Санкт-Петербургский государственный политехнический университет

Институт Прикладной математики и механики  
Кафедра Прикладной математики

В.С. Погодина

МОДЕЛИРОВАНИЕ ДВИЖЕНИЯ СВЯЗАННЫХ СПУТНИКОВ В ГРАВИТАЦИОННОМ ПОЛЕ ЗЕМЛИ НА JAVASCRIPT

Курсовой проект

Направление подготовки бакалавров: 010800 Механика и математическое моделирование

Профиль ООП: 010800.62.01 Механика деформируемого твердого тела

Группа 23604/1

Руководитель проекта: Панченко А.Ю.

Допущена к защите:

«\_\_» 20\_\_ г.

Санкт-Петербург

2014

# **Оглавление**

[***Оглавление*** 2](#_Toc388226015)

[***Введение*** 3](#_Toc388226016)

[***Глава 1. Значение и выбор фреймворка*** 4](#_Toc388226017)

[1.1 Что такое библиотеки JavaScript 4](#_Toc388226018)

[1.2 Выбор JS фреймворка 5](#_Toc388226019)

[***Глава 2. Создание приложения*** 6](#_Toc388226021)

[2.1 Начало работы с проектом и JavaScript 6](#_Toc388226022)

[2.2 Написание кода 7](#_Toc388226023)

[2.3 Итог работы](#_Toc388226025) 12

[***Заключение***](#_Toc388226026) 13

[***Литература***](#_Toc388226027) 14

# **Введение**

Целью данной курсовой работы является создание приложения с использованием одного из существующий JS-фреймворков. В процессе выполнения курсовой работы необходимо решить следующие задачи: .

* Выбрать одну из JavaScript библиотек для создания программы
* Создать приложение с использованием одного из фреймворков, позволяющую визуально представить влияние двух сил на движение спутников вокруг Земли

В первой главе приведено определение JavaScript библиотек, немного истории создания. Выбирается один из этих фреймворков для создания клиентского приложения.

Во второй главе описывается то, как создается это приложение с примерами кодов страниц.

# **Глава 1. Значение и выбор фреймворка**

## 1.1 Что такое библиотеки JavaScript

Библиотека JavaScript — сборник классов и/или функций на языке JavaScript.

Язык JavaScript, изначально разработанный Netscape (а затем развиваемый Mozilla), долгое время использовался в сети на многих сайтах, но широкую популярность получил с приходом Веб 2.0 — периода развития компьютерных систем, в котором JavaScript совместно с различными диалектами XML стал активно использоваться в разработке пользовательских интерфейсов как веб-приложений, так и настольных приложений. JavaScript в связке с CSS используется для создания динамических сайтов, более доступных, чем основанные на Flash альтернативы.

С увеличением популярности JavaScript, простота создания динамических элементов пользовательского интерфейса стала играть ключевую роль в веб-разработке. Этим обусловлен лавинообразный характер появления различных библиотек JavaScript, таких как Ext и Dojo. С другой стороны, одним из последствий войны браузеров стала разница в реализации объектной модели документа и это обусловило необходимость затрачивать дополнительные усилия для реализации корректной работы различных браузеров. Данное обстоятельство обусловило появление библиотек JavaScript, предоставляющих кроссбраузерный интерфейс к методам DOM, таких как Prototype, script.aculo.us или jQuery.

Удобство использования библиотек JavaScript привело к тому, что Microsoft, Yahoo! и другие крупные ИТ-компании разрабатывают свои собственные основанные на JavaScript библиотеки элементов пользовательского интерфейса, встраиваемые в веб-приложения, разрабатываемые этими компаниями.

Практически все библиотеки JavaScript выпускаются под лицензиями копицентр и копилефт, чтобы обеспечить свободное от лицензионных отчислений разработку, использование и модификацию.

Более того, некоторые библиотеки JavaScript позволяют упростить взаимодействие JavaScript с другими языками, такими как CSS, PHP, Ruby и Java. Это позволяет упростить запуск приложений JavaScript с приложениями, написанными на других языках программирования.

## 1.2 Выбор Js фреймворка

**jQuery** — библиотека JavaScript, фокусирующаяся на взаимодействии JavaScript и HTML. Библиотека jQuery помогает легко получать доступ к любому элементу DOM, обращаться к атрибутам и содержимому элементов DOM, манипулировать ими. Также библиотека jQuery предоставляет удобный API для работы с AJAX.

Основные возможности **jQuery**:

* Движок кроссбраузерных CSS-селекторов ;
* Переход по дереву DOM;
* События;
* Визуальные эффекты;
* AJAX-дополнения;
* JavaScript-плагины.

**jQuery** - очень простой, быстрый, широко распространенный. Выгоден для малых и средних проектов. Хорошая документация и много сторонних примеров в сети. Последнее и послужило для меня причиной выбрать именно эту библиотеку.

# **Глава 2. Создание приложения**

## 2.1 Начало работы с проектом и JavaScript

Итак, выбрав необходимый нам фреймворк, встала задача: как написать код (тема была уже определена, т.к. требовалась моделирование движения спутников).

Программа должна включать в себя следующие опции:

1. Корректное отображение исследуемого объекта, с плавной анимацией ;
2. График движения грузика
3. Запуск движения системы
4. Возможность отображать траекторию движения центра тяжести

## 2.2 Написание кода

Для начала создадим стартовую страницу index.htm на которую и будем выводить наш фрейм для моделирования, кнопку пуска, и checkbox для выбора отображения траектории центра тяжести системы.

<!DOCTYPE html>

<html>

<head>

<meta charset="utf-8">

<title></title>

<script src="ocanvas-251.js"></script>

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

</head>

<body>

<canvas id="canvasMech" width="800" height="600" style="border:1px solid #000000; float: left;"></canvas>

<input type="button" style="width: 50px; height:50px; margin-left:100px" name="" onclick="app.PUSK();" value="ПУСК"><br>

<input type="checkbox" id="check" style = "margin-left: 100px" value="1"> Траектория

</div>

<script type="text/javascript">var app = new MainMech(document.getElementById('canvasMech'), document.getElementById('check'));</script>

</body>

</html> Для того чтобы создать меню, нам нужна библиотека jQuery. Для этого зайдем на официальный сайт http://jquery.com/ и скачиваем оттуда самую последнюю версию. Скачиваем с http://ocanvas.org/ файл ocanvas.min.js и помещаем в ту же папку. Используем oCanvas для вывода анимации движения спутников.

Основную часть занимает скрипт, отвечающий за моделирование движения системы.

function MainMech(canvas) {

// Предварительные установки

var context = canvas.getContext("2d");

// \*\*\* Задание физических параметров \*\*\*

var Pi = 3.1415926; // число "пи"

var M = 1e15; // Масса Земли

var R = 1.6; // расстояние между Землей и 1 спутником

var l0 = 0.2; //длина троса

var R2 = R + l0; // расстояние между Землей и 2 спутником

var G = 1000000000; //гравитационная постоянная

var m1 = 1e2; //масса 1 спутника

var m2 = 1e2; //масса 2 спутника

var c = 1e27; //жесткость пружины

var T = 2 \* Pi / Math.sqrt(G \* M / R / R / R); // масштаб времени (период колебаний исходной системы)

var Ny = 5; // Число шаров, помещающихся по вертикали в окно (задает размер шара относительно размера окна)

// \*\*\* Задание вычислительных параметров \*\*\*

var fps = 24; // frames per second - число кадров в секунду (качеcтво отображения)

var spf = 2; // steps per frame - число шагов интегрирования между кадрами (скорость расчета)

var dt = 0.01 \* T ; // шаг интегрирования (качество расчета)

var steps = 0; // количество шагов интегрирования

// \*\*\* Выполнение программы \*\*\*

var scale = canvas.height / Ny; // масштабный коэффициент для перехода от расчетных к экранным координатам

var w = canvas.width / scale; // ширина окна в расчетных координатах

var h = canvas.height / scale; // высота окна в расчетных координатах

var x = w / 2; var y = h / 2 + R; // начальное положение 1 спутника

var p = w / 2; var q = h / 2 + R2; // начальное положение 2 спутника

var count = true; // проводить ли расчет системы

this.PUSK = function(){

count = false;

};

// создаем объект, связанный с элементом canvas на html странице

var ocanvas = oCanvas.create({

canvas: "#canvasMech", // canvasMech - id объекта canvas на html странице

fps: fps // сколько кадров в секунду

});

// создаем Землю

var arc = ocanvas.display.arc({

x: w / 2 \* scale,

y: h / 2 \* scale,

radius: 1 / 3 \* scale,

end: 360, // круг 360 градусов

fill: "rgba(0, 0, 255, 1)" // цвет

}).add();

// создаем 1 спутник

var arc1 = ocanvas.display.arc({

x: x \* scale,

y: y \* scale,

radius: 1 / 12 \* scale,

end: 360, // круг 360 градусов

fill: "rgba(0, 255, 0, 1)" // цвет

}).add();

// создаем 2 спутник

var arc2 = ocanvas.display.arc({

x: p \* scale,

y: q \* scale,

radius: 1 / 12 \* scale,

end: 360, // круг 360 градусов

fill: "rgba(0, 255, 0, 1)" // цвет

}).add();

var line = ocanvas.display.line({ //линия соединяющая центры 2-х спутников

start: { x: arc1.x, y: arc1.y },

end: { x: arc2.x, y: arc2.y }}).add();

var vx = -Math.sqrt(G \* M / R); var vy = 0; // скорость 1 спутника

var vx2 = vx \* R2 / R; var vy2 = 0; // скорость 2 спутника

function physics(){ // то, что происходит каждый шаг времени

if (count) return;

for (var s=1; s<spf; s++) {

x1 = (arc1.x - arc.x)/scale; //координаты 1 спутника относительно Земли

y1 = (arc1.y - arc.y)/scale;

x2 = (arc2.x - arc.x)/scale; //координаты 2 спутника относительно Земли

y2 = (arc2.y - arc.y)/scale;

absR = Math.sqrt(x1 \* x1 + y1 \* y1); //расстояние между 1 спутником и Землей

absR2 = Math.sqrt(x2 \* x2 + y2 \* y2); //расстояние между 2 спутником и Землей

l = Math.sqrt((x1 - x2) \* (x1 - x2) + (y1 - y2) \* (y1 - y2)); //расстояние между спутниками

dl = l - l0; //удлинение пружины

if ( dl >= 0 )

F = c \* dl;

else F = 0;

vx += (-G \* M / (absR \* absR \* absR) \* x1 - F / l \* (x1 - x2) / m1) \* dt; //считаем скорости на каждом шаге

vy += (-G \* M / (absR \* absR \* absR) \* y1 - F / l \* (y1 - y2) / m2) \* dt;

arc1.x += vx \* dt \* scale; //считаем координаты накаждом шаге

arc1.y += vy \* dt \* scale;

absR2 = Math.sqrt(x2 \* x2 + y2 \* y2);

vx2 += (-G \* M / (absR2 \* absR2 \* absR2) \* x2 - F / l \* (x2 - x1) / m2) \* dt;

vy2 += (-G \* M / (absR2 \* absR2 \* absR2) \* y2 - F / l \* (y2 - y1) / m2) \* dt;

arc2.x += vx2 \* dt \* scale;

arc2.y += vy2 \* dt \* scale;

line.start.x = arc1.x; //рисуем линию на каждом шаге

line.start.y = arc1.y;

line.end.x = arc2.x;

line.end.y = arc2.y;

if(check.checked)DrawPoint((arc1.x + arc2.x) /2, (arc1.y + arc2.y) /2); //рисуем траекторию

}

}

function DrawPoint(x, y)

{

ocanvas.display.ellipse({

x: x,

y: y,

radius: 1,

fill: "red"

}).add();

}

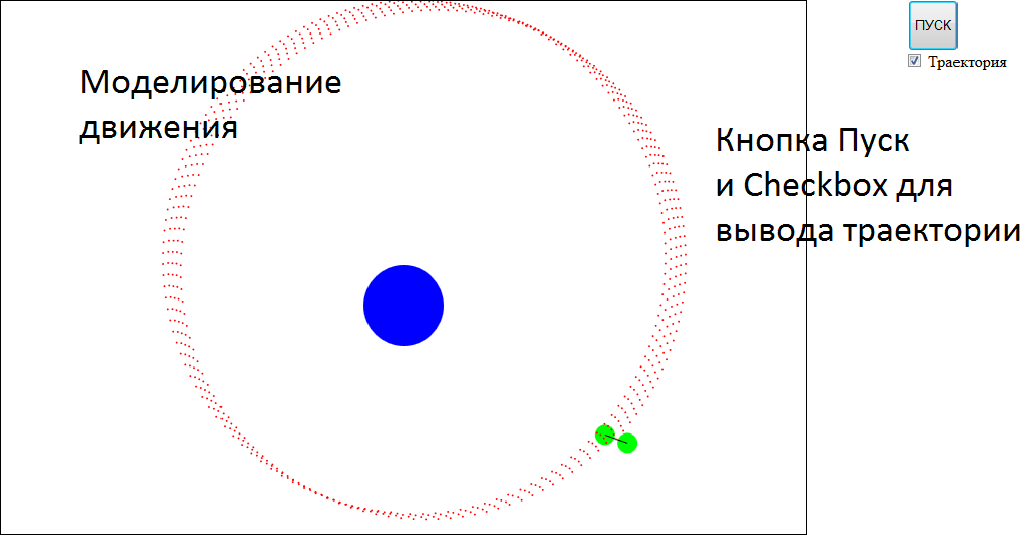
ocanvas.setLoop(physics).start(); // функция, выполняющаяся на каждом шаге

}

**2.3 Итог работы**

Результатом работы стала программа «Моделирование движения связанных спутников в гравитационном поле Земли».

Так выглядит страница после загрузки:



***Заключение***

В ходе написания курсовой работы был прочитан и систематизирован теоретический материал по данной теме, а так же выполнены следующие поставленные во введении задачи:

* Описано что такое JS-фреймворки
* Была разработана программа с использованием фреймворка jQuery
* Программа позволяет визуально представить движение спутников под воздействием гравитационной силы и силы упругости

# ***Литература***

1. http://ru.wikipedia.org/wiki/Библиотека\_JavaScript
2. http://habrahabr.ru/
3. http://jqbook.net.ru/
4. http://ru.html.net/
5. Антон Шевчук. Jquery. Учебник для начинающих.: 2013. – 123с.
6. Бенедетти Р., Крэнли Р. Изучаем работу с jQuery. – СПб.: Питер, 2012. – 528 с.
7. Бер Бибо, Иегуда Кац. jQuery. Подробное руководство по продвинутому JavaScript, 2-е издание. – Пер. с англ. – СПб.: Символ-Плюс, 2011. – 624 с.
8. Каслдайн Э., Шарки К. Изучаем jQuery. 2-е изд. – СПб.: Питер, 2012. – 400 с.
9. Самков Г. А. jQuery. Сборник рецептов. — 2-е изд., перераб. и доп. — СПб.: БХВ-Петербург, 2011. – 416 с.
10. JavaScript: Подробное руководство (Definitive Guide), Давид Финнерман. Спб, 2007г.
11. JavaScript. Библия пользователя, Фленов Иван, Спб, 2005г.
12. Javascript и DHTML, сборник рецептов, Д.Гудман, Спб, 2004г
13. Размещено на Allbest.ru