

RT² Manual

Contents

RT ² Manual	1
1 Introduction	5
2 Installation	6
3 Physics	6
4 RT ² Input Format	7
4.1 Format of Argument Card	7
4.2 Format of List Card	8
4.3 Common Cards	8
4.3.1 MACRO	8
4.3.2 ENTRY	9
5 RT2builder	9
5.1 Program Arguments	9
5.2 GEO_GLOBAL	10
5.3 Surface Cards	10
5.3.1 CUBE	11
5.3.2 ICOSPHERE	11
5.3.3 UVSPHERE	12
5.3.4 ELLIPSOID	13
5.3.5 CYLINDER	13
5.3.6 CONE	14
5.3.7 TORUS	14
5.3.8 REVOLUTION	15
5.3.9 TOROID	16
5.3.10 MODEL	16
5.3.11 VOXEL	17
5.4 Transform Cards	17
5.4.1 ROT_DEFI	18
5.4.2 TRANSFORM_BEGIN / TRANSFORM_END	18
5.5 REGION	19
5.6 Example of MCNP PRIMER	20
6 RT2mc	22
6.1 Program Arguments	22
6.2 CUDA_SETTINGS	22

6.3	Physics Cards	24
6.3.1	PHOTON	25
6.3.2	ELECTRON	25
6.3.3	POSITRON	27
6.3.4	VACANCY	27
6.3.5	NEUTRON	27
6.3.6	NEUTRON_HIGH	28
6.3.7	DELTA	28
6.3.8	GENERIC_ION	28
6.3.9	ION_INELASTIC	29
6.3.10	INCL_SETTINGS	30
6.3.11	CUTOFF	30
6.4	Material Cards	31
6.4.1	COMPOUND	31
6.4.2	MAT_PROP	34
6.4.3	ASSIGN	34
6.5	Source Cards	34
6.5.1	BEAM_GLOBAL	35
6.5.2	BEAM_CUBIC	35
6.5.3	BEAM_CYLINDER	39
6.5.4	BEAM_SPHERE	40
6.5.5	BEAM_PS	40
6.6	Scoring Cards	40
6.6.1	DENSITY	41
6.6.2	TRACK	41
6.6.3	CROSS	42
6.6.4	MESH_DENSITY	42
6.6.5	MESH_TRACK	43
6.6.6	DETECTOR	44
6.6.7	PHASE_SPACE	45
7	RT2viewer	45
8	RT2interactive	47
9	RT2moduleTester	49

10	Python Interface	49
10.1	rt2.hadron.....	49
10.1.1	class NuclearMassData().....	49
10.2	rt2.scoring	50
10.2.1	class Density(file_name).....	50
10.2.2	class Track(file_name).....	50
10.2.3	class Cross(file_name).....	51
10.2.4	class MeshDensity(file_name).....	52
10.2.5	class MeshTrack(file_name).....	53
10.2.6	class Detector(file_name).....	55
10.2.7	class PhaseSpace(file_name).....	55
11	Auxiliary Scripts	56
11.1	RT2dicom.....	56
11.2	RT2nifti.....	57
11.3	RT2tetra	58
11.4	RT2fluka.....	58
11.5	RT2geant4	58

1 Introduction

Ray-Tracing accelerated Radiation Transport Monte Carlo (RT² Monte Carlo) 코드는 GPU 상에서 여러 종류의 방사선 입자를 동시에 계산하기 위해 개발되었다. RT²는 MCNP, Geant4 및 FLUKA 등 방사선물리에서 널리 사용되는 Monte Carlo 코드와 비교 가능한, 유사한 연산 정밀도를 확보 하면서 동시에 훨씬 빠른 연산 속도를 제공하여 고성능이 필요한 방사선 전산 모사 하위 분야의 생산성을 확보하는 것을 목표로 한다. 이 매뉴얼에서는 RT²의 설치 방법, 환경 및 입력문의 문법과 실행 시 자주 발생하는 오류에 대해 설명한다.

RT² Monte Carlo 코드는 Compute Unified Device Architecture (CUDA) 기술 기반으로 개발되어 Nvidia GPU 환경에서 작동한다. 중성자, 광자, 전자, 양전자, 양성자 및 산소 원자까지의 중이온을 지원하고 있으며 1 GeV 혹은 nuclei projectile의 경우 1 GeV/u의 에너지까지 계산할 수 있다. Voxel 구조와 Constructed Solid Geometry (CSG) 모두 지원한다. 지원되는 feature의 범위에 따라 기대되는 응용 분야는 다음과 같다.

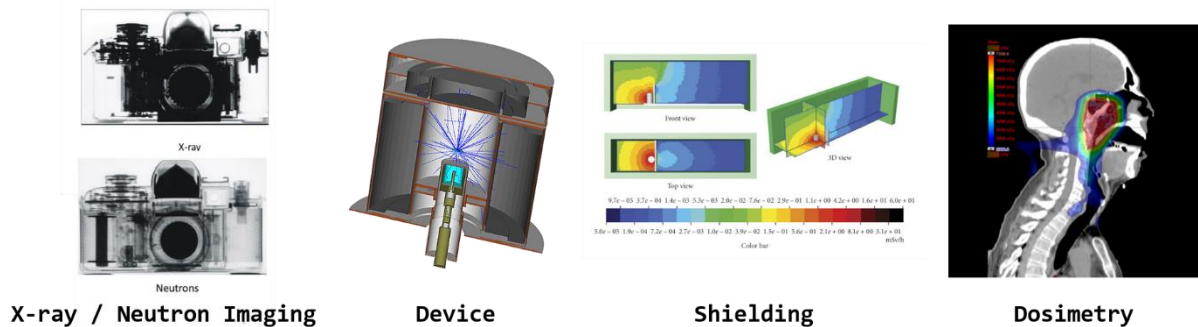


Figure 1. RT²가 적용 가능한 어플리케이션

지원되는 특징과 한계를 정리하면 다음과 같다.

Features

- CUDA C++ 기반 핵심 알고리즘 & Python 기반 인터페이스의 복합 구조
- Nvidia OptiX™ 프레임워크를 활용한 geometry tracking 하드웨어 가속
- Cross-platform (Windows and Linux)
- 동일 규모(가치)의 CPU 서버 환경에서의 CPU Monte Carlo 알고리즘 대비 50배 이상의 연산성능 제공
- EGSnrc 기반 전자, 광자, 양전자 수송
- EGSnrc PRESTA-II 기반 condensed history 알고리즘
- 메시 기반의 Constructive Solid Geometry (CSG) 구조 지원
- SolidWorks 도면 기반의 3차원 모델 (STL, OFF 등) 직접 입력 가능

- DICOM, NIFTI 등의 CT 데이터 기반의 voxel 구조 지원
- CSG & voxel 복합 구조 지원
- Thermal neutron scattering이 포함된 groupwised neutron transport 알고리즘
- 최대 산소 원자까지의 중이온 수송 가능

Limitations

- Nvidia 외의 그래픽장치 (AMD 등) 미지원
- RT코어 미탑재 하드웨어에서는 성능이 일정 수준 감소할 수 있음
- 논리 geometry가 일부 제한됨 (평면 등)
- 고에너지 물리 지원하지 않음 (파이온, 뮤온 등의 입자 수송 미지원)
- 1 keV 미만의 저에너지 영역 지원하지 않음

2 Installation

설치 절차 확정되지 않음.

3 Physics

다음은 RT²가 지원하는 입자 및 에너지 범위를 나타낸다.

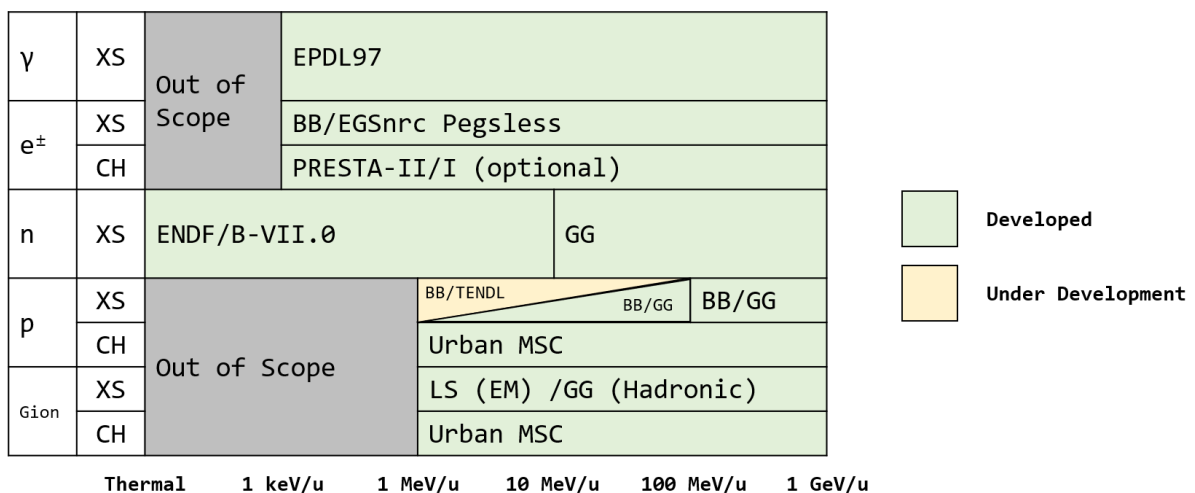


Figure 2 RT²의 physics scope

- BB: Bethe-Bloch Model
- LS: Lindhard-Sorensen Model
- GG: Glauber-Gribov Model

4 RT² Input Format

RT²의 입력문 시스템은 FLUKA, MCNP 등 주로 사용되는 Monte Carlo 코드들의 입력문 형식을 참고하여 개발되었다. Fortran 기반의 두 코드에 비해 입력문의 형식은 상대적으로 자유롭다. RT² 입력문은 여러 개의 카드로 구성되며 카드는 크게 argument type, list type으로 구분된다. 카드들의 순서는 대체로 제한이 없지만, 종속관계가 존재할 경우 순서를 지켜야 한다.

입력된 라인의 공백이 아닌 첫 번째 문자가 '#'인 경우 해당 라인은 주석 처리된다. 또한 라인의 모든 문자가 공백인 경우 해당 라인은 무시된다.

4.1 Format of Argument Card

Argument type card는 다음과 같은 형식을 가진다.

```
CARD_KEY arg1=value1 arg2=value2 ...
```

CARD_KEY는 카드의 타입을 결정한다. arg1, arg2 ... 는 카드에 종속된 인수의 이름을 의미하고, 공백 없이 뒤따르는 '='를 이용하여 해당 변수에 값을 입력할 수 있다. 각 변수들 사이는 공백으로 구분되어야 하고, 공백의 개수 및 탭 문자와의 혼합은 자유롭다. 변수의 순서는 상관이 없다. 입력되는 값은 필요에 따라 두 개 이상일 수 있으며 이 경우 각 변수들은 공백 없는 쉼표를 이용하여 연속적으로 입력할 수 있다. 예시는 다음과 같다.

```
CARD_KEY arg1=1.5,0.0,0.0 ...
```

이 경우 'arg1'이라는 변수에 길이가 3인 벡터 {1.5, 0.0, 0.0} 이 입력된다. 카드에 종속된 argument들은 정해진 길이의 값이 입력되어야 하며 정의된 길이와 다를 경우 오류가 발생할 수 있다. 또한 다음과 같이 쉼표 뒤에 아무것도 입력되지 않은 경우 정상적으로 처리되지 않고 오류가 발생할 수 있다.

```
CARD_KEY arg1=1.5,0.0,
```

카드에 정의되지 않은 argument는 무시된다. 변수의 기본값이 정의된 경우 카드에 명시적으로 입력하지 않아도 기본값으로 처리된다. Bool 타입의 값을 요구하는 경우 0을 입력하면 false, 나머지 모든 값은 true로 처리된다. 특정 변수의 경우 범위가 제한되는데, 범위를 벗어날 경우 오류가 발생하고 이는 출력 log에서 확인할 수 있다.

한 카드의 길이가 지나치게 길어질 경우 '\' 문자를 line 마지막에 표시하여 이어서 입력할 수 있다. 예시는 다음과 같다.

```
CARD_KEY arg1=value1 arg2=value2 \
          arg3=value3 arg4=value4 \
          arg5=value5 arg6=value6
```

4.2 Format of List Card

List type card는 다음과 같은 형식을 가진다.

```
CARD_KEY value1 value2 value3 value4 ...
```

CARD_KEY는 카드의 타입을 결정한다. value1, value2, ... 는 리스트 원소들을 의미하고 각 카드에 정의된 type을 가져야 한다. 예를 들어 float 타입의 변수를 요구하는 위치에 string 타입의 변수가 입력되면 오류가 발생할 수 있다. 일반적인 경우 리스트 길이는 제한이 없지만, 특정한 카드의 경우 리스트 길이에 대한 제한 규칙이 있다. value 사이의 공백이나 개행 문자 규칙은 argument card와 동일하다.

4.3 Common Cards

4.3.1 MACRO

Macro 카드는 입력문의 변수를 일괄적으로 조정하기 위해 제공된다. Macro 카드는 list card 형식을 따르며 문법은 다음과 같다.

```
$key$ value
```

Macro key는 앞뒤에 decorator 문자 '\$'가 필요하며 line의 맨 처음 나타나는 문자열이 해당 형식을 따를 때 macro card로 인식된다. Macro card 하위의 텍스트 중 \$key\$ 문자열은 모두 'value' 문자열로 치환된다. 다음은 macro 카드의 사용 예시이다.

```
$prefix$ whatever
$size$ 10.0,10.0,10.0
$pos$ 10.0
...
CUBE name=$prefix$1 origin=-$pos$,0.0,0.0 \
      size=$size$
CUBE name=$prefix$2 origin=+$pos$,0.0,0.0 \
      size=$size$
...
```

해당 입력문은 최종적으로 다음과 같이 처리된다.


```

...
CUBE  name=whatever1  origin=-10.0,0.0,0.0  \
      size=10.0,10.0,10.0
CUBE  name=whatever2  origin=+10.0,0.0,0.0  \
      size=10.0,10.0,10.0
...

```

Macro 카드의 각 key들은 중복되지 않아야 하며 nested definition이 허용된다. Nested definition이 사용될 경우 macro 카드들은 종속관계에 맞게 나열되어야 한다.

4.3.2 ENTRY

ENTRY 카드는 길이가 고정되지 않은 리스트를 생성하기 위해 사용된다. Entry 카드는 list card 형식을 따르며 문법은 다음과 같다.

```
ENTRY  name    element1  element2  ...
```

name은 entry의 이름을 의미하며 entry 내에서는 고유한 값을 가져야 한다. 이후 나타나는 모든 값은 element로 처리되며, float 타입을 가진다. Entry 카드는 좌표 리스트, 히스토그램을 생성하는 데 사용되며 entry를 사용하는 카드들은 이후에 설명한다.

5 RT2builder

RT²의 주요 개발목적 중 하나인 방사선치료 분야에서는 동일한 geometry에서 여러 번의 연산을 요구하는 경우가 있다. 따라서 연산 효율성을 위해 geometry 생성 코드와 Monte Carlo 코어 알고리즘이 분리되었다. RT2builder는 geometry 생성 파트만 따로 분리된 코드로 RT2mc 코드와 동일한 입력문 포맷을 가진다. 이하 내용에서는 RT2builder 코드에서 사용되는 card들을 설명한다.

5.1 Program Arguments

RT2builder는 Windows 명령 프롬프트 혹은 Linux 터미널에서 RT2builder 명령어를 실행하여 열 수 있다. 프로그램은 실행 인수로 요구하는 것들이 있는데, -h를 인수로 입력하거나 잘못된 인수를 입력했을 때 사용할 수 있는 인수의 리스트와 설명을 확인할 수 있다. 다음은 윈도우 10 환경에서 'RT2builder -h'를 입력했을 때 출력되는 결과물이다.

```

RT2builder -h
Parameters: --input      | -i <filename>      File for MC input
             --output    | -o <filename>      File for MC output
             --result     | -r <directory>     Where output & tally saved
             --debug      | -d                 Print detail
             --syntax     | -s                 Print MC parameter info
             --help       | -h                 Print this message

```

Figure 3 RT2builder 프로그램의 입력 인수 목록

--input은 프로그램 실행에 필수로 요구하는 인수로, RT2builder 입력문 텍스트 파일 이름을 의미한다. --output은 프로그램 실행 시 생성되는 log 파일 이름을 의미하며 입력하지 않을 시 log가 터미널에 출력된다. 해당 인수는 파일시스템 디렉토리 형태를 가지고 있지 않고, 단일 파일 이름 형태를 가져야만 한다. --result는 프로그램 실행 시 생성되는 log 파일 및 기타 데이터가 저장될 위치를 지정할 때 사용된다. 지정하지 않을 시 프로그램이 실행되고 있는 작업 위치에 기록된다. --debug는 데이터 파일 읽기/쓰기, 테이블 생성 등 내부 동작을 로그 파일에 추가로 저장하는 데 사용되는 인수로, 프로그램 오동작 혹은 알 수 없는 오류로 강제종료가 발생할 때 추적하는 데 사용된다. --syntax는 프로그램에 사용되는 card들의 자세한 설명을 출력한다.

5.2 GEO_GLOBAL

GEO_GLOBAL 카드는 geometry 생성 알고리즘 및 파일 출력을 제어하기 위해 사용된다. Argument card 문법을 따르며 입력 가능한 인수는 다음과 같다.

- stepsize [float]: The step size epsilon for boundary crossing evaluator [cm]. (Default=1e-6)
- error_tolerance [float]: The geometrical error tolerance for boundary crossing evaluator. For each surface, if the area where the error occur is larger than this parameter compared to the total area, is raised. (Default=0.001)
- save_primitive [bool]: If true, all surface primitives are saved to object file format (off). (Default=false)
- save_corefined [bool] : If true, all corefined surfaces are saved to object file format (off). (Default=false)

stepsize 인수는 corefine 연산 이후 개별 삼각형에 대해 안쪽과 바깥쪽의 region을 결정하는 데 사용된다. 만약 부동소수점 오류로 인해 생성된 구조에 오류가 발생한다면 stepsize 인수를 늘려야 한다. save_primitive와 save_corefined 인수는 각각 입력문으로 생성된 모든 surface의 원본 혹은 corefine 연산 결과를 object file format (off) 파일로 저장하는 데 사용된다. 활성화될 경우 3차원 모델 파일들이 프로그램 실행 위치에 저장된다.

5.3 Surface Cards

Surface card는 특정한 형태의 닫힌 공간 혹은 voxel 구조를 생성한다. RT²는 surface card로 생

성된 공간들의 Boolean 조합으로 전체 geometry를 구성한다. 해당 카드들은 FLUKA의 body, MCNP의 cell, Geant4의 solid와 비슷한 개념을 가진다고 볼 수 있다.

Surface cards를 제외한 다른 모든 card들은 종속관계가 성립하지 않는다면 순서에 상관없이 input의 아무 위치에나 작성하면 된다. 하지만 surface cards의 경우 nested transformation을 적용하기 위해 시작과 끝을 선언해야 하는데 SURFACE_BEGIN과 SURFACE_END의 두 가지 카드들로 정의할 수 있다. Surface를 정의하기 위한 입력문은 반드시 다음 형태를 가져야 한다.

```
SURFACE_BEGIN
  (... surface / transformation cards ...)
SURFACE_END
```

SURFACE_BEGIN과 SURFACE_END 카드는 아무 인수도 가지지 않으며 입력문 내에 단 한 개만 존재해야 한다. 또한 두 카드 사이에는 surface 및 transformation 형식의 카드를 제외한 다른 카드가 존재하지 않아야 한다. 단 공백이나 주석의 경우에는 허용된다. 이 두 카드 사이에 존재하지 않는 surface 및 transformation 형식의 카드는 입력문에서 무시된다.

RT²에서는 총 11개의 surface card를 지원하며 이하에서는 문법과 설명 및 사용 예시를 보인다.

5.3.1 CUBE

CUBE 카드는 평행 육면체 surface를 생성한다. Argument card 문법을 따르며 입력 가능한 인수는 다음과 같다.

- name [str]: The name of surface primitive. The surface name must be unique.
- center [float]: The center coordinate of this cube [cm].
- size [float]: The length of one side for each axis (xyz) [cm].

center 인수는 육면체의 무게중심 좌표를 설정한다. 인수는 세 개 요구된다. 기본값이 존재하지 않으며 필수로 입력해야 한다. size 인수는 각 변의 XYZ축 길이를 설정한다. Size 인수는 세 값이 모두 0보다 커야 하며 필수로 입력해야 한다. 다음은 한 변의 길이가 10 cm이고 중심의 좌표가 (0, 0, 10) cm인 정육면체를 생성하는 카드의 예시이다.

```
CUBE name=whatever center=0,0,10 size=10,10,10
```

이 육면체의 경우 lower bound와 upper bound가 세 축에 대해 각각 $x=-5$, $x=5$, $y=-5$, $y=5$, $z=0$, $z=10$ 이 된다.

5.3.2 ICOSPHERE

ICOSPHERE 카드는 균일 삼각형으로 구 형태의 surface를 생성한다. Argument card 문법을 따르며 입력 가능한 인수는 다음과 같다.

- name [str]: The name of surface primitive. The surface name must be unique.
- center [float]: The center coordinate of this ico-sphere [cm].
- radius [float]: The radius of the sphere [cm].
- order [int]: The order of ico-sphere tessellation algorithm. Higher order generates smoother mesh. (Default=4)

center 인수는 구의 중심을 설정한다. 이 인수는 필수로 요구된다. radius 인수는 구의 반지름을 설정하고 값은 0보다 커야 하고 필수로 요구된다. order는 ico-sphere 분할 알고리즘의 반복 횟수를 의미하는데, 1 이상 8 이하의 값을 지원한다. 기본값은 4이다. 인수의 값이 1일 때 구는 정십면체와 같은 형태를 가지게 되고, order를 하나 올릴 때 마다 삼각형 하나가 네 개의 삼각형으로 분할되어 더 매끄러운 표면의 구 형태를 가지게 된다.

다음은 원점이 중심이고 반지름이 5 cm인 ico-sphere를 생성하는 card의 예시이다.

```
ICOSPHERE name=whatever center=0,0,0 radius=5
```

order를 입력하지 않았기 때문에 기본값인 4가 사용되고 그 결과 총 삼각형이 5,120개인 4차 ico-sphere가 생성된다.

5.3.3 UVSPHERE

UVSPHERE 카드는 축을 기준으로 polar, azimuthal angle이 균등하게 분할된, 회전체 형태의 구를 생성한다. Argument card 문법을 따르며 입력 가능한 인수는 다음과 같다.

- name [str]: The name of surface primitive. The surface name must be unique.
- center [float]: The center coordinate of the UV sphere [cm].
- direction [float]: The direction vector of the UV sphere. Initial axis of rotation is transformed from z-axis to this variable. (Default={0,0,1})
- radius [float]: The radius of this sphere [cm].
- vertices [int]: Polygon resolution of this solid. (Default={100,100})

center 인수는 구의 중심을 설정한다. 이 인수는 필수로 요구된다. direction은 uv-sphere를 생성하는 회전축 벡터를 설정한다. 이 인수는 길이가 3이며 자동으로 정규화된다. radius는 구의 반지름을 설정한다. vertices 인수는 두 개가 사용되고, 각각 polar 및 azimuthal 방향으로 균등 간격으로 회전하면서 꼭지점을 몇 개 생성할지 결정한다. 두 인수는 3 이상 1000 이하의 값을 설정할 수 있고 기본값은 100이다. 높은 값을 설정할수록 더 매끄러운 표면의 구를 생성한다.

다음은 원점이 중심이고 반지름이 5 cm인 uv-sphere를 생성하는 card의 예시이다.

```
UVSPHERE name=whatever center=0,0,0 radius=5
```

vertices 인수를 설정하지 않았기 때문에 기본값인 100이 사용되었고 그 결과 총 삼각형의 개수가 19,600인 구가 생성된다.

앞서 사용한 두 개의 예시 card에 의해 생성된 ico-sphere와 uv-sphere의 형태는 다음과 같다.

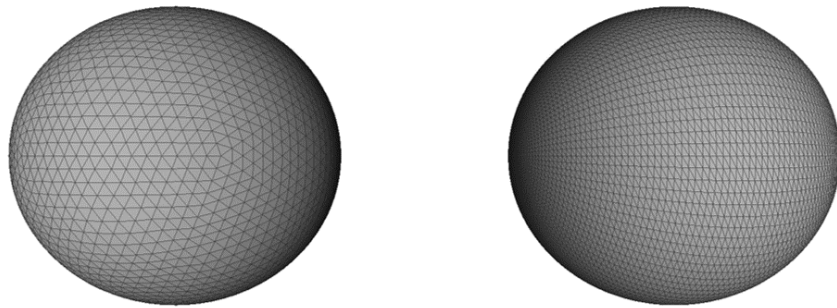


Figure 4. ICOSPHERE 카드로 생성한 구 (왼쪽), UVSPHERE 카드로 생성한 구 (오른쪽)

Ico-sphere는 모든 삼각형의 넓이가 거의 균등하기 때문에, 독립된 region이나 불규칙적인 메시와의 Boolean 조합에 사용할 때 유리하다. 반면 uv-sphere는 회전체 형태를 가지고 있기 때문에 원기둥이나 원뿔과 같은 다른 회전체와 조합하여 사용할 때 유리하다.

5.3.4 ELLIPSOID

ELLIPSOID는 타원체 surface를 생성한다. Uv-sphere 기반으로 생성된 구가 세 축에 대해 개별적인 아핀 변환을 거쳐 타원체로 변환된다. Argument card 문법을 따르며 입력 가능한 인수는 다음과 같다.

- name [str]: The name of surface primitive. The surface name must be unique.
- center [float]: The center coordinate of the ellipsoidal [cm].
- direction [float]: The direction vector of the ellipsoidal. Initial axis of rotation is transformed from z-axis to this variable. (Default={0,0,1})
- radius [float]: The radius of this sphere (x, y and z-axis) [cm].
- vertices [int]: Polygon resolution of this solid. (Default={100,100})

center 인수는 타원체의 중심을 설정한다. 이 인수는 필수로 요구된다. direction은 uv-sphere를 생성하는 회전축 벡터를 설정한다. 이 인수는 길이가 3이며 자동으로 정규화된다. radius는 타원체의 각 축에 대한 반지름을 설정한다. radius 인수는 세 개가 사용되고 각각 x, y, z축에 대한 반지름이 된다. vertices 인수는 두 개가 사용되고, 각각 polar 및 azimuthal 방향으로 균등 간격으로 회전하면서 꼭지점을 몇 개 생성할지 결정한다. 해당 카드로 생성한 타원체는 Uv-sphere에 affine 변환 카드를 적용하여 생성한 타원체와 동일하다.

5.3.5 CYLINDER

CYLINDER는 n각기둥 형태의 메시를 생성한다. Argument card 문법을 따르며 입력 가능한 인수는 다음과 같다.

- name [str]: The name of surface primitive. The surface name must be unique.
- center [float]: The center coordinate of the base of a cylinder [cm].

- radius [float]: The radius of the cylinder [cm].
- height [float]: The height vector of the cylinder [cm]. Length of the vector should be 3.
- vertices [int]: Polygon resolution of this solid. (Default=100)

center 인수는 원기둥 밑면의 중심을 설정한다. radius 인수는 원기둥 반지름을 설정한다. height 인수는 원기둥의 높이 벡터를 설정한다. 원기둥의 높이는 입력된 벡터들의 norm과 같고, 축이 해당 벡터와 같게 된다. 벡터의 norm은 0보다 커야 한다.

5.3.6 CONE

CONE은 n각뿔 형태의 메시를 생성한다. Argument card 문법을 따르며 입력 가능한 인수는 다음과 같다.

- name [str]: The name of surface primitive. The surface name must be unique.
- center [float]: The center coordinate of the base of a cylinder [cm].
- radius [float]: The radius of the cylinder [cm].
- height [float]: The height vector of the cylinder [cm]. Length of the vector should be 3.
- vertices [int]: Polygon resolution of this solid. (Default=100)

모든 인수는 CYLINDER 카드와 같다.

5.3.7 TORUS

Argument card 문법을 따르며 입력 가능한 인수는 다음과 같다.

- name [str]: The name of surface primitive. The surface name must be unique.
- center [float]: The center coordinate of co-planar circular axis [cm].
- direction [float]: The direction vector of this solid. Initial axis rotation is transformed from z-axis to this variable (Default={0,0,1}).
- radius [float]: The radius of co-planar circle (first) and revolving circle (second).
- vertices [int]: Polygon resolution of this solid. (Default=100)

radius 인수는 두 개를 요구하며, 각각 co-planar circle, revolving circle의 반지름을 의미한다. Revolving circle의 반지름은 co-planar circle의 반지름보다 작아야 하고, 그렇지 않을 경우 self-intersection으로 인해 오류가 발생한다.

앞서 설명한 primitive surface들의 예제는 [%MCRT2_HOME%/example/geometry/1_basic_solid]에 있으며 실행 결과는 다음과 같다.

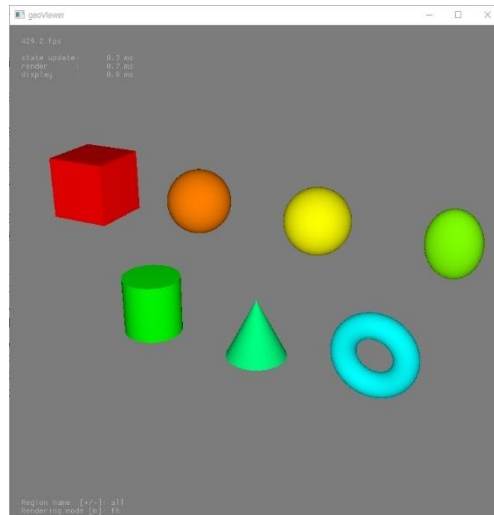


Figure 5. primitive surface들의 생성 예제

5.3.8 REVOLUTION

REVOLUTION 카드는 일반 회전체를 생성한다. Argument card 문법을 따르며 입력 가능한 인수는 다음과 같다.

- name [str]: The name of surface primitive. The surface name must be unique.
- entry [str]: The name of target coordinate entry. Entry list should have form of $\{x_1, z_1, x_2, z_2, \dots, x_n, z_n\}$. The length of entry list must even. The axis of rotation is z-axis, and the start and end points of the entry must be on the z-axis. The lines connecting each point in order must not intersect each other and must be arranged in a clockwise order.
- center [float]: The center coordinate of this solid [cm]. The 'entry' coordinates are the relative position of this center variable.
- direction [float]: The direction vector of this solid. Initial axis of rotation is transformed from z-axis to this variable. (Default={0,0,1})
- vertices [int]: Polygon resolution of this solid. (Default=100)

REVOLUTION 카드는 UVSPHERE와 동일한 알고리즘을 사용한다. 대신 회전체 단면의 형태를 자유롭게 설정할 수 있는데, entry 인수로 단면의 좌표를 지정할 수 있다. 좌표는 x축 위치, z축 위치의 짝으로 구성되어야 한다. 따라서 entry 인수의 리스트 길이는 짝수여야 한다. Entry의 시작과 끝 좌표는 z축 위에 놓여야 하며 좌표는 시계방향으로 나열되어야 한다. 또한 리스트를 이은 선들은 서로 만나지 않아야 한다. 그렇지 않을 경우 self-intersecting facet이 생성되어 오류가 발생할 수 있다.

REVOLUTION 카드로 회전체들을 생성하는 예제는 [%MCRT2_HOME%/example/geometry/4_solid_of_revolution]에 있으며 실행 결과는 다음과 같다.

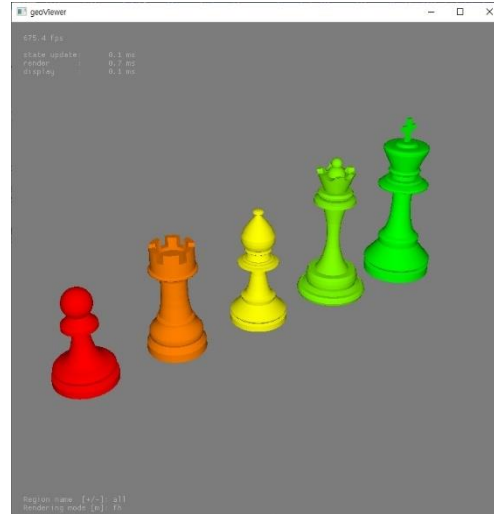


Figure 6. REVOLUTION 카드로 생성한 회전체의 예시

5.3.9 TOROID

TOROID 카드는 일반 원환면을 생성한다. Argument card 문법을 따르며 입력 가능한 인수는 다음과 같다.

- name [str]: The name of surface primitive. The surface name must be unique.
- entry [str]: The name of target coordinate entry. Entry list should have form of $\{x_1, z_1, x_2, z_2, \dots, x_n, z_n\}$. The length of entry list must even. The axis of rotation is z-axis, and the start and end points of the entry are connected. The lines connecting each point in order must not intersect each other and must be arranged in a clockwise order.
- center [float]: The center coordinate of this solid [cm]. The 'entry' coordinates are the relative position of this center variable.
- direction [float]: The direction vector of this solid. Initial axis of rotation is transformed from z-axis to this variable. (Default={0,0,1})
- vertices [int]: Polygon resolution of this solid. (Default=100)

TOROID 카드의 사용법은 REVOLUTION 카드와 거의 유사하지만, entry가 요구하는 좌표의 조건이 다르다. Entry는 이전과 마찬가지로 시계방향으로 나열되어야 하지만, entry의 시작점과 끝점은 z축 위에 있지 않아야 하며 entry의 모든 좌표는 z-axis를 통과하지 않아야 한다. Entry를 원형으로 이은 선분은 self-intersection이 발생하지 않아야 한다.

5.3.10 MODEL

MODEL 카드는 외부에서 3차원 메시 파일을 가져온다. Argument card 문법을 따르며 입력 가능한 인수는 다음과 같다.

- name [str]: The name of surface primitive. The surface name must be unique.
- file [str]: The name of 3D mesh file. Supported file extension are .off and .stl. Mesh data should not have self-intersecting facets and satisfy manifoldness.
- scale [float]: The origin mesh data is scaled by this factor. It requires three values, each a scaling factor for the x, y, and z axes. (Default={1.0,1.0,1.0})

외부 메시 파일은 off 혹은 stl 확장자를 지원하고, self-intersecting facets이 존재하지 않아야 하며 manifoldness를 만족해야 한다. 3차원 메시 파일의 경우 mm 단위로 작성된 경우가 많은데, scale 인수에 {0.1,0.1,0.1}을 입력하여 cm 단위로 변환할 수 있다.

MODEL 카드로 외부 메시 파일을 가져오는 것의 예시는 [%MCRT2_HOME%/example/geometry/5_model]에 있으며 실행 결과는 다음과 같다.

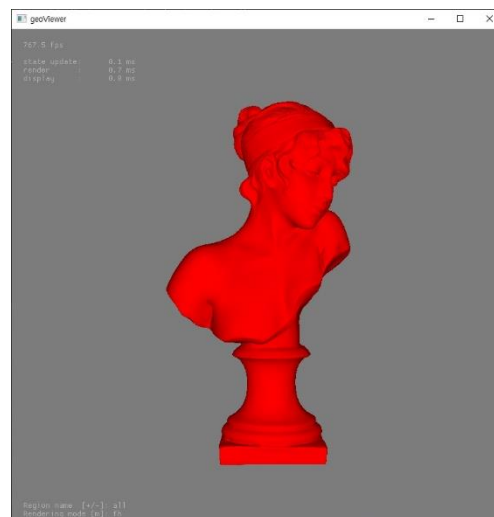


Figure 7. STL 파일 입력 예시

5.3.11 VOXEL

VOXEL 카드는 보조 Python 스크립트로 생성한 voxel 데이터를 입력하는 카드이다. Argument card 문법을 따르며 입력 가능한 인수는 다음과 같다.

- name [str]: The name of surface primitive. The surface name must be unique.
- file [str]: RT2 internal voxel file name. Compressed voxel file can be generated from DICOM or NIFTI file by using RT2dicom or RT2nifti.

Voxel 데이터를 생성하는 RT2dicom 및 RT2nifti Python 스크립트의 사용법에 관해서는 11.1와 11.2에서 확인할 수 있다. 주의할 점은 voxel 구조는 다른 surface에 의해 분할될 수 없다는 것이다. 이 구조는 다른 surface와 결합 구조를 생성할 수는 있지만 여집합 연산자에 의해 분할하면 오류가 발생할 수 있다.

5.4 Transform Cards

Transformation card는 surface 카드들로 생성한 메시들의 좌표를 rotating, translating, scaling 하기 위한 카드들이다. Affine transformation matrix를 기반으로 동작하며 nested transformation이 지원된다. Transformation은 Affine matrix를 정의하는 ROT_DEFI 카드와 변환 적용을 선언하는 TRANSFORM_BEGIN, 그리고 변환 적용 범위의 끝을 정의하는 TRANSFORM_END 카드로 구성된다.

5.4.1 ROT_DEFI

ROT_DEFI 카드는 변환에 필요한 Affine transformation matrix를 생성한다. Argument card 문법을 따르며 입력 가능한 인수는 다음과 같다.

- name [str]: The name of rotation definition. The name must be unique.
- axis [char]: Axis of rotation. This parameter should be 'x', 'y', or 'z'. (Default=z)
- theta [float]: Rotation azimuthal angle [degree]. (Default=0.0)
- delta [float]: Translation vector [cm]. (Default={0.0,0.0,0.0})
- affine [float]: 3x4 Affine transform matrix. If this option is used, final transformation matrix is overwritten by this parameter. The transformation matrix has following structure;

$$\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} = A \begin{pmatrix} x \\ y \\ z \end{pmatrix} + t = \begin{pmatrix} A_1 & A_2 & A_3 \\ A_5 & A_6 & A_7 \\ A_9 & A_{10} & A_{11} \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} + \begin{pmatrix} A_4 \\ A_8 \\ A_{12} \end{pmatrix}$$

affine 인수에는 길이가 12 인 벡터를 입력해야 하며 해당 벡터는 A₁ 부터 A₁₂ 까지의 매트릭스 원소를 의미한다. affine 인수의 초기값은 모두 0인 벡터이고, 하나 이상의 원소가 0이 아니면 입력된 Affine 매트릭스가 직접 사용된다. 그렇지 않을 경우 axis, theta, delta 값을 이용하여 Affine 매트릭스를 생성된다.

5.4.2 TRANSFORM_BEGIN / TRANSFORM_END

TRANSFORM_BEGIN 카드와 TRANSFORM_END 카드는 지정된 범위의 모든 surface에 특정한Affine 변환을 적용한다. TRANSFORM_BEGIN 카드는 argument card 문법을 따르며 입력 가능한 인수는 다음과 같다.

- target [str]: The name of target transformation definition.

target 인수에는 ROT_DEFI 카드로 정의한 변환의 이름을 입력해야 한다. TRANSFORM_BEGIN 카드로 시작된 변환은 SURFACE_END 카드를 만나기 전까지 짝이 맞는 TRANSFORM_END 카드로 끝나야 한다. 두 카드는 일종의 괄호 역할을 하며 사이에 있는 모든 카드들은 해당 변환을 거치게 된다. 입력문은 다음과 같은 형태를 가진다.

```
TRANSFORM_BEGIN target=something
    (... surface / transformation cards ...)
TRANSFORM_END
```

TRANSFORM_END 카드는 어떤 인수도 가지지 않는다. RT²에서는 nested transformation이 가능한데, 여러 개의 TRANSFORM 카드들로 감싸진 surface들에 대해 surface가 정의된 line에서 가장 가까운 변환부터 차례로 행해진다. 예시는 다음과 같다.

```

TRANSFORM_BEGIN target=X
    TRANSFORM_BEGIN target=Y
        CUBE name=cube center=0,0,0 size=10,10,10
    TRANSFORM_END
TRANSFORM_END

```

이 때 surface cube는 두 번의 Affine transform이 적용되는데, transform Y부터 적용되고 그 뒤 transform X가 적용된다. 변환 matrix 식은 다음과 같다.

$$\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} = A_X \left[A_Y \begin{pmatrix} x \\ y \\ z \end{pmatrix} + t_Y \right] + t_X$$

중첩 변환의 입력문 예시는 [%MCRT2_HOME%/example/geometry/2_transformation]에 있으며 실행 결과는 다음과 같다.

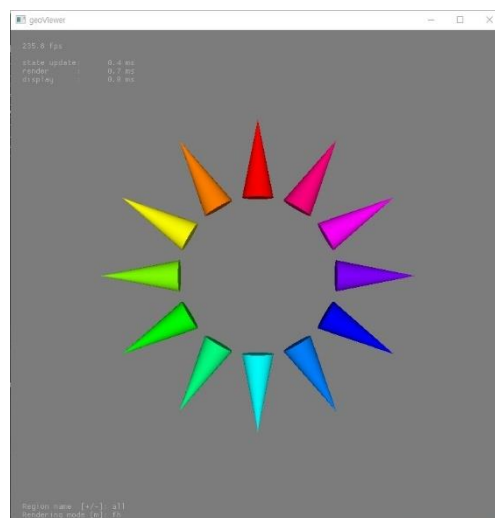


Figure 8. Nested transformation의 예제

여기서 원뿔들의 중심이 전체 geometry의 원점이며, 화면을 바라보는 쪽이 z축이다. 원본 원뿔(빨간색)을 z축을 중심으로 azimuthal 방향으로 30도 간격으로 회전하여 나열한다.

5.5 REGION

REGION 카드는 정의한 surface들을 조합하여 실질적으로 시뮬레이션에 사용되는 geometry를 생성한다. REGION 카드는 list card 형식을 따르며 문법은 다음과 같다.

```

REGION name element1 element2 ...

```

name은 region의 이름을 의미하며 region 리스트 내에서는 고유한 값을 가져야 한다. 이후 나타나는 모든 값은 element로 처리되며, string 타입을 가진다. Element끼리는 공백 문자로 구분될 필

요가 없고 빈칸 없이 붙여도 무방하다. 가능한 element들은 다음과 같다.

- 1) @ Nested region decorator. The next element is considered as region operand (if not, it is surface operand).
- 2) + Inside operator. The next operand indicates the interior of a surface or region (optional).
- 3) - Outside operator. The next operand indicates the exterior of a surface or region.
- 4) & And operator.
- 5) | Or operator.
- 6) (Parentheses begin operator. Paired end operator should exist. Nested parentheses are possible.
- 7)) Parentheses end operator. Paired begin operator should exist.
- 8) Else Name of surface/region operand.

해당 리스트에서 수식의 우선순위는 아래로 갈수록 높아진다. 예를 들어, - 연산자와 괄호 연산자가 연속하여 나타났을 경우 괄호 연산자가 우선 적용된다. 리스트 앞쪽 일곱 개의 연산자들은 수식 delimiter로 처리되며 해당 character들에 의해 분할된 string은 operand 이름으로 처리된다. Operand는 기본 상태에서는 surface 카드에서 정의된 surface 이름을 지정하며, 정의되지 않은 경우 오류가 발생한다. Operand 앞에 @ 연산자가 붙은 경우 해당 피연산자는 surface가 아닌 region을 지정하게 된다. 이 때 지정할 region은 현재 위치보다 앞에서 정의되어 있어야 한다. @ 연산자를 통해 nested definition이 가능하다. 다음은 해당 기능의 예시이다.

```
REGION reg1 surf1 | surf2
REGION reg2 surf3 & -@reg1
```

이 입력문은 nested definition이 해석되어 다음과 같은 입력문과 동일한 논리를 가진다.

```
REGION reg1 surf1 | surf2
REGION reg2 surf3 & -(surf1 | surf2)
```

5.6 Example of MCNP PRIMER

MCNP primer의 simple example of a cylindrical storage cask를 생성하는 입력문에 대해 다룬다. MCNP의 원본 geometry는 다음과 같다.

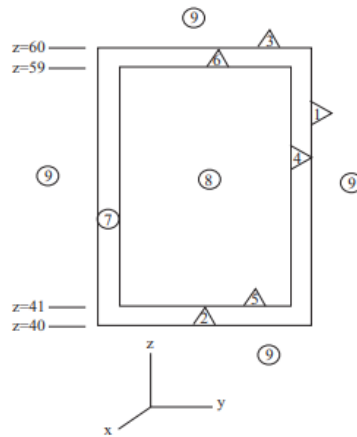


Figure 9. Geometry for a simple cylindrical cask. Numbers in triangles are surface identification numbers and numbers in circles define the cell identification number. The axis of the cask passes through the point ($x = 5$ cm, $y = 5$ cm, $z = 0$)

Figure 9 의 geometry는 다음과 같은 입력문으로 정의할 수 있다.

```
SURFACE_BEGIN
    CYLINDER name=inner_cask center=5,5,41 \
        height=0,0,18 radius=9
    CYLINDER name=outer_cask center=5,5,40 \
        height=0,0,20 radius=10
SURFACE_END

REGION inner_void +inner_cask
REGION cask_shell +outer_cask & -inner_cask
REGION outer_void -outer_cask
```

첫 번째 REGION 카드로 inner_void를 정의하고 해당 영역은 inner_cask surface의 안쪽으로 지정했다. 두 번째 REGION 카드는 and 연산자를 이용했는데, 앞서 설명했듯이 and 연산자보다 not 연산자가 우선도가 높기 때문에, inner_cask의 바깥쪽이 지정된다. inner_cask의 바깥쪽과 outer_cask의 안쪽을 동시에 만족하는 범위가 cask_shell이 된다. 마지막 카드로 outer_cask의 바깥쪽을 outer_void region으로 지정한다. 결과는 다음과 같다.

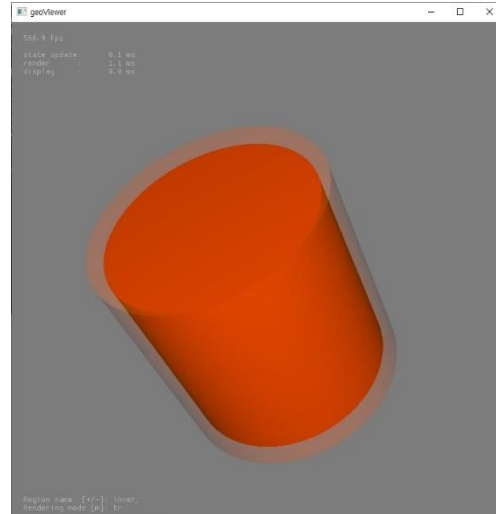


Figure 10. MCNP PRIMER 예제의 변환 결과

6 RT2mc

Geometry 생성 코드를 제외한 나머지 기능들은 RT2mc 코드로 설정하고, 시뮬레이션을 실행할 수 있다. 이하 내용에서는 RT2mc 코드에서 사용되는 card들을 설명한다.

6.1 Program Arguments

RT2mc는 RT2builder와 동일한 프로그램 실행 인수를 요구한다. 다음은 윈도우 10 환경에서 'RT2mc -h'를 입력했을 때 출력되는 결과물이다.

```

RT2mc -h
Parameters: --input      | -i <filename>      File for MC input
              --output   | -o <filename>      File for MC output
              --result    | -r <directory>    Where output & tally saved
              --debug     | -d                Print detail
              --syntax    | -s                Print MC parameter info
              --help      | -h                Print this message
  
```

Figure 11 RT2mc 프로그램의 입력 인수 목록

각각의 인수에 대한 설명은 5.1. 과 동일하다.

6.2 CUDA_SETTINGS

CUDA_SETTINGS 카드는 Monte Carlo 코드 실행 전반에 대한 GPU 환경 설정에 사용된다. Argument card 문법을 따르며 입력 가능한 인수는 다음과 같다.

- gpu [int]: The ID of the GPU on which the code run. GPU list can be found in command 'nvidia-

smi'. (Default=0)

- **block_dim [int]**: The number of threads in a single block. The total number of threads is product of the block_dim, block_per_sm, and the number of stream multiprocessor (SM). Here, the number of SM is determined by hardware specification. (Default=128)
- **block_per_sm [int]**: The number of blocks per stream multiprocessor. If the metric 'Volatile GPU-Util' in the 'nvidia-smi' is not saturated, block_dim should be increased to achieve peak performance. (Default=48)
- **buffer_ratio [float]**: The ratio of memory share of particle and interaction buffer per total GPU memory capacity. If the error caused by the buffer size is raised, this parameter should be increased. (Default=0.6)
- **block_decay_rate [float]**: The decay rate of the block of launched kernel at the residual stage. This parameter can be tuned heuristically to achieve peak performance. Generally, this parameter should have a value close to 1 when activating a computationally intensive interaction (e.g. Doppler broadened Compton scattering). (Default=0.2)
- **block_decay_limit [int]**: The lower bound on the number of the block in the launched kernel. When the decayed number of blocks become smaller than this limit, all remained phase-space of each buffer will be discarded. Peak performance can be achieved by increasing this parameter, as small kernel launch sizes degrade performance. However, accuracy may decrease because remaining particles are removed. (Default=1)
- **seed [int]** : Initial seed of the CURAND XORWOW random number generator. (Default=42)

시스템 상에 여러 개의 GPU가 장비되어 있을 경우 gpu 인수를 조정하여 코드를 실행할 GPU를 선택할 수 있다. GPU 장치 인덱스 및 종류는 터미널에 nvidia-smi 커맨드를 입력하여 확인할 수 있다. 다음은 두 개의 GPU가 장비되어 있는 시스템에서의 nvidia-smi 커맨드 실행 결과 예시이다.

```
> nvidia-smi
```

NVIDIA-SMI 535.183.01			Driver Version: 535.183.01			CUDA Version: 12.2		
GPU	Name	Perf	Persistence-M	Bus-Id	Disp.A	Volatile	Uncorr.	ECC
Fan	Temp		Pwr:Usage/Cap		Memory-Usage	GPU-Util	Compute M.	MIG M.
0	NVIDIA GeForce RTX 4090	P8	Off	00000000:67:00:0	Off	0%	Default	Off
0%	23C		13W / 450W		11MiB / 24564MiB		N/A	
1	NVIDIA GeForce RTX 4090	P8	Off	00000000:68:00:0	Off	0%	Default	Off
0%	23C		18W / 450W		60MiB / 24564MiB		N/A	
Processes:								
GPU	GI	CI	PID	Type	Process name	GPU Memory Usage		
ID	ID	ID						
0	N/A	N/A	1205	G	/usr/lib/xorg/Xorg			4MiB
1	N/A	N/A	1205	G	/usr/lib/xorg/Xorg			36MiB
1	N/A	N/A	1427	G	/usr/bin/gnome-shell			14MiB

Figure 12. nvidia-smi 커맨드 실행 결과의 예시

위쪽 테이블의 GPU 항목의 값이 입력문에 넣어야 할 index를 나타낸다.

block_dim 인수는 하나의 block에 할당될 thread의 개수를 의미한다. Nvidia GPU에서는 warp size가 32이기 때문에, 최적의 성능을 달성하기 위해서는 block_dim이 32의 배수가 되어야 한다. block_per_sm 인수는 하나의 steam multiprocessor(SM)에 할당될 block의 개수를 의미한다. GPU device마다 SM 개수는 하드웨어적으로 고정되어 있는데, 예를 들어 Nvidia RTX 4060 ti 카드의 경우 SM number가 34이며 Nvidia RTX 4090 카드의 경우 SM number가 128이다.

RT² 하위 커널들의 실행 dimension은 다음과 같은 식으로 결정된다.

$$Dimension = block \times thread = block_per_sm \times SM \times block_dim$$

예를 들어, RTX 4060 ti 카드에서 block_per_sm은 128, block_dim은 128로 설정하면 SM number가 34이기 때문에 커널의 dimension은 $(128 \times 34 \times 128) = (4352 \times 128)$ 이 된다. 해당 결과는 Monte Carlo 로그 파일의 World Summaries 항목에서 확인할 수 있는데, 다음은 로그 파일 [%MCRT2_HOME%/example/advanced/scintillator_detector/mc_sample.out]의 일부이다.

```
[INFO] *** World Summaries ***
Number of Facet                2424
Tracer Dimension               4352 x 128
GAS Size                      0.16272 (MiB)
Region Table Size             0.00924683 (MiB)
Tracer Size                    55.25 (MiB)
```

Figure 13 출력 로그 파일의 world summaries 예시

Tracer dimension이 계산한 결과와 동일한 것을 확인할 수 있다.

GPU 기반 코드들은 thread의 총 개수가 충분하지 않을 경우 peak performance를 달성하지 못할 수 있다. Figure 12 와 같이, 터미널에서 nvidia-smi 커맨드로 확인한 Volatile GPU-Util 값이 100%에 한참 못 미칠 경우 block_per_sm을 늘리면 peak performance를 달성할 수 있다.

6.3 Physics Cards

Physics 카드들은 interaction sampling에 사용할 모델, transport 및 production cutoff를 설정하는데 사용된다. 모든 카드들은 activate 옵션을 가지고 있는데, 이 옵션을 비활성화 할 경우 해당 종류의 입자는 수송 및 반응이 완전히 무시된다. 이 기능은 요구되는 buffer의 개수를 줄여 메모리를 확보하기 위함으로, EM 연산에서 중성자 및 중이온 수송을 비활성화 하는 등 특정 입자가 발생할 가능성이 전혀 없을 때 사용할 수 있다. Local deposit 처리도 되지 않기 때문에 그 외의 경우에는 unphysical 결과가 발생할 수 있다. 만약, 특정 입자에 대해 local deposit 처리를 하고 싶을 경우 activate 옵션을 비활성화 하는 대신 transport cutoff를 높게 설정하는 것이 추천된다.

6.3.1 PHOTON

PHOTON 카드는 광자와 관련된 physics 및 transport 옵션을 설정할 수 있다. Argument card 문법을 따르며 입력 가능한 인수는 다음과 같다.

- activate [bool]: Activate photon transport. (Default=true)
- library [str]: Photon cross-section library. (Default=epdl97)
- transport_cutoff [float]: Global photon transport cutoff [MeV]. (Default=0.01)
- production_cutoff [float]: Global photon production cutoff [MeV]. (Default=0.01)
- use_nrc_pair [bool]: Pair production sampling method. If true, NRC alias table is used for interactions under 85 MeV. Bethe-Heitler formula used otherwise. (Default=true)
- do_rayleigh [bool]: If false, Rayleigh scattering cross-section is set to 0 (deactivated). (Default=true)
- simple_photo [bool]: If true, simplified photoelectric algorithm is used. The shell structure is averaged and mean energy of electron is deposited at the creation site. (Default=false)
- sauter [bool] : If true, secondary electron direction is sampled from Sauter distribution. Otherwise, secondary inherit primary photon's direction. (Default=true)
- print_xs [bool] : Print detailed photon cross-section data for each material if this parameter is activated. (Default=true)
- nbin [int]: The number of log-log interpolation bins for the cross-section data. (Default=1000)
- compton_mode [str]: Compton scattering sampling method. Possible options are followed as below;
 - 1) simple Simple compton scattering. Using plain Klein-Nishina equation. (Default)
 - 2) egsnrc EGSnrc's bounded compton & doppler broadening method.
 - 3) livermore Geant4's bounded compton & doppler broadening method.

PHOTON 카드로 설정한 transport cutoff와 production cutoff는 모든 compound에 대해 전역으로 적용되어 특정 compound에 대해 PHOTON_CUTOFF 카드로 덮어씌워지지 않는다면 PHOTON 카드의 값이 사용된다. print_xs 옵션은 각 compound의 total cross section 및 각 interaction의 cross section을 출력하는 데 사용할 수 있다. 해당 기능의 사용 예시는 [%MCRT2_HOME%/example/physics /1_em_cross_section]에서 확인할 수 있다.

6.3.2 ELECTRON

ELECTRON 카드는 전자 및 양전자와 관련된 physics 및 transport 옵션을 설정할 수 있다. Argument card 문법을 따르며 입력 가능한 인수는 다음과 같다.

- activate [bool] : Activate electron transport. (Default=true)
- library [str]: Not used (dummy).
- transport_cutoff [float]: Global electron transport cutoff, kinetic energy [MeV]. (Default=0.1)
- production_cutoff [float]: Global electron production cutoff, kinetic energy [MeV]. (Default=0.1)

- `do_spin [bool]`: Spin correction flag. If true, spin correction will be applied in multiple elastic scattering. (Default=true)
- `use_presta2 [bool]`: PRESTA-II algorithm flag. The PRESTA-II CH algorithm is utilized if true. Otherwise, the PRESTA-I CH algorithm will be used. (Default=true)
- `ie_fudge [bool]`: If true, SLAC-265 Berger & Seltzer's fudge is used to calculate mean ionization energy. (Default=true)
- `print_xs [bool]`: Print detailed electron cross-section data for each material if this parameter is activated. (Default=false)
- `n_brem_split [int]`: The number of bremsstrahlung photon splitting. (Default=5)
- `nbin [int]`: The number of log-log interpolation bins for the cross-section data. (Default=1000)
- `fudgems [float]`: Parameter for sub-threshold inelastic contribution. Check EGSnrc's \$FUDGEMS for more detail. (Default=1.0)
- `smax [float]`: Global maximum step-size restriction for electron and positron transport [cm]. (Default=5.0)
- `estep [float]`: Maximum fractional energy loss per step. (Default=0.2)
- `ximax [float]`: Maximum first elastic scattering moment per step. (Default=0.5)
- `brem_corr [str]`: Bremsstrahlung correction method. Possible options are followed as below;
 - 1) `km` Koch & Motz empirical correction.
 - 2) `lcru` NIST/ICRU correction. (Default)
 - 3) `off` Turn off bremsstrahlung correction.
- `brem_xs [str]`: Cross-section generating method for bremsstrahlung. Possible options are followed as below;
 - 1) `bh` Bethe & Heitler cross section.
 - 2) `nist` NIST cross section database (from ICRU radiative stopping power).
 - 3) `nrc` NIST data including corrections for electron-electron. (Default)
- `brem_angle [str]`: Sampling method for bremsstrahlung secondary X-ray angle. Possible options are followed as below;
 - 1) `simple` Use leading term of Koch & Motz 2BS formula. (Default)
 - 2) `km` Use Koch & Motz 2BS formula.
 - 3) `off` Secondary photon inherits primary electron's direction.
- `eii_mode [str]`: Electron Impact Ionization (EII) mode. Possible options are followed as below;
 - 1) `off` EII sampling algorithm is not used. (Default)
 - 2) `kawrakow` Kawrakow's EII theory is used.
 - 3) `gryzinski` Gryzinski's semi-classical theory is used.
 - 4) `casnati` Casnati's empirical fit to measured data is used.
 - 5) `kolbenstvedt` Kolbenstvedt's revised theory used.
 - 6) `penelope` Bote and Salvat theory, used in PENELOPE's EII database.

Cutoff 관련한 코드 동작은 PHOTON 카드와 동일하다.

6.3.3 POSITRON

양전자와 관련된 모든 인수는 ELECTRON 카드로 정의되어, POSITRON 카드로는 양전자 비활성화만 가능하다. Argument card 문법을 따르며 입력 가능한 인수는 다음과 같다.

- activate [bool] : Activate positron transport. (Default=true)
- library [str]: Not used (dummy).
- transport_cutoff [float]: Not used (dummy).
- production_cutoff [float]: Not used (dummy).

6.3.4 VACANCY

VACANCY 카드는 atomic relaxation 과정과 관련된 값들을 결정한다. Argument card 문법을 따르고 입력 가능한 인수는 다음과 같다.

- activate [bool] : Activate atomic relaxation. (Default=true)
- library [str]: Atomic relaxation library. (Default=eadl)
- transport_cutoff [float]: Not used (dummy).
- production_cutoff [float]: Global vacancy production cutoff [MeV]. (Default=0.01)
- local_deposit [bool]: Potential energy of vacancy is always deposited it's all energy on the creation site if true. (Default=false)
- print_detail [bool]: If true, transition branches and shell structures of all elements are printed. (Default=false)

Atomic relaxation은 광자 하나와 vacancy 하나가 생성되는 radiative, 전자 하나와 vacancy 두개가 생성되는 Auger로 나뉜다. 두 반응에서 생성될 수 있는 photon, electron, vacancy는 각각의 global production cutoff에 따라 cutoff 이상이면 해당 입자가 생성되지만, cutoff 미만이면 입자가 생성되지 않고 local deposit 처리된다. 이와 관련된 예제는 [%MCRT2_HOME%/example/physics/4_relaxation_structure]에서 확인할 수 있다.

6.3.5 NEUTRON

NEUTRON 카드는 20 MeV 미만의 저 에너지 중성자 수송과 관련된 값들을 다룬다. Argument card 문법을 따르고 입력 가능한 인수는 다음과 같다.

- activate [bool] : Activate low energy neutron transport. (Default=true)
- library [str]: Low energy neutron cross-section library. (Default=endf7_260)
- transport_cutoff [float]: Not used (dummy).
- production_cutoff [float]: Not used (dummy).
- print_xs [bool] : Print detailed neutron cross-section data and sampling table for each material and element if this parameter is activated. (Default=false)

중성자의 경우 transport cutoff 및 production cutoff을 지정할 수 없다. 20 MeV 이상의 고에너지 중성자 수송은 NEUTRON_HIGH 카드로 다룰 수 있다. PHOTON, ELECTRON 카드와 동일하게 print_xs 옵션은 각 compound의 total cross section 및 사용된 각 element의 reaction branch들을 출력하는 데 사용할 수 있다. 해당 기능의 사용 예시는 [%MCRT2_HOME%/example/physics/2_low_neutron_cross_section]에서 확인할 수 있다.

6.3.6 NEUTRON_HIGH

NEUTRON_HIGH 카드는 20 MeV 이상의 고 에너지 중성자 수송과 관련된 값들을 다룬다. Argument card 문법을 따르고 입력 가능한 인수는 다음과 같다.

- activate [bool]: Activate high energy neutron transport.
- library [str]: Not used (dummy).
- transport_cutoff [float]: Global high energy neutron transport cutoff [MeV].
- production_cutoff [float]: Not used (dummy).

고에너지 중성자의 경우 model 기반으로 동작하기 때문에 library 및 production cutoff를 설정할 수 없다.

6.3.7 DELTA

DELTA 카드는 고에너지 이온에 의한 delta-ray production과 관련된 값들을 다룬다. Argument card 문법을 따르고 입력 가능한 인수는 다음과 같다.

- activate [bool]: Activate delta-ray production. (Default=true)
- library [str]: Not used (dummy).
- transport_cutoff [float]: Not used (dummy).
- production_cutoff [float]: Hadron-delta production cutoff [MeV]. (Default=0.3)
- free_scatter [bool] : If true, hadron-electron scattering considered as free electron. Otherwise, orbital binding effect is considered. (Default=false)

Delta-ray production에 의해 발생하는 2차 입자는 전자이므로, transport cutoff를 설정할 수 없다. Production cutoff는 delta-ray cross-section을 계산하는 데 사용되는데 너무 낮은 값을 사용할 경우 cross-section이 매우 커져 연산시간이 급격하게 길어진다.

6.3.8 GENERIC_ION

GENERIC_ION 카드는 모든 종류의 charged hadron의 수송을 다룬다. RT²에서는 charge number 8, mass number 18까지의 안정한 원소를 수송 가능하다. 해당 카드는 Argument card 문법을 따르고 입력 가능한 인수는 다음과 같다.

- activate [bool] : Activate generic ion transport. (Default=true)
- library [str]: Not used (dummy).
- transport_cutoff [float]: Global ion transport cutoff, kinetic energy [MeV/u]. (Default=2.0)

- production_cutoff [float]: Not used (dummy).
- nbin [int]: The number of log-log interpolation bins for the cross-section data. (Default=1000)
- lat_disp [bool]: If true, lateral displacement is considered in condensed-history step. (Default=true)
- lat_alg96 [bool]: If true, Urban ALG96 lateral displacement sampling algorithm is activated. (Default=true)
- straggling [bool] : If true, energy loss straggling is considered. (Default=true)
- smax [float]: Global maximum step-size restriction for ion transport [cm]. (Default=1e+10)
- estepe [float]: Maximum fractional energy loss per step. (Default=0.05)

중성자 수송 문제 등에서 발생하는 저 에너지의 가벼운 입자를 계산하는 데는 estepe 변수가 큰 값이 사용되어도 괜찮지만, 고에너지 양성자 혹은 중이온이 primary인 문제에서는 estepe 변수의 값을 줄여야 정상적인 결과를 얻을 수 있다. 이러한 종류의 문제에서 높은 estepe 값을 사용하고, 동시에 scoring mesh 사이즈가 작을 경우 계단 형식의 energy deposition curve가 관찰될 수 있다. 해당 예시는 [%MCRT2_HOME%/example/physics/6_generic_ion_stepsize]에서 확인할 수 있다.

6.3.9 ION_INELASTIC

ION_INELASTIC 카드는 원자핵-원자핵 혹은 고 에너지 중성자-원자핵 반응의 모델 및 primary 테이블을 생성하는데 사용된다. Argument card 문법을 따르고 입력 가능한 인수는 다음과 같다.

- activate [bool]: Not used (dummy).
- library [str]: Not used (dummy).
- transport_cutoff [float]: Not used (dummy).
- production_cutoff [float]: Not used (dummy).
- mode_high [str]: Ion-nuclear reaction model for the high energy ion (> 70 MeV/u). Possible options are followed as below;
 - 1) off Turn off the ion-nuclear reaction. If this option is chosen, low energy ion-nuclear reaction model is inactivated also. (Default)
 - 2) abrasion Wilson abrasion model.
 - 3) qmd Relativistic Jaeri quantum molecular dynamics.
- mode_low [str]: Ion-nuclear reaction model for the high energy ion (< 70 MeV/u). Possible options are followed as below;
 - 1) bme Boltzmann master equation. Not supported now.
 - 2) incl Compound nucleus formation model of Liege Intra Nuclear Cascade. (Default)
- evap_cutoff [float]: Evaporation time cutoff. If the lifetime of projectile ion is shorter than this parameter and the excitation energy is zero, unstable-breakup model is applied. Otherwise, projectile ion is considered as stable nuclei. Note that the in-flight de-excitation is not supported in RT². (Default=1e-8)

- activate_fission [bool]: Activate competitive fission evaporation branch. (Default=true)
- abrasion_ce [bool]: If true, n-body energy conservation algorithm is applied to abrasion model. Otherwise, n-body momentum conservation algorithm is applied. Note that this option valid only in abrasion mode. (Default=false)
- abrasion_nn [bool]: If this parameter is activated, nucleon-nucleon scattering algorithm is applied. Note that this option valid only in abrasion mode. (Default=false)
- use_exact_level [bool]: Not supported now.

evap_cutoff 인수는 안정한 원자핵의 리스트를 생성하는 데 사용된다. Lifetime이 해당 인수보다 작은 모든 원자핵은 안정하다고 가정되고, 해당 원자핵들은 generic ion에서 가능한 projectile로 인식한다. Excitation energy가 0이고 lifetime이 0이면 evaporation 과정이 종료된다.

70 MeV/u 이상의 고에너지 이온은 Wilson abrasion 혹은 QMD 모델을 사용할 수 있다. QMD 모델은 느리지만 상대적으로 실험값과 가깝고, Wilson abrasion 모델은 많은 approximation이 적용되어 빠르지만 상대적으로 부정확하다. 적용하는 문제가 정밀도/속도 중 어느 것을 중요하게 요구하는지에 따라 선택할 수 있다.

6.3.10 INCL_SETTINGS

ION_INELASTIC 카드에서 mode_low 인수에 incl을 선택했을 때 사용되는 카드이다. INCL 모델 내부 parameter를 조정하는 데 사용된다. Argument card 문법을 따르고 입력 가능한 인수는 다음과 같다.

- use_real_mass [bool]: If true, real proton/neutron mass is used to calculate NN cross-section and distance. Otherwise, INCL hadron mass is used. (Default=true)
- ignore_coulomb [bool]: If true, Coulomb deviation is ignored. (Default=false)
- rp_correlation_p [float]: Position and momentum correlation of proton. (Default=0.5)
- rp_correlation_n [float]: Position and momentum correlation of neutron. (Default=0.73)
- fermi_momentum [str]: Fermi momentum calculation algorithm. Possible options are followed as below;
 - 1) constant Fermi momentum is constant to all nucleus. (Default)
 - 2) constant_light Heavy nucleus ($A > 12$) have constant fermi momentum. Pre-defined momentum is used to light nucleus.
 - 3) mass_dependent Using equation to calculate fermi momentum.
- incl_dump_action [bool]: If this parameter is activated, INCL model dump is printed. (Default=false)
- incl_dump_size [bool]: The snapshot size of INCL dump. (Default=1000)

6.3.11 CUTOFF

CUTOFF 카드는 각 compound에 입자의 transport and production cutoff를 개별적으로 설정하는 데 사용된다. CUTOFF 카드는 CUTOFF_* 형식을 따른다. 예를 들어, CUTOFF_PHOTON은 해당 compound의 photon cutoff를 설정하고, CUTOFF_ELECTRON은 electron cutoff를 설정한다. 이 카

드는 Argument card 문법을 따르고 입력 가능한 인수는 다음과 같다.

- target [str]: The name of target compound.
- transport_cutoff [float]: Transport cutoff of this component in the target compound [MeV or MeV/u].
- production_cutoff [float]: Production cutoff of this component in the target compound [MeV or MeV/u].

CUTOFF 카드로 특정되지 않은 compound는 전역 cutoff를 사용하게 되는데, 이는 PHOTON, ELECTRON 등의 카드로 설정된 값이다. 다음은 CUTOFF 카드를 이용하여 서로 다른 compound에 대해 다른 cutoff를 적용하는 예시이다.

```
PHOTON transport_cutoff=0.01

CUTOFF_PHOTON target=water transport_cutoff=0.1
CUTOFF_PHOTON target=carbon transport_cutoff=0.001
```

해당 예시에서 전역 transport cutoff는 0.01 MeV로 지정되고, water와 carbon에 대한 transport cutoff는 각각 0.1 MeV와 0.001 MeV로 지정된다. 만약 입력문에 다른 compound가 더 있을 때에는 이 두 compound를 제외한 모든 compound에 대해 transport cutoff가 0.01 MeV로 지정된다.

6.4 Material Cards

Material 카드들은 compound를 정의하고, REGION 카드로 생성한 region들에 정의된 compound들을 할당하는 데 사용된다. Material 카드들은 순서에 상관없이 input의 아무 위치에나 작성하면 된다.

6.4.1 COMPOUND

COMPOUND 카드는 연산에 필요한 매질을 정의한다. List card 문법을 따르며 문법은 다음과 같다.

```
COMPOUND name density za/compound1 fraction1 ...
```

각각의 component들의 설명은 다음과 같다.

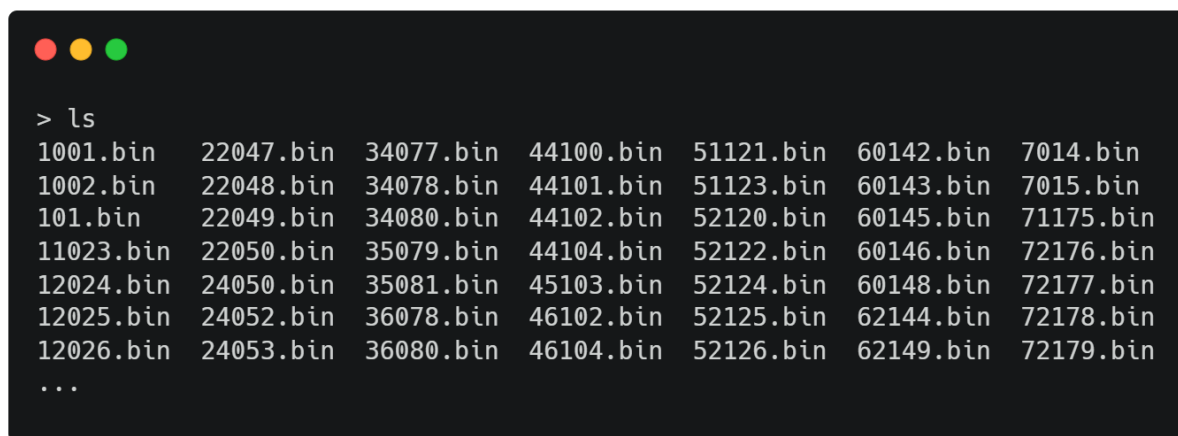
- name [str]: The name of compound. The compound name must be unique.
- density [float]: The density of compound [g/cm³]. The value must be positive or zero. If the density is zero, this compound assumed as vacuum.
- za/compound# [str]: Component of compound. If the value is integer, isotope is inserted. Integer value format is 1000*Z + A. If the value is string, compound previously defined is inserted. Nested definition is possible.
- fraction# [float]: Fraction of component. If the value is positive, the weight fraction is applied. Otherwise, the number fraction is applied. The mixed definition (number and mass) is not

allowed. Note that the list element of component and fraction should be paired. Fractions of all components are normalized in the end.

za/compound와 fraction은 짝을 이뤄야 하고, 길이의 제한은 없다. 변수 fraction은 음수일 경우 atomic number fraction으로 처리되고, 양수일 경우 weight fraction으로 처리된다. 하나의 COMPOUND 카드 내에서 음수 fraction과 양수 fraction이 동시에 사용될 수 없다 za/compound 리스트가 없고 밀도가 0일 경우 해당 물질은 진공으로 처리된다. 다음은 진공을 정의하는 입력문의 예시이다.

```
COMPOUND vacuum 0.0
```

Material을 지정할 때 ZA number를 이용하여 element들을 가져와 사용할 수 있다. 가능한 ZA number들은 디렉토리 %MCRT2_HOME%/resource/neutron/\$library/]에서 확인할 수 있다. \$library는 **NEUTRON** 카드에서 정의한 library 값이다. 다음은 기본값인 endf7_260을 사용할 때 library 폴더 내의 파일들의 리스트를 나타낸다.



```
> ls
1001.bin  22047.bin  34077.bin  44100.bin  51121.bin  60142.bin  7014.bin
1002.bin  22048.bin  34078.bin  44101.bin  51123.bin  60143.bin  7015.bin
101.bin   22049.bin  34080.bin  44102.bin  52120.bin  60145.bin  71175.bin
11023.bin 22050.bin  35079.bin  44104.bin  52122.bin  60146.bin  72176.bin
12024.bin 24050.bin  35081.bin  45103.bin  52124.bin  60148.bin  72177.bin
12025.bin 24052.bin  36078.bin  46102.bin  52125.bin  62144.bin  72178.bin
12026.bin 24053.bin  36080.bin  46104.bin  52126.bin  62149.bin  72179.bin
...
```

Figure 14 RT²에서 사용 가능한 원자 라이브러리 목록의 일부

여기서 확장자를 제외한 숫자를 사용할 수 있다. 예를 들어 1001은 ¹H 원소를 나타내며 7014는 ¹⁴N 원소를 나타낸다. 101의 경우 thermal neutron scattering을 위한 라이브러리로, S(a,b) 테이블이 적용된 water-bound hydrogen을 나타낸다.

Hydrogen 및 oxygen 동위원소들을 무시할 때 water는 다음과 같은 입력문으로 정의할 수 있다.

```
COMPOUND water 1.0 101 0.1119 8016 0.8881
```

Fraction weight는 마지막에 전체 합이 1로 정규화 되기 때문에 위의 입력문은 다음과 동일하다.

```
COMPOUND water 1.0 101 11.19 8016 88.81
```

Atomic number fraction을 사용하면 위의 입력문은 다음과 동일하다.


```
COMPOUND water 1.0 101 -2 8016 -1
```

RT²에서는 nested compound definition을 지원한다. 해당 기능으로 이전에 정의된 compound를 조합하여 새로운 compound를 정의할 수 있다. 다음은 염분 농도가 3.5%인 해수를 정의하는 입력문의 예시이다. 해수의 다른 구성 요소는 고려하지 않았다.

```
COMPOUND hydrogen 1e-8 101 99.9855 1002 0.0145
COMPOUND oxygen 1e-8 8016 99.8 8017 0.2
COMPOUND water 1.0 hydrogen -2 oxygen -1

COMPOUND sodium 0.969 11023 100.0
COMPOUND chlorine 1e-8 17035 76.0 17037 24.0
COMPOUND salt 2.17 sodium -1 chlorine -1

COMPOUND seawater 1.03 water 96.5 salt 3.5
```

해당 입력문의 실행 예제는 [%MCRT2_HOME%/example/material/1_seawater]에서 확인할 수 있다. 출력 log 파일의 Compound Summaries 항목에서 정의된 material의 물리량 및 구성 정보를 확인할 수 있는데 위의 입력문으로 정의된 seawater의 경우 다음과 같다.

```
[INFO] *** Compound Summaries ***

...

[ seawater ]
Density 1.03 (g/cm3)
Effective Z 3.41227
Effective A 6.17592
Ionization energy 77.2765 (eV)
Radiation length 34.2505 (cm)

...

<Composition>
Isotope Mass [Dalton] Mass Fraction [%] Number Fraction [%]
H-1 1.0078 10.7978 66.1685
H-2 2.0141 1.5659e-03 4.8016e-03
O-16 15.9949 85.5293 33.0244
O-17 16.9991 0.1714 6.2272e-02
Na-23 22.9895 1.3774 0.3700
Cl-35 34.9688 1.6132 0.2849
Cl-37 36.9659 0.5094 8.5111e-02
```

Figure 15 output log 파일에 출력된 seawater에 대한 물리량 및 구성 정보

6.4.2 MAT_PROP

MAT_PROP 카드는 특정한 compound의 mean ionization energy를 임의의 값으로 설정하기 위해 사용된다. Argument card 형식을 따르고, 요구되는 인수는 다음과 같다.

- compound [str]: The name of target compound.
- mean_ie [float]: User-specified mean ionization energy [eV]. The equation-derived energy will be overwritten by this value.

MAT_PROP 카드가 사용되지 않은 compound들은 equation으로 얻은 mean ionization energy를 사용한다. **Figure 15**와 같이 해당 값은 log 파일에서 확인할 수 있다.

6.4.3 ASSIGN

ASSIGN 카드는 COMPOUND 카드로 정의한 물질을 각 region에 할당하는 데 사용된다. Argument card 형식을 따르고 요구되는 인수는 다음과 같다.

- region [str]: The name of target region.
- compound [str]: The name of assigned compound.

예를 들어, **6.4.1** 에서 정의한 seawater를 target이라는 이름을 가진 region에 정의하기 위한 입력문은 다음과 같이 작성할 수 있다.

```
ASSIGN region=target compound=seawater
```

Monte Carlo 코드가 정상적으로 실행되기 위해서는 한 region에 대해 ASSIGN 카드가 두 번 이상 사용되지 않아야 하고, 아무런 material이 지정되지 않은 region이 존재하지 않아야 한다. 예를 들어 다음과 같은 입력문은 오류가 발생한다.

```
ASSIGN region=target compound=seawater
ASSIGN region=target compound=water
```

6.5 Source Cards

Source card는 특정한 형태의 primary beam을 정의한다. RT²에서는 FLUKA의 BEAM, BEAMPOS 카드, MCNP의 SDEF 카드와 동일한 역할을 수행하는 macro source card와 기록된 데이터를 사용하는 phase-space card 두 가지를 지원한다.

Macro source card를 사용할 경우 서로 다른 종류를 여러 개 정의할 수 있고, 이 경우에 각각이 샘플링 될 확률은 beam weight에 비례하게 된다. Phase-space source card는 한 개만 정의할 수 있고, macro source card와 동시에 사용하는 것은 허용되지 않는다.

Source 카드들은 공통적으로 입자의 종류를 정의하는 pid 인수를 요구한다. 다음은 RT²에서 설정할 수 있는 primary particle type의 종류의 목록이다. 해당 값들은 scoring 카드에서도 동일하게 사용된다.

Table 1 Source 카드에서 사용 가능한 pid 목록

ID	Type	Description
-1	Electron	
0	Photon	
1	Positron	
6	Low-energy neutron	Up to 20 MeV
7	High-energy neutron	From 20 MeV
26	Generic ion	For the scoring
ZZZAAA	Generic ion	$Z * 1000 + A$, for the beam

Low-energy neutron과 high-energy neutron은 각각의 에너지 영역에서 샘플링 방법이 다르기 때문에, 정상적인 동작을 위해서는 에너지 영역과 pid 인수를 구분해 주어야 한다. Generic ion은 ZA number를 사용하여 입력할 수 있는데, 예를 들어 1001은 proton, 6012는 ^{12}C ion을 의미한다. 사용 가능한 동위원소의 리스트는 log 파일의 [*** Generic Ion Transport Summaries ***] 항목에서 확인할 수 있다.

RT²에서는 source 컨트롤 카드, 세 개의 macro source card, 한 개의 phase-space source card를 제공하며 이하에서는 문법과 설명 및 사용 예시를 보인다.

6.5.1 BEAM_GLOBAL

BEAM_GLOBAL 카드는 사용할 beam의 종류 및 계산할 primary의 총 개수를 설정한다. Argument card 문법을 따르고 입력 가능한 인수는 다음과 같다.

- mode [str]: Mode of the primary beam. Possible options are followed as below;
 - 1) macro Macrobody beam.
 - 2) ps Phase-space beam.
- nps [int]: The total number of primary of this run. (Default=1e+6)

BEAM_GLOBAL 카드는 입력문에 한 개 이상 존재해야 하며, 여러 개 존재할 경우 가장 마지막에 등장한 카드만 적용된다. macro 옵션을 선택할 경우 macro beam 카드들만 사용되게 되고, phase-space beam 카드들은 무시된다. 반면 phase-space 옵션을 선택할 경우 phase-space beam 카드만 사용되게 되고, macro beam 카드들은 무시된다. 이 경우에는 phase-space 카드는 입력문 전체에 단 한 개만 존재해야 한다.

6.5.2 BEAM_CUBIC

BEAM_CUBIC 카드는 육면체 형태의 source를 정의한다. Argument card 문법을 따르고 입력 가능한 인수는 다음과 같다.

- pid [int]: Primary particle ID. (Default=1)
- name [str]: Name of this macro beam.

- weight [float]: The weight of this macro beam. (Default=1)
- position [float]: Center of cubic beam [cm]. (Default=0,0,0)
- direction [float]: Direction of beam axis vector. (Default=0,0,1)
- rotation [float]: Azimuthal angle of beam rotation along beam axis [degree]. (Default=0)
- type [str]: Energy binning type of this macro beam. Possible options are followed as below;
 - 1) linear The sampled primary energy is linearly uniform in a single energy bin. (Default)
 - 2) log The sampled primary energy is log uniform in a single energy bin.
- energy [str]: Energy spectrum entries. First and second entry present energy bin [MeV] and probability. If the length of energy bin and the length of probability are same, energy spectrum is interpreted as discrete spectrum. In other case, if the length of the energy bin is one more than the length of the probability, energy spectrum is interpreted as histogram spectrum.
- divergence [str]: Beam divergence spectrum entries. First and second entry present directional cosine bin [$\cos\theta = \mu$] and probability. If the length of divergence bin and the length of probability are same, divergence is interpreted as discrete spectrum. In other case, if the length of the divergence bin is one more than the length of the probability, divergence is interpreted as histogram spectrum.
- size [float]: The size of cubic beam. The length of size entry is three. Each element present x, y, and z size of the cubic beam [cm]. These values should be positive or zero. If one of them is zero, the primary beam is rectangular source. In a case of two, the primary beam is line source. If all of them are zero, the primary beam is considered as the point source.

pid 인수는 **Table 1** 에 정의된 값들 중 하나를 선택해서 사용해야 한다. weight 인수는 여러 개의 macro beam을 설정할 때 해당 macro beam 정의에서 primary가 샘플링될 확률을 의미한다. 예를 들어, 다음과 같은 macro beam 정의들이 있다면 beam1과 beam2에서 샘플링될 확률은 각각 60%, 40%가 된다.

BEAM_CUBIC	name=beam1	\
	pid=0	\
	weight=0.6	\
	...	
BEAM_CUBIC	name=beam2	\
	pid=-1	\
	weight=0.4	\
	...	

weight는 마지막 처리 과정에서 normalize 되므로 비율만 의도한 대로 정의하면 된다. 여러 개의 beam을 정의하고 결과를 확인하는 입력문의 예시는 [%MCRT2_HOME%/example/source /1_isotropic_two]에 있으며 실행 결과는 다음과 같다.

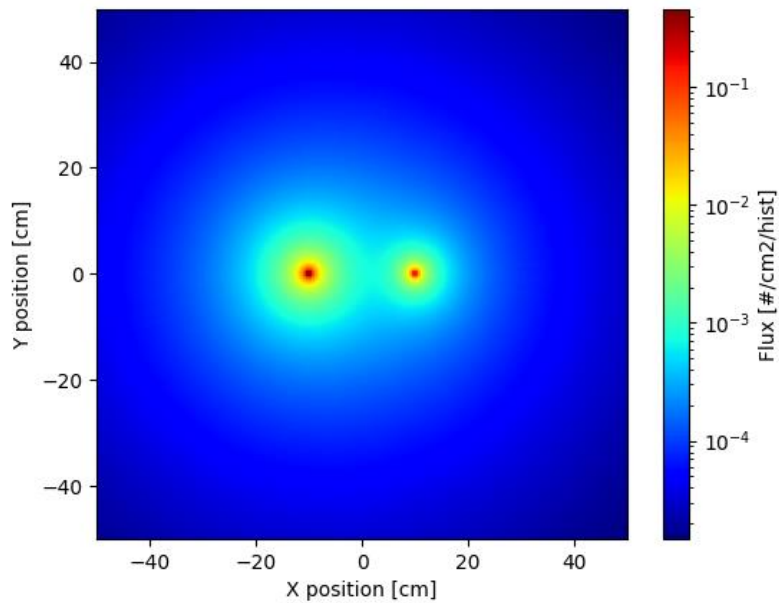


Figure 16. 서로 다른 weight을 가진 source들의 생성 예제

energy와 divergence 인수는 각각의 스펙트럼 혹은 히스토그램을 정의하는 데 사용된다. 여기에는 **ENTRY** 카드로 정의한 리스트를 두 개 입력해야 한다. 첫 번째 entry는 샘플링될 에너지나 각도의 abscissa, 두 번째 entry는 확률을 정의한다. 두 entry 길이가 똑같으면 discrete spectrum으로 정의되고 첫 번째 entry가 두 번째 entry 길이보다 하나 길면 histogram으로 정의된다. 첫 번째 entry는 반드시 오름차순으로 입력되어야 한다.

다음은 histogram과 discrete energy spectrum을 정의하는 입력문의 예시이다.

```
# histogram for p1
ENTRY ebin1 .1 .3 .5 1. 2.5 # bins
ENTRY eprob1 .2 .4 .3 .1 # probability

# discrete for p2
ENTRY ebin2 .3 .5 .9 1.25 # peak
ENTRY eprob2 .2 .1 .3 .4

ENTRY dbin -1 1 # isotropic
ENTRY dprob 1
```

첫 번째 entry들은 길이가 하나 차이가 나기 때문에 histogram으로 정의되고, 두 번째 entry들은 길이가 같기 때문에 discrete spectrum으로 정의된다. 해당 entry들은 BEAM_CUBIC 카드의 energy 인수에 입력하여 사용할 수 있다. 예시는 다음과 같다.

BEAM_CUBIC	name=p1	\
	pid=0	\
	weight=0.4	\
	energy=ebin1, eprob1	\
	divergence=dbin, dprob	\
	...	
BEAM_CUBIC	name=p2	\
	pid=0	\
	weight=0.6	\
	energy=ebin2, eprob2	\
	divergence=dbin, dprob	\
	...	

이 입력문에서는 40% 확률로 histogram으로 정의된 p1 source가 사용되고, 60% 확률로 discrete spectrum으로 정의된 p2 source가 사용된다. 에너지 스펙트럼을 얻은 결과는 다음과 같다. 해당 입력문 및 관련 스크립트는 [%MCRT2_HOME%/example/source/14_isotropic_two_spect]에서 확인할 수 있다.

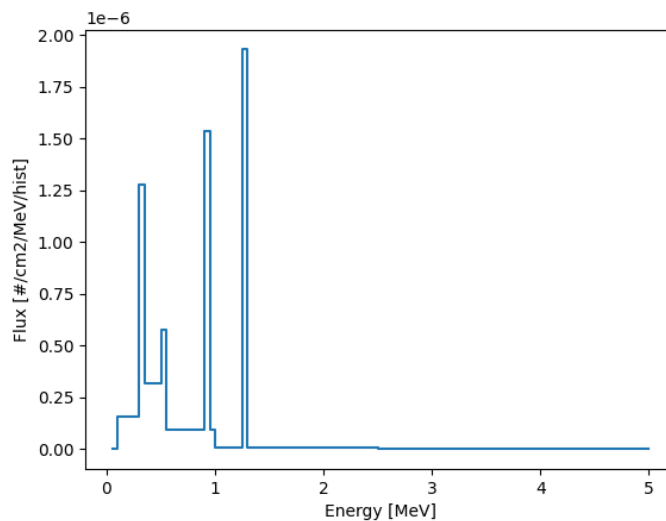


Figure 17 서로 다른 spectrum 형태를 가진 source들의 생성 예시

divergence는 macro beam의 축에 대해 얼마만큼의 각도를 가지는지를 결정한다. Macro beam의 축은 direction 인수로 정의할 수 있고, cubic beam의 경우 cube의 z축 방향이 direction 방향으로 align된다. 모든 beam은 azimuthal uniform하다고 간주된다. divergence 값은 -1부터 1까지 범위 이내여야 한다. 다음은 isotropic beam과 cone beam을 위한 divergence entry들의 예시이다.

```
ENTRY dbin_i -1 1 # isotropic
ENTRY dprob_i 1
ENTRY dbin_c 0.9 1 # cone
ENTRY dprob_c 1
```

Divergence로 첫 번째 entry들을 사용할 경우 $[\cos\theta = \mu]$ 값이 -1부터 1까지에서 linear uniform하게 되고 macro beam은 isotropic 형태가 된다. 두 번째 entry들을 사용할 경우 $[\cos\theta = \mu]$ 값이 0.9에서 1까지 linear uniform 하게 된다. 여기서 $\arccos(0.9) = 25.8^\circ$ 이므로 direction 벡터 방향으로 해당 각도를 가지는 원뿔 형태의 beam이 정의된다. 관련 입력문 및 스크립트는 [%MCRT2_HOME%/example/source/11_diverged_conebeam]에서 확인할 수 있다.

6.5.3 BEAM_CYLINDER

BEAM_CYLINDER 카드는 원기둥 형태의 source를 정의한다. Argument card 문법을 따르고 다른 모든 인수는 **BEAM_CUBIC** 카드에서 사용하는 것과 동일하지만, size 인수 대신 다음 두 개의 인수가 사용된다.

- radius [float]: The radius of cylindrical beam. The length of radius entry is two. First and second element present the inner and outer cylindrical shell radius [cm] respectively. Here, the first value must be same or smaller than the second value. These values should be positive or zero. If both are zero, the primary beam is considered as the line source.
- height [float]: The height of cylindrical beam. If this value is zero, the primary beam is considered as the two-dimensional annular source. If both of radius and height are zero, the primary beam is considered as the point source.

height 인수의 값을 0으로 설정할 경우 degenerated된 annular source가 정의된다. 해당 입력문의 예시는 [%MCRT2_HOME%/example/source/7_isotropic_disk]에서 확인할 수 있다. height 인수는 position 인수로 정의되는 위치에서부터 양쪽 방향으로의 높이를 의미한다. **CYLINDER** 카드와는 정의가 다른 점을 주의해야 한다. 예를 들어 다음과 같은 입력문은 $z=0$ 부터 $z=10$ 까지의 높이를 가지는 cylindrical beam을 정의한다.

BEAM_CYLINDER	name=p1	\
	pid=0	\
	position=0,0,5	\
	direction=0,0,1	\
	height=10	\
	...	

6.5.4 BEAM_SPHERE

BEAM_SPHERE 카드는 구 형태의 source를 정의한다. Argument card 문법을 따르고 다른 모든 인수는 **BEAM_CUBIC** 카드에서 사용하는 것과 동일하지만, size 인수 대신 다음 인수가 사용된다.

- radius [float]: The radius of sphere beam. The length of radius entry is two. First and second element present the inner and outer shell radius [cm] respectively. Here, the first value must be same or smaller than the second value. These values should be positive or zero. If both are zero, the primary beam is considered as the point source.

radius 인수의 값을 {0,0}으로 설정할 경우 degenerated된 point source가 정의된다. radius 인수의 첫 번째 값을 0으로 설정할 경우 일반적인 volumetric sphere beam이 정의되고, 첫 번째 값이 0이 아닐 경우 hollow sphere beam이 정의된다. 해당 입력문의 예시는 [%MCRT2_HOME%/example /source/9_isotropic_hollowsphere]에서 확인할 수 있다.

6.5.5 BEAM_PS

BEAM_PS 카드는 **BEAM_GLOBAL** 카드에서 ps 옵션을 선택했을 때 요구된다. 외부 파일에 저장된 phase-space 데이터를 가져와 primary beam을 샘플링한다. 해당 카드는 입력문 전체에 한 개만 존재해야 한다. Argument card 문법을 따르고 인수는 한 개만 요구한다.

- file [str]: Phase-space binary file. File should have unformatted Fortran binary type, with special data alignment.

Phase-space 데이터는 이후 scoring card 섹션에서 설명할 **PHASE_SPACE** 카드로 저장할 수도 있고, Python interface로 직접 생성할 수도 있다. 예제 파일과 관련된 Python interface는 **10.2.7** 에서 확인할 수 있다.

6.6 Scoring Cards

Scoring 카드는 각 입자들의 flux, energy deposition 등의 물리량을 저장하는 데 사용된다. MCNP의 tallies, FLUKA의 scoring에 대응된다. RT²에서는 다섯 개의 flux, energy deposition 혹은 dose를 계산할 수 있는 일반 scoring 카드, Pulse-height distribution을 측정할 수 있는 **DETECTOR** 카드, phase-space 데이터를 저장할 수 있는 **PHASE_SPACE** 카드를 지원한다. 이 중 마지막 두 카드는 사용할 경우 메모리 요구량이 커지며 성능 저하가 크기 때문에 주의하여 사용할 필요가 있다.

6.6.1 DENSITY

DENSITY 카드는 특정한 region 내의 물리량을 계산한다. Argument card 문법을 따르며 입력 가능한 인수는 다음과 같다.

- name [str]: The name of tally. This variable must be unique.
- pid [int]: Particle ID of tally filter.
- type [str]: Type of scoring quantity. Possible options are followed as below;
 - 1) dose The energy deposition per unit mass [MeV/g/hist].
 - 2) depo The energy deposition per unit volume [MeV/cm³/hist].
- where [str]: The name of target region of this tally.

다음은 'target' region에 전달된 electron에 의한 dose를 계산하는 입력문의 예시이다.

```
ASSIGN region=target compound=water
...
DENSITY name=eden pid=-1 type=dose where=target
```

계산된 데이터는 프로그램 실행 위치에 %a%_%b%.den 파일로 저장되고, %a%와 %b%는 각각 입력문의 파일 이름, tally에 지정된 이름이다. 저장된 파일은 Python interface로 데이터 및 uncertainty를 추출할 수 있다.

6.6.2 TRACK

TRACK 카드는 특정한 region 내에서 방사선 입자의 track-length 기반 fluence를 계산한다. Argument card 문법을 따르고 입력 가능한 인수는 다음과 같다.

- name [str]: The name of tally. This variable must be unique.
- pid [int]: Particle ID of tally filter.
- escale [str]: Energy binning type of this tally. Possible options are followed as below;
 - 1) linear The energy bin is linearly uniform.
 - 2) log The energy bin is log uniform.
- erange [float]: Lower and upper bound of energy bin [MeV]. Upper bound should be larger than lower bound.
- ne [int]: The number of energy bin. This value must be positive.
- where [str]: The name of target region of this tally.

다른 인수들은 **DENSITY** 카드와 똑같지만, 에너지 히스토그램 구조를 결정하는 escale, erange와 ne가 추가된다. Log uniform energy bin을 사용할 경우 erange의 첫 번째 값은 0보다 커야 한다. 계산된 데이터는 프로그램 실행 위치에 %a%_%b%.trk 파일로 저장되고, %a%와 %b%는 각각 입력문의 파일 이름, tally에 지정된 이름이다.

6.6.3 CROSS

CROSS 카드는 두 region의 경계를 통과하는 입자의 flux를 계산한다. Argument card 문법을 따르고 입력 가능한 인수는 다음과 같다.

- name [str]: The name of tally. This variable must be unique.
- pid [int]: Particle ID of tally filter.
- escalate [str]: Energy binning type of this tally. Possible options are followed as below;
 - 1) linear The energy bin is linearly uniform.
 - 2) log The energy bin is log uniform.
- erange [float]: Lower and upper bound of energy bin [MeV]. Upper bound should be larger than lower bound.
- ne [int]: The number of energy bin. This value must be positive.
- from [str]: The name of region before the boundary.
- to [str]: The name of region after the boundary.

다른 모든 인수는 **TRACK** 카드와 똑같지만, where 인수 대신에 from과 to 인수가 추가되었다. 해당 인수들은 측정할 boundary crossing event의 region 경계를 설정한다. 양방향 scoring은 불가능하고, from과 to의 순서가 의도에 맞게 배치되어야 한다. 계산된 데이터는 프로그램 실행 위치에 %a_%b%.crs 파일로 저장되고, %a%와 %b%는 각각 입력문의 파일 이름, tally에 지정된 이름이다.

6.6.4 MESH_DENSITY

MESH_DENSITY 카드는 육면체 voxel 형태로 물리량을 계산한다. Argument card 문법을 따르고 입력 가능한 인수는 다음과 같다.

- name [str]: The name of tally. This variable must be unique.
- pid [int]: Particle ID of tally filter.
- xrange [float]: Lower and upper bound of mesh bin in x-axis [cm]. Upper bound should be larger than lower bound.
- yrange [float]: Lower and upper bound of mesh bin in y-axis [cm].
- zrange [float]: Lower and upper bound of mesh bin in z-axis [cm].
- nx [int]: The number of mesh bin along x-axis. This value must be positive.
- ny [int]: The number of mesh bin along y-axis. This value must be positive.
- nz [int]: The number of mesh bin along z-axis. This value must be positive.
- type [str]: Type of scoring quantity. Possible options are followed as below;
 - 1) dose The energy deposition per unit mass [MeV/g/hist].
 - 2) depo The energy deposition per unit volume [MeV/cm³/hist].

해당 카드는 FLUKA의 usrbn, MCNP의 meshtal 등의 tally와 비슷한 역할을 수행한다. 다음은 한

변이 20cm인 육면체 voxel 내의 전자에 의한 dose를 계산하는 입력문의 예시이다.

```
$SIZE$ 10
$NBIN$ 100
...
MESH_DENSITY name=vdose pid=-1 \
              xrange=-$SIZE$, $SIZE$ \
              yrange=-$SIZE$, $SIZE$ \
              zrange=-$SIZE$, $SIZE$ \
              nx=$NBIN$ ny=$NBIN$ nz=$NBIN$ \
              type=dose
```

이 입력문의 실행 결과로 모든 축에 대해 -10 cm부터 10 cm까지 100개의 bin이 있는 육면체 voxel tally가 설정된다.

계산된 데이터는 프로그램 실행 위치에 %a%_%b%.mdn 파일로 저장되고, %a%와 %b%는 각각 입력문의 파일 이름, tally에 지정된 이름이다. MESH_DENSITY tally에 지나치게 많은 bin을 설정할 경우 메모리 제한 문제로 코드가 실행되지 않을 수 있다.

6.6.5 MESH_TRACK

MESH_TRACK 카드는 육면체 voxel 형태로 track-length를 이용한 fluence를 계산한다. Argument card 문법을 따르고 입력 가능한 인수는 다음과 같다.

- name [str] : The name of tally. This variable must be unique.
- pid [int] : Particle ID of tally filter.
- xrange [float]: Lower and upper bound of mesh bin in x-axis [cm]. Upper bound should be larger than lower bound.
- yrange [float]: Lower and upper bound of mesh bin in y-axis [cm].
- zrange [float]: Lower and upper bound of mesh bin in z-axis [cm].
- nx [int]: The number of mesh bin along x-axis. This value must be positive.
- ny [int]: The number of mesh bin along y-axis. This value must be positive.
- nz [int]: The number of mesh bin along z-axis. This value must be positive.
- escale [str]: Energy binning type of this tally. Possible options are followed as below;
 - 1) linear The energy bin is linearly uniform.
 - 2) log The energy bin is log uniform.
- erange [float]: Lower and upper bound of energy bin [MeV]. Upper bound should be larger than lower bound.

- ne [int]: The number of energy bin. This value must be positive.

다른 모든 인수는 **MESH_DENSITY** 카드와 같고, 에너지 구조를 설정하기 위한 escale, erange, ne 인수가 추가되었다. **MESH_DENSITY** 카드와 마찬가지로 너무 많은 geometry bin이나 energy bin 개수를 설정할 경우 메모리 제한으로 인해 코드가 실행되지 않을 수 있다. 계산된 데이터는 프로그램 실행 위치에 %a%_%b%.mtr 파일로 저장되고, %a%와 %b%는 각각 입력문의 파일 이름, tally에 지정된 이름이다.

6.6.6 DETECTOR

DETECTOR 카드는 설정된 region에서 모든 종류의 방사선 입자에 대한 pulse-height distribution을 측정한다. 앞의 tally들과 다르게 입자의 종류를 설정할 수 없다. 또한 DETECTOR 카드는 한 입력문에 최대 10개로 제한되고 메모리 점유율이 상대적으로 높다. 한 개 이상의 DETECTOR 카드가 입력될 경우 secondary particle들의 종속 관계를 모두 추적하기 때문에 큰 성능 저하가 발생할 수 있다. Argument card 문법을 따르고 입력 가능한 인수는 다음과 같다.

- name [str]: The name of tally. This variable must be unique.
- pid [int]: Not used (dummy).
- escale [str]: Energy binning type of this tally. Possible options are followed as below;
 - 1) linear The energy bin is linearly uniform.
 - 2) log The energy bin is log uniform.
- erange [float]: Lower and upper bound of energy bin [MeV]. Upper bound should be larger than lower bound.
- ne [int]: The number of energy bin. This value must be positive.
- where [str]: The name of target region of this tally.
- a [float]: Gaussian energy broadening parameter [MeV].
- b [float]: Gaussian energy broadening parameter [MeV^{1/2}].
- c [float]: Gaussian energy broadening parameter [MeV⁻¹].

인수 a, b, c는 Gaussian energy broadening 알고리즘에 사용되는데, 하나의 primary에 대한 energy deposition은 다음 full width at half maximum (FWHM) 값을 가지는 gaussian 분포에 의해 샘플링된다.

$$FWHM (MeV) = a + b\sqrt{E} + cE^2$$

다음은 DETECTOR 카드를 이용하여 NaI crystal에 대한 pulse-height distribution를 계산한 결과의 예시이다. 해당 예시의 입력문은 [%MCRT2_HOME%/example/scoring/4_detector]에서 확인할 수 있다.

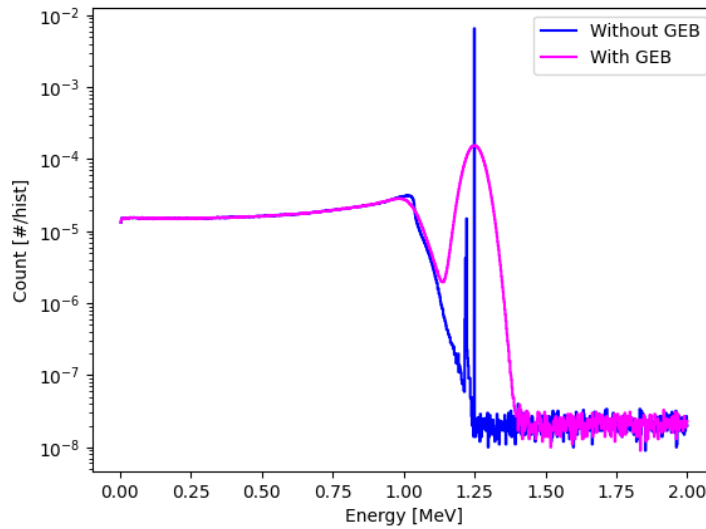


Figure 18 DETECTOR 카드에서 GEB parameter 입력에 따른 pulse-height distribution의 차이

계산된 데이터는 프로그램 실행 위치에 %a%_%b%.det 파일로 저장되고, %a%와 %b%는 각각 입력문의 파일 이름, tally에 지정된 이름이다.

6.6.7 PHASE_SPACE

PHASE_SPACE 카드는 설정된 두 region의 경계에서 boundary crossing event가 일어난 방사선 입자에 대한 phase-space를 기록한다. 해당 카드는 입자의 종류를 설정할 수 없다. Argument card 문법을 따르고 입력 가능한 인수는 다음과 같다.

- name [str]: The name of tally. This variable must be unique.
- from [str]: The name of region before the boundary.
- to [str]: The name of region after the boundary.
- buffer_size [int]: The maximum number of scored phase-space. (Default=1e+6)

buffer_size 인수는 기록될 phase-space 개수의 상한선을 설정한다. Phase-space는 해당 개수 이상으로 기록되지 않고 나머지는 무시된다. 지나치게 높은 값을 설정할 경우 메모리 제한에 의해 코드가 실행되지 않을 수 있다.

기록된 데이터는 프로그램 실행 위치에 %a%_%b%.phs 파일로 저장되고, %a%와 %b%는 각각 입력문의 파일 이름, tally에 지정된 이름이다. 저장된 phase-space 파일은 다른 입력문에서 **BEAM_PS** 카드를 이용하여 primary beam으로 설정할 수 있다. 해당 기능을 이용하여 two-step variation reduction을 수행할 수 있고, 관련된 입력문의 예시는 [%MCRT2_HOME%/example /advance/linear_accelerator/two_step]에서 확인할 수 있다.

7 RT2viewer

RT2builder 어플리케이션으로 생성한 geometry를 확인할 수 있는 viewer 프로그램이다.

RT2viewer는 Windows 명령 프롬프트 혹은 Linux 터미널에서 RT2viewer 명령어를 실행하여 열 수 있다. 이 프로그램은 GLFW 라이브러리를 사용하기 때문에 가상 환경에서는 실행이 되지 않을 수 있다.

실행 인수로 -h를 추가로 입력하면 사용할 수 있는 인수의 리스트와 설명을 확인할 수 있다. 다음은 윈도우 10 환경에서 'RT2viewer -h'를 입력했을 때 출력되는 결과물이다.

```

> RT2viewer -h
Parameters: --input      | -i <filename>      Geometry file
            --resolution | -r <int2>          Renderer resolution
            --eye        | -e <float3>        Camera eye position [cm]
            --look       | -l <float3>        Camera look-at position [cm]
            --fov        | -f <float>         Camera field of view [degree]
            --help       | -h                Print this message

```

Figure 19 RT2viewer의 입력 인수 목록

생성된 geometry 바이너리 파일은 작업 위치의 cache 폴더에 저장되며 --input 인수로 지정할 수 있는 geometry 파일 이름은 입력하지 않을 시 해당 폴더 하의 바이너리 파일을 가리키게 된다. 뷰어의 초기 해상도는 --resolution 인수로 지정할 수 있다. 해당 인수는 두 개의 integer를 지정해야 하며 각각 뷰어 창의 가로 픽셀 길이, 세로 픽셀 길이를 의미한다. --eye, --look, --fov 인수는 렌더링 dimension을 지정하는 데 사용되며 구조는 다음과 같다.

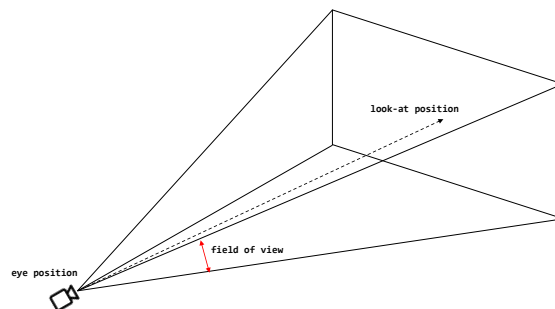


Figure 20 RT2viewer의 카메라 구조

여기서 field of view는 vertical FOV를 의미하게 된다. 코드 기본 설정으로 해상도는 768x768, 카메라 위치는 z축 방향으로 10m 높이에서 원점을 바라보도록 되어 있으며 FOV는 35도로 세팅되어 있다.

프로그램은 첫 번째 GPU를 활용하여 시각화 이미지를 생성한다. 따라서 첫 번째 GPU에서 RT2mc 프로그램을 실행하는 도중에 뷰어를 실행하면 Monte Carlo 성능이 저하될 수 있다. RT2viewer는 트랙볼 움직임을 지원하여 마우스 왼쪽 클릭 후 드래그시 look at position을 고정하고 카메라를 회전한다. 마우스 오른쪽 클릭 후 드래그시 카메라를 고정하고 look at position을 회

전한다. 마우스 휠을 조정하면 현재 카메라의 축을 기준으로 카메라를 앞뒤로 움직인다.

키보드 'm'을 입력하면 렌더링 모드를 변경할 수 있는데, 기본값으로 first-hit rendering 모드로 설정되어 있다. 이 경우 현재 카메라 위치에서 첫 번째로 만나는 region 경계를 렌더링한다. 두 번째 렌더링 모드는 transparent로, 표면을 투명화 처리하여 내부 region을 보여준다. 키보드 '-'혹은 '='를 입력하면 타겟 region을 조정할 수 있다. 기본값으로 모든 region을 렌더링하도록 설정되어 있고 키보드 입력으로 원하는 region만 렌더링하도록 조정할 수 있다. 다음은 특정 region을 first-hit 모드 그리고 transparent 모드로 렌더링한 결과이다.

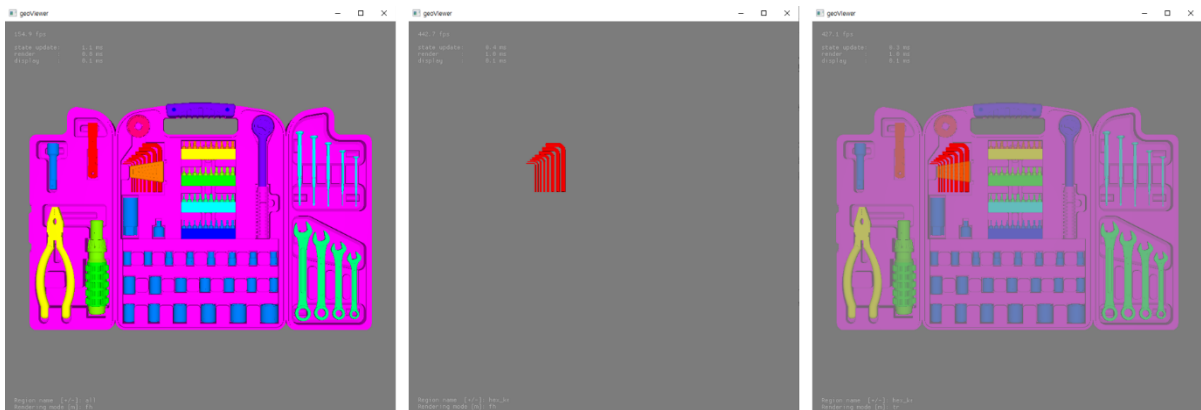


Figure 21 RT2viewer 렌더링 결과 예시

8 RT2interactive

특정한 문제(방사선 치료 등)에서는 다른 모든 환경은 동일하게 유지하고, primary beam의 조건만 바꾸어 여러 번 연산해야 하는 경우가 있다. 일반적인 Monte Carlo 알고리즘에서는 한 번의 연산이 완료되면 프로그램이 종료되고, 다음 연산을 위해 다시 실행하는 과정에서 데이터 전처리가 반복적으로 발생하게 된다. RT²는 이러한 종류의 문제에서 성능 최적화를 위해 primary beam을 제외한 모든 메모리 구조 및 데이터를 유지하고 반복적으로 계산하는 RT2interactive 프로그램을 제공한다.

RT2interactive 프로그램은 환경변수가 설정되어 있을 경우 터미널에서 직접 실행할 수 있다. 실행 인수로 -h를 추가로 입력하면 사용할 수 있는 인수의 리스트와 설명을 확인할 수 있다. 실행 인수는 Figure 11과 같이 RT2mc 프로그램과 동일하다.

입력문은 RT2mc와 동일하다. RT2mc와 다르게 RT2interactive는 실행했을 때 Monte Carlo loop가 실행되지 않고, 데이터 전처리만 진행된다. 데이터 전처리 과정이 끝났을 때 터미널에 Ready 문자열이 출력된다. 다음은 실행 예시이다.

```

> RT2interactive -i mc.inp
[2025-01-02 15:08:09]
[INFO] Read input file stream 'mc.inp'

...

[INFO] *** Buffer Summaries ***
Available memories          15001 (MiB)
Memory usage ratio         60 (%)
Number of buffer            16
Size per buffer             16154624 (Banks)

Ready

```

Figure 22 RT2interactive 프로그램의 실행 결과 예시

Ready 문자열이 출력된 후 RT2interactive 프로그램은 입력 스트림 대기상태로 들어가게 되고, 여기서 입력할 수 있는 인수는 다음과 같다.

```

...
Ready
> -h
Parameters: --input   | -i <filename>   File for MC input
             --output  | -o <filename>   File for MC output
             --result   | -r <directory> Where output & tally saved
             --debug    | -d             Print detail
             --exit     | -e             Exit the interactive mode
             --help     | -h             Print this message

```

Figure 23 전처리가 완료된 RT2interactive 프로그램의 입력 가능한 인수

--exit 인수를 입력하면 RT2interactive 프로그램이 즉시 종료되고, 사용된 메모리는 모두 해제된다. --input 인수를 통해 추가적인 입력문을 읽을 수 있고, 이 입력문에서는 beam 정보만 읽어 이전에 처리된 geometry, material, 및 physics 환경 하에서 연산을 수행한다. 다음은 추가적인 입력문을 읽어 Monte Carlo 연산을 진행했을 때 터미널에 출력된 결과의 예시이다.


```

...
Ready
> -i interact_child.inp
[2025-01-02 15:28:36]
[INFO] Read input file stream 'interact_child.inp'
...
[      depo_posit ]
Part                Positron
Origin              -20,-20,-20 (cm)
Size                0.4,0.4,0.4 (cm)
Shape               100,100,100
Total volume        64000 (cm3)
Quantity            E depo (MeV/cm3/hist)
Maximum depo at    (0,0,0)
with                0 (MeV/cm3/hist)
finished

```

Figure 24 RT2interactive 반복 작업 실행의 예시

연산이 완료되면 관련된 로그 파일 및 tally 데이터가 작업 공간에 저장되고, 저장이 완료되면 터미널에 'finished' 문자열이 출력된다. 해당 문자열이 출력되면 프로그램은 다시 입력 스트림 대기상태로 들어가게 되고 다른 beam 구조를 가진 입력문을 읽어 반복적인 연산이 가능하다. 해당 과정은 Python subprocess 모듈 혹은 shell script를 활용하여 자동화할 수 있다. Python script를 활용하여 자동화하여 반복 연산을 수행하는 입력문의 예시는 [%MCRT2_HOME%/example/advance/interactive]에서 확인할 수 있다.

9 RT2moduleTester

디버깅 & physics 확인 용도 프로그램, 추가예정

10 Python Interface

Phase-space 데이터, dump 파일, tally 데이터 등의 RT²에서 사용되는 binary 형태의 데이터를 해석하고 처리하기 위해 Python으로 개발된 코드들이 제공된다. 해당 코드들은 RT² 빌드 과정에서 rt2 패키지로 현재 사용하는 Python3 pip에 설치된다. 이하 내용에서는 RT² 시스템에서 사용할 수 있는 Python module들을 설명한다.

10.1 rt2.hadron

hadron 모듈은 RT² 알고리즘 내에서 적용되는 nuclear mass data를 확인하기 위해 사용된다.

10.1.1 class NuclearMassData()

- **Attributes**

- None

- **Methods**

- `getMass()`: Get the table mass of nucleus if exist. Otherwise, compute nuclear mass by using Weizsaecker's formula [MeV/c²].

10.2 rt2.scoring

scoring 모듈은 **Scoring Cards** 에서 얻을 수 있는 binary 데이터를 해석하거나 병합하는 데 사용된다.

10.2.1 class Density(file_name)

● Attributes

- `file_name` [str]: DENSITY type binary tally file.
- `part` [str]: Unit of scored values.
- `data` [float]: Scored value, single variable.
- `unc` [float]: Uncertainty of this tally.

● Methods

- `summary()`: Print tally information.
- `write(file_name)`: Save current tally data to binary file.
- operator [+,-,/,*]: Do arithmetic operations for other constants or tallies that have same 'Density' class type. The opposing operand should be 'Density' object or numeric value. The error-propagation is applied automatically to calculate net uncertainty.

네 가지 연산자들을 이용하여 다른 density tally나 상수와 연산을 수행할 수 있다. 다음은 서로 다른 tally의 값을 병합하는 Python script의 예시이다.

```
from rt2.scoring import Density

tally1 = Density('foo1.den') # Unit is MeV/cm3/hist
tally2 = Density('foo2.den')

tally_sum = tally1 + tally2
tally_GeV = tally1 / 1e+3
```

Figure 25 DENSITY tally에 대한 연산 예시

첫 번째 연산으로 서로 다른 두 tally가 병합되고, 두 번째 연산으로 tally 데이터가 1000으로 나누어진다. 두 번째 tally는 단위가 [MeV/cm³/hist]에서 [GeV/cm³/hist]로 바뀔 것이라는 것을 알 수 있다.

10.2.2 class Track(file_name)

● Attributes

- `file_name` [str]: TRACK type binary tally file.
- `part` [str]: Unit of scored values.

- data [np.ndarray(float32)]: Scored value, 1D Numpy array.
- unc [np.ndarray(float32)]: Uncertainty of this tally, 1D Numpy array.
- **Methods**
 - summary(): Print tally information.
 - write(file_name): Save current tally data to binary file.
 - operator [+,-,/,*]: Do arithmetic operations for other constants or tallies that have same 'Track' class type. The opposing operand should be 'Track' object or numeric value. If opposing operand is 'Track' object, energy bin structure must be same to apply arithmetic operation. The error-propagation is applied automatically to calculate net uncertainty.
 - etype(): Get the type of energy binning ['linear' or 'log']
 - ebin(): Get the Numpy array of energy bin.
 - eindex(energy): Get the energy bin index of specific energy [MeV].

ebin method와 matplotlib 라이브러리를 활용하여 TRACK tally 데이터의 히스토그램을 그릴 수 있다. 다음은 관련된 Python script의 예시이다.

```
import matplotlib.pyplot as plt
from rt2.scoring import Track

tally = Track('foo.trk')

plt.figure()
plt.step(tally.ebin()[1:], tally.data)
plt.show()
```

Figure 26 TRACK tally의 스펙트럼 plot 예시

10.2.3 class Cross(file_name)

- **Attributes**
 - file_name [str]: CROSS type binary tally file.
 - part [str]: Unit of scored values.
 - data [np.ndarray(float32)]: Scored value, 1D Numpy array.
 - unc [np.ndarray(float32)]: Uncertainty of this tally, 1D Numpy array.
- **Methods**
 - summary(): Print tally information.
 - write(file_name): Save current tally data to binary file.
 - operator [+,-,/,*]: Do arithmetic operations for other constants or tallies that have same 'Cross' class type. The opposing operand should be 'Cross' object or numeric value. If opposing operand is 'Cross' object, energy bin structure must be same to apply arithmetic

operation. The error-propagation is applied automatically to calculate net uncertainty.

- `etype()`: Get the type of energy binning ['linear' or 'log']
- `ebin()`: Get the Numpy array of energy bin.
- `eindex(energy)`: Get the energy bin index of specific energy [MeV].

Cross 클래스의 구조는 Track 클래스와 동일하다.

10.2.4 class MeshDensity(file_name)

● Attributes

- `file_name [str]`: CROSS type binary tally file.
- `part [str]`: Unit of scored values.
- `data [np.ndarray(float32)]`: Scored value, 3D Numpy array.
- `unc [np.ndarray(float32)]`: Uncertainty of this tally, 3D Numpy array.

● Methods

- `summary()`: Print tally information.
- `write(file_name)`: Save current tally data to binary file.
- `operator [+,-,/,*]`: Do arithmetic operations for other constants or tallies that have same 'MeshDensity' class type. The opposing operand should be 'MeshDensity' object or numeric value. If opposing operand is 'MeshDensity' object, voxel dimension and affine matrix must be same to apply arithmetic operation. The error-propagation is applied automatically to calculate net uncertainty.
- `affine()`: Get the affine matrix, 2D Numpy array [3x4].
- `inverse()`: Get the affine matrix for the inverse transformation [3x4].
- `transform(other)`: Transform tally mesh coordinate by input affine matrix 'other'.
- `translate(x, y, z)`: Translate tally mesh coordinate by input vector (x, y, z) [cm].
- `scale(x, y, z)`: Scale tally mesh bins by input parameters (x, y, z).
- `rotate(theta, axis)`: Rotate tally mesh with polar angle 'theta' [degree] along the 'axis' [0=x, 1=y, 2=z].
- `origin()`: Get the coordinate of left-bottom corner of mesh (x, y, z) [cm].
- `size()`: Get the voxel size (dx, dy, dz) [cm].
- `axisAligned()`: Check whether the mesh is axis-aligned or not.
- `index(x, y, z)`: Get the mesh index from point position.
- `where(l, j, k)`: Get the voxel's absolute position (center of voxel) from mesh index [cm].
- `shape()`: Get the mesh shape (tuple with length of 3).
- `extent()`: Get the mesh extent. Mesh must be axis-aligned (tuple with length of 6, { x_{min} , x_{max} , y_{min} , y_{max} , z_{min} , z_{max} }).
- `image(pos, axis)`: Get the 2D image and 2D extent at the slice position. 'pos' is the slice coordinate [cm], and 'axis' is the slicing axis [0=x, 1=y, 2=z]. return {2D image (np.ndarray),

2D uncertainty (np.ndarray), extent (tuple)).

‘image’ method와 matplotlib 라이브러리를 활용하여 MESH_DENSITY tally 데이터의 단면 이미지를 그릴 수 있다. 다음은 관련 Python script의 예시이다.

```
import matplotlib.pyplot as plt
from rt2.scoring import MeshDensity

tally = MeshDensity('foo.mdn')

image, err, extent = tally.image(10, 2) # XY image at the z=10 cm position

plt.figure()
plt.imshow(image, extent=extent, cmap='jet')
plt.show()
```

Figure 27 MESH_DENSITY tally에 대한 단면 이미지의 plot 예시

Mesh 형태의 tally들은 Numpy 스타일의 fancy slicing을 이용한 subset 추출을 지원한다. 다음은 해당 기능을 사용하는 스크립트의 실행 예시이다.

```
import matplotlib.pyplot as plt
from rt2.scoring import MeshDensity

tally = MeshDensity('foo.mdn') # tally has dimension of 200x200x200

subset = tally[100:150] # now subset has dimension of 50x200x200
```

Figure 28 Fancy slicing을 활용한 MESH_DENSITY tally의 subset 추출 예시

10.2.5 class MeshTrack(file_name)

- **Attributes**

- file_name [str]: CROSS type binary tally file.
- part [str]: Unit of scored values.
- data [np.ndarray(float32)]: Scored value, 3D Numpy array.
- unc [np.ndarray(float32)]: Uncertainty of this tally, 3D Numpy array.

- **Methods**

- summary(): Print tally information.
- write(file_name): Save current tally data to binary file.
- operator [+,-,/,*]: Do arithmetic operations for other constants or tallies that have same

'MeshDensity' class type. The opposing operand should be 'MeshDensity' object or numeric value. If opposing operand is 'MeshDensity' object, voxel dimension and affine matrix must be same to apply arithmetic operation. The error-propagation is applied automatically to calculate net uncertainty.

- `etype()`: Get the type of energy binning ['linear' or 'log']
- `ebin()`: Get the Numpy array of energy bin.
- `eindex(energy)`: Get the energy bin index of specific energy [MeV].
- `affine()`: Get the affine matrix, 2D Numpy array [3x4].
- `inverse()`: Get the affine matrix for the inverse transformation [3x4].
- `transform(other)`: Transform tally mesh coordinate by input affine matrix 'other'.
- `translate(x, y, z)`: Translate tally mesh coordinate by input vector (x, y, z) [cm].
- `scale(x, y, z)`: Scale tally mesh bins by input parameters (x, y, z).
- `rotate(theta, axis)`: Rotate tally mesh with polar angle 'theta' [degree] along the 'axis' [0=x, 1=y, 2=z].
- `origin()`: Get the coordinate of left-bottom corner of mesh (x, y, z) [cm].
- `size()`: Get the voxel size (dx, dy, dz) [cm].
- `axisAligned()`: Check whether the mesh is axis-aligned or not.
- `index(x, y, z)`: Get the mesh index from point position.
- `where(l, j, k)`: Get the voxel's absolute position (center of voxel) from mesh index [cm].
- `shape()`: Get the mesh shape (tuple with length of 3).
- `extent()`: Get the mesh extent. Mesh must be axis-aligned (tuple with length of 6, { x_{min} , x_{max} , y_{min} , y_{max} , z_{min} , z_{max} }).
- `extract()`: Merge all energy channels in a single bin. In a result, MeshTrack tally is collapsed to MeshDensity tally.

MESH_TRACK tally의 데이터는 4차원 형태로 단면 2차원 이미지를 얻을 수 없다. 이미지를 얻기 위해서는 'extract' method를 이용하여 MeshDensity tally 데이터로 변환해야 한다. 이 때 전체 에너지 채널이 하나로 합쳐지기 때문에, fancy slicing을 이용하여 원하는 채널을 선택한 subset을 먼저 생성해야 한다. 다음은 해당 기능을 이용하는 Python script의 예시이다.

```

import matplotlib.pyplot as plt
from rt2.scoring import MeshTrack

tally = MeshTrack('foo.mdn') # tally has dimension of 200x200x200x5 (5 ch)

subset = tally[:, :, :, 0] # we want only the first channel.
mden = subset.extract() # now mden has dimension of 200x200x200 (MeshDensity)

image, err, extent = mden.image(0, 2) # XY image at the z=0 cm position

```

Figure 29 Fancy slicing 및 extract method를 활용한 이미지 추출 예시

10.2.6 class Detector(file_name)

● Attributes

- file_name [str]: DETECTOR type binary tally file.
- part [str]: Unit of scored values.
- data [np.ndarray(float32)]: Scored value, 1D Numpy array.
- unc [np.ndarray(float32)]: Uncertainty of this tally, 1D Numpy array.

● Methods

- summary(): Print tally information.
- write(file_name): Save current tally data to binary file.
- operator [+,-,/,*]: Do arithmetic operations for other constants or tallies that have same 'Detector' class type. The opposing operand should be 'Detector' object or numeric value. If opposing operand is 'Detector' object, energy bin structure must be same to apply arithmetic operation. The error-propagation is applied automatically to calculate net uncertainty.
- etype(): Get the type of energy binning ['linear' or 'log']
- ebin(): Get the Numpy array of energy bin.
- eindex(energy): Get the energy bin index of specific energy [MeV].

Detector 클래스의 구조는 Track 클래스와 동일하다.

10.2.7 class PhaseSpace(file_name)

● Attributes

- file_name [str]: Load phase-space binary file if entered. Generate empty phase-space list elsewhere.

● Methods

- summary(): Print tally information.
- reserve(size): Reserve memory capacity by 'size'.

- `resize(size)`: Resize phase-space array. If 'size' is smaller than current data length, data is trimmed.
- `data()`: Get the phase-space with the Numpy structured array format. The data format is `rt2.particle.DTYPE_PHASE_SPACE`.
- `append(arr)`: Append phase-space list. Require Numpy ndarray, 'arr', with `rt2.particle.DTYPE_PHASE_SPACE` data format.
- `write(file_name)`: Save current phase-space data to binary file. This file can be utilized in BEAM_PS card.

이 클래스는 **BEAM_PS** 카드에서 요구하거나 **PHASE_SPACE** tally로 얻을 수 있는 데이터를 생성하거나 해석하는 데 사용된다.

11 Auxiliary Scripts

외부 데이터를 RT²에서 사용하는 데이터 포맷으로 재구성하는 작업을 간편하게 수행할 수 있도록 작성된 스크립트들을 제공한다. DICOM 혹은 NIFTI 데이터를 RT²에서 사용하는 voxel 데이터로 변환할 수 있는 **RT2dicom** 및 **RT2nifti**, FLUKA와 Geant4의 tally 데이터를 RT²에 대응되는 tally 데이터로 변환하는 **RT2fluka** 및 **RT2geant4**가 있으며 해당 스크립트들은 환경변수가 설정되어 있을 때 터미널에서 직접 실행할 수 있다. 이하 내용에서는 해당 스크립트들의 사용법에 대해 설명한다.

11.1 RT2dicom

RT2dicom 스크립트는 특정 디렉토리 내의 DICOM 파일 세트를 읽고 순서에 맞게 배치하여 RT²에서 사용하는 voxel 데이터로 변환한다. 입력 가능한 인수는 다음과 같다.

```

> RT2dicom -h
Parameters: --input    | -i <path>      Path of DICOM CT set
              --houns   | -ho <filename> File for hounsfield table
              --output   | -o <filename> File for voxel output
              --dicom    | -d             Write dicom binary for visualization
              --help     | -h             Print this message

```

Figure 30 RT2dicom의 입력 인수 목록

-i 인수에는 DICOM CT 데이터셋이 저장되어 있는 디렉토리 위치를 설정해야 한다. 해당 위치에는 연속적인 DICOM 이미지가 있어야 하고, 빠진 슬라이스가 있으면 오류가 발생한다. -ho 인수에는 Hounsfield boundary가 정의된 파일이 입력되어야 한다. 해당 파일의 구조는 다음과 같아야 한다.

Region ₁	Ceil ₁
Region ₂	Ceil ₂
...	
Region _n	Ceil _n

여기서 Region_n은 정의할 n번째 region의 이름을 의미하고, Ceil_n은 n번째 region이 할당될 Hounsfield의 상한값을 의미한다. 이 값은 파일 내에서 오름차순으로 정렬되어 있어야 한다. 해당 파일이 입력되었을 때 입력된 DICOM 데이터셋에 존재하는 모든 pixel에 대해 다음 범위 내에 Hounsfield 값이 존재하면 Region_n으로 정의된다.

$$Ceil_{n-1} \leq HU < Ceil_n$$

-ho에 입력하는 파일의 예시는 다음과 같다.

air	-300
tissue	+300
bone	+9999

이 입력문에서 -300 미만의 HU 값을 가지는 모든 pixel은 region air로 설정되고, -300 이상 300 미만의 HU 값을 가지는 pixel은 tissue, 300 이상 9999 미만은 bone으로 설정된다. 모든 입력 값에 문제가 없으면, -o 인수에 설정된 output 이름을 가지는 vxl 확장자를 가지는 파일이 생성된다. 해당 데이터는 RT2builder의 **VOXEL** 카드에서 사용할 수 있다.

-d 인수는 tally data 형태로 데이터를 추출하기 위해 사용된다. DICOM 파일은 2차원 pixel 구조를 가지고 있어 3차원 데이터로의 재구성이 어렵기 때문에, 해당 기능이 제공된다.

11.2 RT2nifti

RT2nifti 스크립트는 nifti 형식의 파일을 RT²에서 사용하는 voxel 데이터로 변환한다. 입력 가능한 인수는 다음과 같다.

```

>RT2nifti -h
Parameters: --input    | -i <path>      Path of DICOM CT set
              --houns   | -ho <filename> File for hounsfield table
              --output   | -o <filename> File for voxel output
              --help     | -h            Print this message

```

Figure 31 RT2nifti의 입력 인수 목록

실행 논리 및 출력 결과물은 RT2dicom 프로그램과 동일하다.

11.3 RT2tetra

추가예정

11.4 RT2fluka

추가예정

11.5 RT2geant4

추가예정