**Particle Simulation**

1. Case where we have three particles (one Photon, one Neutrino and one Electron) which have as directions ‘DDDDDDDDD’, ‘UUUUUUUUU’, ‘RRRRRRRRR’ respectively (given in the Directions.txt file) and initial coordinates (10,15), (10,9) and (7,12) in a chamber of size 40X25 with maximum x-coordinate 39, minimum x-coordinate 0, maximum y-coordinate 24 and minimum y-coordinate 0. We notice that at the third step the three particles meet. Both the velocities and the directions are deterministic. The output of the Simulation is shown in “Case1.txt”. The code snippet in main is this:

#include<iostream>

using namespace std;

#include< vector>

#include "TheClasses.h"

#include "TheChamber.h"

#include "ProcessingInput.h"

#include "FilenamesReader.h"

#include "InitialiserClass.h"

#include "Directions.h"

#include "Velocities.h"

int main()

{

DeterministicVelocities velocity(10); //It is an abstract class so we need to pass something.

//Reading the filenames

FilenamesReader textfilenames("Filenames.txt");

InitialiserClass initialise(textfilenames,velocity);

initialise();//Passing the input text files into the input processing engine.

vector<int> ChamberTerminalCoordinates;

try{

ChamberTerminalCoordinates=initialise.getChamberTerminalCoordinates();//The terminal coordinates.

}catch(char\* msg)

{

cout<<msg<<endl;

cout<<"Program is terminated. Please press a key to continue."<<endl;

getchar();

exit(0);

}

vector<Particle\*> MyParticles=initialise.getMyParticles();// The particles

vector<vector<char>> TheDeterministicDirections=initialise.getDirections();//The directions.

DeterministicDirections deterdir(TheDeterministicDirections);

TheChamber chamber(MyParticles,deterdir,ChamberTerminalCoordinates[0],ChamberTerminalCoordinates[1],ChamberTerminalCoordinates[2],ChamberTerminalCoordinates[3]);

chamber();

return 0;

}

1. We have again the three particles with the same initial coordinates and the deterministic function of velocities. The code in the main function is the same. The only difference is that at this time we are seeing how each one of the particles behave in terminal conditions i.e the Electron getting stack on the Chamber’s wall, the Neutrino escaping and the Photon being reflected. The only thing that changes is the deterministic directions in the Directions.txt file. They are: ‘DDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDD’, ‘UUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUU’ and ‘RRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRR’ respectively. We see that after simulation number 15 P is reflected with velocity 1( the velocity of the wall at that position) and get’s back to its initial position),after simulation number 16 N leaves the chamber and after simulation number 33 the E is trapped at the Chamber’s wall. The output of this
2. Let’s assume now a more general case where we have many deterministic directions in the Directions.txt file (not the same as before) as well as many particles (not three as before). The Directions.txt file contains 30 lines of 162 directions each. The Coordinates.txt file contains 30 lines of coordinates as well as the Types.txt file contains 30 types. (Remember that if the files do not contain the same number of lines the program will fail (will display an error message and will be terminated) as it will if the number of directions in each line are not the same. The output of that simulation is in Case3.txt. The code of the main function is the same.
3. Let’s assume now the same case as above with stochastic velocities (Velocities are a function of space. For each coordinate of the chamber we add to 1 random number from 0 to 3). The output is shown at Case4.txt. The code in main is:

#include<iostream>

using namespace std;

#include< vector>

#include "TheClasses.h"

#include "TheChamber.h"

#include "ProcessingInput.h"

#include "FilenamesReader.h"

#include "InitialiserClass.h"

#include "Directions.h"

#include "Velocities.h"

int main()

{

StochasticVelocities velocity(10); //It is an abstract class so we need to pass something.

//Reading the filenames

FilenamesReader textfilenames("Filenames.txt");

InitialiserClass initialise(textfilenames,velocity);

initialise();//Passing the input text files into the input processing engine.

vector<int> ChamberTerminalCoordinates;

try{

ChamberTerminalCoordinates=initialise.getChamberTerminalCoordinates();//The terminal coordinates.

}catch(char\* msg)

{

cout<<msg<<endl;

cout<<"Program is terminated. Please press a key to continue."<<endl;

getchar();

exit(0);

}

vector<Particle\*> MyParticles=initialise.getMyParticles();// The particles

vector<vector<char>> TheDeterministicDirections=initialise.getDirections();//The directions.

DeterministicDirections deterdir(TheDeterministicDirections);

TheChamber chamber(MyParticles,deterdir,ChamberTerminalCoordinates[0],ChamberTerminalCoordinates[1],ChamberTerminalCoordinates[2],ChamberTerminalCoordinates[3]);

chamber();

return 0;

}

1. The next case, is a case of stochastic directions and deterministic velocities. This time the file of Directions.txt will not serve as giving directions to the particles but to give the number of particles( even though the file does not change). The number of stochastic simulations is determined in the constructor and for this case we set it to 100. The number of coordinates and types is 30. Case5.txt has the simulation output of this case. The code of the main function is:

#include<iostream>

using namespace std;

#include< vector>

#include "TheClasses.h"

#include "TheChamber.h"

#include "ProcessingInput.h"

#include "FilenamesReader.h"

#include "InitialiserClass.h"

#include "Directions.h"

#include "Velocities.h"

int main()

{

DeterministicVelocities velocity(10); //It is an abstract class so we need to pass something.

//Reading the filenames

FilenamesReader textfilenames("Filenames.txt");

InitialiserClass initialise(textfilenames,velocity);

initialise();//Passing the input text files into the input processing engine.

vector<int> ChamberTerminalCoordinates;

try{

ChamberTerminalCoordinates=initialise.getChamberTerminalCoordinates();//The terminal coordinates.

}catch(char\* msg)

{

cout<<msg<<endl;

cout<<"Program is terminated. Please press a key to continue."<<endl;

getchar();

exit(0);

}

vector<Particle\*> MyParticles=initialise.getMyParticles();// The particles

vector<vector<char>> TheDeterministicDirections=initialise.getDirections();//The directions.

StochasticDirections stochdir(100,TheDeterministicDirections);

TheChamber chamber(MyParticles,stochdir,ChamberTerminalCoordinates[0],ChamberTerminalCoordinates[1],ChamberTerminalCoordinates[2],ChamberTerminalCoordinates[3]);

chamber();

return 0;

}

1. The last case that we present here is the most interesting one. In this case we have both stochastic directions and velocities. The output is saved at Case6.txt and the code of the main is:

#include<iostream>

using namespace std;

#include< vector>

#include "TheClasses.h"

#include "TheChamber.h"

#include "ProcessingInput.h"

#include "FilenamesReader.h"

#include "InitialiserClass.h"

#include "Directions.h"

#include "Velocities.h"

int main()

{

StochasticVelocities velocity(10); //It is an abstract class so we need to pass something.

//Reading the filenames

FilenamesReader textfilenames("Filenames.txt");

InitialiserClass initialise(textfilenames,velocity);

initialise();//Passing the input text files into the input processing engine.

vector<int> ChamberTerminalCoordinates;

try{

ChamberTerminalCoordinates=initialise.getChamberTerminalCoordinates();//The terminal coordinates.

}catch(char\* msg)

{

cout<<msg<<endl;

cout<<"Program is terminated. Please press a key to continue."<<endl;

getchar();

exit(0);

}

vector<Particle\*> MyParticles=initialise.getMyParticles();// The particles

vector<vector<char>> TheDeterministicDirections=initialise.getDirections();//The directions.

StochasticDirections stochdir(100,TheDeterministicDirections);

TheChamber chamber(MyParticles,stochdir,ChamberTerminalCoordinates[0],ChamberTerminalCoordinates[1],ChamberTerminalCoordinates[2],ChamberTerminalCoordinates[3]);

chamber();

return 0;

}