



F1005 ELECTRICIDAD Y MAGNETISMO

## Proyecto Final: Simulador de partículas cargadas

Profesor Edgar René López Mena

*Pablo Muñoz Haro A01222422*  
*Andrés Barro Encinas A00226225*

*November 16, 2016*

## Contents

<b>1</b>	<b>Introducción</b>	<b>2</b>
<b>2</b>	<b>Requerimientos</b>	<b>2</b>
<b>3</b>	<b>Tecnologías utilizadas</b>	<b>2</b>
<b>4</b>	<b>Control de versiones</b>	<b>2</b>
<b>5</b>	<b>Identificación de entidades</b>	<b>3</b>
<b>6</b>	<b>Diagrama de clases (UML)</b>	<b>4</b>
<b>7</b>	<b>Página web</b>	<b>5</b>
<b>8</b>	<b>Física</b>	<b>5</b>
8.1	Ley de Coulomb . . . . .	5
8.2	Dipolo Eléctrico . . . . .	5
8.3	Energía potencial eléctrica . . . . .	5
<b>9</b>	<b>Código Fuente</b>	<b>5</b>
9.1	HTML . . . . .	5
9.2	Manifest constants . . . . .	7
9.3	Globals . . . . .	8
9.4	class Particle . . . . .	8
9.5	class ChargeSystem . . . . .	10
9.6	class TwoChargeSystem . . . . .	12
9.7	class ElectricDipole . . . . .	13
9.8	class ThreeChargeSystem . . . . .	14
9.9	onload script . . . . .	15
<b>10</b>	<b>Capturas de pantalla del programa en funcionamiento</b>	<b>17</b>

# 1 Introducción

Este documento contiene el análisis, diseño y documentación de el proyecto "Simulador de partículas cargadas" desarrollado para la clase de Electricidad y Magnetismo impartida por el profesor Edgar René en el Tecnológico de Monterrey Campus Guadalajara para el semestre de Ago-Dic 2016.

El proyecto consiste de una aplicación que puede ser visistada y utilizada en un navegador web moderno. La aplicación provee al usuario con ciertos sistemas de partículas predefinidos, los cuáles pueden ser modificados en cierta medida y que al momento de presionar "Start" se comienza a generar una animación que simula la interacción de partículas cargadas.

# 2 Requerimientos

- El sistema debe ser accesable a través de un navegador de internet.
- El sistema debe contener sistemas precargados de partículas que sólo requieran de pequeñas customizaciones.
- El sistema debe de incorporar el concepto de "velocidad del tiempo" que quiere decir que el usuario pueda configurar cuanto representa un segundo del mundo real en segundos de la simulación.
- El sistema debe permitir que el usuario seleccione la métrica de "pixeles por metro" de manera que pueda cambiar la escala de lo que es capaz de simular y visualizar.
- Las simulaciones deben correr de manera eficiente, sin interrupciones ni demoras.

# 3 Tecnologías utilizadas

Como cualquier otra página de internet, el proyecto hace uso significativo de los lenguajes de *HTML* y *CSS* (para la presentación) y *JavaScript* (para la lógica). En cuanto a este último también se utilizan las librerías de *PaperJS* para realizar dibujos en el elemento canvas de HTML, *underscore* para poder aplicar el paradigma funcional, *Vue* para sincronizar el input del usuario con los valores de las instancias y *bootstrap* para la interactividad de la interfaz de usuario.

# 4 Control de versiones

El sistema de control de versiones Git será utilizado mediante el portal Github. El repositorio con la rama maestra del código puede ser encontrado en la liga <https://github.com/pablo-munoz/proyecto-electro>.

## 5 Identificación de entidades

El sistema contará con entidades que representarán a las partículas. Cada instancia de estas entidades mantendrá la información necesaria sobre su estado así como las operaciones que permitirán dibujarlas, trasladarlas y modificarlas.

A su vez, el sistema contendrá la entidad de "sistemas de carga", los cuáles estarán compuestos de dos o más instancias de partículas y estarán encargadas de orquestrar la interacción entre estas.

6 Diagrama de clases (UML)

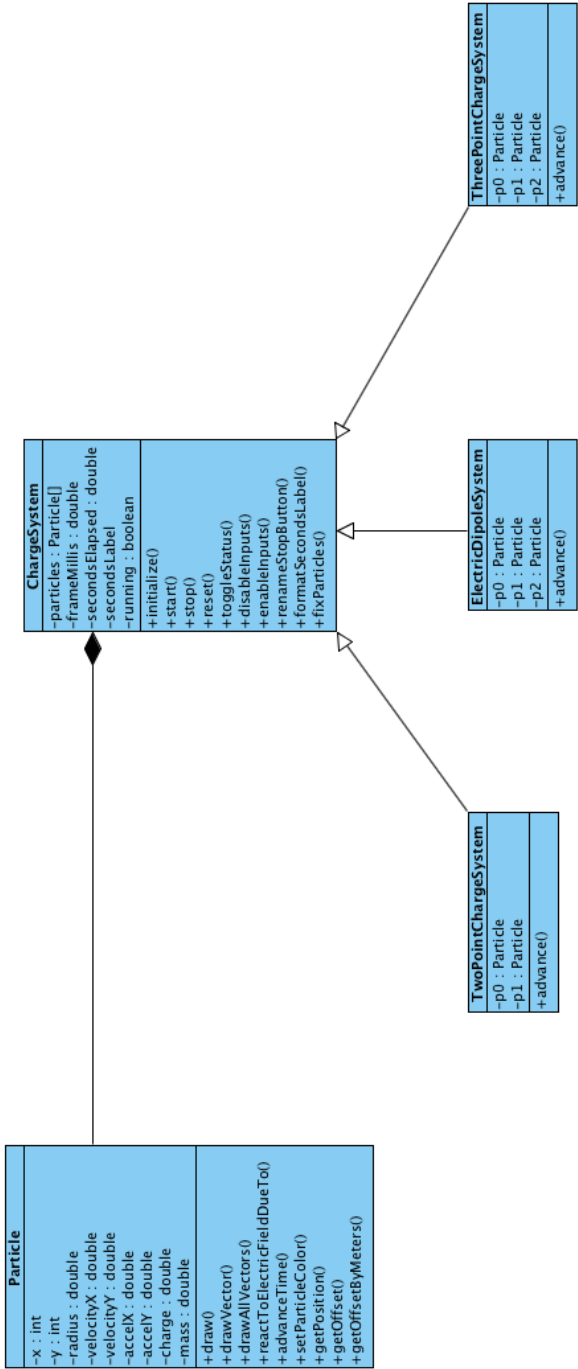


Figure 1: Calendario de trabajo

## 7 Página web

El programa puede ser accesado a través de la liga:

<https://pablo-munoz.github.io/proyecto-electro/index.html>

## 8 Física

### 8.1 Ley de Coulomb

La ley de Coulomb define como es que cargas puntuales interactúan entre sí. Esto incluye cuando las cargas se atraen o repelen. En este proyecto la fuerza eléctrica es calculada utilizando la ecuación de esta ley.

Debido a que la representación interna de las partículas guarda la coordenada  $x$  y  $y$ , el cálculo interno de las fuerzas también se desglosa en componentes  $x$  y  $y$ .

$$F_x = \cos(\theta) \frac{1}{4\pi\epsilon_0} \frac{|q_1 q_2|}{r^2} = \frac{x}{\sqrt{x^2 + y^2}} \frac{1}{4\pi\epsilon_0} \frac{|q_1 q_2|}{(\sqrt{x^2 + y^2})^2} = \frac{x|q_1 q_2|}{4\pi\epsilon_0 (x^2 + y^2)^{\frac{3}{2}}}$$
$$F_y = \sin(\theta) \frac{1}{4\pi\epsilon_0} \frac{|q_1 q_2|}{r^2} = \frac{y}{\sqrt{x^2 + y^2}} \frac{1}{4\pi\epsilon_0} \frac{|q_1 q_2|}{(\sqrt{x^2 + y^2})^2} = \frac{y|q_1 q_2|}{4\pi\epsilon_0 (x^2 + y^2)^{\frac{3}{2}}}$$

### 8.2 Dipolo Eléctrico

Un dipolo eléctrico es un Sistema formado por dos cargas separadas por una distancia. Las magnitudes de las cargas deben ser iguales y los signos opuestos.

Modificando los valores desde la interfaz gráfica, es posible crear un dipolo y observar los valores de las distintas magnitudes para cada una de las cargas individuales.

Si se comienza la simulación, se puede observar una representación aproximada de su comportamiento.

### 8.3 Energía potencial eléctrica

Para cada partícula, se puede obtener una energía potencial con respecto del resto de las partículas. Cada una de las partículas del sistema calculara su energía potencial de manera individual.

Para calcular se utiliza la formula sin modificaciones ya que esta no cuenta con componentes  $x$  o  $y$ , es solo una magnitud.

## 9 Código Fuente

### 9.1 HTML

```

1  <html>
2    <head>
3      <script
4        src="https://code.jquery.com/jquery-3.1.1.min.js"
5        integrity="sha256-hVVnYaiADRTO2PzUGmuLJr8BLUSjGIZsDYGMILv2b8="
6        crossorigin="anonymous"></script>
7      <!-- Latest compiled and minified CSS -->
8      <link rel="stylesheet"
9        ↪ href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
10
11        ↪ integrity="sha384-BVYiisIFeK1dGmJRAkycuHAHRg32OmUcww7on3RYdg4Va+PmSTsz/K68vbdEjh4u"
12        ↪ crossorigin="anonymous">
13      <!-- Latest compiled and minified JavaScript -->
14      <script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js"
15
16        ↪ integrity="sha384-Tc5IQib027qvyjSMfHjOMaLkfuWVxZxUPnCJA712mCWNIpG9mGCD8wGNicPD7Txa"
17        ↪ crossorigin="anonymous"></script>
18      <script src="https://cdn.jsdelivr.net/lodash/4.16.3/lodash.min.js"></script>
19      <script src="https://cdnjs.cloudflare.com/ajax/libs/paper.js/0.10.2/paper-full.js"
20        ↪ type="text/javascript"></script>
21      <script src="https://unpkg.com/vue/dist/vue.js"></script>
22      <script src="projectv2.js" type="text/javascript"></script>
23      <style type="text/css">
24        body {
25          background: url('grey.png') repeat;
26        }
27
28        canvas {
29          border: 1px solid #ccc;
30          background-color: white;
31        }
32
33        .particle-controls {
34          background-color: #fff;
35        }
36
37        .unselectable {
38          -webkit-user-select: none; /* Chrome/Safari */
39          -moz-user-select: none; /* Firefox */
40          -ms-user-select: none; /* IE10+ */
41          /* Rules below not implemented in browsers yet */
42          -o-user-select: none;
43          user-select: none;
44          cursor: pointer;
45        }
46      </style>
47    </head>
48    <body>
49      <div class="container" style="margin-top: 50">
50        <div class="col-md-8" id="canvas-container">

```

```

45     <canvas id="canvas" width="970" height="920"></canvas>
46 </div>
47 <div id="app" class="controls">
48   <div class="col-md-4">
49     <div class="form-group">
50       <select id="charge-system-selector" class="form-control"
51         ↪ v-on:change="changeChargeSystem()">
52         <option value="twoChargeSystem">Two charge system</option>
53         <option value="threeChargeSystem">Three charge system</option>
54         <option value="electricDipole">Two Static Charges</option>
55       </select>
56     </div>
57     <div class="form-group">
58       <div class="input-group">
59         <input class="form-control ppm" type="text" type="number"
60           ↪ v-bind:value="pixelsPerMeter"
61           ↪ v-on:change="updatePixelsPerMeter()" type="number"/>
62         <span class="input-group-addon">pixels per meter.</span>
63       </div>
64     </div>
65     <div class="form-group">
66       <div class="input-group">
67         <span class="input-group-addon">1 real second =</span>
68         <input class="form-control" type="text"
69           ↪ v-model="simulation.frameMillis" type="number"
70           ↪ v-on:change/>
71         <span class="input-group-addon">simul seconds.</span>
72       </div>
73     </div>
74     <button id="start-stop-btn" class="btn btn-primary"
75       ↪ onClick="simulation.toggleStatus();">Start</button>
76     <button id="reset-btn" class="btn btn-danger"
77       ↪ onClick="simulation.reset();">Reset</button>
78
79     <div class="particle-controls" v-for="(particle, index) in
80       ↪ simulation.particles">
81       <particle-controls
82         :particle="particle"
83         :index="index"/>
84     </div>
85   </div>
86 </div>
87 </body>
88 </html>

```

---

## 9.2 Manifest constants

---



```

1  paper.install(window);
2
3  var WINDOW_WIDTH  = 970;
4  var WINDOW_HEIGHT = 720;
5
6  const PERMITTIVITY    = 9 * Math.pow(10, 9);
7  const ELECTRON_CHARGE = -1.602 * Math.pow(10, -19);
8  const PROTON_CHARGE   = -ELECTRON_CHARGE;
9  const PROTON_MASS     = 1.6727 * Math.pow(10, -27);
10 const NEUTRON_MASS    = 1.6750 * Math.pow(10, -27);
11 const ELECTRON_MASS   = 9.110 * Math.pow(10, -31);
12 const VECTOR_WIDTH = 2;

```

---

### 9.3 Globals

```

1  var PIXELS_PER_METER = 100;
2  var simulation = undefined;
3  var app = undefined;

```

---

### 9.4 class Particle

```

1  class Particle {
2      // x, y, radius
3      constructor(args) {
4          _._assign(this, _._defaults(args, {
5              x: 0,
6              y: 0,
7              radius: 8,
8              velocityX: 0,          // m/s
9              velocityY: 0,          // m/s
10             accelX: 0,              // m/s
11             accelY: 0,              // m/s
12             charge: ELECTRON_CHARGE, // C
13             mass: ELECTRON_MASS      // kg
14         }));
15         this.forceX = 0;
16         this.forceY = 0;
17         this.potentialEnergy = 0;
18     }
19
20     draw() {
21         this.forceVector = new Path.Line(new Point(this.x, this.y), new Point(this.x,
22             ↪ this.y));
23         this.forceVector.strokeWidth = VECTOR_WIDTH;
24         this.forceVector.strokeColor = 'rgba(255, 255, 255, 0.5)';
25         this.accelVector = new Path.Line(new Point(this.x, this.y), new Point(this.x,
26             ↪ this.y));

```

```

25     this.accelVector.strokeWidth = VECTOR_WIDTH;
26     this.accelVector.strokeColor = 'rgba(255, 0, 0, 0.5)';
27     this.velocityVector = new Path.Line(new Point(this.x, this.y), new Point(this.x,
    ↪   this.y));
28     this.velocityVector.strokeWidth = VECTOR_WIDTH;
29     this.velocityVector.strokeColor = 'rgba(0, 255, 0, 0.5)';
30     this.circle = new Path.Circle(new Point(this.x * PIXELS_PER_METER + WINDOW_WIDTH/2,
    ↪   -this.y * PIXELS_PER_METER + WINDOW_HEIGHT/2), this.radius);
31     this.label = new PointText(this.x * PIXELS_PER_METER + WINDOW_WIDTH/2 - 2, -this.y
    ↪   * PIXELS_PER_METER + WINDOW_HEIGHT/2 + 2);
32     this.label.strokeColor = 'white';
33     this.label.content = this.name;
34     this.label.fontSize = 8;
35     this.circle.onMouseDown = this.label.onMouseDown = _.bind(function(event) {
36         this.circle.translate(event.delta);
37         this.label.translate(event.delta);
38         this.x = (this.circle.position.x - WINDOW_WIDTH/2) / PIXELS_PER_METER;
39         this.y = -(this.circle.position.y - WINDOW_HEIGHT/2) / PIXELS_PER_METER;
40         this.drawAllVectors();
41     }, this);
42     this.setParticleColor();
43 }
44
45 drawVector(whichVector) {
46     this[whichVector + 'Vector'].segments = [this.getPosition(),
    ↪   this.getOffsetByMeters(new Point(this[whichVector + 'X'] * PIXELS_PER_METER,
    ↪   -this[whichVector + 'Y'] * PIXELS_PER_METER))];
47 }
48
49 drawAllVectors() {
50     _.forEach(['force', 'velocity', 'accel'], _.bind(function(whichVector) {
51         this.drawVector(whichVector);
52     }, this));
53 }
54
55 reactToElectricFieldDueTo(otherParticleList) {
56     this.forceX = this.forceY = this.accelX = this.accelY = this.potentialEnergy = 0;
57
58     _.forEach(otherParticleList, _.bind(function(otherParticle) {
59         const distanceX = (this.x - otherParticle.x);
60         const distanceY = (this.y - otherParticle.y);
61         if((distanceX == 0 && distanceY == 0) || this === otherParticle) {
62             return;
63         }
64         const qq = (this.charge * otherParticle.charge);
65         const auxiliarForce = PERMITIVITY * ( ( qq ) / Math.pow(( distanceX * distanceX
    ↪   + distanceY * distanceY), 3/2) );
66         this.potentialEnergy += auxiliarForce * ( distanceX * distanceX + distanceY *
    ↪   distanceY);
67         this.forceX += distanceX * auxiliarForce;

```

```

68         this.forceY += distanceY * auxiliarForce;
69     }, this));
70     this.accelX = this.forceX / this.mass;
71     this.accelY = this.forceY / this.mass;
72 }
73
74 advanceTime(milliseconds) {
75     const seconds = milliseconds / 1000;
76     this.velocityX += this.accelX * seconds;
77     this.velocityY += this.accelY * seconds;
78     this.x += this.velocityX * seconds;
79     this.y += this.velocityY * seconds;
80     var translatePoint = new Point(this.velocityX * seconds * PIXELS_PER_METER, -1 *
    ↪ this.velocityY * seconds * PIXELS_PER_METER);
81     this.circle.translate(translatePoint);
82     this.label.translate(translatePoint);
83     this.drawAllVectors();
84 }
85
86 setParticleColor() {
87     if (this.charge > 0) {
88         this.circle.fillColor = 'red';
89     } else if (this.charge < 0) {
90         this.circle.fillColor = 'blue';
91     }
92 }
93
94 getPosition() {
95     return new Point(this.circle.position.x, this.circle.position.y);
96 }
97
98 getOffset(offsetPoint) {
99     return this.getPosition().add(offsetPoint);
100 }
101
102 getOffsetByMeters(offsetPointMeters) {
103     return this.getOffset(offsetPointMeters.multiply(PIXELS_PER_METER));
104 }
105
106 }

```

---

## 9.5 class ChargeSystem

```

1 class ChargeSystem {
2     constructor() {
3         this.initialize();
4         this.running = false;
5     }

```

```

6
7   initialize() {
8       paper.project.activeLayer.removeChildren();
9       this.particles = [];
10      this.frameMillis = 1000/60;
11      this.secondsElapsed = 0;
12
13      this.secondsLabel = new PointText(20, 20);
14      this.secondsLabel.fontSize = 16;
15      this.formatSecondsLabel();
16  }
17
18  start() {
19      this.refreshIntervalId = setInterval(_.bind(function() {
20          this.advance();
21          this.formatSecondsLabel();
22          this.secondsElapsed += this.frameMillis / 1000;
23      }, this), 1000/60/*this.frameMillis*/);
24      this.disableInputs();
25      this.running = true;
26      this.renameStartStopButton();
27  }
28
29  stop() {
30      clearInterval(this.refreshIntervalId);
31      this.running = false;
32      this.renameStartStopButton();
33  }
34
35  reset() {
36      PIXELS_PER_METER = 100;
37      app.$set(app, 'pixelsPerMeter', 100);
38      this.secondsElapsed = 0;
39      clearInterval(this.refreshIntervalId);
40      this.refreshIntervalId = undefined;
41      this.initialize();
42      this.running = false;
43      this.enableInputs();
44      this.renameStartStopButton();
45  }
46
47  toggleStatus() {
48      if (!this.running) {
49          this.start();
50      } else {
51          this.stop();
52      }
53  }
54
55  disableInputs() {

```

```

56     $('input').attr('disabled', 'disabled');
57 }
58
59 enableInputs() {
60     $('input').attr('disabled', null);
61 }
62
63 renameStartStopButton() {
64     if (this.running) {
65         $('#start-stop-btn').text('Stop');
66     } else {
67         $('#start-stop-btn').text('Start');
68     }
69 }
70
71 formatSecondsLabel() {
72     this.secondsLabel.content = "t = " + this.secondsElapsed + "s";
73 }
74
75 fixParticles(){
76     _.forEach(this.particles, _.bind(function(particle) {
77         particle.x = (particle.circle.position.x - WINDOW_WIDTH/2) / PIXELS_PER_METER;
78         particle.y = -(particle.circle.position.y - WINDOW_HEIGHT/2) /
79             ↪ PIXELS_PER_METER;
80     }, this));
81 }

```

---

## 9.6 class TwoChargeSystem

---

```

1  class TwoPointChargeSystem extends ChargeSystem {
2      initialize() {
3          super.initialize();
4          this.p0 = new Particle({
5              x: 3,
6              velocityX: 0,
7              velocityY: -5,
8              charge: ELECTRON_CHARGE,
9              mass: ELECTRON_MASS,
10             name: '0'
11         });
12         this.particles.push(this.p0);
13         this.p0.draw();
14
15         this.p1 = new Particle({
16             x: 0,
17             velocityX: 0,
18             velocityY: 0,

```

```

19         charge: PROTON_CHARGE,
20         mass: PROTON_MASS,
21         name: '1'
22     });
23     this.particles.push(this.p1);
24     this.p1.draw();
25 }
26
27 advance() {
28     _.forEach(this.particles, _.bind(function(particle) {
29         particle.reactToElectricFieldDueTo(this.particles);
30     }, this));
31     _.forEach(this.particles, _.bind(function(particle) {
32         particle.advanceTime(this.frameMillis);
33     }, this));
34 }
35 }

```

---

## 9.7 class ElectricDipole

---

```

1 class ElectricDipoleSystem extends ChargeSystem {
2     initialize() {
3         // p1 and p1 are the "fixed" ones
4         super.initialize();
5         this.p0 = new Particle({
6             x: 2,
7             y: 0,
8             charge: ELECTRON_CHARGE,
9             mass: ELECTRON_MASS,
10            name: '0'
11        });
12        this.particles.push(this.p0);
13        this.p0.draw();
14
15        this.p1 = new Particle({
16            y: -1,
17            charge: ELECTRON_CHARGE,
18            mass: ELECTRON_MASS,
19            name: '1'
20        });
21        this.particles.push(this.p1);
22        this.p1.draw();
23
24        this.p2 = new Particle({
25            y: 1,
26            charge: this.p1.charge,
27            mass: this.p1.mass,
28            name: '2'

```

```

29     });
30     this.particles.push(this.p2);
31     this.p2.draw();
32 }
33
34 advance() {
35     this.p0.reactToElectricFieldDueTo(this.particles);
36     this.p0.advanceTime(this.frameMillis);
37 }
38 }

```

---

## 9.8 class ThreeChargeSystem

---

```

1  class ThreePointChargeSystem extends ChargeSystem {
2      initialize() {
3          super.initialize();
4          this.p0 = new Particle({
5              x: 3,
6              velocityX: 0,
7              velocityY: 5,
8              charge: ELECTRON_CHARGE,
9              mass: ELECTRON_MASS,
10             name: '0'
11         });
12         this.particles.push(this.p0);
13         this.p0.draw();
14
15         this.p1 = new Particle({
16             x: 0,
17             velocityX: 0,
18             velocityY: 0,
19             charge: PROTON_CHARGE,
20             mass: PROTON_MASS,
21             name: '1'
22         });
23         this.particles.push(this.p1);
24         this.p1.draw();
25
26         this.p2 = new Particle({
27             x: -3,
28             velocityX: 0,
29             velocityY: -5,
30             charge: ELECTRON_CHARGE,
31             mass: ELECTRON_MASS,
32             name: '2'
33         });
34         this.particles.push(this.p2);
35         this.p2.draw();

```

```

36
37     _.forEach(this.particles, _.bind(function(particle) {
38         particle.reactToElectricFieldDueTo(this.particles);
39     }, this));
40 }
41
42 advance() {
43     _.forEach(this.particles, _.bind(function(particle) {
44         particle.reactToElectricFieldDueTo(this.particles);
45     }, this));
46     _.forEach(this.particles, _.bind(function(particle) {
47         particle.advanceTime(this.frameMillis);
48     }, this));
49 }
50 }

```

---

## 9.9 onload script

```

1 window.onload = function() {
2     $('#canvas').width($('#canvas-container').width());
3
4     WINDOW_WIDTH = $('#canvas-container').width();
5     WINDOW_HEIGHT = $('#canvas-container').height();
6
7     paper.setup($('#canvas'));
8
9     simulation = new TwoPointChargeSystem();
10
11     app = new Vue({
12         el: '#app',
13         data: {
14             pixelsPerMeter: PIXELS_PER_METER,
15             updatePixelsPerMeter: function(event) {
16                 var newValue = $('#input.ppm').val();
17                 PIXELS_PER_METER = newValue;
18                 app.pixelsPerMeter = newValue;
19                 simulation.fixParticles();
20             },
21             simulation: simulation,
22             changeChargeSystem: function() {
23                 var selectedSystem = $('#charge-system-selector').val();
24                 simulation.reset();
25                 app.simulation = simulation = new SYSTEMS_MAP[selectedSystem]();
26             }
27         },
28         components: {
29             "particle-controls": {
30                 props: ['index', 'particle'],

```



```

31         data: function() {
32             return {
33                 showing: true
34             };
35         },
36         template: `
37 <div class="panel panel-warning">
38   <div class="panel-heading unselectable" v-on:click="showing = !showing">
39     <span>Particle {{ index }}</span>
40     <span class="glyphicon glyphicon-chevron-down pull-right" v-show="!showing"></span>
41     <span class="glyphicon glyphicon-chevron-up pull-right" v-show="showing"></span>
42   </div>
43   <div class="panel-body" v-show="showing">
44     <div class="form-group">
45       <div class="input-group">
46         <span class="input-group-addon">q </span>
47         <input class="form-control" type="number" v-model="particle.charge"
48           ↵ v-on:change="particle.setParticleColor()" />
49         <span class="input-group-addon">C</span>
50       </div>
51     </div>
52     <div class="form-group">
53       <div class="input-group">
54         <span class="input-group-addon">m </span>
55         <input class="form-control" type="number" v-model="particle.mass" />
56         <span class="input-group-addon">kg</span>
57       </div>
58     </div>
59     <div class="form-group">
60       <div class="input-group">
61         <span class="input-group-addon">vx </span>
62         <input class="form-control" type="number" v-model="particle.velocityX" />
63         <span class="input-group-addon">m/s</span>
64       </div>
65     </div>
66     <div class="form-group">
67       <div class="input-group">
68         <span class="input-group-addon">ax </span>
69         <input class="form-control" type="number" v-model="particle.accelX" />
70         <span class="input-group-addon">m/s2</span>
71       </div>
72     </div>
73     <div class="form-group">
74       <div class="input-group">
75         <span class="input-group-addon">vy </span>
76         <input class="form-control" type="number" v-model="particle.velocityY" />
77         <span class="input-group-addon">m/s</span>
78       </div>
79     </div>
80   </div>
81 </div>
`

```

```

80         <div class="input-group">
81             <span class="input-group-addon">ay =</span>
82             <input class="form-control" type="number" v-model="particle.accelY"/>
83             <span class="input-group-addon">m/s^2</span>
84         </div>
85     </div>
86 </div>
87 </div>
88 {
89     }
90 }
91 });
92 }

```

---

## 10 Capturas de pantalla del programa en funcionamiento

