F1005 Electricidad y Magnetismo

Proyecto Final: Simulador de partículas cargadas

Profesor Edgar René López Mena

Pablo Muñoz Haro A01222422
Andrés Barro Encinas A00226225

November 15, 2016

Contents

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* (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)

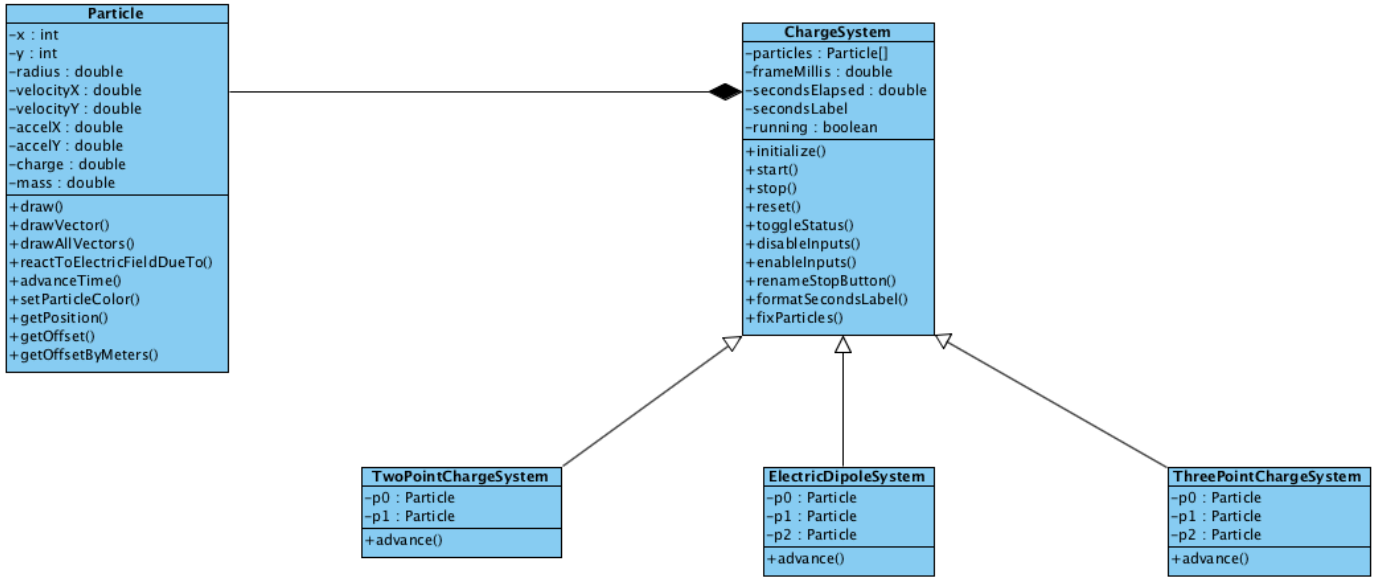


Figure 1: Calendario de trabajo

7 Código Fuente

7.1 HTML

```
<html>
    <head>
        <script
            src="https://code.jquery.com/jquery-3.1.1.min.js"
            integrity="sha256-hVVnYaiADRTO2PzUGmuLJr8BLUSjGIZsDYGmIJLv2b8="
            crossorigin="anonymous"></script>
        <!-- Latest compiled and minified CSS -->
        <link rel="stylesheet"
        ↪  href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"

              ↪  integrity="sha384-BVYiiSIFeK1dGmJRAkycuHAHRg32OmUcww7on3RYdg4Va+PmSTsz/K68vbdEjh4u"
              ↪  crossorigin="anonymous">
        <!-- Latest compiled and minified JavaScript -->
        <script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js"

                ↪  integrity="sha384-Tc5IQib027qvyjSMfHjOMaLkfuWVxZxUPnCJA7l2mCWNIpG9mGCD8wGNIcPD7Txa"
                ↪  crossorigin="anonymous"></script>
        <script src="https://cdn.jsdelivr.net/lodash/4.16.3/lodash.min.js"></script>
        <script src="https://cdnjs.cloudflare.com/ajax/libs/paper.js/0.10.2/paper-full.js"
        ↪  type="text/javascript"></script>
        <script src="https://unpkg.com/vue/dist/vue.js"></script>
        <script src="projectv2.js" type="text/javascript"></script>
        <style type="text/css">
        body {
            background: url('grey.png') repeat;
        }

        canvas {
            border: 1px solid #ccc;
            background-color: white;
        }

        .particle-controls {
            background-color: #fff;
        }

        .unselectable {
            -webkit-user-select: none; /* Chrome/Safari */
            -moz-user-select: none; /* Firefox */
            -ms-user-select: none; /* IE10+ */
            /* Rules below not implemented in browsers yet */
            -o-user-select: none;
            user-select: none;
            cursor: pointer;
        }
```

```
        </style>
    </head>
    <body>
        <div class="container" style="margin-top: 50">
            <div class="col-md-8" id="canvas-container">
                <canvas id="canvas" width="970" height="920"></canvas>
            </div>
            <div id="app" class="controls">
                <div class="col-md-4">
                    <div class="form-group">
                        <select id="charge-system-selector" class="form-control"
                        ↪  v-on:change="changeChargeSystem()">
                            <option value="twoChargeSystem">Two charge system</option>
                            <option value="threeChargeSystem">Three charge system</option>
                            <option value="electricDipole">Two Static Charges</option>
                        </select>
                    </div>
                    <div class="form-group">
                        <div class="input-group">
                            <input class="form-control ppm" type="text" type="number"
                            ↪  v-bind:value="pixelsPerMeter"
                            ↪  v-on:change="updatePixelsPerMeter()" type="number"/>
                            <span class="input-group-addon">piexels per meter.</span>
                        </div>
                    </div>
                    <div class="form-group">
                        <div class="input-group">
                            <span class="input-group-addon">1 real second =</span>
                            <input class="form-control" type="text"
                            ↪  v-model="simulation.frameMillis" type="number"
                            ↪  v-on:change/>
                            <span class="input-group-addon">simul seconds.</span>
                        </div>
                    </div>
                    <button id="start-stop-btn" class="btn btn-primary"
                    ↪  onClick="simulation.toggleStatus();">Start</button>
                    <button id="reset-btn" class="btn btn-danger"
                    ↪  onClick="simulation.reset();">Reset</button>

                    <div class="particle-controls" v-for="(particle, index) in
                    ↪  simulation.particles">
                        <particle-controls
                            :particle="particle"
                            :index="index"/>
                    </div>

                </div>
            </div>
        </div>
    </body>
```

```
</html>
```

7.2 Manifest constants

```
paper.install(window);

var WINDOW_WIDTH  = 970;
var WINDOW_HEIGHT = 720;

const PERMITIVITY     = 9 * Math.pow(10, 9);
const ELECTRON_CHARGE = -1.602 * Math.pow(10, -19);
const PROTON_CHARGE   = -ELECTRON_CHARGE;
const PROTON_MASS     = 1.6727 * Math.pow(10, -27);
const NEUTRON_MASS    = 1.6750 * Math.pow(10, -27);
const ELECTRON_MASS   =  9.110 * Math.pow(10, -31);
const VECTOR_WIDTH = 2;
```

7.3 Globals

```
var PIXELS_PER_METER = 100;
var simulation = undefined;
var app = undefined;
```

7.4 class Particle

```
class Particle {
    // x, y, radius
    constructor(args) {
        _.assign(this, _.defaults(args, {
            x: 0,
            y: 0,
            radius: 8,
            velocityX: 0,            // m/s
            velocityY: 0,            // m/s
            accelX: 0,               // m/s
            accelY: 0,               // m/s
            charge: ELECTRON_CHARGE, // C
            mass: ELECTRON_MASS      // kg
        }));
        this.forceX = 0;
        this.forceY = 0;
        this.potentialEnergy = 0;
    }

    draw() {
```

```
21          this.forceVector = new Path.Line(new Point(this.x, this.y), new Point(this.x,
             ↪  this.y));
22          this.forceVector.strokeWidth = VECTOR_WIDTH;
23          this.forceVector.strokeColor = 'rgba(255, 255, 255, 0.5)';
24          this.accelVector = new Path.Line(new Point(this.x, this.y), new Point(this.x,
             ↪  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.onMouseDrag = this.label.onMouseDrag = _.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 }
```

7.5 class ChargeSystem

```
class ChargeSystem {
    constructor() {
        this.initialize();
        this.running = false;
    }

    initialize() {
        paper.project.activeLayer.removeChildren();
        this.particles = [];
        this.frameMillis = 1000/60;
        this.secondsElapsed = 0;

        this.secondsLabel = new PointText(20, 20);
        this.secondsLabel.fontSize = 16;
        this.formatSecondsLabel();
    }

    start() {
        this.refreshIntervalId = setInterval(_.bind(function() {
            this.advance();
            this.formatSecondsLabel();
            this.secondsElapsed += this.frameMillis / 1000;
        }, this), 1000/60/*this.frameMillis*/);
        this.disableInputs();
        this.running = true;
        this.renameStartStopButton();
    }

    stop() {
        clearInterval(this.refreshIntervalId);
        this.running = false;
        this.renameStartStopButton();
    }

    reset() {
        PIXELS_PER_METER = 100;
        app.$set(app, 'pixelsPerMeter', 100);
        this.secondsElapsed = 0;
        clearInterval(this.refreshIntervalId);
        this.refreshIntervalId = undefined;
        this.initialize();
        this.running = false;
        this.enableInputs();
        this.renameStartStopButton();
    }

    toggleStatus() {
        if (!this.running) {
            this.start();
        } else {
```

```
            this.stop();
        }
    }

    disableInputs() {
        $('input').attr('disabled', 'disabled');
    }

    enableInputs() {
        $('input').attr('disabled', null);
    }

    renameStartStopButton() {
        if (this.running) {
            $('#start-stop-btn').text('Stop');
        } else {
            $('#start-stop-btn').text('Start');
        }
    }

    formatSecondsLabel() {
        this.secondsLabel.content = "t = " + this.secondsElapsed + "s";
    }

    fixParticles(){
        _.forEach(this.particles, _.bind(function(particle) {
            particle.x = (particle.circle.position.x - WINDOW_WIDTH/2) / PIXELS_PER_METER;
            particle.y = -(particle.circle.position.y - WINDOW_HEIGHT/2) /
             ↪  PIXELS_PER_METER;
        }, this));
    }
}
```

7.6 class TwoChargeSystem

```
class TwoPointChargeSystem extends ChargeSystem {
    initialize() {
        super.initialize();
        this.p0 = new Particle({
            x: 3,
            velocityX: 0,
            velocityY: -5,
            charge: ELECTRON_CHARGE,
            mass: ELECTRON_MASS,
            name: 'O'
        });
        this.particles.push(this.p0);
        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  }
```

7.7 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
```

```
        this.p2 = new Particle({
            y: 1,
            charge: this.p1.charge,
            mass: this.p1.mass,
            name: '2'
        });
        this.particles.push(this.p2);
        this.p2.draw();
    }

    advance() {
        this.p0.reactToElectricFieldDueTo(this.particles);
        this.p0.advanceTime(this.frameMillis);
    }
}
```

7.8 class ThreeChargeSystem

```
class ThreePointChargeSystem extends ChargeSystem {
    initialize() {
        super.initialize();
        this.p0 = new Particle({
            x: 3,
            velocityX: 0,
            velocityY: 5,
            charge: ELECTRON_CHARGE,
            mass: ELECTRON_MASS,
            name: '0'
        });
        this.particles.push(this.p0);
        this.p0.draw();

        this.p1 = new Particle({
            x: 0,
            velocityX: 0,
            velocityY: 0,
            charge: PROTON_CHARGE,
            mass: PROTON_MASS,
            name: '1'
        });
        this.particles.push(this.p1);
        this.p1.draw();

        this.p2 = new Particle({
            x: -3,
            velocityX: 0,
            velocityY: -5,
            charge: ELECTRON_CHARGE,
```

```
            mass: ELECTRON_MASS,
            name: '2'
        });
        this.particles.push(this.p2);
        this.p2.draw();

        _.forEach(this.particles, _.bind(function(particle) {
            particle.reactToElectricFieldDueTo(this.particles);
        }, this));
    }

    advance() {
        _.forEach(this.particles, _.bind(function(particle) {
            particle.reactToElectricFieldDueTo(this.particles);
        }, this));
        _.forEach(this.particles, _.bind(function(particle) {
            particle.advanceTime(this.frameMillis);
        }, this));
    }
}
```

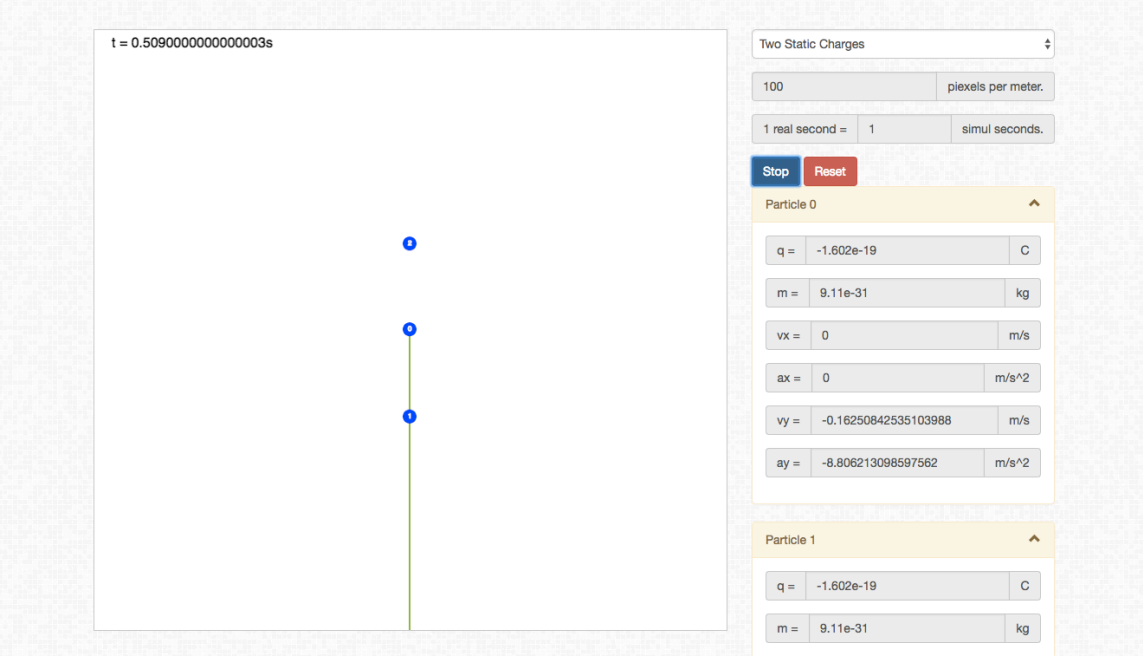
7.9 onload script

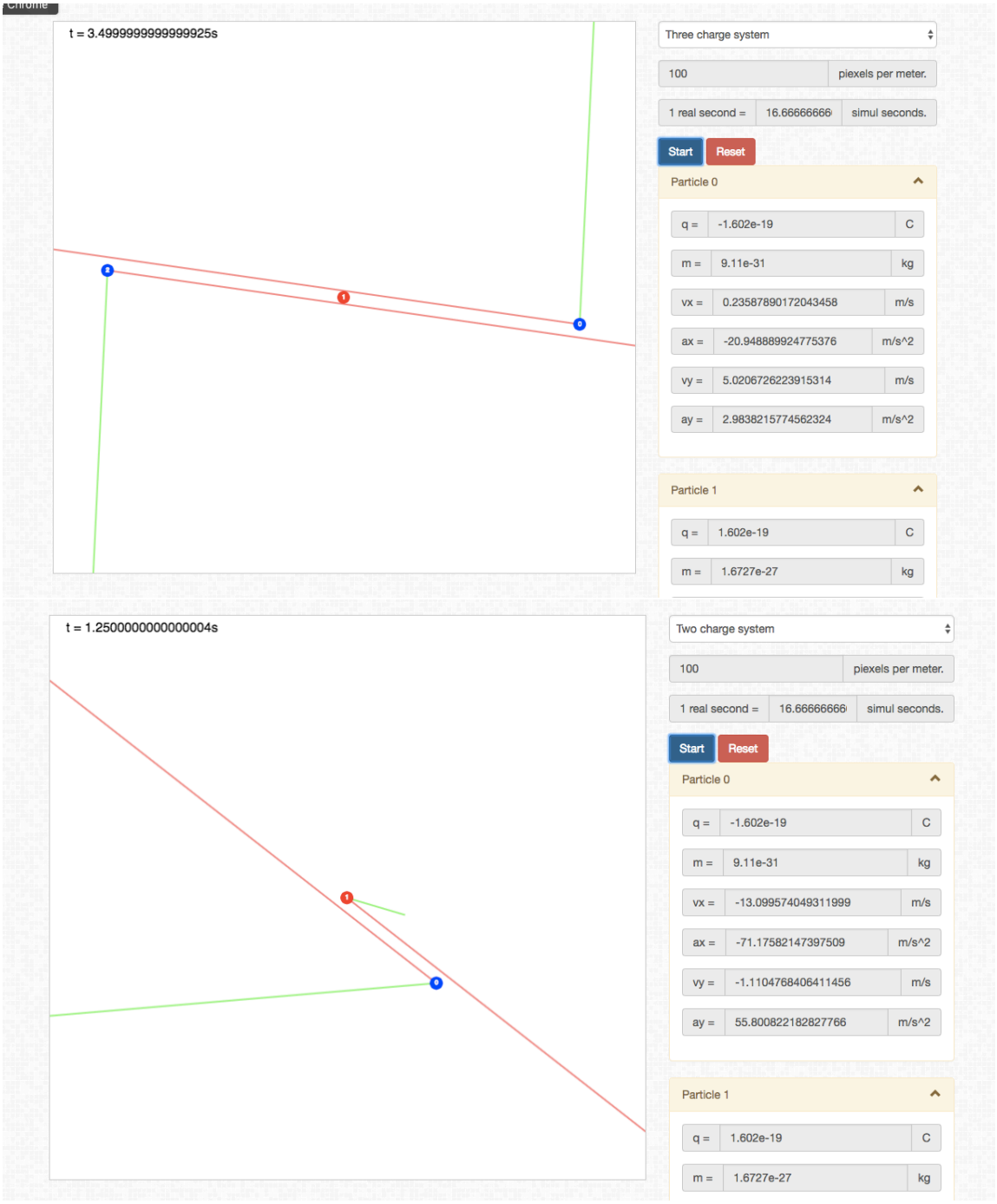
```
window.onload = function() {
    $('#canvas').width($('#canvas-container').width());

    WINDOW_WIDTH  = $('#canvas-container').width();
    WINDOW_HEIGHT = $('#canvas-container').height();

    paper.setup('canvas');

    simulation = new TwoPointChargeSystem();

    app = new Vue({
        el: '#app',
        data: {
            pixelsPerMeter: PIXELS_PER_METER,
            updatePixelsPerMeter: function(event) {
                var newValue = $('input.ppm').val();
                PIXELS_PER_METER = newValue;
                app.pixelsPerMeter = newValue;
                simulation.fixParticles();
            },
            simulation: simulation,
            changeChargeSystem: function() {
                var selectedSystem = $('#charge-system-selector').val();
                simulation.reset();
                app.simulation = simulation = new SYSTEMS_MAP[selectedSystem]();
```

```
            }
        },
        components: {
            "particle-controls": {
                props: ['index', 'particle'],
                data: function() {
                    return {
                        showing: true
                    };
                },
                template: `
<div class="panel panel-warning">
    <div class="panel-heading unselectable" v-on:click="showing = !showing">
        <span>Particle {{ index }}</span>
        <span class="glyphicon glyphicon-chevron-down pull-right" v-show="!showing"></span>
        <span class="glyphicon glyphicon-chevron-up pull-right" v-show="showing"></span>
    </div>
    <div class="panel-body" v-show="showing">
        <div class="form-group">
            <div class="input-group">
                <span class="input-group-addon">q =</span>
                <input class="form-control" type="number" v-model="particle.charge"
                 ↪  v-on:change="particle.setParticleColor()"/>
                <span class="input-group-addon">C</span>
            </div>
        </div>
        <div class="form-group">
            <div class="input-group">
                <span class="input-group-addon">m =</span>
                <input class="form-control" type="number" v-model="particle.mass"/>
                <span class="input-group-addon">kg</span>
            </div>
        </div>
        <div class="form-group">
            <div class="input-group">
                <span class="input-group-addon">vx =</span>
                <input class="form-control" type="number" v-model="particle.velocityX"/>
                <span class="input-group-addon">m/s</span>
            </div>
        </div>
        <div class="form-group">
            <div class="input-group">
                <span class="input-group-addon">ax =</span>
                <input class="form-control" type="number" v-model="particle.accelX"/>
                <span class="input-group-addon">m/s^2</span>
            </div>
        </div>
        <div class="form-group">
            <div class="input-group">
                <span class="input-group-addon">vy =</span>
```

```
                <input class="form-control" type="number" v-model="particle.velocityY"/>
                <span class="input-group-addon">m/s</span>
            </div>
        </div>
        <div class="form-group">
            <div class="input-group">
                <span class="input-group-addon">ay =</span>
                <input class="form-control" type="number" v-model="particle.accelY"/>
                <span class="input-group-addon">m/s^2</span>
            </div>
        </div>
    </div>
</div>
`
            }
        }
    });
}
```

8 Capturas de pantalla del programa en funcionamiento




t = 3.4999999999999925s
Three charge system
100
piexels per meter.
1 real second =
16.66666666
simul seconds.
Start
Reset
Particle 0
q =
-1.602e-19
C
m =
9.11e-31
kg
vx =
0.23587890172043458
m/s
ax =
-20.948889924775376
m/s^2
vy =
5.0206726223915314
m/s
ay =
2.9838215774562324
m/s^2
Particle 1
q =
1.602e-19
C
m =
1.6727e-27
kg
t = 1.2500000000000004s
Two charge system
100
piexels per meter.
1 real second =
16.66666666
simul seconds.
Start
Reset
Particle 0
q =
-1.602e-19
C
m =
9.11e-31
kg
vx =
-13.099574049311999
m/s
ax =
-71.17582147397509
m/s^2
vy =
-1.1104768406411456
m/s
ay =
55.800822182827766
m/s^2
Particle 1
q =
1.602e-19
C
m =
1.6727e-27
kg