ГБОУ "Президентский ФМЛ №239"

**Моделирование поведения заряженных частиц в электромагнитном поле**

*Годовой проект по информатике*

# Работу выполнил Ученик 10-2 класса Поляков Владислав

Учитель Информатики: Филатов Владимир Константинович

### Санкт-Петербург 2017

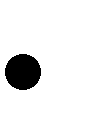
**1.Постановка задачи**

#### .Формулировка задачи

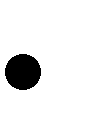
Необходимо найти, как именно будут двигаться заряженные частицы в электромагнитном поле. Программа должна создать анимацию движения этих двух частиц под действием изначально заданного поля.

**2.Уточнение исходных и выходных данных**

#### Исходные данные

 Дано электромагнитное поле, заданное шестью алгебраическими функциями - проекциями векторов B (магнитной индукции) и E (напряженности электрического поля) на координатные оси.

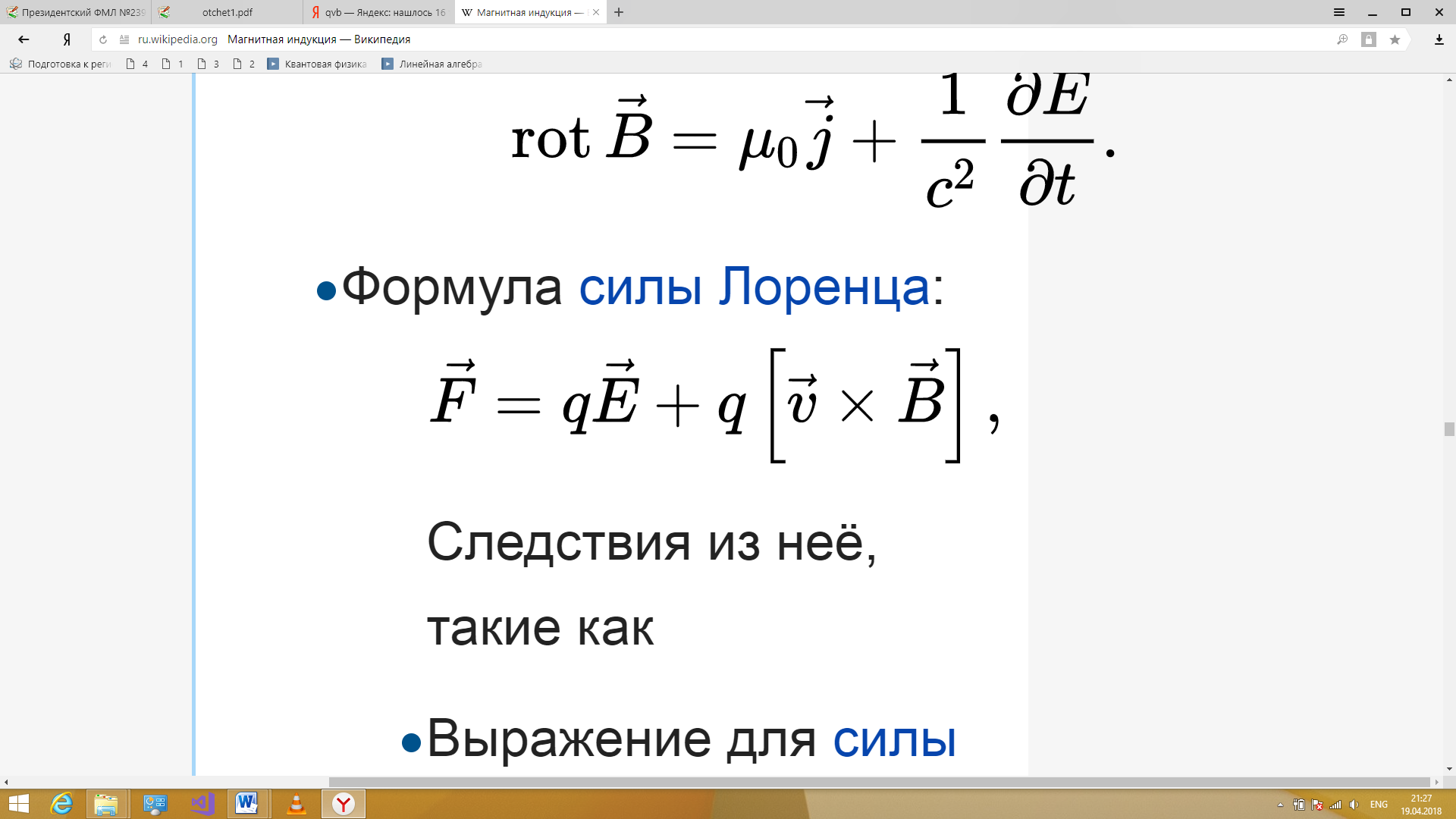
#### Выходные данные

 Наглядная демонстрация движения частиц.

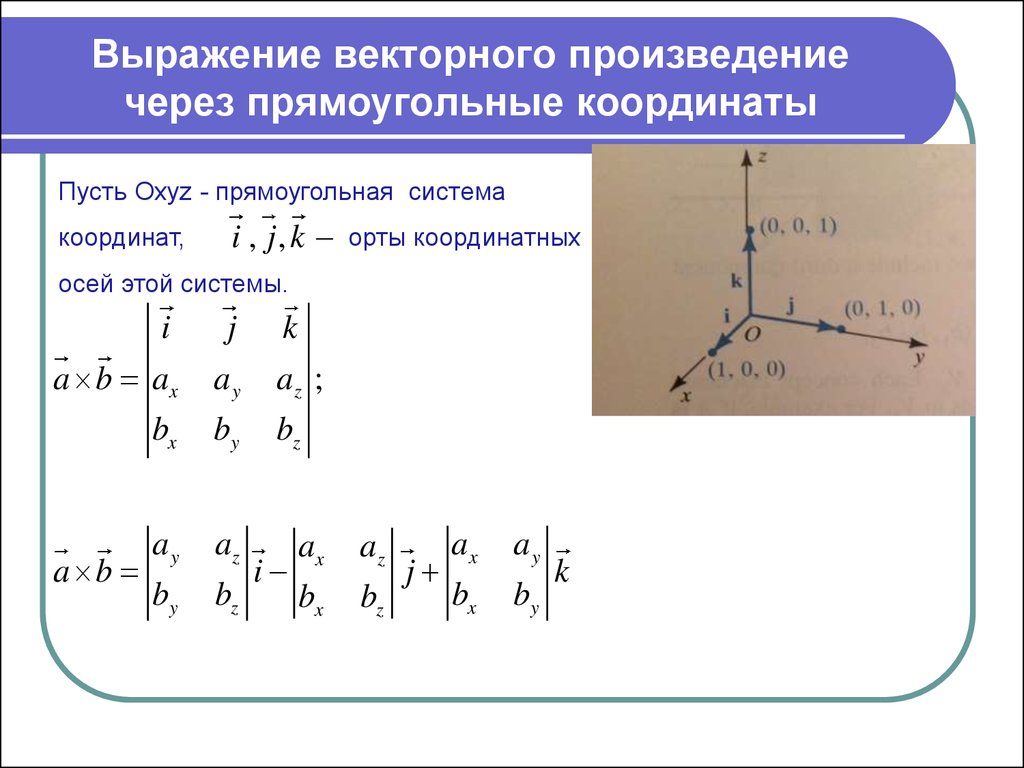
## 3.Математическая модель

#### 3.1 Используемые формулы

#### Сила, действующая на заряженное тело.



#### Разложение векторного произведения по базису.



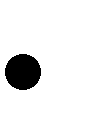
#### Формулы для пути, пройденного телом за малый промежуток времени и изменения скорости тела за этот же промежуток.

#### 

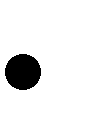
#### 

## 4.Анализ используемой структуры данных

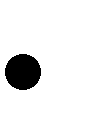
#### Входные данные

 Для каждой частицы в специальном окне вводятся данные из пункта 2.1.

#### Промежуточные данные

 Во время работы программы для каждой частицы хранятся её актуальные параметры.

#### Выходные данные

 Анимация движения частиц с возможностью просмотра их текущих параметров.

## 5.Метод решения

#### Решение задачи

5.1.1 *Выбор ПО*

Для решения данной задачи был выбран игровой движок Unity3D.   
Этот выбор обусловлен тем, что в Unity уже существуют некоторые стандартные объекты (например, шар), работать с которыми очень удобно. Их особенность состоит в том, что для них существует компонент Rigidbody, который позволяет сопоставлять объекту такие физические параметры, как скорость и положение. Эта функция позволяет, например, каждый такт не вручную перерисовывать объект на экране, а сразу задать его смещение.  
 Ещё один большой плюс Unity-встроенная возможность работы с векторами.

5.1.2 *Алгоритм*

В трехмерном пространстве для каждой частицы создается объект типа «сфера». После этого для каждого объекта создается скрипт, обеспечивающий обновление параметров частиц.

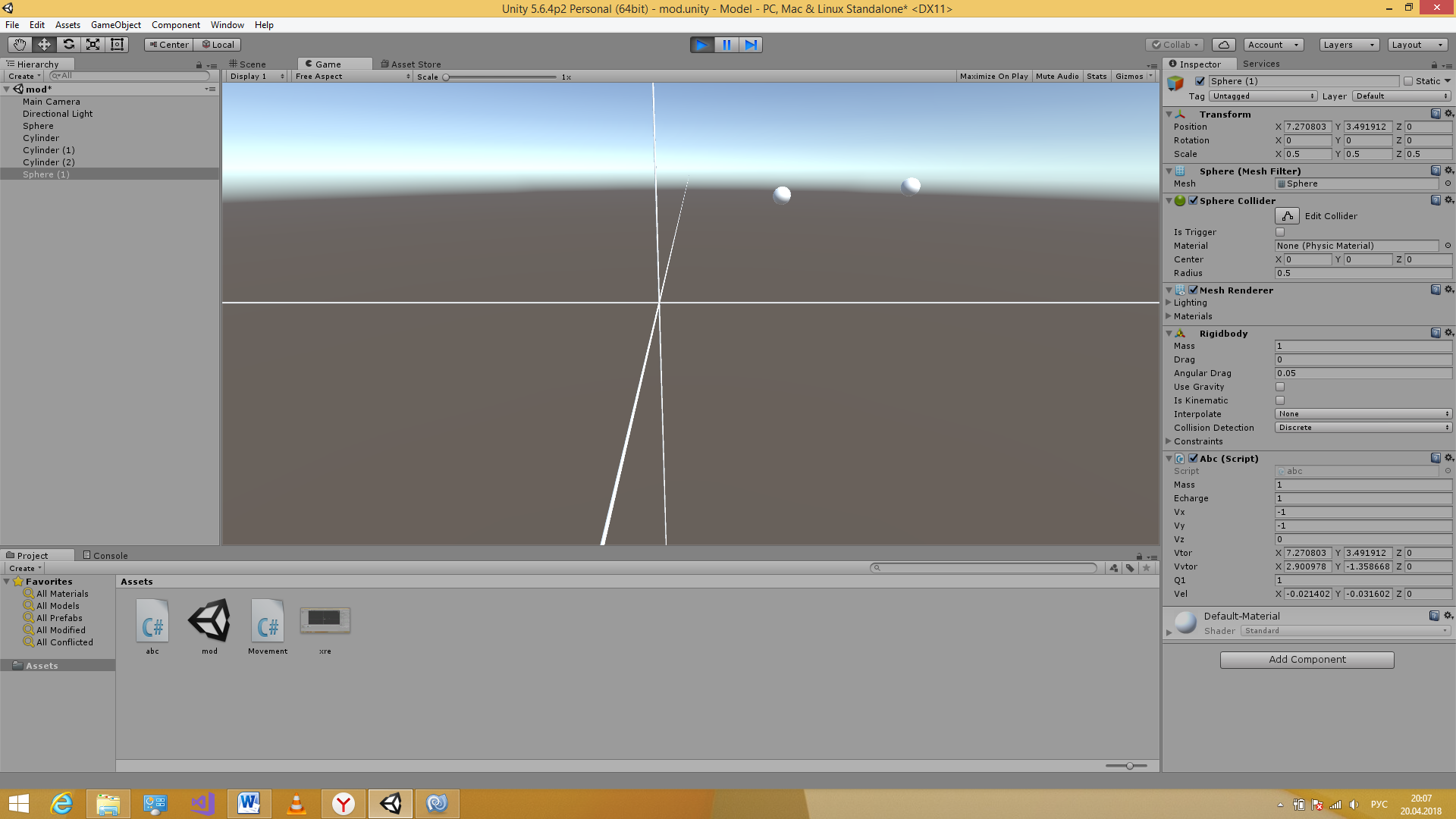
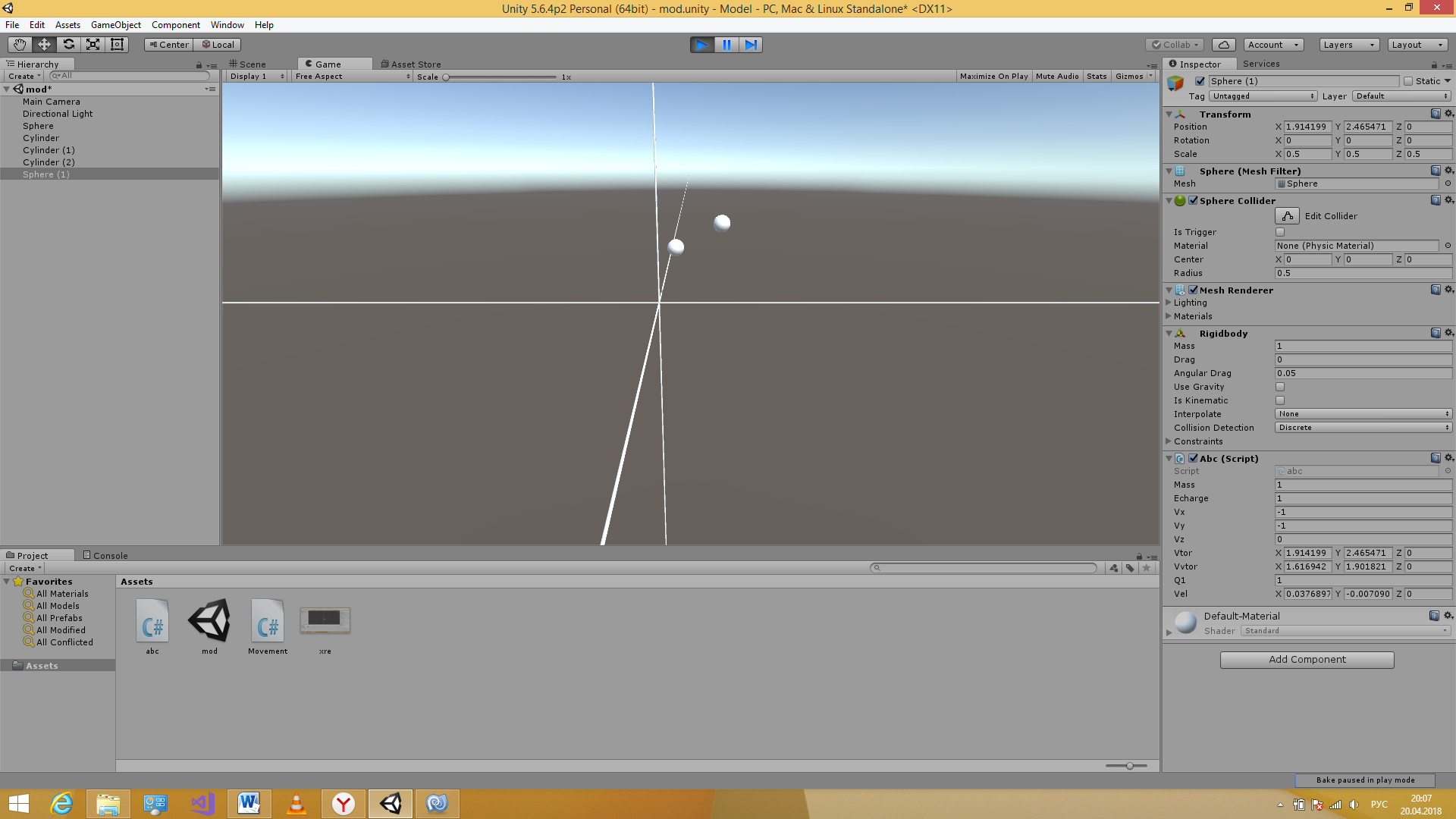
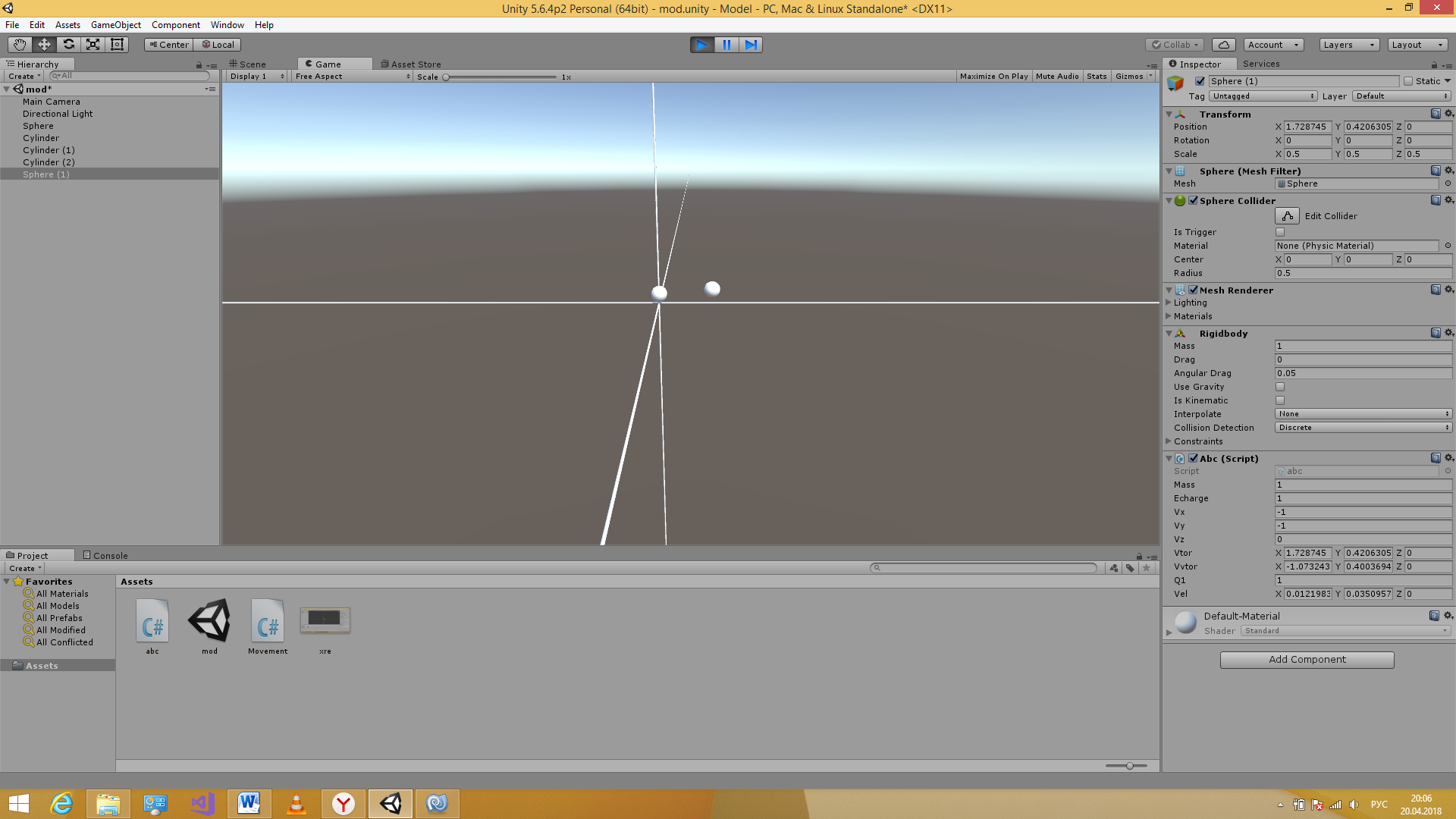
5.1.3 *Скрипт*

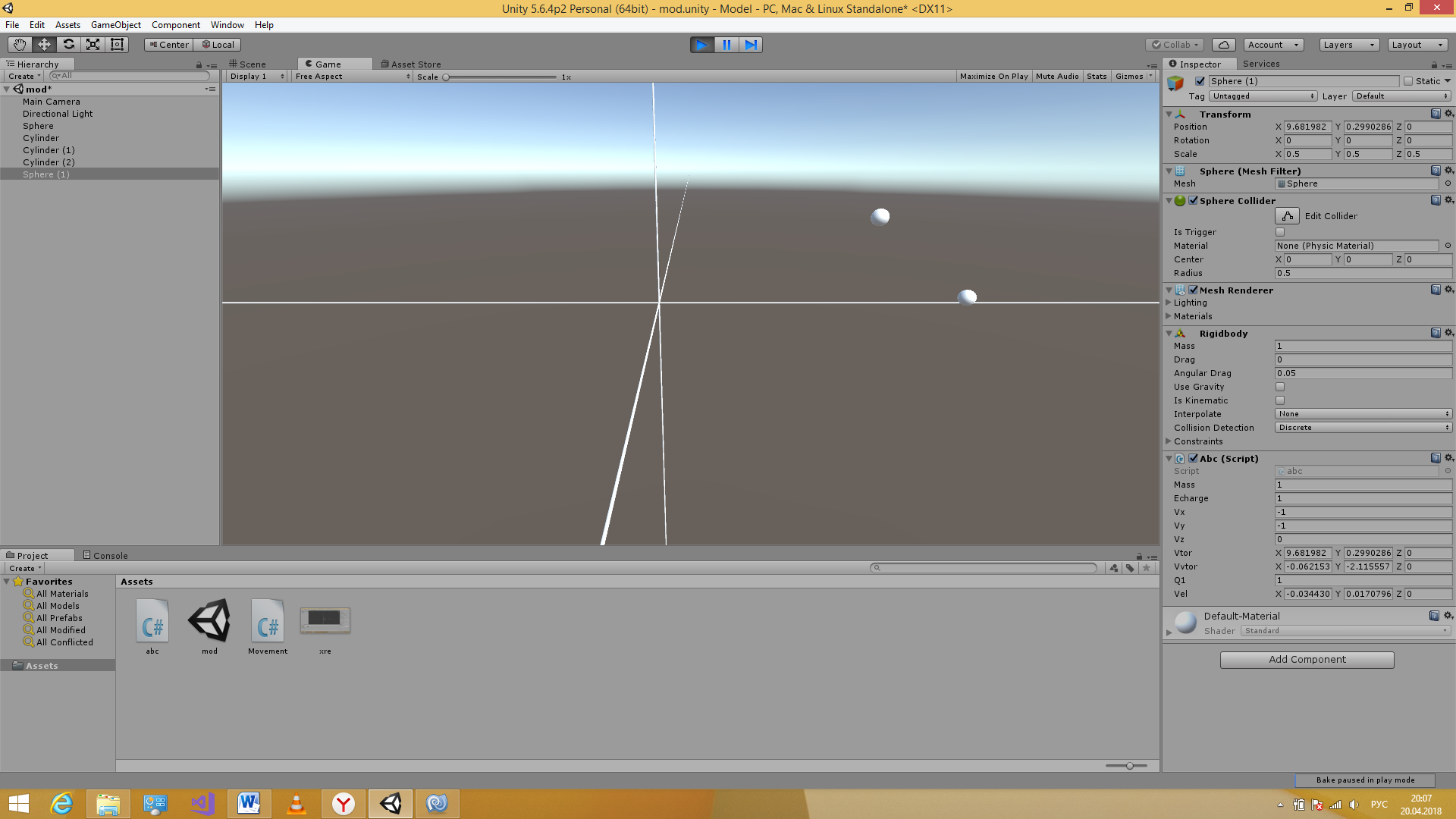
С частотой 60 раз в секунду выполняется следующие инструкции: Используя описанные в п.3 физические законы вычисляется необходимые параметры: Сначала вычисляется сила (F). Далее из силы находится ускорение (a). Из ускорения получаем необходимые величины: изменение скорости (dV) и координаты (dS). После необходимых вычислений обновляем изображение.

**6.Комментрированный листинг**

|  |
| --- |
| using System.Collections; |
|  | using System.Collections.Generic; |
|  | using UnityEngine; |
|  |  |
|  | public class Movement : MonoBehaviour { //класс, описывающий поведение объекта |
|  | public float mass; /\*начальные данные |
|  | public float echarge; |
|  | public float vx; |
|  | public float vy; |
|  | public float vz; \*/ |
|  | public Vector3 perv = new Vector3 (0, 0, 0); |
|  | public Vector3 vperv = new Vector3 (0, 0, 0); |
|  | Vector3 vtor = new Vector3 (0, 0, 0); |
|  | Vector3 vvtor = new Vector3 (0, 0, 0); |
|  | public float q2; |
|  | Rigidbody rb; //компонент, позволяющий использовать физику |
|  |  |
|  | public static Vector3 vzaim (Vector3 r1,Vector3 r2,Vector3 v1,Vector3 v2,float qu1,float qu2){ |
|  | Vector3 x = new Vector3 (0, 0, 0); |
|  | x = r1 - r2; |
|  | float y=Mathf.Sqrt(x.x\*x.x+x.y\*x.y+x.z\*x.z); |
|  | x \*= qu1 \* qu2 / (y \* y \* y); |
|  | return x; |
|  | } //функция, возвращающая вектор силы взаимодействия двух тел |
|  |  |
|  |  |
|  |  |
|  | public static float elx (Vector3 t){ |
|  | return 0; |
|  | } |
|  | public static float ely (Vector3 t){ |
|  | return 1; |
|  | } |
|  | public static float elz (Vector3 t){ |
|  | return 0; |
|  | } |
|  |  |
|  |  |
|  |  |
|  | public static float mx (Vector3 t){ |
|  | return 0; |
|  | } |
|  | public static float my (Vector3 t){ |
|  | return 0; |
|  | } |
|  | public static float mz (Vector3 t){ |
|  | return 1; |
|  | }//функции, задающие электромагнитное поле. В данном случае это |
|  | //однородние поле с едичной магнитной индукцией и напряжённостью |
|  |  |
|  |  |
|  |  |
|  | void Start () {//метод, в котором задаются начальные условия для тел  //входные данные привязываются к объекту |
|  | rb = GetComponent<Rigidbody> (); |
|  | Vector3 speed = new Vector3 (vx,vy,vz); |
|  | perv = rb.position; |
|  | vperv = rb.velocity; |
|  | vtor = GameObject.Find ("Sphere (1)").GetComponent<abc> ().vtor; |
|  | vvtor = GameObject.Find ("Sphere (1)").GetComponent<abc> ().vvtor; |
|  | q2 = GameObject.Find ("Sphere (1)").GetComponent<abc> ().echarge; |
|  | rb.velocity = speed; |
|  | } |
|  | public Vector3 vel = new Vector3 (0,0,0); |
|  | void Update () {//метод, повторяющийся каждый такт |
|  | float fx,fy,fz; |
|  | float dt = 1/60f; |
|  | perv = rb.position; |
|  | vperv = rb.velocity; |
|  | vtor = GameObject.Find ("Sphere (1)").GetComponent<abc> ().vtor; |
|  | vvtor = GameObject.Find ("Sphere (1)").GetComponent<abc> ().vvtor; |
|  | Vector3 abc = new Vector3 (0,0,0); |
|  | abc = vzaim (rb.position, vtor, rb.velocity, vvtor, echarge, q2);  //считается взаимодействие тел |
|  | fx = echarge\*(elx(rb.position)+rb.velocity.y\*mz(rb.position)-rb.velocity.z\*my(rb.position))+abc.x; |
|  | fy = echarge\*(ely(rb.position)+rb.velocity.z\*mx(rb.position)-rb.velocity.x\*mz(rb.position))+abc.y; |
|  | fz = echarge\*(elz(rb.position)+rb.velocity.x\*my(rb.position)-rb.velocity.y\*mx(rb.position))+abc.z;  //изменяются скорости и кординаты. |
|  | vel.x=(fx/mass)\*dt; |
|  | vel.y=(fy/mass)\*dt; |
|  | vel.z=(fz/mass)\*dt; |
|  | rb.velocity = rb.velocity + vel; |
|  | } |
|  | } |

## 7.Пример работы программы





**8.Анализ правильности решения**

Данный метод решения при стремлении времени между шагами к 0 является бесконечно точным. Ввиду конечности этого промежутка времени допускается ошибка. При небольших скоростях движения расчёты и качественный анализ показывают, что визуально данное движение почти неотличимо от истинного. При больших скоростях наблюдались значительные отклонения. Эту проблему можно разрешить масштабированием (увеличением или уменьшением некоторых величин таким образом, чтобы качественно траектория оставалась прежней, но параметры объектов менялись (также возможно уменьшение скорости)) или уменьшением времени шага. Однако последний вариант может значительно сказаться на производительности.