Introduction to PHITS-UDM (User Defined Model)

Yasuhito Sakaki

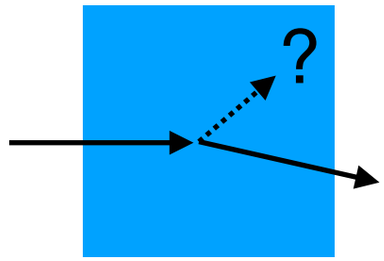
KEK (Radiation Science Center)

Joint Symposium on Nuclear Data and PHITS in 2023, iVil, 15-17 Nov. 2023

Introduction

- The **major particles and interactions are already implemented in PHITS**.

- **However**, in some applications, **we want to implement new particles and interactions in PHITS.**



- In Ver 3.33, a function "**User Defined Model (UDM)**" was implemented to realize them.

- We will add the detail in the manual of Ver 3.34.

  (Or, see the manual on github of PHITS-UDM.)

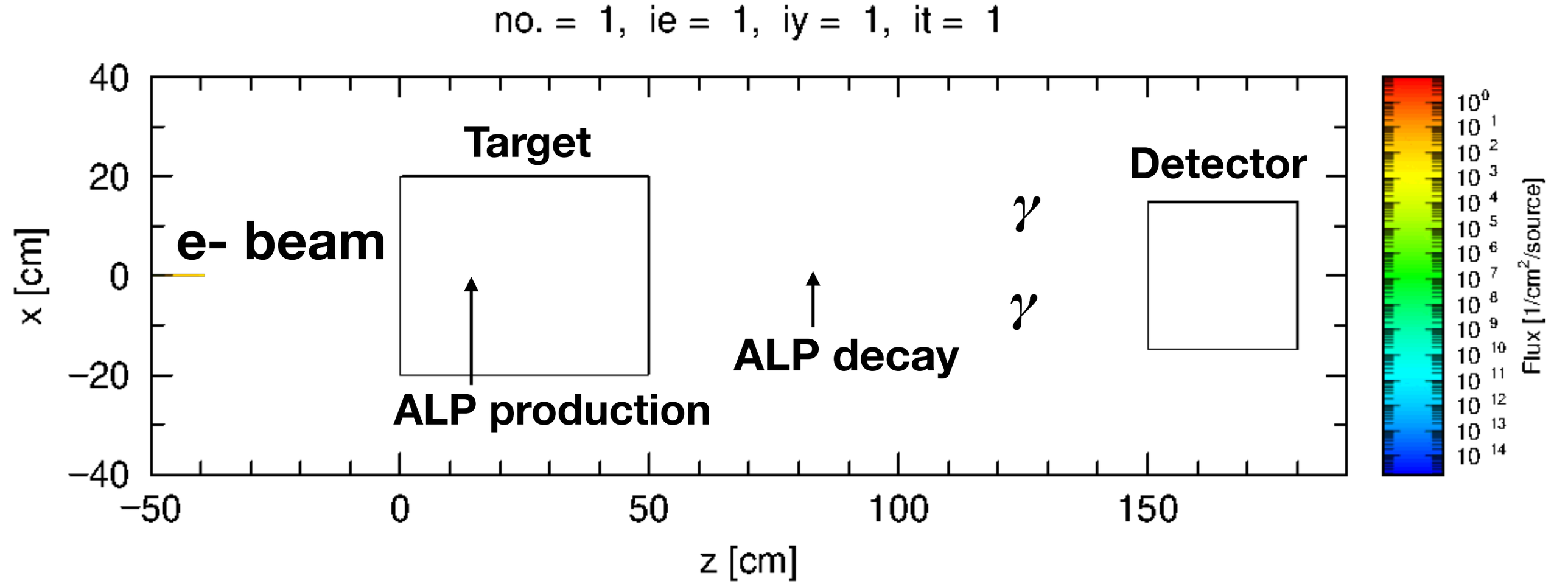
User Defined Model

Model = Interaction + Particle

Two new sections:

```
[ User Defined Interaction ]
[ User Defined Particle ]
```

An example ...



https://rcwww.kek.jp/research/shield/sakaki/_/2023/2d_xz.gif

How to use

1. Create or obtain user code containing information on interactions and particles.
2. Place them in the phits/src/ folder.
3. Execute 'make' (to compile).

Sample codes

(interaction) → `(1) udm_int_sample.f90`

(particle) → `(2) udm_part_sample.f90`

`(3) udm_manager.f90`

2 functions

for user defined interaction

`function Xsec_per_atom(...)`

Define the total cross section.

`subroutine generate_final_state`

Sampling final state energies, momenta, etc.

(1) udm_int_sample.f90

$e^{\pm} + (Z, A) \rightarrow e^{\pm} + X + \ldots$

```
!=============================================================
module udm_int_sample
!=============================================================

use udm_Parameter
use udm_Utility
implicit none
private ! Functions and variables are set to private by de
public :: caller ! The 'caller' subroutine should be publi

!-------------------------------------------------------------
! Default variables
!-------------------------------------------------------------
character(len=99), parameter :: Name = "my_interaction" !
double precision, allocatable, save :: Parameters(:) ! Par
integer, parameter :: num_initial = 2 ! The number of inci
integer, save :: kf_initial(num_initial) = (/ 11, -11 /) !
```

Define a `Name` for this interaction.
This is used in the input file.

The number of incident particles causing this interaction.

`kf-codes` of the incident particles causing this interaction.
In this case, the electron (11) and positron (-11)

(1) udm_int_sample.f90

Define the total cross section per atom

```
48  !=======================================================
49  double precision function Xsec_per_atom(Kin,Z,A)
50  ! Integrated cross section per an atom.
51  ! Unit: barn (10^-24 cm2)
52  implicit none
53  double precision Kin ! Kinetic energy of incident
54                   integer Z ! Atomic number of target at
55                   integer A ! Mass   number of target at
56  !-------------------------------------------------------
57  ! [Variables available in this function]
58  ! udm_kf_incident : The kf-codes (particle IDs) o
59  !=======================================================
60
61  Xsec_per_atom=1.0d-6*Z
62
63  return
64  end
```

10^{-6} * Z barn

(1) udm_int_sample.f90

```
112  !=======================================================================
113  subroutine generate_final_state
114  !=======================================================================
115  ! Subroutine to determine final state information (4-momenta, etc
116  ! In this example, the X and e+- momenta of the final state are d
```

Sampling final states

```
163      ! Set number of final states to be recorded in event history.
164      set_final_state_number = 2
```

The number of final states

```
168      ! Set 4-momentum of X
169      set_kf                          (1) = 900000
170      set_Total_Energy_in_MeV(1) = E_X
171      set_Px_in_MeV                   (1) = p_X*sin(theta_X)
172      set_Py_in_MeV                   (1) = 0.0d0
173      set_Pz_in_MeV                   (1) = p_X*cos(
```

Assign energy, etc to "*set*" arrays

```
181      ! Set 4-momentum of e+-
182      set_kf                          (2) = udm_kf_incident
183      set_Total_Energy_in_MeV(2) = E_e
184      set_Px_in_MeV                   (2) = p_e*sin(theta_e)
185      set_Py_in_MeV                   (2) = 0.0d0
186      set_Pz_in_MeV                   (2) = p_e*cos(theta_e)
```

```
207      call fill_final_state
208      ! ---------------------------------------------------
209      return
```

End

(Example of input file)

```
[ user defined interaction ]
  number = 2

$ Name                    Bias
  my_interaction          1
  my_interaction_2        1
```

4 functions

for user defined particle

`function mass()`

Define the mass of a particle.

`function lifetime()`

Define the lifetime.

`function charge()`

Define the charge.

`subroutine decay`

Define the decay patern.

(2) udm_part_sample.f90

```
3   !=============================================================
4   module udm_part_sample
5   !=============================================================
6
7   use udm_Parameter
8   use udm_Utility
9   implicit none
10  private ! Functions and variables are set to private
11  public :: caller ! The 'caller' subroutine should be
12
13  !-------------------------------------------------------------
14  ! Default variables
15  !-------------------------------------------------------------
16  character(len=99), parameter :: Name = "my_particle"
```

Define a `Name` for this particle.
This is used in the input file.

(2) udm_part_sample.f90

```
46  !=======================================================================
47  double precision function mass() ! Unit: MeV
48  !=======================================================================
49  mass=50.0 ! 50 MeV
50  return
51  end
52
53  !=======================================================================
54  double precision function lifetime() ! Unit: Second
55  ! The value should be greater than 0.
56  !=======================================================================
57  lifetime=0.1e-9 ! 0.1 nano second
58  return
59  end
60
61  !=======================================================================
62  subroutine decay
63  !=======================================================================
64  ! [Variables available in this subroutine]
65  ! udm_kin      : Kinetic Energy of the incident particle [MeV]
66  !-----------------------------------------------------------------------
67  if(0.5 > get_random_0to1()) then
68    call two_body_decay_uniform(12,12)       ! X -> 2 neutrinos (50%)
69  else
70    call three_body_decay_uniform(12,12,12) ! X -> 3 neutrinos (50%)
71  endif
72  end subroutine decay
```

Set mass to 50 MeV

Set lifetime to 0.1 x 10^{-9} second

Set decay patters

Branching Ratio 50%

`two_body_decay_uniform (kf1, kf2)`
Subroutine for 2-body decay, where kf-code of the final state is kf1 and kf2.
`three_body_decay_uniform (kf1, kf2, kf3)`
Subroutine for 3-body decay, where kf-code of the final state is kf1, kf2, and kf3.

(Input file)

```
[ user defined particle ]
  number = 1

$ Name                kfcode    Parameters
  my_particle         900000    100   1.0e-9
```

Parameters(1)　　Parameters(2)

(udm_part_sample.f90)

```
46    !=======================================================
47    double precision function mass() ! Unit: MeV
48    !=======================================================
49    mass=Parameters(1) ! MeV
50    return
51    end
52
53    !=======================================================
54    double precision function lifetime() ! Unit: Second
55    ! The value should be greater than 0.
56    !=======================================================
57    lifetime=Parameters(2) ! second
58    return
59    end
```

One additional note

- UDM can also change the properties (lifetime, decay pattern, distribution, etc.) of particles already implemented in PHITS.

ex) Muon decay distribution

(3) udm_manager.f90

Write the module name of the user code you want to use in udm_Manager.f90.

Line up all modules you want to use.

```
1  module udm_Manager
2  use udm_Parameter
3
4  !=============================================================
5  ! [udm_int]
6  use udm_int_sample_1,   caller_udm_int_sample_1  => caller
7  use udm_int_sample_2,   caller_udm_int_sample_2  => caller
8  ! [udm_part]
9  use udm_part_sample_1,  caller_udm_part_sample_1 => caller
10 use udm_part_sample_2,  caller_udm_part_sample_2 => caller
11 !=============================================================
12
```

use <module name>, caller_<module name> => caller

```
15
16
17
18
19 subroutine user_defined_interaction(action,index)
20 integer action,index
21 !=============================================================
22 call caller_udm_int_sample_1(action,index)
23 call caller_udm_int_sample_2(action,index)
24 !=============================================================
25 end subroutine user_defined_interaction
```

Line up all interactions you want to use.

call caller_<module name>(action,index)

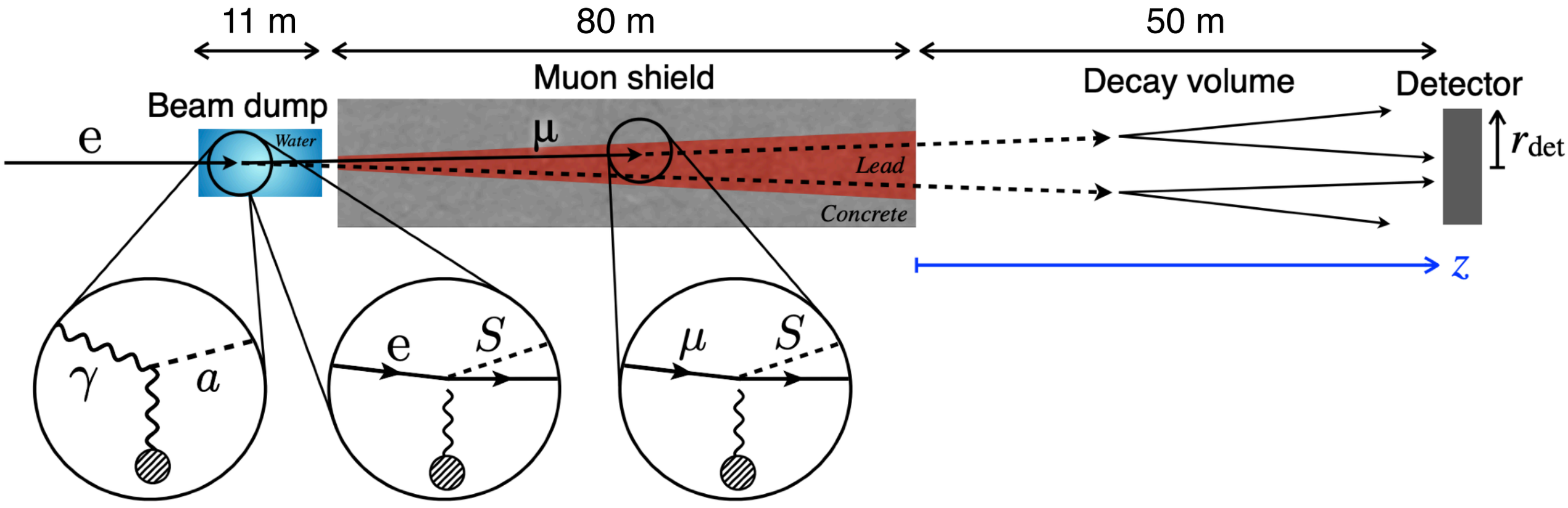
```
28
29
30 subroutine user_defined_particle(action)
31 integer action,index
32 do index=1,udm_part_nMax
33 !=============================================================
34 call caller_udm_part_sample_1(action,index)
35 call caller_udm_part_sample_2(action,index)
36 !=============================================================
37 enddo
```

Line up all particles you want to use.

call caller_<module name>(action,index)

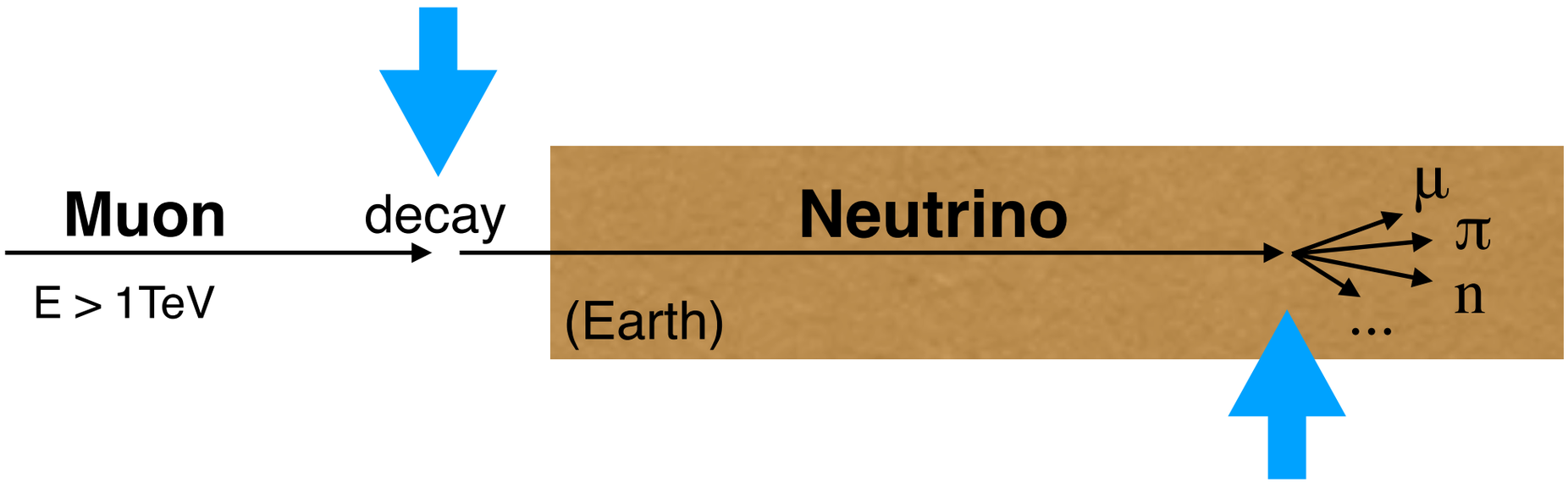
```
40 end module udm_Manager
```

Application example: Search for new particles



Application example: Neutrino-induced effective dose

High precision of decay distribution by UDM

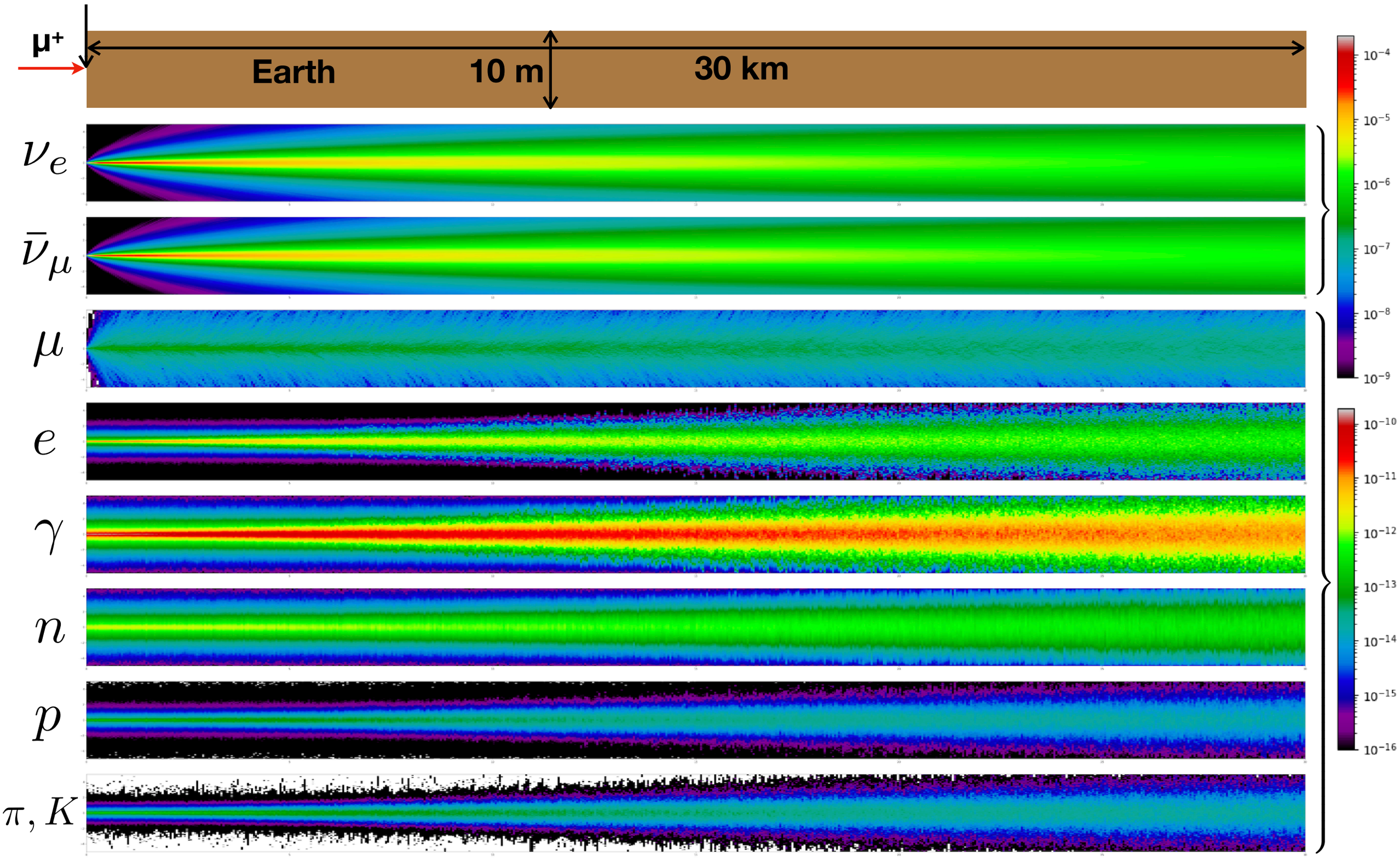


Interaction of high energy neutrinos with matter

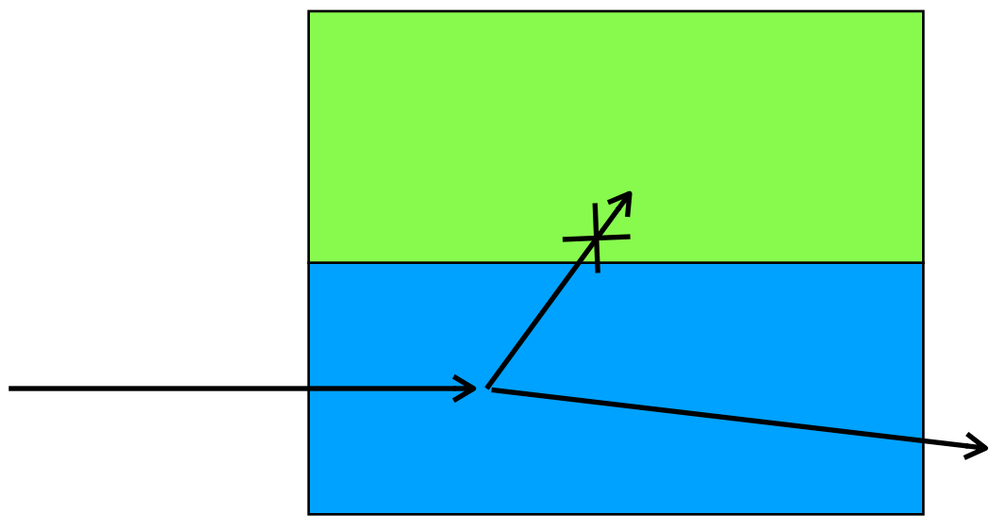
Create a database of the energy and momentum of the end-states with a neutrino-specific generator.

In the *"generate_final_state"* subroutine, read those data and assign them to the "*set*" array.

Neutrino induced fluxes



Application example: Kill particles



For computational speed-up, forcibly decay particles that meet certain conditions (cell, energy, etc).

Summary

- We can easily modify the transport process in PHITS with UDM.
  - ✓ [User Defined Interaction]
  - ✓ [User Defined Particle]

- By sharing user codes, other users can easily use the modification.
  - ➡ It would be interesting if it could be shared on the PHITS forum, etc. (Any good ideas?)