Reference Manual

Generated by Doxygen 1.5.8

Thu Jun 3 13:10:12 2010

Contents

Chapter 1

MC-GPU v1.1

MC-GPU is an x ray transport simulation code that can generate radiographic projection images and computed tomography (CT) scans of voxelized objects, including realistic human anatomy phantoms. The code implements a massively multi-threaded Monte Carlo simulation algorithm for the transport of x rays in a voxelized geometry. The program has been developed using the **CUDA** programming model and the simulation can be executed in parallel in a state-of-the-art GPU from **NVIDIA**, giving an speed up of the order of 15-25 times, compared to a CPU execution. The x ray interaction models and cross sections have been adapted from **PENELOPE 2006**. Currently, the code does not transport secondary electrons and the electrons that would be created in photoelectric and Compton events are assumed to be locally absorbed (dose is not reported).

The MC-GPU code has been described in different scientific publications. A brief description of the code features is given below. This description has been taken from the main paper that can be cited to refer to this code:

```
Andreu Badal and Aldo Badano, "Accelerating Monte Carlo simulations of photon transport in a voxelized
geometry using a massively parallel Graphics Processing Unit", Medical Physics 36, pp. 4878-4880 (2009)
```

This code is still in development, please report to the authors any issue/bug that you may encounter. Feel free to suggest improvements to the code too.

1.1 DISCLAIMER

1.2 Code features

MC-GPU does not currently simulate the transport of electrons. The interactions between the photons and the material objects are simulated using the well-known interaction sampling models from the PENELOPE 2006 subroutine package.

In order to speed up the ray-tracing of the code and minimize the access to the slow GPU main memory, the photon trajectories across the voxels are computed using the Woodcock tracking algorithm. With this technique the photons perceive the geometry as a uniform medium composed of the material of the most attenuating voxel. In this way, the voxel boundaries do not have to be explicitly calculated and multiple voxels can be crossed in a single step. To keep the simulation unbiased, some of the interactions are considered "virtual" (i.e., do not change the photon energy or direction of movement), depending on the actual energy and the material at the interaction site. In a typical simulation, several thousand threads are launched simultaneously in the GPU, each one of them simulating a batch of 10000, or more, photon tracks.

The random number generator used in PENELOPE, ranecu, is also used in the GPU program. To ensure that the simulated tracks are not correlated, each thread initializes the generator to a unique position in the random sequence, far enough from the other threads, using the algorithm implemented in the seedsMLCG code.

The new code is currently used in the study of scatter in x-ray imaging and includes a tally to generate radiographic images. The image is formed by counting the energy that enters a user-defined 2D grid of pixels, which is a simple approximation to a noise-free flat-panel detector with 100% detection efficiency; the pixel values have units of eV/cm^2. Four different images are reported at the end of the simulation, corresponding to the signal produced by non-scattered, single Compton, single Rayleigh, and multi-scattered photons. The radiation source is implemented as a point source emitting monoenergetic photons within a fan beam, producing a rectangular field on the detector equivalent to a collimated cone beam.

1.3 Code compilation and execution

MC-GPU has been tested only in the Linux operating system. A Makefile script is provided to compile the MC-GPU code in Linux. The CUDA libraries and the GNU GCC compiler must be previously installed. The Makefile may have to be edited to modify the library path.

A README text file is provided with the MC-GPU source code. Read this file for more information on the code usage. An example simulation input file is also provided.

MC-GPU uses CUDA to access the GPU but all the actual computations are coded in standard C code. All the CUDA specific commands are enclosed within preprocessor if statements. Defining the pre-processor variable "USING_CUDA" (i.e., compiling with "-DUSING_CUDA") the particle transport is executed in parallel in an NVIDIA GPU using CUDA. Otherwise, the code is sequentially executed in the CPU.

1.4 Parallel simulation of CT scans:

From version 1.1, MC-GPU allows the simulation of a CT scan. The CT is simulated generating multiple projection images around the static voxelized geometry. To speed up the CT simulation, the MPI library is used to address multiple GPUs and obtain multiple projections in parallel. In order to activate the MPI code, the pre-processor variable "USING_MPI" has to be defined (ie, compiling with "-DUSING_MPI"). To use the code in parallel in N GPUs (in a single computer), the user has to run the program with N MPI threads in the CPU (eg, "mpirun -np 4 ./MC-GPU.x

MC-GPU.in"). Each thread will get a unique id in the CPU (myID=0->N) and will address a unique GPU. The CT simulation will then be split so that the threads simulate consecutive projections independently, avoiding any intercommunication between threads.

Chapter 2

Class Index

2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 3

File Index

3.1 File List

Here is a list of all documented files with brief descriptions:

Chapter 4

Class Documentation

4.1 compton_struct Struct Reference

Structure storing the data of the Compton interaction sampling model (equivalent to PENELOPE's common block /CGCO/).

`#include <MC-GPU_v1.1.h>`

Public Attributes

- float **fco** [MAX_MATERIALS *MAX_SHELLS]
- float **uico** [MAX_MATERIALS *MAX_SHELLS]
- float **fj0** [MAX_MATERIALS *MAX_SHELLS]
- int **noscco** [MAX_MATERIALS]

4.1.1 Detailed Description

Structure storing the data of the Compton interaction sampling model (equivalent to PENELOPE's common block /CGCO/).

Definition at line 195 of file MC-GPU_v1.1.h.

The documentation for this struct was generated from the following file:

- **MC-GPU_v1.1.h**

4.2 detector_struct Struct Reference

Structure storing the data defining the x-ray detector.

`#include <MC-GPU_v1.1.h>`

Public Attributes

- float **sdd**
- float3 **corner_min_rotated_to_Y** [MAX_NUM_PROJECTIONS]
- float **rot_inv** [MAX_NUM_PROJECTIONS][9]
- float **width_X**
- float **height_Z**
- float **inv_pixel_size_X**
- float **inv_pixel_size_Z**
- int2 **num_pixels**
- int **total_num_pixels**
- int **rotation_flag**

4.2.1 Detailed Description

Structure storing the data defining the x-ray detector.

For a CT, the struct stores for each angle the detector location and the rotations to transport the detector to a plane perpendicular to +Y.

Definition at line 148 of file MC-GPU_v1.1.h.

The documentation for this struct was generated from the following file:

- **MC-GPU_v1.1.h**

4.3 linear_interp Struct Reference

Structure with the basic data required by the linear interpolation of the mean free paths: number of values and energy grid.

`#include <MC-GPU_v1.1.h>`

Public Attributes

- int **num_values**
- float **e0**
- float **ide**

4.3.1 Detailed Description

Structure with the basic data required by the linear interpolation of the mean free paths: number of values and energy grid.

Definition at line 182 of file MC-GPU_v1.1.h.

The documentation for this struct was generated from the following file:

- **MC-GPU_v1.1.h**

4.4 rayleigh_struct Struct Reference

Structure storing the data of the Rayleigh interaction sampling model (equivalent to PENELOPE's common block /CGRA/).

`#include <MC-GPU_v1.1.h>`

Public Attributes

- float **xco** [NP_RAYLEIGH *MAX_MATERIALS]
- float **pco** [NP_RAYLEIGH *MAX_MATERIALS]
- float **aco** [NP_RAYLEIGH *MAX_MATERIALS]
- float **bco** [NP_RAYLEIGH *MAX_MATERIALS]
- float **pmax** [MAX_ENERGYBINS *MAX_MATERIALS]
- unsigned char **itlco** [NP_RAYLEIGH *MAX_MATERIALS]
- unsigned char **ituco** [NP_RAYLEIGH *MAX_MATERIALS]

4.4.1 Detailed Description

Structure storing the data of the Rayleigh interaction sampling model (equivalent to PENELOPE's common block /CGRA/).

Definition at line 208 of file MC-GPU_v1.1.h.

The documentation for this struct was generated from the following file:

- **MC-GPU_v1.1.h**

4.5 source_struct Struct Reference

Structure storing the data defining the source model.

`#include <MC-GPU_v1.1.h>`

Public Attributes

- float3 **position** [MAX_NUM_PROJECTIONS]
- float3 **direction** [MAX_NUM_PROJECTIONS]
- float **rot_fan** [MAX_NUM_PROJECTIONS][9]
- float **energy**
- float **cos_theta_low**
- float **phi_low**
- float **D_cos_theta**
- float **D_phi**
- float **max_height_at_y1cm**

4.5.1 Detailed Description

Structure storing the data defining the source model.

When a CT is simulated, multiple sources will be stored, one for each projection angle.

Definition at line 127 of file MC-GPU_v1.1.h.

The documentation for this struct was generated from the following file:

- **MC-GPU_v1.1.h**

4.6 voxel_struct Struct Reference

Structure defining a voxelized box with the back-lower corner at the coordinate origin.

`#include <MC-GPU_v1.1.h>`

Public Attributes

- int3 **num_voxels**
- float3 **inv_voxel_size**
- float3 **size_bbox**

4.6.1 Detailed Description

Structure defining a voxelized box with the back-lower corner at the coordinate origin.

Definition at line 169 of file MC-GPU_v1.1.h.

The documentation for this struct was generated from the following file:

- **MC-GPU_v1.1.h**

Chapter 5

File Documentation

5.1 MC-GPU_kernel_v1.1.cu File Reference

Definition of the CUDA GPU kernel for the simulation of x ray tracks in a voxelized geometry.

Defines

- #define **LEAP_DISTANCE** 256

  Upper limit of the number of random values sampled in a single track.

- #define **a1_RANECU** 40014

  Multipliers and moduli for the two MLCG in RANECU.

- #define **m1_RANECU** 2147483563
- #define **a2_RANECU** 40692
- #define **m2_RANECU** 2147483399

Functions

- void **init_image_array_GPU** (unsigned long long int *image, int pixels_per_image)

  Initialize the image array, ie, set all pixels to zero Essentially, this function has the same effect as the command: "cutilSafeCall(cudaMemcpy(image_device, image, image_bytes, cudaMemcpyHostToDevice))";.

- void **track_particles** (int history_batch, int histories_per_thread, int num_p, int seed_input, unsigned long long int *image, float2 *voxel_mat_dens, float2 *mfp_Woodcock_table, float3 *mfp_table_a, float3 *mfp_table_b, struct **rayleigh_struct** *rayleigh_table, struct **compton_struct** *compton_table)

  Main function to simulate x-ray tracks inside a voxelized geometry.

- void **tally_image** (int *num_p, float *energy, float3 *position, float3 *direction, signed char *scatter_state, unsigned long long int *image, float3 *detector_center_SHARED)

  Tally a radiographic projection image.

- void **source** (int *num_p, float3 *position, float3 *direction, float *energy, int2 *seed, int *absvox)

Source that creates primary x rays, according to the defined source model.

- int **set_position** (int *num_p, float *dist, float3 *position, float3 *direction)

  Evaluate if the input distance will move the particle inside the voxels or if another distance has to be used.

- void **init_PRNG** (int history_batch, int histories_per_thread, int seed_input, int2 *seed)

  Initialize the pseudo-random number generator (PRNG) RANECU to a position far away from the previous history (leap frog technique).

- int **abMODm** (int m, int a, int s)

  *Calculate "(a1*a2) MOD m" with 32-bit integers and avoiding the possible overflow, using the Russian Peasant approach modulo m and the approximate factoring method, as described in: L'Ecuyer and Cote, ACM Trans.*

- float **ranecu** (int2 *seed)

  Pseudo-random number generator (PRNG) RANECU returning a float value (single precision version).

- double **ranecu_double** (int2 *seed)

  Pseudo-random number generator (PRNG) RANECU returning a double value.

- int **locate_voxel** (float3 *position)

  Find the voxel that contains the current position.

- void **rotate_double** (float3 *direction, double costh, double phi)

  Rotates a vector; the rotation is specified by giving the polar and azimuthal angles in the "self-frame", as determined by the vector to be rotated.

- void **GRAa** (float *energy, double *costh_Rayleigh, int *mat, float *pmax_current, int2 *seed, struct **rayleigh_struct** *cgra)

  Sample a Rayleigh interaction using the sampling algorithm used in PENELOPE 2006.

- void **GCOa** (float *energy, double *costh_Compton, int *mat, int2 *seed, struct **compton_struct** *cgco_SHARED)

  Random sampling of incoherent (Compton) scattering of photons, using the sampling algorithm from PENELOPE 2006: Relativistic impulse approximation with analytical one-electron Compton profiles.

5.1.1 Detailed Description

Definition of the CUDA GPU kernel for the simulation of x ray tracks in a voxelized geometry.

This kernel has been optimized to yield a good performance in the GPU but can still be compiled in the CPU without problems. All the CUDA especific commands are enclosed in pre-processor directives that are skipped if the parameter "USING_CUDA" is not defined at compilation time.

Author:

Andreu Badal (Andreu.Badal-Soler@fda.hhs.gov)

Date:

2010/05/14

Definition in file **MC-GPU_kernel_v1.1.cu**.

5.1.2 Function Documentation

5.1.2.1 int abMODm (int *m*, int *a*, int *s*) [inline]

Calculate "(a1∗a2) MOD m" with 32-bit integers and avoiding the possible overflow, using the Russian Peasant approach modulo m and the approximate factoring method, as described in: L'Ecuyer and Cote, ACM Trans.

Math. Soft. 17 (1991).

This function has been adapted from "seedsMLCG.f", see: Badal and Sempau, Computer Physics Communications 175 (2006)

Parameters:

← ***m,a,s*** MLCG parameters

Returns:

(a1∗a2) MOD m

Definition at line 882 of file MC-GPU_kernel_v1.1.cu.

Referenced by init_PRNG().

5.1.2.2 void GCOa (float ∗ *energy*, double ∗ *costh_Compton*, int ∗ *mat*, int2 ∗ *seed*, struct compton_struct ∗ *cgco_SHARED*) [inline]

Random sampling of incoherent (Compton) scattering of photons, using the sampling algorithm from PENELOPE 2006: Relativistic impulse approximation with analytical one-electron Compton profiles.

Parameters:

↔ ***energy*** incident and final photon energy (eV)

→ ***costh_Compton*** cosine of the polar scattering angle

← ***material*** Current voxel material

← ***seed*** RANECU PRNG seed

Definition at line 1256 of file MC-GPU_kernel_v1.1.cu.

References compton_struct::fco, compton_struct::fj0, MAX_MATERIALS, max_value, compton_struct::noscco, ranecu(), and compton_struct::uico.

Referenced by track_particles().

5.1.2.3 void GRAa (float ∗ *energy*, double ∗ *costh_Rayleigh*, int ∗ *mat*, float ∗ *pmax_current*, int2 ∗ *seed*, struct rayleigh_struct ∗ *cgra*) [inline]

Sample a Rayleigh interaction using the sampling algorithm used in PENELOPE 2006.

Parameters:

- ← ***energy*** Particle energy (not modified with Rayleigh)
- → ***costh_Rayleigh*** Cosine of the angular deflection
- ← ***material*** Current voxel material

Definition at line 1150 of file MC-GPU_kernel_v1.1.cu.

References rayleigh_struct::aco, rayleigh_struct::bco, rayleigh_struct::itlco, rayleigh_struct::ituco, rayleigh_struct::pco, ranecu_double(), and rayleigh_struct::xco.

Referenced by track_particles().

5.1.2.4 void init_image_array_GPU (unsigned long long int * *image*, int *pixels_per_image*)

Initialize the image array, ie, set all pixels to zero Essentially, this function has the same effect as the command: "cutilSafeCall(cudaMemcpy(image_device, image, image_bytes, cudaMemcpyHostToDevice))";.

CUDA performs some initialization work the first time a GPU kernel is called. Therefore, calling a short kernel before the real particle tracking is performed may improve the accuracy of the timing measurements in the relevant kernel.

Parameters:

- ↔ ***image*** Pointer to the image array.
- ← ***pixels_per_image*** Number of pixels in the image (ie, elements in the array).

Definition at line 59 of file MC-GPU_kernel_v1.1.cu.

Referenced by main().

5.1.2.5 void init_PRNG (int *history_batch*, int *histories_per_thread*, int *seed_input*, int2 * *seed*) [inline]

Initialize the pseudo-random number generator (PRNG) RANECU to a position far away from the previous history (leap frog technique).

Each calculated seed initiates a consecutive and disjoint sequence of pseudo-random numbers with length LEAP_DISTANCE, that can be used to in a parallel simulation (Sequence Splitting parallelization method). The basic equation behind the algorithm is: S(i+j) = (a**j * S(i)) MOD m = [(a**j MOD m)*S(i)] MOD m , which is described in: P L'Ecuyer, Commun. ACM 31 (1988) p.742

This function has been adapted from "seedsMLCG.f", see: A Badal and J Sempau, Computer Physics Communications 175 (2006) p. 440-450

Parameters:

- ← ***history*** Particle bach number.
- ← ***seed_input*** Initial PRNG seed input (used to initiate both MLCGs in RANECU).
- → ***seed*** Initial PRNG seeds for the present history.

Definition at line 804 of file MC-GPU_kernel_v1.1.cu.

References a1_RANECU, abMODm(), and LEAP_DISTANCE.

Referenced by track_particles().

5.1.2.6 int locate_voxel (float3 * *position*) [inline]

Find the voxel that contains the current position.

Parameters:

← ***position*** Particle position

← ***voxel_data*** Pointer to a structure containing the voxel number and size.

Returns:

Returns "absvox", the voxel number where the particle is located (negative if position outside the voxel bbox).

Definition at line 994 of file MC-GPU_kernel_v1.1.cu.

References voxel_struct::inv_voxel_size, voxel_struct::num_voxels, voxel_struct::size_bbox, and voxel_data_CONST.

Referenced by track_particles().

5.1.2.7 float ranecu (int2 * *seed*) [inline]

Pseudo-random number generator (PRNG) RANECU returning a float value (single precision version).

Parameters:

↔ ***seed*** PRNG seed (seed kept in the calling function and updated here).

Returns:

PRN double value in the open interval (0,1)

Definition at line 928 of file MC-GPU_kernel_v1.1.cu.

Referenced by GCOa(), source(), and track_particles().

5.1.2.8 void rotate_double (float3 * *direction*, double *costh*, double *phi*) [inline]

Rotates a vector; the rotation is specified by giving the polar and azimuthal angles in the "self-frame", as determined by the vector to be rotated.

This function is a literal translation from Fortran to C of PENELOPE (v. 2006) subroutine "DIRECT".

Parameters:

↔ ***(u,v,w)*** input vector (=d) in the lab. frame; returns the rotated vector components in the lab. frame

← ***costh*** cos(theta), angle between d before and after turn

← ***phi*** azimuthal angle (rad) turned by d in its self-frame

Definition at line 1072 of file MC-GPU_kernel_v1.1.cu.

Referenced by track_particles().

5.1.2.9 int set_position (int * *num_p*, float * *dist*, float3 * *position*, float3 * *direction*) [inline]

Evaluate if the input distance will move the particle inside the voxels or if another distance has to be used.

Parameters:

← ***dist***

↔ ***position***

Returns:

1 (true) or 0 (false) integer value telling if the distance is acceptable or not.

Definition at line 751 of file MC-GPU_kernel_v1.1.cu.

References source_struct::position, voxel_struct::size_bbox, source_data_CONST, and voxel_data_CONST.

Referenced by source().

5.1.2.10 void source (int * *num_p*, float3 * *position*, float3 * *direction*, float * *energy*, int2 * *seed*, int * *absvox*) [inline]

Source that creates primary x rays, according to the defined source model.

The particles are automatically moved to the surface of the voxel bounding box, to start the tracking inside a real material. If the sampled particle do not enter the voxels, it is init in the focal spot and the main program will check if it arrives at the detector or not.

Parameters:

← ***source_data*** Structure describing the source.

→ ***position*** Initial particle position (particle transported inside the voxel bbox).

→ ***direction*** Sampled particle direction (cosine vectors).

→ ***energy*** Sampled energy of the new x ray.

← ***seed*** Current seed of the random number generator, requiered to sample the movement direction.

→ ***absvox*** Set to <0 if primary particle will not cross the voxels, not changed otherwise (>0).

Definition at line 485 of file MC-GPU_kernel_v1.1.cu.

References source_struct::cos_theta_low, source_struct::D_cos_theta, source_struct::D_phi, detector_data_CONST, source_struct::energy, source_struct::max_height_at_y1cm, source_struct::phi_low, source_struct::position, ranecu(), source_struct::rot_fan, detector_struct::rotation_flag, set_position(), voxel_struct::size_bbox, source_data_CONST, and voxel_data_CONST.

Referenced by track_particles().

5.1.2.11 void tally_image (int * *num_p*, float * *energy*, float3 * *position*, float3 * *direction*, signed char * *scatter_state*, unsigned long long int * *image*, float3 * *detector_center_SHARED*) [inline]

Tally a radiographic projection image.

This function is called whenever a particle escapes the voxelized volume. The code checks if the particle would arrive at the detector if it kept moving in a straight line after exiting the voxels (assuming vacuum enclosure). An ideal image formation model is implemented: each pixel counts the total energy of the x rays that enter the pixel (100% detection efficiency for any energy). The image due to primaries and different kinds of scatter is tallied separately.

In the GPU, and atomicAdd() function is used to make sure that multiple threads do not update the same pixel at the same time, which would result in a lose of information. Since the atomicAdd function is only available for 'unsigned long long int' data, the float pixel values are scaled by a factor "SCALE_eV" defined in the header file (eg, define SCALE_eV 10000.0f) and stored as unsigned long long integers in main memory.

WARNING! If the total tallied signal (for all particles) is larger than "1.8e19/SCALE_eV", there will be a bit overflow and the value will be reset to 0 giving bogus results.

Parameters:

- ← ***energy*** X-ray energy
- ← ***position*** Particle position
- ← ***direction*** Particle direction (cosine vectors)
- ← ***scatter_state*** Flag marking primaries, single Compton, single Rayleigh or multiple scattered radiation
- → ***image*** Integer array containing the image, ie, the pixel values (in tenths of meV)

Definition at line 370 of file MC-GPU_kernel_v1.1.cu.

References detector_struct::corner_min_rotated_to_Y, detector_data_CONST, source_struct::direction, detector_struct::inv_pixel_size_X, detector_struct::inv_pixel_size_Z, detector_struct::num_pixels, detector_struct::rot_inv, detector_struct::rotation_flag, SCALE_eV, detector_struct::sdd, source_data_CONST, and detector_struct::total_num_pixels.

Referenced by track_particles().

5.1.2.12 void track_particles (int *history_batch*, int *histories_per_thread*, int *num_p*, int *seed_input*, unsigned long long int * *image*, float2 * *voxel_mat_dens*, float2 * *mfp_Woodcock_table*, float3 * *mfp_table_a*, float3 * *mfp_table_b*, struct rayleigh_struct * *rayleigh_table*, struct compton_struct * *compton_table*)

Main function to simulate x-ray tracks inside a voxelized geometry.

Secondary electrons are not simulated (in photoelectric and Compton events the energy is locally deposited).

The following global variables, in the GPU __constant__ memory are used: voxel_data_CONST, source_data_CONST, detector_data_CONST, mfp_table_data_CONST.

Parameters:

- ← ***history_batch*** Particle batch number (only used in the CPU version when CUDA is disabled!, the GPU uses the built-in variable threadIdx)

← ***num_p*** Projection number in the CT simulation. This variable defines a specific angle and the corresponding source and detector will be used.

← ***histories_per_thread*** Number of histories to simulate for each call to this function (ie, for GPU thread).

← ***seed_input*** Random number generator seed (the same seed is used to initialize the two MLCGs of RANECU).

← ***voxel_mat_dens*** Pointer to the voxel densities and material vector (the voxelized geometry), stored in GPU glbal memory.

← ***mfp_Woodcock_table*** Two parameter table for the linear interpolation of the Woodcock mean free path (MFP) (stored in GPU global memory).

← ***mfp_table_a*** First element for the linear interpolation of the interaction mean free paths (stored in GPU global memory).

← ***mfp_table_b*** Second element for the linear interpolation of the interaction mean free paths (stored in GPU global memory).

← ***rayleigh_table*** Pointer to the table with the data required by the Rayleigh interaction sampling, stored in GPU global memory.

← ***compton_table*** Pointer to the table with the data required by the Compton interaction sampling, stored in GPU global memory.

↔ ***image*** Pointer to the image vector in the GPU glbal memory.

Definition at line 111 of file MC-GPU_kernel_v1.1.cu.

References detector_data_CONST, source_struct::direction, linear_interp::e0, source_struct::energy, GCOa(), GRAa(), linear_interp::ide, init_PRNG(), locate_voxel(), MAX_MATERIALS, mfp_table_data_CONST, rayleigh_struct::pmax, source_struct::position, ranecu(), ranecu_double(), rotate_double(), detector_struct::sdd, source(), source_data_CONST, and tally_image().

Referenced by main().

5.2 MC-GPU_v1.1.cu File Reference

```
#include <MC-GPU_v1.1.h>
#include <MC-GPU_kernel_v1.1.cu>
```

Functions

- int **main** (int argc, char **argv)

  Main program to transport x rays in a 3D voxel geometry using the GPU.

- void **read_input** (int argc, char **argv, int myID, unsigned long long int *total_histories, int *seed_input, int *gpu_id, int *num_threads_per_block, int *histories_per_thread, struct **detector_struct** *detector_data, unsigned long long int **image_ptr, int *image_bytes, struct **source_struct** *source_data, char *file_name_voxels, char file_name_materials[MAX_MATERIALS][250], char *file_name_output, int *num_projections, double *D_angle, double *angularROI_0, double *angularROI_1, double *initial_angle)

  Read the input file given in the command line and return the significant data.

- void **trim_name** (char *input_line, char *file_name)

  Extract a file name from an input text line, trimming the initial blanks, trailing comment (#) and stopping at the first blank (the file name should not contain blanks).

- char * **fgets_trimmed** (char *trimmed_line, int num, FILE *file_ptr)

  Read a line of text and trim initial blancks and trailing comments (#).

- void **load_voxels** (int myID, char *file_name_voxels, float *density_max, struct **voxel_struct** *voxel_data, float2 **voxel_mat_dens_ptr, unsigned int *voxel_mat_dens_bytes)

  Read the voxel data and allocate the material and density matrix.

- void **load_material** (int myID, char file_name_materials[MAX_MATERIALS][250], float *density_max, float *density_nominal, struct **linear_interp** *mfp_table_data, float2 **mfp_Woodcock_table_ptr, int *mfp_Woodcock_table_bytes, float3 **mfp_table_a_ptr, float3 **mfp_table_b_ptr, int *mfp_table_bytes, struct **rayleigh_struct** *rayleigh_table_ptr, struct **compton_struct** *compton_table_ptr)

  Read the material input files and set the mean free paths and the "linear_interp" structures.

- int **report_host** (char *file_name_output, struct **detector_struct** *detector_data, struct **source_struct** *source_data, unsigned long long int *image, double time_elapsed, unsigned long long int total_histories, int current_projection, int num_projections, double D_angle, double initial_angle, int myID, int numprocs)

  Report the final results, from the host CPU.

- void **set_CT_trajectory** (int myID, int num_projections, double D_angle, double angularROI_0, double angularROI_1, struct **source_struct** *source_data, struct **detector_struct** *detector_data)

  Sets the CT trajectory: store in memory the source and detector rotations that are needed to calculate the multiple projections.

5.2.1 Detailed Description

Author:

Andreu Badal (Andreu.Badal-Soler@fda.hhs.gov)

Date:

2010/05/14 – First version: 2009/03/17

Definition in file **MC-GPU_v1.1.cu**.

5.2.2 Function Documentation

5.2.2.1 char* fgets_trimmed (char * *trimmed_line*, int *num*, FILE * *file_ptr*)

Read a line of text and trim initial blancks and trailing comments (#).

Parameters:

← ***num*** Characters to read
← ***file_ptr*** Pointer to the input file stream
→ ***trimmed_line*** Trimmed line from input file, skipping empty lines and comments

Definition at line 969 of file MC-GPU_v1.1.cu.

Referenced by read_input().

5.2.2.2 void load_material (int *myID*, char *file_name_materials*[MAX_MATERIALS][250], float * *density_max*, float * *density_nominal*, struct linear_interp * *mfp_table_data*, float2 ** *mfp_Woodcock_table_ptr*, int * *mfp_Woodcock_table_bytes*, float3 ** *mfp_table_a_ptr*, float3 ** *mfp_table_b_ptr*, int * *mfp_table_bytes*, struct rayleigh_struct * *rayleigh_table_ptr*, struct compton_struct * *compton_table_ptr*)

Read the material input files and set the mean free paths and the "linear_interp" structures.

Find the material nominal density. Set the Woodcock trick data.

Parameters:

← ***file_name_materials*** Array with the names of the material files.
← ***density_max*** maximum density in the geometry (needed to set Woodcock trick)
→ ***density_nominal*** Array with the nominal density of the materials read
→ ***mfp_table_data*** Constant values for the linear interpolation
→ ***mfp_table_a_ptr*** First element for the linear interpolation.
→ ***mfp_table_b_ptr*** Second element for the linear interpolation.

Definition at line 1166 of file MC-GPU_v1.1.cu.

References rayleigh_struct::aco, rayleigh_struct::bco, linear_interp::e0, compton_struct::fco, compton_struct::fj0, linear_interp::ide, rayleigh_struct::itlco, rayleigh_struct::ituco, MASTER_THREAD, compton_struct::noscco, linear_interp::num_values, rayleigh_struct::pco, rayleigh_struct::pmax, compton_struct::uico, and rayleigh_struct::xco.

Referenced by main().

5.2.2.3 void load_voxels (int *myID*, char * *file_name_voxels*, float * *density_max*, struct voxel_struct * *voxel_data*, float2 ** *voxel_mat_dens_ptr*, unsigned int * *voxel_mat_dens_bytes*)

Read the voxel data and allocate the material and density matrix.

Also find and report the maximum density defined in the geometry.

Parameters:

← ***file_name_voxels*** Name of the voxelized geometry file.

→ ***density_max*** Array with the maximum density for each material in the voxels.

→ ***voxel_data*** Pointer to a structure containing the voxel number and size.

→ ***voxel_mat_dens_ptr*** Pointer to the vector with the voxel materials and densities.

Definition at line 1029 of file MC-GPU_v1.1.cu.

References voxel_struct::inv_voxel_size, MASTER_THREAD, MAX_MATERIALS, voxel_struct::num_voxels, and voxel_struct::size_bbox.

Referenced by main().

5.2.2.4 int main (int *argc*, char ** *argv*)

Main program to transport x rays in a 3D voxel geometry using the GPU.

This function reads the description of the simulation from an external file given in the command line. This input file defines the number of particles to simulate, the characteristics of the x-ray source and the detector, the number and spacing of the projections (if simulating a CT), the location of the material files containing the interaction mean free paths, and the location of the voxelized geometry file.

Author:

Andreu Badal

Date:

2010/03/19

Definition at line 170 of file MC-GPU_v1.1.cu.

References detector_struct::corner_min_rotated_to_Y, source_struct::cos_theta_low, source_struct::D_phi, detector_data_CONST, source_struct::direction, linear_interp::e0, source_struct::energy, linear_interp::ide, init_image_array_GPU(), detector_struct::inv_pixel_size_X, detector_struct::inv_pixel_size_Z, load_material(), load_voxels(), MASTER_THREAD, MAX_MATERIALS, mfp_table_data_CONST, detector_struct::num_pixels, linear_interp::num_values, source_struct::position, read_input(), report_host(), detector_struct::sdd, set_CT_trajectory(), source_data_CONST, track_particles(), and voxel_data_CONST.

5.2.2.5 void read_input (int *argc*, char ** *argv*, int *myID*, unsigned long long int * *total_histories*, int * *seed_input*, int * *gpu_id*, int * *num_threads_per_block*, int * *histories_per_thread*, struct detector_struct * *detector_data*, unsigned long long int ** *image_ptr*, int * *image_bytes*, struct source_struct * *source_data*, char * *file_name_voxels*, char *file_name_materials*[MAX_MATERIALS][250], char * *file_name_output*, int * *num_projections*, double * *D_angle*, double * *angularROI_0*, double * *angularROI_1*, double * *initial_angle*)

Read the input file given in the command line and return the significant data.

Example input file:

1000000 [Total number of histories to simulate] geometry.vox [Voxelized geometry file name] material.mat [Material data file name]

Parameters:

- ← ***argc*** Command line parameters
- ← ***argv*** Command line parameters: name opf input file
- → ***total_histories*** Total number of particles to simulate
- → ***seed_input*** Input random number generator seed
- → ***num_threads_per_block*** Number of CUDA threads for each GPU block
- → ***detector_data***
- → ***image***
- → ***source_data***
- → ***file_name_voxels***
- → ***file_name_materials***
- → ***file_name_output***

!DeBuG!! Force square field for any phi!!

Definition at line 542 of file MC-GPU_v1.1.cu.

References detector_struct::corner_min_rotated_to_Y, source_struct::cos_theta_low, source_struct::D_cos_theta, source_struct::D_phi, source_struct::direction, source_struct::energy, fgets_trimmed(), detector_struct::height_Z, detector_struct::inv_pixel_size_X, detector_struct::inv_pixel_size_Z, MASTER_THREAD, source_struct::max_height_at_y1cm, detector_struct::num_pixels, source_struct::phi_low, source_struct::position, source_struct::rot_fan, detector_struct::rot_inv, detector_struct::rotation_flag, detector_struct::sdd, detector_struct::total_num_pixels, trim_name(), and detector_struct::width_X.

Referenced by main().

5.2.2.6 int report_host (char * *file_name_output*, struct detector_struct * *detector_data*, struct source_struct * *source_data*, unsigned long long int * *image*, double *time_elapsed*, unsigned long long int *total_histories*, int *current_projection*, int *num_projections*, double *D_angle*, double *initial_angle*, int *myID*, int *numprocs*)

Report the final results, from the host CPU.

Parameters:

- ← ***file_name_output*** File where tallied image is reported
- ← ***detector_data*** Detector description read from the input file (pointer to **detector_struct** (p. 10))
- ← ***image*** Tallied image (in meV per pixel)
- ← ***time_elapsed*** Time elapsed during the main loop execution (in seconds)
- ← ***total_histories*** Total number of x-rays simulated

Definition at line 1543 of file MC-GPU_v1.1.cu.

References source_struct::energy, detector_struct::inv_pixel_size_X, detector_struct::inv_pixel_size_Z, detector_struct::num_pixels, and SCALE_eV.

Referenced by main().

5.2.2.7 void set_CT_trajectory (int *myID*, int *num_projections*, double *D_angle*, double *angularROI_0*, double *angularROI_1*, struct source_struct * *source_data*, struct detector_struct * *detector_data*)

Sets the CT trajectory: store in memory the source and detector rotations that are needed to calculate the multiple projections.

The first projection (0) was previously initialized in function "read_input".

ASSUMPTIONS: the CT scan plane must be perpendicular to the Z axis, ie, the initial direction of the particles must have w=0!

Definition at line 1650 of file MC-GPU_v1.1.cu.

References detector_struct::corner_min_rotated_to_Y, source_struct::direction, detector_struct::height_Z, MASTER_THREAD, source_struct::position, source_struct::rot_fan, detector_struct::rot_inv, detector_struct::sdd, and detector_struct::width_X.

Referenced by main().

5.2.2.8 void trim_name (char * *input_line*, char * *file_name*)

Extract a file name from an input text line, trimming the initial blanks, trailing comment (#) and stopping at the first blank (the file name should not contain blanks).

Parameters:

- ← ***input_line*** Input sentence with blanks and a trailing comment
- → ***file_name*** Trimmed file name

Definition at line 941 of file MC-GPU_v1.1.cu.

Referenced by read_input().

5.3 MC-GPU_v1.1.h File Reference

Header file containing the declarations for the MC-GPU code.

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <math.h>
#include <time.h>
#include <vector_types.h>
```

Classes

- struct **double3**
- struct **source_struct**

  Structure storing the data defining the source model.

- struct **detector_struct**

  Structure storing the data defining the x-ray detector.

- struct **voxel_struct**

  Structure defining a voxelized box with the back-lower corner at the coordinate origin.

- struct **linear_interp**

  Structure with the basic data required by the linear interpolation of the mean free paths: number of values and energy grid.

- struct **compton_struct**

  Structure storing the data of the Compton interaction sampling model (equivalent to PENELOPE's common block /CGCO/).

- struct **rayleigh_struct**

  Structure storing the data of the Rayleigh interaction sampling model (equivalent to PENELOPE's common block /CGRA/).

Defines

- #define **MASTER_THREAD** if(0==myID)

  MPI macro: mark commands to be executed only by the master thread (myID==0).

- #define **MAX_NUM_PROJECTIONS** 540

  Maximum number of projections allowed in the CT simulation (limited by the constant memory size):.

- #define **MAX_MATERIALS** 9

  Constants values for the Compton and Rayleigh models:.

- #define **MAX_SHELLS** 30
- #define **MAX_ENERGYBINS** 45005
- #define **NP_RAYLEIGH** 128
- #define **PI** 3.14159265358979323846
- #define **RAD2DEG** 180.0/PI
- #define **DEG2RAD** PI/180.0
- #define **SCALE_eV** 10000.0f

  Value to scale the deposited energy in the pixels so that it can be stored as a long long integer instead of a double precision float.

- #define **EPS_SOURCE** 0.000020f
- #define **NEG_EPS_SOURCE** -0.000020f
- #define **INF** 500000.0f
- #define **INF_minus1** 499999.0f
- #define **NEG_INF** -500000.0f
- #define **max_value**(a, b) (((a) > (b)) ? (a) : (b))

  Preprocessor macro to calculate maximum and minimum values:.

- #define **min_value**(a, b) (((a) < (b)) ? (a) : (b))

Typedefs

- typedef struct double3 **double3**

Functions

- void **read_input** (int argc, char **argv, int myID, unsigned long long int *total_histories, int *gpu_id, int *seed_input, int *num_threads_per_block, int *histories_per_thread, struct **detector_struct** *detector_data, unsigned long long int **image_ptr, int *image_bytes, struct **source_struct** *source_data, char *file_name_voxels, char file_name_materials[MAX_MATERIALS][250], char *file_name_output, int *num_projections, double *D_angle, double *angularROI_0, double *angularROI_1, double *initial_angle)

  Read the input file given in the command line and return the significant data.

- void **load_voxels** (int myID, char *file_name_voxels, float *density_max, struct **voxel_struct** *voxel_data, float2 **voxel_mat_dens_ptr, unsigned int *voxel_mat_dens_bytes)

  Read the voxel data and allocate the material and density matrix.

- void **load_material** (int myID, char file_name_materials[MAX_MATERIALS][250], float *density_max, float *density_nominal, struct **linear_interp** *mfp_table_data, float2 **mfp_Woodcock_table, int *mfp_Woodcock_table_bytes, float3 **mfp_table_a_ptr, float3 **mfp_table_b_ptr, int *mfp_table_bytes, struct **rayleigh_struct** *rayleigh_table_ptr, struct **compton_struct** *compton_table_ptr)

  Read the material input files and set the mean free paths and the "linear_interp" structures.

- void **trim_name** (char *input_line, char *file_name)

  Extract a file name from an input text line, trimming the initial blanks, trailing comment (#) and stopping at the first blank (the file name should not contain blanks).

- char * **fgets_trimmed** (char *trimmed_line, int num, FILE *file_ptr)

  Read a line of text and trim initial blancks and trailing comments (#).

- int **report_host** (char *file_name_output, struct **detector_struct** *detector_data, struct **source_struct** *source_data, unsigned long long int *image, double time_elapsed, unsigned long long int total_histories, int current_projection, int num_projections, double D_angle, double initial_angle, int myID, int numprocs)

  Report the final results, from the host CPU.

- void **set_CT_trajectory** (int myID, int num_projections, double D_angle, double angularROI_0, double angularROI_1, struct **source_struct** *source_data, struct **detector_struct** *detector_data)

  Sets the CT trajectory: store in memory the source and detector rotations that are needed to calculate the multiple projections.

- void **track_particles** (int history_batch, int histories_per_thread, int num_p, int seed_input, unsigned long long int *image, float2 *voxel_mat_dens, float2 *mfp_Woodcock_table, float3 *mfp_table_a, float3 *mfp_table_b, struct **rayleigh_struct** *rayleigh_table, struct **compton_struct** *compton_table)

  Main function to simulate x-ray tracks inside a voxelized geometry.

- void **source** (int *num_p, float3 *position, float3 *direction, float *energy, int2 *seed, int *absvox)

  Source that creates primary x rays, according to the defined source model.

- int **set_position** (int *num_p, float *dist, float3 *position, float3 *direction)

  Evaluate if the input distance will move the particle inside the voxels or if another distance has to be used.

- void **tally_image** (int *num_p, float *energy, float3 *position, float3 *direction, signed char *scatter_state, unsigned long long int *image, float3 *detector_center_SHARED)

  Tally a radiographic projection image.

- void **init_PRNG** (int history_batch, int histories_per_thread, int seed_input, int2 *seed)

  Initialize the pseudo-random number generator (PRNG) RANECU to a position far away from the previous history (leap frog technique).

- int **abMODm** (int m, int a, int s)

  *Calculate "(a1*a2) MOD m" with 32-bit integers and avoiding the possible overflow, using the Russian Peasant approach modulo m and the approximate factoring method, as described in: L'Ecuyer and Cote, ACM Trans.*

- float **ranecu** (int2 *seed)

  Pseudo-random number generator (PRNG) RANECU returning a float value (single precision version).

- double **ranecu_double** (int2 *seed)

  Pseudo-random number generator (PRNG) RANECU returning a double value.

- int **locate_voxel** (float3 *position)

Find the voxel that contains the current position.

- void **rotate_double** (float3 *direction, double cos_theta, double phi)

  Rotates a vector; the rotation is specified by giving the polar and azimuthal angles in the "self-frame", as determined by the vector to be rotated.

- void **GRAa** (float *energy, double *costh_Rayleigh, int *mat, float *pmax_current, int2 *seed, struct **rayleigh_struct** *cgra)

  Sample a Rayleigh interaction using the sampling algorithm used in PENELOPE 2006.

- void **GCOa** (float *energy, double *costh_Compton, int *mat, int2 *seed, struct **compton_struct** *cgco_SHARED)

  Random sampling of incoherent (Compton) scattering of photons, using the sampling algorithm from PENELOPE 2006: Relativistic impulse approximation with analytical one-electron Compton profiles.

Variables

- struct **voxel_struct voxel_data_CONST**

  Global variable to be stored in the GPU constant memory defining the size of the voxel phantom.

- struct **source_struct source_data_CONST**

  Global variable to be stored in the GPU constant memory defining the x-ray source.

- struct **detector_struct detector_data_CONST**

  Global variable to be stored in the GPU constant memory defining the x-ray detector.

- struct **linear_interp mfp_table_data_CONST**

  Global variable to be stored in the GPU constant memory defining the linear interpolation data.

5.3.1 Detailed Description

Header file containing the declarations for the MC-GPU code.

This file declares all the host and device functions and structures, the library files to include in the compilation, various constants parameters of the simulation, pre-processor macro functions, etc.

Author:

Andreu Badal (`Andreu.Badal-Soler@fda.hhs.gov`)

Date:

2010/05/14

Definition in file **MC-GPU_v1.1.h**.

5.3.2 Define Documentation

5.3.2.1 #define SCALE_eV 10000.0f

Value to scale the deposited energy in the pixels so that it can be stored as a long long integer instead of a double precision float.

The integer values have to be used in order to use the atomicAdd function in CUDA.

Definition at line 73 of file MC-GPU_v1.1.h.

Referenced by report_host(), and tally_image().

5.3.3 Function Documentation

5.3.3.1 int abMODm (int *m*, int *a*, int *s*) [inline]

Calculate "(a1*a2) MOD m" with 32-bit integers and avoiding the possible overflow, using the Russian Peasant approach modulo m and the approximate factoring method, as described in: L'Ecuyer and Cote, ACM Trans.

Math. Soft. 17 (1991).

This function has been adapted from "seedsMLCG.f", see: Badal and Sempau, Computer Physics Communications 175 (2006)

Parameters:

← ***m,a,s*** MLCG parameters

Returns:

(a1*a2) MOD m

Definition at line 882 of file MC-GPU_kernel_v1.1.cu.

Referenced by init_PRNG().

5.3.3.2 char* fgets_trimmed (char * *trimmed_line*, int *num*, FILE * *file_ptr*)

Read a line of text and trim initial blancks and trailing comments (#).

Parameters:

← ***num*** Characters to read
← ***file_ptr*** Pointer to the input file stream
→ ***trimmed_line*** Trimmed line from input file, skipping empty lines and comments

Definition at line 969 of file MC-GPU_v1.1.cu.

Referenced by read_input().

5.3.3.3 void GCOa (float * *energy*, double * *costh_Compton*, int * *mat*, int2 * *seed*, struct compton_struct * *cgco_SHARED*) [inline]

Random sampling of incoherent (Compton) scattering of photons, using the sampling algorithm from PENELOPE 2006: Relativistic impulse approximation with analytical one-electron Compton

profiles.

Parameters:

- ↔ ***energy*** incident and final photon energy (eV)
- → ***costh_Compton*** cosine of the polar scattering angle
- ← ***material*** Current voxel material
- ← ***seed*** RANECU PRNG seed

Definition at line 1256 of file MC-GPU_kernel_v1.1.cu.

References compton_struct::fco, compton_struct::fj0, MAX_MATERIALS, max_value, compton_struct::noscco, ranecu(), and compton_struct::uico.

Referenced by track_particles().

5.3.3.4 void GRAa (float * *energy*, double * *costh_Rayleigh*, int * *mat*, float * *pmax_current*, int2 * *seed*, struct rayleigh_struct * *cgra*) [inline]

Sample a Rayleigh interaction using the sampling algorithm used in PENELOPE 2006.

Parameters:

- ← ***energy*** Particle energy (not modified with Rayleigh)
- → ***costh_Rayleigh*** Cosine of the angular deflection
- ← ***material*** Current voxel material

Definition at line 1150 of file MC-GPU_kernel_v1.1.cu.

References rayleigh_struct::aco, rayleigh_struct::bco, rayleigh_struct::itlco, rayleigh_struct::ituco, rayleigh_struct::pco, ranecu_double(), and rayleigh_struct::xco.

Referenced by track_particles().

5.3.3.5 void init_PRNG (int *history_batch*, int *histories_per_thread*, int *seed_input*, int2 * *seed*) [inline]

Initialize the pseudo-random number generator (PRNG) RANECU to a position far away from the previous history (leap frog technique).

Each calculated seed initiates a consecutive and disjoint sequence of pseudo-random numbers with length LEAP_DISTANCE, that can be used to in a parallel simulation (Sequence Splitting parallelization method). The basic equation behind the algorithm is: S(i+j) = (a**j * S(i)) MOD m = [(a**j MOD m)*S(i)] MOD m , which is described in: P L'Ecuyer, Commun. ACM 31 (1988) p.742

This function has been adapted from "seedsMLCG.f", see: A Badal and J Sempau, Computer Physics Communications 175 (2006) p. 440-450

Parameters:

- ← ***history*** Particle bach number.
- ← ***seed_input*** Initial PRNG seed input (used to initiate both MLCGs in RANECU).
- → ***seed*** Initial PRNG seeds for the present history.

Definition at line 804 of file MC-GPU_kernel_v1.1.cu.

References a1_RANECU, abMODm(), and LEAP_DISTANCE.

Referenced by track_particles().

5.3.3.6 void load_material (int *myID*, char *file_name_materials*[MAX_MATERIALS][250], float * *density_max*, float * *density_nominal*, struct linear_interp * *mfp_table_data*, float2 ** *mfp_Woodcock_table_ptr*, int * *mfp_Woodcock_table_bytes*, float3 ** *mfp_table_a_ptr*, float3 ** *mfp_table_b_ptr*, int * *mfp_table_bytes*, struct rayleigh_struct * *rayleigh_table_ptr*, struct compton_struct * *compton_table_ptr*)

Read the material input files and set the mean free paths and the "linear_interp" structures.

Find the material nominal density. Set the Woodcock trick data.

Parameters:

- ← ***file_name_materials*** Array with the names of the material files.
- ← ***density_max*** maximum density in the geometry (needed to set Woodcock trick)
- → ***density_nominal*** Array with the nominal density of the materials read
- → ***mfp_table_data*** Constant values for the linear interpolation
- → ***mfp_table_a_ptr*** First element for the linear interpolation.
- → ***mfp_table_b_ptr*** Second element for the linear interpolation.

Definition at line 1166 of file MC-GPU_v1.1.cu.

References rayleigh_struct::aco, rayleigh_struct::bco, linear_interp::e0, compton_struct::fco, compton_struct::fj0, linear_interp::ide, rayleigh_struct::itlco, rayleigh_struct::ituco, MASTER_THREAD, compton_struct::noscco, linear_interp::num_values, rayleigh_struct::pco, rayleigh_struct::pmax, compton_struct::uico, and rayleigh_struct::xco.

Referenced by main().

5.3.3.7 void load_voxels (int *myID*, char * *file_name_voxels*, float * *density_max*, struct voxel_struct * *voxel_data*, float2 ** *voxel_mat_dens_ptr*, unsigned int * *voxel_mat_dens_bytes*)

Read the voxel data and allocate the material and density matrix.

Also find and report the maximum density defined in the geometry.

Parameters:

- ← ***file_name_voxels*** Name of the voxelized geometry file.
- → ***density_max*** Array with the maximum density for each material in the voxels.
- → ***voxel_data*** Pointer to a structure containing the voxel number and size.
- → ***voxel_mat_dens_ptr*** Pointer to the vector with the voxel materials and densities.

Definition at line 1029 of file MC-GPU_v1.1.cu.

References voxel_struct::inv_voxel_size, MASTER_THREAD, MAX_MATERIALS, voxel_struct::num_voxels, and voxel_struct::size_bbox.

Referenced by main().

5.3.3.8 int locate_voxel (float3 * *position*) [inline]

Find the voxel that contains the current position.

Parameters:

← ***position*** Particle position

← ***voxel_data*** Pointer to a structure containing the voxel number and size.

Returns:

Returns "absvox", the voxel number where the particle is located (negative if position outside the voxel bbox).

Definition at line 994 of file MC-GPU_kernel_v1.1.cu.

References voxel_struct::inv_voxel_size, voxel_struct::num_voxels, voxel_struct::size_bbox, and voxel_data_CONST.

Referenced by track_particles().

5.3.3.9 float ranecu (int2 * *seed*) [inline]

Pseudo-random number generator (PRNG) RANECU returning a float value (single precision version).

Parameters:

↔ ***seed*** PRNG seed (seed kept in the calling function and updated here).

Returns:

PRN double value in the open interval (0,1)

Definition at line 928 of file MC-GPU_kernel_v1.1.cu.

Referenced by GCOa(), source(), and track_particles().

5.3.3.10 void read_input (int *argc*, char ** *argv*, int *myID*, unsigned long long int * *total_histories*, int * *seed_input*, int * *gpu_id*, int * *num_threads_per_block*, int * *histories_per_thread*, struct detector_struct * *detector_data*, unsigned long long int ** *image_ptr*, int * *image_bytes*, struct source_struct * *source_data*, char * *file_name_voxels*, char *file_name_materials*[MAX_MATERIALS][250], char * *file_name_output*, int * *num_projections*, double * *D_angle*, double * *angularROI_0*, double * *angularROI_1*, double * *initial_angle*)

Read the input file given in the command line and return the significant data.

Example input file:

1000000 [Total number of histories to simulate] geometry.vox [Voxelized geometry file name] material.mat [Material data file name]

Parameters:

← ***argc*** Command line parameters

← ***argv*** Command line parameters: name opf input file

→ ***total_ histories*** Total number of particles to simulate

→ ***seed_ input*** Input random number generator seed

→ ***num_ threads_ per_ block*** Number of CUDA threads for each GPU block

→ ***detector_ data***

→ ***image***

→ ***source_ data***

→ ***file_ name_ voxels***

→ ***file_ name_ materials***

→ ***file_ name_ output***

!DeBuG!! Force square field for any phi!!

Definition at line 542 of file MC-GPU_v1.1.cu.

References detector_struct::corner_min_rotated_to_Y, source_struct::cos_theta_low, source_struct::D_cos_theta, source_struct::D_phi, source_struct::direction, source_struct::energy, fgets_trimmed(), detector_struct::height_Z, detector_struct::inv_pixel_size_X, detector_struct::inv_pixel_size_Z, MASTER_THREAD, source_struct::max_height_at_y1cm, detector_struct::num_pixels, source_struct::phi_low, source_struct::position, source_struct::rot_fan, detector_struct::rot_inv, detector_struct::rotation_flag, detector_struct::sdd, detector_struct::total_num_pixels, trim_name(), and detector_struct::width_X.

Referenced by main().

5.3.3.11 int report_host (char * *file_ name_ output*, struct detector_struct * *detector_ data*, struct source_struct * *source_ data*, unsigned long long int * *image*, double *time_ elapsed*, unsigned long long int *total_ histories*, int *current_ projection*, int *num_ projections*, double *D_ angle*, double *initial_ angle*, int *myID*, int *numprocs*)

Report the final results, from the host CPU.

Parameters:

← ***file_ name_ output*** File where tallied image is reported

← ***detector_ data*** Detector description read from the input file (pointer to **detector_ struct** (p. 10))

← ***image*** Tallied image (in meV per pixel)

← ***time_ elapsed*** Time elapsed during the main loop execution (in seconds)

← ***total_ histories*** Total number of x-rays simulated

Definition at line 1543 of file MC-GPU_v1.1.cu.

References source_struct::energy, detector_struct::inv_pixel_size_X, detector_struct::inv_pixel_size_Z, detector_struct::num_pixels, and SCALE_eV.

Referenced by main().

5.3.3.12 void rotate_double (float3 * *direction*, double *costh*, double *phi*) [inline]

Rotates a vector; the rotation is specified by giving the polar and azimuthal angles in the "self-frame", as determined by the vector to be rotated.

This function is a literal translation from Fortran to C of PENELOPE (v. 2006) subroutine "DIRECT".

Parameters:

- ↔ ***(u,v,w)*** input vector (=d) in the lab. frame; returns the rotated vector components in the lab. frame
- ← ***costh*** cos(theta), angle between d before and after turn
- ← ***phi*** azimuthal angle (rad) turned by d in its self-frame

Definition at line 1072 of file MC-GPU_kernel_v1.1.cu.

Referenced by track_particles().

5.3.3.13 void set_CT_trajectory (int *myID*, int *num_projections*, double *D_angle*, double *angularROI_0*, double *angularROI_1*, struct source_struct * *source_data*, struct detector_struct * *detector_data*)

Sets the CT trajectory: store in memory the source and detector rotations that are needed to calculate the multiple projections.

The first projection (0) was previously initialized in function "read_input".

ASSUMPTIONS: the CT scan plane must be perpendicular to the Z axis, ie, the initial direction of the particles must have w=0!

Definition at line 1650 of file MC-GPU_v1.1.cu.

References detector_struct::corner_min_rotated_to_Y, source_struct::direction, detector_struct::height_Z, MASTER_THREAD, source_struct::position, source_struct::rot_fan, detector_struct::rot_inv, detector_struct::sdd, and detector_struct::width_X.

Referenced by main().

5.3.3.14 int set_position (int * *num_p*, float * *dist*, float3 * *position*, float3 * *direction*) [inline]

Evaluate if the input distance will move the particle inside the voxels or if another distance has to be used.

Parameters:

- ← ***dist***
- ↔ ***position***

Returns:

1 (true) or 0 (false) integer value telling if the distance is acceptable or not.

Definition at line 751 of file MC-GPU_kernel_v1.1.cu.

References source_struct::position, voxel_struct::size_bbox, source_data_CONST, and voxel_-data_CONST.

Referenced by source().

5.3.3.15 void source (int * *num_p*, float3 * *position*, float3 * *direction*, float * *energy*, int2 * *seed*, int * *absvox*) [inline]

Source that creates primary x rays, according to the defined source model.

The particles are automatically moved to the surface of the voxel bounding box, to start the tracking inside a real material. If the sampled particle do not enter the voxels, it is init in the focal spot and the main program will check if it arrives at the detector or not.

Parameters:

- ← ***source_data*** Structure describing the source.
- → ***position*** Initial particle position (particle transported inside the voxel bbox).
- → ***direction*** Sampled particle direction (cosine vectors).
- → ***energy*** Sampled energy of the new x ray.
- ← ***seed*** Current seed of the random number generator, requiered to sample the movement direction.
- → ***absvox*** Set to <0 if primary particle will not cross the voxels, not changed otherwise (>0).

Definition at line 485 of file MC-GPU_kernel_v1.1.cu.

References source_struct::cos_theta_low, source_struct::D_cos_theta, source_struct::D_-phi, detector_data_CONST, source_struct::energy, source_struct::max_height_at_y1cm, source_struct::phi_low, source_struct::position, ranecu(), source_struct::rot_fan, detector_-struct::rotation_flag, set_position(), voxel_struct::size_bbox, source_data_CONST, and voxel_data_CONST.

Referenced by track_particles().

5.3.3.16 void tally_image (int * *num_p*, float * *energy*, float3 * *position*, float3 * *direction*, signed char * *scatter_state*, unsigned long long int * *image*, float3 * *detector_center_SHARED*) [inline]

Tally a radiographic projection image.

This function is called whenever a particle escapes the voxelized volume. The code checks if the particle would arrive at the detector if it kept moving in a straight line after exiting the voxels (assuming vacuum enclosure). An ideal image formation model is implemented: each pixel counts the total energy of the x rays that enter the pixel (100% detection efficiency for any energy). The image due to primaries and different kinds of scatter is tallied separately.

In the GPU, and atomicAdd() function is used to make sure that multiple threads do not update the same pixel at the same time, which would result in a lose of information. Since the atomicAdd function is only available for 'unsigned long long int' data, the float pixel values are scaled by a factor "SCALE_eV" defined in the header file (eg, define SCALE_eV 10000.0f) and stored as unsigned long long integers in main memory.

WARNING! If the total tallied signal (for all particles) is larger than "1.8e19/SCALE_eV", there will be a bit overflow and the value will be reset to 0 giving bogus results.

Parameters:

- ← ***energy*** X-ray energy
- ← ***position*** Particle position
- ← ***direction*** Particle direction (cosine vectors)
- ← ***scatter_state*** Flag marking primaries, single Compton, single Rayleigh or multiple scattered radiation
- → ***image*** Integer array containing the image, ie, the pixel values (in tenths of meV)

Definition at line 370 of file MC-GPU_kernel_v1.1.cu.

References detector_struct::corner_min_rotated_to_Y, detector_data_CONST, source_struct::direction, detector_struct::inv_pixel_size_X, detector_struct::inv_pixel_size_Z, detector_struct::num_pixels, detector_struct::rot_inv, detector_struct::rotation_flag, SCALE_eV, detector_struct::sdd, source_data_CONST, and detector_struct::total_num_pixels.

Referenced by track_particles().

5.3.3.17 void track_particles (int *history_batch*, int *histories_per_thread*, int *num_p*, int *seed_input*, unsigned long long int * *image*, float2 * *voxel_mat_dens*, float2 * *mfp_Woodcock_table*, float3 * *mfp_table_a*, float3 * *mfp_table_b*, struct rayleigh_struct * *rayleigh_table*, struct compton_struct * *compton_table*)

Main function to simulate x-ray tracks inside a voxelized geometry.

Secondary electrons are not simulated (in photoelectric and Compton events the energy is locally deposited).

The following global variables, in the GPU __constant__ memory are used: voxel_data_CONST, source_data_CONST, detector_data_CONST, mfp_table_data_CONST.

Parameters:

- ← ***history_batch*** Particle batch number (only used in the CPU version when CUDA is disabled!, the GPU uses the built-in variable threadIdx)
- ← ***num_p*** Projection number in the CT simulation. This variable defines a specific angle and the corresponding source and detector will be used.
- ← ***histories_per_thread*** Number of histories to simulate for each call to this function (ie, for GPU thread).
- ← ***seed_input*** Random number generator seed (the same seed is used to initialize the two MLCGs of RANECU).
- ← ***voxel_mat_dens*** Pointer to the voxel densities and material vector (the voxelized geometry), stored in GPU glbal memory.
- ← ***mfp_Woodcock_table*** Two parameter table for the linear interpolation of the Woodcock mean free path (MFP) (stored in GPU global memory).
- ← ***mfp_table_a*** First element for the linear interpolation of the interaction mean free paths (stored in GPU global memory).
- ← ***mfp_table_b*** Second element for the linear interpolation of the interaction mean free paths (stored in GPU global memory).
- ← ***rayleigh_table*** Pointer to the table with the data required by the Rayleigh interaction sampling, stored in GPU global memory.

← ***compton_table*** Pointer to the table with the data required by the Compton interaction sampling, stored in GPU global memory.

↔ ***image*** Pointer to the image vector in the GPU glbal memory.

Definition at line 111 of file MC-GPU_kernel_v1.1.cu.

References detector_data_CONST, source_struct::direction, linear_interp::e0, source_struct::energy, GCOa(), GRAa(), linear_interp::ide, init_PRNG(), locate_voxel(), MAX_MATERIALS, mfp_table_data_CONST, rayleigh_struct::pmax, source_struct::position, ranecu(), ranecu_double(), rotate_double(), detector_struct::sdd, source(), source_data_CONST, and tally_image().

Referenced by main().

5.3.3.18 void trim_name (char * *input_line*, char * *file_name*)

Extract a file name from an input text line, trimming the initial blanks, trailing comment (#) and stopping at the first blank (the file name should not contain blanks).

Parameters:

← ***input_line*** Input sentence with blanks and a trailing comment

→ ***file_name*** Trimmed file name

Definition at line 941 of file MC-GPU_v1.1.cu.

Referenced by read_input().

Index