Федеральное государственное автономное образовательное учреждение

высшего образования

«Санкт-Петербургский государственный политехнический университет Петра Великого»

Институт прикладкой математики и механики

Кафедра «Теоретическая механика»

**Курсовой проект**

по дисциплине «Информатика»

**«Моделирование и визуализация движения космических тел»**

|  |
| --- |
| Зачтено «\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2016 г. с оценкой \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
| Преподаватель |  | А.Ю.Панченко |
|  |  |  |
| Студент группы 13604/1 |  | А.В. Давыдова |

Санкт-Петербург

2015 г.

Оглавление

[Введение 2](#_Toc453864991)

[Возможности JavaScript 3](#_Toc453864992)

[Обзор литературы 4](#_Toc453864993)

[Описание методов, используемых в программе 5](#_Toc453864994)

[Результаты 14](#_Toc453864995)

[Возможности развития программы 15](#_Toc453864996)

[Список литературы 16](#_Toc453864997)

# Введение

Данная программа была написана для расчета и визуализации движения тел в космосе, под действием силы гравитационного взаимодействия между ними. В частности, одной из целей создания программы было прогнозирование поведения системы планета – спутник при пролете инородного тела вблизи неё. Инородное тело может стать спутником планеты, может захватить спутник и улететь от нее по гиперболической орбите, может измениться траектория спутника, также могут быть и другие варианты конфигурации получившейся системы. Это полностью зависит от начальных параметров системы, таких как скорости, массы, положения тел в пространстве. Возможности программы позволяют моделировать поведение трех взаимодействующих объектов, массы которых могут быть от 0 до 1040 килограммов (верхний предел больше, чем массы звезд сверхгигантов), которые движутся с нерелятивистскими скоростями (случай такого взаимодействия в программе пока не рассмотрен).

Программа написана на HTML и JavaScript. Её запуск производится через интернет браузер (например, Mozilla Firefox или Google Chrome) и не требует установки дополнительных программ.

# Возможности JavaScript

JavaScript является объектно-ориентированным языком, но используемое в языке [прототипирование](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D1%82%D0%BE%D1%82%D0%B8%D0%BF%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5%22%20%5Co%20%22%D0%9F%D1%80%D0%BE%D1%82%D0%BE%D1%82%D0%B8%D0%BF%D0%BD%D0%BE%D0%B5%20%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5) обуславливает отличия в работе с объектами по сравнению с традиционными класс-ориентированными языками. Кроме того, JavaScript имеет ряд свойств, присущих функциональным языкам — функции как [объекты первого класса](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82_%D0%BF%D0%B5%D1%80%D0%B2%D0%BE%D0%B3%D0%BE_%D0%BA%D0%BB%D0%B0%D1%81%D1%81%D0%B0), объекты как списки, [карринг](https://ru.wikipedia.org/wiki/%D0%9A%D0%B0%D1%80%D1%80%D0%B8%D0%BD%D0%B3%22%20%5Co%20%22%D0%9A%D0%B0%D1%80%D1%80%D0%B8%D0%BD%D0%B3), анонимные функции, замыкания — что придаёт языку дополнительную гибкость.

Несмотря на схожий с Си синтаксис, JavaScript по сравнению с языком [Си](https://ru.wikipedia.org/wiki/%D0%A1%D0%B8_%28%D1%8F%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F%29) имеет коренные отличия:

* [объекты](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82_%28%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5%29), с возможностью [интроспекции](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D1%80%D0%BE%D1%81%D0%BF%D0%B5%D0%BA%D1%86%D0%B8%D1%8F_%28%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5%29);
* функции как [объекты первого класса](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82_%D0%BF%D0%B5%D1%80%D0%B2%D0%BE%D0%B3%D0%BE_%D0%BA%D0%BB%D0%B0%D1%81%D1%81%D0%B0);
* автоматическое [приведение типов](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%B8%D0%B2%D0%B5%D0%B4%D0%B5%D0%BD%D0%B8%D0%B5_%D1%82%D0%B8%D0%BF%D0%BE%D0%B2);
* автоматическая [сборка мусора](https://ru.wikipedia.org/wiki/%D0%A1%D0%B1%D0%BE%D1%80%D0%BA%D0%B0_%D0%BC%D1%83%D1%81%D0%BE%D1%80%D0%B0_%28%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5%29);
* [анонимные функции](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%BE%D0%BD%D0%B8%D0%BC%D0%BD%D1%8B%D0%B5_%D1%84%D1%83%D0%BD%D0%BA%D1%86%D0%B8%D0%B8).

В языке отсутствуют такие полезные вещи, как:

* [модульная система](https://ru.wikipedia.org/wiki/%D0%9C%D0%BE%D0%B4%D1%83%D0%BB%D1%8C_%28%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5%29): JavaScript не предоставляет возможности управлять зависимостями и изоляцией областей видимости;
* [стандартная библиотека](https://ru.wikipedia.org/wiki/%D0%A1%D1%82%D0%B0%D0%BD%D0%B4%D0%B0%D1%80%D1%82%D0%BD%D0%B0%D1%8F_%D0%B1%D0%B8%D0%B1%D0%BB%D0%B8%D0%BE%D1%82%D0%B5%D0%BA%D0%B0): в частности, отсутствует [интерфейс программирования приложений](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D1%80%D1%84%D0%B5%D0%B9%D1%81_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F_%D0%BF%D1%80%D0%B8%D0%BB%D0%BE%D0%B6%D0%B5%D0%BD%D0%B8%D0%B9) по работе с файловой системой, управлению потоками ввода-вывода, базовых типов для [бинарных данных](https://ru.wikipedia.org/wiki/%D0%94%D0%B2%D0%BE%D0%B8%D1%87%D0%BD%D1%8B%D0%B9_%D1%84%D0%B0%D0%B9%D0%BB);
* стандартные [интерфейсы](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D1%80%D1%84%D0%B5%D0%B9%D1%81) к [веб-серверам](https://ru.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D1%81%D0%B5%D1%80%D0%B2%D0%B5%D1%80%22%20%5Co%20%22%D0%92%D0%B5%D0%B1-%D1%81%D0%B5%D1%80%D0%B2%D0%B5%D1%80) и [базам данных](https://ru.wikipedia.org/wiki/%D0%91%D0%B0%D0%B7%D0%B0_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85);
* [система управления пакетами](https://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0_%D1%83%D0%BF%D1%80%D0%B0%D0%B2%D0%BB%D0%B5%D0%BD%D0%B8%D1%8F_%D0%BF%D0%B0%D0%BA%D0%B5%D1%82%D0%B0%D0%BC%D0%B8), которая бы отслеживала зависимости и автоматически устанавливала их.

# Обзор литературы

При написании работы использовались статьи на различных интернет ресурсах, посвященные программированию на языках HTML и JavaScript, такие как www.w3schools.com и learn.javascript.ru. Основные принципы программирования на этих языках были взяты из статьи Д.В. Цветкова «Обучение JavaScript». В программе используется объектный подход, описанный в книге «Объектно-ориентированное программирование в С++» Роберта Лафоре.

# Описание методов, используемых в программе

Код программы содержится в нескольких файлах. Первый – это файл HTML кода страницы, который содержит весь интерфейс, разметку страницы, которую видит пользователь. Кнопки, с помощью которых запускается программа, также описаны в HTML файле. Ниже приведен код этой страницы.

<!DOCTYPE html>

<html>

<head>

<title>Planets</title>

<meta charset="UTF-8" />

<script src="alg\_lib.js"></script>

<script src="Vector.js"></script>

<script src="PlanetsJS.js"></script>

</head>

<body onload ="main\_planets();">

<canvas id="Planets" width="800" height="800" style="border:1px solid #000000;"></canvas> <br>

<input id="button\_calc" type="button" value="Сосчитать">

<input id="button\_pause" type="button" value="Пауза" >

масса тела 1, кг<input type ="number" id="PlanetM1" min=0 max=10 style="width: 50px">

\*10^<input type ="number" id="expm1" min=1 max=100 style="width: 50px"><br>

скорость вдоль оси 0x <input type ="number" id="PlanetV1\_X" min=0 max=2000 style="width: 50px"> км/с<br>

скорость вдоль оси 0y <input type ="number" id="PlanetV1\_Y" min=0 max=2000 style="width: 50px"> км/с<br>

Положение по Х <input type ="number" id="PlanetR1\_X" min=0 max=30 style="width: 50px"> а.е.<br>

Положение по Y <input type ="number" id="PlanetR1\_Y" min=0 max=30 style="width: 50px"> а.е.<br>

<br><br>

масса тела 2, кг<input type ="number" id="PlanetM2" min=0 max=10 style="width: 50px">

\*10^<input type ="number" id="expm2" min=1 max=100 style="width: 50px"><br>

скорость вдоль оси 0x <input type ="number" id="PlanetV2\_X" min=0 max=2000 style="width: 50px"> км/с<br>

скорость вдоль оси 0y <input type ="number" id="PlanetV2\_Y" min=0 max=2000 style="width: 50px"> км/с<br>

Положение по Х <input type ="number" id="PlanetR2\_X" min=0 max=30 style="width: 50px"> а.е.<br>

Положение по Y <input type ="number" id="PlanetR2\_Y" min=0 max=30 style="width: 50px"> а.е.<br>

<br><br>

масса тела 3, кг<input type ="number" id="PlanetM3" min=0 max=10 style="width: 50px">

\*10^<input type ="number" id="expm3" min=1 max=100 style="width: 50px"><br>

скорость вдоль оси 0x<input type ="number" id="PlanetV3\_X" min=0 max=2000 style="width: 50px"> км/с<br>

скорость вдоль оси 0y<input type ="number" id="PlanetV3\_Y" min=0 max=2000 style="width: 50px"> км/с<br>

Положение по Х <input type ="number" id="PlanetR3\_X" min=0 max=30 style="width: 50px"> а.е.<br>

Положение по Y <input type ="number" id="PlanetR3\_Y" min=0 max=30 style="width: 50px"> а.е.<br>

<br><br>

время <input type = "number" id = "time" min=1, max=10 style ="width: 50px"> <br>

<span id="span\_name">Здесь будет находиться выводимый текст</span>

</body>

</html>

Во втором файле содержится JavaScript код, выполнение которого начинается сразу после загрузки страницы. Он содержит основную последовательность команд, которая записана с помощью функций. Код страницы приведен ниже:

function main\_planets()

{

 var Planet1 = { //создание объекта - планеты

 m:2,

 expm:30,

 r:new Vector(0.0, 0.0), // расстояние в a.e.= 149 597 870 000 м

 v:new Vector(0.0, 0.0) // корость в км/с

 };

 PlanetM1.value = Planet1.m; // присвоение переменным с HTML страницы

 expm1.value = Planet1.expm; // значений полей объекта

 PV1x=Planet1.v.x\*149597870;

 PV1y=Planet1.v.y\*149597870;

 PlanetV1\_X.value = PV1x;

 PlanetV1\_Y.value = PV1y;

 PlanetR1\_X.value = Planet1.r.x;

 PlanetR1\_Y.value = Planet1.r.y;

 var Planet2 = {

 m:5.97,

 expm:24,

 r:new Vector(1.0, 0.0),

 v:new Vector(0.0,0.00000020) // корость в км/с

 };

 PlanetM2.value = Planet2.m;

 expm2.value = Planet2.expm;

 PlanetV2\_X.value = Planet2.v.x\*149597870;

 PlanetV2\_Y.value = Planet2.v.y\*149597870;

 PlanetR2\_X.value = Planet2.r.x;

 PlanetR2\_Y.value = Planet2.r.y;

 var Planet3 = {

 m:7.3,

 expm:22,

 r:new Vector(1.00256, 0.0), // расстояние в a.e.= 149 597 870 000 m

 v:new Vector(-0.0000000,0.00000020668) // корость в км/с

 };

 PlanetM3.value = Planet3.m;

 expm3.value = Planet3.expm;

 PlanetV3\_X.value = Planet3.v.x\*149597870;

 PlanetV3\_Y.value = Planet3.v.y\*149597870;

 PlanetR3\_X.value = Planet3.r.x;

 PlanetR3\_Y.value = Planet3.r.y;

 t=1;

 ti=100000;

 time.value = t

 console.log(Planet1.r.lenght())

 PlanetM1.oninput= function() {

 Planet1.m = parseFloat(PlanetM1.value);

 //span\_name.innerHTML=Planet1.m;

 }

 expm1.oninput= function()

 {

 Planet1.expm=parseFloat(expm1.value); //функции, обновляющие соответствующие переменные, при вводе пользователем новых значений

 }

 PlanetV1\_X.oninput = function()

 {

 Planet1.v.x=parseFloat(PlanetV1\_X.value);

 Planet1.v.x = Planet1.v.x/149597870;

 }

 PlanetV1\_Y.oninput = function()

 {

 Planet1.v.y=parseFloat(PlanetV1\_Y.value);

 Planet1.v.y = Planet1.v.y/149597870;

 }

 PlanetR1\_X.oninput = function()

 {

 Planet1.r.x=parseFloat(PlanetR1\_X.value);

 }

 PlanetR1\_Y.oninput = function()

 {

 Planet1.r.y=parseFloat(PlanetR1\_Y.value);

 }

 PlanetM2.oninput= function()

 {

 Planet2.m = parseFloat(PlanetM2.value);

 }

 expm2.oninput= function()

 {

 Planet2.expm=parseFloat(expm2.value);

 }

 PlanetV2\_X.oninput = function()

 {

 Planet2.v.x=parseFloat(PlanetV2\_X.value)/149597870;

 }

 PlanetV2\_Y.oninput = function()

 {

 Planet2.v.y=parseFloat(PlanetV2\_Y.value)/149597870;

 }

 PlanetR2\_X.oninput = function()

 {

 Planet2.r.x=parseFloat(PlanetR2\_X.value);

 }

 PlanetR2\_Y.oninput = function(){

 Planet2.r.y=parseFloat(PlanetR2\_Y.value);

 }

 PlanetM3.oninput= function() {

 Planet3.m = parseFloat(PlanetM3.value);

 }

 expm3.oninput= function()

 {

 Planet3.expm=parseFloat(expm3.value);

 }

 PlanetV3\_X.oninput = function()

 {

 Planet3.v.x=parseFloat(PlanetV3\_X.value)/149597870;

 }

 PlanetV3\_Y.oninput = function()

 {

 Planet3.v.y=parseFloat(PlanetV3\_Y.value)/149597870;

 }

 PlanetR3\_X.oninput = function()

 {

 Planet3.r.x=parseFloat(PlanetR3\_X.value);

 }

 PlanetR3\_Y.oninput = function(){

 Planet3.r.y=parseFloat(PlanetR3\_Y.value);

 }

 time.oninput = function()

 {

 t= parseFloat(time.value);

 }

 var timerId;

 button\_calc.onclick = function () // кнопка "сосчитать"

 { Planet1.m = parseFloat(PlanetM1.value);

 Planet1.expm=parseFloat(expm1.value);//обновление всех переменных в памяти компьютера в соответствии со значениями на странице браузера

 Planet1.v.x=parseFloat(PlanetV1\_X.value);

 Planet1.v.x = Planet1.v.x/149597870;

 Planet1.v.y=parseFloat(PlanetV1\_Y.value);

 Planet1.v.y = Planet1.v.y/149597870;

 Planet1.r.x=parseFloat(PlanetR1\_X.value);

 Planet1.r.y=parseFloat(PlanetR1\_Y.value);

 Planet2.m = parseFloat(PlanetM2.value);

 Planet2.expm=parseFloat(expm2.value);

 Planet2.v.x=parseFloat(PlanetV2\_X.value)/149597870;

 Planet2.v.y=parseFloat(PlanetV2\_Y.value)/149597870;

 Planet2.r.x=parseFloat(PlanetR2\_X.value);

 Planet2.r.y=parseFloat(PlanetR2\_Y.value);

 Planet3.m = parseFloat(PlanetM3.value);

 Planet3.expm=parseFloat(expm3.value);

 Planet3.v.x=parseFloat(PlanetV3\_X.value)/149597870;

 Planet3.v.y=parseFloat(PlanetV3\_Y.value)/149597870;

 Planet3.r.x=parseFloat(PlanetR3\_X.value);

 Planet3.r.y=parseFloat(PlanetR3\_Y.value);

 t= parseFloat(time.value);

 step();

 //span\_name.innerHTML=Planet1.m;

 //setInterval(step, 1000 / 60);

 timerId = setInterval(

 step, 1000\*t/ 60);

 }

 button\_pause.onclick = function() //кнопка паузы

 {

 clearInterval(timerId);

 }

 function step() { // функция, выполняющаяся 60 раз в секунду после нажатия кнопки "сосчитать"

 physics();

 draw();

 }

 function physics() { // функция, считающая координаты тел в следующий момент времени

 planet1a=ALG.moment\_accel(Planet2.r,Planet1.r,Planet3.r,Planet2.m,Planet2.expm,Planet3.m,Planet3.expm);

 planet2a=ALG.moment\_accel(Planet1.r,Planet2.r,Planet3.r,Planet1.m,Planet1.expm,Planet3.m,Planet3.expm);

 planet3a=ALG.moment\_accel(Planet1.r,Planet3.r,Planet2.r,Planet1.m,Planet1.expm,Planet2.m,Planet2.expm);

 Planet1.v= ALG.momentV(Planet1.v,planet1a,ti);

 Planet1.r=ALG.momentR(Planet1.r,Planet1.v,ti);

 Planet2.v= ALG.momentV(Planet2.v,planet2a,ti);

 Planet2.r=ALG.momentR(Planet2.r,Planet2.v,ti);

 Planet3.v= ALG.momentV(Planet3.v,planet3a,ti);

 Planet3.r=ALG.momentR(Planet3.r,Planet3.v,ti);

 }

var ctx = Planets.getContext("2d");

 ctx.strokeStyle = "#f00";

 ctx.fillStyle = "#00f";

 ctx.lineWidth = 1;

 var w = Planets.width;

 var h = Planets.height;

 function draw() { //функция, рисующая планеты

 ctx.clearRect(0, 0, w, h); // отчищение холста после предыдущего кадра

 ctx.beginPath();

 ctx.fillStyle = "#00f";

 ctx.arc(Planet1.r.x\*200+w/2,h/2-Planet1.r.y\*200, 15, 0, 2\*Math.PI)

 ctx.fill();

 ctx.beginPath();

 ctx.fillStyle = "#0f0";

 ctx.arc(Planet2.r.x\*200+w/2,h/2-Planet2.r.y\*200 , 5, 0, 2\*Math.PI)

 ctx.fill();

 ctx.beginPath();

 ctx.fillStyle = "#f00";

 ctx.arc(Planet3.r.x\*200+w/2,h/2-Planet3.r.y\*200 , 3, 0, 2\*Math.PI)

 ctx.fill();

 }

}

Эти функции, в свою очередь, описаны в другом JavaScript файле – библиотеке функций, написанной специально для этой программы. Это было сделано для того, чтобы не «загромождать» основной код описанием мелких действий.

ALG = {}

ALG.sum = function(a, b) {

 return a + b;

}

ALG.VectNumbIncr = function(a,n){

 c = new Vector;

 c.x=a.x\*n;

 c.y=a.y\*n;

 return c;

}

ALG.VectLenth = function(a){

 c=a.x\*a.x+a.y\*a.y;

 return Math.sqrt(c);

}

ALG.Vectsum = function(a,b) {

 c = new Vector;

 c.x=a.x+b.x;

 c.y=a.y+b.y;

 return c;

}

ALG.VectScalIncr = function (a,b){

 return a.x\*b.x+a.y\*b.y;

}

ALG.VectDecreace = function(a,b){

 c = new Vector;

 c.x = a.x-b.x;

 c.y = a.y-b.y;

 return c;

}

ALG.moment\_accel=function(r1,r2,r3,m1,expm1,m3,expm3){ //ускорение тела r2 m2

 r21=ALG.VectDecreace(r1,r2);

 eR1=ALG.VectNumbIncr(r21,1/ALG.VectLenth(r21));

 r23=ALG.VectDecreace(r3,r2);

 eR3=ALG.VectNumbIncr(r23,1/ALG.VectLenth(r23));

 absA21 = 2.98043 \*m1/ALG.VectScalIncr(r21,r21);

 absA23 = 2.98043 \*m3/ALG.VectScalIncr(r23,r23); //0.044614

 expm1=expm1-33;

 expm3=expm3-33;

 a21=ALG.VectNumbIncr(eR1,absA21);

 a23=ALG.VectNumbIncr(eR3,absA23);

 exp1 = Math.pow(10, expm1);

 exp3 = Math.pow(10, expm3);

 a21=ALG.VectNumbIncr(a21,exp1);

 a23=ALG.VectNumbIncr(a23,exp3);

 a2=ALG.Vectsum(a21,a23);

 a2=ALG.VectNumbIncr(a2,1/149597870000);

 return a2;

}

ALG.momentV = function (v,a,ti){

 dv=ALG.VectNumbIncr(a,ti);

 v1=ALG.Vectsum(dv,v);

 return v1;

}

ALG.momentR = function(r,v,ti){

 dr=ALG.VectNumbIncr(v,ti);

 r1=ALG.Vectsum(dr,r);

 return r1;

}

Также используется отдельный файл, содержащий описание действий с векторными величинами. Простые операции как сложение векторов, нормирование вектора, скалярное произведение и многие другие очень часто используются при решении данной проблемы. Использование векторных функций, описанных в библиотеке, вместо использования в программе действий с координатами векторов сильно сократило размер программы и облегчило понимание кода, поскольку большинство величин, с которыми мы сталкиваемся в этой задаче – векторные.

В вопросе систематизации переменных был применен подход объектно-ориентированного программирования: планеты создавались как объекты, обладающие различными свойствами: массой, скоростью, положением в пространстве.

# Результаты

Результатом работы является программа, наглядно демонстрирующая поведение системы, заданной пользователем конфигурации. 

Рисунок . Результат работы программы



Рисунок . Вид панели управления программы

Пользователь может ввести свои данные, и после нажатия кнопки «сосчитать» запускается анимация, которую можно остановить, нажав на кнопку «Пауза». Чтобы ввести новые значения и посмотреть, каким будет поведение системы без перезагрузки страницы, нужно нажать кнопку «Пауза», ввести значения, нажать кнопку «Сосчитать» и наблюдать результат. Параметр «время» также можно менять – от этого будет меняться скорость воспроизведения анимации.

# Возможности развития программы

В данной модели рассматривается пока только плоское движение. В плане развития программы разработка трехмерной визуализации движения. С точки зрения физики программа при этом не усложнится, увеличится часть кода, отвечающая за визуализацию.

Вторым важным моментом, требующем доработки является автоматическая установка масштаба изображения. Это нужно для того, чтобы пользователь мог видеть всю картину происходящего, при чем достаточно крупно.

Третьим пунктом возможного развития продукта является создание функции, анализирующей систему с точки зрения связей. То есть пользователь помимо анимации будет получать еще и информацию, какие тела находятся на устойчивых орбитах, а какие по прошествии большого времени будут иметь практически прямую траекторию и не вернутся в систему.

# Список литературы

1. www.w3schools.com
2. learn.javascript.ru.
3. http://tm.spbstu.ru/\_JavaScript
4. http://tm.spbstu.ru/Различные\_цветовые\_модели\_и\_их\_использование
5. Роберт Лафоре. Объектно-ориентированное программирование в С++. СПб.: Питер, 2004. -923 с
6. <https://ru.wikipedia.org/wiki/JavaScript>
7. <http://habrahabr.ru/>
8. Бер Бибо, Иегуда Кац. jQuery. Подробное руководство по продвинутому JavaScript, 2-е издание. – Пер. с англ. – СПб.: Символ-Плюс, 2011. – 624 с.
9. <http://ru.html.net/>
10. http://professorweb.ru/my/javascript/js\_theory/level1/javascript\_index.php