```
/*
 * ParticleGrid.cpp
 *
 *  Created on: Nov 23, 2013
 *      Author: arjunnar
 */

#include "ParticleGrid.h"


ParticleGrid::ParticleGrid()
{
    ParticleGrid(Vector3f::ZERO, 0.6, 0.9, 0.6);
}

ParticleGrid::ParticleGrid(Vector3f origin, float sizeX, float sizeY, float sizeZ)
{
    grid = std::vector<std::list<int>>(100 * 100 * 100);
    for (int i = 0; i < grid.size(); ++i)
    {
        grid[i] = std::list<int>();
    }

    mapIndexesToGridCoords = map<int, Tuple::tuple<int, 3>>();

    sideLengthX = sizeX;
    sideLengthY = sizeY;
    sideLengthZ = sizeZ;

    topRightCorner = origin + Vector3f(sideLengthX, sideLengthY, sideLengthZ);

    // Dimensions of a single cell in the grid
    gridSideLengthX = sideLengthX / NUM_CELLS_PER_DIMEN;
    gridSideLengthY = sideLengthY / NUM_CELLS_PER_DIMEN;
    gridSideLengthZ = sideLengthZ / NUM_CELLS_PER_DIMEN;
}

ParticleGrid::~ParticleGrid()
{
}

void ParticleGrid::initializeGrid(std::vector<Vector3f> &particleLocations)
{
    for (int particleIndex = 0; particleIndex < particleLocations.size(); +
+particleIndex)
    {
        Vector3f particleLoc = particleLocations[particleIndex];

        Tuple::tuple<int, 3> gridCoords = getGridCoordinates(particleLoc);
        int newi = gridCoords[0];
        int newj = gridCoords[1];
        int newk = gridCoords[2];

        if ( !(isCoordValid(newi) && isCoordValid(newk) && isCoordValid(newj)) )
        {
            // Particle is outside of the grid
             continue;
        }

        bool oldCoordsInMap = mapIndexesToGridCoords.find(particleIndex) !=
mapIndexesToGridCoords.end();

        int oldi;
        int oldj;
        int oldk;

        if (oldCoordsInMap)
        {
            Tuple::tuple<int, 3> oldGridCoords = mapIndexesToGridCoords[particleIndex];
            oldi = oldGridCoords[0];
            oldj = oldGridCoords[1];
            oldk = oldGridCoords[2];
        }

        if (oldCoordsInMap && newi == oldi && newj == oldj && newk == oldk)
        {
            // Old grid coordinates and new grid coordinates are the same. No need to
update the grid.
            continue;
        }

        else
        {
            if (!oldCoordsInMap)
            {
                // This is the first time we are computing grid coordinates for the
particle
                grid[getGridIndex(newi, newj, newk)].push_back(particleIndex);
            }

            else
            {
                // Remove the particle from the old cell and add it to the new cell
                grid[getGridIndex(oldi, oldj, oldk)].remove(particleIndex);
                grid[getGridIndex(newi, newj, newk)].push_back(particleIndex);
                mapIndexesToGridCoords.erase(particleIndex);
            }

            // Save the new grid coordinates for the particle in our map
            mapIndexesToGridCoords.insert(std::pair<int, Tuple::tuple<int,
3>>(particleIndex, Tuple::tuple<int, 3>(newi, newj, newk)));
        }
    }
}

Tuple::tuple<int, 3> ParticleGrid::getGridCoordinates(Vector3f &particleLoc)
{
    int i = (int) ( particleLoc.x() / gridSideLengthX );
    int j = (int) ( particleLoc.y() / gridSideLengthY );
    int k = (int) ( particleLoc.z() / gridSideLengthZ );
    return Tuple::tuple<int, 3>(i, j, k);
}

std::vector<int> ParticleGrid::getNeighborParticleIndexes(int particleIndex, Vector3f
&particleLoc)
{
    std::vector<int> neighborParticleIndexes;
    Tuple::tuple<int, 3> gridCoordsOfParticle = getGridCoordinates(particleLoc);

    int iParticle = gridCoordsOfParticle[0];
    int jParticle = gridCoordsOfParticle[1];
    int kParticle = gridCoordsOfParticle[2];

    for (int iIncr = -1; iIncr <= 1; ++iIncr)
    {
        int iNeighbor = iParticle + iIncr;
        if (!isCoordValid(iNeighbor)) { continue; }

        for (int jIncr = -1; jIncr <= 1; ++jIncr)
        {
            int jNeighbor = jParticle + jIncr;
            if (!isCoordValid(jNeighbor)) { continue; }

            for (int kIncr = -1; kIncr <= 1; ++kIncr)
            {
                int kNeighbor = kParticle + kIncr;
                bool inSameCell = iNeighbor == iParticle && jNeighbor == jParticle &&
kNeighbor == kParticle;
```

```
                if (!isCoordValid(kNeighbor)) { continue; }

                std::list<int> neighborsInCell = getGridListAt(iNeighbor, jNeighbor,
kNeighbor);

                for (int neighborIndex : neighborsInCell)
                {
                    if (!inSameCell || neighborIndex != particleIndex)
                    {
                        neighborParticleIndexes.push_back(neighborIndex);
                    }
                }
            }
        }
    }

    return neighborParticleIndexes;
}

// Helper functions
inline bool ParticleGrid::isCoordValid(int val)
{
    return 0 <= val && val < NUM_CELLS_PER_DIMEN;
}

inline int ParticleGrid::getGridIndex(int i, int j, int k)
{
    return NUM_CELLS_PER_DIMEN * NUM_CELLS_PER_DIMEN * i + NUM_CELLS_PER_DIMEN * j + k;
}
inline list<int> ParticleGrid::getGridListAt(int i, int j, int k)
{
    return grid[NUM_CELLS_PER_DIMEN * NUM_CELLS_PER_DIMEN * i + NUM_CELLS_PER_DIMEN * j
+ k];
}
```