САНКТ-ПЕТЕРБУРГСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ПЕТРА ВЕЛИКОГО

Институт прикладной математики и механики

Кафедра теоретической механики

# Отчёт по дисциплине «Компьютерные технологии»
# «Моделирование столкновения плоской пули с стеной»

Выполнил: студент группы 53604/1
Черногорский Вячеслав Константинович

Руководитель:
Ле-Захаров А.А.

Санкт-Петербург
2015

# Содержание

## Постановка задачи

Одной из интересных задач для моделирования взаимодействия частиц является задача моделирования столкновения пули с стеной. Результаты моделирования данной задачи позволяют наглядно увидеть особенности взаимодействия частиц разного рода материалов, влияние скоростей, массы и диаметра частиц на эти взаимодействия. Еще одна важная прикладная особенность заключается в том, что при добавлении большего числа внешних сил на систему, можно моделировать более реальные ситуации столкновения пули с различными объектами и пытаться снизить последствия этих взаимодействий к нулю.

Целью данной задачи являлось создание примитивной рабочей модели столкновения стальной пули с бетонной стенкой. На данном этапе, взаимодействие частиц между собой происходит через потенциал Леннард-Джонса.

## Реализация

Для реализации опыта была разработана библиотека классов ParticlesSimulation на языке C# с использованием основ ООП. Библиотека позволяет создавать модель пространства, заполненного частицами, после чего может определять динамическое поведение созданной системы для выбранного потенциала парного взаимодействия. Были разработаны следующие классы:

1. *Constants.cs* — класс, содержащий основные константы для всех вычислений — массы, заряды и диаметры частиц; время и шаг интегрирования; константы, задающие потенциал;
2. *Creator.cs* — класс, предназначенный для создания системы Space: заполнение её частицами Particle с заданными массами, зарядами, координатами и скоростями;
3. *IntegratorBase.cs* — класс, позволяющий рассчитывать поведение системы Space, используя потенциал гармонического осциллятора;

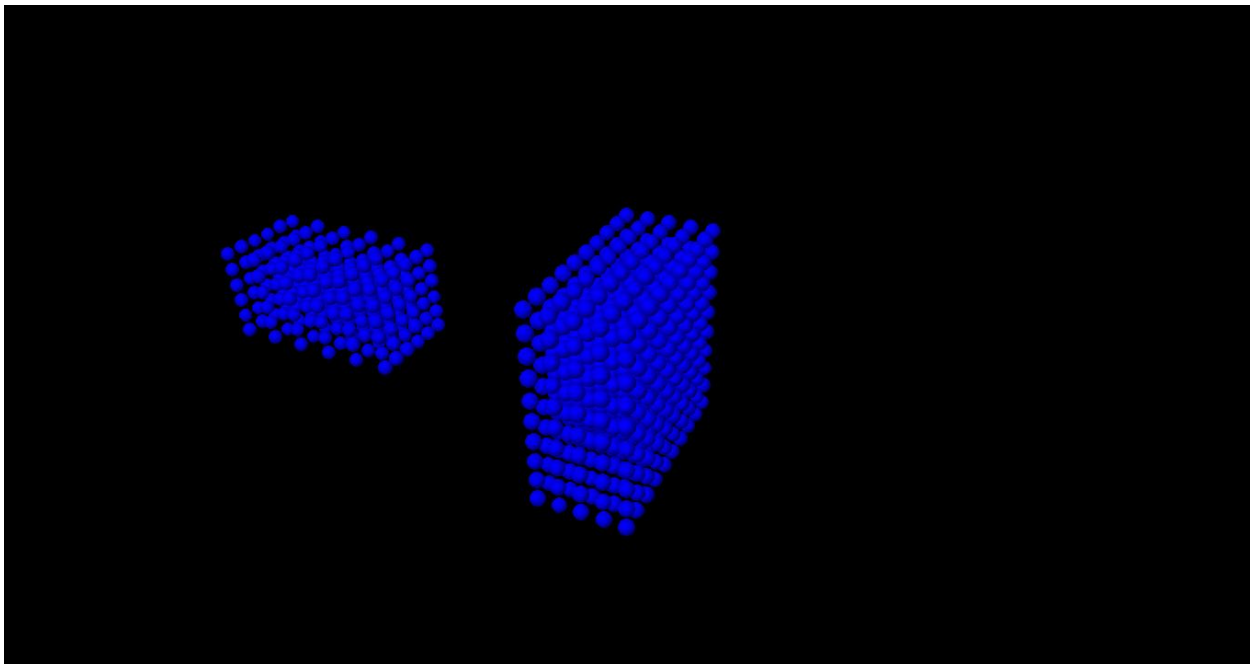
4. *IntegratorLJ.cs* — производный класс IntegratorBase.cs, позволяющий рассчитывать поведение системы Space, используя потенциал Леннард-Джонса;
5. *IntegratorQ.cs* — производный класс IntegratorBase.cs, позволяющий рассчитывать поведение системы Space, используя потенциал Кулона;
6. *OutputHelper.cs* — класс, позволяющий выводить данные системы Space в текстовый файл;
7. *OutputHelperA3R.cs* — производный класс OutputHelper.cs, позволяющий выводить данные системы Space в формате, пригодном для визуализации в программе A3R.
8. *Particle.cs* — класс, описывающий частицу;
9. *Space.cs* — класс, описывающий систему;
10. *Vector3D.cs* — класс, описывающий трехмерные вектора;

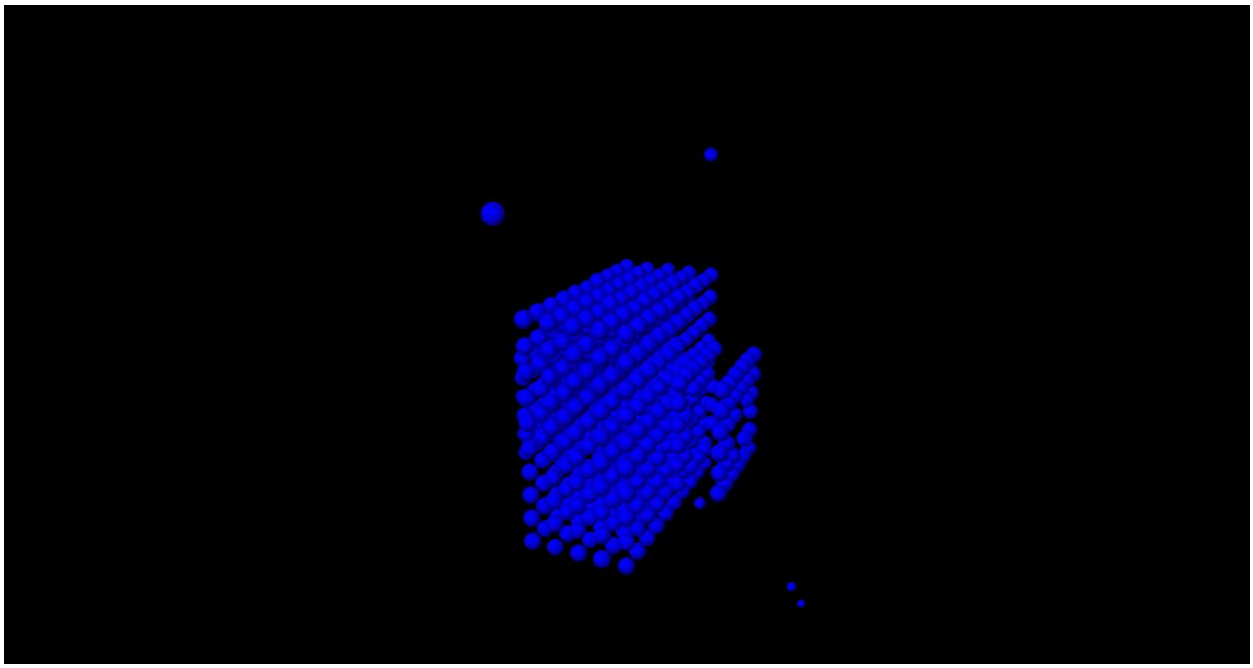
## Создание модели и расчет взаимодействия частиц

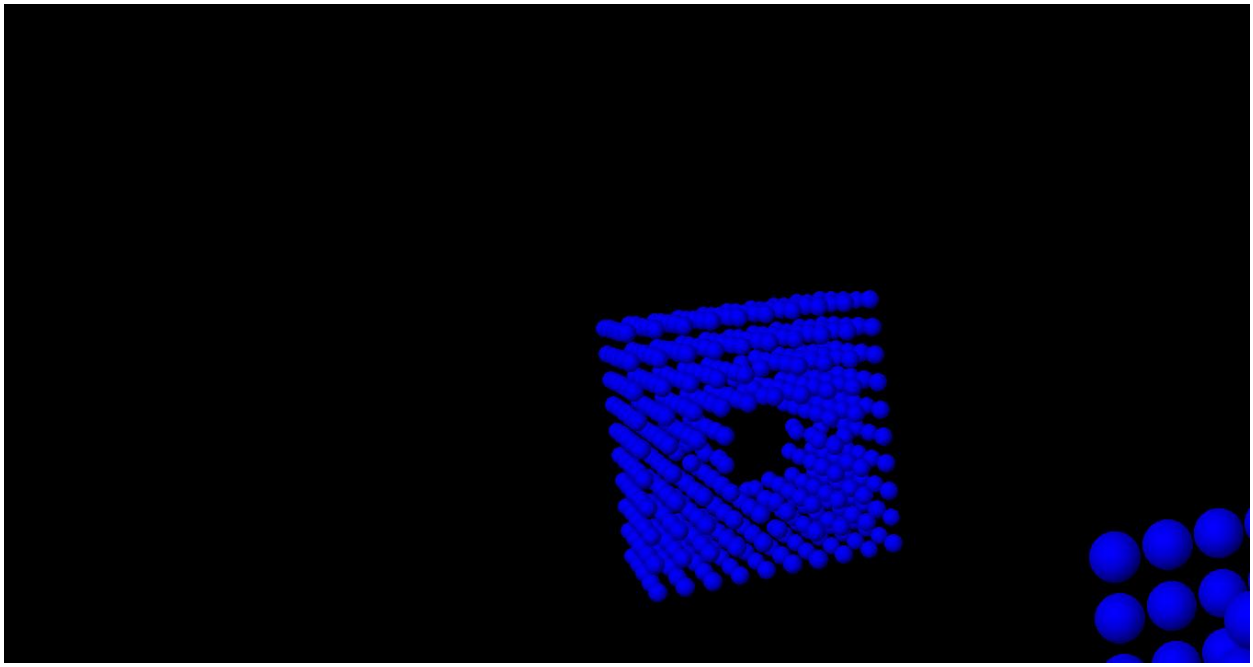
Для создания модели, была создана упрощенная модель пули в виде параллелепипеда с размерами граней 5x5x10 частиц, диаметр частиц – 5 у.е., а так же с другими заданными параметрами указанными в приложении Б. Для стенки бетона была выбрана модель параллелепипеда с размерами граней 8x8x5 частиц, диаметр частиц 1. Воспользовавшись справочной литературой, было установлено, что плотность бетона варьируется в пределах от $0{,}6\ ^{гр}/_{см^3}$ («легкий бетон») до $2{,}6\ ^{гр}/_{см^3}$(«особо тяжелый»), а плотность никелированной стали $7{,}8\ ^{гр}/_{см^3}$. Выполнив не сложные расчеты, было установлено, что даже при рассмотрении «легкого бетона», количество частиц в пуле в 80 раз больше чем в бетоне того же объема. Так как подобная задача требует серьезных ресурсозатрат, было решено увеличить диаметр частицы пули в 8 раз, а количество частиц в 13 раз в сравнении с количеством частиц в стенке. Таким образом была смоделирована задача  о столкновении пули с стенкой.

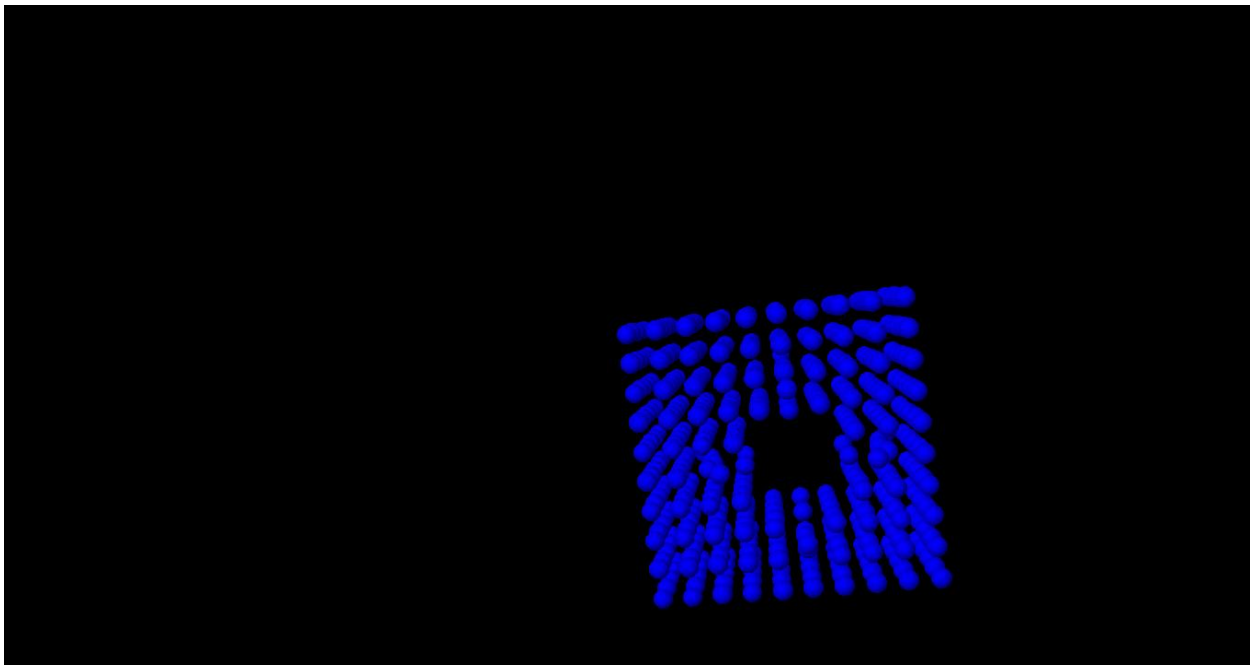
# Результаты

В результате моделирования было замечено, что пуля проходит сквозь стенку насквозь, оставляя после себя ровное отверстие соответствующее размерам пули. Однако вскоре после прохождения пули, за счет взаимодействия частиц между собой в стенке, данное отверстие «затягивается». Это можно объяснить отталкиванием соседних частиц по потенциалу Леннард-Джонсона в следствии нарушения первоначальной структуры.









# Выводы

Была успешно построена модель и сделан расчет взаимодействия частиц разного вида при столкновении пули и стенки. Данная модель может быть развита для дальнейших, более сложных условий взаимодействия частиц. Также получен опыт реализации численных вычислений на языке C#, изучены основы и преимущества объектно-ориентированного подхода к написанию решения задач.

# Приложение А.
# **Creator.cs**

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace ParticlesSimulation
{
     public class Creator
    {
         public Space Create()
         {
             Space model = new Space();


              for (int i = 0; i < 10; ++i) //setka
              {
                  for (int j = 0; j < 10; ++j)
                  {
                      for (int k = 0; k < 5; ++k)
                      {
                          Particle p = new Particle(i * 6.0, j * 6.0, k * 6.0);
                          //p.D = Constants.DefaultD * 5;
                          p.M = Constants.DefaultMass * 2000;
                          model.Add(p);
                      }
                  }
              }

              for (int i = 0; i < 6; ++i) //setka
              {
                  for (int j = 0; j < 6; ++j)
                  {
                      for (int k = 0; k < 6; ++k)
                      {
                          Particle p2 = new Particle(i * 5.0 + 15, j * 5.0 + 15, k * 10.0
+ 80.0);
                          p2.D = Constants.DefaultD * 5;
                          p2.M = Constants.DefaultMass * 26000;
                          p2.V.z = -40.0;
                          p2.V.x = 0.0;
                          p2.V.y = 0.0;
                          model.Add(p2);
                      }
                  }
              }



             return model;
         }
    }
       }
```

# IntegratorLJ.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace ParticlesSimulation
{
    public class IntegratorLJ : IntegratorBase
    {
        public IntegratorLJ(Space s) : base(s) { }

        public override void RecalcParticleForce(Particle pi, Particle pj)
        {
            double a = (pi.D / 2 + pj.D / 2);
            double r = (pi.R - pj.R).Norm();
            double modfij = (12.0/a)*(Math.Pow(a/r,12.0)-Math.Pow(a/r,7.0));
            pi.F = pi.F + (modfij / (pi.R - pj.R).Norm()) * (pj.R - pi.R);
        }
    }
}
```

# Constants.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace ParticlesSimulation
{
   public static class Constants
    {
       public static readonly double
           MinMass = 0.001, DefaultMass = 1, DefaultD = 1, MinDistance = 0.0001, dt =
0.05, MaxTime = 200.0, K=1.0, DefaultQ = 1.0;


    }
}
```