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 2018

1.Постановка задачи

1.1.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.

1.2 Цели проекта

Абсолютно точное вычисление траектории не требуется. Для точных измерений (например, для вычисления места падения метеорита на землю) чаще всего требуется единственная точка этой траектории. В случае с метеоритом - это точка его падения. Для нахождения этой точки требуется произвести очень точную оценку решения дифференциальных уравнений, описывающих движение тела. С более чем достаточной точностью это могут сделать даже такие общедоступные сайты, как Wolfram Alpha. Этот проект же должен показать анимацию движения заряженных тел, с сохранением качественных особенностей движения.

1.3 Практическое применение

Для примера можно рассмотреть следующую задачу.

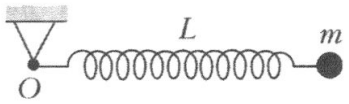


Рис. 32

В однородном поле тяжести находится тело массой m, которое прикреплено пружиной к точке подвеса, находящейся на одном с ним уровне. Жёсткость пружины k, начальная длина L, длина недеформированной пружины мала. Требуется найти удлинение пружины в тот момент, когда она вертикальна.

На первый взгляд, нерешаемая аналитически задача. Но описав электрическое поле так, чтобы сила, действующая на тело в каждой точке пространства, была такой же, как и в этой задаче, можно оценить, как будет двигаться тело в таком поле, а, следовательно, и найти приближенный ответ.

Моделирование показало, что тело будет двигаться по прямой. Этот результат оказался полезнее полученного в Wolfram Alpha тем, что **появилось предположение о том, что траектория – прямая**. А доказать, что это прямая, уже гораздо проще, чем решить исходную задачу напрямую. Дальнейшие расчеты довольно просты. В этом случае качественный анализ зависимости дал более полезную информацию, чем точнейший ответ Wolfram Alpha (гипотеза, возникшая при моделировании, позволила прийти к общей формуле для ответа, а не получить конкретное число для конкретных значений m, k и L).

2.Уточнение исходных и выходных данных

2.1. Исходные данные

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

2.2. Выходные данные

- Наглядная демонстрация движения частиц.

3.Математическая модель

3.1 Используемые формулы

Сила, действующая на заряженное тело (формула 1).

$$\vec{F} = q\vec{E} + q\left[\vec{v} \times \vec{B}\right]$$

Разложение векторного произведения по базису (формула 2).

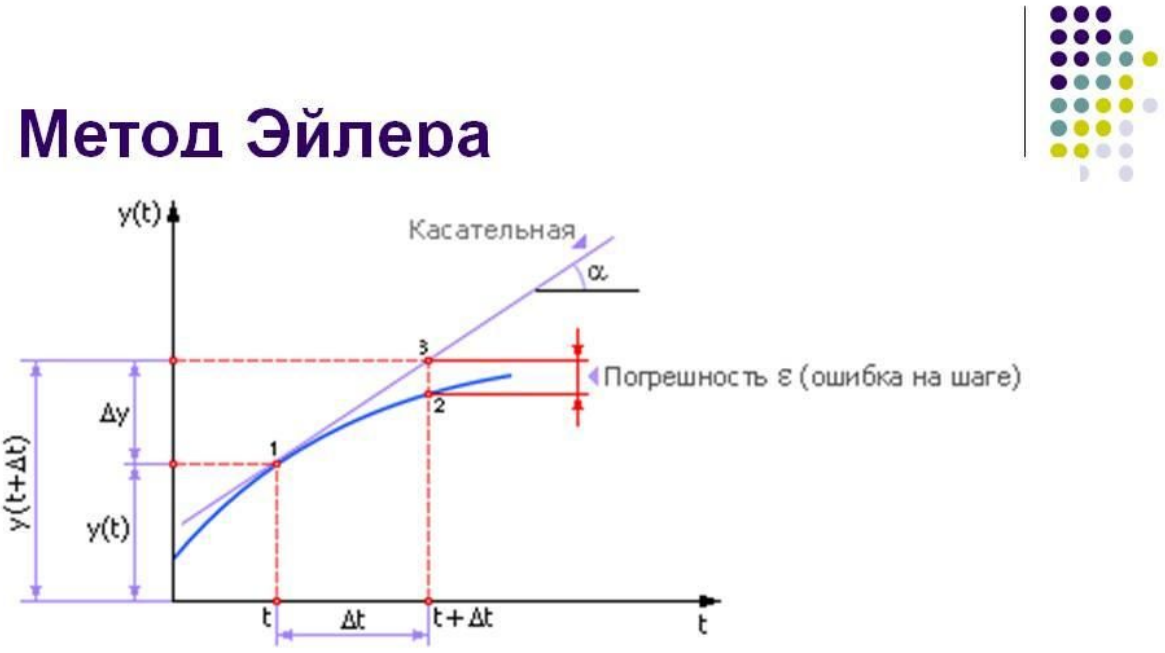
$$\vec{a} \times \vec{b} = \begin{vmatrix} a_y & a_z \\ b_y & b_z \end{vmatrix} \vec{i} - \begin{vmatrix} a_x & a_z \\ b_x & b_z \end{vmatrix} \vec{j} + \begin{vmatrix} a_x & a_y \\ b_x & b_y \end{vmatrix} \vec{k}$$

Формулы для пути, пройденного телом за малый промежуток времени и изменения скорости тела за этот же промежуток (формулы 3 и 4).

$$d\vec{S} = \vec{V} * dt$$

$$d\vec{V} = \frac{\vec{F}}{m} * dt$$

3.2 Математические методы



$$y(t+\Delta t) \approx y(t) + \Delta y = y(t) + \mathrm{tg}(\alpha) \cdot \Delta t = y(t) + \frac{\Delta y}{\Delta t} \cdot \Delta t = y(t) + f(x, y, t) \cdot \Delta t$$

Если имеется дифференциальное уравнение $\frac{df(x)}{dx} = g(x)$, где g(x) известна, а также известно значение f(x) хотя бы в одной точке, то рекурсивно можно найти приближенное значение f(x) в любой точке области определения. Находится это значение путём приближения f(x) в малой окрестности некоторой точки к участку прямой и пересчета коэффициента наклона этой прямой при сдвиге на dx. Более подробное описание этого метода можно найти в интернете, но для понимания алгоритма решения этой информации достаточно.

4.Анализ используемой структуры данных

4.1 Входные данные

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

4.2 Промежуточные данные

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

4.3 Выходные данные

- Анимация движения частиц с возможностью просмотра их текущих параметров.

5.Метод решения

5.1 Решение задачи

5.1.1 *Выбор ПО*

Для решения данной задачи был выбран игровой движок Unity3D.

Игровые движки предназначены для написания игр, а моделирование движения частиц очень похоже на игру. Следовательно, выбор состоял из сред разработки игр, а Unity-одна из наиболее простых для понимания сред разработки, в которой игровая физика реализована наиболее близко к реальной, что не свойственно для многих движков. Например, в Unigine Engine метод, добавляющий скорость, увеличивал её лишь до какого-то значения (для разработки игр это удобно, так как при нажатии клавиши W игровой персонаж должен плавно разогнаться до некоторой скорости, а не увеличивать её до бесконечности), что категорически не подходит для моделирования.
также 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-множество реализованных математических классов. Один из этих классов - класс векторов. Реализованы классы Vector2 и Vector3 (двумерный и трехмерный вектор соответственно). Нет необходимости заводить новый класс для вектора или использовать вместо него набор переменных. Подобных примеров на самом деле гораздо больше.

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.Комментированный листинг

```
using System.Collections.Generic;
using UnityEngine;
using System.Collections;

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
        //однородное поле с единичной магнитной индукцией и напряженностью



        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 координаты.
                vel.x=(fx/mass)*dt;
                vel.y=(fy/mass)*dt;
                vel.z=(fz/mass)*dt;
                rb.velocity = rb.velocity + vel;
        }
}
```

7.Пример работы программы

Здесь приведён пример работы программы. На рис. 1-4 находятся изображения частиц в соответствующий момент времени t от начала движения. В данном случае обе частицы заряжены положительно, магнитное поле однородно и направлено от наблюдателя, электрическое поле однородно и направлено вниз. Заряды частиц подобраны так, чтобы вклад отталкивания двух шаров в их скорость был значительно меньше вклада поля. Из теоретических соображений заряды будут двигаться по циклоиде с небольшой добавкой из-за их отталкивания. Как видно из рисунков, качественно результат моделирования совпал с теоретическим.

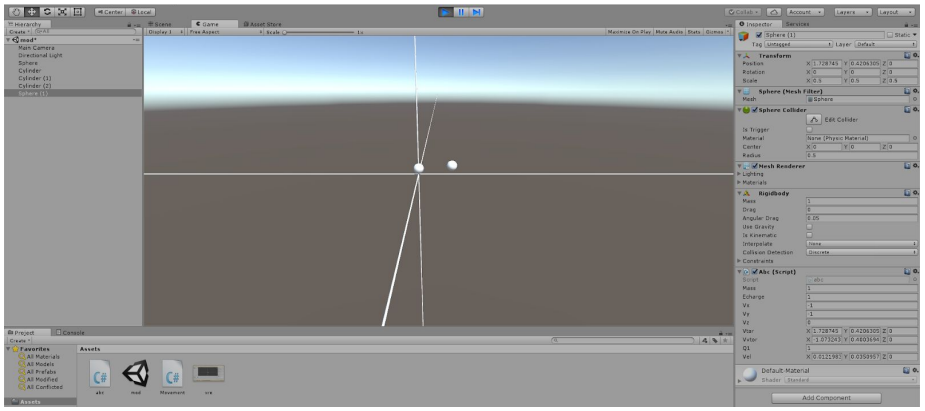


Рис. 1 t<1c

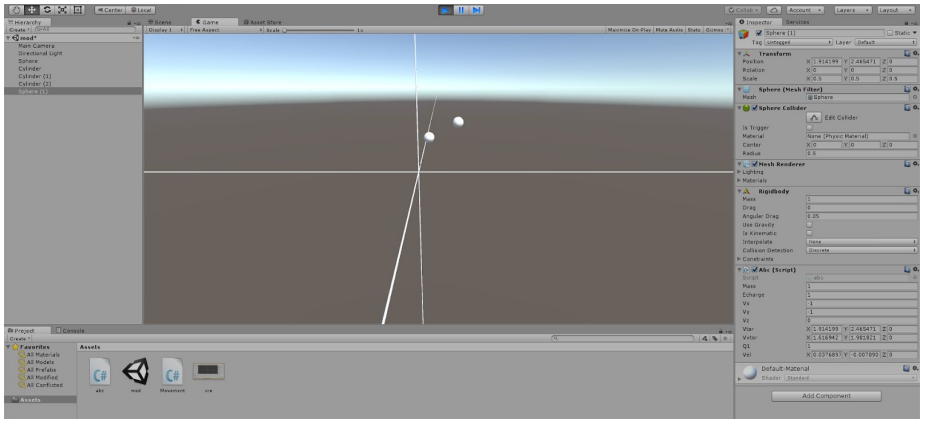


Рис. 2 t~1c

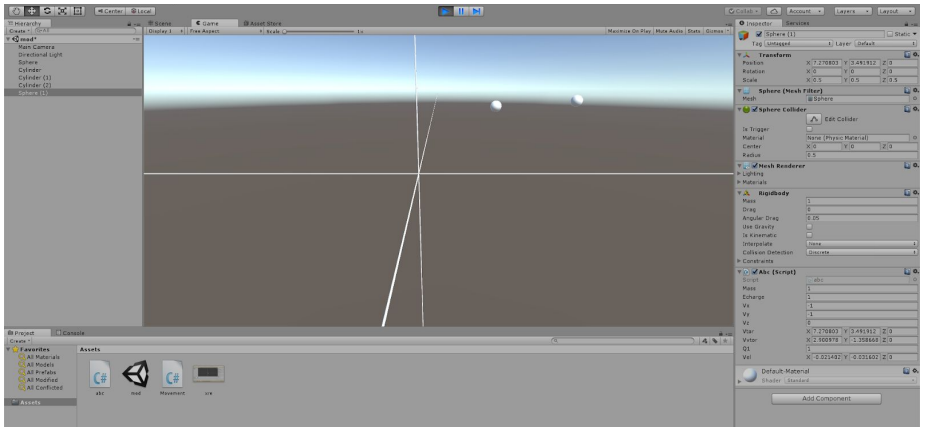


Рис. 3 t~2c

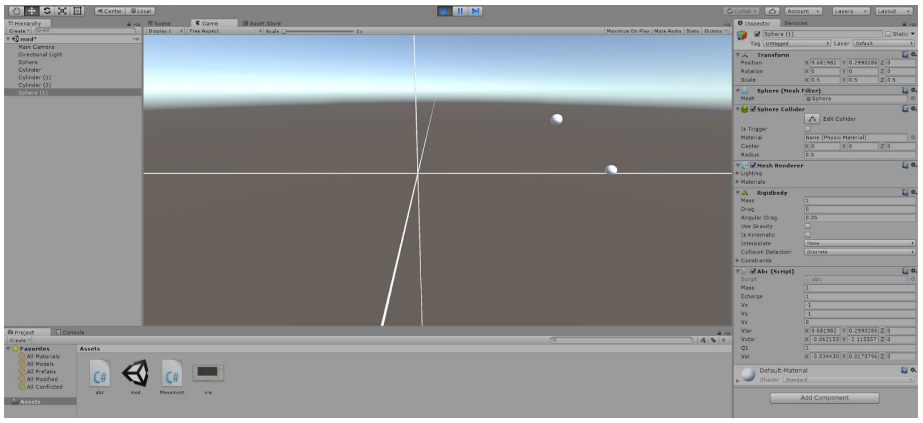


Рис. 4 t~3c

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. Также, как известно, в памяти не может храниться бесконечное количество цифр, а, следовательно, точность ответа может увеличиться от увеличения числа шагов меньше, чем она уменьшится из-за того, что мы большее количество раз округлим обыкновенную дробь до десятичной с конкретным количеством занков после запятой. Также было выдвинуто справедливое замечание об устойчивости системы дифференциальных уравнений, используемой для решения поставленной задачи. Доказательство её устойчивости- сложное математическое задание. К тому же, как было сказано выше, экспериментально проверено, что в большинстве случаев полученная траектория всё же близка к теоретически расчитанной (моделирование проводилось для таких полей, для которых несложно получить аналитическое решение).