Санкт-Петербургский государственный политехнический университет

Институт Прикладной математики и механики  
Кафедра Прикладной математики

П. Д. Киселев

МОДЕЛИРОВАНИЕ ДВИЖЕНИЯ БЕСКОНЕЧНОЙ ЦЕПИ ТЕЛ,  
СВЯЗАННЫХ МЕЖДУ СОБОЙ ПРУЖИНАМИ

Курсовой проект

Направление подготовки бакалавров: 010800 Механика и математическое моделирование

Профиль ООП: 010800.62.01 Механика деформируемого твердого тела

Группа 33604/1

Руководитель проекта: Панченко А.Ю.

Допущен 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 колебание n-тел, связанных пружинами, при изменении начального положения одного из тел.

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 n-тел связанных между собой пружинами).

Программа должна включать в себя следующие опции:

1. Корректное отображение исследуемого объекта, с плавной анимацией ;
2. Запуск движения системы (изменение начального положения одного из n тел)
3. Возможность отображать взаимное колебание всех тел (их взаимодействие)

## 2.2 Написание кода

Для начала создадим стартовую страницу index.html на которую и будем выводить наш фрейм для моделирования, также слайдеры для изменения массы тел и жесткости каждой из пружин.

<!DOCTYPE html>

<html>

<head>

<meta charset="utf-8">

<title></title>

<link href="http://fonts.googleapis.com/css?family=Karla:400,700" rel="stylesheet" />

<link href="css/default.css" rel="stylesheet" type="text/css" media="all" />

<link href="css/fonts.css" rel="stylesheet" type="text/css" media="all" />

<link href="css/jquery-ui.css" rel="stylesheet" type="text/css" media="all" />

<script src="http://cdnjs.cloudflare.com/ajax/libs/ocanvas/2.6.0/ocanvas.min.js"></script>

<script type="text/javascript" src="js/jquery.min.js"></script>

<script type="text/javascript" src="js/jquery.flot.axislabels.js"></script>

<script type="text/javascript" src="js/jquery-ui.min.js"></script>

<script type="text/javascript" src="js/flot/jquery.flot.js"></script>

<script type="text/javascript" src="js/ocanvas.min.js"></script>

<script type="text/javascript" src="js/TM.js"></script>

<script type="text/javascript" src="js/MainMech.js"></script>

</head>

<body>

<canvas id="canvasMech" width="700" height="100" style="border:5px solid #00AABB; float: left;"></canvas>

<script type="text/javascript">MainMech(document.getElementById('canvasMech'));</script>

<!--слайдеры-->

<div style="margin-left:20px; margin-top:20px; margin-bottom:20px; float:left;">

<div id="input\_slider\_m" style="width:300px; float:left"></div>

<label for="input\_m" style="margin-left:20px; float:left">m = </label>

<input onKeyUp="onMPress(event);" type="text" id="input\_m" style="margin-left:20px; float:left; border:1; color:#00AABB; font-weight:bold;">

<br><br>

<div id="input\_slider\_C1" style="width:300px; float:left"></div>

<label for="input\_C1" style="margin-left:20px; float:left">C1 = </label>

<input onKeyUp="onC1Press(event);" type="text" id="input\_C1" style="margin-left:20px; float:left; border:1; color:#00AABB; font-weight:bold;">

<br><br>

<div id="input\_slider\_C2" style="width:300px; float:left"></div>

<label for="input\_C2" style="margin-left:20px; float:left">C2 = </label>

<input onKeyUp="onC2Press(event);" type="text" id="input\_C2" style="margin-left:20px; float:left; border:1; color:#00AABB; font-weight:bold;">

<br><br>

<div id="input\_slider\_C3" style="width:300px; float:left"></div>

<label for="input\_C3" style="margin-left:20px; float:left">C3 = </label>

<input onKeyUp="onC3Press(event);" type="text" id="input\_C3" style="margin-left:20px; float:left; border:1; color:#00AABB; font-weight:bold;">

<br><br>

<div id="input\_slider\_B" style="width:300px; float:left"></div>

<label for="input\_B" style="margin-left:20px; float:left">B = </label>

<input onKeyUp="onBPress(event);" type="text" id="input\_B" style="margin-left:20px; float:left; border:1; color:#00AABB; font-weight:bold;">

<br><br>

</div></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 тел.

Основную часть занимает скрипт, отвечающий за моделирование движения системы.

function MainMech(canvas) {

var context = canvas.getContext("2d"); // предварительное создание плоскости  
 // задание физических констант

const Pi = 3.1415926; // число Пи

const m0 = 1; // масса тел

const T0 = 1; // Период колебании

const k0 = 2 \* Pi / T0; // частота колебании

const C0 = m0 \* k0 \* k0; // жесткость пружин

const B0 = 2 \* m0 \* k0; // Вязкость среды

var m = 1 \* m0; // масса тела «масштаб»

var C1 = 0.1 \* C0; // частота колебании 1 пружины

var C2 = 0.1; // частота колебании 2 пружины

var C3 = 0.1; // частота колебании 3 пружины

var B = 0;; // вязкость среды

const fps = 50; // frames per second - число кадров в секунду (качество отображения)

const spf = 20; // steps per frame - число шагов интегрирования между кадрами (скорость расчета)

const dt = 0.001; // шаг интегрирования (качество расчета)

var steps = 0; // количество шагов интегрирования

// Выполнение программы

var TM\_obj = new TM(); // создаем новый объект

this.setM = function(new\_m){m = new\_m \* m0;}; // установка массы

this.setC1 = function(new\_C1){C1 = new\_C1 \* C0;}; // установка частоты для 1 пружины

this.setC2 = function(new\_C2){C2 = new\_C2 \* C0;}; // установка частоты для 2 пружины

this.setC3 = function(new\_C3){C3 = new\_C3 \* C0;}; // установка частоты для 3 пружины

this.setB = function(new\_B){B = new\_B ;}; // установка вязкости среды

TM\_obj.addInputSlider("#input\_slider\_m", "#input\_m", 0.01, 10, 0.01, 1, this.setM, "onMPress");   
 // перенос значения массы каждого тела в строку состояния

TM\_obj.addInputSlider("#input\_slider\_C1", "#input\_C1", 0, 10, 0.01, 0.1, this.setC1, "onC1Press"); // перенос значения частоты колебания 1 пружины в строку состояния

TM\_obj.addInputSlider("#input\_slider\_C2", "#input\_C2", 0, 10, 0.01, 0.1, this.setC2, "onC2Press"); // перенос значения частоты колебания 2 пружины в строку состояния

TM\_obj.addInputSlider("#input\_slider\_C3", "#input\_C3", 0, 10, 0.01, 0.1, this.setC3, "onC3Press"); // перенос значения частоты колебания 3 пружины в строку состояния

TM\_obj.addInputSlider("#input\_slider\_B", "#input\_B", 0, 10, 0.01, 0, this.setB, "onBPress");

"); // перенос значения вязкости среды в строку состояния

var count = true; // проводить ли расчет системы

var v1 = 0, v2 = 0; // инициализация скоростей 1 и 2 тела

// 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

});

var rw = 50;

var rh = 65;

var x0 = 230; // начальная координата откуда строится 2 пружина

var x02 = 430; // начальная координата откуда строится 3 пружина

var y0 = 17; // координата конца линии, составляющей пружину

// создаем 1 пружины

const coil1 = 10; // количество витков

var lines1 = []; // массив линии, составляющих пружину

for (var i = 0; i < coil1; i++ ) {

lines1[i] = ocanvas.display.line({

start: { x:0, y:65 },

end: { x:0, y:65 },

stroke: "5px black",

cap: "round"

}).add(); // рисуем линии таким образом: количество их совпадает с числом витков, назначаем координаты начала и конца линии, также она черного цвета, толщина абриса, с округленными концами

}

// аналогично рисуем вторую пружину

const coil2 = 10;

var lines2 = [];

for (var i = 0; i < coil2; i++ ) {

lines2[i] = ocanvas.display.line({

start: {x:x0, y:50},

end: {x:x0, y:50},

stroke: "5px red",

cap: "round"

}).add();

}

// Рисуем третью пружину

const coil3 = 10;

var lines3 = [];

for (var i = 0; i < coil3; i++ ) {

lines3[i] = ocanvas.display.line({

start: { x:x02, y:50},

end: { x:x02, y:50 },

stroke: "5px blue",

cap: "round"

}).add();

}

// Рисуем тело №1

var ellipse1 = ocanvas.display.ellipse({

x: x0,

y: y0+35, // координаты центра

radius: 39, // радиус тела

fill: "rgba(0, 170, 187, 1)" // цвет

}).add();

// Рисуем тело №2

var ellipse2 = ocanvas.display.ellipse({

x: x02,

y: y0+35,

radius: 39,

fill: "rgba(0, 170, 187, 1)"

}).add();

ellipse1.dragAndDrop({

changeZindex: true,

start: function () { count = false; this.fill = "rgba(0, 170, 187, 1)"; }, // отключаем расчет и делаем объект полупрозрачным

move: function () { this.y = y0+35; v1 = 0; drawSpring1(); drawSpring2();}, // запрещаем перемещение по вертикали

end: function () { count = true; this.fill = "rgba(0, 170, 187, 1)"; } // включаем расчет и убираем полупрозрачность

});

ellipse2.dragAndDrop({

changeZindex: true,

start: function () { count = false; this.fill = "rgba(0, 170, 187, 1)"; }, // отключаем расчет и делаем объект полупрозрачным

move: function () { this.y = y0+35; v2 = 0; drawSpring2(); drawSpring3();}, // запрещаем перемещение по вертикали

end: function () { count = true; this.fill = "rgba(0, 170, 187, 1)"; } // включаем расчет и убираем полупрозрачность

});

var vGraph = new TM\_graph(

"#vGraph", // на html-элементе #vGraph

"steps", "x", // подписи на осях

350, // сколько шагов по оси "x" отображается

-1, 1, 0.2); // мин. значение оси Y, макс. значение оси Y, шаг по оси Y

ocanvas.bind("mousedown", function () {count = false;});

function dynamics()

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

{

var f1 = - C1 \* (ellipse1.x - x0) - C2 \* (ellipse2.x - x02) - B \* v1;   
 // сила действующая на первый груз со стороны двух пружин

var f2 = f1 - C2 \* (ellipse2.x - x02) - C3 \* (ellipse2.x - x02);   
 // сила действующая на второй груз со стороны двух пружин

v1 += f1 / m \* dt; // высчитываем скорость 1 тела

v2 += f2 / m \* dt; // высчитываем скорость 2 тела

ellipse1.x += v1 \* dt; // высчитываем координаты 1 тела

ellipse2.x += v2 \* dt; // высчитываем координаты 2 тела

drawSpring1(); // обращаемся к функции перерисовки пружин №1

drawSpring2(); // обращаемся к функции перерисовки пружин №2

drawSpring3(); // обращаемся к функции перерисовки пружин №3

}

}

// обрисовка всех линии

function drawSpring1 ()

{

for (var i = 0; i < coil1; i++ )

{

lines1[i].start.x = (ellipse1.x + 25) / coil1 \* i;

lines1[i].end.x = (ellipse1.x + 25) / coil1 \* (i+1);

lines1[i].start.y = 50 + ( (i%2==0)? 1:-1) \* 25;

lines1[i].end.y = 50 + ( (i%2==0)? -1: 1) \* 25;

if (i==0) lines1[i].start.y = 50;

if (i==(coil1-1)) lines1[i].end.y = 50;

}

}

function drawSpring2()

{

for (var i = 0; i < coil2; i++ )

{

lines2[i].start.x = ellipse1.x + (ellipse2.x - ellipse1.x) / coil2 \* i;

lines2[i].end.x = ellipse1.x + (ellipse2.x - ellipse1.x) / coil2 \* (i+1);

lines2[i].start.y = 50 + ( (i%2==0)? 1:-1) \* 25;

lines2[i].end.y = 50 + ( (i%2==0)? -1: 1) \* 25;

if (i==0) lines2[i].start.y = 50;

if (i==(coil2-1)) lines2[i].end.y = 50;

}

}

function drawSpring3()

{

for (var i = 0; i < coil3; i++ )

{

lines3[i].start.x = ellipse2.x + (700 - ellipse2.x) / coil3 \* i;

lines3[i].end.x = ellipse2.x + (700 - ellipse2.x) / coil3 \* (i+1);

lines3[i].start.y = 50 + ( (i%2==0)? 1:-1) \* 25;

lines3[i].end.y = 50 + ( (i%2==0)? -1: 1) \* 25;

if (i==0) lines3[i].start.y = 50;

if (i==(coil3-1)) lines3[i].end.y = 50;

}

}

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

**2.3 Итог работы**

Результатом работы стала программа   
 «Моделирование колебаний n тел, связанных пружинами ».  
 В данном частном случае мы используем 2 тела.



***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 колебание n тел, связанных друг с другом

# Пружинами. В данном случае рассмотрено 2 тела, связанных между собой 3-мя пружинами. Программа написана таким образом, что добавить дополнительно связку груз-пружина не составляет труда. ***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. JavaScript: Подробное руководство (Definitive Guide), Давид Финнерман. Спб, 2007г.
9. JavaScript. Библия пользователя, Фленов Иван, Спб, 2005г.
10. Javascript и DHTML, сборник рецептов, Д.Гудман, Спб, 2004г
11. Размещено на Allbest.ru