the modernized nuclotron / Proceedings of RuPAC 2008, Zvenigorod, Russia - Zvenigorod, 2008 - P. 392-395.

13. *Krylov A., Paraipan M., SobolevskyN., et al.* MGeant4, MCNPX, and SHIELD code comparison concerning relativistic heavy-ion interaction with matter / Письма в ЭЧАЯ - 2014. - Т.11 №4(188) - С.847-850

14. SHIELD [Электронный ресурс]. — Режим доступа: http://www.inr.ru/shield.

15. ИЯИ РАН [Электронный ресурс]. — Режим доступа: http://www.inr.ru.

16. *Brun R., Rademakers F.* ROOT — An object oriented data analysis framework / Proceedings AIHENP'96 Workshop, Lausanne, Sep. 1996, Nucl. Inst. & Meth. in Phys. Res. A 389 (1997) - Lausanne, 1997 - P. 81-86.

17. *ROOT team* ROOT Data Analysis Framework. User's Guide / CERN Collaboration, Geneva, 2014 - 642 pp.

18. *Немнюгин С., Стесик О.* Современный Фортран. Самоучитель. / "БХВ Санкт-Петербург, 2004.

19. G77 compiler [Электронный ресурс]. — Режим доступа: http://www.mbr-pwrc.usgs.gov/software/g77.html.

20. GNU Fortran compiler [Электронный ресурс]. — Режим доступа: http://gcc.gnu.org/fortran.

21. GCC, the GNU Compiler Collection [Электронный ресурс]. — Режим доступа: http://gcc.gnu.org.

22. CMake [Электронный ресурс]. — Режим доступа: http://www.cmake.org.

23. *L. Garren et al,* Monte Carlo Particle Numbering Scheme / RPP-2010, J. Phys. G 37, 075021, - Berkeley, 2010 - P. 364-367.

24. *Garren L.* StdHep. Monte Carlo Standardization at FNAL Fortran and C Implementation / Fermilab Collaboration, Chicago, 2006. - 58pp

25. *Bassler N., Luhr A., Hansen D., Sobolevsky N.* SHIELD-HIT12A - User's Guide. Rev r597. / Aarhus, 2014. - 79pp

Приложение 1. Пример входного файла for023.dat системы SHIELD

```
1 RNDSEED 89736501
2 JPART0  25
3 HIPROJ  12.0     6.0
4 TMAX0   400.0    0.0
5 BEAMPOS 0.0      0.0      -1.50
6 NSTAT   20000    5000
```

Приложение 2. Пример входного файла for022.dat системы SHIELD

```
 1 NUMMED= 1
 2                                MEDIUM NO.  1  (Al)
 3 MEDTYP=1
 4  1 NUCLID= 13
 5                                MEDIUM NO.  2  (Air)
 6 STATE=   2
 7 RHO=      1.19
 8  1 NUCLID= 1         8
 9 IVALUE=    21.9
10  2 NUCLID= 6         5
11  3 NUCLID= 8         2
```

Приложение 3. Пример входного файла pasin.dat системы SHIELD

```
1      0      0                TOTAL OUTPUT (P, 1 GeV, Bodin, 12.04.97)
2    SPH      1          0.0          0.0          0.0        100.0
3    RPP      2        -10.0         10.0        -10.0         10.0          0.0          0.1
4    END
5    T01             +2
6    T02             +1        -2
7    END
8      1      2
9      1   1000
```

Приложение 4. Листинг части файла исходного кода shield_geo.c

```
/*-------------------------------------------------------------------+*/
/*                        1   2   3   4   5   6   7   8   9  |  10  |*/
/* Number at struct       0   1   2   3   4   5   6   7   8  |  9   |*/
/* Body                  SPH WED ARB BOX RPP RCC REC TRC ELL | END  |*/
/* N Parameters           6  30  36  36  18  21  22  22  17  |  0   |*/
/* N Surfaces             1   5   6   6   6   3   3   3   1  |  0   |*/
/*-------------------------------------------------------------------+*/
/*-------------------------------------------------------------------+*/
/* Surface type:      Equation:                        N Parameters:  |*/
/*          1         X-C=0                                  1      |*/
/*          2         Y-C=0                                  1      |*/
/*          3         Z-C=0                                  1      |*/
/*          4         A*x+B*y+C*z-D=0                        4      |*/
/*          5         (x-x0)**2+(y-y0)**2+(z-z0)**2-R**2=0   4      |*/
/*          6         x**2+y**2-R**2=0                       1      |*/
/*          7         x**2/A2+y**2/B2-1=0                    2      |*/
/*          8         x**2+y**2-(z-A1)**2/B2=0               2      |*/
/*          9         x**2/A2+y**2/B2+z**2/C2-1=0            3      |*/
/*-------------------------------------------------------------------+*/
struct {
    int count; //count of types of bodies
    int countOfParameters[9];
    char namesOfBodies[9][3];
} ShieldBodies = {9, {6, 30, 36, 36, 18, 21, 22, 22, 17}, {"SPH", "WED", "ARB"
    , "BOX", "RPP", "RCC", "REC", "TRC", "ELL"}};
int ShieldBodiesMaxCountOfParameters() {
    int i, max = 0;
    for (i = 0; i < ShieldBodies.count; i++)
        if (ShieldBodies.countOfParameters[i] > max) {
            max = ShieldBodies.countOfParameters[i];
        }
    return max;
}

struct SGeoBody createRPP(double *parameters) {
    struct SGeoBody body = {4, {}};
    if (parameters[0] >= parameters[1] ||
            parameters[2] >= parameters[3] ||
            parameters[4] >= parameters[5]) {
```

```
39          printf("%s: X1 must be less than X2", ShieldBodies.namesOfBodies[4]);
40          printf("%s: Y1 must be less than Y2", ShieldBodies.namesOfBodies[4]);
41          printf("%s: Z1 must be less than Z2", ShieldBodies.namesOfBodies[4]);
42          return;
43      }
44      int i;
45      for (i = 0; i < 3; i++) {
46          body.parameters[i * 6] = i + 1;
47          body.parameters[i * 6 + 1] = parameters[i * 2];
48          body.parameters[i * 6 + 2] = 1;
49          body.parameters[i * 6 + 3] = i + 1;
50          body.parameters[i * 6 + 4] = parameters[i * 2 + 1];
51          body.parameters[i * 6 + 5] = -1;
52      }
53      return body;
54  }
55  struct SGeoBody createBody(int type, double *parameters) {
56      switch (type) {
57          case 0:
58              return createSPH(parameters);
59              break;
60          case 1:
61              return createWED(parameters);
62              break;
63          case 2:
64              return createARB(parameters);
65              break;
66          case 3:
67              return createBOX(parameters);
68              break;
69          case 4:
70              return createRPP(parameters);
71              break;
72          case 5:
73              return createRCC(parameters);
74              break;
75          case 6:
76              return createREC(parameters);
77              break;
78          case 7:
```

```
79              return createTRC(parameters);
80              break;
81          case 8:
82              return createELL(parameters);
83              break;
84          default:
85              printf("Error in number of body");
86              struct SGeoBody defaultBody = { -1, {}};
87              return defaultBody;
88              break;
89      }
90  }
91  int shield_add_body(int type, double *parameters) {
92      struct SGeoBody body = createBody(type, parameters);
93      if (body.type == -1) {
94          return -1;
95      }
96      int bodynum = findBody(body);
97      if (bodynum != -1) {
98          return bodynum;
99      }
100     if(SGeometry.countBodies>=1000){ //Parameter MAXB at fortran code
101         printf("shield_init_geometry ERROR: Count of bodies large then array
                for bodies.\n");
102         return;
103     }
104     SGeometry.bodies[SGeometry.countBodies] = body;
105     return SGeometry.countBodies++; //returned SGeometry.countBodies before
            increment
106 }
107 //Return value is boolean.
108 int shield_add_zone(int countBodies,  int *zoneParameters, struct SGeoBody *
        bodies, struct MediumData medium) {
109     int i, c;
110     int bodynum;
111     struct SGeoZone zone = {0, 0, {}};
112     for (i = 0, c = 0 ; c < countBodies; i++) {
113         zone.definition[zone.countELements * 2] = zoneParameters[i];
114         switch (zoneParameters[i]) {
115             case 0:
```

```
116                 break;
117             case 1:
118             case -1:
119                 bodynum = shield_add_body(bodies[c].type, bodies[c].parameters
                        );
120                 if (bodynum == -1) {
121                     printf("shield_init_geometry ERROR: Cannot add body\n");
122                     return 1;
123                 }
124                 zone.definition[zone.countELements * 2 + 1] = bodynum;
125                 c++;
126                 break;
127             default:
128                 printf("shield_init_geometry ERROR: Uncorrect array for zone\n
                        ");
129                 return 1;
130                 break;
131         }
132         zone.countELements++;
133         if(zone.countELements>5000){ // Parameter, count of elements at array
                at struct
134             printf("shield_init_geometry ERROR: Count of elements at zone
                    defenition more then capacity\n");
135             return 1;
136         }
137     }
138     int zonenum = -1;
139     for (i = 0; i < SGeometry.countZones; i++) {
140         if (isZonesEqual(0, zone, SGeometry.zones[i])) {
141             zonenum = i;
142             break;
143         }
144     }
145     if (zonenum != -1) {
146 #ifdef _DEBUG
147         printf("shield_init_geometry WARNING: Doublicate of zone, skipping\n")
                ;
148 #endif
149         return 0;
150     }
```

```
151     if (medium.nType == 0 || medium.nType == 1000) {
152         zone.mediaNumber = medium.nType-1;
153     } else {
154         zone.mediaNumber = shield_add_medium(medium);
155     }
156 
157     if(SGeometry.countZones>=1000){ //Parameter MAXZ at fortran code
158         printf("shield_init_geometry ERROR: Count of zones large then array
                for zones.\n");
159         return 1;
160     }
161     SGeometry.zones[SGeometry.countZones] = zone;
162     SGeometry.countZones++;
163 //      return SGeometry.countZones++; //returned SGeometry.countBodies before
    increment
164     return 0;
165 }
166 
167 void shield_init_geometry() {
168     int i, j;
169 //      printf("\tCurrent count of elements: %i, %i\n", SGeometry.countBodies,
    SGeometry.countZones);
170     gdata0.NUMBOD = SGeometry.countBodies;
171     gdata0.NUMZON = SGeometry.countZones;
172     int curPos = 0;
173     struct SGeoBody body;
174     struct SGeoZone zone;
175     for (i = 0; i < SGeometry.countBodies; i++) {
176         body = SGeometry.bodies[i];
177         gdata2.BODYDA[i] = curPos + 1;
178         gdata2.BODYDB[curPos] = body.type + 1;
179         curPos++;
180         for (j = 0; j < ShieldBodies.countOfParameters[body.type]; j++) {
181             gdata2.BODYDB[curPos + j] = body.parameters[j];
182         }
183         curPos += ShieldBodies.countOfParameters[body.type];
184     }
185     curPos = 0;
186     for (i = 0; i < SGeometry.countZones; i++) {
187         zone = SGeometry.zones[i];
```

```
188             if(curPos + zone.countELements + 3 > 5000){
189                 printf("shield_init_geometry ERROR: Too large zones.\n");
190                 return;
191             }
192             gdata3.ZONEDA[i] = curPos + 1;
193             gdata3.ZONEDB[curPos] = zone.countELements;
194             gdata3.ZONEDB[curPos + 1] = zone.mediaNumber + 1;
195             int typeOfElements = 1;
196             for (j = 0; j < zone.countELements; j++) {
197                 if (zone.definition[j * 2] == 0) {
198                     typeOfElements = 0;
199                 } else if (typeOfElements != 0 && zone.definition[j * 2] == -1) {
                        //in gemca:1515
200                     typeOfElements = -1;
201                 }
202                 gdata3.ZONEDB[curPos + 3 + j] = zone.definition[j * 2] * (zone.
                        definition[j * 2 + 1] + 1);
203             }
204             gdata3.ZONEDB[curPos + 2] = typeOfElements;
205             curPos += zone.countELements + 3;
206         }
207     }
208     void shield_clean_geometry(){
209         SGeometry.countBodies=0;
210         SGeometry.countZones=0;
211         shield_clean_medium();
212     }
```

Приложение 5. Листинг части файла исходного кода THadgen.cxx

```
1  //
       ------------------------------------------------------------------------

2  void THadgen::GenerateEvent() {
3      if (hadgen_initialize()) {
4          printf("THadgen error: THadgen was not initialized!");//report error
5          return;
6      }
7      if (fAutoSeed) SetRandomSeed();
8      hadgen_generate();
9      hadgen_terminate();
10 //      CopyToInternalArray(); // copy all data into array of THadgen class
11 }
12
13 //
       ------------------------------------------------------------------------

14 TObjArray *THadgen::ImportParticles(Option_t *) {
15     CopyToInternalArray();
16     return fParticles;
17 }
18
19 //
       ------------------------------------------------------------------------

20 Int_t THadgen::ImportParticles(TClonesArray *particles, Option_t *) {
21     CopyToInternalArray();
22     particles->Clear();
23     TClonesArray &a = *particles;
24     for(int i=0; i<fParticlesNum; ++i){
25         new(a[i]) TParticle(*(TParticle*)(fParticles->At(i)));
26     }
27     return fParticlesNum;
28 }
29
30 //
       ------------------------------------------------------------------------

31 void THadgen::CopyToInternalArray(){
```

```
32      const Int_t MAX_PARTICLES = hadgen_get_max_stp();
33      const Int_t MAX_NUCLEI    = hadgen_get_max_snu();
34
35      fParticles->Clear();
36      TClonesArray &a = *((TClonesArray *)fParticles);
37      Int_t n_spt = 0;
38
39      {
40          hadgen_iter_stp_reset();   // reset the iterator
41          struct HadgenParticle_t buf;
42          for (Int_t i = 0; i < MAX_PARTICLES; i++) {
43              hadgen_iter_stp(&buf);
44              if (buf.type != PARTICLE_NONE) {
45                  Int_t pdg = hadgen_get_pdg_code(buf.type);
46                  new(a[i]) TParticle(pdg, 0, 0, 0, 0, 0,
47                                      buf.Px, buf.Py, buf.Pz, buf.Energy + buf.
                                          Weight,   // TOTAL ENERGY STORED
48                                      0, 0, 0, 0);
49                  n_spt ++;
50              } else break;
51          }
52      }
53      Int_t n_snu = 0;
54      {
55          hadgen_iter_snu_reset();   // reset second iterator for nuclei
56          struct HadgenResidualNuclei_t buf;
57          for (Int_t i = 0; i < MAX_NUCLEI; i++) {
58              hadgen_iter_snu(&buf);
59              if (buf.A != 0) {
60                  Int_t pdg = hadgen_get_pdg_code_nuclei(buf.A, buf.Z);
61                  new(a[i + n_spt]) TParticle(pdg, 0, 0, 0, 0, 0,
62                                              buf.Px, buf.Py, buf.Pz, buf.Energy
                                                  + 0.940 * buf.A, // TOTAL
                                                  ENERGY
63                                              0, 0, 0, 0);
64                  n_snu ++;
65              } else break;
66          }
67      }
68      fParticlesNum = n_spt + n_snu;
```

```
69         return;
70     }
```

Приложение 6. Листинг части файла исходного кода MpdGeneralGenerator.cxx

```
1  Bool_t MpdGeneralGenerator::ReadEvent(FairPrimaryGenerator *primGen) {
2      printf("MpdGeneralGenerator::ReadEvent\n");
3  //      TClonesArray *p = new TClonesArray("TParticles");
4      fGenerator->GenerateEvent();
5  //      fGenerator->ImportParticles(p, "");
6      TObjArray *p = fGenerator->ImportParticles();
7
8      TParticle *par = 0;
9      int numEntries = p->GetEntriesFast();
10     for (Int_t k = 0; k < numEntries; k++) {
11         par = (TParticle *)(p-> UncheckedAt(k));
12         if (fDebug)
13             printf("GeneralGen: kf=%d, p=(%.2f, %.2f, %.2f) GeV, x=(%.1f, %.1f
                   , %.1f) cm\n",
14                 par->GetPdgCode(), par->Px(), par->Py(), par->Pz(), par->Vx
                       (), par->Vz(), par->Vz());
15         if(par->GetPDG() == NULL) continue;
16         primGen->AddTrack(par->GetPdgCode(), par->Px(), par->Py(), par->Pz(),
               par->Vx(), par->Vz(), par->Vz());
17     }
18     return kTRUE;
19 }
```

Приложение 7. Файл изменений THadgen_runMC.patch

```
diff --git a/macro/mpd/runMC.C b/macro/mpd/runMC.C
index 41e2890..5eba407 100644
--- a/macro/mpd/runMC.C
+++ b/macro/mpd/runMC.C
@@ -180,6 +180,16 @@ void runMC (TString inFile = "auau.09gev.mbias.98k.ftn14"
    , TString outFile = "$V
    if (nEvents == 0)
        nEvents = MpdGetNumEvents::GetNumQGSMEvents(dataFile.Data()) -
           nStartEvent;

+#else
+#ifdef HADGEN
+  THadgen* hadGen = new THadgen();
+  hadGen->SetRandomSeed(clock() + time(0));
+  hadGen->SetParticleFromPdgCode(0, 196.9665, 79);
+  hadGen->SetEnergy(6.5E3);
+  MpdGeneralGenerator* generalHad = new MpdGeneralGenerator(hadGen);
+  primGen->AddGenerator(generalHad);
+
+#endif
 #endif
 #endif
 #endif
```

Приложение 8. Листинг файла исходного кода TShieldGeometryBool.cxx

```
1  // ------------------------------------------------------------------------
2  // -----          TShieldGeometry source file                          -----
3  // -----          Created by D. Sosnov                                 -----
4  // ------------------------------------------------------------------------
5
6  #include "TShieldGeometry.h"
7  namespace tgeanttoshield {
8  zoneElement notElement(zoneElement el) {
9      return std::pair<int, SGeoBody>(el.first * -1, el.second);
10 }
11 zoneList notZone(zoneList list) {
12     int cur, count = 1;
13     for (unsigned int i = 0; i < list.size(); ++i) {
14         count *= list.at(i).size();
15     }
16     zoneList out;
17     std::vector<zoneElement> tmp;
18     for (int k = 0; k < count; ++k) {
19         cur = k;
20         tmp.clear();
21         for (unsigned int i = 0; i < list.size(); ++i) {
22             int cursize = list.at(i).size();
23             tmp.push_back(notElement(list.at(i).at(cur % cursize)));
24             cur /= cursize;
25         }
26         out.push_back(tmp);
27     }
28     return out;
29 }
30 zoneList orZone(zoneList list1, zoneList list2) {
31     zoneList outList = list1;
32     outList.insert(outList.end(), list2.begin(), list2.end());
33     return outList;
34 }
35 zoneList andZone(zoneList list1, zoneList list2) {
36     zoneList outList;
37     std::vector<zoneElement> tmp;
38     for (unsigned int i = 0; i < list1.size(); ++i) {
39         for (unsigned int j = 0; j < list2.size(); ++j) {
```

```
40                tmp = list1.at(i);
41                tmp.insert(tmp.end(), list2.at(j).begin(), list2.at(j).end());
42                outList.push_back(tmp);
43            }
44        }
45        return outList;
46    }
47    }
```

Приложение 9. Листинг части файла исходного кода TShieldGeometryConvert.cxx

```
1  std::pair<zoneList, zoneList> getZoneFromShape(TGeoShape *shape, TGeoHMatrix
       currTransformation, double scale) {
2      std::pair<zoneList, zoneList> zonePair;
3      if (dynamic_cast<TGeoTubeSeg *>(shape) != NULL) {
4          zonePair = tubeSegToZones((TGeoTubeSeg *)shape, currTransformation,
               scale);
5      } else if (dynamic_cast<TGeoTube *>(shape) != NULL) {
6          zonePair = tubeToZones((TGeoTube *)shape, currTransformation, scale);
7      } else if (dynamic_cast<TGeoCtub *>(shape) != NULL) { //TODO
8          printf("TGeoCtub not implemented yet\n");
9      } else if (dynamic_cast<TGeoConeSeg *>(shape) != NULL) {
10         zonePair = coneSegToZones((TGeoConeSeg *)shape, currTransformation,
               scale);
11     } else if (dynamic_cast<TGeoCone *>(shape) != NULL) {
12         zonePair = coneToZones((TGeoCone *)shape, currTransformation, scale);
13     } else if (dynamic_cast<TGeoPara *>(shape) != NULL) { //TODO
14         printf("TGeoPara not implemented yet\n");
15     } else if (dynamic_cast<TGeoTrd1 *>(shape) != NULL) {
16         zonePair = trd1ToZones((TGeoTrd1 *)shape, currTransformation, scale);
17     } else if (dynamic_cast<TGeoTrd2 *>(shape) != NULL) {
18         zonePair = trd2ToZones((TGeoTrd2 *)shape, currTransformation, scale);
19     } else if (dynamic_cast<TGeoTrap *>(shape) != NULL) {
20         zonePair = trapToZones((TGeoTrap *)shape, currTransformation, scale);
21     } else if (dynamic_cast<TGeoSphere *>(shape) != NULL) {
22         zonePair = sphereToZones((TGeoSphere *)shape, currTransformation,
               scale);
23     } else if (dynamic_cast<TGeoPgon *>(shape) != NULL) {
24         zonePair = polyhedraToZones((TGeoPgon *)shape, currTransformation,
               scale);
25     } else if (dynamic_cast<TGeoPcon *>(shape) != NULL) {
26         zonePair = polyconeToZones((TGeoPcon *)shape, currTransformation,
               scale);
27     } else if (dynamic_cast<TGeoEltu *>(shape) != NULL) {
28         zonePair = ellipticalTubeToZones((TGeoEltu *)shape, currTransformation
               , scale);
29     } else if (dynamic_cast<TGeoArb8 *>(shape) != NULL) {
30         zonePair = arb8ToZones((TGeoArb8 *)shape, currTransformation, scale);
31     } else if (dynamic_cast<TGeoGtra *>(shape) != NULL) { //TODO
32         printf("TGeoGtra not implemented yet\n");
```

```
33      } else if (dynamic_cast<TGeoHype *>(shape) != NULL) { //TODO
34          printf("TGeoHype not implemented yet\n");
35      } else if (dynamic_cast<TGeoTorus *>(shape) != NULL) { //TODO
36          printf("TGeoTorus not implemented yet\n");
37      } else if (dynamic_cast<TGeoParaboloid *>(shape) != NULL) { //TODO
38          printf("TGeoParaboloid not implemented yet\n");
39      } else if (dynamic_cast<TGeoXtru *>(shape) != NULL) { //TODO
40          printf("TGeoXtru not implemented yet\n");
41      } else if (dynamic_cast<TGeoHalfSpace *>(shape) != NULL) { //TODO
42          printf("TGeoHalfSpace not implemented yet\n");
43      } else if (dynamic_cast<TGeoCompositeShape *>(shape) != NULL) { //TODO
44          printf("TGeoCompositeShape not implemented yet\n");
45      } else if (dynamic_cast<TGeoBBox *>(shape) != NULL) { //All of this shapes
            in subclasses of TGeoBBox
46          zonePair = boxToZones((TGeoBBox *)shape, currTransformation, scale);
47      } else {
48          printf("ELSE");
49      }
50      return zonePair;
51  }
52
53  SGeoBody createCone(TGeoTranslation startPoint, TGeoTranslation vectorToEnd,
        double rStart, double rEnd) {
54      SGeoBody outBody;
55      if (doubleEQ(rStart, rEnd)) {
56          outBody = {5, {0, 0, 0, 0, 0, 0, rStart}};
57          addVectorToElement(outBody, 0, startPoint);
58          addVectorToElement(outBody, 3, vectorToEnd);
59      } else {
60          startPoint = (rStart > rEnd) ? startPoint : startPoint + vectorToEnd;
                //WARNING
61          vectorToEnd = (rStart > rEnd) ? vectorToEnd : vectorToEnd.Inverse();
                //WARNING
62          double rMax = (rStart > rEnd) ? rStart : rEnd;
63          double rMin = (rStart > rEnd) ? rEnd : rStart;
64          outBody = {7, {0, 0, 0, 0, 0, 0, rMax, rMin}};
65          addVectorToElement(outBody, 0, startPoint);
66          addVectorToElement(outBody, 3, vectorToEnd);
67      }
68      return outBody;
```

```
69 }
70 zoneList cutByPhi(TGeoHMatrix currTransformation,
71                   double sPhi, double dPhi, double halfX, double halfY, double
                      halfZ) {
72     zoneList outList;
73     TGeoHMatrix currRotation = TGeoHMatrix(currTransformation); currRotation.
          SetTranslation(kNullVector);
74     TGeoTranslation currTranslation = TGeoTranslation(currTransformation);
75     double ePhi = sPhi + dPhi;
76     if (doubleGE(dPhi, 360)) return {{}};
77     TGeoHMatrix tmp = TGeoHMatrix(); tmp.RotateZ(sPhi);
78     TGeoHMatrix innerRot = currRotation * tmp;
79     TGeoTranslation startFirstBox = currTranslation + TGeoTranslation(innerRot
          * TGeoTranslation(-halfX, 0, -halfZ));
80     TGeoTranslation vec11 = TGeoTranslation(innerRot * TGeoTranslation(2 *
          halfX, 0, 0));
81     TGeoTranslation vec12 = TGeoTranslation(innerRot * TGeoTranslation(0, -
          halfY, 0));
82     TGeoTranslation vec13 = TGeoTranslation(innerRot * TGeoTranslation(0, 0, 2
          * halfZ));
83     SGeoBody box1 = {3, {}};
84     addVectorToElement(box1, 0, startFirstBox);
85     addVectorToElement(box1, 3, vec11);
86     addVectorToElement(box1, 6, vec12);
87     addVectorToElement(box1, 9, vec13);
88     tmp = TGeoHMatrix(); tmp.RotateZ(ePhi);
89     TGeoHMatrix outerRot = currRotation * tmp;
90     TGeoTranslation startSecondBox = currTranslation + TGeoTranslation(
          outerRot * TGeoTranslation(halfX, 0, -halfZ));
91     TGeoTranslation vec21 = TGeoTranslation(outerRot * TGeoTranslation(-2 *
          halfX, 0, 0));
92     TGeoTranslation vec22 = TGeoTranslation(outerRot * TGeoTranslation(0,
          halfY, 0));
93     TGeoTranslation vec23 = TGeoTranslation(outerRot * TGeoTranslation(0, 0, 2
          * halfZ));
94     SGeoBody box2 = {3, {}};
95     addVectorToElement(box2, 0, startSecondBox);
96     addVectorToElement(box2, 3, vec21);
97     addVectorToElement(box2, 6, vec22);
98     addVectorToElement(box2, 9, vec23);
```

```
99      if (doubleLE(dPhi, 180)) {
100         outList = {{std::make_pair(-1, box1), std::make_pair(-1, box2)}};
101     } else {
102         outList = {{std::make_pair(-1, box1)}, {std::make_pair(-1, box2)}};
103     }
104     return outList;
105 }
106 inline std::pair<zoneList, zoneList> polyconeToZones(TGeoPcon *polycone,
        TGeoHMatrix currTransformation, double scale) {
107     zoneList out, outerShell; //outer shell for conjunction at generating
            mother volume
108     TGeoHMatrix currRotation = TGeoHMatrix(currTransformation); currRotation.
            SetTranslation(kNullVector);
109     double sPhi = (double)polycone->GetPhi1(); //In Pcon angle converted to
            degrees, and TMatrixes work with degrees. (?)
110     double dPhi = (double)polycone->GetDphi();
111     int nz = polycone->GetNz();
112     double zCurrent, zPrev, rMaxCurrent, rMaxPrev, rMinCurrent, rMinPrev;
113     double *zValues = polycone->GetZ();
114     double *rMinValues = polycone->GetRmin();
115     double *rMaxValues = polycone->GetRmax();
116     double zMin = zValues[0] * scale, zMax = zValues[nz - 1] * scale, rMaxMax
            = rMaxValues[0] * scale;
117     TGeoTranslation startPoint, endVec;
118     zoneElement currentOuterCone, currentInnerCone;
119     double zCenter = (zMax + zMin) / 2.0;
120     for (int i = 1; i < nz; ++i) {
121         zPrev = zCenter + (zValues[i - 1] - zCenter) * scale;
122         zCurrent = zCenter + (zValues[i] - zCenter) * scale;
123         if (zPrev == zCurrent)continue;
124         rMaxPrev = rMaxValues[i - 1] * scale;
125         rMaxCurrent = rMaxValues[i] * scale;
126         rMinPrev = rMinValues[i - 1] * scale;
127         rMinCurrent = rMinValues[i] * scale;
128         startPoint = TGeoTranslation(currTransformation * TGeoTranslation(0,
                0, zPrev));
129         endVec = TGeoTranslation(currRotation * TGeoTranslation(0, 0, zCurrent
                - zPrev));
130         rMaxMax = (rMaxCurrent > rMaxMax) ? rMaxCurrent : rMaxMax;
131         if (doubleNE(rMaxPrev, 0) || doubleNE(rMaxCurrent, 0)) {
```

```
132             currentOuterCone = std::make_pair(1, createCone(startPoint, endVec
                    , rMaxPrev, rMaxCurrent));
133             outerShell = orZone(outerShell, {{currentOuterCone}});
134         }
135         if (doubleNE(rMinPrev, 0) || doubleNE(rMinCurrent, 0)) {
136             currentInnerCone = std::make_pair(1, createCone(startPoint, endVec
                    , rMinPrev, rMinCurrent));
137             out = orZone(out, {{currentInnerCone}});
138         }
139     }
140     out = andZone(outerShell, notZone(out));
141     TGeoHMatrix phiTransformation = currTransformation * TGeoTranslation(0, 0,
            zMin + (zMax - zMin) / 2.0);
142     zoneList phiZone = cutByPhi(phiTransformation, sPhi, dPhi, rMaxMax,
            rMaxMax, (zMax - zMin) / 2.0);
143 //      startPoint = TGeoTranslation(currRotation * TGeoTranslation(0, 0, zMin
        + (zMax - zMin) / 2.0));
144 //      zoneList phiZone = cutByPhi(startPoint, currRotation, sPhi, dPhi,
        rMaxMax, rMaxMax, (zMax - zMin) / 2.0);
145     out = andZone(out, phiZone);
146     outerShell = andZone(outerShell, phiZone);
147     return std::make_pair(out, outerShell);
148 }
149
150 MediumData getMedium(TGeoNode *node) {
151     double g = 1;
152     double cm3 = getDefaultScale() * getDefaultScale() * getDefaultScale();
153     MediumData out = {0, 0, 0, {}};
154     TGeoMaterial *medium = node->GetMedium()->GetMaterial();
155     out.nChemEl = medium->GetNelements();
156     out.nType = (out.nChemEl == 1) ? 1 : 2;
157     out.Rho = medium->GetDensity() / (g / cm3);
158     if (out.Rho == 0) {
159         printf("Is medium a Vacuum?");
160         out.nType = 1000;
161         return out;
162     }
163     for (int i = 0; i < out.nChemEl; ++i) {
164         out.Elements[i].Nuclid = medium->GetElement(i)->Z();
165         out.Elements[i].A = medium->GetElement(i)->N();
```

```
166             out.Elements[i].Z = medium->GetElement(i)->Z();
167             out.Elements[i].Density = (medium->IsMixture()) ? out.Rho * (((
                    TGeoMixture *)medium)->GetWmixt())[i] : out.Rho;
168             out.Elements[i].Conc = out.Elements[i].Density / medium->GetElement(i)
                    ->A() * TMath::Na() * 1E-27;
169 //              out.Elements[i].Conc = mat->GetVecNbOfAtomsPerVolume()[i] * cm3 * 1
        E-27; //Correct
170 //              out.Elements[i].ionEv = mat->GetElement(i)->GetIonisation()->
        GetMeanExcitationEnergy() / eV; //As I unserstand, it is, but it can fill
        automatically.
171             for (int k = 0; k < i; ++k) {
172                 if (out.nType != 3 && out.Elements[i].Z == out.Elements[k].Z)
173                     out.nType = 3;
174             }
175         }
176         return out;
177     }
178 }
```

Приложение 10. Файл изменений TShield_MpdRoot.patch

```
diff --git a/base/FairMCApplication.cxx b/base/FairMCApplication.cxx
index 701b945..bcbe58a 100644
--- a/base/FairMCApplication.cxx
+++ b/base/FairMCApplication.cxx
@@ -218,6 +218,8 @@ void FairMCApplication::InitMC(const char* setup, const char* cuts)
       fMcVersion = 1;
     } else if(MCName == "TFluka") {
       fMcVersion = 2;
+  } else if(MCName == "TShield") {
+    fMcVersion = 4;
     } else {
       fMcVersion = 3;  //Geane
     }
diff --git a/base/FairRunSim.cxx b/base/FairRunSim.cxx
index c9628d1..a976b9b 100644
--- a/base/FairRunSim.cxx
+++ b/base/FairRunSim.cxx
@@ -325,6 +325,33 @@ void FairRunSim::SetMCConfig()
         if(AbsPath) { ConfigMacro = fUserConfig; }
         else { ConfigMacro =work_config+fUserConfig; }
       }
+    //-------------------------------------------------Shield Config ----------------------------------------
+  } else if(strcmp(GetName(),"TShield") == 0 ) {
+    TString shieldLibMacro="shieldlibs.C";
+    TString shieldMacro="shieldConfig.C";
+    if(fUserConfig.IsNull()) {
+      shieldMacro="shieldConfig.C";
+      fUserConfig = shieldMacro;
+      fLogger->Info(MESSAGE_ORIGIN,"-------------- Standard Config is called ----------------------------------");
+    } else {
+      if (fUserConfig.Contains("/")) { AbsPath=kTRUE; }
+      shieldMacro = fUserConfig;
+      fLogger->Info(MESSAGE_ORIGIN,"---------------User config is used :  ", shieldMacro.Data());
+    }
+    if (TString(gSystem->FindFile(config_dir.Data(),shieldLibMacro)) !=
```

```
    TString("")) {
+       fLogger->Info(MESSAGE_ORIGIN,"---User path for Configuration (
    shieldlibs.C) is used : %s " , config_dir.Data());
+     } else {
+       shieldLibMacro=work_config+"shieldlibs.C";
+     }
+     LibMacro=shieldLibMacro;
+     LibFunction="shieldlibs()";
+     if (!AbsPath && TString(gSystem->FindFile(config_dir.Data(),shieldMacro))
     != TString("")) {
+       fLogger->Info(MESSAGE_ORIGIN,"---User path for Configuration (
    shieldConfig.C) is used : %s", config_dir.Data());
+       ConfigMacro=shieldMacro;
+     } else {
+       if(AbsPath) { ConfigMacro = fUserConfig; }
+       else { ConfigMacro =work_config+fUserConfig; }
+     }
    }
    //-------------------------------------------------SetCuts
        ------------------------------------------------
    if (TString(gSystem->FindFile(config_dir.Data(),cuts)) != TString("")) {
diff --git a/gconfig/shieldConfig.C b/gconfig/shieldConfig.C
new file mode 100644
index 0000000 100755
--- /dev/null
+++ b/gconfig/shieldConfig.C
@@ -0,0 +1,10 @@
+ void Config()
+ {
+
+  /// Create the G4 VMC
+  TShieldMC* shield = new TShieldMC("TShield", "The Shield Monte Carlo");
+  cout << "TShield has been created." << endl;
+
+  shield->SetRandomSeed(clock() + time(0));
+
+}
diff --git a/gconfig/shieldlibs.C b/gconfig/shieldlibs.C
new file mode 100644
index 0000000 100755
```

```
71 --- /dev/null
72 +++ b/gconfig/shieldlibs.C
73 @@ -0,0 +1,20 @@
74 + #include <iostream>
75 +
76 +Bool_t isLibrary(const char* libName)
77 + {
78 +   if (TString(gSystem->DynamicPathName(libName, kTRUE)) != TString(""))
79 +     return kTRUE;
80 +   else
81 +     return kFALSE;
82 +}
83 +
84 +void shieldlibs()
85 +{
86 +   cout << "Loading TShield libraries ..." << endl;
87 +
88 +   gSystem->Load("libSHIELD.so");
89 +   gSystem->Load("libTShield.so");
90 +   gSystem->Load("libTShieldMC.so");
91 +
92 +   cout << "Loading TShield libraries ... finished" << endl;
93 +}
```