Лабораторная работа №6

Моделирование и визуализация системы N взаимодействующих тел с использованием технологий OpenGL и CUDA.

Цель работы. Использование GPU для моделирования и визуализации системы N взаимодействующих тел**.** Взаимодействие технологий CUDA и OpenGL: vbo + texture. Решение проблемы коллизий множества объектов. Создание простейшей “игры”.

Задание.

Сцена. Виртуальный куб, у которого отрисовывается только нижняя грань (пол) и ребра. Внутри куба находятся N частиц (текстурированные сферы), которые отталкиваются друг от друга и от стенок куба. Условно предполагается что стенки куба и частицы одноименно заряжены. Для частиц учитывается ускорение свободного падения. Нижняя грань куба закрашивается в соответствии с напряженностью электрического поля создаваемого частицами (строится карта напряженности).

Игрок. Камера может перемещаться по пространству без каких-либо ограничений. Управление кнопками и мышкой. При приближении камеры к частицам, они должны “убегать” от неё (предполагается наличии большого одноименного заряда у игрока). По нажатию кнопки мыши, игрок совершает “выстрел” сильно заряженной частицей в направлении взгляда. Эта частица движется равномерно и действует только на другие частицы, а на неё саму никто не влияет.

Математическая постановка.

Сторона виртуального куба $2a$, центр куба имеет координаты $(0,0,a)^T$. Обозначения:

$x_i, y_i, z_i, vx_i, vy_i, vz_i, q_i$ - положение, скорость и заряд i-ой частицы.

$x_c, y_c, z_c, vx_c, vy_c, vz_c, q_c$ - положение, скорость и заряд игрока (камеры).

$x_b, y_b, z_b, vx_b, vy_b, vz_b, q_b$ - положение, скорость и заряд пули.

$l_{ij} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2 + (z_i - z_j)^2}$ - расстояние между i-ой и j-ой частицей.

$l_{ic} = \sqrt{(x_i - x_c)^2 + (y_i - y_c)^2 + (z_i - z_c)^2}$ - расстояние между i-ой частицей и камерой.

$l_{ib} = \sqrt{(x_i - x_b)^2 + (y_i - y_b)^2 + (z_i - z_b)^2}$ - расстояние между i-ой частицей и пулей.

Все приведенные ниже параметры необходимо варьировать и подбирать экспериментально, чтобы получающийся результат был наиболее реалистичным и красивым.

$a = 15$ - половина стороны куба, $N = 150$ - количество частиц, $\varepsilon = 10^{-3}$ - маленькое число, которое прибавляется в знаменатель, чтобы избежать деление на близкое к нулю значение, $w = 0.99$ - коэффициент замедления, $K = 50$ - коэффициент пропорциональности, $dt = 0.01$ - шаг интегрирования, $g = 20$ - “ускорение свободного падения”, $q_i = 1,\ q_c = 30,\ q_b = 50$ - заряды частиц, камеры и пули, $v_b = 10$ - скорость пули, $shift_z = 0.75$ - сдвиг для карты напряженности, для более четкой картины.

Начальные положения и скорости частиц:

$$x_i^{(0)} = rand(-a, a),\ y_i^{(0)} = rand(-a, a),\ z_i^{(0)} = rand(0, 2a)$$
$$vx_i^{(0)} = vy_i^{(0)} = vz_i^{(0)} = 0$$

На каждом шаге положения и скорости частиц пересчитываются по формулам:

$$vx_i^{(k+1)} = w \cdot vx_i^{(k)} + K \cdot q_i \left[\sum_{j=1}^{N} q_j \frac{x_i^{(k)}-x_j^{(k)}}{l_{ij}^3+\varepsilon} + q_i \frac{x_i^{(k)}-a}{\left|x_i^{(k)}-a\right|^3+\varepsilon} + q_i \frac{x_i^{(k)}+a}{\left|x_i^{(k)}+a\right|^3+\varepsilon} + q_c \frac{x_i^{(k)}-x_c^{(k)}}{l_{ic}^3+\varepsilon} + q_b \frac{x_i^{(k)}-x_b^{(k)}}{l_{ib}^3+\varepsilon} \right] dt$$
$$vy_i^{(k+1)} = w \cdot vy_i^{(k)} + K \cdot q_i \left[\sum_{j=1}^{N} q_j \frac{y_i^{(k)}-y_j^{(k)}}{l_{ij}^3+\varepsilon} + q_i \frac{y_i^{(k)}-a}{\left|y_i^{(k)}-a\right|^3+\varepsilon} + q_i \frac{y_i^{(k)}+a}{\left|y_i^{(k)}+a\right|^3+\varepsilon} + q_c \frac{y_i^{(k)}-y_c^{(k)}}{l_{ic}^3+\varepsilon} + q_b \frac{y_i^{(k)}-y_b^{(k)}}{l_{ib}^3+\varepsilon} \right] dt$$
$$vz_i^{(k+1)} = w \cdot vz_i^{(k)} + K \cdot q_i \left[\sum_{j=1}^{N} q_j \frac{z_i^{(k)}-z_j^{(k)}}{l_{ij}^3+\varepsilon} + q_i \frac{z_i^{(k)}-2a}{\left|z_i^{(k)}-2a\right|^3+\varepsilon} + q_i \frac{z_i^{(k)}}{\left|z_i^{(k)}\right|^3+\varepsilon} + q_c \frac{z_i^{(k)}-z_c^{(k)}}{l_{ic}^3+\varepsilon} + q_b \frac{z_i^{(k)}-z_b^{(k)}}{l_{ib}^3+\varepsilon} \right] dt - g \cdot dt$$
$$x_i^{(k+1)} = x_i^{(k)} + vx_i^{(k+1)} dt$$
$$y_i^{(k+1)} = y_i^{(k)} + vy_i^{(k+1)} dt$$
$$z_i^{(k+1)} = z_i^{(k)} + vz_i^{(k+1)} dt$$

Слагаемые $q_i \frac{x_i^{(k)}-a}{\left|x_i^{(k)}-a\right|^3+\varepsilon},\ q_i \frac{x_i^{(k)}+a}{\left|x_i^{(k)}+a\right|^3+\varepsilon}$ отвечают за отталкивание от правой и левой стенки.

Слагаемые $q_i \frac{y_i^{(k)}-a}{\left|y_i^{(k)}-a\right|^3+\varepsilon},\ q_i \frac{y_i^{(k)}+a}{\left|y_i^{(k)}+a\right|^3+\varepsilon}$ отвечают за отталкивание от передней и задней стенки.

Слагаемые $q_i \frac{z_i^{(k)}-2a}{\left|z_i^{(k)}-2a\right|^3+\varepsilon}, q_i \frac{z_i^{(k)}}{\left|z_i^{(k)}\right|^3+\varepsilon}$ отвечают за отталкивание от пола и потолка.

Слагаемые $q_c \frac{x_i^{(k)}-x_c^{(k)}}{l_{ic}^3+\varepsilon},\ q_c \frac{y_i^{(k)}-y_c^{(k)}}{l_{ic}^3+\varepsilon},\ q_c \frac{z_i^{(k)}-z_c^{(k)}}{l_{ic}^3+\varepsilon}$ отвечают за отталкивание от игрока.

Слагаемые $q_b \frac{x_i^{(k)}-x_b^{(k)}}{l_{ib}^3+\varepsilon},\ q_b \frac{y_i^{(k)}-y_b^{(k)}}{l_{ib}^3+\varepsilon},\ q_b \frac{z_i^{(k)}-z_b^{(k)}}{l_{ib}^3+\varepsilon}$ отвечают за отталкивание от пули.

Так как на пулю ничего не влияет, поэтому она движется с постоянной скоростью:

$$x_b^{(k+1)} = x_b^{(k)} + vx_b dt$$
$$y_b^{(k+1)} = y_b^{(k)} + vy_b dt$$
$$z_b^{(k+1)} = z_b^{(k)} + vz_b dt$$

В момент выстрела вектор скорости пули совпадает с направление объектива камеры:

$$vx_b = v_b cos(yaw)\ cos(pitch)$$

$$vy_b = v_b sin(yaw)\ cos(pitch)$$

$$vz_b = v_b sin(pitch)$$

А начальное положение “немного” смещено относительно положения камеры:

$$x_b^{(0)} = x_c + vx_b$$

$$y_b^{(0)} = y_c + vy_b$$

$$z_b^{(0)} = z_c + vz_b$$

Для простоты можно полагать что пуля всего одна, и в момент выстрела она просто меняет свое положение и скорость.

Вычисление карты напряженности. Пусть текстура квадратная и имеет размер $np \times np$. Переход от текстурных координат $(i,\ j)$ к реальным $(x,\ y)$:

$$x = \left(2\frac{i}{np} - 1\right)a,\ y = \left(2\frac{j}{np} - 1\right)a$$

Напряженность вычисляется по формуле:

$$E = K \cdot \left[\sum_{p=1}^{N} \frac{q_p}{(x_p-x)^2+\left(y_p-y\right)^2+(z_p-shift_z)^2+\varepsilon} + \frac{q_b}{(x_b-x)^2+(y_b-y)^2+(z_b-shift_z)^2+\varepsilon}\right]$$

Цвет пикселя $(i,\ j)$ можно определить например так:

$$(0, 0, min(E, 255))^T$$

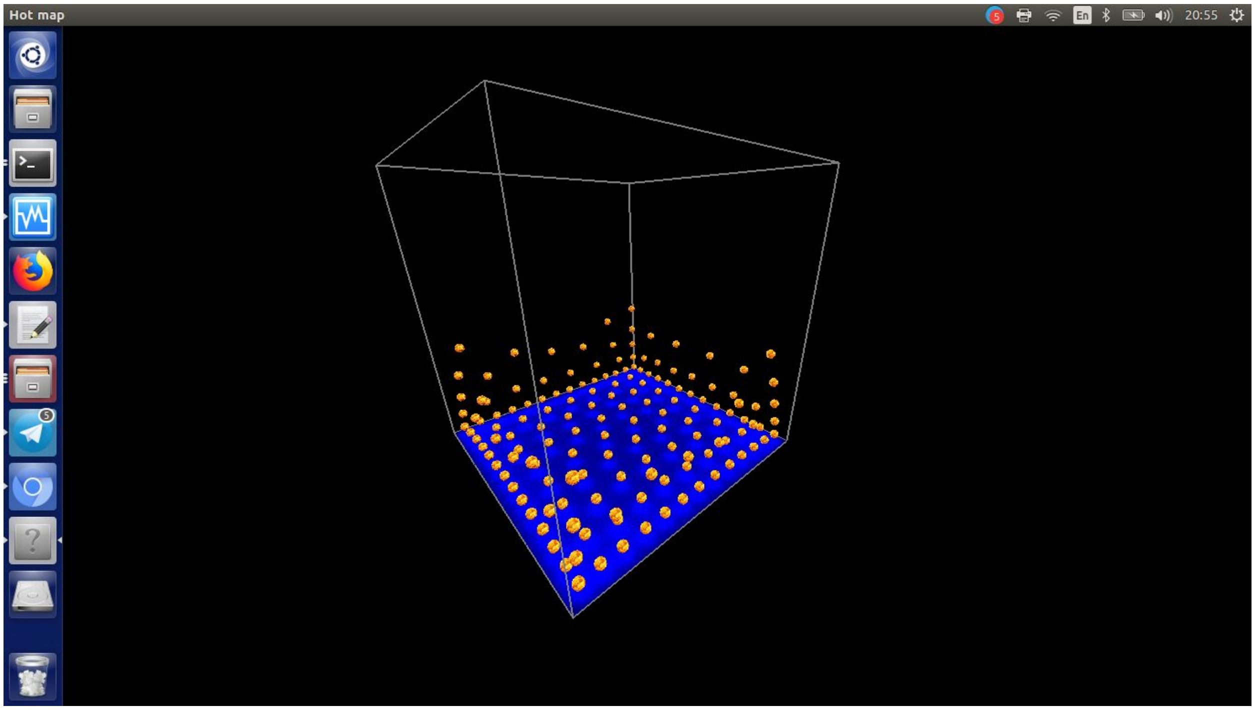
Программа должна компилироваться следующей командой:

/usr/local/cuda-7.0/bin/nvcc --std=c++11 -Werror cross-execution-space-call -lm -lcublas -lcurand -lGL -lGLU -lglut -lGLEW

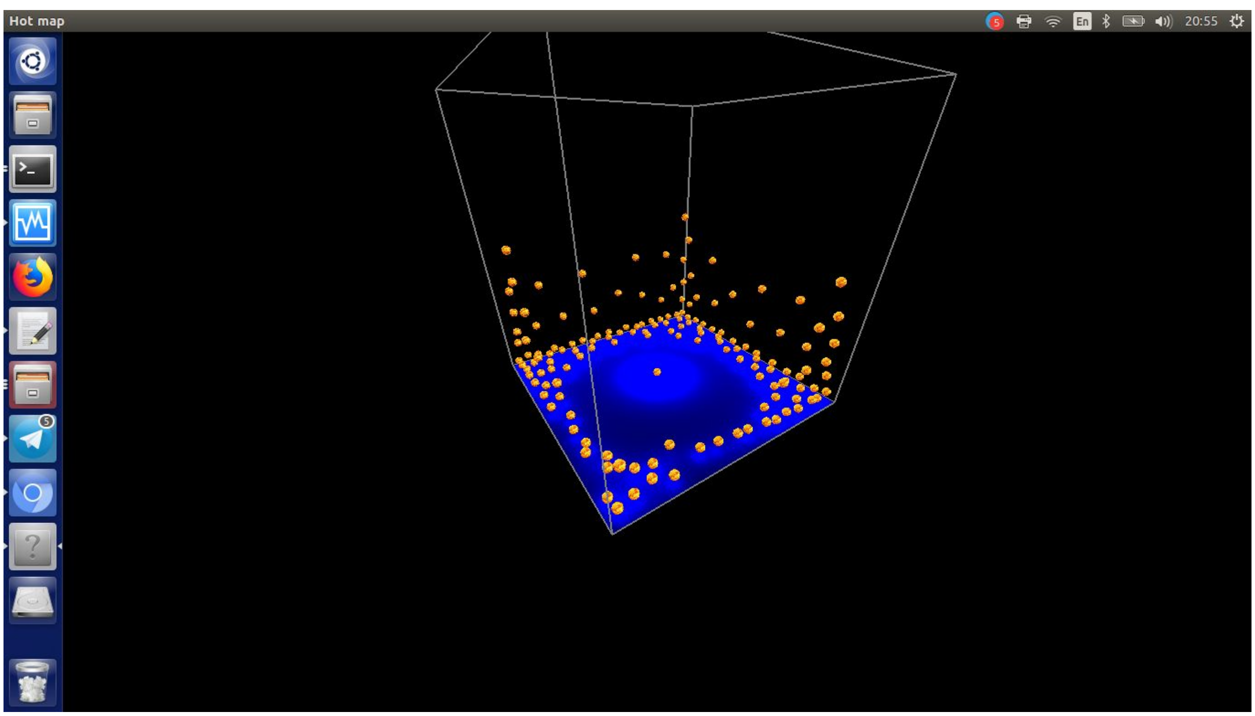
и отправлена на чекер с темой письма pgp:6

Примеры.

Если не перемещаться по сцене и не стрелять, то спустя некоторое время все частицы стабилизируются и не будут двигаться:



После выстрела:



Когда игрок подошел близко, частицы расступаются: