Investigación sobre las Aplicaciones de la Física en la Computación

Autores: Jonathan Tombe Yalanda, Santiago Rojas
Institución: centro de Comercio y Turismo
Instructor: Rafael Roberto Pérez Grisales
Ficha: 2722493
Fecha de Elaboración: 12/02/2024

2. Tabla de contenido

3. Introducción

La intersección entre la física y la computación ha sido un campo fascinante y en constante evolución en la era moderna de la tecnología. La física, como ciencia fundamental que estudia las interacciones de la materia y la energía en el universo, proporciona los principios subyacentes que impulsan muchos de los avances en la informática. Desde la simulación de fenómenos físicos en videojuegos hasta la optimización de algoritmos utilizando conceptos físicos, las aplicaciones de la física en la computación son amplias y diversas.

En este informe, nos sumergiremos en este fascinante mundo, explorando las diversas formas en que la física se entrelaza con la computación para impulsar la innovación y resolver problemas complejos. A lo largo de nuestras investigaciones, examinaremos casos de estudio relevantes y ejemplos específicos para comprender mejor cómo se aplican estos principios en la práctica.

Nuestro objetivo es no solo comprender la relación entre la física y la computación, sino también explorar cómo podemos aprovechar estos conocimientos para desarrollar proyectos prácticos y significativos en el ámbito computacional. Desde la simulación de sistemas físicos complejos hasta la optimización de algoritmos utilizando métodos inspirados en la naturaleza, nuestro objetivo es demostrar cómo la integración de la física y la computación puede conducir a soluciones innovadoras y eficaces.

A lo largo de este informe, examinaremos en detalle los principios físicos seleccionados y cómo se aplican en el contexto de la computación. Además, describiremos un proyecto específico que hemos desarrollado para demostrar estos principios en acción. Finalmente, nos prepararemos para presentar nuestros hallazgos en clase, destacando la importancia de esta intersección entre dos campos aparentemente dispares.

4. Investigación de Aplicaciones Físicas en la Computación

Realizamos una investigación exhaustiva sobre las aplicaciones de la física en la computación, examinando cómo se utilizan los principios físicos en diferentes áreas

tecnológicas. A continuación, se presentan algunas áreas clave identificadas durante nuestra investigación:

4.1. Simulación física en videojuegos

Los videojuegos emplean modelos físicos para simular el movimiento de objetos, la colisión y la interacción con el entorno. Desde la física clásica hasta la física cuántica, los videojuegos utilizan una amplia gama de principios físicos para crear mundos virtuales realistas y experiencias inmersivas.

4.2. Optimización de algoritmos

Los algoritmos basados en principios físicos, como los algoritmos genéticos inspirados en la evolución biológica o los algoritmos de colonia de hormigas inspirados en el comportamiento animal, se utilizan para resolver problemas complejos de optimización en áreas como la inteligencia artificial, la logística y la planificación.

4.3. Diseño de hardware

En el diseño de hardware, se aplican conceptos físicos para crear dispositivos electrónicos eficientes y confiables. Desde la teoría de circuitos hasta la nanotecnología, la física desempeña un papel crucial en el desarrollo de componentes electrónicos, sistemas integrados y dispositivos de última generación.

5. Principios Físicos Seleccionados

Basándonos en nuestra investigación, seleccionamos dos principios físicos fundamentales que consideramos particularmente relevantes y aplicables en el contexto de la computación:

5.1. Aplicaciones de la Física Cuántica en la Computación

Explora cómo los principios de la física cuántica, como la superposición y la entrelazamiento, se utilizan en áreas como la criptografía cuántica y la computación cuántica.

5.2. Realidad Virtual y Realidad Aumentada

Investiga cómo se aplican los conceptos físicos en el desarrollo de entornos virtuales y aumentados, y cómo estos campos están revolucionando industrias como la educación, el entretenimiento y la medicina.

5.3. Gravedad

Es un fenómeno natural en el cual los objetos con masa son atraídos entre si nosotros la implementamos en simulación así: La simulación incluye un efecto de gravedad que afecta el movimiento de las bolas en el lienzo. Cada bola experimenta una fuerza de gravedad que actúa sobre ella, lo que causa que caigan hacia abajo y reboten cuando alcanzan los límites inferiores del lienzo. Este principio físico es fundamental para simular el movimiento realista de los objetos en un entorno gravitacional..

Ecuación de Gravedad

La gravedad de la

tierra es: 9.8 m/s2

F gravedad$=m \cdot g$

Donde
F es la fuerza gravitacional,
m es la masa del objeto,
g es la aceleración debido a la gravedad.
En tu simulación, las bolas tienen una masa constante y la aceleración debida a la gravedad está representada por la variable **gravity**.

5.6. Fricción

Aunque la fricción no es un principio físico fundamental en sí mismo, es una fuerza importante que se opone al movimiento en el mundo físico. En tu simulación, la fricción se utiliza para simular la resistencia al movimiento de las bolas en el lienzo. La fricción afecta la velocidad de las bolas y puede influir en su comportamiento. Por lo tanto, la fricción podría considerarse como otro principio físico aplicado en nuestro programa; el cual aplicaremos al aumentar o disminuir esta sobre el suelo en el cual cae la pelota.

5.7. Ecuación de Fricción

F fricción $=-k \cdot v$

Donde:
F es la fuerza de fricción,
k es el coeficiente de fricción,
v es la velocidad del objeto.

En la simulación, el efecto de la fricción se logra multiplicando la velocidad de las bolas por un factor de fricción en la función **updateBall**.

5.8. Velocidad Inicial: la velocidad inicial es la velocidad de un objeto cuando se empieza a observar el movimiento de este; en nuestra aplicación la usaremos para alterar la velocidad de movimiento de, una pelota en caída libre controlado por el usuario

Ecuación velocidad inicial

la ecuación de la velocidad inicial es:

Vi= Vf-(at)

Vi: velocidad inicial.

Vf: velocidad final.

A: aceleración

t: tiempo

6. Descripción del Proyecto

Nuestro proyecto tiene como objetivo desarrollar una simulación interactiva que demuestre los principios físicos seleccionados: la gravedad y el fricción. Implementaremos un programa en JavaScript que simule el movimiento de objetos bajo la influencia de estos principios, permitiendo al usuario interactuar con la simulación y explorar diferentes escenarios físicos.

6.1. Implementación técnica

Utilizaremos HTML, CSS y JavaScript para desarrollar la simulación. El lienzo de HTML5 nos proporcionará una plataforma para dibujar objetos y visualizar el movimiento, mientras que JavaScript se utilizará para calcular y actualizar la posición de los objetos en cada fotograma de la animación.

6.2. Visualización y gráficos

Mejoraremos la visualización de la simulación con gráficos mejorados y efectos visuales realistas. Utilizaremos técnicas de renderizado avanzadas, como sombreado y texturizado, para crear una experiencia visualmente atractiva y envolvente para el usuario.

7. Resultados y Análisis

Durante los resultados y análisis, se lograron los siguientes resultados significativos:

7.1 . Implementación Exitosa de la Simulación Interactiva: Se logró implementar una simulación interactiva que demuestra los principios físicos seleccionados (gravedad y magnetismo) de manera efectiva y precisa. La simulación permite al usuario interactuar con objetos en movimiento y explorar diferentes escenarios físicos dentro del entorno del navegador.

7.2 . Optimización del Rendimiento: Se realizaron esfuerzos significativos para optimizar el rendimiento de la simulación, asegurando una experiencia fluida y receptiva para el usuario incluso en entornos con recursos limitados. Esto se logró mediante la optimización del código JavaScript y el uso eficiente de los recursos del navegador.

7.3 . Visualización Atractiva y Realista: La simulación se presenta con una visualización atractiva y realista, gracias al uso de técnicas de rende rizado avanzadas, como sombreado y texturizado. Esto ayuda a mejorar la experiencia del usuario y a hacer que la simulación sea más inmersa y envolvente.

7.4 .Validación de los Principios Físicos: Se validaron los principios físicos seleccionados (gravedad y magnetismo) mediante la observación directa de su

aplicación en la simulación. Los objetos en movimiento siguen trayectorias realistas y se comportan de acuerdo con las leyes físicas establecidas.

8. Anexo.

En esta sección, se presentan los códigos fuente utilizados en el desarrollo del proyecto de simulación de física en computación. Estos códigos incluyen el código HTML que proporciona la estructura básica de la página web, el código CSS que define los estilos visuales y la apariencia de la simulación, y el código JavaScript que implementa la lógica y el comportamiento de la simulación.

8.1. Código HTML utilizado en el proyecto

El código HTML proporciona la estructura básica para la simulación. Define un lienzo (**canvas**) en el cual se dibujarán las bolas y se aplicarán las animaciones. Además, incluye botones para iniciar, pausar y reiniciar la simulación, así como controles deslizantes para ajustar la gravedad, la fricción y la velocidad inicial de las bolas, también se agrego una breve explicación de como funcionaran las pelotas si se le cambian los valores asi mismo se mostrara los números de los valores que se pueden cambiar en la simulación; este también cuanta con un modal donde se explicara brevemente la aplicación finalmente tiene una explicación sencilla de como actuaran los valores que cambien las personas que usen nuestra aplicación.

8.2 Código CSS para estilizar la simulación

El código CSS proporciona estilos para la simulación, como el tamaño y el color del lienzo, así como el estilo de los botones y controles deslizantes. Estos estilos ayudan a mejorar la apariencia y la usabilidad de la simulación en el navegador; también proporciona el diseño como se vera la modal y la explicación.

8.3 .Código JavaScript para la lógica de la simulación.

El código JavaScript contiene la lógica de la simulación. Define las propiedades y comportamientos de las bolas, como su posición, velocidad, gravedad y fricción. Además, incluye funciones para dibujar las bolas en el lienzo, actualizar su posición en cada fotograma de la animación y controlar la interacción del usuario con la simulación; así como se comportara la modal en el navegador también los números en los campos editables.

8.4 .Diseño en el navegador.

Esta captura muestra cómo se ve el programa en el navegador web. Se puede observar la representación visual de la simulación de la pelota en movimiento dentro del lienzo del navegador.

9. Desafíos y Soluciones

Durante la implementación del proyecto, nos enfrentamos a varios desafíos que requerían soluciones creativas y técnicas. Uno de los principales desafíos fue garantizar un rendimiento óptimo de la simulación en todas las plataformas y dispositivos, especialmente cuando se trataba de la animación de objetos en el lienzo HTML5.

Uno de los desafíos técnicos más significativos fue optimizar el rendimiento de la animación para garantizar una experiencia fluida y sin problemas para el usuario. Esto implicó ajustes en los cálculos de la física de los objetos, así como en la frecuencia de actualización de la animación para evitar retrasos o tartamudeos visuales. Para abordar estos desafíos, dedicamos tiempo a investigar y experimentar

con diferentes enfoques y técnicas de optimización. Realizamos pruebas exhaustivas en una variedad de dispositivos y navegadores para identificar y corregir posibles problemas de rendimiento y compatibilidad. Además, colaboramos estrechamente para aprovechar el conocimiento y la experiencia del equipo en el desarrollo web y la programación JavaScript.

En última instancia, pudimos superar estos desafíos mediante un enfoque iterativo de desarrollo, donde probamos y refinamos continuamente el código para mejorar su eficiencia y estabilidad. Esta experiencia nos permitió aprender y crecer como desarrolladores, fortaleciendo nuestra comprensión de los principios fundamentales de la programación y la física aplicada en el contexto de la computación.

10. Conclusión

En este informe, hemos explorado las diversas aplicaciones de la física en la computación, destacando cómo estos dos campos aparentemente dispares convergen para impulsar la innovación y resolver problemas complejos. Desde la simulación física en videojuegos hasta el diseño de hardware inspirado en principios físicos, hemos investigado cómo la integración de la física y la computación ha dado lugar a avances significativos en la tecnología moderna.

Durante nuestra investigación, identificamos áreas clave donde los principios físicos juegan un papel fundamental en el desarrollo de soluciones tecnológicas innovadoras. Desde la optimización de algoritmos utilizando métodos inspirados en la naturaleza hasta la aplicación de la física cuántica en la criptografía y la computación cuántica, hemos explorado cómo estos conceptos subyacentes han transformado diversas industrias y campos de estudio.

Nuestro proyecto específico, una simulación interactiva que demuestra los principios de la gravedad y el magnetismo, ilustra cómo estos conceptos físicos pueden ser implementados en un contexto computacional práctico. Utilizando HTML, CSS y JavaScript, desarrollamos una herramienta que permite a los usuarios explorar y experimentar con escenarios físicos simulados, destacando la relevancia y el potencial de esta intersección entre la física y la computación. A través de este trabajo, hemos demostrado la importancia de comprender la relación entre la física y la computación, no solo en términos de avances tecnológicos, sino también en términos de desarrollo de habilidades y capacidad para abordar problemas complejos desde una perspectiva interdisciplinaria. Esperamos que este informe

inspire futuras investigaciones y proyectos en esta fascinante área de estudio, y que continúe impulsando la innovación y el progreso en el campo de la tecnología computacional.

Concluimos nuestro informe con la convicción de que la intersección entre la física y la computación representa un vasto y emocionante campo de exploración, lleno de oportunidades para descubrir y crear soluciones innovadoras que impacten positivamente en nuestra sociedad y en el mundo en general.

11. Bibliografía

1. tomado de: open tex 2024 fricción: 6.2 Fricción - Física universitaria volumen 1 | OpenStax.

2. tomado de: google https://www.google.com/?hl=es.

3. tomado de: Wikipedia gravedad https://www.mineduc.gob.gt/DIGECADE/documents/Telesecundaria/Recursos%20Digitales/20%20Recursos%20Digitales%20TS%20BY-SA%203.0/01%20CIENCIAS%20NATURALES/U6%20pp%20134%20gravedad.pdf.

4 tomado de: Conceptos Fuerzas en el Universo https://www.ign.es/web/resources/actividades/gravimetria/Conceptos.pdf.

Anexos

anexo 1. index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
    <link rel="stylesheet" href="style.css">
    <link rel="preconnect" href="https://fonts.googleapis.com">
    <link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>
    <link
```

```
href="https://fonts.googleapis.com/css2?family=Raleway:ital,wght@0,100..900;1,100..900&family=Roboto:ital,wght@0,300;0,400;0,500;0,900;1,500&display=swap" rel="stylesheet">
</head>
<body>
  <div id="guide" class="guide">
    <p id="guideText"></p>
    <button id="nextStep">Siguiente</button>
  </div>
  <div class="info">
    <p>Animación de pelota que rebota</p>
  </div>
  <div class="contenedor">
    <canvas id="canvas"></canvas>
  </div>

  <div class="controls">
    <button id="startButton">Comenzar</button>
    <button id="pauseButton" class="opaque">Pausa</button>
    <button id="resetButton" class="opaque">Reiniciar</button>
    <label for="radiusRange" class="opaque">Tamaño de la Bola</label>
    <input type="range" id="radiusRange" min="10" max="50" step="1" value="20" class="opaque">
    <span id="radiusValue" class="opaque">20</span>
    <label for="gravityRange" class="opaque">Gravedad</label>
    <input type="range" id="gravityRange" min="0" max="2" step="0.1" value="0.5" class="opaque">
    <span id="gravityValue" class="opaque">0.5</span>
    <label for="frictionRange" class="opaque">Fricción</label>
    <input type="range" id="frictionRange" min="0.9" max="1" step="0.01" value="0.99" class="opaque">
    <span id="frictionValue" class="opaque">0.99</span>
    <label for="velocityRange" class="opaque">Velocidad inicial</label>
    <input type="range" id="velocityRange" min="5" max="20" step="1" value="10" class="opaque">
    <span id="velocityValue" class="opaque" >10</span>
  </div>

  <div class="container">
```

```
      <div class="column">
        <h1>Al aumentar la gravedad la pelota no llegará muy lejos.</h1>
      </div>
      <div class=" column2">
        <h2>Al aumentar la fricción, la pelota no se moverá de su posición de caída
final.</h2>
        <h3>Al disminuir la velocidad inicial, la pelota se moverá más lento.</h3>
      </div>
    </div>

    <script src="javascript.js"></script>
</body>
</html>
```

anexo 2. style.css

```
body,

html {



margin:

0;



padding:

0;

    height:

100%;



overflow:

hidden;



backgrou
```

```
nd-color:
black;
}
.contened
or{
   width:
100%;
   height:
auto;

backgrou
nd:
url('src/i
mg/Free\
Photo\
_\
Abstract\
luxury\
blur\
dark\
grey\
and\
black\
gradient\
,\ used\
```

as\

backgrou

nd\

studio\

wall\ for\

display\

your\

products

_.jpg');

backgrou

nd-

repeat:

no-

repeat;

backgrou

nd-size:

cover;

backgrou

nd-

position:

center

center;

image-

```
rendering
: crisp-
edges;
   box-
shadow:
0 0px 5px
hsl(220,
16%,
96%)
   ;
}
canvas {

display:
block;

margin:
auto;

backgrou
nd-color:
transpare
nt;
}
```

```
.info {
    text-
align:
center;
    font-
family:
Arial,
sans-
serif;
    font-
size:
3.5rem;
    text-
shadow:
0 0 10px;
    color:
white;
}

.controls
{
    text-
align:
center;
```

```
margin-
top:
20px;
  font-
size:
20px;
  color:
white;
  text-
shadow:
0 0 3px;
}

.controls
button {

margin: 0
10px;

padding:
10px
15px;
  font-
```

size:

20px;

font-

family:

'Times

New

Roman',

Times,

serif;

border-

radius:

8px;

text-

shadow:

0 0 3px;

}

.controls

input[typ

e="range

"] {

margin:

5px;

}

```
.guide {

position:
fixed;
   top:
50%;
   left:
50%;

transfor
m:
translate(
-50%, -
50%);

backgrou
nd-color:
rgba(255,
255, 255,
0.9);

padding:
20px;
   border-
```

radius:

5px;

box-

shadow:

0 0 10px

rgba(0, 0,

0, 0.3);

z-

index:

100;

text-

align:

center;

}

.

container

{

display:

flex;

justify-

content:

space-

around;

width:

```
40%;

}


.column {


padding:

10px;

   justify-

content:

center;

}


.column2

{


display:

flex;

   align-

items:

center;

   justify-

content:

space-

between;
```

```
}

h1, h2, h3
{
   font-
family:
'Roboto',
sans-
serif;
   color:
#eeeaea;
  font-
size:
14px;
  margin-
right:
10px;
}

@media(
min-
width:
380px){
```

```
}

@media(
min-
width:
580px){



}

@media(
min-
width:
720px) {



}

@media(
min-
width:
920px) {



}

@media(
min-
width:
1200px)
{}
```

anexo 3. javascript.js

```
const canvas =
document.getElementById('canvas');
const ctx = canvas.getContext('2d');

const width = canvas.width =
window.innerWidth - 30;
const height = canvas.height =
window.innerHeight - 300;

const balls = [];
const numBalls = 20;
const startX = 10;
const startY = 10;
let animationId;
let isPaused = false;
let isRunning = false;
let gravity = 9.8;
let friction = 0.99;
let initialVelocity = 10;
let radius = 20;

const guideSteps = [
   "Bienvenido a la animación de pelota que
```

```
rebota. Haz clic en 'Siguiente' para
continuar.",
    "El botón 'Comenzar' inicia la animación
de las pelotas.",
    "El botón 'Pausa' detiene temporalmente
la animación.",
    "El botón 'Reiniciar' reinicia la animación
con la configuración actual.",
    "El control deslizante 'Tamaño de la Bola'
ajusta el radio de las pelotas.",
    "El control deslizante 'Gravedad' ajusta la
fuerza de gravedad que actúa sobre las
pelotas.",
    "El control deslizante 'Fricción' ajusta la
cantidad de fricción entre las pelotas y las
paredes.",
    "El control deslizante 'Velocidad inicial'
ajusta la velocidad inicial de las pelotas.",
    "¡Eso es todo! Ahora puedes jugar con la
animación y ajustar los controles a tu
gusto."
];
let currentStep = 0;

function disableControls() {
    const startButton =
```

```
document.getElementById('startButton');
    const pauseButton =
document.getElementById('pauseButton');
    const resetButton =
document.getElementById('resetButton');
    const radiusRange =
document.getElementById('radiusRange');
    const gravityRange =
document.getElementById('gravityRange');
    const frictionRange =
document.getElementById('frictionRange');
    const velocityRange =
document.getElementById('velocityRange');

    startButton.disabled = true;
    pauseButton.disabled = true;
    resetButton.disabled = true;
    radiusRange.disabled = true;
    gravityRange.disabled = true;
    frictionRange.disabled = true;
    velocityRange.disabled = true;
}

function enableControls() {
    const startButton =
document.getElementById('startButton');
```

```
    const pauseButton =
document.getElementById('pauseButton');
    const resetButton =
document.getElementById('resetButton');
    const radiusRange =
document.getElementById('radiusRange');
    const gravityRange =
document.getElementById('gravityRange');
    const frictionRange =
document.getElementById('frictionRange');
    const velocityRange =
document.getElementById('velocityRange');

    startButton.disabled = false;
    pauseButton.disabled = false;
    resetButton.disabled = false;
    radiusRange.disabled = false;
    gravityRange.disabled = false;
    frictionRange.disabled = false;
    velocityRange.disabled = false;
}



function updateOpacity(step) {
    const startButton =
document.getElementById('startButton');
```

```
    const pauseButton =
document.getElementById('pauseButton');
    const resetButton =
document.getElementById('resetButton');
    const radiusLabel =
document.getElementById('radiusRange').p
reviousElementSibling;
    const radiusRange =
document.getElementById('radiusRange');
    const radiusValue =
document.getElementById('radiusValue');
    const gravityLabel =
document.getElementById('gravityRange').
previousElementSibling;
    const gravityRange =
document.getElementById('gravityRange');
    const gravityValue =
document.getElementById('gravityValue');
    const frictionLabel =
document.getElementById('frictionRange').
previousElementSibling;
    const frictionRange =
document.getElementById('frictionRange');
    const frictionValue =
document.getElementById('frictionValue');
    const velocityLabel =
```

```
document.getElementById('velocityRange').
previousElementSibling;
    const velocityRange =
document.getElementById('velocityRange');
    const velocityValue =
document.getElementById('velocityValue');

    startButton.classList.add('opaque');
    pauseButton.classList.add('opaque');
    resetButton.classList.add('opaque');
    radiusLabel.classList.add('opaque');
    radiusRange.classList.add('opaque');
    radiusValue.classList.add('opaque');
    gravityLabel.classList.add('opaque');
    gravityRange.classList.add('opaque');
    gravityValue.classList.add('opaque');
    frictionLabel.classList.add('opaque');
    frictionRange.classList.add('opaque');
    frictionValue.classList.add('opaque');
    velocityLabel.classList.add('opaque');
    velocityRange.classList.add('opaque');
    velocityValue.classList.add('opaque');

    switch (step) {
        case 1:
```

```
startButton.classList.remove('opaque');
            break;
        case 2:

pauseButton.classList.remove('opaque');
            break;
        case 3:

resetButton.classList.remove('opaque');
            break;
        case 4:

radiusLabel.classList.remove('opaque');

radiusRange.classList.remove('opaque');

radiusValue.classList.remove('opaque');
            break;
        case 5:

gravityLabel.classList.remove('opaque');

gravityRange.classList.remove('opaque');

gravityValue.classList.remove('opaque');
            break;
```

```
        case 6:

frictionLabel.classList.remove('opaque');

frictionRange.classList.remove('opaque');

frictionValue.classList.remove('opaque');
            break;
        case 7:

velocityLabel.classList.remove('opaque');

velocityRange.classList.remove('opaque');

velocityValue.classList.remove('opaque');
            break;

    }
}

function getRandomColor() {
    const letters = '0123456789ABCDEF';
    let color = '#';
    for (let i = 0; i < 6; i++) {
        color +=
letters[Math.floor(Math.random() * 16)];
```

```
    }
    return color;
}

function createBall() {
    const ball = {
        x: startX,
        y: startY,
        radius: radius,
        color: getRandomColor(),
        velocityX: initialVelocity,
        velocityY: initialVelocity,
        gravity: gravity,
        friction: friction
    };
    balls.push(ball);
}

for (let i = 0; i < numBalls; i++) {
    createBall();
}

function drawBall(ball) {
    ctx.beginPath();
    ctx.arc(ball.x, ball.y, ball.radius, 0,
Math.PI * 2);
```

```
    ctx.fillStyle = ball.color;
    ctx.fill();
    ctx.closePath();
}

function updateBall(ball) {
    ball.velocityY += ball.gravity;
    ball.x += ball.velocityX;
    ball.y += ball.velocityY;

    if (ball.y + ball.radius > height) {
        ball.velocityY *= -ball.friction;
        ball.y = height - ball.radius;
    } else if (ball.y - ball.radius < 0) {
        ball.velocityY *= -ball.friction;
        ball.y = ball.radius;
    }

    if (ball.x + ball.radius > width) {
        ball.velocityX = -(initialVelocity / 2) *
ball.friction;
        ball.x = width - ball.radius;
    } else if (ball.x - ball.radius < 0) {
        ball.velocityX = (initialVelocity / 2) *
ball.friction;
        ball.x = ball.radius;
```

```
    } else {
        ball.velocityX *= ball.friction;
        if (Math.abs(ball.velocityX) < 0.1) {
            ball.velocityX = 0;
        }
    }
}

function animate() {
    ctx.clearRect(0, 0, width, height);

    for (let i = 0; i < balls.length; i++) {
        drawBall(balls[i]);
        updateBall(balls[i]);
    }

    if (!isPaused) {
        animationId =
requestAnimationFrame(animate);
    }
}

document.getElementById('startButton').ad
dEventListener('click', function () {
    if (!isRunning) {
        animate();
```

```
        isRunning = true;
    }
});
document.getElementById('pauseButton').a
ddEventListener('click', function () {
    isPaused = !isPaused;
    if (!isPaused && isRunning) {
        animate();
    }
});

document.getElementById('resetButton').a
ddEventListener('click', function () {
    cancelAnimationFrame(animationId);
    isRunning = false;
    isPaused = false;
    balls.length = 0;
    for (let i = 0; i < numBalls; i++) {
        createBall();
    }
    animate();
    isRunning = true;
});

document.getElementById('radiusRange').a
ddEventListener('input', function (e) {
```

```
    radius = parseInt(e.target.value);

document.getElementById('radiusValue').te
xtContent = radius;
    balls.forEach(ball => {
        ball.radius = radius;
    });
});


document.getElementById('gravityRange').
addEventListener('input', function (e) {
    gravity = parseFloat(e.target.value);

document.getElementById('gravityValue').t
extContent = gravity;
    balls.forEach(ball => {
        ball.gravity = gravity;
    });
});


document.getElementById('frictionRange').
addEventListener('input', function (e) {
    friction = parseFloat(e.target.value);
```

```
document.getElementById('frictionValue').t
extContent = friction;
    balls.forEach(ball => {
        ball.friction = friction;
    });
});

document.getElementById('velocityRange').
addEventListener('input', function (e) {
    initialVelocity = parseInt(e.target.value);

document.getElementById('velocityValue').t
extContent = initialVelocity;
    balls.forEach(ball => {
        ball.velocityX = initialVelocity;
        ball.velocityY = initialVelocity;
    });
});



function showGuide() {
            const guideText =
document.getElementById('guideText');
            const nextButton =
document.getElementById('nextStep');
            const guide =
```

```
document.getElementById('guide');

        disableControls();
        updateOpacity(currentStep);

        guideText.textContent =
guideSteps[currentStep];

        if (currentStep === guideSteps.length
- 1) {
            nextButton.textContent =
'Finalizar';
        }

        guide.style.display = 'block';

        nextButton.addEventListener('click',
nextGuideStep);
    }

    function nextGuideStep() {
        const guide =
document.getElementById('guide');

        if (currentStep === guideSteps.length
- 1) {
```

```
            guide.style.display = 'none';
            enableControls();
        } else {
            currentStep++;
            showGuide();
        }
    }

    window.addEventListener('load',
showGuide);
```

Anexo 4. diseño programa