kernelvoidPointWaves( // ОбъявлениеядраPointWaves

constintNp, // объявление и инициализация константы типа int с именемNp – число частиц.

сonstintNm, // объявление и инициализация константы типа int с именем Nm – число материалов.

constintNd, // объявлениеиинициализацияконстантытипаintсименемNd – расстояние

constintNx, // Количество ячеек по оси х

constintNy, // Количество ячеек по оси y

constintNz,// Количество ячеек по оси z

\_\_globalfloat\* CRD, //Глобальная переменная доступная всем функциям, объявленным в данном файле CRD – массив содержащий координаты и скорости частиц.

\_\_globalchar\* MTR, // объявление массива MTR – массив содержащий тип материала каждой частицы

\_\_globalfloat\* FRC, // объявление массива FRC – массив, описывающий силу взаимодействия между частицами в зависимости от расстояний

\_\_globalunsignedint\* ID3, // объявление массива ID3 – массив, содержащий идентификаторы частиц в каждой ячейке

\_\_globalunsignedint\* ID0, // Массив, содержащий идентификационный номер каждой частицы в трехмерном массиве ячеекNx \* Ny \* Nz в начальный момент времени

constfloatd, // период решетки

constfloatdt) // шаг по времени

{

Constintidr = get\_global\_id(0); // переменнаяidr являющаясяконстантой = глобальному идентификатору процесса

if (idr>=Nz) return; // если (idr>=Nz) выходгдеNz – количество ячеек по оси z

intnx,ny,nz,nnx,nny,nnz,n,nn,size,np,nnp,nr,mtr,mtrr,nF; // объявлениепеременныхтипомint

floatx,y,z,x0,y0,z0,xx,yy,zz,vx,vy,vz,r,F,Fx,Fy,Fz,nr\_,kr,r1,d2,krf,d9; // объявлениепеременныхтипомfloat

nz=idr; // идентификатор частицы

size=Nx\*Ny\*Nz; // размер расчетной сетки

d2=d\*0.5; // расстояние при котором частица не покидает пределы ячейки.

d9=d2\*0.99; // расстояние до границы области моделирования

krf=1.0/(7\*d); //коэффициент пространственного масштаба для пересчёта на дискретную сетку

n=nz\*Nx\*Ny; // номер ячейки

for (ny=0;ny<Ny;ny++) // Цикл по номеру ячейки по оси Y

for (nx=0;nx<Nx;nx++) // Цикл по номеру ячейки по оси X

{

if (ID0[n]!=0) // еслиID0 не равен нулю. Где ID0 - массив, содержащий старый идентификационный номер каждой частицы {\n\

np=ID0[n]; // номер частицы определяется идентификационным номер содержащимся в массиве ID0[n]

x=CRD[np\*10+3]; // координата частицы х

y=CRD[np\*10+4]; // координата частицы у

z=CRD[np\*10+5]; // координата частицы z

vx=CRD[np\*10+6];// скорость частиц вдоль оси х

vy=CRD[np\*10+7]; // скорость частицы вдоль оси у

vz=CRD[np\*10+8]; // скорость частицы вдоль оси z

mtr=MTR[np]; // материал определяется из массива MTR

Fx=0;Fy=0;Fz=0; // обнуляем силовое воздействие

for (nnz=-4;nnz<=4;nnz++) // производим опрос соседних ячеек на наличие частицы в них по оси z

for (nny=-4;nny<=4;nny++) // производим опрос соседних ячеек на наличие частицы в них по оси у

for (nnx=-4;nnx<=4;nnx++) // производим опрос соседних ячеек на наличие частицы в них по оси х

{

nn=n+nnx+nny\*Nx+nnz\*Nx\*Ny; // номер соседней частицы

if((nn>=0)&&(nn<size)&&(nn!=n)) // Если соседняя частица не выходит за пределы массива и не равна по номеру текущей частице.

{

if (ID0[nn]!=0)//Проверка является ли соседняя ячейка пустой по ID

{

nnp=ID0[nn]; // текущий номер соседней частицы в соседней ячейке

mtrr=MTR[nnp]; // материал соседней частицы

x0=CRD[nnp\*10+3]+nnx\*d; // координаты соседней частицы

y0=CRD[nnp\*10+4]+nny\*d; // координаты соседнейчастицы

z0=CRD[nnp\*10+5]+nnz\*d; // координатысоседней частицы

r=sqrt((x-x0)\*(x-x0)+(y-y0)\*(y-y0)+(z-z0)\*(z-z0)); // расстояние между текущей частицей и соседней

if (r>0) {r1=1/r;

nr\_=r\*Nd\*krf;//вычисление численного расстояния между текущей частицей и соседней

if (nr\_>Nd-2)nr\_=Nd-2; // проверка на условие не выхода за пределы массива

nr=nr\_;вычисление целочисленного расстояния между текущей частицей и соседней

kr=nr\_-nr; коэффициент интерполяции

nF=(mtr+mtrr\*Nm)\*Nd+nr+Nm; выбор силового графика для конкретного сочетания материалов

F=FRC[nF]\*(1-kr)+FRC[nF+1]\*kr; // определение силы взаимодействия между текущей частицей и соседней

Fx=Fx-F\*(x-x0)\*r1; // силовое воздействие вдоль оси х

Fy=Fy-F\*(y-y0)\*r1; // силовое воздействие вдоль оси у

Fz=Fz-F\*(z-z0)\*r1;} // силовое воздействие вдоль оси z

}

}

}

Fx=Fx/FRC[mtr];//Force devide by mass

Fy=Fy/FRC[mtr];//Force devide by mass

Fz=Fz/FRC[mtr];//Force devide by mass

ktr=0.05 // коэффициент трения

vx=vx+(Fx-vx\*ktr)\*dt; // скорость частиц через промежуток времени dt с учетом коэффициента трения

vy=vy+(Fy-vy\*ktr)\*dt; // скорость частиц

vz=vz+(Fz-vz\*ktr)\*dt; // скорость частиц

CRD [np\*10+6]=vx; // скорость частиц в массиве CRD

CRD[np\*10+7]=vy; // скорость частиц в массиве CRD

CRD[np\*10+8]=vz; // скорость частиц в массиве CRD

x=x+vx\*dt; // определение координат частиц

y=y+vy\*dt; // определение координат частиц

z=z+vz\*dt; // определение координат частиц

// Ограничение координаты частицы при выходе на границу области моделирования:

if ((x<-d2)&&(nx==0))x=-d2; if ((y<-d2)&&(ny==0))y=-d2; if ((z<-d2)&&(nz==0))z=-d2;

if ((x>d9)&&(nx==Nx-1))x=d9; if ((y>d9)&&(ny==Ny-1))y=d9; if ((z>d9)&&(nz==Nz-1))z=d9;

nnx=0; nny=0; nnz=0; // обнулить шаг перехода в соседнюю ячейку

//Если коодината частицы выходит за пределы ячейки, то изменить номер ячейки:

if (x<-d2) nnx=-1; if (y<-d2) nny=-1; if (z<-d2) nnz=-1; if (y>=d2) nny=1; if (x>=d2) nnx=1; if (z>=d2) nnz=1;

x=x-nnx\*d; // определение новых координат частиц в новой ячейке

y=y-nny\*d; // определение новых координат частиц в новой ячейке

z=z-nnz\*d; // определение новых координат частиц в новой ячейке

nn=n+nnx+nny\*Nx+nnz\*Nx\*Ny; // определение соседней частицы

ID3[n]=0; // обнуляем массив, содержащий идентификационный номер каждой частицы в 3D-массиве

ID3[nn]=np; // определяем текущий номер частицы в массиве ID3

CRD[np\*10+9]=\*((float\*)(&nn)); // транспонируем массив из int в float

CRD[np\*10+0]=x;// определяем координату частицы в массиве CRD

CRD[np\*10+1]=y; // определяем координату частицы в массиве CRD

CRD[np\*10+2]=z; // определяем координату частицы в массиве CRD

}

n++;

}

};

";