Aplicación Software para el Desarrollo de Prácticas de Física Mecánica - ANEXOS

Trabajo de Grado

Mónica Andrea Camacho Dorado
Wilson Harvey Imbachi Meneses

Director: Ing. Francisco Franco Obando Díaz
Codirector(a): Ing. Judy Cristina Realpe Chamorro

Universidad del Cauca
Facultad de Ingeniería Electrónica y Telecomunicaciones
Departamento de Electrónica, Instrumentación y Control.
Popayán, octubre de 2016.

Contenido

Anexo A

ECUACIONES DE MOVIMIENTO

En el presente apartado se realiza el desarrollo completo de las ecuaciones que representan cada uno de los movimientos de los diferentes cuerpos a través de los sistemas físicos definidos en el capítulo 2. Estas ecuaciones son la base para la implementación de las distintas secuencias de animación contenidas en cada práctica

1. PRÁCTICA DE VECTORES

La presente práctica está basada en el esquema mostrado en la figura 1. En dicho sistema se involucran algunos de los conceptos generales más importantes relacionados con la temática de vectores, mediante los cuales se llevan a cabo los procedimientos necesarios que permiten su implementación y operación con otros vectores dentro de un plano.

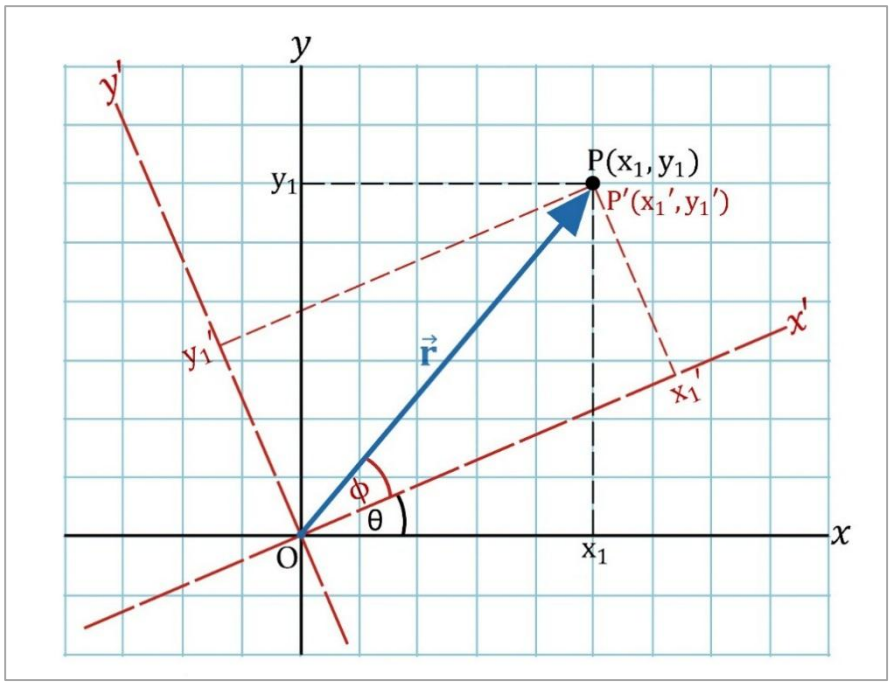


Figura 2. Práctica de vectores – Sistema general.

Considerando $\beta = \theta + \Phi$, las expresiones generales más importantes considerados dentro de la práctica de vectores son las siguientes:

- Representación de las coordenadas (x_1, y_1) de un punto P en términos de sus coordenadas polares,:

$$x_1 = r\cos\beta \quad \textbf{(1)}$$

$$y_1 = r\sin\beta \quad \textbf{(2)}$$

- Magnitud del segmento r en función de los téminos x y y:

$$r = \sqrt{x_1{}^2 + y_1{}^2} \quad \textbf{(3)}$$

- **Representación de un vector:** Teniendo el vector $\vec{A} = \overrightarrow{OP}$, el cual está conformado por las componentes $\overrightarrow{A_x}$ y $\overrightarrow{A_y}$, su representación se puede hacer a través de sus vectores componentes de la siguiente forma:

$$\vec{A} = \overrightarrow{A_x} + \overrightarrow{A_y} \quad \textbf{(4)}$$

- **Adición de vectores:** Si se tienen los vectores $\vec{\mathrm{A}}$ y $\vec{\mathrm{B}}$, la adición de ellos se lleva a cabo sumando los vectores componentes de cada uno de ellos:

$$\vec{A} + \vec{B} = \left(\overrightarrow{A_x} + \overrightarrow{A_y}\right) + \left(\overrightarrow{B_x} + \overrightarrow{B_y}\right) = \left(\overrightarrow{A_x} + \overrightarrow{B_x}\right) + \left(\overrightarrow{A_y} + \overrightarrow{B_y}\right) \quad \textbf{(5)}$$

- Las componentes A_x y A_y, la magnitud del vector $\vec{\mathrm{A}}$ y la dirección β para un sistema de coordenadas rectangular se calculan respectivamente por medio de las siguientes expresiones:

$$A_x = A\cos\beta \quad \textbf{(6)}$$

$$A_y = A\sin\beta \quad \textbf{(7)}$$

$$A = \sqrt{A_x{}^2 + A_y{}^2} \quad \textbf{(8)}$$

$$\beta = \tan^{-1}\left(\frac{A_y}{A_x}\right) \quad \textbf{(9)}$$

- **Rotación de ejes coordenados:** De acuerdo a la figura 1 Las ecuaciones para determinar las coordenadas del punto dentro del plano con ejes rotados (denominadas ecuaciones de transformación) son las siguientes:

$$y = x'\sin\theta + y'\cos\theta \quad \textbf{(10)}$$

$$x = x'\cos\theta - y'\sin\theta \quad \textbf{(11)}$$

2. PRÁCTICA DE MOVIMIENTO EN DOS DIMENSIONES.

El sistema diseñado para la práctica de movimiento bidimensional se divide en cuatro tramos a través de los cuales el cuerpo se desplaza realizando diferentes tipos de movimiento, como se muestra en la figura 2.

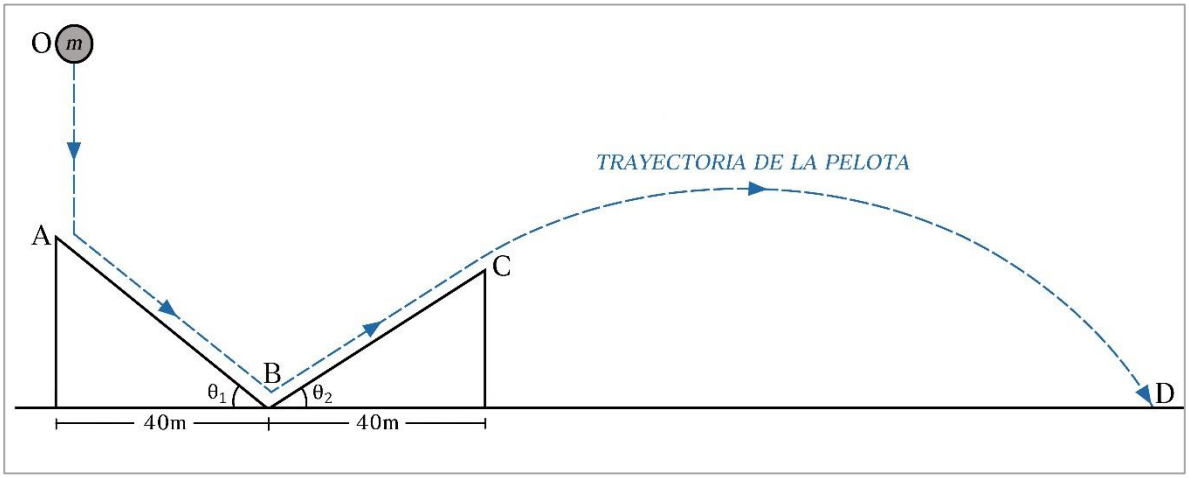


Figura 2. Práctica de vectores – Sistema general.

Las ecuaciones que describen el movimiento de la pelota a través de los tramos definidos se muestran a continuación:

2.1. MOVIMIENTO DE CAIDA LIBRE (TRAMO O-A):

El primer movimiento que realiza la pelota es el de caída libre, el cual comprende el tramo O-A del sistema general, como lo muestra la figura 3.

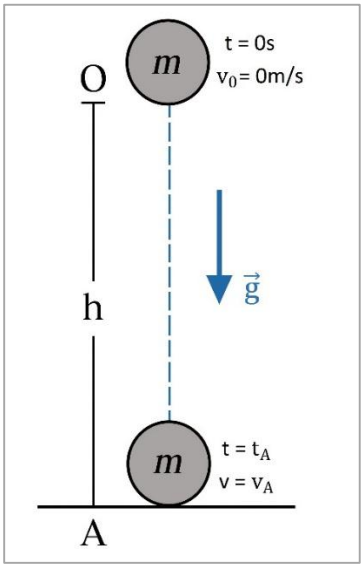


Figura 3. Movimiento de caída Libre.

En primera instancia, la pelota de masa m se ubica a una altura inicial h respecto del vértice superior de la rampa 1 (punto A) y se deja caer bajo la acción de la fuerza de gravedad (los efectos causados por la fricción del aire se desprecian). Tomando como sistema de referencia el punto A y suponiendo negativa la dirección del movimiento de la pelota se determinan las expresiones generales de aceleración, velocidad y posición en función del tiempo dentro del movimiento de caída libre. De acuerdo al sistema de referencia escogido se tiene que la aceleración de la partícula debida a la gravedad se expresa de la siguiente forma:

$$a = -g \qquad \textbf{(12)}$$

Este valor de aceleración es constante durante todo el recorrido de la pelota. Por su parte, la expresión general de velocidad se obtiene integrando el valor aceleración en función del tiempo:

$$v = \int a\,dt = \int (-g)\,dt = -gt + c_1$$

Se determina el valor de la constante c_1, para ello se evalúa la función de velocidad v en el instante $t = 0$:

$$v(t = 0) = v_0 = 0$$

$$v_0 = -g(0) + c_1 = 0 \rightarrow c_1 = 0$$

Por lo tanto la función de velocidad para el movimiento de caída libre queda definida como:

$$v = -gt \qquad \textbf{(13)}$$

De forma similar se calcula la expresión para la posición en y de la pelota, integrando la función general de velocidad respecto al tiempo:

$$y = \int v dt = \int (-gt) dt = \frac{-gt^2}{2} + c_2$$

Se determina el valor de la constante c_2 evaluando la función de posición y en el instante *t = 0*:

$$\mathrm{y}(t=0) = y_0 = h$$

$$y_0 = \frac{-g(0)^2}{2} + c_2 = h \rightarrow c_2 = h$$

De esta forma la ecuación de posición en dirección y para el movimiento de caída libre es:

$$y = \frac{-gt^2}{2} + h \qquad \textbf{(14)}$$

Con las ecuaciones de movimiento del tramo de caída libre se calcula el tiempo t_A que demora la pelota en ir desde el punto O (*h* metros) hasta el punto A (*0* metros). Para ello se toma la función de posición como $y = 0$:

$$y = 0 = \frac{-g(t_A)^2}{2} + h$$

$$\frac{g(t_A)^2}{2} = h$$

$$t_A = \sqrt{\frac{2h}{g}} \tag{15}$$

Conociendo el tiempo t_A se determina el valor de velocidad de la pelota en el punto A:

$$v(t = t_A) = v_A = -g\,t_A = -g\left(\sqrt{\frac{2h}{g}}\right)$$

$$v_A = -\sqrt{2gh} \tag{16}$$

2.2. MOVIMIENTO SOBRE EL PLANO INCLINADO DESCENDENTE (TRAMO A-B):

Después del movimiento de caída libre la pelota se sitúa dentro de la rampa 1, como lo muestra la figura 4. Este plano inclinado de tipo descendente cuenta con una base de 40 metros de longitud y un ángulo de inclinación variable (θ_1). Las únicas fuerzas que actúan sobre la pelota son la normal (N) y el peso (W), pues se considera que la superficie de la rampa es completamente lisa y por tanto no presenta ningún tipo de fricción. La velocidad inicial de la pelota dentro de la rampa es v_A (velocidad final de la partícula en caída libre).

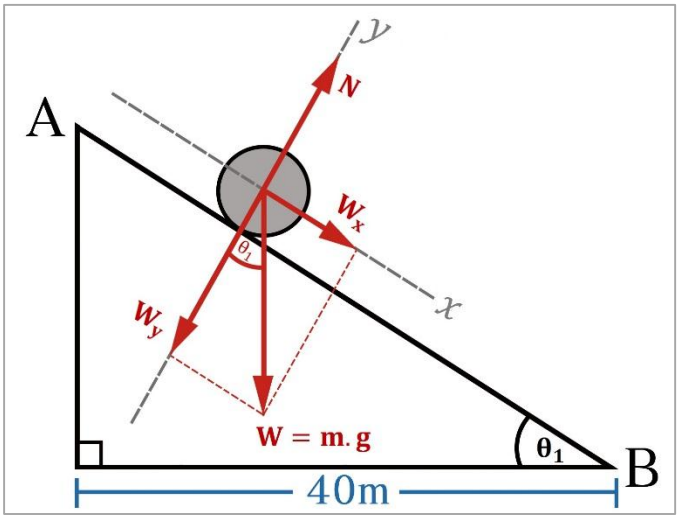


Figura 4. Plano Inclinado descendente.

Se establece un sistema de coordenadas xy acorde con el tipo de movimiento de la pelota. De acuerdo a la figura 4 se observa que todo el movimiento se realiza en dirección x, por lo tanto la aceleración de la partícula dentro de la rampa 1 equivale al valor de su componente en dicha dirección ($a_y = 0$). Con base en la segunda ley de Newton se determinan los valores correspondientes de aceleración en dirección x y fuerza normal ejercida por la rampa sobre la pelota:

$$\sum F_x = mg\sin\theta_1 = ma_x$$

$$a_x = g\sin\theta_1 \qquad \textbf{(17)}$$

$$\sum F_y = N - mg\cos\theta_1 = ma_y = 0$$

$$N = mg\cos\theta_1 \qquad \textbf{(18)}$$

Conociendo el valor de la aceleración se define la expresión general de posición de la pelota a lo largo de la rampa, para ello se recurre a la ecuación de desplazamiento en función del tiempo del movimiento de partículas en una dimensión. Relacionando dicha ecuación con los parámetros correspondientes de este subsistema se tiene:

$$x - x_0 = v_0 t + \frac{1}{2}at^2$$

$$x - x_0 = v_{x01} t + \frac{1}{2}a_x t^2$$

Considerando que la partícula inicia su movimiento en el punto A se tiene que $x_0 = 0$, de esta forma su función de posición es:

$$x = v_{x01} t + \frac{1}{2}\mathrm{a}_x t^2 = v_A t + \frac{1}{2}g\sin\theta_1 t^2$$

$$x = \frac{1}{2}g\sin\theta_1 t^2 + v_A t \qquad \textbf{(19)}$$

En el tiempo $t = t_B$ la pelota recorre toda la longitud de la rampa 1 ($x = d_1$). Al igualar este valor con la ecuación (19) se tiene que:

$$x = d_1 = \frac{1}{2} g \sin\theta_1 t_B{}^2 + v_A t_B$$

Por lo tanto el tiempo t_B equivale a:

$$\frac{1}{2} g \sin\theta_1 \left(t_B\right)^2 + v_A \left(t_B\right) - \mathrm{d}_1 = 0$$

$$t_B = \frac{-v_A \pm \sqrt{v_A{}^2 + 2g \sin\theta_1 d_1}}{g \sin\theta_1}$$

De la solución de la ecuación cuadrática, la expresión del tiempo t_B es:

$$t_B = \frac{-v_A + \sqrt{v_A{}^2 + 2g \sin\theta_1 d_1}}{g \sin\theta_1} \qquad \textbf{(20)}$$

Al igual que en el caso de la aceleración, la velocidad de la partícula es una cantidad vectorial y se puede expresar por medio de sus componentes en x y y dentro del sistema coordenado establecido, sin embargo debido a que no hay movimiento de la pelota en dirección y su velocidad depende solo de la componente en x, por lo tanto:

$$\vec{v} = \overrightarrow{v_x} + \overrightarrow{v_y}$$

$$v_y = 0 \rightarrow v = v_x$$

Se determina la velocidad con que la pelota llega al final de la rampa 1, para ello se utiliza la expresión de velocidad como función del desplazamiento definida dentro del movimiento unidimensional de partículas:

$$v^2 - v_0{}^2 = 2a\left(x - x_0\right)$$

Considerando que la velocidad inicial de la pelota es $v_0 = v_{x01} = v_A$ y que su recorrido al final de la rampa 1 es $x = d_1$ se calcula la velocidad final $v = v_B$ como:

$$v_x^{\,2} = v_{x01}^{\,2} + 2a_x\left(x - x_0\right)$$

$$v_x^{\,2} = v_A^{\,2} + 2g\sin\theta_1 d_1$$

$$v_x = v_B = \sqrt{v_A^{\,2} + 2g\sin\theta_1 d_1} \qquad \textbf{(21)}$$

En términos generales la velocidad de la pelota en función del tiempo para cualquier punto de la rampa 1 es:

$$v = v_{x01} + a_x t$$

$$v = v_A + g\sin\theta_1 t \qquad \textbf{(22)}$$

2.3. MOVIMIENTO SOBRE EL PLANO INCLINADO ASCENDENTE (TRAMO B-C):

Después de recorrer la rampa 1 la pelota realiza su movimiento a través del plano inclinado ascendente, el cual comprende el tramo B-C del sistema general, como lo muestra la figura 5. Este plano cuenta también con una base de 40 metros de longitud y un ángulo de elevación variable θ_2. Al igual que en la primera rampa, las únicas fuerzas que actúan sobre la pelota son la fuerza normal (N) y el peso (W) representado por sus componentes en x y y. En este caso tampoco se considera algún tipo de fricción en la superficie de la rampa.

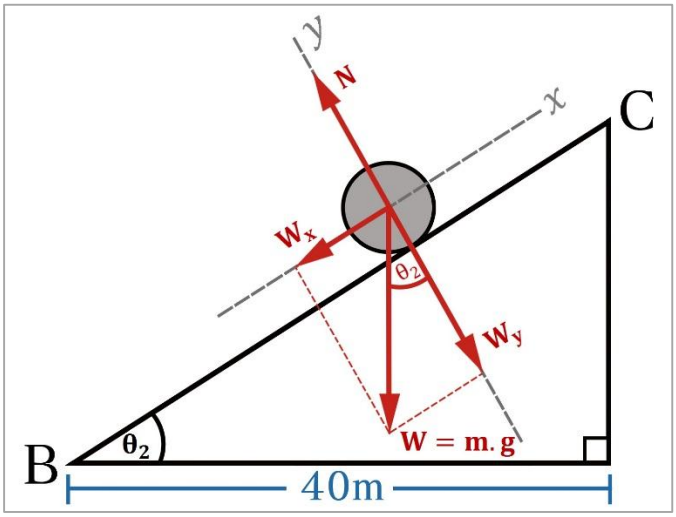


Figura 5. Plano Inclinado ascendente.

Partiendo de la segunda ley de Newton los valores de aceleración en x y la fuerza normal (N) vienen dados como:

$$\sum F_x = -mg\sin\theta_2 = ma_x$$

$$a_x = -g\sin\theta_2 \qquad \textbf{(23)}$$

$$\sum F_y = N - mg\cos\theta_2 = ma_y = 0$$

$$N = mg\cos\theta_2 \qquad \textbf{(24)}$$

Tomando la ecuación de desplazamiento respecto al tiempo del movimiento unidimensional se tiene lo siguiente:

$$x - x_0 = v_0 t + \frac{1}{2}\mathrm{a}\,t^2$$

$$x - x_0 = v_{x02} t + \frac{1}{2}\mathrm{a}_x\, t^2$$

Del mismo modo que en el caso anterior se toma la posición inicial de la pelota (punto B) como $x_0 = 0$ y la velocidad inicial como $v_0 = v_{x02} = v_B$. De esta manera la función de posición en x se expresa como:

$$x = v_{x02}t + \frac{1}{2}\mathrm{a}_x t^2 = v_B t - \frac{1}{2} g \sin\theta_2 t^2$$

$$x = -\frac{1}{2} g \sin\theta_2 t^2 + v_B t \qquad \textbf{(25)}$$

En el tiempo $t = t_C$ la pelota recorre la longitud total de la rampa 2 ($x = d_2$). Al igualar este valor con la ecuación () se tiene que:

$$x = d_2 = -\frac{1}{2} g \sin\theta_2 {t_C}^2 + v_B t_C$$

Por lo tanto el tiempo t_C equivale a:

$$-\frac{1}{2} g \sin\theta_2 \left(t^2\right) + v_A\left(t\right) - d_2 = 0$$

$$t_C = \frac{-v_B \pm \sqrt{{v_B}^2 - 2g\sin\theta_2 d_2}}{-g\sin\theta_2}$$

$$t_C = \frac{v_B - \sqrt{{v_B}^2 - 2g\sin\theta_2 d_2}}{g\sin\theta_2} \qquad \textbf{(26)}$$

La velocidad de la pelota solo depende de su componente en x ya que en la dirección y no existe movimiento, por lo tanto:

$$\vec{v} = \overrightarrow{v_x} + \overrightarrow{v_y}$$

$$v_y = 0 \rightarrow v = v_x$$

Se determina la velocidad con que la pelota llega al final de la rampa 2, para esto se utiliza la ecuación de velocidad en función del desplazamiento del movimiento unidimensional:

$$v^2 - v_0^{\,2} = 2a\left(x - x_0\right)$$

Reemplazando los valores correspondientes de velocidad inicial, aceleración y recorrido de la pelota se determina el valor de velocidad al final de la rampa 2 ($v_x = v_c$):

$$v_x^{\,2} = v_{x02}^{\,2} + 2a_x\left(x - x_0\right)$$

$$v_x^{\,2} = v_B^{\,2} - 2g\sin\theta_2 d_2$$

$$v_x = v_C = \sqrt{v_B^{\,2} - 2g\sin\theta_2 d_2} \qquad \textbf{(27)}$$

Para este caso la expresión general de velocidad de la pelota en cualquier punto de la rampa 2 es:

$$v = v_{x02} + a_x t$$

$$v = v_B - g\sin\theta_2 t \qquad \textbf{(28)}$$

2.4. MOVIMIENTO PARABÓLICO (TRAMO C-D):

Finalmente el último tramo del sistema general comprende un movimiento de tipo parabólico como se observa en la figura 6. El peso de la partícula es la única fuerza presente durante el movimiento ya que se desprecian los efectos resistivos del aire.

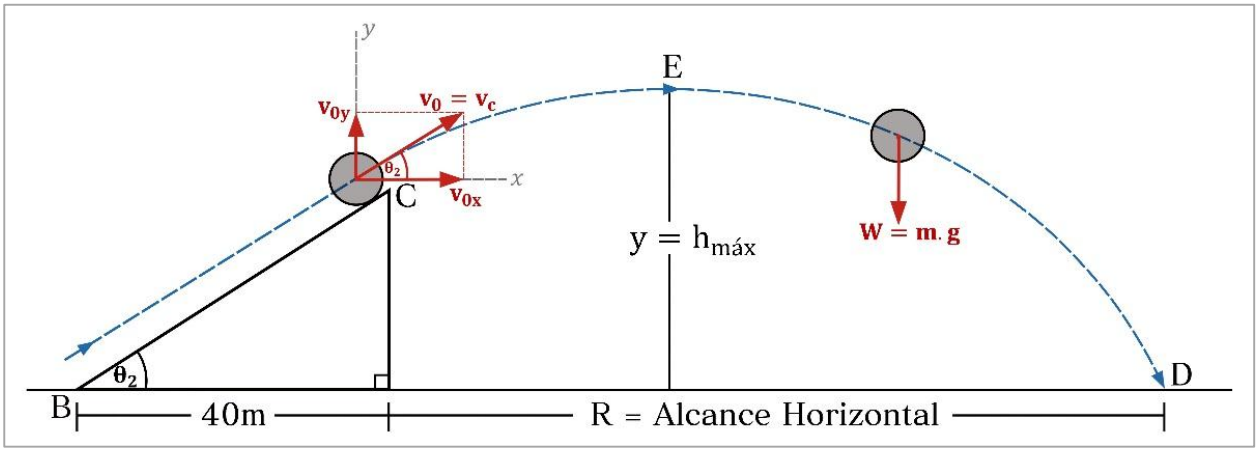


Figura 6. Movimiento parabólico.

Tomando la velocidad inicial de la pelota dentro como $v_0 = v_c$ y un sistema de ejes coordenados xy, se definen las respectivas componentes del vector de velocidad en el punto C de la siguiente forma:

$$v_{0x} = v_0 \cos\theta_2 = v_C \cos\theta_2$$

$$v_{0y} = v_0 \sin\theta_2 = v_C \sin\theta_2 \qquad \textbf{(29)}$$

Donde θ_2 es el ángulo de elevación de la rampa 2 en la que se origina el movimiento del proyectil (pelota). Dado que el movimiento parabólico se compone de un movimiento horizontal uniforme y uno vertical acelerado, se tienen los siguientes valores de aceleración en las direcciones x y y:

$$\sum F_x = ma_x \rightarrow 0 = ma_x$$

$$a_x = 0 \qquad \textbf{(30)}$$

$$\sum F_y = ma_y \rightarrow -mg = ma_y$$

$$a_y = -g \qquad \textbf{(31)}$$

Con estos valores de aceleración se determinan las expresiones generales de velocidad de la pelota en ambas direcciones. De esta forma, la velocidad en x es:

$$v_x = \int a_x dt = \int (0) dt = c_1$$

$$v_x(t=0) = v_{0x} = v_C \cos\theta_2 = c_1$$

$$v_x = v_C \cos\theta_2 \quad \textbf{(32)}$$

La ecuación (32) muestra que la velocidad en dirección x es constante durante todo el movimiento de la partícula. Por su parte, la velocidad en dirección y se define como:

$$v_y = \int a_y dt = \int (-g) dt = -gt + c_2$$

$$v_y(t=0) = v_{0y} = v_C \sin\theta_2 = -g(0) + c_2$$

$$v_{0y} = c_2 = v_C \sin\theta_2$$

$$v_y = -gt + v_C \sin\theta_2 \quad \textbf{(33)}$$

De igual manera se determinan las ecuaciones de posición de la pelota integrando las expresiones respectivas de velocidad. Considerando que la partícula inicia su movimiento en $x = 0$ metros, su función de posición en x es:

$$x = \int v_x dt = \int v_C \cos\theta_2 dt = v_C \cos\theta_2 t + c_3$$

$$x(t=0) = 0 = v_C \cos\theta_2 (0) + c_3 \rightarrow c_3 = 0$$

$$x = v_C \cos\theta_2 t \quad \textbf{(34)}$$

Dado que el movimiento parabólico empieza al final de la rampa 2 (punto C), se debe considerar la altura a la cual se encuentra la pelota en el instante inicial (altura de la rampa 2: h_{r2}). Teniendo en cuenta este valor inicial, la función general de posición en la dirección y se calcula de la siguiente forma:

$$y = \int v_y dt = \int \left(-gt + v_C \sin\theta_2\right) dt = -\frac{1}{2} g t^2 + v_C \sin\theta_2 t + c_4$$

$$\mathrm{y}\left(t = 0\right) = h_{Rampa2} = h_{r2} = -\frac{1}{2} g \left(0\right)^2 + v_C \sin\theta_2 \left(0\right) + c_4 \rightarrow c_4 = h_{r2}$$

$$y = -\frac{1}{2} g t^2 + v_C \sin\theta_2 t + h_{r2} \qquad \textbf{(35)}$$

Con base en las expresiones generales de posición se hallan las máximas distancias alcanzadas por la pelota durante el movimiento parabólico: altura máxima ($h_{máx}$) y alcance horizontal (R). Para encontrar el valor de altura máxima se considera un tiempo $t = t_E$ en el cual la pelota alcanza una posición vertical equivalente a $y = h_{máx}$. En este punto la velocidad en dirección y es cero, por lo tanto:

$$v_y = -g t_E + v_C \sin\theta_2 = 0$$

$$t_E = \frac{v_C \sin\theta_2}{g} \qquad \textbf{(36)}$$

Con el valor de $t = t_E$ la altura $h_{máx}$ equivale a:

$$h_{máx} = -\frac{1}{2} g \left(\frac{v_C \sin\theta_2}{g} \right)^2 + v_C \sin\theta_2 \left(\frac{v_C \sin\theta_2}{g} \right) + h_{r2}$$

$$h_{máx} = -\frac{1}{2} g \left(\frac{{v_C}^2 \sin^2\theta_2}{g^2} \right) + \frac{{v_C}^2 \sin^2\theta_2}{g} + h_{r2}$$

$$h_{máx} = \frac{{v_C}^2 \sin^2\theta_2}{2g} + h_{r2} \qquad \textbf{(37)}$$

Del mismo modo se calcula el alcance horizontal de la pelota. Se utiliza la función de posición en y para encontrar un tiempo $t = t_D$ en el cual la pelota desciende hasta una altura igual a cero ($y = 0$):

$$y = 0 = -\frac{1}{2} g \left(t_D\right)^2 + v_C \sin\theta_2 \left(t_D\right) + h_{r2}$$

$$t_D = \frac{-v_C \sin\theta_2 \pm \sqrt{{v_C}^2 \sin^2\theta_2 + 2gh_{r2}}}{-g}$$

$$t_D = \frac{v_C}{g}\left(\sin\theta_2 + \sqrt{\sin^2\theta_2 + \frac{2gh_{r2}}{{v_C}^2}} \right) \quad \textbf{(38)}$$

Finalmente con el valor de t_D se determina el valor del alcance horizontal ($x = R$), el cual es equivalente a:

$$x = R = v_C \cos\theta_2 t_D$$

$$R = v_C \cos\theta_2 \left[\frac{v_C}{g}\left(\sin\theta_2 + \sqrt{\sin^2\theta_2 + \frac{2gh_{r2}}{{v_C}^2}} \right) \right]$$

$$R = \frac{{v_C}^2}{g}\left(\sin\theta_2 + \sqrt{\sin^2\theta_2 + \frac{2gh_{r2}}{{v_C}^2}} \right)\cos\theta_2 \quad \textbf{(39)}$$

3. PRÁCTICA DE MOVIMIENTO CIRCULAR.

El sistema general para la práctica de movimiento circular se muestra en la figura 7. Las ecuaciones que describen el movimiento del motociclista en los dos tramos que componen este sistema se muestran a continuación:

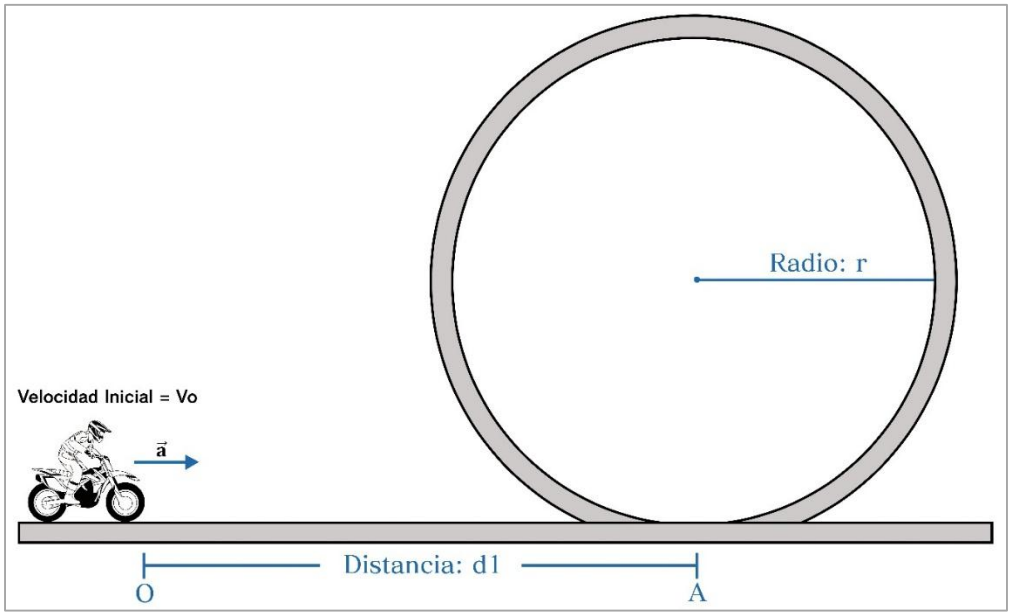


Figura 7. Práctica de movimiento circular – Sistema general.

3.1. SEGMENTO RECTILINEO (TRAMO OA):

La figura 8 muestra el recorrido del motociclista a través del segmento recto de la pista. La velocidad con la que inicia su movimiento es v_0 y mantiene un valor de aceleración constante a_1 durante todo el recorrido por el tramo lineal, de modo que para un tiempo posterior $t = t_A$ alcanza una velocidad final equivalente a $v = v_A$.

Figura 8. Segmento rectilíneo.

Partiendo de estos datos iniciales, la aceleración del motociclista viene expresada como:

$$a = a_1 \tag{40}$$

Integrando la función de aceleración se obtiene la expresión general de velocidad para el tramo O-A:

$$v = \int a dt = \int (a_1) dt = a_1 t + c_1 \tag{41}$$

En el instante $t = t_0$ la velocidad del piloto es v_0, por lo tanto:

$$v(t = 0) = v_0 = a_1(0) + c_1 \rightarrow c_1 = v_0$$

$$v = a_1 t + v_0 \tag{42}$$

De modo similar se integra la expresión de velocidad obtenida para encontrar la función de posición del piloto en dirección x:

$$x = \int v dt = \int (a_1 t + v_0) dt = \frac{a_1 t^2}{2} + v_0 t + c_2$$

En el tiempo $t = t_0$ se tiene que $x = 0$, por lo tanto la ecuación de posición viene dada como:

$$\mathrm{x}(t = 0) = 0 = \frac{a_1 (0)^2}{2} + v_0 (0) + c_2 \rightarrow c_2 = 0$$

$$x = \frac{a_1 t^2}{2} + v_0 t \tag{43}$$

3.2. SEGMENTO CIRCULAR (TRAMO A-A'):

Cuando el piloto ingresa al tramo circular su aceleración total cambia de un punto a otro debido a las variaciones de la velocidad en términos de magnitud y dirección. Para determinar el valor de aceleración total se deben conocer

respectivamente las magnitudes de las aceleraciones centrípeta (a_c) y tangencial (a_t) en cada punto de la circunferencia. Con base en esto se definen una serie de puntos a lo largo del segmento circular con el fin de encontrar dichos valores de aceleración radial (centrípeta) y tangencial, como se muestra en la figura 9, y obtener así una expresión general de la aceleración del piloto dentro de este tramo. Los puntos escogidos para realizar el respectivo análisis son los siguientes: A (0 y 2π), B (punto entre 0 y $\pi/2$), C ($\pi/2$), D (punto entre $\pi/2$ y π), E (π), F (punto entre π y $3\pi/2$), G ($3\pi/2$), H (punto entre $3\pi/2$ y 2π).

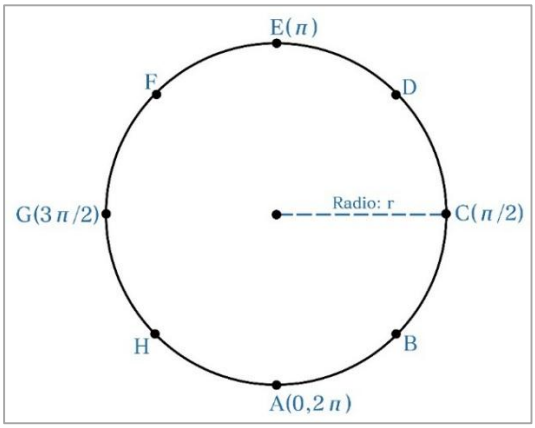


Figura 9. Puntos escogidos dentro del segmento circular.

3.2.1. Aceleración radial y tangencial (Puntos A, C, E y G):

En la figura 10 se muestran las fuerzas que actúan sobre el piloto en su paso por los puntos A (0 y 2π), C ($\pi/2$), E (π) y G ($3\pi/2$) del segmento circular.

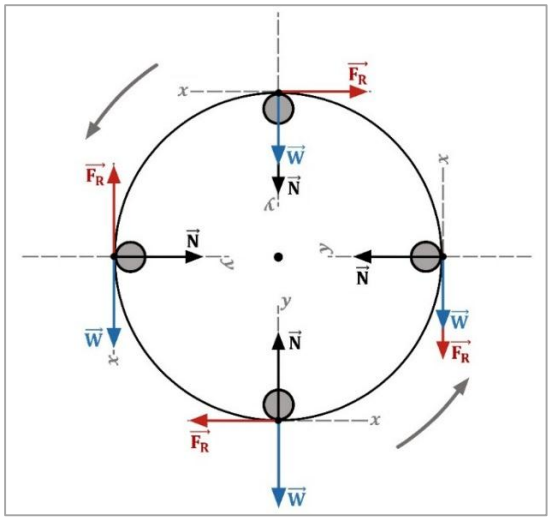


Figura 10. Fuerzas dentro del segmento circular – Parte 1.

Conociendo la dirección de cada fuerza en los puntos designados con base en un sistema de ejes coordenados xy se recurre a la segunda ley de Newton para calcular los valores de aceleración tangencial (a_t) y aceleración radial (a_r) como se muestra a continuación:

- **Punto A (0 y 2π):**

$$\sum F_x = -F_R = ma_t$$

$$-\mu N = ma_t$$

$$a_t = \frac{-\mu N}{m} \qquad \textbf{(44)}$$

$$\sum F_y = N - mg = ma_r$$

$$a_r = \frac{N}{m} - g \qquad \textbf{(45)}$$

- **Punto C ($\pi/2$):**

$$\sum F_x = -mg - F_R = ma_t$$

$$-mg - \mu N = ma_t$$

$$a_t = -g - \frac{\mu N}{m} \qquad \textbf{(46)}$$

$$\sum F_y = N = ma_r$$

$$a_r = \frac{N}{m} \qquad \textbf{(47)}$$

- **Punto E (π):**

$$\sum F_x = -F_R = ma_t$$

$$-\mu N = ma_t$$

$$a_t = \frac{-\mu N}{m} \quad \textbf{(48)}$$

$$\sum F_y = N + mg = ma_r$$

$$a_r = \frac{N}{m} + g \quad \textbf{(49)}$$

- **Punto G ($3\pi/2$):**

$$\sum F_x = -F_R = ma_t$$

$$-\mu N = ma_t$$

$$a_t = \frac{-\mu N}{m} \quad \textbf{(50)}$$

$$\sum F_y = N + mg = ma_r$$

$$a_r = \frac{N}{m} + g \quad \textbf{(51)}$$

3.2.2. Aceleración radial y tangencial (Puntos B, D, F y H):

En la figura 11 se observan las fuerzas que actúan sobre el piloto dentro del segmento circular en los puntos B (entre *0* y *π/2*), D (entre *π/2* y *π*), F (entre *π* y *3π/2*) y H (entre *3π/2* y *2π*).

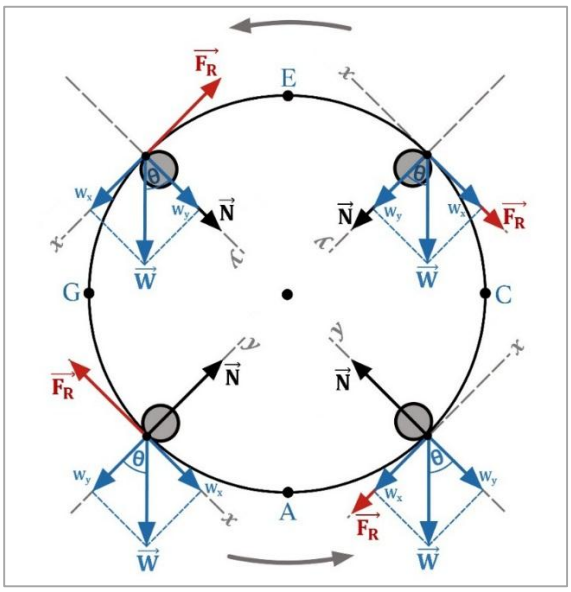


Figura 11. Fuerzas dentro del segmento circular – Parte 2.

De manera similar al caso anterior se determinan los valores de aceleración tangencial (a_t) y aceleración radial (a_r) en los puntos designados con base en la segunda ley de Newton:

- **Punto B (entre *0* y *π/2*):**

$$\sum F_x = -W_x - F_R = ma_t$$

$$-mg\sin\theta - F_R = -mg\sin\theta - \mu N = ma_t$$

$$a_t = -g\sin\theta - \frac{\mu N}{m} \qquad \textbf{(52)}$$

$$\sum F_y = N - W_y = ma_r$$

$$N - mg\cos\theta = ma_r$$

$$a_r = \frac{N}{m} - g\cos\theta \qquad \textbf{(53)}$$

- **Punto D (entre *π/2* y *π*):**

$$\sum F_x = -W_x - F_R = ma_t$$

$$-mg\sin\theta - F_R = -mg\sin\theta - \mu N = ma_t$$

$$a_t = -g\sin\theta - \frac{\mu N}{m} \quad \textbf{(54)}$$

$$\sum F_y = N + W_y = ma_r$$

$$N + mg\cos\theta = ma_r$$

$$a_r = \frac{N}{m} + g\cos\theta \quad \textbf{(55)}$$

- **Punto F (entre *π* y *3π/2*):**

$$\sum F_x = W_x - F_R = ma_t$$

$$mg\sin\theta - F_R = mg\sin\theta - \mu N = ma_t$$

$$a_t = g\sin\theta - \frac{\mu N}{m} \quad \textbf{(56)}$$

$$\sum F_y = N + W_y = ma_r$$

$$N + mg\cos\theta = ma_r$$

$$a_r = \frac{N}{m} + g\cos\theta \quad \textbf{(57)}$$

- **Punto H (entre 3π/2 y 2π):**

$$\sum F_x = W_x - F_R = ma_t$$

$$mg\sin\theta - F_R = mg\sin\theta - \mu N = ma_t$$

$$a_t = g\sin\theta - \frac{\mu N}{m} \quad \textbf{(58)}$$

$$\sum F_y = N - W_y = ma_r$$

$$N - mg\cos\theta = ma_r$$

$$a_r = \frac{N}{m} - g\cos\theta \quad \textbf{(59)}$$

De acuerdo a los valores de aceleración tangencial (a_t) y aceleración radial (a_r) encontrados en cada uno de los puntos de la circunferencia se deducen las expresiones generales para dichas aceleraciones. Estas funciones son las siguientes:

$$a_t = -g\sin\theta - \frac{\mu N}{m} \quad \textbf{(60)}$$

$$a_r = \frac{N}{m} - g\cos\theta \quad \textbf{(61)}$$

Sin embargo, considerando que el motociclista inicia su movimiento circular en $\theta = -\pi/2$, las ecuaciones generales de aceleración tangencial y radial quedan definidas de la siguiente manera:

$$a_t = -g\sin\left(\theta - \frac{\pi}{2}\right) - \frac{\mu N}{m} \quad \textbf{(62)}$$

$$a_r = \frac{N}{m} - g\cos\left(\theta - \frac{\pi}{2}\right) \quad \textbf{(63)}$$

La ecuación general de aceleración total viene dada como:

$$a = \sqrt{{a_t}^2 + {a_r}^2} \tag{64}$$

Se relacionan las ecuaciones de velocidad y posición de una partícula dentro del movimiento unidimensional con las expresiones de aceleración definidas para el movimiento del motociclista a través del trayecto circular de la siguiente forma:

$$v = v_0 + at \rightarrow v = v_0 + a_t t \tag{65}$$

$$x = v_0 t + \frac{at^2}{2} \rightarrow x = v_0 t + \frac{a_t t^2}{2} \tag{66}$$

Se reescribe el término de aceleración tangencial (a_t) en función de la velocidad del piloto y del ángulo recorrido dentro de la circunferencia. Por lo tanto, de la ecuación (63) se despeja la fuerza normal (N) y se sustituye el término de aceleración radial (centrípeta) por v^2/r, de acuerdo con la teoría del movimiento circular:

$$a_r = \frac{N}{m} - g\cos\left(\theta - \frac{\pi}{2}\right)$$

$$N = m\left[a_r + g\cos\left(\theta - \frac{\pi}{2}\right)\right]$$

$$N = m\left[\frac{v^2}{r} + g\cos\left(\theta - \frac{\pi}{2}\right)\right] \tag{67}$$

Reemplazando el valor de la fuerza normal (N) en la ecuación (62) se llega a la nueva expresión de aceleración tangencial (a_t):

$$a_t = -g\sin\left(\theta - \frac{\pi}{2}\right) - \frac{\mu N}{m}$$

$$a_t = -g\sin\left(\theta - \frac{\pi}{2}\right) - \frac{\mu}{m}\left[\frac{v^2}{r} + g\cos\left(\theta - \frac{\pi}{2}\right)\right]m$$

$$a_t = -g\sin\left(\theta - \frac{\pi}{2}\right) - \mu\left[\frac{v^2}{r} + g\cos\left(\theta - \frac{\pi}{2}\right)\right] \quad \textbf{(68)}$$

Ahora bien, tomando x como la longitud del arco recorrido por el motociclista dentro del segmento circular se tiene que:

$$x = \theta_{rad} r = v_0 t + \frac{a_t t^2}{2} \quad \textbf{(69)}$$

Se obtiene el valor del tiempo empleado por el piloto en recorrer la longitud de arco dada por la ecuación (69):

$$\frac{a_t}{2}(t)^2 + v_0(t) - \theta_{rad} r = 0$$

$$t = \frac{-v_0 \pm \sqrt{{v_0}^2 + 2a_t\theta_{rad} r}}{a_t}$$

$$t = \frac{-v_0 + \sqrt{{v_0}^2 + 2a_t\theta_{rad} r}}{a_t} \quad \textbf{(70)}$$

Reemplazando este valor de tiempo en la ecuación (65) se tiene la expresión general de velocidad del piloto dentro del tramo circular en función del ángulo recorrido, del coeficiente de fricción de la superficie de contacto y de su velocidad inicial:

$$v = v_0 + a_t t = v_0 + a_t\left(\frac{-v_0 + \sqrt{{v_0}^2 + 2a_t\theta_{rad} r}}{a_t}\right)$$

$$v = v_0 - v_0 + \sqrt{{v_0}^2 + 2a_t\theta_{rad} r} = \sqrt{{v_0}^2 + 2a_t\theta_{rad} r}$$

$$v^2 = {v_0}^2 + 2\theta_{rad} r\left\{-g\sin\left(\theta - \frac{\pi}{2}\right) - \mu\left[\frac{v^2}{r} + g\cos\left(\theta - \frac{\pi}{2}\right)\right]\right\}$$

$$v^2 = {v_0}^2 - 2\theta_{rad} rg \sin\left(\theta - \frac{\pi}{2}\right) - 2\theta_{rad}\mu v^2 - 2\theta_{rad}\mu rg\cos\left(\theta - \frac{\pi}{2}\right)$$

$$v^2 + 2\theta_{rad}\mu v^2 = {v_0}^2 - 2\theta_{rad} rg \sin\left(\theta - \frac{\pi}{2}\right) - 2\theta_{rad}\mu rg\cos\left(\theta - \frac{\pi}{2}\right)$$

$$v^2\left(1 + 2\theta_{rad}\mu\right) = {v_0}^2 - 2\theta_{rad} rg \sin\left(\theta - \frac{\pi}{2}\right) - 2\theta_{rad}\mu rg\cos\left(\theta - \frac{\pi}{2}\right)$$

$$v = \sqrt{\frac{{v_0}^2 - 2\theta_{rad} rg \sin\left(\theta - \frac{\pi}{2}\right) - 2\theta_{rad}\mu rg\cos\left(\theta - \frac{\pi}{2}\right)}{1 + 2\theta_{rad}\mu}} \qquad \textbf{(71)}$$

Finalmente, se determina si la velocidad inicial con que el motociclista entra al segmento circular es suficiente para no caer en su intento de lograr un giro completo. Igualando a cero la función de velocidad y despejando el término de velocidad inicial (v_{01}) se conoce el valor mínimo de dicha velocidad inicial:

$$v = 0$$

$${v_{01}}^2 - 2\theta_{rad} rg \sin\left(\theta - \frac{\pi}{2}\right) - 2\theta_{rad}\mu rg\cos\left(\theta - \frac{\pi}{2}\right) = 0$$

$${v_{01}}^2 = 2\theta_{rad} rg \sin\left(\theta - \frac{\pi}{2}\right) + 2\theta_{rad}\mu rg\cos\left(\theta - \frac{\pi}{2}\right)$$

$$v_{01} = \sqrt{2\theta_{rad} rg \sin\left(\theta - \frac{\pi}{2}\right) + 2\theta_{rad}\mu rg\cos\left(\theta - \frac{\pi}{2}\right)}$$

De esta manera la velocidad mínima con la que el piloto debe llegar a la rampa circular para dar un giro completo es:

$$v_{01} = v_A \geq \sqrt{2\theta_{rad} rg \sin\left(\theta - \frac{\pi}{2}\right) + 2\theta_{rad}\mu rg\cos\left(\theta - \frac{\pi}{2}\right)} \qquad \textbf{(72)}$$

De la ecuación (72) se obtiene la expresión que determina el ángulo θ recorrido por el piloto dentro de la circunferencia en función de la velocidad inicial con que inicia su movimiento dentro del tramo circular.

$$\left[\sin\left(\theta-\frac{\pi}{2}\right)+\mu\cos\left(\theta-\frac{\pi}{2}\right)\right]\theta_{rad}\leq\frac{v_{01}{}^{2}}{2rg} \qquad \textbf{(73)}$$

4. PRACTICA DE LEYES DEL MOVIMIENTO

La figura 12 muestra el esquema diseñado para la práctica de leyes del movimiento. Las ecuaciones que definen el movimiento del bloque a través de la rampa inclinada (con y sin resorte) y dentro de la trayectoria parabólica se muestran a continuación:

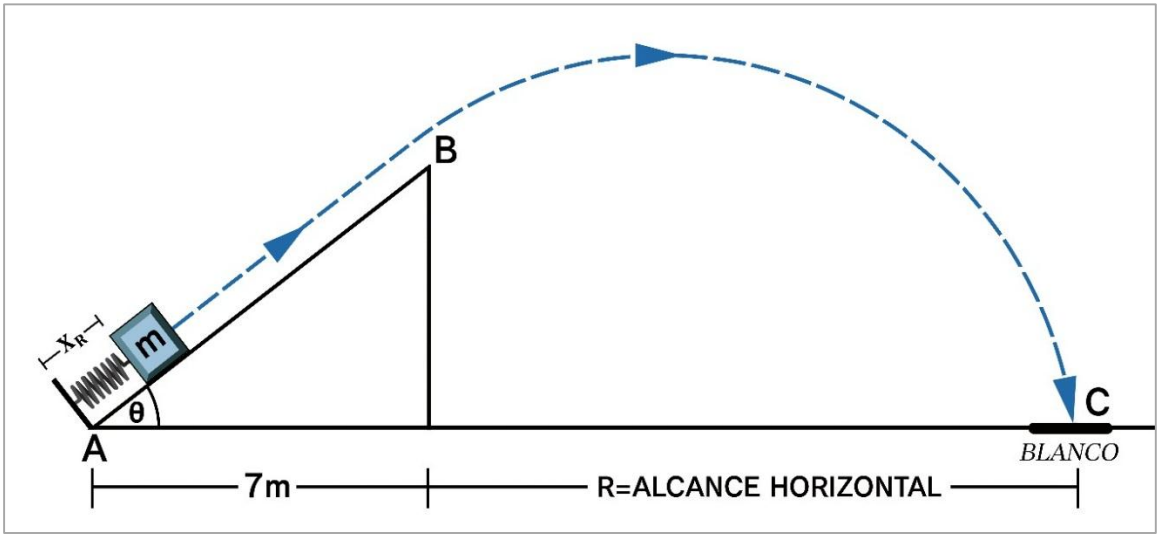


Figura 12. Leyes del movimiento – Sistema general.

4.1. Sistema masa-resorte (Tramo A-A'):

La figura 13 muestra el desplazamiento del sistema compuesto por la masa y el resorte a través de la rampa inclinada.

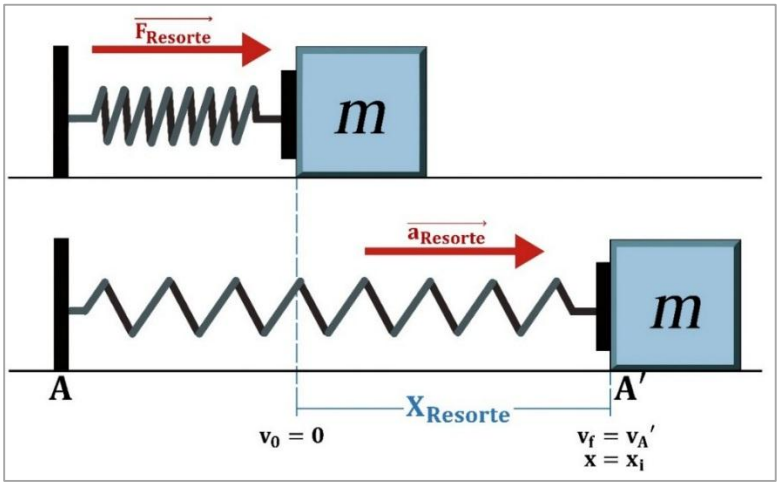


Figura 13. Longitud de compresión del resorte.

Dentro del plano inclinado se debe considerar en primera instancia el movimiento del bloque unido al resorte (tramo A-A') como resultado de la fuerza que este genera sobre el cuerpo debido al proceso de compresión inicial. La longitud que recorren ambos es igual a la distancia de compresión del resorte ($X_{Resorte}$), ya que al llegar a la posición de reposo del muelle ($x = x_i$) el bloque se separa y continúa su movimiento solo.

Al momento de ejercer la fuerza de compresión y alcanzar la longitud deseada ($X_{Resorte}$) el bloque tiene una velocidad inicial $v_0 = 0m/s$ (punto A). Cuando esta fuerza desaparece inicia el movimiento del sistema masa-resorte con un valor de aceleración dada por la fuerza con que el muelle empuja al cuerpo, como se muestra en la figura 13. Al llegar al punto de reposo (punto A') el bloque alcanza una velocidad $v = v_A{}'$ dada por la siguiente expresión:

$$v_f{}^2 - v_0{}^2 = 2as$$

$$v_f{}^2 - v_0{}^2 = 2a_{\mathrm{Re}sorte}x_{\mathrm{Re}sorte}$$

$$v_f{}^2 = 2a_{\mathrm{Re}sorte}x_{\mathrm{Re}sorte}$$

$$v_f = v_{A'} = \sqrt{2a_{\mathrm{Re}sorte}x_{\mathrm{Re}sorte}} \qquad \textbf{(74)}$$

La aceleración en el movimiento del sistema masa-resorte se determina con base en las fuerzas que actúan sobre el bloque dentro del tramo A-A'. De acuerdo a la

figura 14 las fuerzas presentes son la fuerza normal (N), la fuerza de fricción de la superficie (F_R), el peso del bloque (P) y la fuerza del resorte ($F_{Resorte}$).

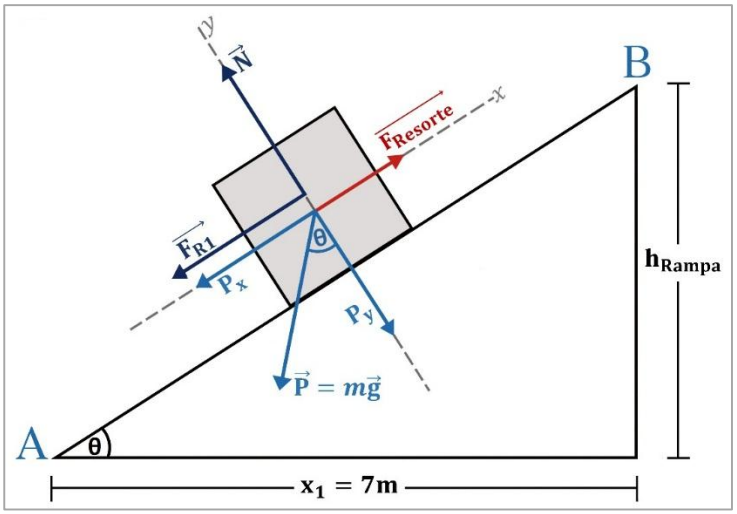


Figura 14. Fuerzas presentes en el sistema masa-resorte.

Las componentes del peso en dirección x y y respectivamente son:

$$P_x = mg\sin\theta$$

$$P_y = mg\cos\theta$$

Con base en la segunda ley de Newton se hallan las expresiones generales de fuerza normal (N) y aceleración ($a_{Resorte}$) dentro del movimiento del sistema masa-resorte. En la dirección y se tiene:

$$\sum F_y = N - mg\cos\theta = 0$$

$$N = mg\cos\theta \qquad \textbf{(75)}$$

Mientras que en la dirección x:

$$\sum F_x = F_{\mathrm{Re}sorte} - P_x - F_R = ma_{\mathrm{Re}sorte}$$

$$kx_{\mathrm{Re}sorte} - mg\sin\theta - \mu mg\cos\theta = ma_{\mathrm{Re}sorte}$$

$$a_{\text{Resorte}} = \frac{kx_{\text{Resorte}} - mg\left(\sin\theta + \mu\cos\theta\right)}{m}$$

$$a_{\text{Resorte}} = \frac{kx_{\text{Resorte}}}{m} - g\left(\sin\theta + \mu\cos\theta\right) \quad \textbf{(76)}$$

4.2. Movimiento del bloque sobre el plano inclinado: Sin resorte (Tramo A'-B):

Dependiendo de la distancia de compresión del resorte el bloque continúa su movimiento desde la posición de reposo del muelle (punto A') hasta el final del plano inclinado (punto B), como se muestra en la figura 15. Se considera que el recorrido total dentro de la rampa es igual a L (despreciando la longitud que el resorte en reposo le quita al plano inclinado), por lo tanto dado que en el movimiento del bloque unido al resorte se recorrió una distancia igual a la longitud de compresión del muelle ($X_{Resorte}$), para este caso el recorrido equivale a $L - X_{Resorte}$.

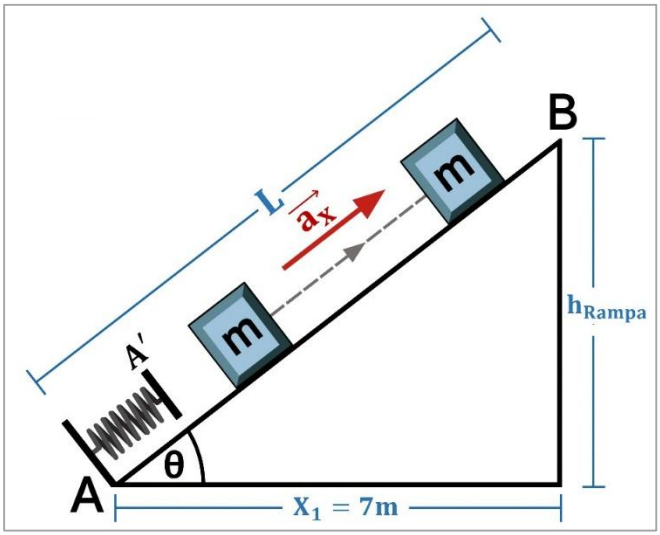


Figura 15. Movimiento del bloque sobre el plano inclinado

Conociendo el valor de la base de la rampa ($x_1 = 7m$) se calcula la medida de su altura y la longitud del segmento AB de la siguiente forma:

$$h_{Rampa} = x_1 \tan\theta \quad \textbf{(77)}$$

$$L = \frac{x_1}{\cos\theta} \tag{78}$$

De acuerdo a la figura 15 el bloque se mueve dentro del tramo A'-B con un valor de aceleración $a = a_x$. Esta aceleración está definida por las mismas fuerzas presentes en el sistema anterior (sistema masa-resorte), exceptuando la fuerza del resorte ($F_{Resorte}$), la cual no tiene incidencia dentro de este movimiento, como se muestra en la figura 16.

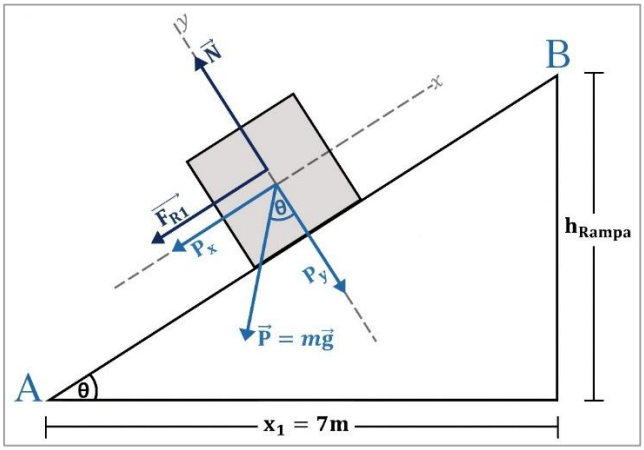


Figura 16. Fuerzas presentes en el bloque (plano inclinado).

Partiendo de la segunda ley de Newton y considerando que no hay movimiento en dirección y dentro del sistema coordenado escogido, la aceleración $a = a_x$ equivale a:

$$\sum F_x = -P_x - F_R = ma_x$$

$$-mg\sin\theta - \mu mg\cos\theta = ma_x$$

$$a_x = \frac{-mg(\sin\theta + \mu\cos\theta)}{m}$$

$$a_x = -g(\sin\theta + \mu\cos\theta) \tag{79}$$

Se considera que la velocidad inicial del bloque para este movimiento equivale a su velocidad final dentro del tramo A-A', es decir:

$$v_0 = v_{A'} = \sqrt{2a_{Resorte}x_{Resorte}} \quad \textbf{(80)}$$

Una vez calculado este valor se obtiene la ecuación de velocidad del bloque en el punto B:

$$v_f{}^2 - v_0{}^2 = 2as$$

$$v_f{}^2 - v_{A'}{}^2 = 2a_x\left(L - x_{Resorte}\right)$$

$$v_f{}^2 = 2a_x\left(L - x_{Resorte}\right) + v_{A'}{}^2$$

$$v_f = v_B = \sqrt{2a_x\left(L - x_{Resorte}\right) + 2a_{Resorte}x_{Resorte}} \quad \textbf{(81)}$$

A partir de la ecuación (81) se determina el valor de constante elástica del resorte (K) necesaria para que el bloque recorra toda la longitud de la rampa:

$$2a_x\left(L - x_{Resorte}\right) + 2a_{Resorte}x_{Resorte} > 0$$

Reemplazando la expresión de aceleración se encuentra el valor de la constante K requerida:

$$-2g\left(\sin\theta + \mu\cos\theta\right)\left(L - x_{Resorte}\right)$$
$$+2x_{Resorte}\left[\frac{kx_{Resorte}}{m} - g\left(\sin\theta + \mu\cos\theta\right)\right] > 0$$

$$-2g\left(\sin\theta + \mu\cos\theta\right)L + 2g\left(\sin\theta + \mu\cos\theta\right)x_{Resorte} + \frac{2kx^2{}_{Resorte}}{m}$$
$$-2x_{Resorte}g\left(\sin\theta + \mu\cos\theta\right) > 0$$

$$-2g\left(\sin\theta + \mu\cos\theta\right)L + \frac{2kx^2{}_{Resorte}}{m} > 0$$

$$\frac{2kx^2_{Resorte}}{m} > 2g(\sin\theta + \mu\cos\theta)L$$

$$k > \frac{2gmL(\sin\theta + \mu\cos\theta)}{2x^2_{Resorte}} \quad \textbf{(82)}$$

4.3. Movimiento parabólico del bloque (Tramo B-C):

Cuando la distancia de compresión del resorte para un valor de K definido es la correcta se logra que el bloque atraviese por completo el plano inclinado, llegando incluso más allá del límite de la rampa. En este caso el cuerpo realiza un movimiento parabólico debido a la inclinación del plano por el que acaba de desplazarse, como lo muestra la figura 17.

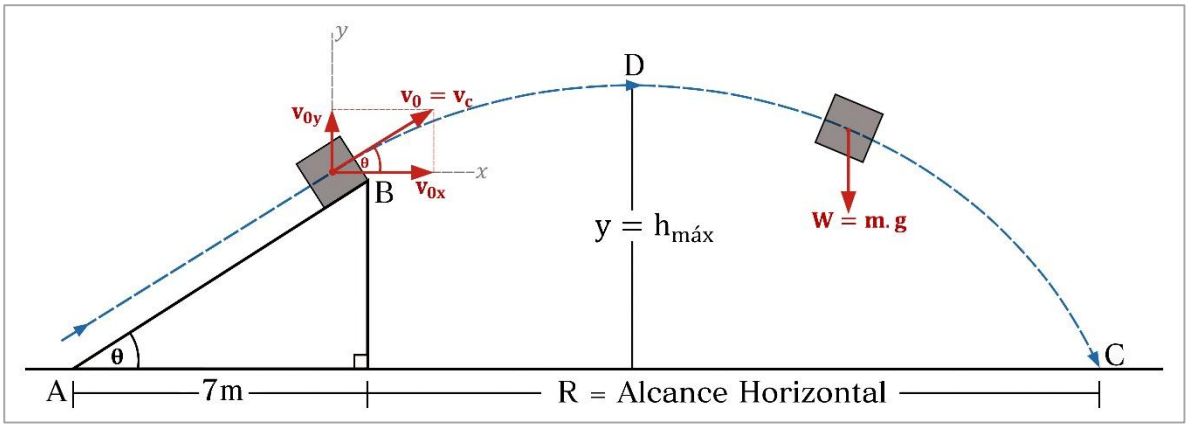


Figura 17. Movimiento parabólico.

Dado que este tipo de movimiento ya fue estudiado en la sección 2.4 se deducen las mismas expresiones generales para los diferentes parámetros del subsistema. Por lo tanto dichas ecuaciones son las siguientes:

Componentes de velocidad en x y y al inicio del movimiento parabólico:

$$v_{0x} = v_0\cos\theta = v_B\cos\theta \quad \textbf{(83)}$$

$$v_{0y} = v_0\sin\theta = v_B\sin\theta \quad \textbf{(84)}$$

Componentes de aceleración en x y y:

$$a_x = 0 \quad \textbf{(85)}$$

$$a_y = -g \tag{86}$$

Componentes de velocidad en dirección x y y respectivamente:

$$v_x = v_B \cos\theta \tag{87}$$

$$v_y = -gt + v_B \sin\theta \tag{88}$$

Ecuaciones de posición en x y y del bloque durante el movimiento parabólico:

$$x = v_B \cos\theta t \tag{89}$$

$$y = -\frac{1}{2} g t^2 + v_B \sin\theta t + h_{rampa} \tag{90}$$

Altura máxima ($h_{máx}$) alcanzada por el bloque dentro del movimiento parabólico y el tiempo (t_D) empleado en llegar hasta dicha posición:

$$t_D = \frac{v_B \sin\theta}{g} \tag{89}$$

$$h_{máx} = \frac{{v_B}^2 \sin^2\theta}{2g} + h_{rampa} \tag{90}$$

Alcance horizontal (R) y el tiempo (t_C) empleado por el bloque para llegar a tal posición:

$$t_C = \frac{v_B}{g}\left(\sin\theta + \sqrt{\sin^2\theta + \frac{2gh_{rampa}}{{v_B}^2}}\right) \tag{91}$$

$$R = \frac{{v_B}^2}{g}\left(\sin\theta + \sqrt{\sin^2\theta + \frac{2gh_{rampa}}{{v_B}^2}}\right)\cos\theta \tag{92}$$

5. PRACTICA N° DE TRABAJO Y ENERGIA.

El sistema a través del cual el obrero realiza la traslación de las cajas por los diferentes caminos hasta cada uno de los camiones se muestra en la figura 18. Considerando tramo a tramo su desplazamiento por los diferentes puntos del recorrido, se obtienen las siguientes las ecuaciones de movimiento del sistema:

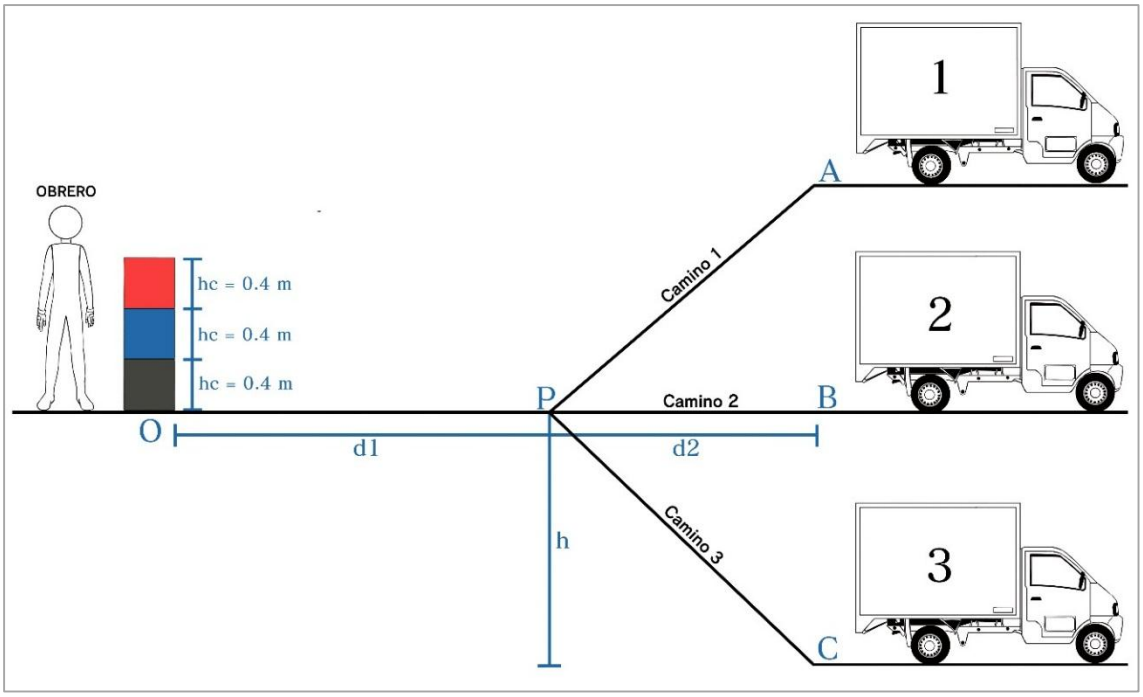


Figura 18. Práctica de trabajo y energía – Sistema general.

5.1. Trabajo del obrero (Tramo O-P):

De acuerdo a la figura 19 se determina la expresión general del trabajo que realiza el obrero levantando las cajas en el punto de inicio del sistema, considerando que la fuerza que se aplica sobre ellas es equivalente al peso de cada una ($F = mg$) y la altura a la cual son levantadas depende de la posición inicial en que están organizadas.

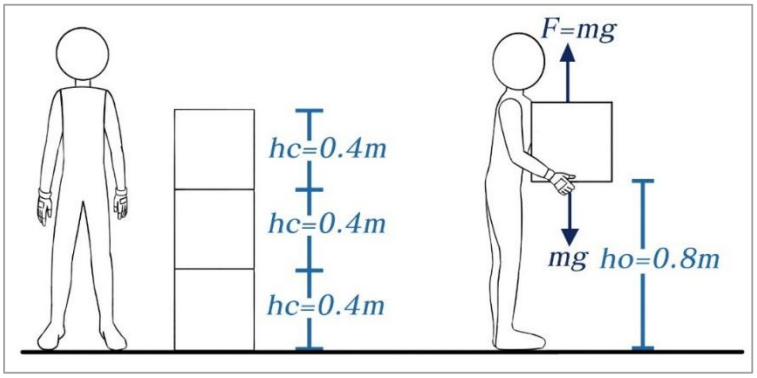


Figura 19. Punto O – Levantar cajas.

Dado que la fuerza aplicada sobre cada caja está en la misma dirección del desplazamiento ($\theta = 0°$) se tiene que:

$$W_F = mgh = mg(0) = 0$$

$$W_F = mgh \quad \textbf{(93)}$$

De esta manera el trabajo hecho por el obrero levantando la caja azul (ubicada a 0.8 metros del suelo), la caja roja (ubicada a 0.4 metros del suelo) y la caja negra (ubicada al nivel del suelo) respectivamente viene dado por las siguientes expresiones:

$$W_F = mgh = mg(0) = 0 \quad \textbf{(94)}$$

$$W_F = mgh = mg(0.4) = 0.4mg \quad \textbf{(95)}$$

$$W_F = mgh = mg(0.8) = 0.8mg \quad \textbf{(96)}$$

Independientemente del camino escogido, el obrero debe recorrer siempre el tramo O-P, y dado que la dirección de aplicación de la fuerza para sostener las cajas es perpendicular a su desplazamiento, se mantiene el valor inicial de trabajo determinado en el punto de partida. Ahora bien, si se escogen las opciones de recorrer el camino 1 o el camino 3 el obrero llega al punto P y desciende la caja hasta el nivel del suelo, como se muestra en la figura 20.

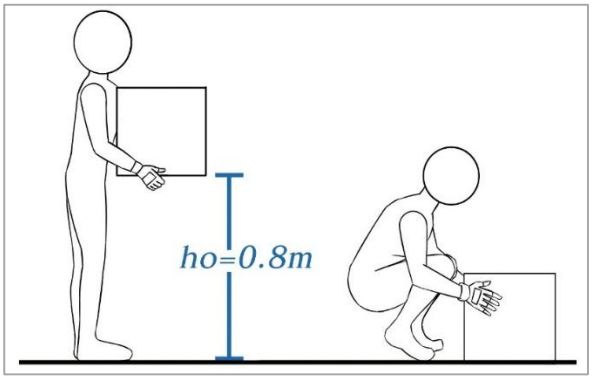


Figura 20. Punto P – Bajar caja.

En este punto el trabajo llevado a cabo por el obrero conserva la misma magnitud que el realizado en el punto de partida (punto O) pero con signo opuesto debido a que en este caso la dirección en que se aplica la fuerza es opuesta a la dirección del desplazamiento de las cajas.

5.2. Trabajo del obrero (TRAMO P-A):

El trabajo realizado por el obrero para llevar la caja del punto P hasta el punto A (plano inclinado ascendente) se determina con base en el esquema de la figura 21, en la cual se observan las diferentes fuerzas que actúan sobre el cuerpo a medida que es arrastrado por la rampa empinada.

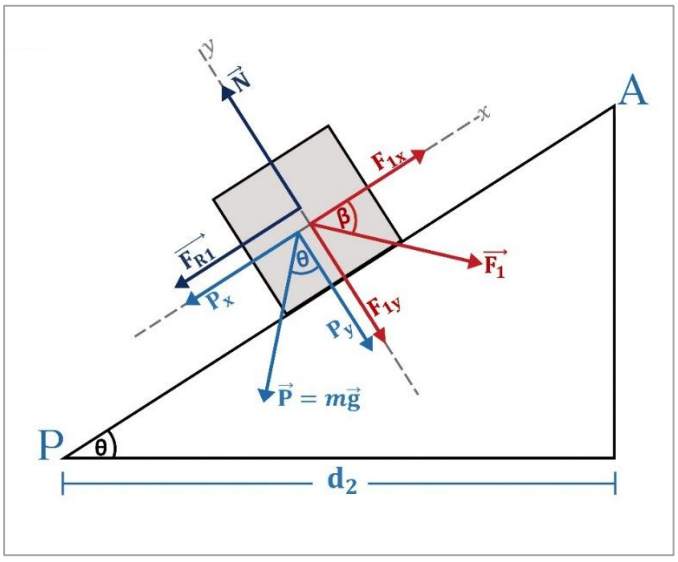


Figura 21. Fuerzas presentes en la caja - Plano inclinado ascendente.

En primer lugar se determina la longitud del plano inclinado (L_1), cuyo valor depende del ángulo de elevación θ ingresado:

$$\cos\theta_1 = \frac{d_2}{L_1} \rightarrow L_1 = \frac{d_2}{\cos\theta_1} \quad \textbf{(197)}$$

Para subir cualquiera de las cajas a través de la rampa inclinada el obrero aplica una fuerza F_1 en dirección $\beta = -40°$ respecto del eje x del sistema coordenado escogido. El valor de esta fuerza depende de los datos iniciales correspondientes al ángulo de inclinación de la rampa (θ), el coeficiente de rozamiento de su superficie (μ_1) y la masa de las cajas (m). Con base en el diagrama del cuerpo libre de la figura 21 las componentes rectangulares de la fuerza F_1 se expresan como:

$$F_{1x} = F_1 \cos\beta \quad \textbf{(98)}$$

$$F_{1y} = F_1 \sin\beta \quad \textbf{(99)}$$

Por su parte, la fuerza que representa al peso de las cajas (P) se descompone de la siguiente forma:

$$P_x = mg\sin\theta_1 \quad \textbf{(100)}$$

$$P_y = mg\cos\theta_1 \quad \textbf{(101)}$$

Aplicando la segunda ley de Newton se determina la expresión que representa la fuerza de rozamiento de la superficie de la rampa 1(F_{R1}):

$$\sum F_y = N - P_y - F_{1y} = 0$$

$$N = P_y + F_{1y} \quad \textbf{(102)}$$

$$F_{R1} = \mu_1 N = \mu_1 \left(P_y + F_{1y}\right) \quad \textbf{(103)}$$

Una vez definidas las fuerzas que actúan sobre las cajas dentro del plano inclinado ascendente se determina el trabajo individual que aporta cada una de ellas. De esta manera, el trabajo respectivo de la fuerza aplicada a la caja (componentes de F_1), del peso (componentes de P) y de la fuerza de rozamiento (F_{R1}) se definen de la siguiente forma:

$$W_{F_{1x}} = \left(F_1 \cos\beta\right) L_1 \cos\alpha = \left(F_1 \cos\beta\right) L_1 \cos\left(0\right) = \left(F_1 \cos\beta\right) L_1 \quad \textbf{(104)}$$

$$W_{F_{1y}} = \left(F_1 \sin\beta\right) L_1 \cos\alpha = \left(F_1 \sin\beta\right) L_1 \cos\left(-90\right) = 0 \quad \textbf{(105)}$$

$$W_{P_x} = \left(mg \sin\theta_1\right) L_1 \cos\alpha = \left(mg \sin\theta_1\right) L_1 \cos\left(180\right) = -\left(mg \sin\theta_1\right) L_1 \quad \textbf{(106)}$$

$$W_{P_y} = \left(mg \cos\theta_1\right) L_1 \cos\alpha = \left(mg \cos\theta_1\right) L_1 \cos\left(-90\right) = 0 \quad \textbf{(107)}$$

$$W_{F_{R1}} = \mu_1 \left(P_y + F_{1y}\right) L_1 \cos\alpha = \mu_1 \left(P_y + F_{1y}\right) L_1 \cos\left(180\right)$$

$$W_{F_{R1}} = -\mu_1 \left(P_y + F_{1y}\right) L_1 \quad \textbf{(109)}$$

El trabajo neto compuesto por los trabajos individuales de cada una de las fuerzas que intervienen en este tramo del sistema se expresa de la siguiente forma:

$$W_{Neto} = W_{F_{1x}} + W_{P_x} + W_{F_{R1}}$$

$$W_{Neto} = \left(F_1 \cos\beta\right) L_1 - \left(mg \sin\theta_1\right) L_1 - \mu_1 \left(P_y + F_{1y}\right) L_1$$

$$W_{Neto} = L_1 \left[F_1 \cos\beta - mg \sin\theta_1 - \mu_1 \left(P_y + F_{1y}\right) \right] \quad \textbf{(110)}$$

Se calcula la velocidad con que la caja llega al final del plano inclinado. Para ello se recurre a los valores de energía cinética (K) de la caja en los puntos P y A:

$$K_i = 0 \quad \textbf{(111)}$$

$$K_f = \frac{1}{2} m v_f^{\,2} \quad \textbf{(112)}$$

Se expresa el trabajo neto hallado en la ecuación (110) como la diferencia entre la energía cinética final (K_f) y la energía cinética inicial (K_i) (teorema del trabajo y la energía). De esta manera la velocidad final de la caja en el punto A ($v_f = v_A$) es:

$$W_{Neto} = K_f - K_i = \frac{1}{2} m v_f^{\,2}$$

$$L_1\left[F_1\cos\beta - mg\sin\theta_1 - \mu_1\left(P_y + F_{1y}\right)\right] = \frac{1}{2}m{v_f}^2$$

$${v_f}^2 = \frac{2L_1}{m}\left[F_1\cos\beta - mg\sin\theta_1 - \mu_1\left(P_y + F_{1y}\right)\right]$$

$$v_f = v_A = \sqrt{\frac{2L_1}{m}\left[F_1\cos\beta - mg\sin\theta_1 - \mu_1\left(P_y + F_{1y}\right)\right]} \qquad \textbf{(113)}$$

Con el valor de velocidad calculado se determina la aceleración con que la caja realiza su movimiento ascendente a través del plano inclinado. Considerando que al inicio de la rampa la caja parte con una velocidad $v_i = 0m/s$ y que la distancia total recorrida es $s = L_1$, se utiliza la ecuación de velocidad en función del desplazamiento del movimiento unidimensional para encontrar el valor de dicha aceleración:

$${v_f}^2 - {v_i}^2 = 2as$$

$${v_f}^2 = {v_A}^2 = 2aL_1$$

$$a = \frac{{v_A}^2}{2L_1} \qquad \textbf{(114)}$$

Reemplazando el valor de velocidad final en la ecuación () se tiene:

$$a = \frac{2L_1\left[F_1\cos\beta - mg\sin\theta_1 - \mu_1\left(P_y + F_{1y}\right)\right]}{2L_1 m}$$

$$a = \frac{F_1\cos\beta - mg\sin\theta_1 - \mu_1 mg\cos\theta_1 - \mu_1 F_1\sin\beta}{m}$$

$$a = \frac{F_1\cos\beta - \mu_1 F_1\sin\beta}{m} - g\sin\theta_1 - \mu_1 g\cos\theta_1 \qquad \textbf{(115)}$$

Para desplazar las cajas a través del plano inclinado el obrero debe aplicar una fuerza F_1 lo suficientemente grande como para sobrepasar la magnitud de las dos

fuerzas que se oponen al movimiento de las mismas (P_x y F_{R1}). Partiendo de esta premisa se deduce el valor de F_1 de la siguiente forma:

$$F_1 > P_x + F_{R1}$$

$$F_1 \cos\beta > mg \sin\theta_1 + \mu_1 P_y + \mu_1 F_{1y}$$

$$F_1 \cos\beta > mg \sin\theta_1 + \mu_1 mg \cos\theta_1 + \mu_1 F_1 \sin\beta$$

$$F_1 \cos\beta > mg\left(\sin\theta_1 + \mu_1 \cos\theta_1\right) + \mu_1 F_1 \sin\beta$$

$$F_1 \cos\beta - \mu_1 F_1 \sin\beta > mg\left(\sin\theta_1 + \mu_1 \cos\theta_1\right)$$

$$F_1\left(\cos\beta - \mu_1 \sin\beta\right) > mg\left(\sin\theta_1 + \mu_1 \cos\theta_1\right)$$

$$F_1 > \frac{mg\left(\sin\theta_1 + \mu_1 \cos\theta_1\right)}{\left(\cos\beta - \mu_1 \sin\beta\right)} \qquad \textbf{(116)}$$

Al llegar al final de la rampa 1 el obrero debe recoger la caja y llevarla desde su posición del suelo hasta un punto del camión ubicado a 0.8 metros del mismo. El trabajo realizado es igual al del punto inicial (punto O) cuando se escoge llevar la caja negra. Otro caso similar se presenta cuando se escoge la opción de transitar el camino 3, ya que al finalizar el recorrido por la rampa 2 la caja debe ser levantada y depositada en el camión, como se observa en la figura 21.

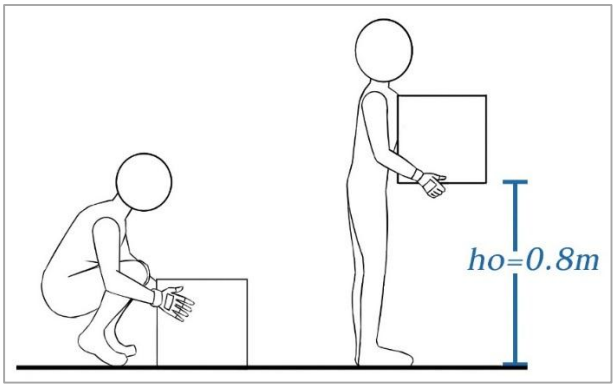


Figura 22. Punto A – Depositar la caja dentro del camión.

5.3. Trabajo realizado en el tramo P-C:

La figura 23 muestra el tramo final del camino C, el cual corresponde a una rampa inclinada de tipo descendente:

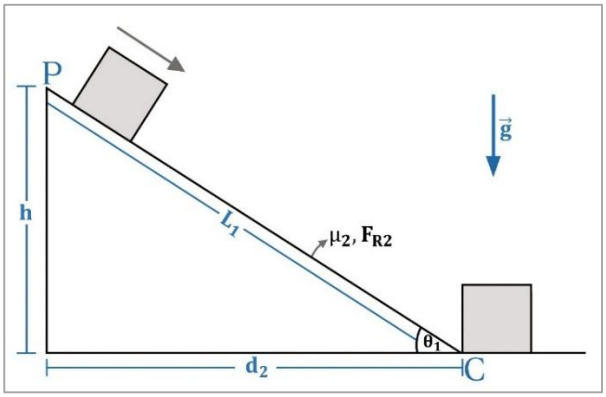


Figura 23. Camino 2 – Plano inclinado descendente.

Inicialmente se calcula la longitud del plano inclinado descendente (L_2) por el cual se desplaza la caja bajo la acción de la gravedad:

$$L_2 = \frac{h}{\sin\theta_2} \tag{117}$$

En el punto P la caja se encuentra en el lugar más alto de la rampa inclinada ($y = h$) y su movimiento descendente a partir de esta posición lo realiza desde el reposo ($v_0 = 0m/s$). Partiendo de estos datos se calculan los valores iniciales de energía cinética y de energía potencial en la parte superior del plano descendente de la siguiente forma:

$$K_i = \frac{1}{2}mv_0^{\,2} = 0 \tag{118}$$

$$U_i = mgy_i = mgh \tag{119}$$

Por su parte, en el inferior de la rampa 2 (punto C) las cajas adquieren una determinada velocidad (v_f) y una altura final igual a cero ($h = 0$). De esta forma los valores finales de energía cinética y potencial vienen dados como:

$$K_f = \frac{1}{2}mv_f^{\,2} \tag{120}$$

$$U_f = mgy_f = mg(0) = 0 \quad \textbf{(121)}$$

Aunque se tienen los valores de energía cinética final y energía potencia inicial, no se cumple la igualdad $U_i = K_f$ ya que existe una fuerza no conservativa que extrae energía mecánica del sistema (F_{R2}). En este caso el trabajo total del sistema lo realiza la fuerza de fricción, el cual puede ser expresado como la diferencia entre los valores de dichas energías (K_f y U_i) como se muestra a continuación:

$$\Delta K_{ext} = -Fs = -F_{R2}L_2$$

$$\Delta K_{ext} = K_f - U_i$$

$$-F_{R2}L_2 = \frac{1}{2}mv_f^{\,2} - mgh \quad \textbf{(122)}$$

Finalmente de la expresión (122) se despeja la variable correspondiente a la velocidad final de la caja en el punto C:

$$v_f^{\,2} = \frac{2\left(-F_{R2}L_2 + mgh\right)}{m}$$

$$v_f = \sqrt{\frac{2\left(-F_{R2}L_2 + mgh\right)}{m}} \quad \textbf{(123)}$$

6. PRÁCTICA DE MOMENTO LINEAL Y COLISIONES

La figura 24 muestra el esquema general del juego de billar pool diseñado para la práctica de colisiones. El procedimiento para determinar el ingreso de la bola 2 (bola objetivo) dentro de las buchacas habilitadas se puede realizar con base en el estudio de los subsistemas que componen la totalidad del juego. La primera parte del análisis (numerales 6.1 y 6.2) se realiza considerando que la bola 1 ha sido direccionada correctamente y apunta hacia la bola 2, dando por sentado la colisión entre ambas.

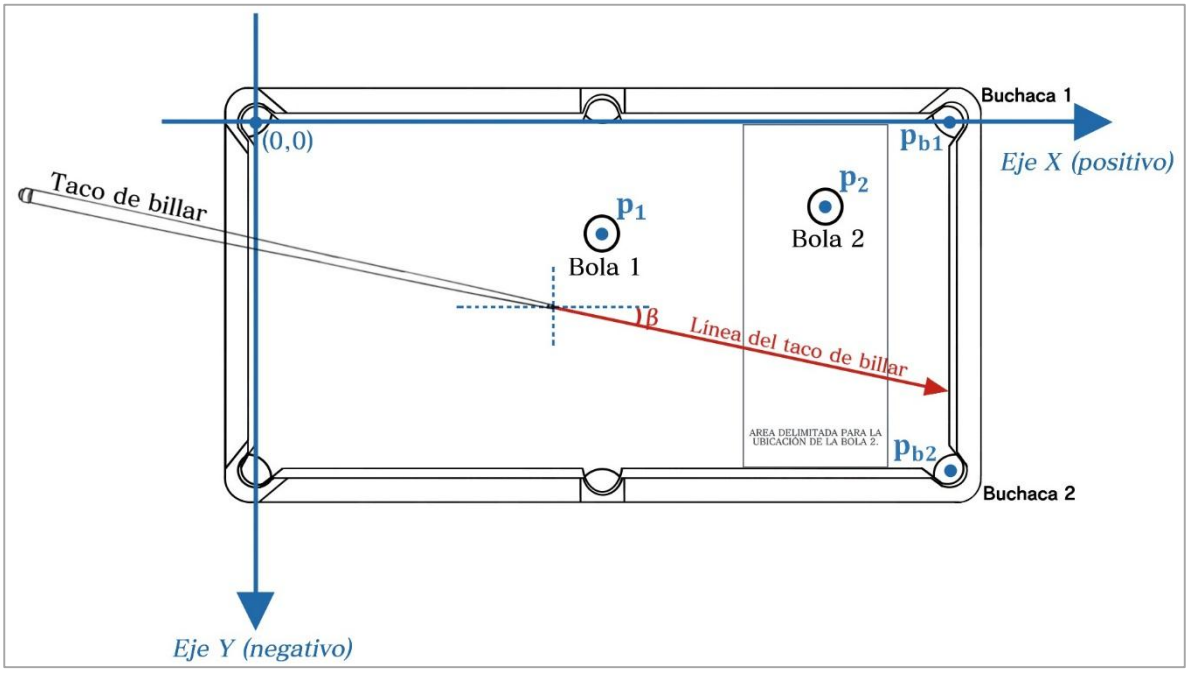


Figura 24. Práctica de momento lineal y colisiones – Sistema general.

Posteriormente en el numeral 6.3 se establecen los casos posibles de colisión, a partir de los cuales se determinan las direcciones que toman las bolas 1 y 2 después de dicho evento. Finalmente, con base en las direcciones encontradas se calculan las velocidades iniciales de las dos bolas justo después de que han chocado, así como también la velocidad final con que la bola 2 ingresa a la buchaca escogida (numerales 6.4 y 6.5 respectivamente).

6.1. Sistema taco de billar – Bola 1:

El primer subsistema que se considera dentro del juego de billar está conformado por la interacción inicial del taco de billar con la bola 1 (bola de impacto), como lo muestra la figura 25. En un instante dado el taco ejerce sobre la bola 1 un impulso (I) a través de la fuerza externa F_t, el cual tiene una corta duración (Δt), provocando un cambio en el momento inicial de la bola (ΔP).

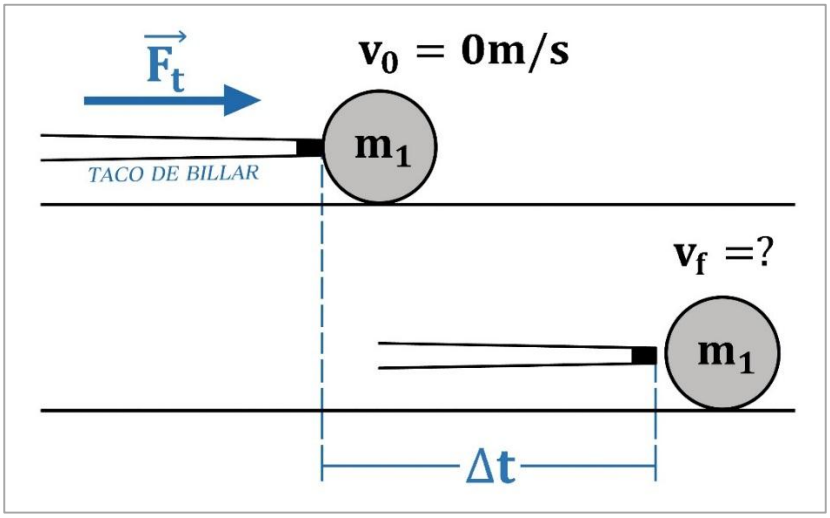


Figura 25. Sistema taco de billar – bola 1.

De esta manera el impulso expresado como el cambio del momento de la partícula queda definido de la siguiente forma:

$$I = \Delta P = P_f - P_i = m_1 v_f - m_1 v_i$$

$$I = m_1 v_f \tag{124}$$

Donde v_i y v_f corresponden a la velocidad de la bola justo antes y después del impulso dado por la fuerza externa F_t respectivamente. Para determinar la velocidad final de la bola después de que ha abandonado el taco de billar primero se expresa el impulso como una función del tiempo y se reemplaza en la ecuación (124). De esta nueva expresión se despeja la velocidad final v_f requerida:

$$I = \Delta t F_t \rightarrow m_1 v_f = \Delta t F_t$$

$$v_f = \frac{\Delta t F_t}{m_1} \tag{125}$$

6.2. Movimiento de la bola 1 antes de la colisión:

En un instante posterior al impulso dado por el taco de billar, la bola 1 se dirige hacia la bola 2, partiendo con una velocidad inicial v_{01} equivalente a la velocidad final hallada en el apartado anterior, como se muestra en la figura 26.

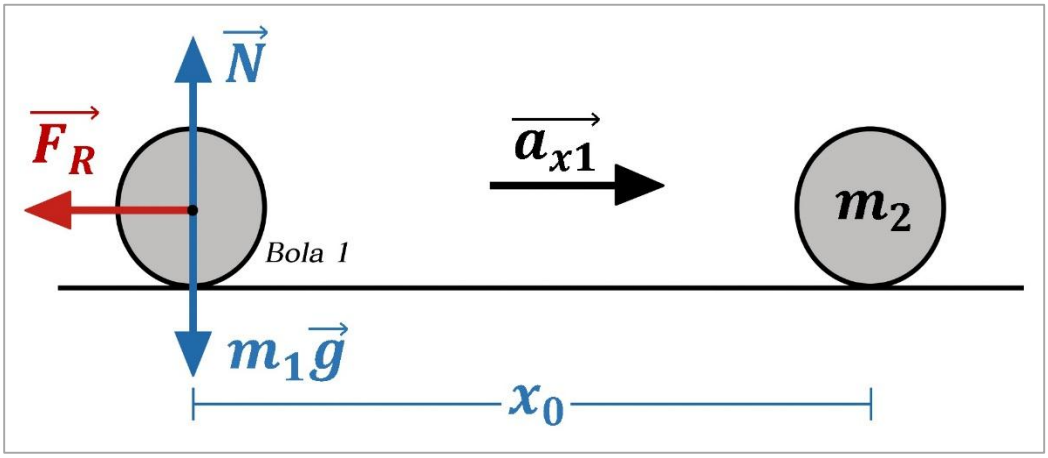


Figura 26. Movimiento de la bola 1 antes de la colisión.

De esta manera la velocidad inicial de la bola 1 viene dada como:

$$v_{01} = v_f = \frac{\Delta t F_t}{m_1} \tag{126}$$

Con base en la segunda ley de Newton se calculan los valores respectivos de la fuerza normal (N) ejercida por la mesa de billar sobre la bola 1 y la aceleración de esta en dirección x (a_{x1}):

$$\sum F_y = N - m_1 g = 0$$

$$N = m_1 g \tag{127}$$

$$\sum F_x = -F_R = m_1 a_{x1}$$

$$-\mu N = m_1 a_{x1}$$

$$-\mu m_1 g = m_1 a_{x1}$$

$$a_{x1} = -\mu g \tag{128}$$

Donde μ y g corresponden al coeficiente de fricción de la superficie de la mesa de billar y a la aceleración gravitacional respectivamente. Con base en la aceleración encontrada para la bola 1 se calcula la velocidad (v_{f1}) con que colisiona con la bola 2. Para ello se hace uso de la ecuación de velocidad como función del desplazamiento realizado por la bola del movimiento en una dimensión,

considerando que la distancia de separación de las bolas 1 y 2 dentro de la mesa de billar es X_0. De esta forma v_{f1} se define como:

$$v_{f1}{}^2 - v_{01}{}^2 = 2a_{x1}x_0$$

$$v_{f1}{}^2 = v_{01}{}^2 + 2a_{x1}x_0$$

$$v_{f1} = \sqrt{v_{01}{}^2 + 2a_{x1}x_0}$$

$$v_{f1} = \sqrt{\left(\frac{\Delta t F_t}{m_1}\right)^2 - 2\mu g x_0} \qquad \textbf{(129)}$$

6.3. Colisión de las bolas de billar – Parte 1:

Para llevar a cabo el análisis que permita determinar las ecuaciones de movimiento dentro de la presente práctica se considera que:

A. **SÍ** existe colisión entre las dos bolas de billar.

B. La dirección en que apunta el taco de billar (ángulo β) para golpear la bola 1 es la correcta y permite que la bola 2 sea depositada dentro de la buchaca 1.

Con base en estas suposiciones y con los datos de entrada de β, P_1, P_2, P_{b1} y P_{b2} se define el siguiente procedimiento denominado “geometría de la colisión” para determinar las direcciones de las bolas (θ y Φ) y sus velocidades después del choque entre ellas.

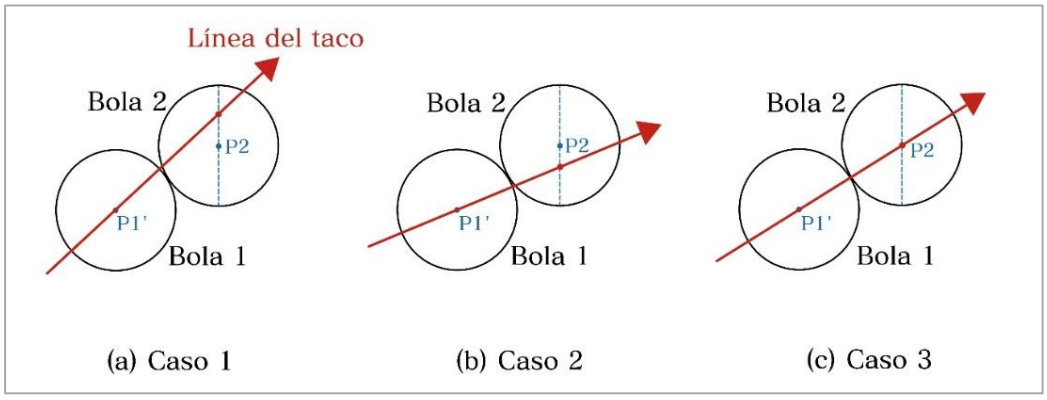


Figura 27. Posibles casos de colisión de las bolas 1 y 2.

De acuerdo a la figura 27 los casos posibles de colisión entre las bolas son los siguientes:

- **Caso 1 (figura 27-a):** Se da cuando la línea del taco de billar prolongada sobre el centro de la bola 1 en el sitio de colisión pasa por encima del centro de la bola 2.

- **Caso 2 (figura 27-b):** se presenta cuando la línea del taco de billar prolongada sobre el centro de la bola 1 en el sitio de colisión pasa por debajo del centro de la bola 2.

- **Caso 3 (figura 27-c):** se da cuando la línea del taco de billar prolongada sobre el centro de la bola 1 en el sitio de colisión pasa exactamente por el centro de la bola 2.

➢ GEOMETRÍA DE LA COLISIÓN – CASO 1:

La figura 28 muestra en detalle la colisión de las bolas 1 y 2 para el primer caso. La construcción geométrica se ha realizado considerando que para una dirección de inclinación del taco de billar (ángulo β) sobre la bola 1 se consigue ingresar la bola 2 dentro de la buchaca 1, representada por el punto P_{b1}. Es importante destacar que el punto de contacto de las bolas (P_c) siempre estará contenido dentro del segmento que une los centros de estas ($\overline{P_1' P_2}$), y es la dirección de este segmento respecto a la línea del taco de billar (ángulo θ) la que determina hacia donde se va a dirigir la bola 2 después del choque.

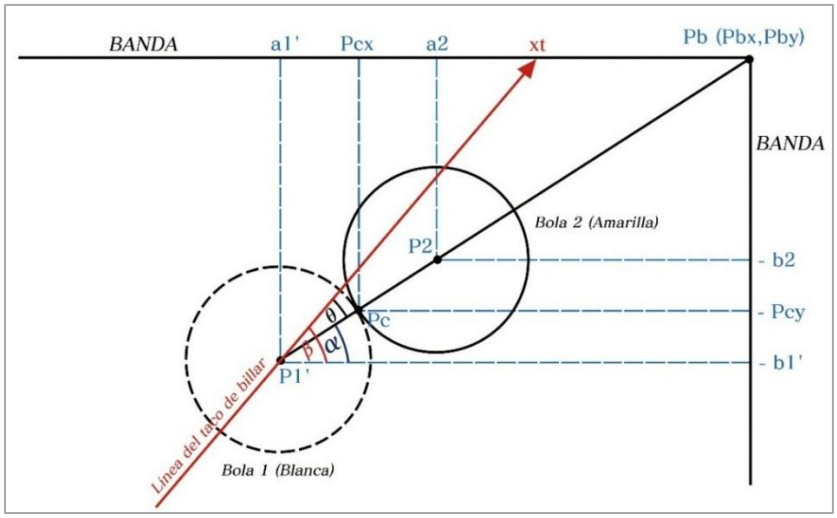


Figura 28. Caso 1 de colisión.

La descripción completa de los puntos y los ángulos involucrados dentro de esta colisión se realiza en la tabla 1:

Punto/Angulo	Descripción	Coordenadas
P_1	Centro de la bola 1 (bola blanca)	$(a_1, -b_1)$
$P_1{}'$	Centro de la bola 1 (bola blanca) durante la colisión	$(a_1{}', -b_1{}')$
P_2	Centro de la bola 2 (bola amarilla)	$(a_2, -b_2)$
P_c	Punto de contacto de las bolas 1 y 2	$(P_{cx}, -P_{cy})$
P_{b1}	Vértice de la buchaca 1	$(P_{b1x}, 0)$
β	Dirección del taco de billar respecto al eje x	-
α	Dirección del segmento $\overline{P_1{}'\,P_2}$ respecto al eje x	-
θ	Dirección que toma la bola 2 respecto a la línea del taco después de la colisión	-

Tabla 1. Ángulos y puntos considerados dentro de la colisión de las bolas.

Una vez definidos los elementos de la tabla 1 dentro de la colisión de las bolas de billar se realiza el procedimiento matemático para determinar el valor del ángulo θ (el ángulo Φ se calcula en un procedimiento posterior). Los pasos son los siguientes:

Paso 1: Se determina la distancia desde el centro de la bola 2 hasta el punto que representa a la buchaca 1:

$$\overline{P_2P_b} = X_1 = \sqrt{\left(P_{bx} - a_2\right)^2 + \left(0 - (-b_2)\right)^2} = \sqrt{\left(P_{bx} - a_2\right)^2 + \left(b_2\right)^2} \qquad \textbf{(130)}$$

Paso 2: Con base en el valor del segmento X_1 y las coordenadas de los puntos P_2 y P_{b1} se halla el valor del ángulo α (ver figura 29):

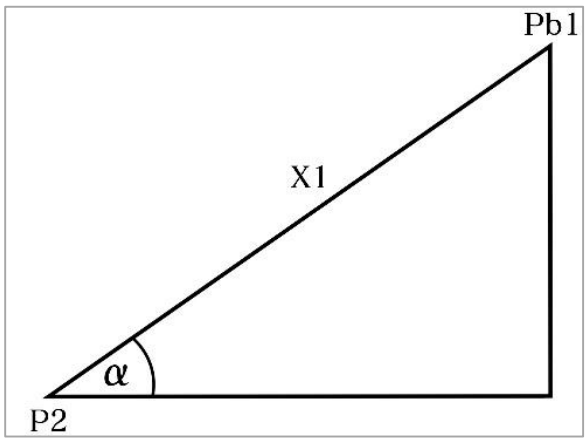


Figura 29. Determinación del ángulo α.

$$\alpha = \sin^{-1}\left[\frac{\left(0-\left(-b_2\right)\right)}{X_1}\right] = \sin^{-1}\left(\frac{b_2}{X_1}\right) \quad \textbf{(131)}$$

Paso 3: Se calcula la distancia de separación de las bolas de billar ($\overline{P_1P_2} = C = X_0$):

$$\overline{P_1P_2} = C = X_0 = \sqrt{\left(a_2-a_1\right)^2+\left(b_2-b_1\right)^2} \quad \textbf{(132)}$$

Paso 4: Se determinan las coordenadas del punto de contacto (P_c) de las bolas 1 y 2 de acuerdo con la figura 30. Para ello se calcula la longitud de los segmentos L_x y L_y, después, estos valores son restados respectivamente a las coordenadas en x y y del punto P_2.

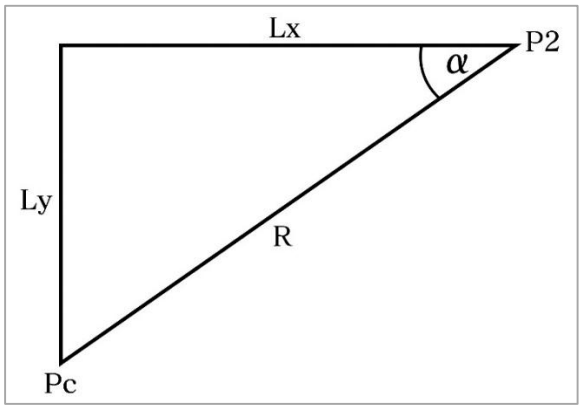


Figura 30. Determinación del punto de contacto de las bolas 1 y 2.

Los segmentos L_x y $L_y$vienen dados como:

$$\cos\alpha = \frac{L_x}{R} \rightarrow L_x = R\cos\alpha \quad \textbf{(133)}$$

$$\sin\alpha = \frac{L_y}{R} \rightarrow L_y = R\sin\alpha \quad \textbf{(134)}$$

Por lo tanto, las coordenadas del punto de contacto son:

$$\mathrm{p}_{cx} = \mathrm{a}_2 - L_x \quad \textbf{(135)}$$

$$\mathrm{p}_{cy} = -b_2 - L_y \quad \textbf{(136)}$$

Paso 5: Conociendo la posición del punto de contacto (P_c) se pueden determinar las coordenadas del centro de la bola 1 (P_1') en el sitio de colisión. De acuerdo a la figura 31 se realiza un procedimiento similar al paso anterior encontrando esta vez las magnitudes de los segmentos d_x y d_y, para después restar estos valores a las coordenadas en x y y del punto de contacto (P_c) respectivamente. Los segmentos d_x y d_y vienen dados como:

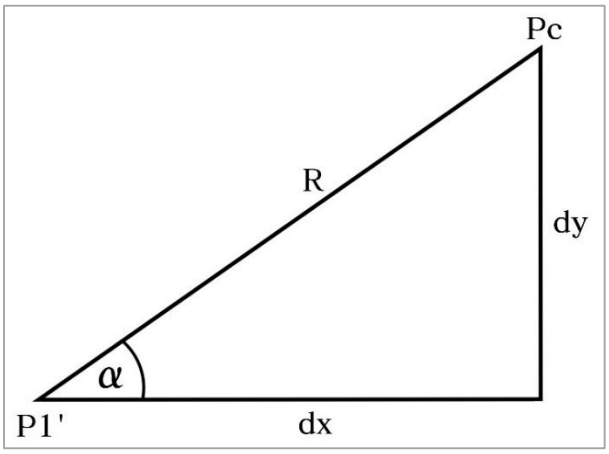


Figura 31. Determinación del centro de la bola 1 en la colisión.

$$\cos\alpha = \frac{d_x}{R} \rightarrow d_x = R\cos\alpha \quad \textbf{(137)}$$

$$\sin\alpha = \frac{d_y}{R} \rightarrow d_y = R\,sen\alpha \quad \textbf{(138)}$$

Por lo tanto, las coordenadas del centro de la bola 1 en el sitio de colisión son:

$$a_1' = P_{cx} - d_x \quad \textbf{(139)}$$

$$b_1' = P_{cy} - d_y \quad \textbf{(140)}$$

Paso 6: Se determina el valor del ángulo θ, tomando como referencia los triángulos que se forman dentro de la colisión y que son mostrados en la figura 32.

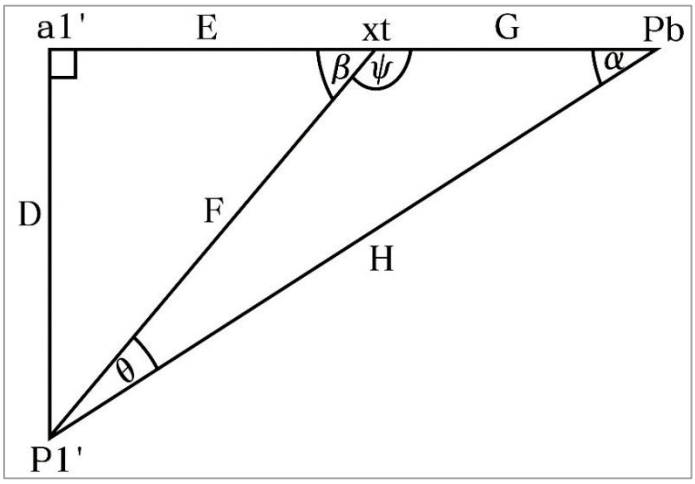


Figura 32. Determinación del ángulo θ – Caso 1 de colisión.

En este punto se deben calcular, de manera secuencial, los valores de los segmentos D, F, E, G y H respectivamente:

$$D = 0 - \left(-b_1'\right) = b_1' \quad \textbf{(141)}$$

$$F = \frac{D}{\sin\beta} \quad \textbf{(142)}$$

$$E = F\cos\beta \quad \textbf{(143)}$$

$$G = \left(p_{b1x} - \mathrm{a}_1'\right) - E \quad \textbf{(144)}$$

$$H = X_1 + 2R \tag{145}$$

Una vez calculadas las magnitudes de los segmentos del triángulo formado en la colisión de las bolas se determina el ángulo ψ, suplementario al ángulo β:

$$\psi = 180 - \beta \tag{146}$$

Con estos datos se aplica la ley trigonométrica de los senos para determinar el ángulo θ requerido:

$$\frac{\sin\alpha}{F} = \frac{\sin\psi}{H} = \frac{\sin\theta}{G} \tag{147}$$

- **GEOMETRÍA DE LA COLISIÓN – CASO 2:**

La figura 33 muestra en detalle el segundo caso de posible colisión entre las bolas 1 y 2. El ángulo α se determina siguiendo el mismo procedimiento visto en el caso 1.

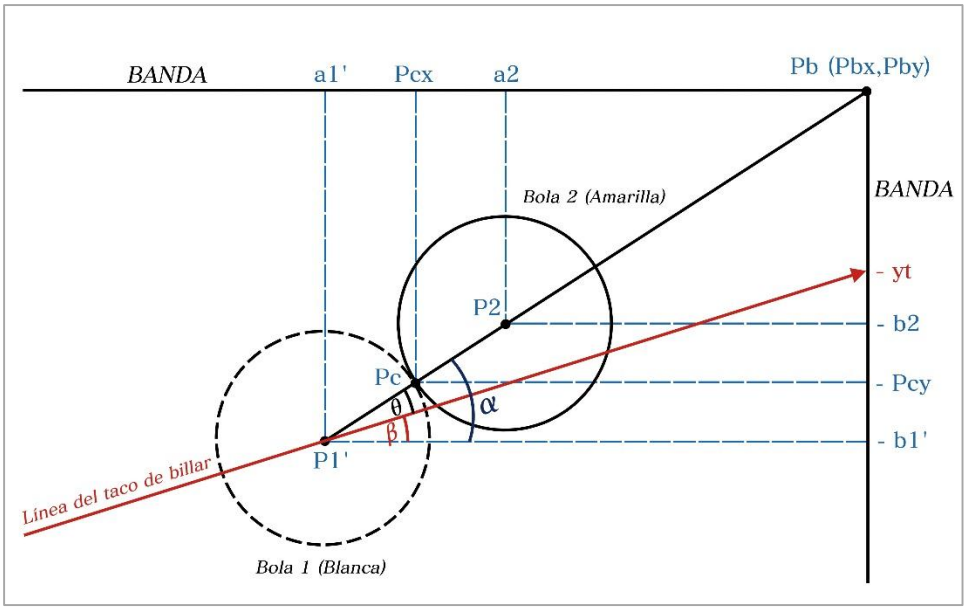


Figura 33. Caso 2 de colisión.

Con los ángulos α y β se calcula el valor del ángulo θ (dirección de la bola 2 después de la colisión) de la siguiente forma:

$$\alpha = \theta + \beta \quad \textbf{(148)}$$

$$\theta = \alpha - \beta \quad \textbf{(149)}$$

- **GEOMETRÍA DE LA COLISIÓN – CASO 3:**

La figura 34 muestra en detalle la colisión de las bolas 1 y 2 para el tercer caso. Como se puede observar, al estar alineados los centros de las dos bolas con el punto que representa a la buchaca 1, los ángulos α *y* β son iguales, de esta forma el ángulo θ equivale a 0° y por tanto la bola 2 se mueve en la misma dirección del taco de billar después de la colisión.

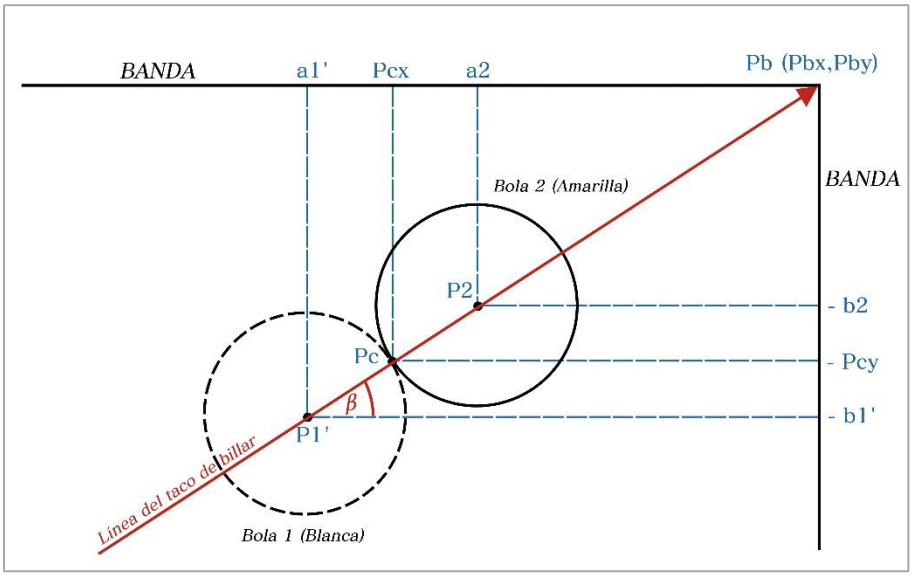


Figura 34. Caso 3 de colisión.

El procedimiento para determinar el ángulo α es el mismo que en los dos casos anteriores.

$$\alpha = \beta \quad \textbf{(150)}$$

$$\theta = 0 \quad \textbf{(151)}$$

Nota: Los tres casos de colisión presentados son generales para cualquier posición de la bola 2, por lo tanto si se escoge la buchaca 2 como objetivo, se realiza un procedimiento similar para encontrar los valores de α y θ.

6.4. Colisión de las bolas de billar – Parte 2:

La figura 35 muestra el tipo de colisión que realizan las bolas de billar (choque elástico) e ilustra los instantes anterior y posterior a dicho evento. Antes del choque las bolas 1 y 2 tienen velocidades $v_{1i} = v_{f1}$(hallada en el numeral 6.2) y $v_{2i} = 0$ respectivamente, mientras que después de la colisión poseen velocidades v_{1f} y v_{2f}.

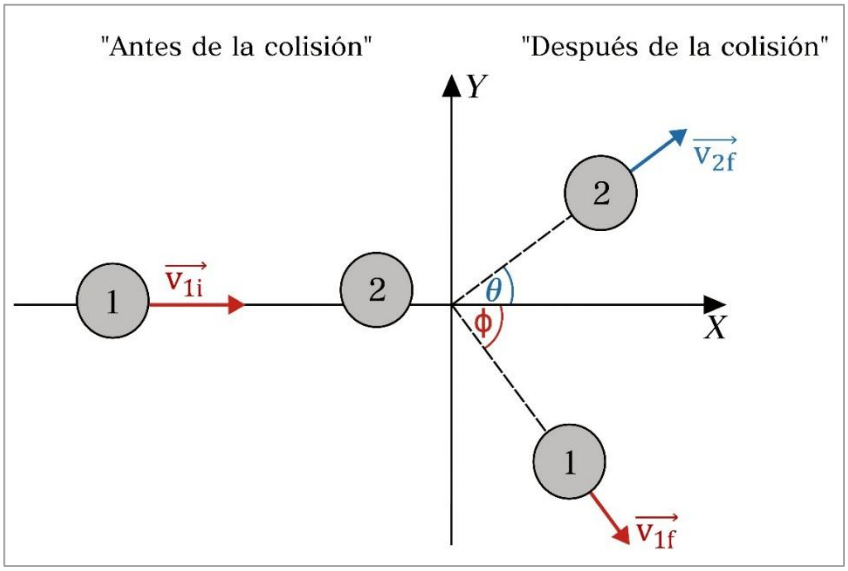


Figura 35. Colisión bidimensional de las bolas de billar.

Tomando como referencia el valor del ángulo θ hallado en el numeral anterior y considerando el ángulo Φ (aún sin determinar) como la dirección que toma la bola 1 después de chocar con la bola 2 se definen las componentes rectangulares para estas dos velocidades de la siguiente forma:

$$v_{2fx} = v_{2f}\cos\theta \quad \textbf{(152)}$$

$$v_{2fy} = v_{2f}\sin\theta \quad \textbf{(153)}$$

$$v_{1fx} = v_{1f}\cos\phi \quad \textbf{(154)}$$

$$v_{1fy} = -v_{1f}\sin\phi \quad \textbf{(155)}$$

Para ilustra mejor el proceso de colisión de las bolas se establece un sistema de ejes coordenados en el que el eje x se sitúa en la misma dirección del movimiento

de la bola 1 (dirección β). Aplicando la ley de conservación del momento para el choque elástico se tiene lo siguiente:

$$m_1 v_{1i} + m_2 v_{2i} = m_1 v_{1f} + m_2 v_{2f} \quad \textbf{(156)}$$

Como el choque es bidimensional se deben considerar dos ecuaciones de componentes para la conservación del momento (componentes en dirección x y y). La conservación del momento en dirección x viene dada como:

$$m_1 v_{1ix} + m_2 v_{2ix} = m_1 v_{1fx} + m_2 v_{2fx} \quad \textbf{(157)}$$

$$v_{1ix} + v_{2ix} = v_{1fx} + v_{2fx}$$

$$v_{1ix} = v_{1fx} + v_{2fx}$$

$$v_{1ix} = v_{1f}\cos\phi + v_{2f}\cos\theta \quad \textbf{(158)}$$

Antes del choque las bolas 1 y 2 no tienen componentes de velocidad en y, por lo tanto la conservación del momento en esta dirección queda definido de la siguiente manera:

$$m_1 v_{1iy} + m_2 v_{2iy} = m_1 v_{1fy} + m_2 v_{2fy} \quad \textbf{(159)}$$

$$v_{1iy} + v_{2iy} = v_{1fy} + v_{2fy}$$

$$0 = v_{1fy} + v_{2fy}$$

$$0 = -v_{1f}\sin\phi + v_{2f}\sin\theta \quad \textbf{(160)}$$

De la ecuación (160) se obtiene la expresión de velocidad de la bola 2 después de la colisión (v_{2f}):

$$v_{1f}\sin\phi = v_{2f}\sin\theta$$

$$v_{2f} = \frac{v_{1f}\sin\phi}{\sin\theta} \quad \textbf{(161)}$$

Se establece que después del choque las bolas se separan siguiendo trayectorias ortogonales, debido a que las masas de ambas son iguales y que el tipo de choque es elástico. De esta manera se puede determinar el ángulo Φ de a siguiente forma:

$$\theta + \phi = 90^\circ$$

$$\theta = 90^\circ - \phi$$

$$\phi = 90^\circ - \theta \qquad \textbf{(162)}$$

Reemplazando el valor de Φ en la ecuación (162) se obtiene una nueva expresión de v_{2f} en términos del ángulo θ:

$$v_{2f} = \frac{v_{1f}\sin\left(90^\circ - \theta\right)}{\sin\theta} = \frac{v_{1f}\cos\theta}{\sin\theta}$$

$$v_{2f} = v_{1f}\left(\frac{\cos\theta}{\sin\theta}\right) \qquad \textbf{(163)}$$

Reemplazando este nuevo valor de v_{2f} en la ecuación (158) se llega a la expresión general de v_{1f}, es decir, la velocidad de la bola 1 después de la colisión en términos de su velocidad inicial:

$$v_{1ix} = v_{1i} = v_{1f}\cos\phi + v_{1f}\left(\frac{\cos\theta}{\sin\theta}\right)\cos\theta = v_{1f}\cos\left(90^\circ - \theta\right) + v_{1f}\left(\frac{\cos^2\theta}{\sin\theta}\right)$$

$$v_{1ix} = v_{1i} = v_{1f}\sin\theta + v_{1f}\left(\frac{\cos^2\theta}{\sin\theta}\right) = \frac{v_{1f}\sin^2\theta + v_{1f}\cos^2\theta}{\sin\theta}$$

$$v_{1i}\sin\theta = v_{1f}\left(\sin^2\theta + \cos^2\theta\right)$$

$$v_{1f} = v_{1i}\sin\theta \qquad \textbf{(164)}$$

De esta manera la velocidad de la bola 2 (v_{2f}) después del choque queda definida en términos de la velocidad inicial de la bola 1 (v_{1i}):

$$v_{2f} = v_{1f}\left(\frac{\cos\theta}{\sin\theta}\right) = v_{1i}\sin\theta\left(\frac{\cos\theta}{\sin\theta}\right)$$

$$v_{2f} = v_{1i}\cos\theta \tag{165}$$

6.5. Movimiento de la bola 2 después de la colisión:

Como punto final se establecen las ecuaciones de movimiento de la bola 2 (bola objetivo) después de haber sido colisionada por la bola 1, de acuerdo a la figura 36.

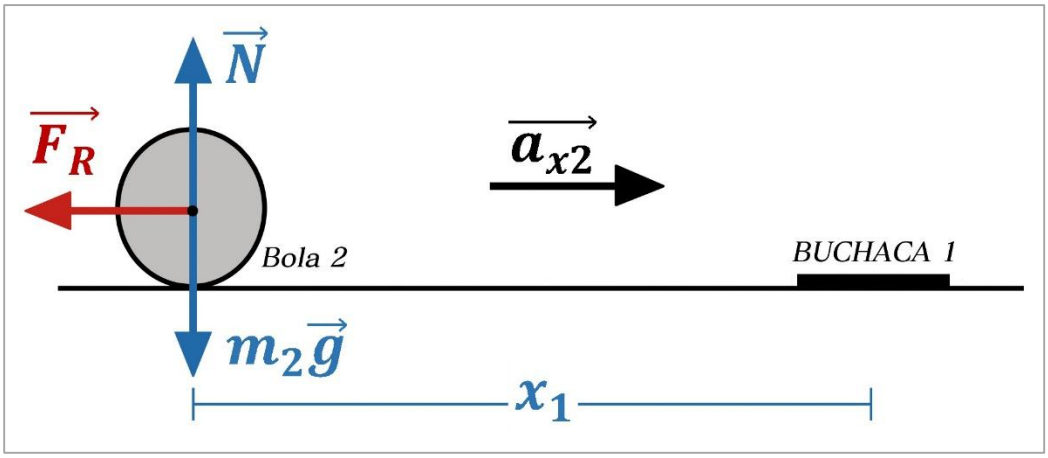


Figura 36. Movimiento de la bola 2 después de la colisión.

En primer lugar se define la velocidad inicial de la bola 2 (v_{02}) para este nuevo movimiento, la cual equivale a su velocidad justo después del choque, como se observa en la figura 36:

$$\mathrm{v}_{02} = v_{2f} = v_{1i}\cos\theta \tag{166}$$

A partir de la segunda ley de Newton se establecen los valores de la fuerza normal (N) y de la aceleración de la bola 2 (a_{x2}) respectivamente:

$$\sum F_y = N - m_2 g = 0$$

$$N = m_2 g \tag{167}$$

$$\sum F_x = -F_R = m_2 a_{x2}$$

$$-\mu N = m_2 a_{x2} \rightarrow -\mu m_2 g = m_2 a_{x2}$$

$$a_{x2} = -\mu g \qquad \textbf{(168)}$$

Con base en la aceleración encontrada para la bola 2 se calcula la velocidad (v_{f2}) con que esta llega a la buchaca 1, teniendo en cuenta que la distancia que las separa es X_1. De esta forma la velocidad v_{f2} se define como:

$$v_{f2}{}^2 - v_{02}{}^2 = 2a_{x2}x_1$$

$$v_{f2}{}^2 = v_{02}{}^2 + 2a_{x2}x_1$$

$$v_{f2} = \sqrt{v_{02}{}^2 + 2a_{x2}x_1}$$

$$v_{f2} = \sqrt{\left(v_{1i}\cos\theta\right)^2 - 2\mu g x_1} \qquad \textbf{(169)}$$

Anexo B

GUÍAS DE ACTIVIDADES PARA LAS PRÁCTICAS DE FÍSICA MECÁNICA

En el presente anexo se dan a conocer las diferentes guías de actividades para las prácticas de física mecánica, elaboradas con el fin de brindar soporte a los estudiantes que vayan a hacer uso de la herramienta de simulación. Las guías contienen un componente teórico y práctico referente a los temas más importantes tratados dentro de cada práctica.

PRÁCTICA DEMOSTRATIVA N° 1
(VECTORES)

1. INTRODUCCIÓN

Dentro de la física las cantidades vectoriales se describen por medio de segmentos orientados denominados vectores, los cuales poseen características tales como magnitud y dirección en el espacio. Dichas características pueden ser determinadas matemáticamente tomando como referencia un sistema de coordenadas espaciales en las que la posición de cada punto dentro del espacio se encuentra completamente definido. Existe un cierto número de sistemas de coordenadas espaciales para la representación de dichos puntos, sin embargo, dadas las características de la presente práctica en la que las trayectorias y desplazamientos realizados por una partícula se llevan a cabo en un espacio bidimensional (plano) se consideran únicamente los sistemas de coordenadas cartesianas y polares.

2. OBJETIVOS

Validar mediante simulación los conceptos estudiados de los vectores como elementos fundamentales en el estudio del movimiento y los fenómenos físicos asociados con cantidades vectoriales.

3. CONCEPTOS GENERALES

a) Sistema de coordenadas cartesianas: En este sistema cada punto del plano se representa por medio de sus coordenadas (x, y) como se muestra en la figura 1-1.

b) Sistema de coordenadas polares: En este sistema cada punto del plano se representa por medio de la distancia desde el origen de coordenadas hasta el punto denotado con coordenadas (x, y) llamada r y por el ángulo entre dicho segmento y el eje fijo x, denominado θ, como se muestra en la figura 1-2.

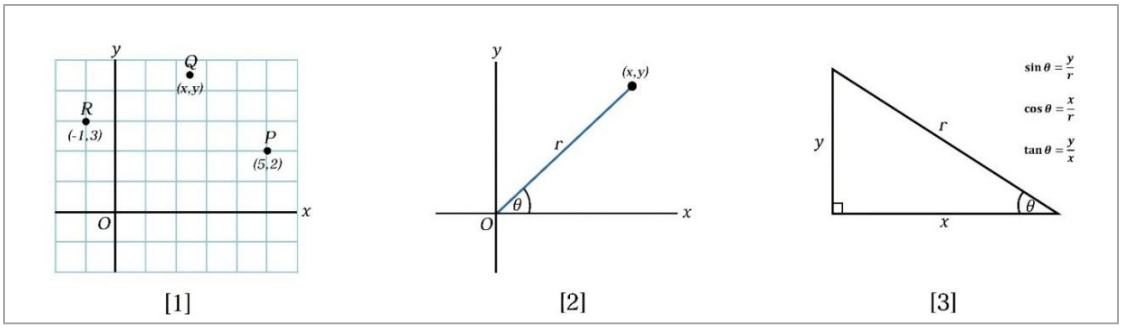


Figura 1. Sistema de coordenadas cartesianas y coordenadas polares.

La figura 1-3 muestra la relación que existe entre los dos sistemas de coordenadas descritos anteriormente. De esta manera es posible encontrar la equivalencia para cada uno de los parámetros de uno u otro sistema por medio de las siguientes expresiones:

$$\sin\theta = \frac{y}{r} \quad \textbf{(1)}$$

$$\cos\theta = \frac{x}{r} \quad \textbf{(2)}$$

$$\tan\theta = \frac{y}{x} \quad \textbf{(3)}$$

De las ecuaciones (1) y (2) se obtienen las expresiones que relacionan de forma más clara los dos tipos de coordenadas:

$$y = r \sin\theta \qquad \textbf{(4)}$$

$$x = r \cos\theta \qquad \textbf{(5)}$$

Así mismo, partiendo de las ecuaciones (4) y (5) se obtiene la siguiente expresión para calcular la magnitud *r*:

$$r = \sqrt{x^2 + y^2} \qquad \textbf{(6)}$$

La magnitud de una cantidad vectorial, representada por medio de un vector $\vec{\mathbf{A}}$, es una cantidad escalar A definida de la siguiente manera:

$$\text{Magnitud de } \vec{\mathbf{A}} = |\,\vec{\mathbf{A}}\,| = A \qquad \textbf{(7)}$$

Dos cantidades vectoriales pueden sumarse si y solo si representan la misma cantidad física. Dicha suma estará determinada por las reglas de la adición de vectores. De esta forma, si se suman dos vectores $\vec{\mathrm{A}}$ y $\vec{\mathrm{B}}$, el vector resultante $\vec{\mathrm{C}}$ puede obtenerse gráficamente a través de dos métodos geométricos. Uno de ellos es del denominado método del triángulo, en el cual $\vec{\mathrm{C}}$ es un vector que va desde el origen de $\vec{\mathrm{A}}$ hasta la punta o extremo del vector $\vec{\mathrm{B}}$ (ver figura 2-1), el otro, llamado método del paralelogramo en el que el vector resultante $\vec{\mathrm{C}}$ es la diagonal de un paralelogramo de lados A y B respectivamente (ver figura 2-2).

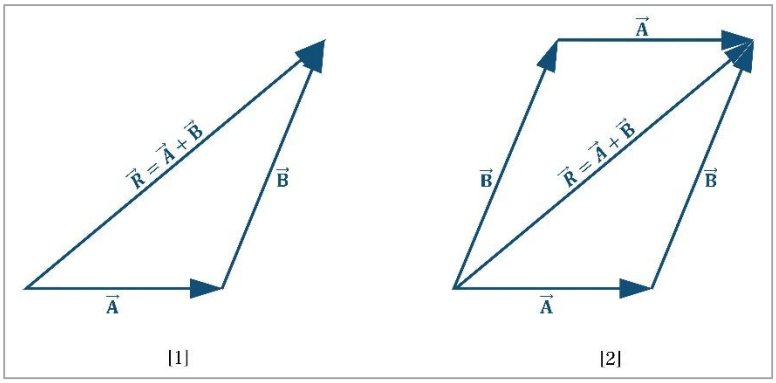


Figura 2. Métodos gráficos para la adición de vectores

Otra forma de llevar a cabo la suma vectorial es haciendo uso de las proyecciones de cada vector sobre los ejes de un sistema de coordenadas rectangular (Ver figura 3-1). Este método permite representar cualquier vector situado en el plano xy como la suma de dos vectores, uno situado en el eje x (vector componente en x) y otro sobre el eje y (vector componente en y) (Ver figura 3-2).

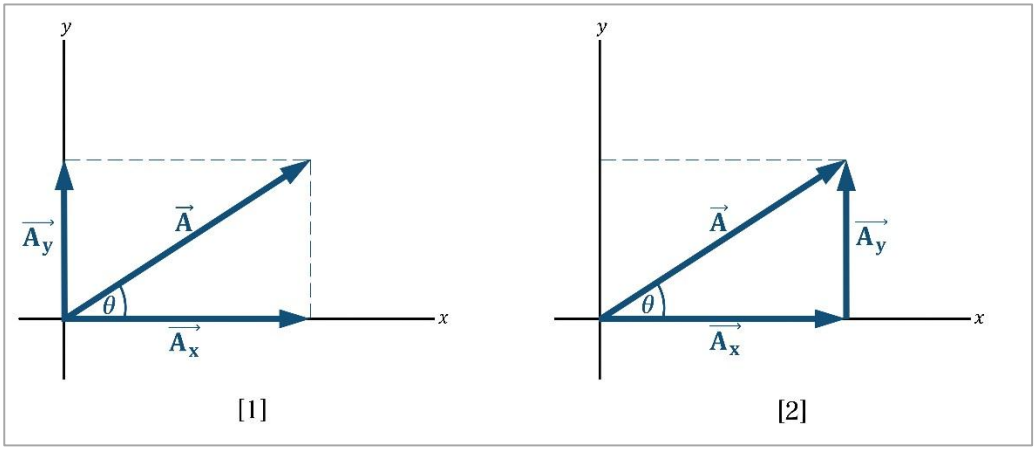


Figura 3. Componentes de un vector

Si se tiene un vector $\vec{A}$ su representación por medio de los vectores componentes es la siguiente:

$$\vec{A} = \overrightarrow{A_x} + \overrightarrow{A_y} \qquad \textbf{(8)}$$

De esta manera, la adición de vectores se puede realizar sumando respectivamente los vectores componentes de cada uno de ellos. Por ejemplo, si se tienen los vectores $\vec{A}$ y $\vec{B}$ la suma queda expresada así:

$$\vec{A} + \vec{B} = \left(\overrightarrow{A_x} + \overrightarrow{A_y}\right) + \left(\overrightarrow{B_x} + \overrightarrow{B_y}\right) = \left(\overrightarrow{A_x} + \overrightarrow{B_x}\right) + \left(\overrightarrow{A_y} + \overrightarrow{B_y}\right) \qquad \textbf{(9)}$$

Las componentes A_x y A_y, la magnitud del vector $\vec{A}$ y la dirección θ para un sistema de coordenadas rectangular se calculan por medio de las siguientes expresiones:

$$A_x = A\cos\theta \qquad \textbf{(10)}$$

$$A_y = A\sin\theta \qquad \textbf{(11)}$$

$$A = \sqrt{A_x{}^2 + A_y{}^2} \tag{12}$$

$$\theta = \tan^{-1}\left(\frac{A_y}{A_x}\right) \tag{13}$$

En cuanto a la dirección de un vector, esta puede estar dada en referencia a los puntos cardinales Norte, Este, Oeste y Sur, tal como se muestra en la figura 4:

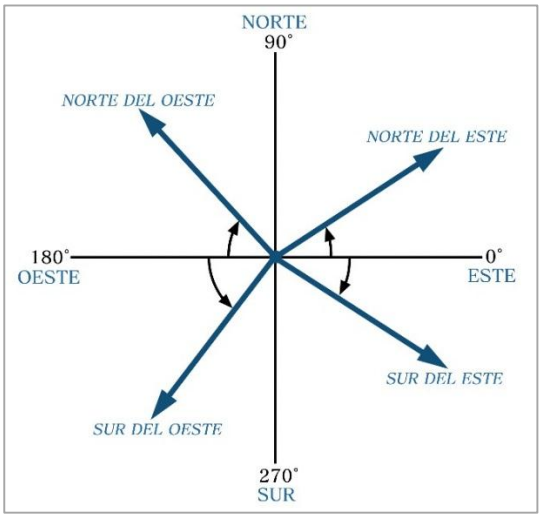


Figura 4. Dirección de un vector

Ahora bien, algunas de las cantidades vectoriales representadas por los vectores son la fuerza, la velocidad, la aceleración, la cantidad de movimiento, el desplazamiento de una partícula, entre otras. La presente práctica considera dos tipos de vector: Vector de posición y vector de desplazamiento.

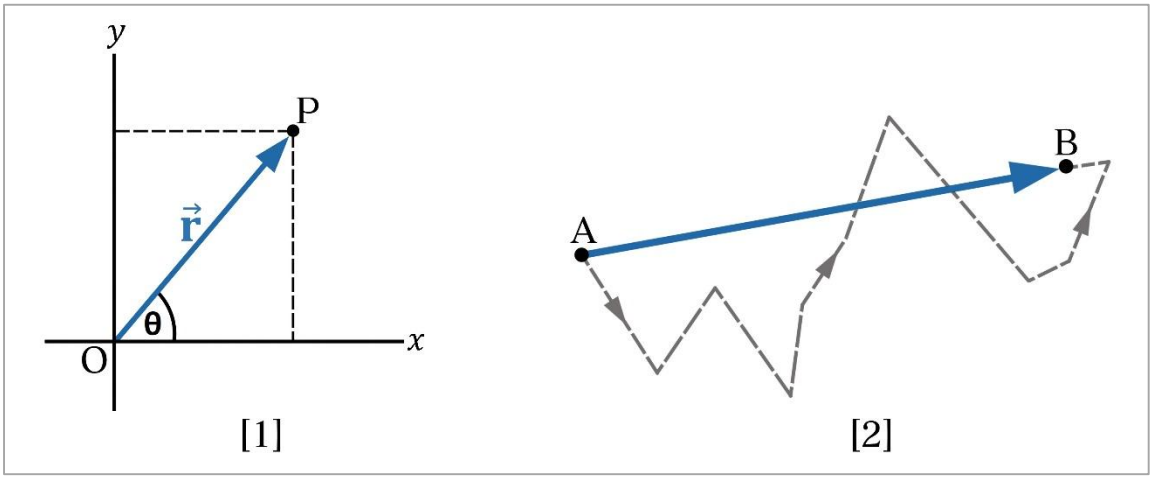


Figura 5. Vectores de posición y de desplazamiento.

a) Vector de Posición: Se utiliza para representar la posición de un cuerpo respecto al origen de un sistema de referencia. Para un cuerpo situado dentro de un sistema de coordenadas cartesianas su vector de posición es aquel que une el origen de los ejes con el punto p que representa la posición de dicho cuerpo (ver figura 5-1).

b) Vector de desplazamiento: El desplazamiento se define como el vector distancia que va desde una posición inicial de un cuerpo hasta la posición final del mismo. De esta manera, si un cuerpo parte de un punto *A* y llega un punto *B*, el vector que representa su desplazamiento será aquel que inicia en *A* y termina en *B*, sin importar el recorrido que se haya hecho (Ver figura 5-2).

Finalmente dentro de la práctica se involucra el concepto de rotación de los ejes de un plano cartesiano teniendo como referencia un punto ubicado dentro del plano original xy, el cual está definido por las coordenadas x e y respectivamente como se muestra en la figura 7. Al girar el sistema de ejes coordenados un cierto ángulo θ, el punto de referencia cambia sus coordenadas iniciales a unas nuevas denominadas (x', y') respecto al plano cartesiano $x'y'$.

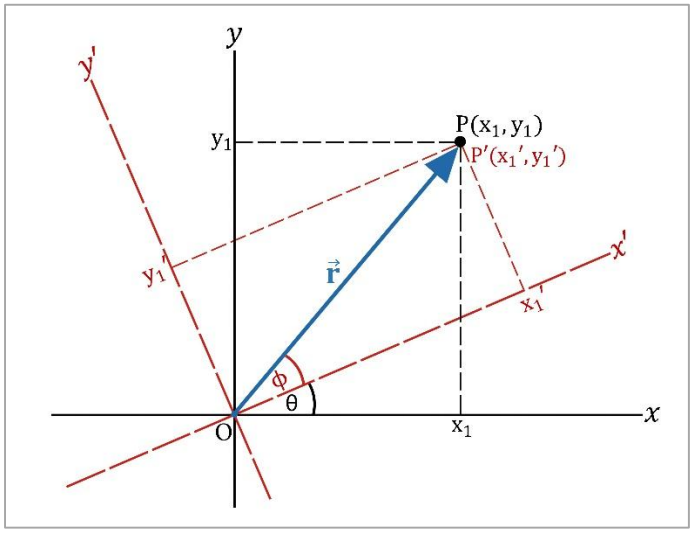


Figura 6. Rotación de los ejes coordenados xy.

Las ecuaciones para determinar las coordenadas del punto dentro del plano con ejes rotados (denominadas ecuaciones de transformación) son las siguientes:

$$y = x'\sin\theta + y'\cos\theta \qquad \textbf{(14)}$$

$$x = x'\cos\theta - y'\sin\theta \qquad \textbf{(15)}$$

4. DESARROLLO DE LA PRÁCTICA

Para desarrollar la práctica de vectores el estudiante debe tener acceso a un computador con conexión a internet, el cual debe contar con el explorador “Google Chrome” dentro de sus herramientas de navegación.

Nota: Antes de correr la aplicación tenga en cuenta las siguientes recomendaciones:

- La resolución del equipo donde se va a realizar la simulación debe ser ajustada a un valor de 1366 x 768.

- El tamaño de zoom del navegador “Google Chrome” debe estar en un valor de 100%.

- La página web donde se aloja la aplicación debe estar totalmente maximizada durante todo el desarrollo de la práctica.

PROCEDIMIENTO:

- **ACTIVIDAD 1: REPRESENTACIÓN DE UN VECTOR EN COORDENADAS POLARES.**

5. Ingrese a la siguiente página web: http://www.unicauca.edu.co/experimentos_mecanica/aplicacion.html para tener acceso a la aplicación web que contiene las prácticas virtuales relacionadas con la temática del curso, como se muestra en la figura 8.

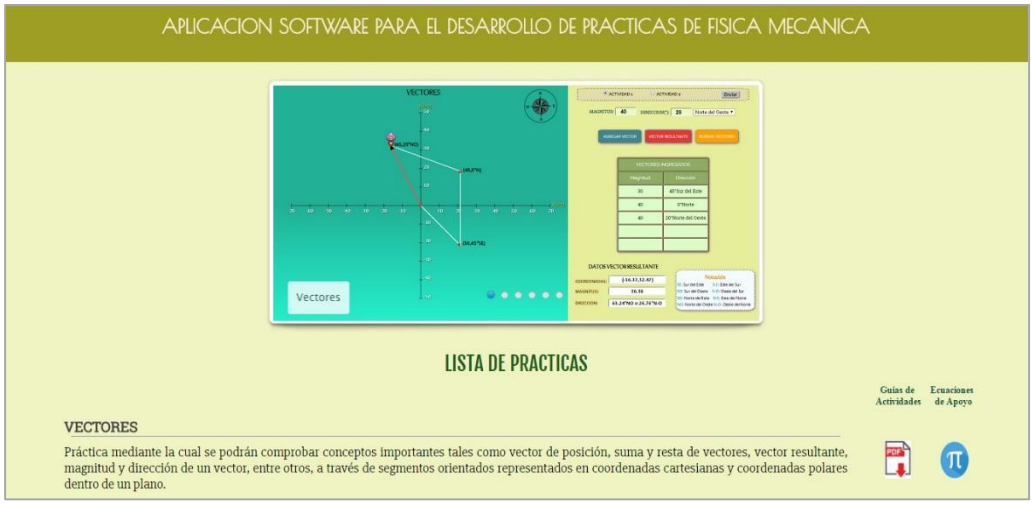


Figura 8. Interfaz principal de la aplicación web.

6. Para ingresar a la práctica de vectores haga click en el respectivo enlace, dentro de la "**Lista de prácticas**", como se muestra en la figura 9.

Figura 9. Enlace para la práctica de vectores.

A continuación el sistema desplegará la interfaz mostrada en la figura 10.

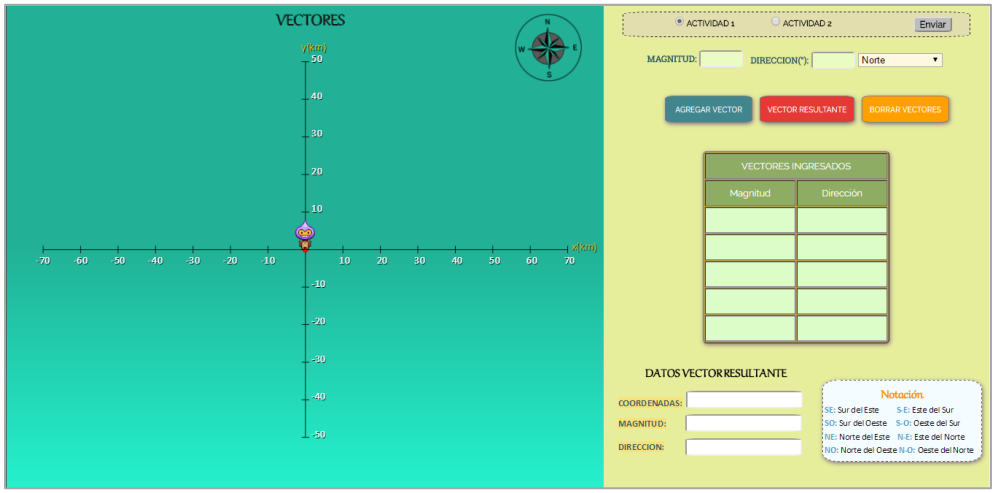


Figura 10. Interfaz principal de la práctica de vectores.

7. Seleccione la opción **"Actividad 1"** en la parte superior derecha de la interfaz y dé click en **"Enviar"** para ingresar a la práctica de Vectores con representación en coordenadas polares (Ver figura 11).

Figura 11. Selección de la actividad 1.

8. Ingrese los valores de 50 Km y 60 grados con dirección Oeste del Norte dentro de las casilla de **"Magnitud"** y **"Dirección"** respectivamente (Ver figura 12).

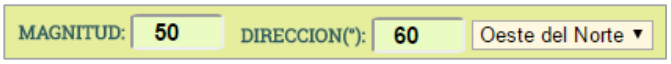


Figura 12. Magnitud y dirección del primer vector.

9. Haga click en el botón **"Agregar Vector"** para guardar los datos del vector ingresado (Ver figura 13).

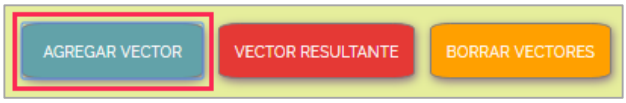


Figura 13. Agregar vector en coordenadas polares.

El vector que se acaba de agregar queda consignado dentro de la tabla **"Vectores Ingresados"** (Ver figura 14)**.** Esta tabla admite un total de 5 vectores.

Figura 14. Tabla de vectores ingresados – Vector 1.

De manera simultánea el vector agregado es dibujado por el personaje dentro del plano cartesiano xy desde el origen de coordenadas hasta el punto correspondiente (50 Km, 60° al Oeste del Norte) dado en coordenadas polares (ver figura 15). Consigne los datos de este vector en la tabla 1 (Vectores ingresados – Actividad 1).

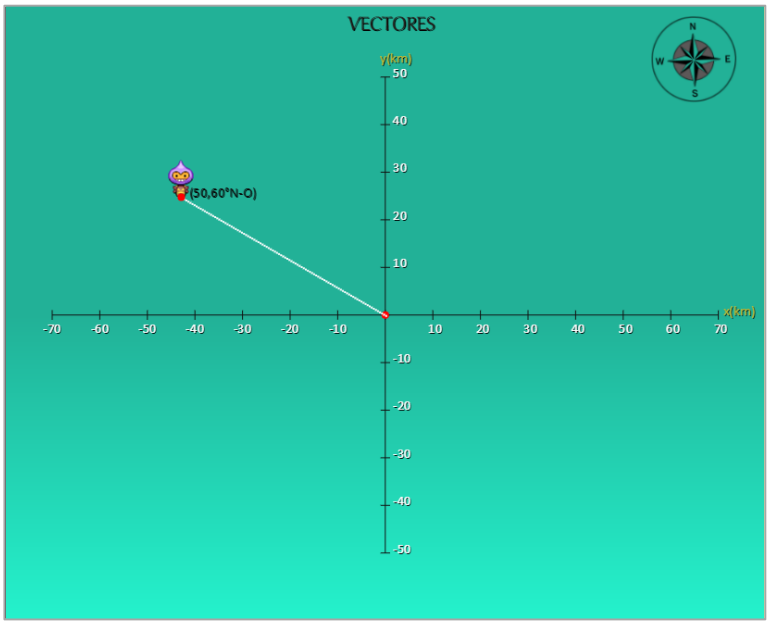


Figura 15. Vector 1 representado dentro del plano cartesiano xy.

10. Reescriba sobre los campos de **“Magnitud”** y **“Dirección”** los datos correspondientes a un nuevo vector y haga click en el botón **“Agregar Vector”**. De igual forma consigne estos valores dentro de la tabla 1.

Nota: El nuevo vector puede ser ubicado en cualquiera de los cuadrantes y debe estar dentro de las dimensiones del plano cartesiano.

11. Ingrese tres vectores más de modo que el personaje recorra todos los cuadrantes del plano cartesiano sin salirse de sus dimensiones. Con estos vectores complete la tabla 1.

VECTORES INGRESADOS		
Vector	**Magnitud**	**Dirección**
1		
2		
3		
4		
5		

Tabla 1. Vectores ingresados – Actividad 1.

12. Una vez que haya ingresado los cinco vectores dentro del plano cartesiano dé click en el botón **"Vector resultante"** para determinar la resultante de todos los desplazamientos realizados (Ver figura 16). De manera similar, el vector resultante es dibujado por la aplicación dentro del plano cartesiano.

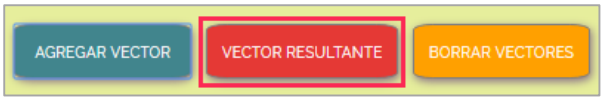


Figura 16. Vector resultante.

La figura 17 muestra los datos correspondientes al vector resultante dados por el software. Los datos que se generan son los siguientes: coordenadas en x y y, magnitud y dirección de dicho vector. Adicionalmente se muestra la notación utilizada que acompaña al valor de la dirección del vector resultante. Consigne en la tabla 2 los datos del vector resultante dados por la aplicación.

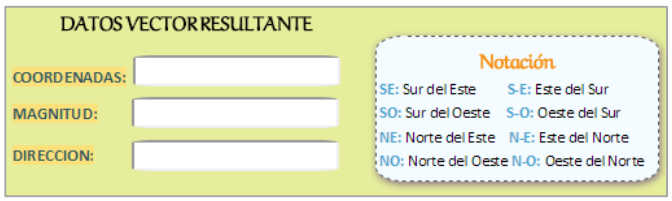


Figura 17. Datos del vector resultante.

DATOS DEL DESPLAZAMIENTO RESULTANTE (Software)			
Componente X	Componente Y	Magnitud	Dirección

Tabla 2. Datos del desplazamiento resultante – Actividad 1.

13. Haga click en el botón "**Borrar Vectores"** si desea trabajar en un nuevo problema (ver figura 18).

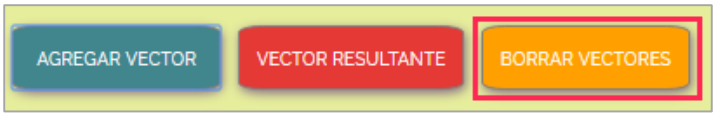


Figura 18. Borrar vectores ingresados.

14. PRUEBA DE CONOCIMIENTO – ACTIVIDAD 1:

A. Con base en los vectores consignados en la tabla 1 determine matemáticamente los siguientes valores:

- Componentes en x e y para cada vector ingresado.
- Dirección de cada vector con respecto al Norte.

Consigne estos resultados en la tabla 3.

Vectores	Componente *x*	Componente *y*	Dirección
Vector 1			
Vector 2			
Vector 3			
Vector 4			
Vector 5			

Tabla 3. Componentes y dirección de los vectores ingresados.

B. Con los valores de la tabla 3 determine matemáticamente la magnitud y dirección del vector resultante. Consigne estos valores en la tabla 4 y corrobore los datos de la tabla 2.

CÁLCULO DEL DESPLAZAMIENTO RESULTANTE			
Componente X	Componente Y	Magnitud	Dirección

Tabla 4. Cálculo del desplazamiento resultante.

➢ ACTIVIDAD 2: REPRESENTACIÓN DE UN VECTOR EN COORDENADAS CARTESIANAS

15. Seleccione la opción **"Actividad 2"** en la parte superior derecha de la interfaz y dé click en el botón **"Enviar"** para ingresar a la práctica de Vectores con representación en coordenadas cartesianas (Ver figura 19).

Figura 19. Selección de la Actividad 2.

El sistema desplegará la siguiente interfaz para que el estudiante lleve a cabo la práctica de vectores en coordenadas cartesianas (Ver figura 20).

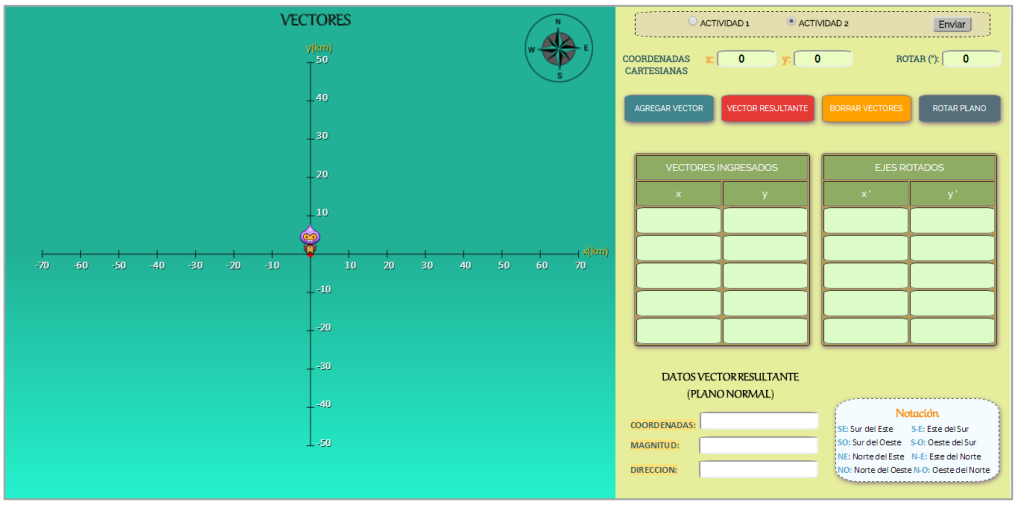


Figura 20. Interfaz gráfica para la actividad 2.

16. Defina un primer vector que no sobrepase los límites del plano cartesiano ingresando sus coordenadas (x, y) dentro de los campos **"X"** y **"Y"** de **"Coordenadas cartesianas"** respectivamente**.** El campo **"Rotar"** se debe dejar en 0 grados (ver figura 21).

Figura 21. Ingreso de coordenadas del primer vector.

17. Haga click en el botón **“Agregar Vector”** para guardar los datos del vector ingresado (Ver figura 22).

Figura 22. Agregar vector en coordenadas cartesianas.

El vector que se acaba de agregar queda consignado dentro de la tabla de **“Vectores Ingresados”** (Ver figura 23)**.** Esta tabla admite un total de 5 vectores.

Figura 23. Tabla de vectores ingresados.

18. Para ingresar un nuevo vector reescriba sobre los campos **“X”** y **“Y”** de **“Coordenadas cartesianas”** los valores correspondientes a sus coordenadas y dé click en el botón **“Agregar Vector”**. Repita este procedimiento hasta completar un total de cinco vectores ingresados. Consigne estos valores en la tabla 5.

VECTORES INGRESADOS		
Vector	**Coordenada x**	**Coordenada y**
1		
2		
3		
4		
5		

Tabla 5. Vectores ingresados – Actividad 2

19. Haga click en el botón **"vector Resultante"** para determinar la resultante de todos los vectores ingresados (Ver figura 24).

Figura 24. Vector resultante.

Los datos correspondientes al vector resultante dados por el software se muestran en la figura 25. Estos datos son los siguientes: coordenadas en x e y, magnitud y dirección de dicho vector. Adicionalmente se muestra la notación utilizada que acompaña al valor de la dirección del vector resultante. Consigne estos resultados en la tabla 6.

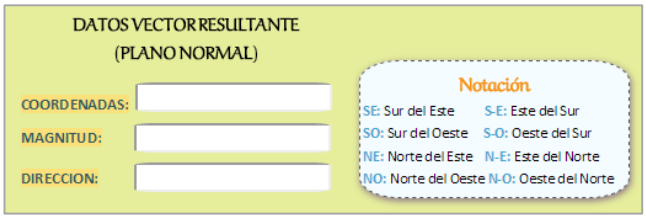


Figura 25. Datos del vector desplazamiento resultante.

DATOS DEL DESPLAZAMIENTO RESULTANTE (Software)			
Coordenada X	Coordenada Y	Magnitud	Dirección

Tabla 6. Datos del desplazamiento resultante – Actividad 2.

20. Ingrese un valor dentro del campo **"Rotar"** entre 0 y 180 grados con el fin de rotar el sistema de ejes coordenados (ver figura 26).

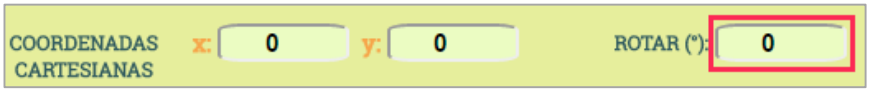


Figura 26. Ángulo de rotación de los ejes coordenados.

21. Haga click en el botón **"Rotar Plano"** para realizar la rotación de los ejes del plano cartesiano xy (Ver figura 27). La rotación se realizará en sentido contrario a las manecillas del reloj de acuerdo al valor del ángulo ingresado en el punto anterior.

Figura 27. Rotación de los ejes.

Con esto, los puntos registrados en la tabla 5 han cambiado sus coordenadas respecto al nuevo sistema de ejes coordenados denominado $x'y'$. Los puntos con las nuevas coordenadas los entrega el software en la tabla de **"Ejes Rotados"** (ver figura 28). Consigne estos valores en la tabla 7.

Figura 28. Tabla de coordenadas - ejes rotados.

VECTORES DENTRO DEL SISTEMA DE EJES ROTADOS		
Vector	**Coordenada x'**	**Coordenada y'**
1		
2		
3		
4		
5		

Tabla 7. Coordenadas de los vectores - sistema de ejes rotados.

22. PRUEBA DE CONOCIMIENTO 2:

Plano no Rotado:

A. Determine matemáticamente la magnitud y dirección de cada uno de los vectores de la tabla 5. Consigne estos valores en la tabla 8.

Vectores	Magnitud	Dirección
Vector 1		
Vector 2		
Vector 3		
Vector 4		

Vector 5		

Tabla 8. Magnitud y dirección de los vectores ingresados (plano no rotado).

B. Determine matemáticamente las componentes en x e y, la magnitud y dirección del vector resultante. Consigne estos valores en la tabla 9 y corrobore los valores de la tabla 6.

CÁLCULO DEL DESPLAZAMIENTO RESULTANTE			
Componente x	Componente y	Magnitud	Dirección

Tabla 9. Cálculo del vector resultante (plano no rotado).

Plano Rotado:

A. Obtenga matemáticamente las coordenadas (x', y') dentro del plano rotado para cada uno de los vectores de la tabla 5 y registre estos valores en la tabla 10. Compare estos resultados con los valores correspondientes de la tabla 7.

VECTORES DENTRO DEL PLANO ROTADO		
Vector	**Coordenada x'**	**Coordenada y'**
1		
2		
3		
4		
5		

Tabla 10. Coordenadas de los vectores ingresados (plano rotado).

B. Con base en estas nuevas coordenadas determine matemáticamente la magnitud y dirección de los cinco vectores dentro del plano rotado y consigne dichos valores dentro de la tabla 11.

Vectores	**Magnitud**	**Dirección**
Vector 1'		
Vector 2'		
Vector 3'		
Vector 4'		
Vector 5'		

Tabla 11. Magnitud y dirección de los vectores (plano rotado).

C. Determine matemáticamente las componentes x e y, la magnitud y dirección del vector de desplazamiento resultante dentro del plano rotado. Consigne estos valores en la tabla 12 y corrobore los valores de la tabla 9.

DATOS DEL DESPLAZAMIENTO RESULTANTE (Matemáticamente)			
Componente X	Componente Y	Magnitud	Dirección

Tabla 12. Datos del vector resultante (plano rotado).

D. De acuerdo a los cálculos realizados responda:

D.1. ¿En qué afecta la rotación del plano a los vectores ingresados inicialmente en la tabla 5?

D.2. ¿Qué ocurre con el vector resultante dentro del nuevo sistema con el plano rotado?

PRÁCTICA DEMOSTRATIVA N° 2

(MOVIMIENTO EN DOS DIMENSIONES)

1. INTRODUCCIÓN

Un movimiento de tipo bidimensional es aquel en el que un cuerpo realiza una serie de traslaciones en diferentes direcciones dentro de un plano definido. Las ecuaciones cinemáticas para este tipo de movimiento se deducen de igual forma que para el de una dimensión, es decir, partiendo de los conceptos de posición, velocidad y aceleración, pero considerando que cada variable cuenta con dos componentes, generalmente en las direcciones x y y. Dentro de la presente práctica se trabajará un sistema general compuesto por tres tipos de movimiento (movimiento de un cuerpo en caída libre, movimiento de un cuerpo sobre un plano inclinado y movimiento parabólico de un proyectil) en el que se observará el comportamiento de un objeto al ser introducido dentro de este sistema.

2. OBJETIVO

Esta práctica busca fortalecer los conceptos de cinemática haciendo uso de una herramienta software para emular el movimiento de cuerpos en el plano considerando los efectos de la aceleración y el comportamiento de estos en el movimiento parabólico.

3. MOVIMIENTO EN DOS DIMENSIONES CON ACELERACIÓN CONSTANTE

Se caracteriza por que los valores de magnitud y dirección de la aceleración no varían con el tiempo. De este modo, para un cuerpo que se mueve con aceleración constante sobre un plano xy las expresiones de posición y velocidad vienen dadas de la siguiente forma:

a. **Vector de posición:** Determina el movimiento de la partícula dentro del plano debido a la variación de las coordenadas x y y con relación al tiempo. Viene dado por la siguiente expresión:

$$r = xi + yj \qquad \textbf{(1)}$$

b. Vector de velocidad: Se obtiene derivando la función del vector de posición. Esta derivada representa el cambio de posición del cuerpo a medida que transcurre el tiempo:

$$v = \frac{dr}{dt} = \frac{d}{dt}\left(xi + yj\right) = \frac{dx}{dt}i + \frac{dy}{dt}j = v_x i + v_y j \quad \textbf{(2)}$$

donde:

$$v_x = v_{x0} + a_x t \quad \textbf{(3)}$$

$$v_y = v_{y0} + a_y t \quad \textbf{(4)}$$

De la ecuación (3), los términos v_x, v_{x0} y a_x corresponden a las componentes de velocidad, velocidad inicial y aceleración en dirección x respectivamente. Así mismo en la ecuación (4), los términos v_y, v_{y0} y a_y representan las mismas componentes pero en dirección y.

Al sustituir las ecuaciones (3) y (4) en la ecuación (2) se obtiene la expresión general de velocidad en función del tiempo:

$$v = v_0 + at \quad \textbf{(5)}$$

donde v, v_0 y a corresponden a las expresiones generales de velocidad, velocidad inicial y aceleración para el movimiento bidimensional. Con la ecuación general de velocidad y retomando la ecuación (1) se encuentran las expresiones para las coordenadas x y y de la partícula:

$$x = x_0 + v_{x0}t + \frac{1}{2}a_x t^2 \quad \textbf{(6)}$$

$$y = y_0 + v_{y0}t + \frac{1}{2}a_y t^2 \quad \textbf{(7)}$$

donde x_0 y y_0 representan las posiciones iniciales del objeto en las direcciones x y y respectivamente.

3.1. MOVIMIENTO DE CAÍDA LIBRE:

El movimiento de un objeto que cae desde cierta altura en ausencia de fuerzas resistivas está gobernado por una aceleración constante debida a la fuerza de gravedad terrestre, lo que hace que dicho cuerpo se desplace en caída libre hacia el centro de la tierra (ver figura 1).

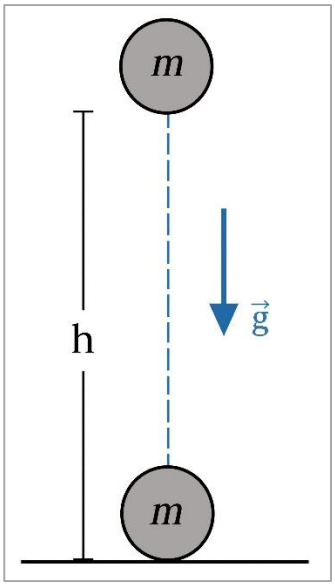


Figura 1. Movimiento de caída libre.

Las ecuaciones (4) y (7) describen la velocidad y posición de la partícula para este tipo de movimiento, mientras que la aceleración toma un valor constante igual al de la gravedad terrestre ($a = g$). Dado que la caída libre se lleva a cabo en la dirección y, los valores de x, v_x y a_x se consideran iguales a cero.

3.2. MOVIMIENTO PARABÓLICO:

En esta clase de movimiento los cuerpos parten con un valor inicial de velocidad y describen un tipo de trayectoria definida principalmente por los efectos de la aceleración gravitacional y de la oposición del aire sobre ellos. La figura 2 ilustra el caso ideal de movimiento parabólico, el cual cuenta con las siguientes características: Un ángulo inicial para el disparo del objeto, una aceleración constante debida a la gravedad (no se tienen en cuenta los efectos resistivos del aire), una velocidad inicial siempre mayor que cero y componentes de velocidad constante en x y variable en y durante todo el movimiento.

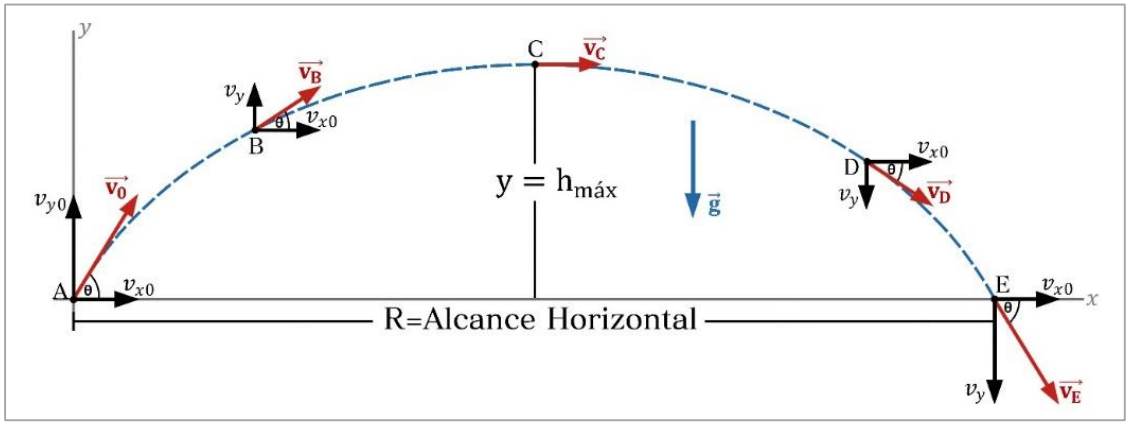


Figura 2. Movimiento de proyectiles.

De acuerdo a la figura 4 las expresiones de velocidad inicial en x y y, altura máxima ($h_{máx}$) y alcance horizontal del proyectil (R) respectivamente son las siguientes:

$$v_{x0} = v_0 \cos\theta \quad \textbf{(8)}$$

$$v_{y0} = v_0 \sin\theta \quad \textbf{(9)}$$

$$h_{máx} = \frac{\left(v_{y0}\right)^2}{2g} \quad \textbf{(10)}$$

$$R = \frac{\left(v_0\right)^2 \sin\left(2\theta\right)}{g} \quad \textbf{(11)}$$

donde v_0 es la velocidad inicial con que parte el proyectil y θ es el ángulo inicial con el que es disparado.

4. DESARROLLO DE LA PRÁCTICA

Para desarrollar la práctica de movimiento en dos dimensiones el estudiante debe tener acceso a un computador con conexión a internet, el cual debe contar con el explorador “Google Chrome” dentro de sus herramientas de navegación.

Nota: Antes de correr la aplicación tenga en cuenta las siguientes recomendaciones:

- La resolución del equipo donde se va a realizar la simulación debe ser ajustada a un valor de 1366 x 768.

- El tamaño de zoom del navegador “Google Chrome” debe estar en un valor de 100%.

- La página web donde se aloja la aplicación debe estar totalmente maximizada durante todo el desarrollo de la práctica.

PROCEDIMIENTO:

5. Ingrese a la siguiente página web: http://www.unicauca.edu.co/experimentos_mecanica/aplicacion.html para tener acceso a la aplicación web que contiene las prácticas virtuales relacionadas con la temática del curso, como se muestra en la figura 8.

Figura 3. Interfaz principal de la aplicación web.

6. Para ingresar a la práctica de movimiento bidimensional haga click en el respectivo enlace, dentro de la “**Lista de prácticas**”, como se muestra en la figura 4. A continuación el sistema desplegará la interfaz mostrada en la figura 5.

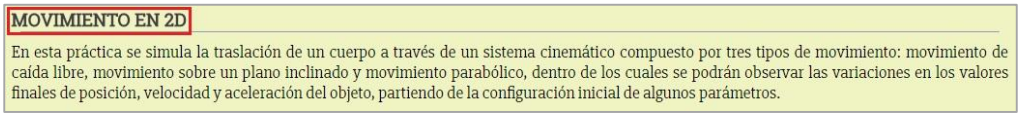


Figura 4. Enlace para la práctica de movimiento en 2D.

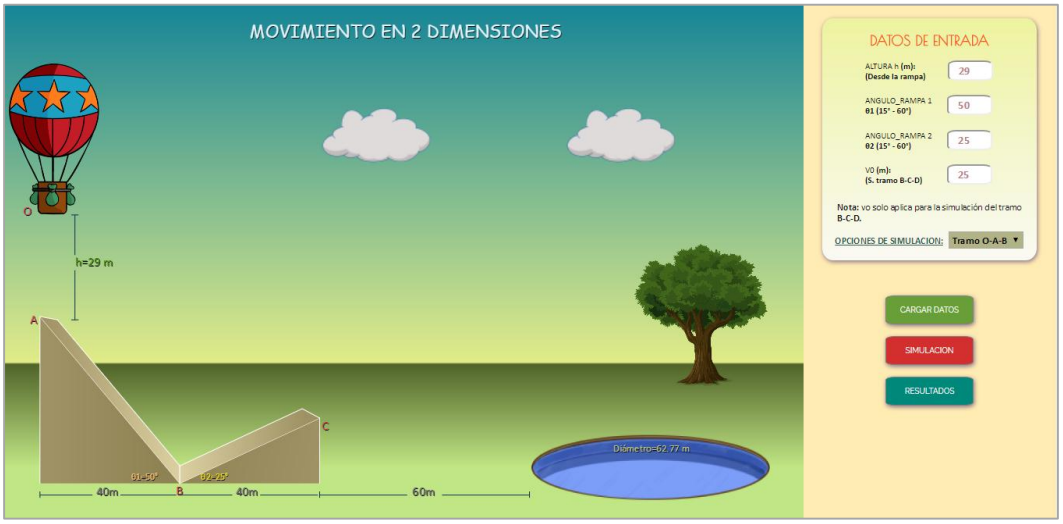


Figura 5. Interfaz principal – Movimiento en dos dimensiones.

EXPERIMENTO 1: ANÁLISIS DE CAÍDA LIBRE + MOVIMIENTO DESCENDENTE SOBRE UN PLANO INCLINADO (Tramo O-A-B).

7. Seleccione la opción **"Tramo O-A-B"** dentro del cuadro de **"Opciones de Simulación"** e ingrese en los campos correspondientes los valores que aparecen en la figura 6: Altura inicial de la pelota (h) igual a 55m (medida desde el punto A), ángulo de elevación de la rampa 1 (θ_1) de 40° y ángulo de elevación de la rampa 2 (θ_2) de 15°. El campo de velocidad inicial v_0 se debe dejar vacío. Consigne los datos ingresados en una tabla para el informe escrito.

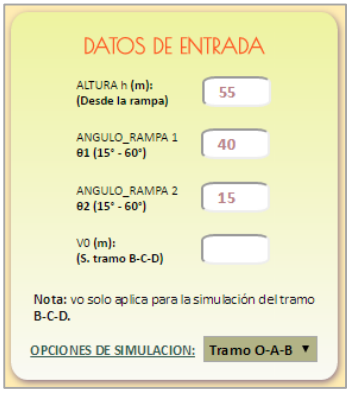


Figura 6. Datos de entrada del experimento 1.

8. Cargue los datos ingresados con el botón **“Cargar Datos”** e inicie la simulación haciendo click en el botón **“Simulación”** (ver figura 7). El software realizará la simulación del recorrido hecho por la pelota desde el punto O hasta el punto B.

Figura 7. Botón de cargar datos y de inicio de simulación.

9. Cuando se haya completado la simulación dé click en el botón **“Resultados”** (ver figura 8), con esto, en la parte inferior de la aplicación se despliegan las gráficas de posición en x y y, velocidad y aceleración correspondientes al experimento 1 (ver figura 9). De igual forma el software muestra los resultados numéricos del experimento dentro de la tabla de **“Datos y resultados numéricos”** (ver figura 10). Consigne los resultados numéricos en la tabla 1 y guarde pantallazos de las gráficas generadas en la simulación.

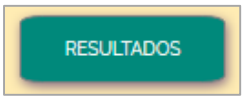


Figura 8. Botón de despliegue de resultados.

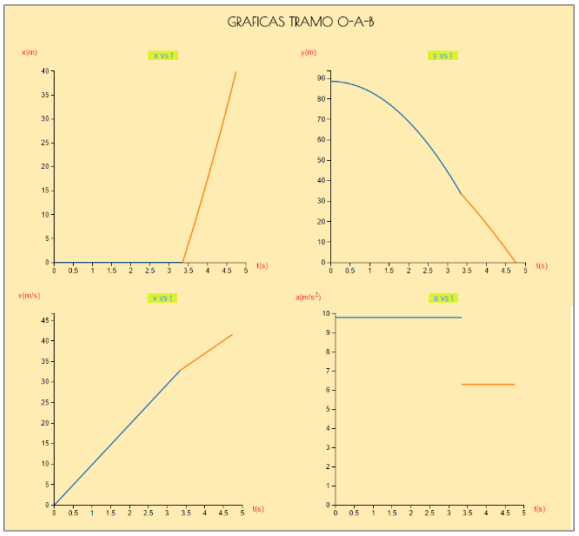
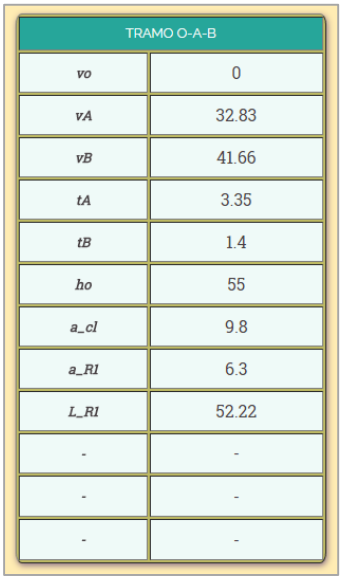


Figura 9. Gráficas generadas en la simulación.



TRAMO O-A-B	
vo	0
vA	32.83
vB	41.66
tA	3.35
tB	1.4
ho	55
a_cl	9.8
a_R1	6.3
L_R1	52.22
-	-
-	-
-	-

Figura 10. Resultados numéricos dados por la herramienta.

Datos y resultados numéricos	Valor
Velocidad inicial: v_0 (m/s)	
Velocidad en el punto A: v_A	
Velocidad en el punto B: v_B	
Tiempo transcurrido hasta el punto A: t_A	
Tiempo transcurrido hasta el punto B: t_B	
Altura inicial de la pelota desde la rampa 1: h_0	
Aceleración Movimiento de Caída Libre: a_{cl}	
Aceleración Movimiento en la Rampa 1: a_{R1}	
Longitud de la rampa 1: L_{R1}	

Tabla 1. Datos y resultados dados por el software – Experimento 1.

10. PRUEBA DE CONOCIMIENTO – EXPERIMENTO 1:

A. Con base en el experimento 1 realizado a partir de los datos de la figura 6 demuestre matemáticamente los valores obtenidos en la tabla 1.

B. Describa con sus palabras cada una de las gráficas de movimiento generadas por el simulador.

C. ¿En qué proporción cambiará la velocidad de la pelota en el punto B si el ángulo de elevación de la rampa 1 (θ_1) se incrementa hasta un valor de 60°?

EXPERIMENTO 2: ANÁLISIS DE MOVIMIENTO ASCENDENTE SOBRE UN PLANO INCLINADO + MOVIMIENTO PARABÓLICO (Tramo B-C-D).

11. Seleccione la opción **"Tramo B-C-D"** dentro del cuadro de **"Opciones de Simulación"** e ingrese en los campos correspondientes los valores que aparecen en la figura 11: Altura inicial de la pelota (h) igual a 70 metros (medida desde el punto A), ángulo de elevación de la rampa 1 (θ_1) de 15°, ángulo de elevación de la rampa 2 (θ_2) de 50° y velocidad inicial del balón (v_0) de 40 m/s. Consigne los datos ingresados en una tabla para el informe escrito.

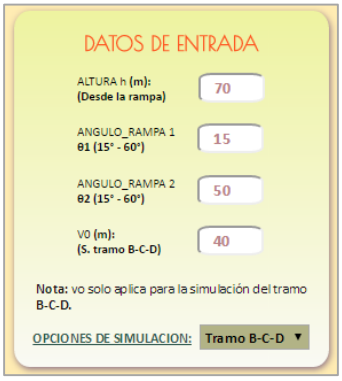


Figura 11. Datos de entrada del experimento 2.

12. Cargue los datos ingresados con el botón **"Cargar Datos"** e inicie la simulación haciendo click en el botón **"Simulación"** (ver figura 12). El software realizará la simulación del recorrido hecho por la pelota desde el punto B hasta el punto D.

Figura 12. Botón de cargar datos y de inicio de simulación.

13. Cuando se haya completado la simulación dé click en el botón **"Resultados"** (ver figura 13). Al igual que en el caso anterior, se desplegarán las gráficas de

posición en x y y, velocidad y aceleración correspondientes al experimento 2 junto con los datos y resultados numéricos asociados al mismo. Consigne los resultados numéricos en la tabla 2 y guarde pantallazos de las gráficas generadas en la simulación.

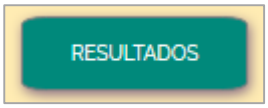


Figura 13. Botón de despliegue de resultados.

Datos y resultados numéricos	Valor
Velocidad inicial: v_0 (m/s)	
Velocidad en el punto C: v_C (m/s)	
Velocidad en el punto D: v_D (m/s)	
Tiempo transcurrido hasta el punto C: t_C	
Tiempo transcurrido hasta el punto D: t_D	
Aceleración Movimiento en la Rampa 2: a_{R2}	
Longitud de la rampa 1: L_{R2}	
Aceleración Movimiento Parabólico: a_{mp}	
Altura Máxima Movimiento Parabólico: $h_{máx}$	
Alcance Horizontal Movimiento Parabólico: R	

Tabla 2. Datos y resultados dados por el software – Experimento 2.

14. PRUEBA DE CONOCIMIENTO – EXPERIMENTO 2:

A. Con base en el experimento 2 realizado a partir de los datos de la figura 11 demuestre matemáticamente los resultados de la tabla 2.

B. Partiendo de los valores de $\theta_1 = 40°$ y $\theta_2 = 50°$ (ángulos de elevación de las rampas 1 y 2 respectivamente) realice los cálculos necesarios para determinar el rango de valores que debe tomar la altura inicial de la pelota (h) de modo que se garantice que esta caiga siempre dentro del estanque de agua (el diámetro del estanque es de 62.77 metros y la distancia de separación de este con el borde de la rampa 2 es de 60 metros).

Cuando haya completado los cálculos, con la ayuda del software compruebe si los valores obtenidos de altura inicial de la pelota son correctos. Para ello reescriba en el campo **"Altura h (m)"** del cuadro de

“Datos de Entrada” los valores encontrados. Reescriba también los datos de θ_1 y θ_2 en los campos correspondientes y seleccione la opción **“Tramo Total”** dentro del cuadro de **“Opciones de Simulación”** (ver figura 14), cargue los datos ingresados con el botón **“Cargar Datos”** e inicie la simulación haciendo click en el botón **“Simulación”.**

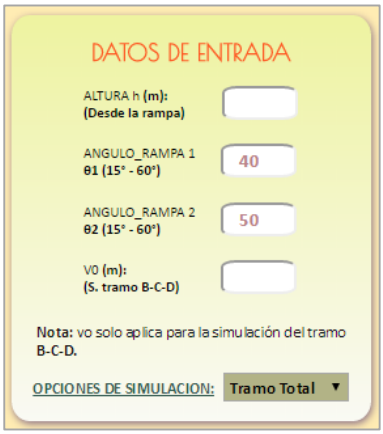


Figura 14. Campos asignados para el ingreso de los datos.

Cuando se haya completado la simulación dé click en el botón **“Resultados”** para generar las gráficas de movimiento y los resultados asociados con este experimento**.** Consigne los resultados numéricos en la tabla 3 y guarde pantallazos de las gráficas obtenidas en la simulación.

Datos y resultados numéricos	Valor
Velocidad inicial: v_0 (m/s)	
Velocidad en el punto A: v_A (m/s)	
Velocidad en el punto B: v_B (m/s)	
Velocidad en el punto C: v_C (m/s)	
Velocidad en el punto D: v_D (m/s)	
Tiempo transcurrido hasta el punto A: t_A	
Tiempo transcurrido hasta el punto B: t_B	
Tiempo transcurrido hasta el punto C: t_C	
Tiempo transcurrido hasta el punto D: t_D	
Altura inicial de la pelota desde la rampa 1: h_0	
Altura Máxima Movimiento Parabólico: $h_{máx}$	
Alcance Horizontal Movimiento Parabólico: R	

Tabla 3. Datos y resultados dados por el software – Tramo total.

C. Explique cada una de las gráficas de movimiento generadas por el simulador para el experimento planteado en el punto anterior (numeral B).

PRÁCTICA DEMOSTRATIVA N° 3
(MOVIMIENTO CIRCULAR)

1. INTRODUCCIÓN

La presente práctica involucra los conceptos principales referentes a la dinámica circular, la cual, como parte de la mecánica, permite el estudio de las condiciones que debe cumplir un cuerpo para que realice un movimiento circular. En la vida real existen muchos ejemplos de esta clase de movimiento, sin embargo, se deben distinguir algunas características importantes para determinar si una partícula está llevando a cabo un movimiento circular de tipo uniforme o de tipo uniformemente variado.

2. OBJETIVOS

- Validar los conceptos de aceleración radial y aceleración tangencial dentro de un sistema dinámico con movimiento circular.

- Observar el efecto de las fuerzas en un sistema de movimiento mixto.

3. MOVIMIENTO CIRCULAR UNIFORME

Un cuerpo que recorre una trayectoria circular o semicircular con rapidez constante estará llevando a cabo un movimiento circular uniforme (MCU). En este caso, aunque el valor de la rapidez siempre sea el mismo durante todo el movimiento, existe una aceleración dada por el cambio en la dirección del vector de velocidad de dicho objeto (recordar que la velocidad al ser una cantidad vectorial posee magnitud y dirección). Esta aceleración, denominada aceleración centrípeta, está representada por un vector que apunta hacia el centro del círculo y es perpendicular en todo momento a la trayectoria que realiza la partícula (ver figura 1).

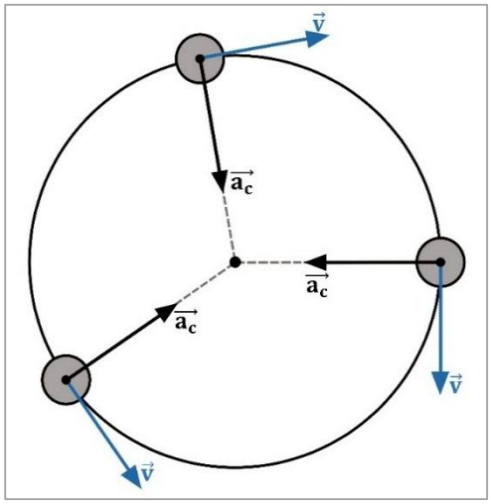


Figura 1. Aceleración centrípeta MCU.

En el MCU la aceleración total equivale a la aceleración radial o centrípeta (a_c) y se expresa matemáticamente de la siguiente forma:

$$a_c = \frac{v^2}{R} \quad \textbf{(1)}$$

donde v representa la rapidez con que se mueve el cuerpo y *r* es el radio de la trayectoria circular que este recorre. El tiempo requerido para recorrer el trayecto circular se denomina periodo (T) y se expresa como:

$$T = \frac{2\pi r}{v} \quad \textbf{(2)}$$

Donde el término $2\pi r$ representa el diámetro de la circunferencia que recorre el objeto y v es la rapidez con que este se mueve.

4. MOVIMIENTO CIRCULAR UNIFORMEMENTE VARIADO

Ahora bien, cuando el objeto se mueve con rapidez variable durante todo el recorrido se dice que está realizando un movimiento circular uniformemente variado. Este tipo de movimiento se caracteriza por que el vector de velocidad es variable tanto en magnitud como en dirección, lo que hace que la aceleración total cambie de un punto a otro. En este caso la aceleración tiene una componente adicional denominada aceleración tangencial (a_t), la cual, como su nombre lo

indica, es tangente a la trayectoria del movimiento que realiza la partícula (ver figura 2).

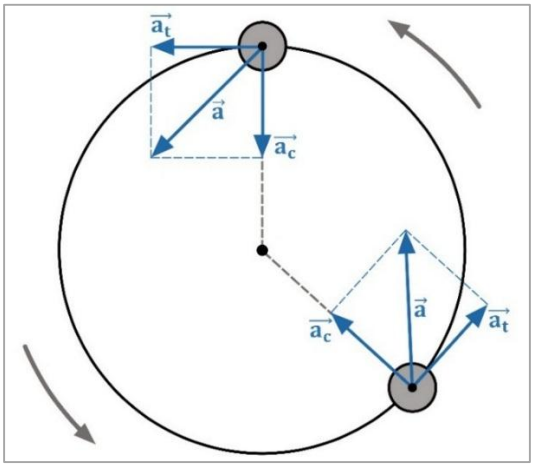


Figura 2. Aceleración total.

Matemáticamente la aceleración tangencial viene dada como:

$$a_t = \left|\frac{dv}{dt}\right| \tag{3}$$

Esta ecuación representa la variación de la rapidez del cuerpo a medida que transcurre el tiempo. Por su parte la aceleración total viene dada como la suma vectorial de las aceleraciones radial y tangencial respectivamente, como se ilustra en la figura 2. Tanto la suma vectorial como el valor escalar de la aceleración total se expresan como:

$$\vec{a} = \vec{a_r} + \vec{a_t} \tag{4}$$

$$a = \sqrt{a_r{}^2 + a_t{}^2} \tag{5}$$

donde a_r representa la componente radial de la aceleración total o aceleración centrípeta.

Un aspecto importante que debe considerarse dentro de la dinámica circular son las fuerzas que actúan sobre un cuerpo que realiza un movimiento de este tipo. En la figura 3 se puede observar el caso en el que un objeto de masa *m* se mueve

dentro de una superficie circular vertical con rapidez variable durante todo el recorrido.

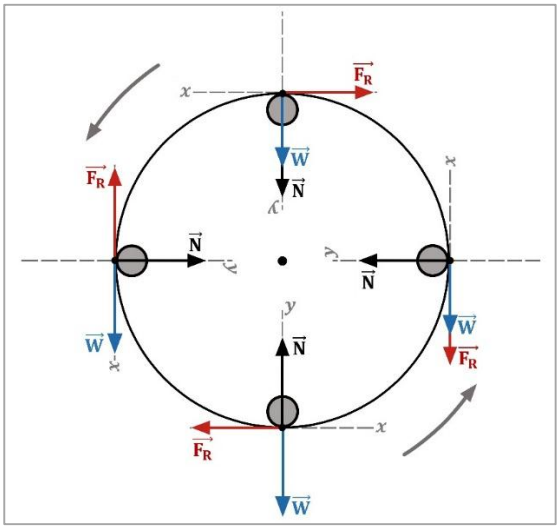


Figura 3. Fuerzas presentes dentro del movimiento circular.

De acuerdo a la figura 3 las fuerzas presentes en este sistema son las siguientes: la fuerza normal (N) ejercida por la superficie de contacto, la fuerza de fricción de dicha superficie (F_R) y el peso del objeto ($W = mg$). La resultante de todas las fuerzas que actúan en dirección radial en cada instante de tiempo se denomina fuerza centrípeta (F_c)**.** De esta forma, partiendo del diagrama de cuerpo libre del objeto en movimiento ubicado en un punto cualquiera de la trayectoria circular se tienen las fuerzas que actúan en dirección radial (eje y) y tangencial (eje x). Por ejemplo, si el piloto se ubica en un punto entre 0 y 2π se tiene:

$$\sum F_x = -mg\sin\theta - F_R = ma_t \qquad \textbf{(6)}$$

$$\sum F_y = N - mg\cos\theta = ma_r \qquad \textbf{(7)}$$

En donde los términos $mgsin\theta$ y $mgcos\theta$ corresponden a las componentes tangencial y radial del peso respectivamente. Partiendo de las ecuaciones (6) y (7) se obtienen las expresiones generales de la aceleración tangencial y radial para todo el movimiento circular:

$$a_t = -g\sin\theta - \frac{\mu N}{m} \quad \textbf{(8)}$$

$$a_r = \frac{N}{m} - g\cos\theta \quad \textbf{(9)}$$

Los términos m y μ corresponden a la masa del objeto que realiza el movimiento y al coeficiente de fricción de la superficie de contacto respectivamente. Las ecuaciones del movimiento rectilíneo se pueden asociar con las del movimiento circular de la siguiente forma:

$$v = v_0 + at \rightarrow v = v_0 + a_t t \quad \textbf{(10)}$$

$$x = v_0 t + \frac{at^2}{2} \rightarrow x = v_0 t + \frac{a_t t^2}{2} \quad \textbf{(11)}$$

$$2a_t x = {v_f}^2 - {v_0}^2 \quad \textbf{(12)}$$

donde v y v_0 corresponden a la velocidad final e inicial del cuerpo en el movimiento circular, x es la longitud recorrida por el objeto y t es el tiempo empleado en realizar este desplazamiento.

5. DESARROLLO DE LA PRÁCTICA

Para desarrollar la práctica de movimiento circular el estudiante debe tener acceso a un computador con conexión a internet, el cual debe contar con el explorador "Google Chrome" dentro de sus herramientas de navegación.

Nota: Antes de correr la aplicación tenga en cuenta las siguientes recomendaciones:

- La resolución del equipo donde se va a realizar la simulación debe ser ajustada a un valor de 1366 x 768.

- El tamaño de zoom del navegador "Google Chrome" debe estar en un valor de 100%.

- La página web donde se aloja la aplicación debe estar totalmente maximizada durante todo el desarrollo de la práctica.

PROCEDIMIENTO:

6. Ingrese a la siguiente página web: http://www.unicauca.edu.co/experimentos_mecanica/aplicacion.html para tener acceso a la aplicación web que contiene las prácticas virtuales relacionadas con la temática del curso, como se muestra en la figura 4.

Figura 4. Interfaz principal de la aplicación web.

7. Para ingresar a la práctica de movimiento circular haga click en el respectivo enlace, dentro de la "**Lista de prácticas**", como se muestra en la figura 5. A continuación el sistema desplegará la interfaz mostrada en la figura 6.

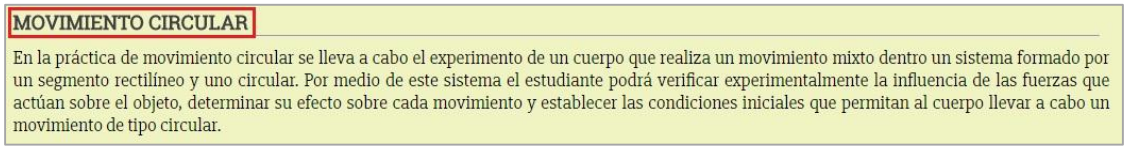


Figura 5. Enlace para la práctica de movimiento circular.

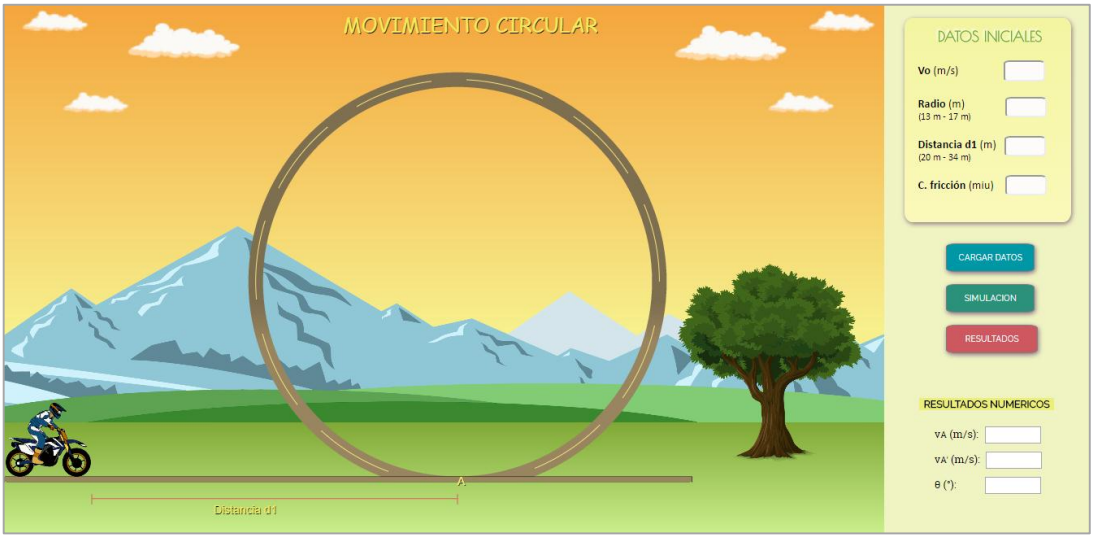


Figura 6. Interfaz principal – Movimiento Circular.

EXPERIMENTO 1:

8. En los campos respectivos del cuadro de **"Datos Iniciales"** (ver figura 7) ingrese los siguientes valores:

- Velocidad inicial $\boldsymbol{v_0}$: Cualquier valor entre 33 y 50 m/s.
- Radio de la circunferencia $r = 15$ metros.
- Distancia o longitud del tramo recto d_1: Cualquier valor entre 20 y 34 metros.
- Coeficiente de fricción de la superficie $\mu = 0$.

Consigne los valores fijados para cada parámetro en la tabla 1.

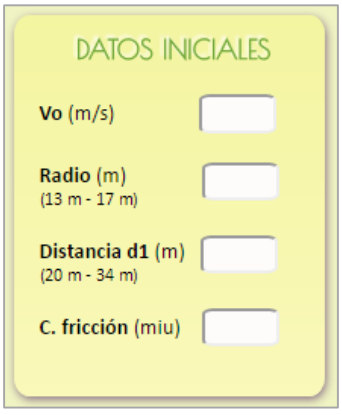


Figura 7. Datos iniciales.

Datos	Valor
Velocidad (v_0)	
Radio (r)	
Distancia tramo recto (d_1)	
C. Fricción (μ)	

Tabla 1. Datos de entrada.

Nota: El valor del coeficiente de fricción ingresado corresponde al de toda la superficie de contacto.

9. Cargue los datos ingresados con el botón **"Cargar Datos"** e inicie la simulación haciendo click en el botón **"Simulación"** (ver figura 8).

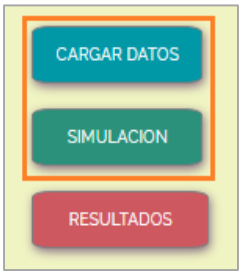


Figura 8. Botón de cargar datos y de inicio de simulación.

El software lleva a cabo la simulación del recorrido del motociclista a través del segmento rectilíneo y del segmento circular.

10. Cuando se haya completado la simulación haga click en el botón **"Resultados"** (ver figuras 9 y 10). El software desplegará algunos resultados numéricos tales como: v_A (velocidad inicial de la moto en la trayectoria circular), $v_A{}'$ (velocidad final de la moto en la trayectoria circular) y θ (ángulo alcanzado por la moto en su recorrido dentro del tramo circular). Estos resultados deben ser consignados en la tabla 2.

De forma similar el sistema mostrará las siguientes gráficas relacionadas con el movimiento total de la motocicleta: velocidad y aceleración contra tiempo dentro del tramo rectilíneo y velocidad, fuerza normal, aceleración radial, aceleración tangencial y aceleración total contra teta (θ) para el tramo circular (ver figura 11). Consigne los resultados numéricos de la simulación en la tabla 2 y guarde pantallazos de las gráficas generadas en la misma.

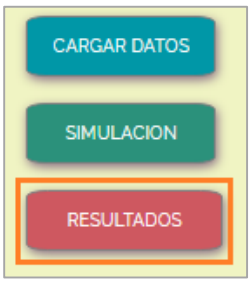


Figura 9. Botón de despliegue de resultados.

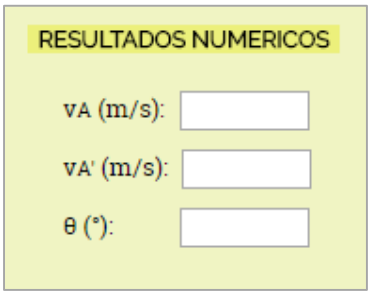


Figura 10. Despliegue de resultados numéricos.

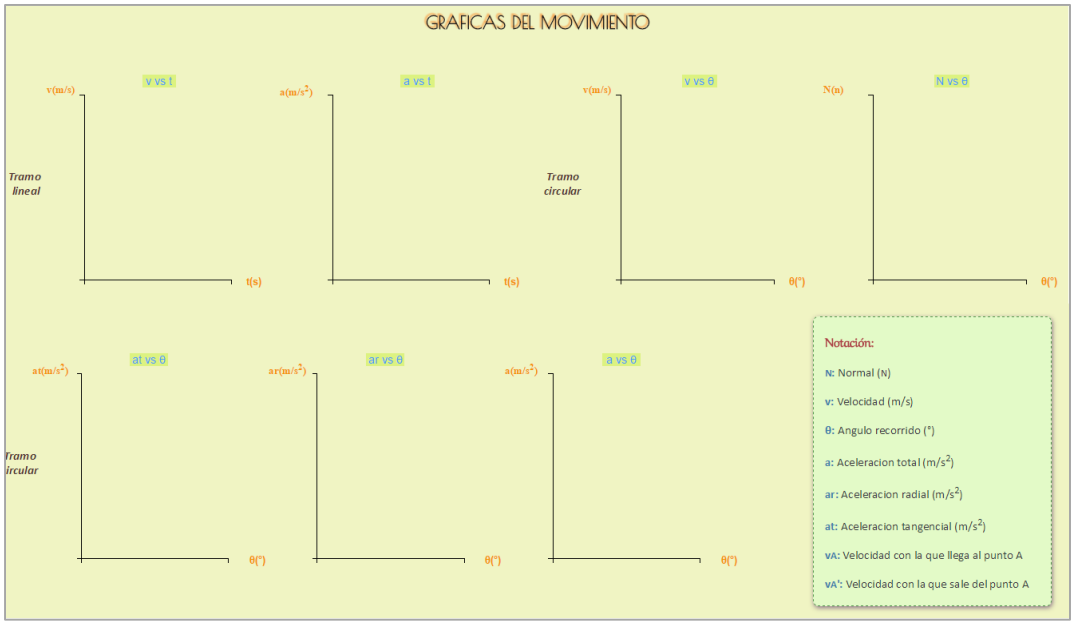


Figura 11. Gráficas de movimiento.

Datos	Valor
Velocidad al inicio del tramo circular: $v_A(m/s)$	
Velocidad al final del tramo circular: $v_A{}'(m/s)$	
Recorrido circular: $\theta(°)$	

Tabla 2. Resultados numéricos de la simulación.

11. PRUEBA DE CONOCIMIENTO 1:

Con el experimento realizado a partir de los datos de la tabla 1 responda:

A. ¿Cuál debe ser la expresión matemática para la velocidad tangencial de la moto dentro del círculo?

B. Con los datos obtenidos encuentre la velocidad que el motociclista alcanza cuando ha recorrido 45° dentro del tramo circular.

C. Realice el cálculo de la fuerza centrípeta que el motociclista siente durante el trayecto circular.

D. Describa cada una de las gráficas de movimiento obtenidas a partir de la simulación en la prueba 1.

EXPERIMENTO 2:

12. Reescriba sobre los campos correspondientes los siguientes datos iniciales:

- Velocidad inicial $\boldsymbol{v_0}$: Cualquier valor menor o igual a 33 m/s.
- Radio de la circunferencia $r = 17$ metros.
- Distancia o longitud del tramo recto d_1: Cualquier valor entre 20 y 34 metros.
- Coeficiente de fricción de la superficie μ: Cualquier valor entre 0.2 y 0.9.

Consigne los nuevos valores fijados para cada parámetro en la tabla 3:

Datos	Valor
Velocidad (v_0)	
Radio (r)	
Distancia tramo recto (d_1)	
C. Fricción (μ)	

Tabla 3. Datos de entrada.

13. Cargue los nuevos datos con el botón **"Cargar Datos"** y realice la simulación dando click en el botón **"Simulación"**.

14. Haga click en el botón **"Resultados".** Consigne los resultados numéricos dados por la herramienta software en la tabla 4. De igual forma, tome pantallazos de las gráficas generadas en la simulación.

Datos	Valor
Velocidad al inicio del tramo circular: $v_A(m/s)$	
Velocidad al final del tramo circular: $v_A{}'(m/s)$	
Recorrido circular: $\theta(°)$	

Tabla 4. Resultados numéricos.

15. PRUEBA DE CONOCIMIENTO 2:

Con el experimento realizado a partir de los datos de la tabla 3 responda:

A. Si el coeficiente de fricción (μ) no es cero, ¿de qué manera afecta al movimiento circular?

B. ¿Cómo afecta al movimiento circular la variación de la longitud inicial (d_1)?

C. ¿Para un movimiento de tipo circular es conveniente una velocidad baja o una velocidad alta al inicio del mismo?

D. Describa cada una de las gráficas de movimiento obtenidas a partir de la simulación en el experimento 2 y realice una comparación con las gráficas obtenidas en el experimento 1.

PRÁCTICA DEMOSTRATIVA N° 4
(LEYES DEL MOVIMIENTO)

1. INTRODUCCIÓN

Los conceptos tratados en capítulos anteriores conforman una parte de la mecánica conocida como cinemática, la cual permite el estudio de las características que describen los diferentes tipos de movimiento pero sin considerar las causas que los originan. De esta parte se encarga la dinámica, otra rama de la mecánica que se centra en establecer la relación entre el movimiento de un objeto y las fuerzas que actúan sobre él. Para estudiar los principios de la dinámica es necesario recurrir a los conceptos de masa, fuerza y aceleración, términos mediante los cuales Isaac Newton formuló sus conocidas leyes del movimiento.

2. OBJETIVO

Validar la relación que existe entre la fuerza neta aplicada sobre un objeto, su masa y la aceleración producida por dicha fuerza.

3. CONCEPTO DE FUERZA

La fuerza es una magnitud física capaz de alterar el estado de reposo o de movimiento de un cuerpo como producto de la interacción con otro cuerpo o con el medio que lo rodea. Físicamente la fuerza hace parte de las denominadas cantidades vectoriales, por lo tanto su descripción dentro de cualquier sistema se hace por medio de su magnitud, dirección y sentido. Dependiendo del modo de aplicación existen dos tipos de fuerzas: fuerzas de contacto y fuerzas de largo alcance. Una fuerza de contacto es aquella que, como su nombre lo indica, requiere del contacto directo entre dos cuerpos para ser aplicada. A este grupo pertenecen la fuerza normal, la fuerza de fricción y la fuerza de tensión.

- **Fuerza normal ($\vec{N}$):** Es la fuerza que ejerce toda superficie sobre un cuerpo que está en contacto con ella. Se denomina normal porque esta fuerza siempre actúa de manera perpendicular a la superficie (ver figura 1).

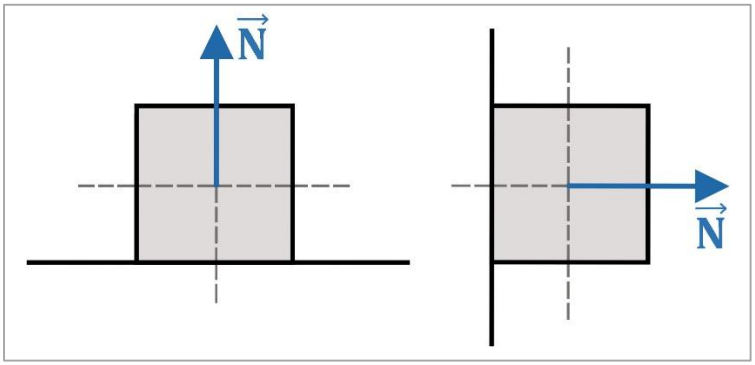


Figura 1. Fuerza normal.

- **Fuerza de fricción ($\vec{F}_R$):** Es la fuerza que ejerce una superficie sobre un cuerpo que se mueve a través de ella. La dirección de esta fuerza siempre es opuesta al movimiento del objeto y actúa de forma paralela a la superficie de contacto (ver figura 2).

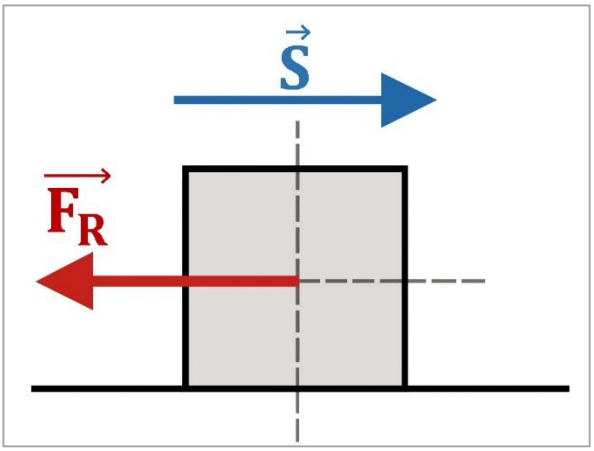


Figura 2. Fuerza de fricción.

Matemáticamente la fuerza de fricción viene expresada como:

$$\vec{F_R} = \mu \vec{N} \qquad \textbf{(1)}$$

donde μ se conoce como el coeficiente de fricción de la superficie sobre la que se desliza el objeto y $\vec{N}$ es la fuerza normal ejercida por la misma superficie.

Las fuerzas denominadas de largo alcance son aquellas que actúan sobre cuerpos que están separados una determinada distancia. Ejemplos de este tipo de fuerzas son la fuerza de atracción gravitacional, la fuerza eléctrica entre dos cargas, etc.

Cuando se tiene un cuerpo en reposo o moviéndose con velocidad constante se dice que este se encuentra en equilibrio. Dicho de otra manera, para que un cuerpo se encuentre en equilibrio no deben haber fuerzas actuando sobre él, y si las hay, la fuerza resultante o fuerza neta debe ser igual a cero (cada componente de la fuerza debe ser cero).

4. SEGUNDA LEY DE NEWTON

Es una de las leyes fundamentales de la llamada mecánica clásica mediante la cual se establece la relación básica entre fuerza y movimiento. La segunda ley de Newton expresa que la fuerza neta aplicada sobre un cuerpo equivale a la masa de este multiplicada por la aceleración que se produce. De esta manera, cuando una fuerza neta actúa sobre un objeto en reposo generando un movimiento, la aceleración que se produce tiene la misma dirección que la fuerza aplicada (ver figura 3). Dentro de esta ley solo se consideran fuerzas externas ejercidas por uno o varios cuerpos sobre otro dentro de un mismo entorno y es válida solo para marcos de referencia inerciales, es decir, aquellos donde se cumple la primera ley de Newton.

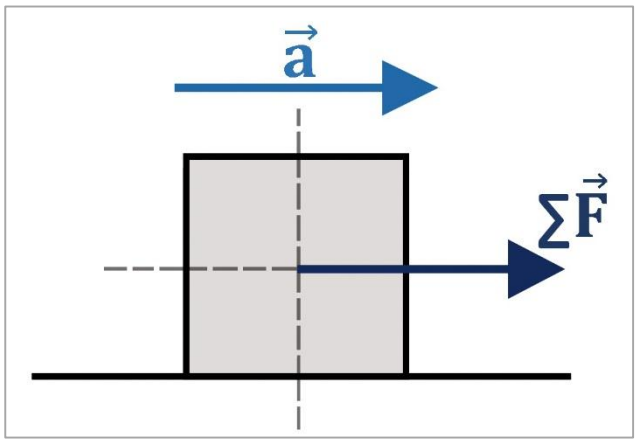


Figura 3. Segunda ley de Newton.

En forma vectorial la segunda ley de Newton viene dada como:

$$\sum \vec{F} = m\vec{a} \qquad \textbf{(2)}$$

Donde $\vec{F}$ representa la fuerza neta aplicada, m es la masa del cuerpo y $\vec{a}$ es la aceleración producida por la acción de la fuerza.

5. SISTEMA MASA – RESORTE

Es un sistema compuesto por un objeto de masa m que se desplaza sobre una superficie y que está unida a un punto fijo por medio de un muelle o resorte como se muestra en la figura 4.

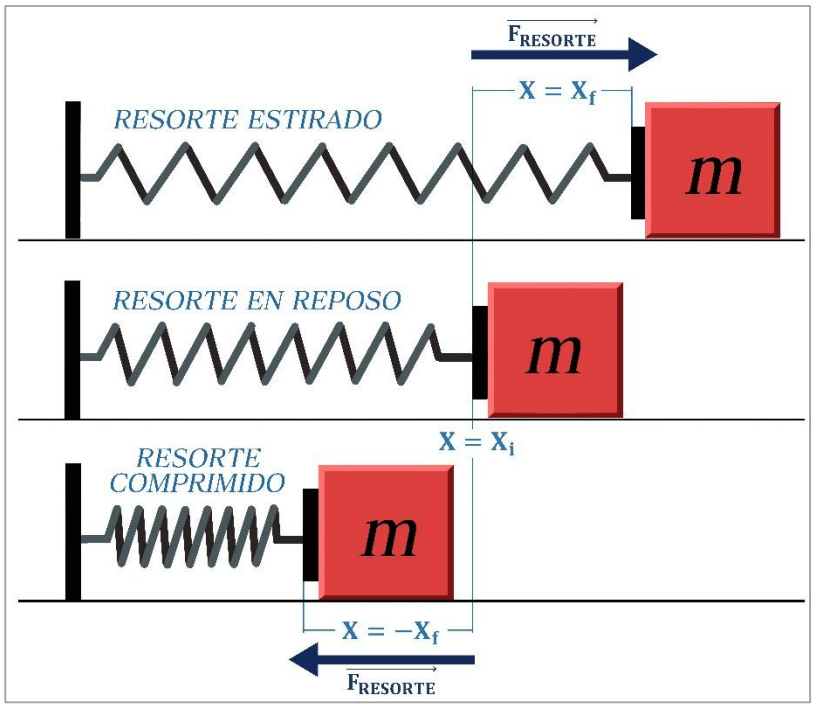


Figura 4. Sistema Masa-Resorte.

Cuando el resorte se deforma respecto de su posición inicial en x como producto de la acción de una fuerza externa, aparece instantáneamente una fuerza $\overrightarrow{F_R}$ producida por el resorte sobre el elemento que está ejerciendo dicha fuerza externa y cuya magnitud es:

$$F = -KX_R \qquad \textbf{(3)}$$

donde X_R representa el valor del desplazamiento del resorte respecto de su posición de reposo ($x = 0$), *k* es una constante propia de cada resorte denominada constante elástica, que indica el grado de dureza o rigidez de estos elementos y el signo negativo indica que la fuerza del resorte siempre apunta en dirección opuesta al desplazamiento generado por la fuerza externa. La ecuación (3) se conoce en física como *Ley de Hooke*. Se debe tener en cuenta que esta expresión es válida para pequeños desplazamientos del resorte ya que si este se estira

demasiado puede sufrir una deformación que le impida recuperar su forma o estado natural.

6. MOVIMIENTO SOBRE UN PLANO INCLINADO

Para analizar el movimiento de un cuerpo sobre un plano inclinado se deben considerar las fuerzas aplicadas sobre tal objeto. De acuerdo a la figura 5 dichas fuerzas son las siguientes: El peso del objeto en dirección vertical ($\vec{P} = m\,\vec{g}$), la fuerza normal o fuerza de reacción de la superficie ($\vec{N}$) y la fuerza de fricción o de rozamiento de la superficie ($\vec{F}_R$).

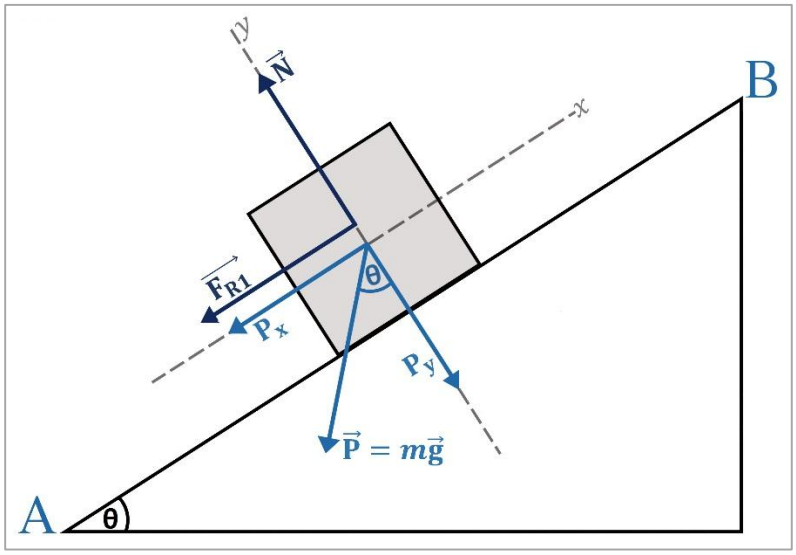


Figura 5. Movimiento sobre un plano inclinado.

Al descomponer cada fuerza y aplicando la segunda ley de Newton se puede llegar a las siguientes expresiones:

$$\sum F_x = -P_x - F_R = ma_x \qquad \textbf{(4)}$$

$$\sum F_x = N - P_y = 0 \qquad \textbf{(5)}$$

$$a_x = -g\left(\sin\theta + \mu\cos\theta\right) \qquad \textbf{(6)}$$

En donde a_x hace referencia a la aceleración del bloque en la dirección x. Dado que no hay movimiento del bloque en dirección y se tiene que $a_y = 0$.

7. DESARROLLO DE LA PRÁCTICA

Para desarrollar la práctica de movimiento en dos dimensiones el estudiante debe tener acceso a un computador con conexión a internet, el cual debe contar con el explorador “Google Chrome” dentro de sus herramientas de navegación.

Nota: Antes de correr la aplicación tenga en cuenta las siguientes recomendaciones:

- La resolución del equipo donde se va a realizar la simulación debe ser ajustada a un valor de 1366 x 768.

- El tamaño de zoom del navegador “Google Chrome” debe estar en un valor de 100%.

- La página web donde se aloja la aplicación debe estar totalmente maximizada durante todo el desarrollo de la práctica.

➢ DESCRIPCIÓN DEL SISTEMA:

La práctica de leyes del movimiento está basada en el sistema mostrado en la figura 6, el cual está formado por un plano inclinado con ángulo de elevación variable y un lado fijo de 7 metros de longitud. Sobre este se encuentra ubicado un pequeño cuerpo de masa fija unido a un resorte con constante elástica definida. El sistema está diseñado para que el objeto de masa m se deslice sobre el plano inclinado gracias a la fuerza que el resorte le imprime, como resultado de la compresión a la que es sometido. La distancia de compresión del resorte debe ser suficiente para que el bloque atraviese el plano inclinado y realice una trayectoria de tipo parabólica hasta llegar a un punto o blanco determinado ubicado a una distancia (previamente definida) medida desde el final de la base del plano inclinado.

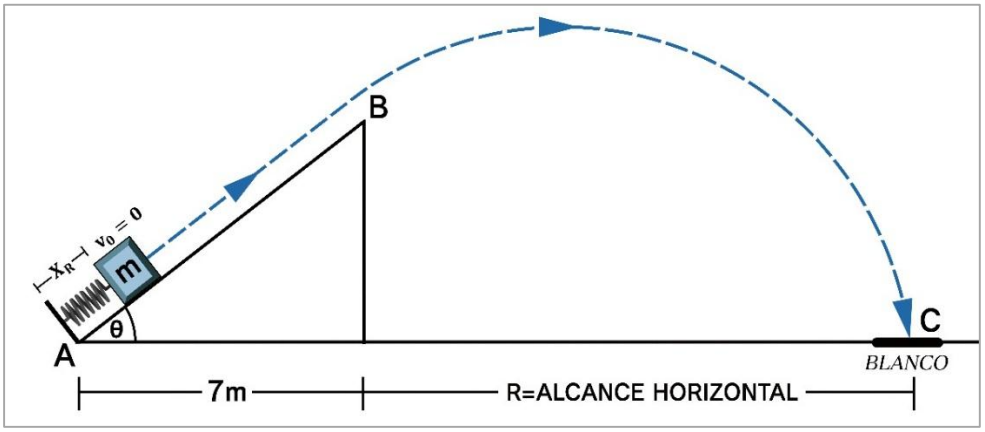


Figura 6. Sistema general.

- **PROCEDIMIENTO MATEMÁTICO:**

8. Con base en el sistema de la figura 6 fije el valor del alcance horizontal en $R = 10m$ y escoja un valor de coeficiente de fricción μ del plano inclinado entre 0.1 y 0.6, además de un valor para el ángulo del plano inclinado entre $28°$ y $35°$, para con ello completar la tabla de datos del sistema (tabla 1). Con estos datos determine matemáticamente la longitud de compresión del resorte (X_R) de modo que el bloque pueda ser impulsado a través del plano inclinado, impactando directamente en el orificio del blanco (considere el blanco como un punto).

Datos	Valor
Masa del bloque unido al resorte (Kg)	0.4
Constante elástica K (N/m)	4000
Angulo de elevación del plano inclinado θ (°)	
Coeficiente de fricción del plano inclinado μ	
Distancia al blanco R (m)	10

Tabla 1. Datos del sistema.

Nota: Considere que el resorte comprimido no aporta longitud para la trayectoria inicial del movimiento.

9. A partir de las ecuaciones planteadas en el punto anterior calcule los siguientes valores: velocidad del bloque en el punto B, tiempo que demora el bloque en recorrer el plano inclinado (tramo A-B), altura de la rampa o plano inclinado, aceleración del bloque en la rampa con y sin resorte, altura máxima alcanzada

en el movimiento parabólico, tiempo de vuelo, velocidad en el punto C. Consigne estos resultados en la tabla 2:

Datos	Valor
Velocidad en el punto B: $v_B(m/s)$	
Tiempo tramo A-B: $t_{AB}(s)$	
Altura de la rampa (m): $h_{rampa}(m)$	
Aceleración sistema masa-resorte: $a_{Resorte}(m/s^2)$	
Aceleración bloque sin resorte en la rampa: $a_x(m/s^2)$	
Altura máxima: $h_{máx}(m)$	
Tiempo de vuelo tramo B-C: $t_v(s)$	
Velocidad en el punto: $v_C(m/s)$	

Tabla 2. Valores calculados.

Con el valor de compresión del resorte hallado el siguiente paso es corroborar si los cálculos realizados garantizan que el bloque caiga exactamente dentro del blanco escogido.

➢ SIMULACIÓN

10. Ingrese a la siguiente página web: http://www.unicauca.edu.co/experimentos_mecanica/aplicacion.html para tener acceso a la aplicación web que contiene las prácticas virtuales relacionadas con la temática del curso, como se muestra en la figura 7.

Figura 7. Interfaz principal de la aplicación web.

11. Para ingresar a la práctica de leyes del movimiento haga click en el respectivo enlace, dentro de la "**Lista de prácticas**", como se muestra en la figura 8. El sistema desplegará la interfaz mostrada en la figura 9.

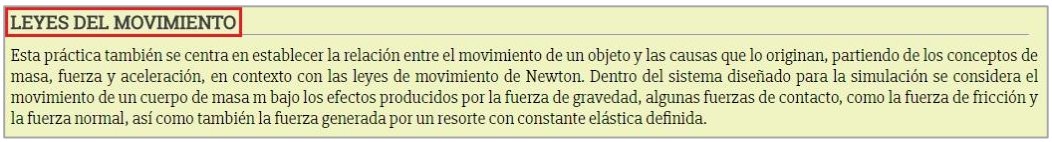


Figura 8. Enlace para la práctica de leyes del movimiento.

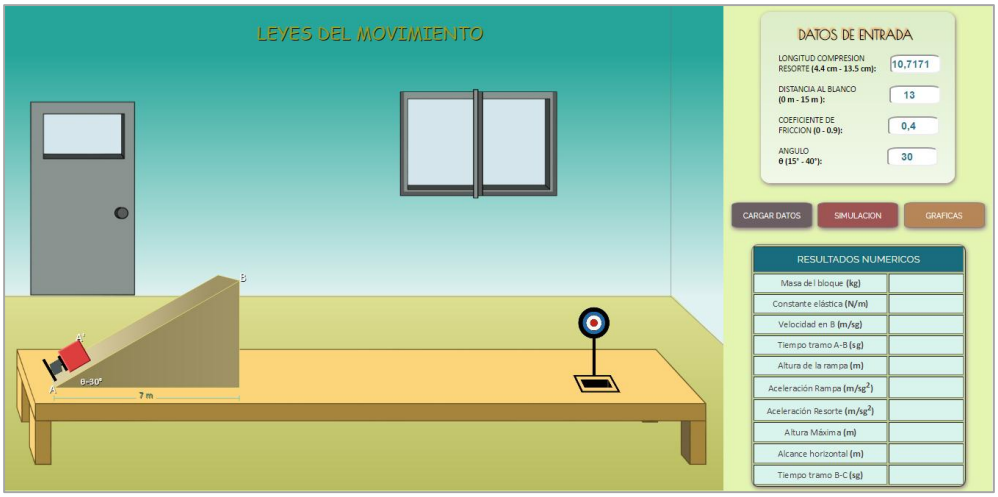


Figura 9. Interfaz principal – Leyes del movimiento.

12. En los campos respectivos del cuadro de **"datos de entrada"** (ver figura 10) ingrese el valor de longitud de compresión del resorte X_R calculada en el numeral 1, al igual que los datos iniciales de distancia del blanco R, coeficiente de fricción de la rampa μ y ángulo de elevación θ de la tabla 1.

Figura 10. Datos de entrada del simulador.

13. Cargue los datos ingresados con el botón **"cargar datos"** e inicie la simulación haciendo click en el botón **"simulación"** (ver figura 11). El software realizará la simulación del movimiento del bloque impulsado por el resorte.

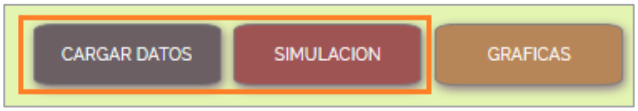


Figura 11. Botón de cargar datos e inicio de simulación.

A medida que transcurre la simulación el software irá desplegando los diferentes resultados numéricos asociados con el movimiento del bloque, de modo que al finalizar se obtendrá una tabla con el resumen de los valores más relevantes calculados por la herramienta (ver figura 12). Consigne estos resultados numéricos en la tabla 3.

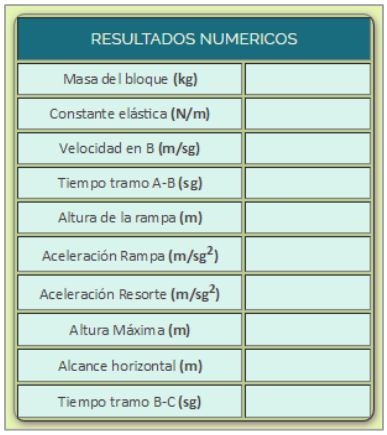


Figura 12. Resultados numéricos desplegados por el software.

Datos	Valor
Velocidad en el punto B: $v_B(m/s)$	
Tiempo tramo A-B: $t_{AB}(s)$	
Altura de la rampa (m): $h_{rampa}(m)$	
Aceleración bloque con resorte en la rampa: $a_{Resorte}(m/s^2)$	
Aceleración bloque sin resorte en la rampa: $a_x(m/s^2)$	
Altura máxima: $h_{máx}(m)$	
Tiempo de vuelo tramo B-C: $t_v(s)$	
Velocidad en el punto: $v_C(m/s)$	

Tabla 3. Resultados numéricos dados por el software.

14. Compruebe si el bloque cayó justo en el blanco, de ser así pase al punto 9, sino compare los valores hallados matemáticamente con los resultados dados por la herramienta y determine cuál fue su error. Recalcule los datos errados de modo que pueda encontrar un nuevo valor de longitud de compresión del resorte (X_R), cargue nuevamente los datos de entrada en el simulador y compruebe si se logró el objetivo de dar en el blanco. Repita este procedimiento las veces que sea necesario hasta lograr que el bloque caiga exactamente en el punto deseado. Dentro de la tabla 2 reescriba sobre los valores errados los resultados corregidos que encontró al realizar nuevamente el proceso matemático.

15. Cuando haya dado en el blanco obtenga las curvas de movimiento del bloque haciendo click en el botón "Gráficas" (ver figura 13). El software le mostrará los resultados gráficos correspondientes a: desplazamiento en x (x vs t), desplazamiento en y (y vs t), magnitud de la velocidad del bloque (v vs t) y magnitud de la aceleración del bloque (a vs t), como se muestra en la figura 14.

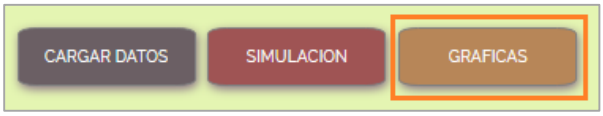


Figura 13. Botón de despliegue de gráficas.

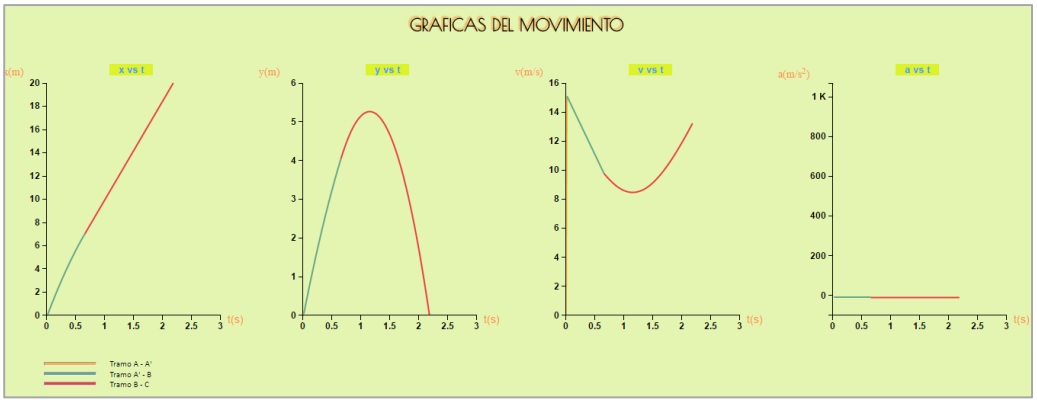


Figura 14. Gráficas desplegadas por el simulador.

16. PRUEBA DE CONOCIMIENTO:

A. Explique la curva de x vs t de acuerdo al movimiento de la masa.

B. Explique la curva de y vs t de acuerdo al movimiento de la masa.

C. Relacione la curva de velocidad con la de aceleración y explique cada tramo del movimiento según estas gráficas.

D. De acuerdo a las ecuaciones encontradas, si se duplica la masa, ¿cuál debería ser la compresión del resorte para dar en el mismo blanco?

E. De las ecuaciones del movimiento, si el ángulo de la rampa se parte a la mitad, ¿cuál debería ser la compresión del resorte para dar en el blanco?

PRÁCTICA DEMOSTRATIVA N° 5
(TRABAJO Y ENERGÍA)

1. INTRODUCCIÓN

Hasta el momento el análisis de los sistemas que involucran fuerzas y cuerpos en movimiento ha sido posible gracias a las leyes y conceptos dados en conjunto por la dinámica y la cinemática, sin embargo, existe una amplia gama de fenómenos físicos para los cuales las soluciones planteadas hasta este punto no son suficientes a la hora de abordar problemas que los involucre. Ante esto se requiere una alternativa de solución que permita afrontar con éxito los inconvenientes relacionados con estos nuevos fenómenos y es aquí donde toma gran importancia el concepto de trabajo y energía.

2. OBJETIVO

Afianzar el concepto de trabajo y energía por medio de una herramienta software que permite simular un sistema en el cual se observe el cambio de la energía cinética y potencial de un cuerpo como la variación del trabajo realizado por las fuerzas que actúan sobre él, de modo que se pueda utilizar este principio para la solución de problemas de mecánica.

3. TRABAJO REALIZADO POR UNA FUERZA CONSTANTE

El trabajo W realizado por una fuerza *F* constante (en magnitud y dirección) aplicada sobre un cuerpo de masa m se expresa matemáticamente por medio de la siguiente expresión:

$$W = Fs\cos\theta \qquad \textbf{(1)}$$

donde s representa el desplazamiento del cuerpo debido a la acción de la fuerza y θ corresponde a la dirección en la que dicha fuerza ha sido aplicada. El término $cos\theta$ indica la componente de F en la dirección del desplazamiento de la partícula. Las unidades del trabajo vienen expresadas en Newtons por metro (N.m) o Joules (J). Con base en lo expresado en la ecuación (1) se pueden tener los siguientes casos:

- Cuando $\boldsymbol{\theta = 0°}$, F se ha aplicado paralelamente al movimiento del cuerpo y $W = Fs$.

- Cuando $\boldsymbol{\theta = 90°}$, F ha sido aplicada perpendicularmente al movimiento del cuerpo y $W = 0$ (No se realiza trabajo).

- Cuando $\boldsymbol{0° < \theta < 90°}$, F tiene una componente en la misma dirección del desplazamiento de la partícula, por lo tanto el trabajo W es positivo.

- Cuando $\boldsymbol{90° < \theta < 180°}$, F tiene una componente opuesta a la dirección del desplazamiento de la partícula, por lo tanto el trabajo W es negativo.

- Cuando $\boldsymbol{s = 0}$, el trabajo W realizado es cero, ya que la fuerza F no produce ningún desplazamiento del cuerpo.

4. ENERGÍA CINÉTICA

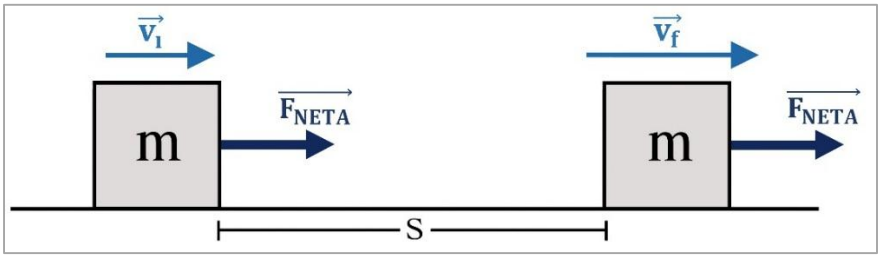


Figura 1. Movimiento de una partícula por la acción de una fuerza.

Una fuerza neta que actúa sobre un cuerpo produce una aceleración durante el movimiento de este, por lo tanto, si se evalúa el cambio en la velocidad de la partícula durante su desplazamiento se puede determinar el valor de esta aceleración (ver figura 1). Partiendo de la definición general de trabajo realizado por una fuerza constante *F* aplicada sobre una partícula de masa m junto con la segunda ley de Newton se tiene que:

$$W_{Neto} = (ma)s \qquad \textbf{(2)}$$

Retomando las expresiones de aceleración y desplazamiento en términos de velocidad y tiempo tratadas en el capítulo de movimiento en una dimensión y al remplazarlas en la ecuación (2) se obtiene una expresión matemática que relaciona el trabajo W realizado por la fuerza neta, como función de las

velocidades inicial y final del cuerpo de masa m sobre el cual se ha aplicado dicha fuerza:

$$W_{Neto} = \frac{1}{2}mv_f^{\,2} - \frac{1}{2}mv_i^{\,2} \quad \textbf{(3)}$$

El término $\frac{1}{2}\,mv^2$ de la ecuación (3) representa la energía cinética (K) del cuerpo asociada a su movimiento. De igual manera la ecuación (3) representa matemáticamente el **"Teorema del trabajo y energía",** el cual establece que el trabajo realizado por una fuerza neta aplicada sobre una partícula equivale al cambio de su energía cinética:

$$W_{Neto} = K_f - K_i = \Delta K \quad \textbf{(4)}$$

Las unidades de la energía cinética vienen dadas en Julios (J).

5. FUERZAS CONSERVATIVAS Y NO CONSERVATIVAS

Una fuerza es conservativa cuando el trabajo necesario para mover una partícula entre dos puntos no depende de la trayectoria que se escoja para unir dichos puntos. Algunas fuerzas que dependen de la posición de la partícula son conservativas, entre ellas están: la fuerza elástica, la fuerza electromagnética y la fuerza gravitacional, entre otras. Por otra parte, se conocen como fuerzas no conservativas o fuerzas disipativas aquellas en las que el trabajo realizado para mover un cuerpo entre dos puntos si dependen de la trayectoria que se tome. La característica principal de las fuerzas no conservativas es que al actuar dentro de cualquier sistema producen un cambio en la energía mecánica del mismo.

6. ENERGÍA POTENCIAL

De acuerdo a lo planteado anteriormente, el trabajo realizado por una fuerza conservativa no depende de la trayectoria ni de la rapidez con que se mueve el cuerpo sobre el cual se ha aplicado la fuerza, en este caso se debe considerar el trabajo como una variación de la energía, dada por el cambio de posición de la partícula dentro del espacio. A este tipo de energía se le conoce como energía potencial (U). Por lo tanto, para el caso de una fuerza conservativa como la gravitacional, el trabajo equivale a la variación de energía potencial (gravitacional)

dada por las diferentes posiciones verticales de un objeto respecto de la superficie terrestre, como lo muestra la figura 2.

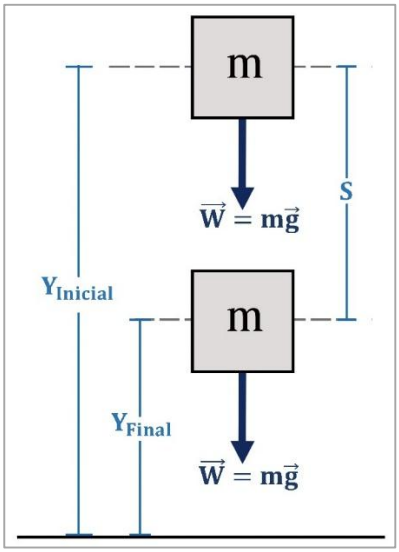


Figura 2. Energía potencial gravitacional.

Matemáticamente la energía potencial gravitacional (U_g) viene dado como:

$$U_g = mgy \qquad \textbf{(5)}$$

Donde m corresponde a la masa del cuerpo, g es la fuerza de gravedad y y es la altura respecto de la superficie terrestre desde donde se deja caer el cuerpo. El trabajo de la fuerza gravitacional, expresado como el cambio de energía potencial en los puntos inicial y final del desplazamiento de la partícula se define matemáticamente por medio de la siguiente expresión:

$$W_g = (mg)s = (-mg)\,j\,(y_f - y_i)\,j = mgy_i - mgy_f \qquad \textbf{(6)}$$

La ecuación (6) también puede escribirse de la siguiente forma:

$$W_g = U_i - U_f \qquad \textbf{(7)}$$

Las unidades de la energía potencial también vienen dadas en Julios (J).

7. CONSERVACIÓN DE LA ENERGÍA MECÁNICA

Cuando una fuerza conservativa produce el movimiento de una partícula, el trabajo que realiza es igual a la variación de la energía cinética del cuerpo (ecuación 4). Debido a que la fuerza es conservativa, el trabajo puede expresarse también como la reducción en la energía potencial de la partícula:

$$W = -\Delta U = U_i - U_f \quad \textbf{(8)}$$

La ecuación (8) indica también que el trabajo realizado por una fuerza conservativa se puede expresar como el valor negativo del cambio en la energía potencial asociado con dicha fuerza. Igualando las ecuaciones (4) y (7) se tiene:

$$\Delta K = -\Delta U \quad \textbf{(9)}$$

A partir de la ecuación (8) se deduce que la energía mecánica total se define como la suma de la energía cinética y la energía potencial:

$$E = K + U \quad \textbf{(10)}$$

La ecuación (10) da pie a una de las leyes fundamentales de la física, denominada **"ley de conservación de la energía"**, la cual establece que la energía mecánica total de un sistema permanece constante siempre y cuando sobre él actúen fuerzas conservativas:

$$K_i + U_i = K_f + U_f \quad \textbf{(11)}$$

8. DESARROLLO DE LA PRÁCTICA

Para desarrollar la práctica de movimiento en dos dimensiones el estudiante debe tener acceso a un computador con conexión a internet, el cual debe contar con el explorador "Google Chrome" dentro de sus herramientas de navegación.

Nota: Antes de correr la aplicación tenga en cuenta las siguientes recomendaciones:

- La resolución del equipo donde se va a realizar la simulación debe ser ajustada a un valor de 1366 x 768.

- El tamaño de zoom del navegador “Google Chrome” debe estar en un valor de 100%.

- La página web donde se aloja la aplicación debe estar totalmente maximizada durante todo el desarrollo de la práctica.

- De ser necesario actualice la página web por si se presenta algún problema.

PROCEDIMIENTO:

9. Ingrese a la siguiente página web: http://www.unicauca.edu.co/experimentos_mecanica/aplicacion.html para tener acceso a la aplicación web que contiene las prácticas virtuales relacionadas con la temática del curso, como se muestra en la figura 3.

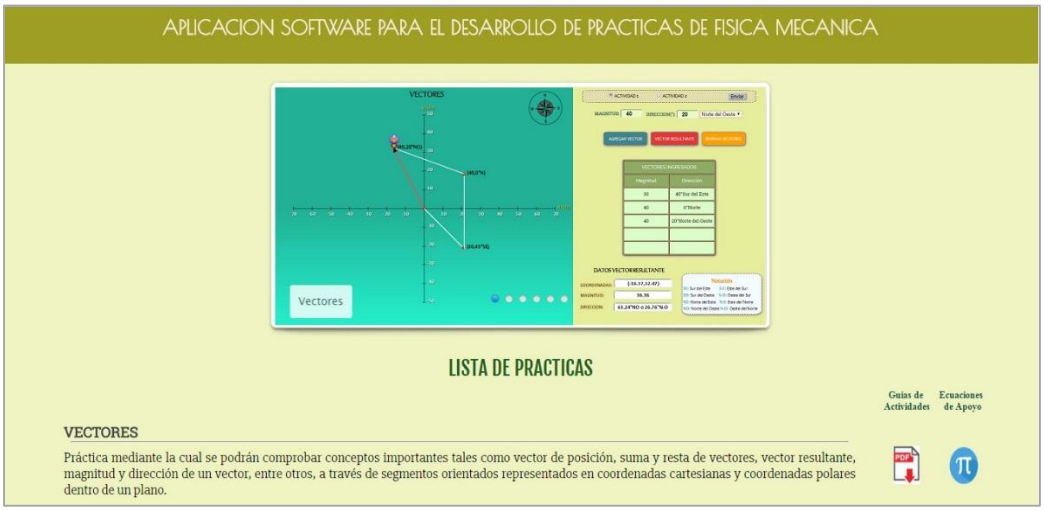


Figura 3. Interfaz principal de la aplicación web.

10. Para ingresar a la práctica de trabajo y energía haga click en el respectivo enlace, dentro de la “**Lista de prácticas**”, como se muestra en la figura 4. El sistema desplegará la interfaz mostrada en la figura 5.

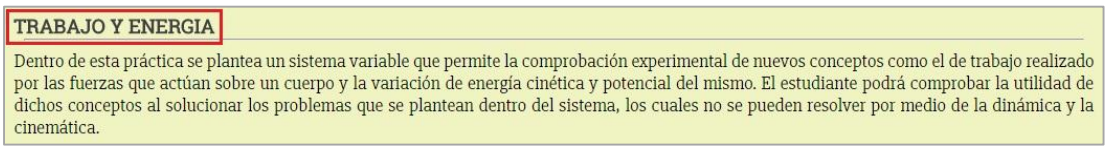


Figura 4. Enlace para la práctica de leyes del movimiento.

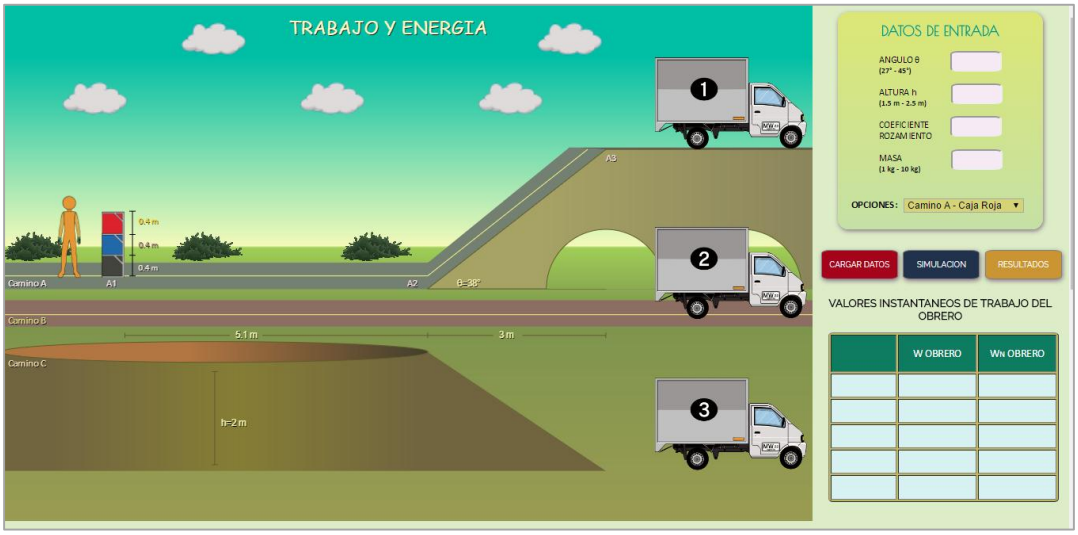


Figura 5. Interfaz principal – Trabajo y energía.

11. Dentro de la aplicación vaya al cuadro de **"datos de entrada"** e ingrese los siguientes datos al simulador (ver figura 6): ángulo de elevación del plano inclinado ascendente (valores entre 27° y 45°), altura del plano inclinado descendente (valores entre 1.5m y 2.5m), coeficiente de fricción para todas las superficies (valores entre 0 y 0.9), masa de las cajas (valores de 1Kg a 10Kg).

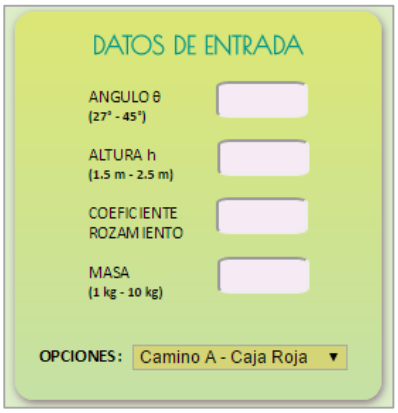


Figura 6. Interfaz principal – Trabajo y energía.

12. Con los datos de entrada fijados vaya a la pestaña de **"Opciones"**. Dentro de estas opciones el estudiante tendrá la posibilidad de escoger el camino que seguirá el obrero y la caja que se desea transportar, con el fin de ser llevada desde el punto de partida hasta cualquiera de los camiones ubicados al final de cada trayecto (ver figura 7).

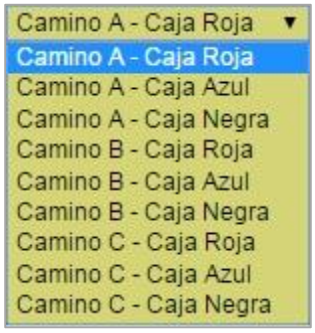


Figura 7. Opciones de simulación.

Los caminos que se pueden escoger tienen las siguientes características:

- **Camino A:** consta de un tramo rectilíneo de 5.1 metros y un plano inclinado ascendente de longitud variable que depende del valor de θ que se ingrese.

- **Camino B:** Corresponde a todo el segmento rectilíneo. Tiene una longitud de 8.1 metros.

- **Camino C:** consta de un tramo rectilíneo de 5.1 metros y un plano inclinado descendente de longitud variable que depende del valor de altura h que se ingrese.

Por su parte las cajas están ubicadas a diferentes alturas respecto del suelo:

- **Caja negra:** a 0 metros del suelo.

- **Caja azul:** a 0.4 metros del suelo.

- **Caja roja:** a 0.8 metros del suelo.

Nota: La altura a la que el obrero transporta las cajas durante el tramo rectilíneo de los tres caminos siempre es de 0.8 metros. El punto en el que el obrero deposita las cajas dentro del camión también está ubicado a 0.8 metros del suelo.

13. Escoja la opción **"Camino A – Caja Roja"** para realizar el experimento en que el obrero toma la caja roja, transita el camino A y la deposita en el camión 1 (ver figura 8).

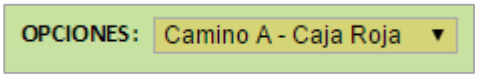


Figura 8. Opciones de simulación.

14. Cargue los datos ingresados con el botón **"Cargar Datos"** e inicie la simulación haciendo click en el botón **"Simulación"** (ver figura 9).

Figura 9. Botón de cargar datos y de simulación.

A medida que transcurre la simulación el software irá mostrando de forma simultánea el trabajo realizado por el obrero en cada punto y en cada tramo del camino escogido. También se mostrará el trabajo neto llevado a cabo por el obrero (ver figura 10).

	W OBRERO	WN OBRERO
A1	0	0
A1-A2	0	0
A2	-39.2	-39.2
A2-A3	126.33	87.13
A3	39.2	126.33

Figura 10. Valores instantáneos de trabajo del obrero.

Nota: Solo cuando se escoge alguna de las opciones en las que el obrero se mueve por el camino 1, el software calculará el valor de la fuerza necesaria que se debe usar para subir la caja por el plano inclinado ascendente y lo mostrará en la parte superior del cuadro de valores instantáneos de trabajo del obrero. El ángulo de aplicación de dicha fuerza sobre la caja siempre se considerará igual a 40°.

15. Cuando se haya completado la simulación haga click en el botón **"Resultados",** el software desplegará algunos resultados numéricos relacionados con el trabajo realizado por las fuerzas presentes en el sistema,

además de otros datos importantes (ver figura 11). **Consigne estos resultados en la tabla 1.**

LONGITUD DEL TRAMO A2-A3 (m)	VELOCIDAD FINAL TRAMO A2-A3 (m/s)	ACELERACION TRAMO A2-A3 (m/s^2)	TRABAJO NETO DEL OBRERO (J)	TRABAJO NETO DEL PESO (J)	TRABAJO NETO FUERZA DE ROZAMIENTO (J)	TRABAJO NETO (J)
3.81	2.14	0.6	126.33	-114.85	0	11.48

Figura 11. Resultados numéricos dados por el software.

16. Con los mismos datos de entrada del sistema realice los demás experimentos que aparecen en el cuadro de **"opciones"**, de modo que el obrero lleve todas las cajas hasta cada uno de los camiones. Recuerde que cada vez que elija una opción nueva de simulación debe cargar los datos (botón **"cargar datos"**), realizar la simulación (botón **"simulación")** y desplegar los resultados numéricos (botón **"resultados"**). También, cada vez que despliegue el cuadro de resultados numéricos consigne los valores en los campos respectivos de la tabla 1.

RESULTADOS NUMÉRICOS – CAMINO A							
Caja	Longitud tramo $A_2 - A_3$	Velocidad final tramo $A_2 - A_3$	Aceleración final tramo $A_2 - A_3$	Trabajo neto del obrero	Trabajo neto del peso	Trabajo de la fuerza de rozamiento	Trabajo de la fuerza neta
Roja							
Azul							
Negra							
RESULTADOS NUMÉRICOS – CAMINO B							
Caja	Trabajo neto del obrero	Trabajo neto del peso	Trabajo de la fuerza neta				
Roja							
Azul							
Negra							
RESULTADOS NUMÉRICOS – CAMINO C							
Caja	Longitud tramo $C_2 - C_3$	Velocidad final tramo $C_2 - C_3$	Aceleración final tramo $C_2 - C_3$	Trabajo neto del obrero	Trabajo neto del peso	Trabajo de la fuerza de rozamiento	Trabajo de la fuerza neta
Roja							
Azul							

Negra							

Tabla 1. Resultados numéricos dados por el software.

17. PRUEBA DE CONOCIMIENTO:

A. Encuentre la expresión matemática que le permita calcular el trabajo que debe realizar el obrero para llevar la caja roja hasta cada uno de los camiones. Corrobore cada resultado con los valores correspondientes consignados en la tabla 1. Determine también el cambio de energía en cada tramo.

B. Con los mismos valores de entrada del sistema vaya al cuadro de opciones y escoja la opción **"Camino C – Caja Negra"**, realice los pasos pertinentes para llevar a cabo la simulación y para generar los resultados numéricos. Explique, tramo a tramo, cada una de las gráficas que proporciona el simulador (gráfica de trayectoria, gráfica de energía y gráfica de trabajo del obrero).

C. Determine matemáticamente el valor de velocidad de la caja al final de la rampa 2 (plano inclinado descendente).

D. ¿Dónde realiza más trabajo el obrero, llevando las cajas al camión 1, al camión 2 o al camión 3?

E. ¿Dónde se hace más trabajo, llevando la caja 1, la caja 2 o la caja 3 al camión 3?

F. ¿Cuál es el trabajo que realiza la rampa 1 (plano inclinado ascendente) cuando se lleva alguna de las cajas al camión 1?

G. De acuerdo a la tabla 1, ¿En cuál de los casos la caja pierde mayor energía?

PRÁCTICA DEMOSTRATIVA N° 6
(COLISIONES)

1. INTRODUCCIÓN

La presente práctica se basa en una serie de conceptos de gran importancia que ayudan a entender y analizar cierto tipo de eventos en los que se presentan las colisiones elásticas entre dos cuerpos con determinadas características. Con base en la definición de momento lineal se podrá llevar a cabo una descripción del movimiento de un objeto desde una perspectiva nueva, distinta a la realizada por medio de la dinámica y la cinemática, en la que se integran los conceptos de masa y velocidad dentro de una misma definición, lo que conduce a la aplicación de una segunda ley de conservación dentro de un sistema aislado, denominada conservación de la cantidad del momento.

2. OBJETIVO

Comprobar de manera experimental los conceptos de impulso y conservación del momento lineal dentro de un sistema animado que represente la colisión de dos objetos de igual masa a través de una herramienta de simulación.

3. MOMENTO LINEAL

El momento lineal de un cuerpo de masa m que se mueve con una velocidad v se define como:

$$\vec{p} = m\vec{v} \qquad \textbf{(1)}$$

De acuerdo a la ecuación (1) el momento P es una cantidad vectorial cuya dirección está determinada por el vector de velocidad del movimiento de la partícula y su magnitud equivale al producto de la masa del cuerpo por el valor de dicha velocidad. La unidad de medida del momento en el SI es el Kg m/s.

Cuando un cuerpo se mueve en el espacio con dirección arbitraria el momento P se expresa en función de sus componentes en x, y y z:

$$P_x = mv_x \qquad P_y = mv_y \qquad P_z = mv_z \qquad \textbf{(2)}$$

Los conceptos de momento lineal y de fuerza resultante aplicada sobre una partícula se pueden relacionar por medio de la segunda ley de Newton de la siguiente forma:

$$\vec{F} = m\vec{a} = m\frac{d\vec{v}}{dt} = \frac{d\left(m\vec{v}\right)}{dt} = \frac{d\vec{P}}{dt} \quad \textbf{(3)}$$

La ecuación (3) establece que la fuerza resultante que actúa sobre una partícula equivale a la tasa de cambio en el tiempo del momento lineal.

4. IMPULSO

Cuando el momento lineal de una partícula cambia es porque su velocidad varía, y si la masa de este cuerpo es constante existe una aceleración que es producida por una fuerza resultante. De este modo, mientras mayor sea la fuerza aplicada, mayor será el cambio de velocidad y por ende el cambio del momento lineal será mayor. Sin embargo si se considera el tiempo de aplicación para un mismo valor de fuerza se puede demostrar que el momento lineal es mayor siempre que dicha fuerza se ejerza durante un intervalo de tiempo más largo. De la ecuación (3), el momento lineal se define como:

$$d\vec{P} = \vec{F}dt \quad \textbf{(4)}$$

Al integrar la ecuación (4) se obtiene la variación del momento de la partícula, desde un valor $\vec{P}_i$ hasta uno $\vec{P}_f$ en un intervalo de tiempo que va desde t_i hasta t_f respectivamente, como lo demuestra la ecuación (7):

$$\vec{P}_f - \vec{P}_i = \Delta\vec{P} = \int_{t_i}^{t_f} \vec{F}dt \quad \textbf{(5)}$$

El término de la derecha de la ecuación (5) se conoce como el impulso I de la fuerza $\vec{F}$ en un intervalo de tiempo definido dt.

$$I = \int_{t_i}^{t_f} F\,dt = \Delta P \quad \textbf{(6)}$$

La ecuación (6) se conoce como el "**Teorema del impulso y del momento"**, el cual establece que el impulso de la fuerza neta es igual al cambio del momento lineal de la partícula. De esta forma, mientras mayor sea el impulso mayor será el cambio del momento de la partícula.

En la práctica es difícil estimar la variación de la fuerza a través del tiempo por lo que en muchos casos se requiere definir una fuerza promedio constante que genere el mismo impulso a la partícula que el dado por $\vec{F}$ cuando actúa durante el intervalo de tiempo Δt. Por lo tanto, considerando la fuerza promedio para el mismo intervalo de tiempo Δt, el impulso queda definido como:

$$I = F_m \Delta t = \Delta P \qquad \textbf{(7)}$$

La ecuación (7) se conoce como aproximación del impulso. Esta aproximación en particular es de mucha utilidad para estudiar algunos eventos como las colisiones, en donde las fuerzas que actúan, denominadas fuerzas impulsivas o fuerzas de impacto, tienen mayor magnitud que cualquier otra fuerza presente y son de muy corta duración.

5. CONSERVACIÓN DEL MOMENTO LINEAL

De acuerdo con la segunda ley de Newton, un objeto se acelera cuando sobre él se aplica una fuerza. De manera análoga, si sobre un cuerpo se aplica un impulso habrá un cambio en el valor de su momento. En ambos casos se requiere de un agente externo (fuerza e impulso) para producir los cambios respectivos, por lo tanto si la fuerza neta aplicada es cero el impulso neto es cero y no hay variación en el momento lineal total. Partiendo de estas afirmaciones se puede decir que si sobre un sistema no se ejerce fuerza neta el momento total del sistema no cambia.

Para ilustrar lo anterior considere un sistema mecánico aislado formado por dos partículas que interactúan ejerciendo fuerzas entre sí, como se muestra en la figura 1. Los momentos de las partículas en cierto instante t son P_1 y P_2 respectivamente, F_{12} es la fuerza ejercida por la partícula 2 sobre la 1 y F_{21} es la fuerza ejercida por la partícula 1 sobre la 2.

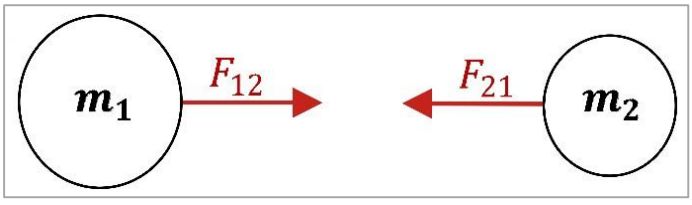


Figura 1. Conservación del momento lineal.

Aplicando la segunda ley de Newton a cada partícula se tiene:

$$\vec{F}_{12} = \frac{d\vec{P}_1}{dt} \qquad \vec{F}_{21} = \frac{d\vec{P}_2}{dt} \tag{8}$$

Aplicando la tercera ley de Newton se tiene:

$$\vec{F}_{12} + \vec{F}_{21} = \frac{d\vec{P}_1}{dt} + \frac{d\vec{P}_2}{dt} = \frac{d\left(\vec{P}_1 + \vec{P}_2\right)}{dt} = 0 \tag{9}$$

$$P_1 + P_2 = constante \tag{10}$$

De la ecuación (10) se concluye que el momento lineal total es constante. Este importante resultado se conoce como ley de conservación del momento lineal. Ahora bien, si se toman los momentos iniciales (P_{1i} y P_{2i}) y finales (P_{1f} y P_{2f}) de las partículas 1 y 2 respectivamente la conservación del momento viene dada de la siguiente manera:

$$P_{1i} + P_{2i} = P_{1f} + P_{2f} \tag{11}$$

$$m_1 v_{1i} + m_2 v_{2i} = m_1 v_{1f} + m_2 v_{2f} \tag{12}$$

6. COLISIONES

Una colisión entre dos partículas es una interacción que ocurre en un espacio limitado durante un pequeño intervalo de tiempo. Durante este evento las partículas en cuestión producen fuerzas impulsivas entre sí, las cuales son mucho mayores que cualquier fuerza externa presente. La ley de conservación del momento lineal es aplicable dentro del fenómeno de las colisiones. Los choques se clasifican básicamente en tres tipos:

- **Choque elástico:** Cuando dos o más cuerpos colisionan sin deformarse y sin producir calor. Hay conservación del momento lineal y de la energía cinética del sistema.

- **Choque inelástico:** Cuando los objetos que chocan se deforman y producen calor durante la colisión. Hay conservación del momento lineal pero no de la energía cinética del sistema.
- **Choque perfectamente inelástico:** Cuando dos objetos colisionan, se deforman, producen calor y permanecen unidos después del choque. Hay conservación del momento lineal y las velocidades finales de las partículas son las mismas.

7. CHOQUES EN DOS DIMENSIONES:

Una colisión en dos dimensiones ocurre cuando una partícula de masa m_1 que se mueve con velocidad inicial choca de costado con otra partícula de masa m_2 que se encuentra inicialmente en reposo (o desplazándose con una determinada velocidad inicial) generando un movimiento final de tipo bidimensional, como se muestra en la figura 2. Después de la colisión las dos partículas se mueven en diferentes direcciones respecto de la línea del movimiento de la partícula 1 (generalmente se toma la dirección del movimiento como el eje x).

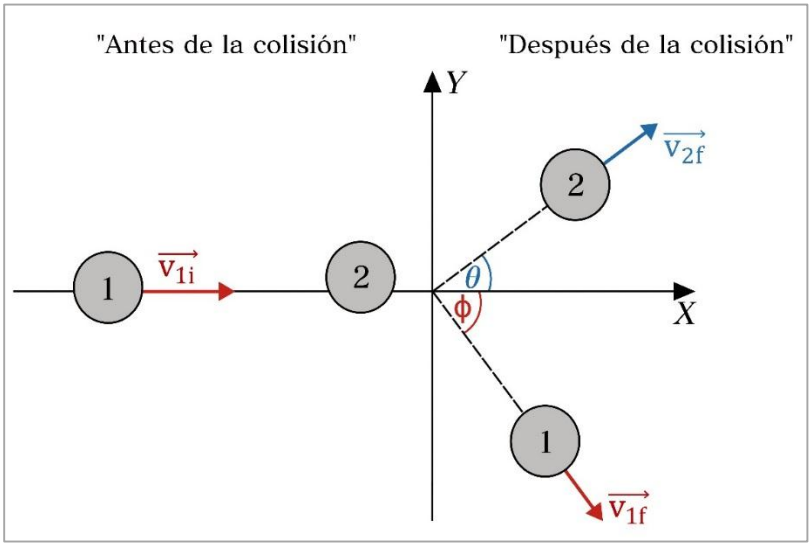


Figura 2. Choque bidimensional.

De acuerdo a la figura 2 y a la ley de conservación del momento lineal se obtienen las siguientes ecuaciones para las componentes x y y del momento:

$$\textit{Eje x:} \qquad P_{ix} = P_{fx} \qquad m_1 v_{1ix} + m_2 v_{2ix} = m_1 v_{1fx} + m_2 v_{2fx} \qquad \textbf{(13)}$$

Eje y: $P_{iy} = P_{fy}$ $m_1 v_{1iy} + m_2 v_{2iy} = m_1 v_{1fy} + m_2 v_{2fy}$ **(14)**

Partiendo de que la velocidad inicial de la partícula 2 es cero y que las velocidades finales v_{1f} y v_{2f} se pueden representar por medio de sus componentes rectangulares, las ecuaciones (13) y (14) quedan expresadas como:

Eje x: $P_{ix} = P_{fx}$ $m_1 v_{1ix} = m_1 v_{1f} \cos\theta + m_2 v_{2f} \cos\phi$ **(15)**

Eje y: $P_{iy} = P_{fy}$ $0 = m_1 v_{1f} \sin\theta - m_2 v_{2f} \sin\phi$ **(16)**

Como este tipo de choque es elástico, por conservación de energía se tiene:

$$\frac{1}{2} m_1 v_{1i}^{\ 2} + \frac{1}{2} m_2 v_{2i}^{\ 2} = \frac{1}{2} m_1 v_{1f}^{\ 2} + \frac{1}{2} m_2 v_{2f}^{\ 2} \quad \textbf{(17)}$$

8. DESCRIPCIÓN DEL SISTEMA

La práctica de colisiones está basada en un juego de billar pool compuesto por los siguientes elementos: dos bolas de billar (bola 1 y bola 2), un taco y dos buchacas (buchaca 1 y buchaca 2) habilitadas para el ingreso de las bolas, las cuales se encuentran ubicadas en dos esquinas de la mesa de billar como se muestra en la figura 3. Dentro de la mesa se ha dispuesto un sistema de ejes coordenados que permiten referenciar la ubicación de dichos elementos a través de puntos en el plano xy. De esta manera, la bola 1 (bola de impacto) se sitúa en un punto fijo P_1 que demarca la posición de su centro mientras que las buchacas 1 y 2 se ubican en los puntos P_{b1} y P_{b2} respectivamente. Por su parte la bola 2 (bola objetivo) podrá posicionarse en cualquier punto P_2 dentro de un área específica de la mesa denominada “área delimitada para la ubicación de la bola 2”.

Con base en las coordenadas suministradas (P_1, P_2, P_{b1} y P_{b2}) dentro del plano cartesiano y el valor del radio de las bolas de billar (R_1 = R_2 = 3.52 cm) el usuario deberá realizar los cálculos matemáticos correspondientes que le permitan encontrar el valor del ángulo β (dirección con que el taco de billar debe apuntar para colisionar la bola 1 con la bola 2) y la fuerza del taco $\mathbf{F_t}$ necesaria para ingresar la bola 2 de forma directa en alguna de las buchacas. Los resultados obtenidos se deberán corroborar a través de la herramienta de simulación.

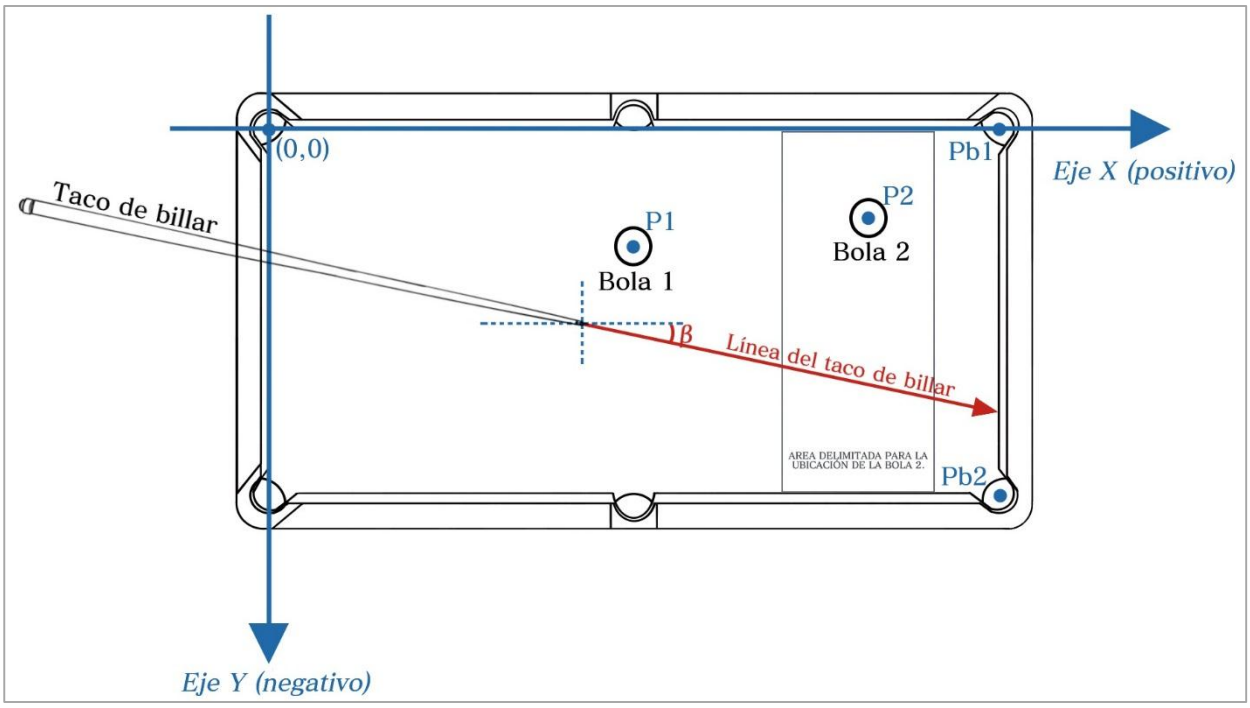


Figura 3. Geometría de colisión de las bolas de billar.

9. GEOMETRÍA PARA LA COLISIÓN DE LAS BOLAS DE BILLAR

Los cálculos matemáticos para determinar la dirección con la que el taco de billar debe apuntar para golpear la bola 1 (ángulo β) se desarrollan a partir de la colisión entre las dos bolas y se basan en un procedimiento que se ha denominado "geometría de la colisión". Dentro de este proceso se determinan secuencialmente las coordenadas del punto de contacto de las dos bolas, las coordenadas del centro de la bola 1 al momento de la colisión y finalmente el ángulo β requerido (se debe considerar que la línea del taco de billar siempre pasa por el centro de la bola 1).

De acuerdo a la figura 4 los casos posibles de colisión entre las bolas son los siguientes:

- **Caso 1 (figura 4-a):** Se da cuando la línea del taco de billar prolongada sobre el centro de la bola 1 en el sitio de colisión pasa por encima del centro de la bola 2.

- **Caso 2 (figura 4-b):** se presenta cuando la línea del taco de billar prolongada sobre el centro de la bola 1 en el sitio de colisión pasa por debajo del centro de la bola 2.

- **Caso 3 (figura 4-c):** se da cuando la línea del taco de billar prolongada sobre el centro de la bola 1 en el sitio de colisión pasa exactamente por el centro de la bola 2.

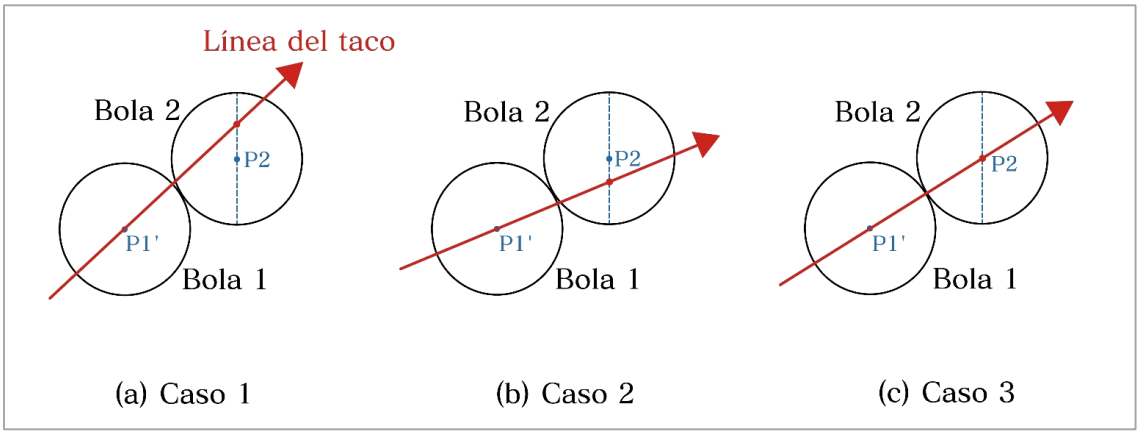


Figura 4. Casos posibles de colisión entre las bolas de billar.

A continuación se muestra el procedimiento para encontrar el ángulo β dentro de cada uno de los tres posibles casos de colisión.

➢ GEOMETRÍA DE LA COLISIÓN – CASO 1:

La figura 5 muestra en detalle la colisión de las bolas 1 y 2 para el primer caso. La construcción geométrica se ha realizado considerando que para una dirección de inclinación del taco de billar (ángulo β) sobre la bola 1 se consigue ingresar la bola 2 dentro de la buchaca 1, representada por el punto P_{b1}. Es importante destacar que el punto de contacto de las bolas (P_c) siempre estará contenido dentro del segmento que une los centros de estas ($\overline{P_1' P_2}$), y es la dirección de este segmento respecto a la línea del taco de billar (ángulo θ) la que determina hacia donde se va a dirigir la bola 2 después del choque.

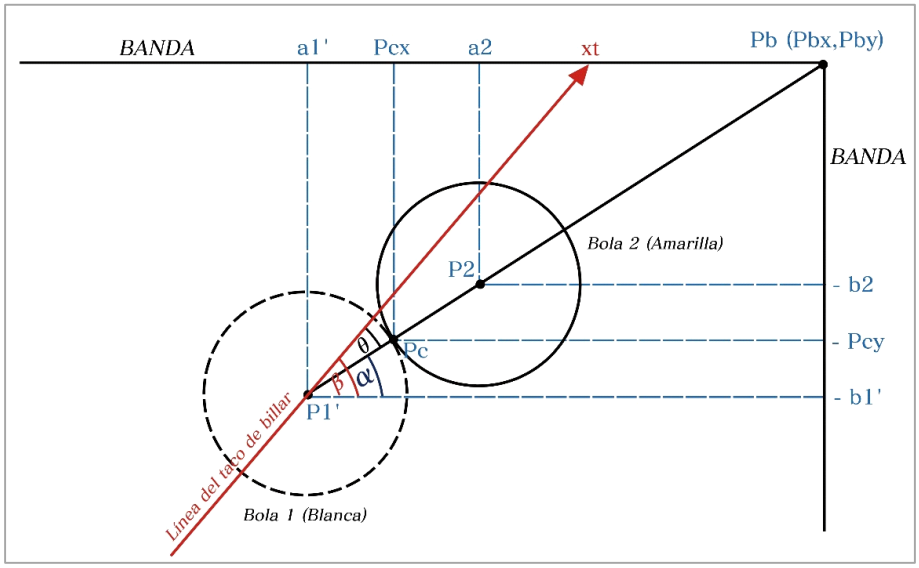


Figura 5. Geometría de la colisión – Caso 1.

La descripción completa de los puntos y los ángulos involucrados dentro de esta colisión se realiza en la tabla 1:

Punto/Angulo	Descripción	Coordenadas
P_1	Centro de la bola 1 (bola blanca)	$(a_1, -b_1)$
P_1'	Centro de la bola 1 (bola blanca) durante la colisión	$(a_1', -b_1')$
P_2	Centro de la bola 2 (bola amarilla)	$(a_2, -b_2)$
P_c	Punto de contacto de las bolas 1 y 2	$(P_{cx}, -P_{cy})$
P_{b1}	Vértice de la buchaca 1	$(P_{b1x}, 0)$
β	Dirección del taco de billar respecto al eje x	-
α	Dirección del segmento $\overline{P_1' P_2}$ respecto al eje x	-
θ	Dirección que toma la bola 2 respecto a la línea del taco después de la colisión	-

Tabla 1. Ángulos y puntos considerados dentro de la colisión de las bolas.

Una vez definidos los elementos de la tabla 1 dentro de la colisión de las bolas de billar se realiza el procedimiento matemático para determinar los valores de los ángulos
β y θ. Los pasos son los siguientes:

Paso 1: Se determina la distancia desde el centro de la bola 2 hasta el punto que representa a la buchaca 1:

$$\overline{P_2P_b} = X_1 = \sqrt{\left(P_{bx} - a_2\right)^2 + \left(b_2\right)^2} \quad \textbf{(18)}$$

Paso 2: Con base en el valor del segmento X_1 y las coordenadas de los puntos P_2 y P_{b1} se halla el valor del ángulo α (ver figura 6):

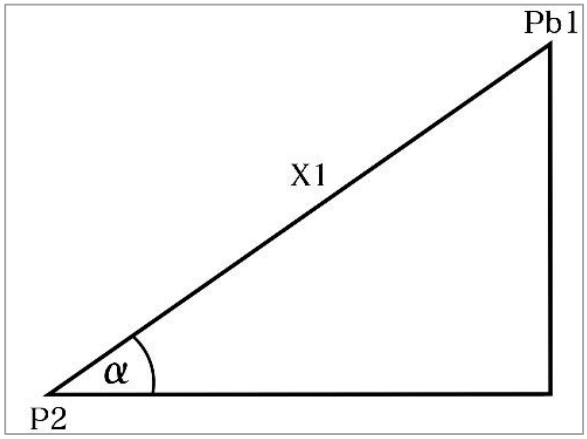


Figura 6. Dirección del segmento $\overline{P_1{}'P_2}$ respecto al eje x

$$\alpha = \sin^{-1}\left[\frac{\left(0-\left(-b_2\right)\right)}{X_1}\right] = \sin^{-1}\left(\frac{b_2}{X_1}\right) \quad \textbf{(19)}$$

Paso 3: Se determinan las coordenadas del punto de contacto (P_c) de las bolas 1 y 2 de acuerdo con la figura 7(a). Para ello se calcula la longitud de los segmentos L_x y L_y, después, estos valores se restan a las coordenadas en x y y del punto P_2 respectivamente.

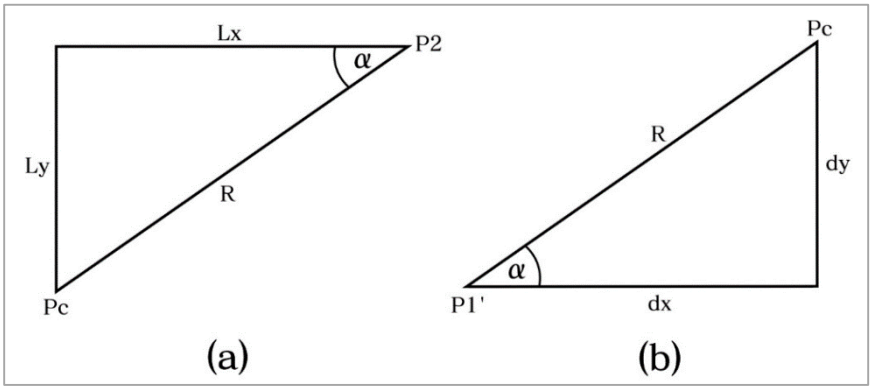


Figura 7. Geometría de colisión de las bolas de billar.

Los segmentos L_x y $L_y$vienen dados como:

$$L_x = R\cos\alpha \quad \textbf{(20)}$$

$$L_y = R\sin\alpha \quad \textbf{(21)}$$

Por lo tanto, las coordenadas del punto de contacto son:

$$P_{cx} = a_2 - L_x \quad \textbf{(22)}$$

$$P_{cy} = -b_2 - L_y \quad \textbf{(23)}$$

Paso 4: Conociendo la posición del punto de contacto (P_c) se calculan las coordenadas del centro de la bola 1 (P_1') en el sitio de colisión. De acuerdo a la figura 7(b) se realiza un procedimiento similar al paso anterior encontrando esta vez las magnitudes de los segmentos d_x y d_y, para después restar estos valores a las coordenadas en x y y del punto de contacto (P_c) respectivamente. Los segmentos d_x y d_y vienen dados como:

$$d_x = R\cos\alpha \quad \textbf{(24)}$$

$$d_y = R\,sen\alpha \quad \textbf{(25)}$$

Por lo tanto, las coordenadas del centro de la bola 1 en el sitio de colisión son:

$$a_1' = P_{cx} - d_x \quad \textbf{(26)}$$

$$b_1' = P_{cy} - d_y \quad \textbf{(27)}$$

Paso 5: Con las posiciones de los puntos P_1 y P_1'se determina la dirección en la que debe apuntar el taco de billar (ángulo β) al momento de golpear la bola 1 (ver figura 7):

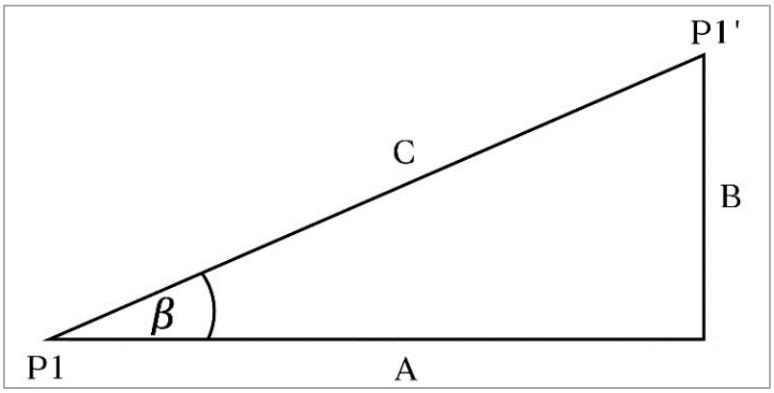


Figura 7. Geometría de colisión de las bolas de billar.

La magnitud de los segmentos A, B y C se calculan de la siguiente manera:

$$A = a_1' - a_1 \quad \textbf{(28)}$$

$$B = b_1' - b_1 \quad \textbf{(29)}$$

$$C = \sqrt{A^2 + B^2} \quad \textbf{(30)}$$

Por lo tanto el ángulo β viene dado como:

$$\beta = \tan^{-1}\left(\frac{B}{A}\right) \quad \textbf{(31)}$$

Paso 6: Se determina el valor del ángulo θ, tomando como referencia los triángulos que se forman dentro de la colisión y que son mostrados en la figura 9.

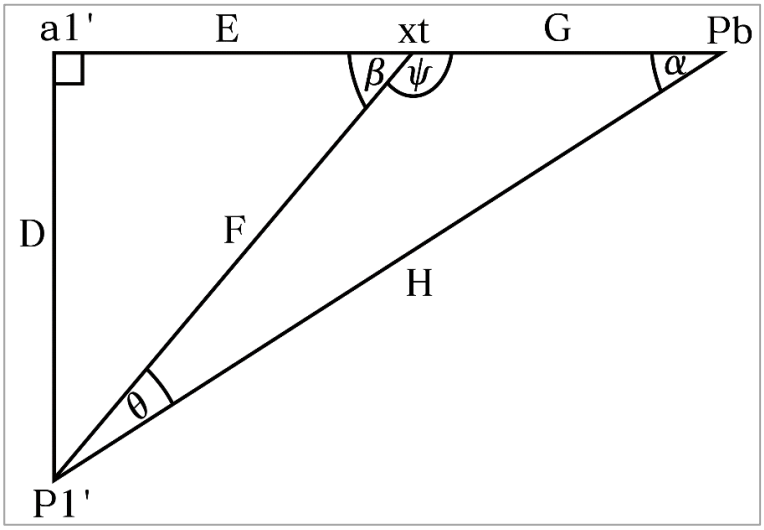


Figura 9. Geometría de colisión de las bolas de billar.

Dentro de este procedimiento se deben calcular, de manera secuencial, los valores de los segmentos D, F, E, G y H respectivamente:

$$D = 0 - \left(-b_1'\right) = b_1' \tag{32}$$

$$F = \frac{D}{\sin\beta} \tag{33}$$

$$E = F\cos\beta \tag{34}$$

$$G = \left(P_{b1x} - \mathrm{a}_1'\right) - E \tag{35}$$

$$H = X_1 + 2R \tag{36}$$

Paso 7: Se calcula el valor del ángulo ψ:

$$\psi = 180 - \beta \tag{37}$$

Paso 8: Con base en los triángulos de la figura 9 se aplica la ley trigonométrica de los senos para determinar el valor del ángulo θ:

$$\frac{\sin\alpha}{F}=\frac{\sin\psi}{H}=\frac{\sin\theta}{G} \tag{38}$$

$$\theta=\sin^{-1}\left(G\frac{\sin\psi}{H}\right) \tag{39}$$

➢ GEOMETRÍA DE LA COLISIÓN – CASO 2:

La figura 8 muestra en detalle el segundo caso de posible colisión entre las bolas 1 y 2. Los ángulos α y β se determinan siguiendo el mismo procedimiento visto en el caso 1. Una vez calculados estos valores se halla el correspondiente valor del ángulo θ (dirección de la bola 2 después de la colisión) de la siguiente forma:

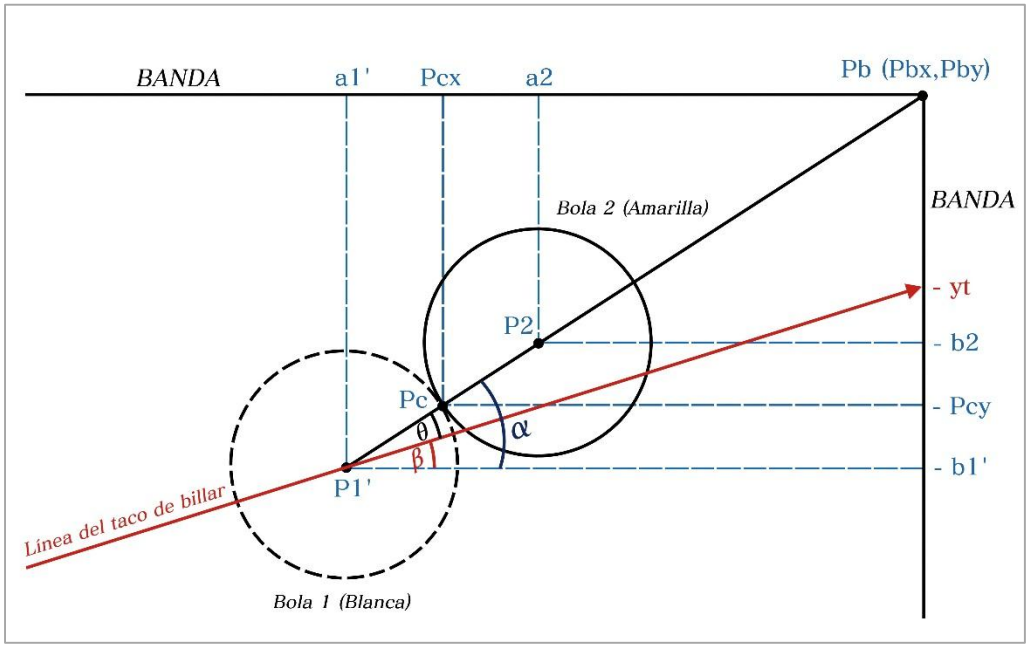


Figura 8. Geometría de la colisión – Caso 2.

$$\alpha=\theta+\beta \tag{40}$$

$$\theta=\alpha-\beta \tag{41}$$

➢ GEOMETRÍA DE LA COLISIÓN – CASO 3:

La figura 10 muestra en detalle la colisión de las bolas 1 y 2 para el tercer caso. Como se puede observar, al estar alineados los centros de las dos bolas con el punto que representa a la buchaca 1, los ángulos α y β son iguales, de esta forma el ángulo θ equivale a 0° y por tanto la bola 2 se mueve en la misma dirección del taco de billar después de la colisión. El procedimiento para determinar los ángulos α y β es el mismo que en los dos casos anteriores.

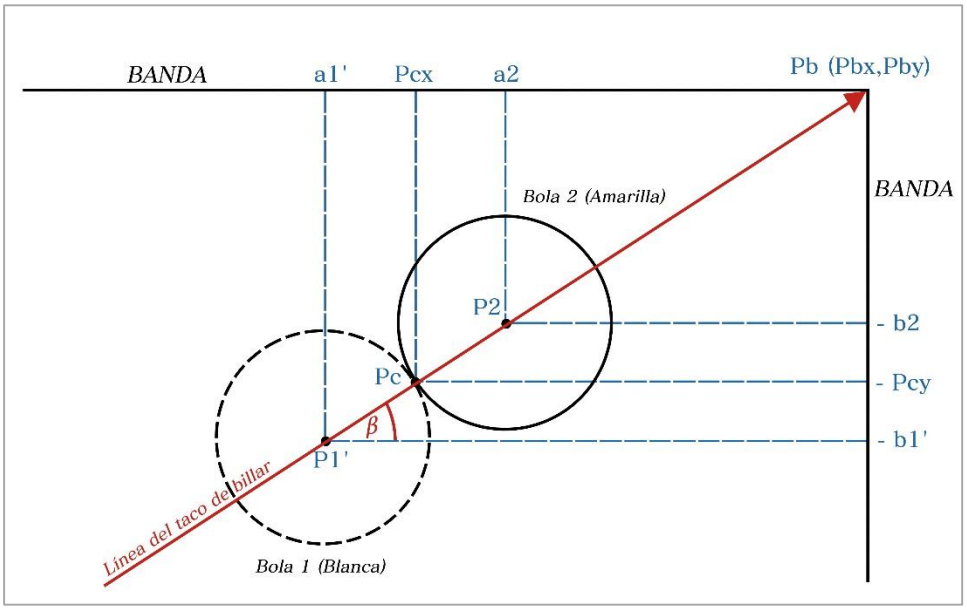


Figura 10. Geometría de la colisión – Caso 3.

Nota: Los tres casos de colisión presentados son generales para cualquier posición de la bola 2, por lo tanto si se escoge la buchaca 2 como objetivo, se realiza un procedimiento similar para encontrar los valores de α, β y θ.

10. DESARROLLO DE LA PRÁCTICA

Para desarrollar la práctica de movimiento en dos dimensiones el estudiante debe tener acceso a un computador con conexión a internet, el cual debe contar con el explorador “Google Chrome” dentro de sus herramientas de navegación.

Nota: Antes de correr la aplicación tenga en cuenta las siguientes recomendaciones:

- La resolución del equipo donde se va a realizar la simulación debe ser ajustada a un valor de 1366 x 768.

- El tamaño de zoom del navegador “Google Chrome” debe estar en un valor de 100%.

- La página web donde se aloja la aplicación debe estar totalmente maximizada durante todo el desarrollo de la práctica.

- De ser necesario actualice la página web por si se presenta algún problema.

PROCEDIMIENTO:

11. Ingrese a la siguiente página web: http://www.unicauca.edu.co/experimentos_mecanica/aplicacion.html para tener acceso a la aplicación web que contiene las prácticas virtuales relacionadas con la temática del curso, como se muestra en la figura 11.

Figura 11. Interfaz principal de la aplicación web.

12. Para ingresar a la práctica de colisiones haga click en el respectivo enlace, dentro de la “**Lista de prácticas**”, como se muestra en la figura 12. El sistema desplegará la interfaz mostrada en la figura 13.

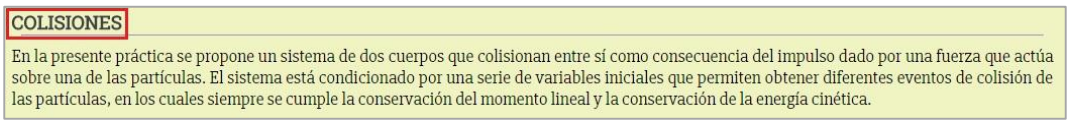


Figura 12. Enlace para la práctica de colisiones.

Figura 13. Interfaz principal – Práctica de Colisiones.

13. Dentro de la aplicación establezca la ubicación de la bola 2, para ello escriba sobre los campos correspondientes x y y del cuadro de **"Posición de la Bola Amarilla"** las coordenadas 178.77 cm y -17.07 cm respectivamente de modo que pueda ser embocada en la buchaca 1. Cuando haya ingresado estos valores dé click en el botón **"Reiniciar"**, de esta manera la bola objetivo queda situada en el punto indicado cuando se carga nuevamente la aplicación web (ver figura 14).

Figura 14. Datos de entrada del simulador.

Con los valores de las coordenadas del punto de ubicación de la bola 2 (bola amarilla) realice los cálculos pertinentes para determinar la dirección (ángulo β) en que debe apuntar el taco de billar para golpear la bola 1 (bola blanca) de modo que pueda ingresar la bola objetivo dentro de la buchaca escogida.

Nota: Para la realización de dichos cálculos utilice también los datos mostrados en la figura 15, los cuales son proporcionados por la herramienta de simulación. Estos datos dados por el sistema son constantes para cualquier ubicación de la bola 2 que se haya escogido.

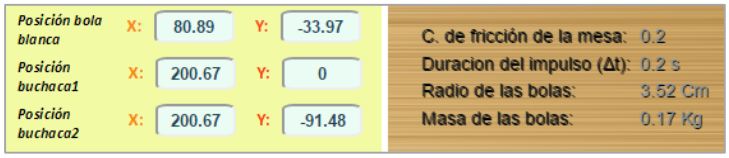


Figura 15. Datos dados por el sistema.

14. Cuando haya determinado matemáticamente el ángulo requerido utilice el puntero del mouse sobre la bola 2 (bola amarilla) para buscar el valor de β dentro de la aplicación. A medida que se mueve el puntero sobre la bola objetivo se irá mostrando dentro del cuadro denominado "**Ángulo de tiro**" una serie de valores en grados que abarcan un rango desde -45° hasta 45°respecto del eje x de la mesa de billar. Cuando haya dado con el valor exacto o haya encontrado un ángulo lo más cercano posible al hallado matemáticamente dé un click para fijar dicho valor como la magnitud de β con la cual se va a realizar la simulación. Una vez que haya fijado el ángulo requerido, haga click en el botón de **"simulación**", como se muestra en la figura 16.

Figura 16. Botón de simulación y reinicio de simulación.

A medida que se lleva a cabo la simulación del juego de billar, el sistema mostrará los resultados numéricos más relevantes relacionados con el movimiento de las bolas 1 y 2 antes y después de la colisión, como se muestra en la figura 17. Si el valor de β encontrado es el correcto, la bola 2 será depositada dentro de la buchaca 1.

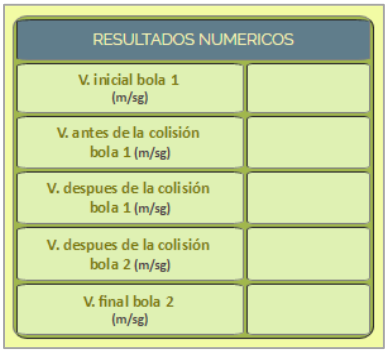


Figura 17. Resultados numéricos desplegados por el software.

Anexo C

PROCEDIMIENTO PARA ALOJAR LA APLICACIÓN WEB EN UN SERVIDOR GRATUITO

En el presente anexo se muestra el procedimiento general llevado a cabo para alojar la aplicación web desarrollada dentro del servidor gratuito **"000webhost"**, por medio de la configuración de una cuenta FTP, haciendo uso de la herramienta **"Filezilla"** como software cliente.

1. ELEMENTOS REQUERIDOS

Las dos herramientas escogidas para acceder a los servicios de hosting gratuito para la aplicación web y un cliente FTP a través del cual se puedan transferir los archivos desde un ordenador hasta dicho servidor se muestran a continuación:

1.1. SERVIDOR 000WEBHOST

000webhost es una de las empresas más grandes de la red que brinda actualmente el servicio de hosting para diferentes tipos de sitios web. En su versión gratuita, el servidor 000webhost ofrece una serie de prestaciones, entre las que se encuentran: Espacio en disco para el alojamiento de sitios web de 350 MB y 100 Gb de tráfico de datos, lo que garantiza un gran flujo de usuarios conectados simultáneamente, permite el uso de PHP y brinda soporte para la creación de sitios por medio de un sencillo editor, permite 2 bases de datos mySQL, ofrece 5 cuentas de correo, 5 dominios y 5 subdominios para cada uno de sus usuarios, entre otros.

1.2. FILEZILLA

El FTP (File Transfer Protocol) es un protocolo de red utilizado para la transferencia de archivos, el cual requiere de un cliente (el encargado de subir los archivos de las páginas web desarrolladas) y un servidor (elemento que recibe los

archivos de la página web y los muestra a través de la misma plataforma). El uso de un software cliente FTP es recomendable ya que proporciona más control sobre los distintos procedimientos que se requieran llevar a cabo, esencialmente subida y descarga de archivos hacia y desde un servidor web.

Los navegadores al contrario pueden brindar los medios para usar FTP pero no son los más adecuados para trabajar como clientes (sus mecanismos generalmente conllevan a muchos errores, sobre todo en la descarga de archivos en este tipo de servidores). Filezilla es un software gratuito que ofrece soporte para FTP y permite la conexión entre un ordenador y un servidor web. Se encuentra disponible para distintas plataformas como por ejemplo Windows, Mac OSX y Linux. De esta manera, partiendo de las prestaciones dadas por las dos herramientas mencionadas se lleva a cabo el procedimiento general que permite el alojamiento de la aplicación web dentro del servidor web designado, a través del software Filezilla:

2. PROCEDIMIENTO GENERAL

Paso 1: Cree una cuenta de usuario para el ingreso a la plataforma de hosting gratuito **000Webhost** dentro del sitio web oficial https://www.000webhost.com/, como se muestra en la figura 1.

Figura 1. Sitio oficial del servidor web 000webhost.

Paso 2: Despliegue el formulario de inscripción dando click en el botón **"Free Sign Up"**, como se observa en la figura 2:

Figura 2. Despliegue del formulario de inscripción.

Paso 3: Dentro del respectivo formulario proporcione una cuenta de correo o email y una contraseña para acceder a los servicios del servicio de hosting gratuito (ver figura 3):

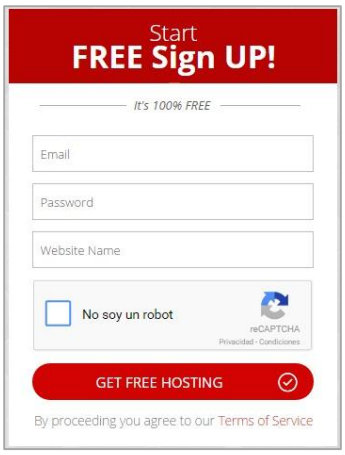


Figura 3. Despliegue del formulario de inscripción.

Una vez que ha sido creada la cuenta el sistema envía un correo de verificación para que el nuevo usuario active su cuenta a través de él. Al momento de ser activada, la cuenta queda disponible para ser utilizada.

Paso 4: Dé click en el botón **"Login"** para ingresar a la plataforma del servicio de hosting (ver figura 4):

Figura 4. Despliegue del formulario de inscripción.

Paso 5: En la ventana desplegada ingrese la cuenta de email y el password proporcionados en el proceso de inscripción realizado en el paso 3 (ver figura 5):

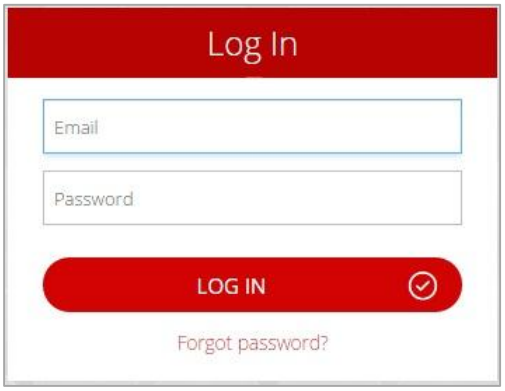


Figura 5. Datos para el ingreso al sistema.

Paso 6: Dentro de la interfaz de la plataforma se encuentra la información del propietario de la cuenta y algunos enlaces para realizar diferentes procedimientos dentro del sitio web (ver lista de cuentas creadas, adquirir la cuenta Premium del servicio, obtener ayuda y salir de la cuenta). La lista de cuentas se muestra en el panel principal, el estado de ellas (activo o inactivo) y el enlace para acceder al panel de configuración de las respectivas cuentas (**Go to CPanel**). También se dispone de un enlace para la creación de una nueva cuenta (**Create New Account**). Para continuar con el procedimiento haga click en dicho enlace:

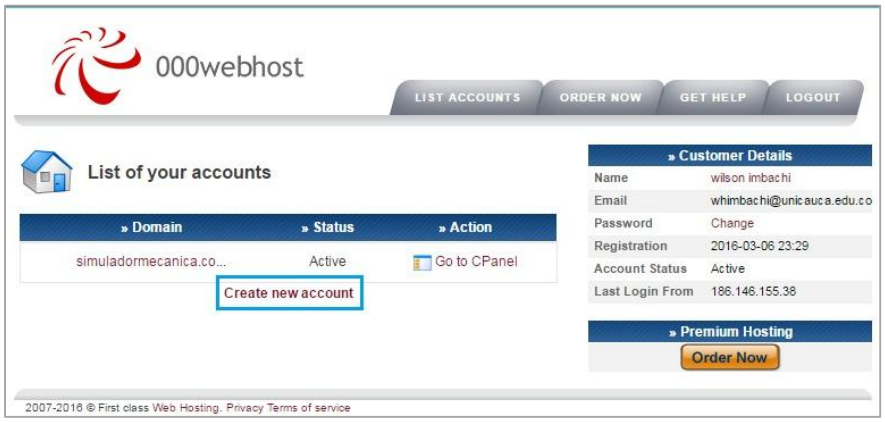


Figura 6. Interfaz principal de la plataforma.

Paso 7: En los campos respectivos de la interfaz se deben suministrar esencialmente los siguientes datos: nombre del dominio con el que va a figurar la página web que será alojada y el password usado para crear la cuenta de usuario en el paso 3. Se aceptan los términos y condiciones del procedimiento y se

procede a la creación del dominio dando click en “**Create my Account**”, como se indica en la figura 7:

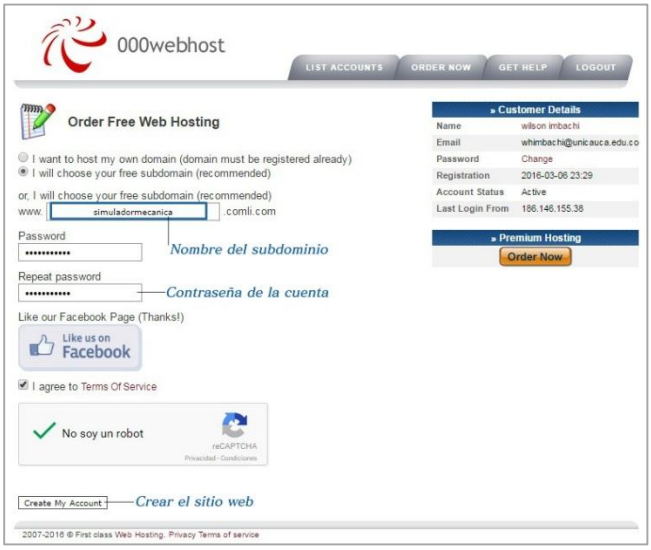


Figura 7. Procedimiento para la creación del dominio de la página web.

Al ser creado el dominio, la plataforma envía un email a la cuenta de correo electrónico suministrada, el cual contiene información relacionada con dicho procedimiento. Del correo se deben rescatar los datos correspondientes a: Dominio de la página web, IP Address (que corresponde a la dirección IP del servidor con el que se va a configurar la cuenta FTP), Username (nombre del usuario que creó esta nueva entrada) y el password (contraseña de acceso al servicio de hosting gratuito), como se resalta en la figura 8.

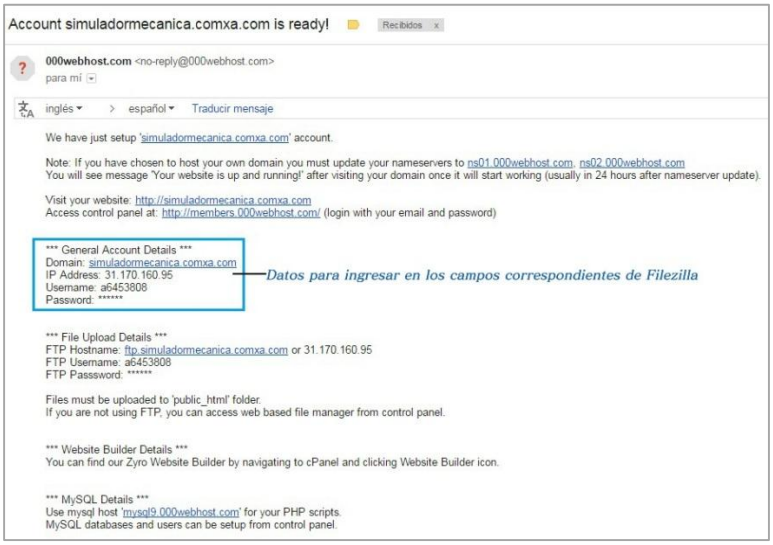


Figura 8. Datos suministrados por la plataforma de hosting gratuito.

Paso 8: Una vez que se ha creado el dominio de la página web se procede a la utilización del software cliente FTP llamado Filezilla. Suponiendo que el usuario no cuenta con esta herramienta, se muestra el procedimiento para obtenerlo de manera gratuita a través de su sitio web oficial https://filezilla-project.org/, como se muestra en la figura 9. Dentro de la página web dé click en el enlace **"Download Filezilla Client"** para acceder al archivo de instalación del producto.

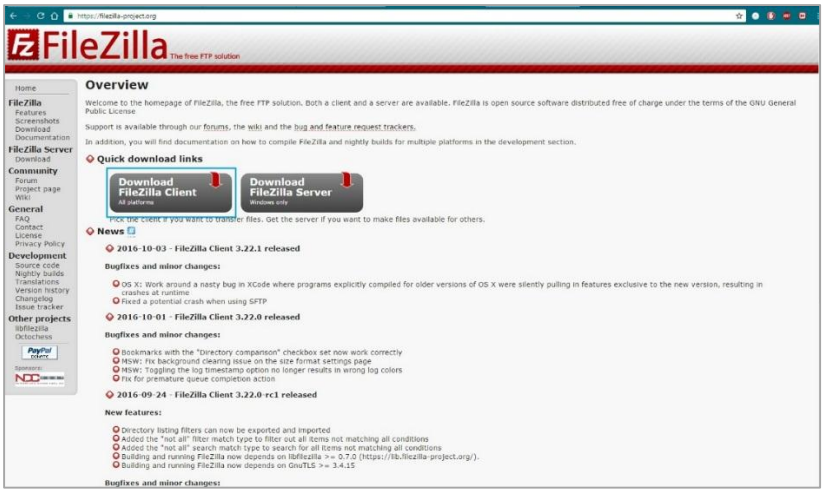


Figura 9. Sitio web oficial del software Filezilla.

Paso 9: Descargue el instalador de Filezilla haciendo click en **"Download Filezilla Client"**, como se muestra en la figura 10. La versión disponible para este caso corresponde a Filezilla v. 3.22.1 para Windows de 32 bits.

Figura 10. Enlace de descarga del instalador de Filezilla.

Paso 10: Después de un tiempo definido el instalador de Filezilla se descarga por completo en el respectivo ordenador. Se obtiene un archivo ejecutable como el mostrado en la figura 11.

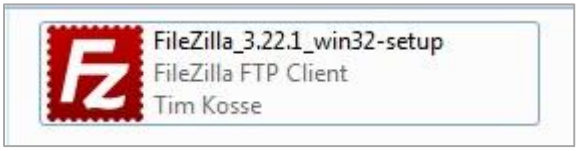


Figura 11. Archivo ejecutable de Filezilla.

Paso 11: Con el click derecho del mouse sobre el instalador de Filezilla escoja la opción **"Ejecutar como administrador"** para iniciar el proceso de instalación del software (ver figura 12).

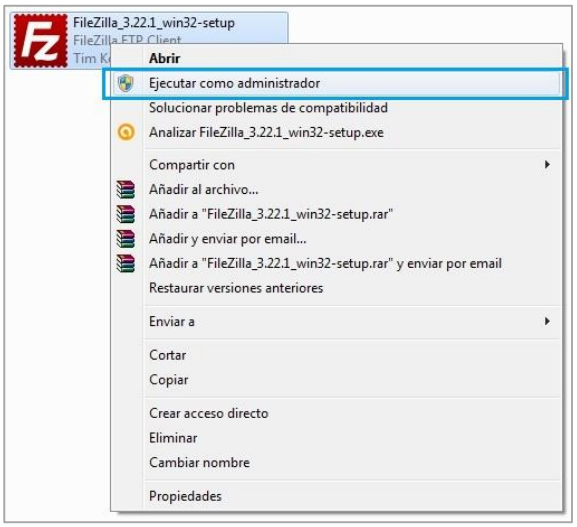


Figura 12. Inicio de instalación de Filezilla.

Paso 12: Acepte el acuerdo de licencia del producto dando click en **"I Agree"** y espere un tiempo determinado hasta que se termine la instalación completa (figuras 13 y 14 respectivamente).

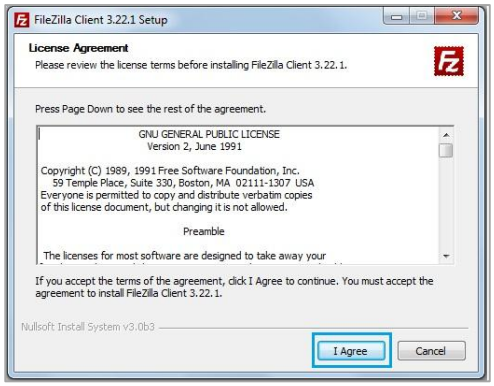


Figura 13. Acuerdo de licencia del producto.

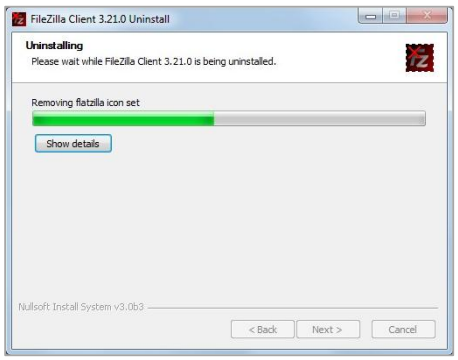


Figura 14. Proceso de instalación.

Paso 13: Finalice el proceso de instalación del software dando click en **"Finish"** y abra el respectivo programa (ver figura 15).

Figura 15. Fin del proceso de instalación.

Paso 14: Dentro de la interfaz principal de Filezilla ingrese en los campos correspondientes los datos suministrados a través de la cuenta de correo, referenciados en el paso 7 (dirección IP del servidor, nombre de usuario y password) y dar **"Enter"** por medio del teclado del ordenador (ver figura 16). Inmediatamente el en campo de registro de mensajes (parte inferior contigua a los campos de ingreso de los datos mencionados) se informa de todos los procedimientos llevados a cabo por Filezilla.

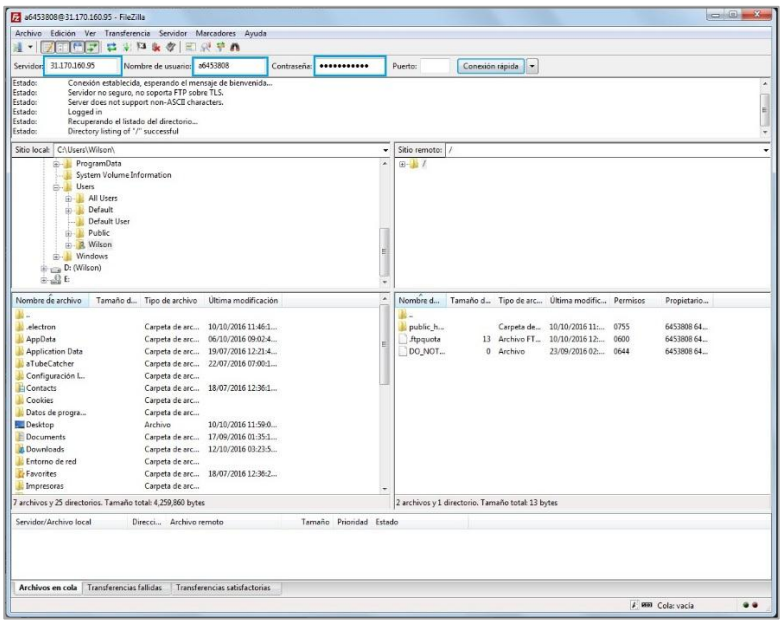


Figura 16. Datos de configuración de la cuenta FTP.

Paso 15: Dentro del campo del directorio remoto haga click en el símbolo "+" para mostrar las carpetas que contiene el servidor web (ver figura 17).

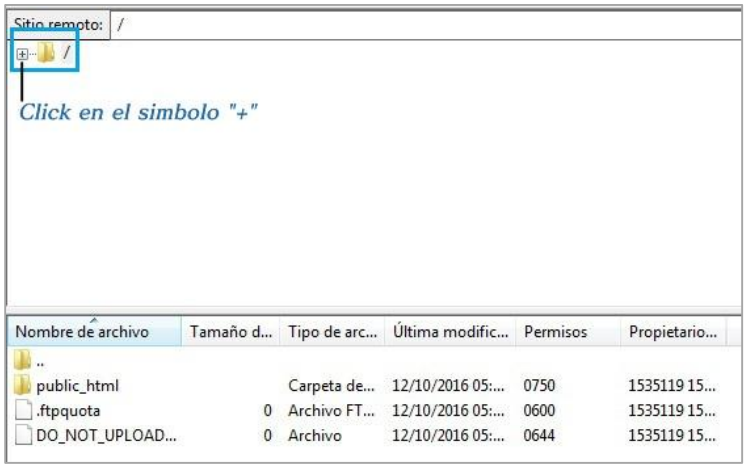


Figura 17. Directorio remoto del servidor web.

Paso 16: Después del click se despliega una carpeta denominada **"public_html"**, la cual va a contener los archivos de la página web que se vayan a subir posteriormente al servidor. Posteriormente haga click sobre dicha carpeta con el fin de mostrar en el campo del contenido del directorio los elementos que están en su interior. Los archivos que se muestran son: **".htacces"** y **"default.php" (**ver figura 18).

Figura 18. Carpeta contenedora del proyecto.

Paso 17: Dé click derecho sobre el archivo **"default.php"** y eliminarlo de la carpeta contenedora (**public_html**) como se muestra en la figura 19 y 20 respectivamente. En este punto la carpeta en cuestión ya está lista para recibir los archivos de la página web que serán subidos al servidor.

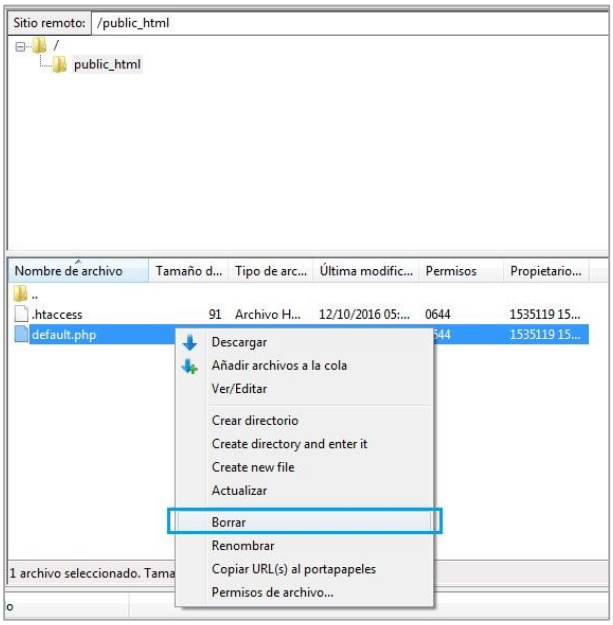


Figura 19. Eliminación del archivo "default.php" – parte 1.

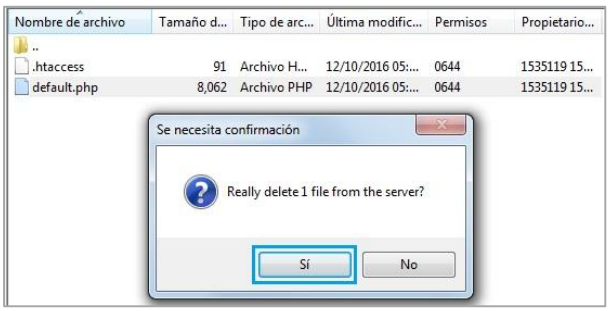


Figura 20. Eliminación del archivo “default.php” – parte 2.

Paso 18: Busque dentro del directorio local la carpeta que contiene los archivos que serán subidos al servidor (ver figura 21).

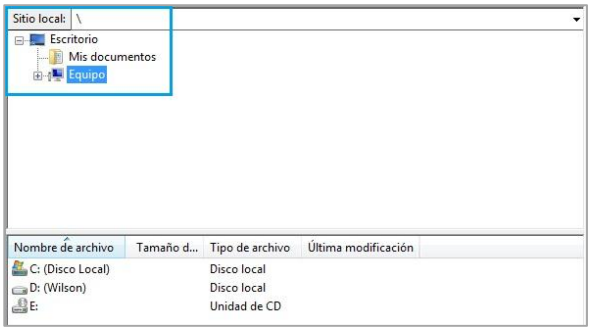


Figura 21. Búsqueda de los archivos.

Paso 19: Al localizar la carpeta, haga click sobre ella de modo que se muestre el contenido del directorio local (archivos) como se muestra en la figura 22.

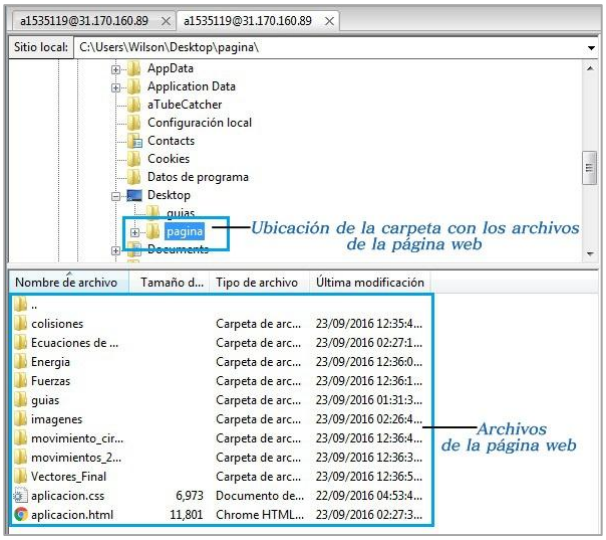


Figura 22. Contenido del directorio local.

Paso 20: Seleccione los archivos contenidos en la carpeta del directorio local, dé click derecho y escoja la opción **"Subir"**, de esta forma a través de Filezilla se iniciará el proceso de subida de la página web al respectivo servidor web **(000webhost)** (ver figura 23).

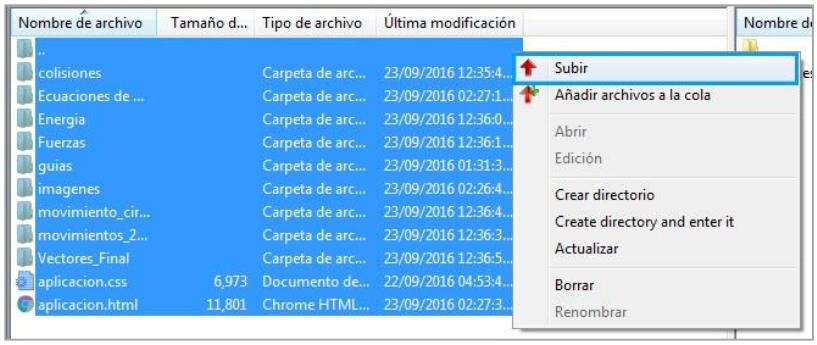


Figura 23. Selección de los archivos que serán subidos al servidor.

En el campo correspondiente a la "cola de transferencia" se irá mostrado el porcentaje del procedimiento de subida de los diferentes archivos realizado por Filezilla (ver figura 24).

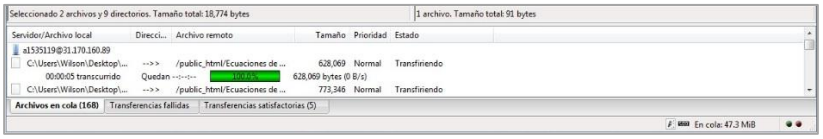


Figura 24. Porcentaje de subida de la página web.

Paso 21: Finalmente cuando en la cola de trasferencia el porcentaje haya alcanzado el 100% cargue la página web subida con el nombre del dominio dado en el correo electrónico del paso 7. Para ello simplemente dé click sobre el enlace correspondiente dado en **"Domain"** o ingrese directamente el nombre del dominio en un explorador web.

Figura 25. Dominio de la página web.

Anexo D

CONFIGURACIÓN DEL CLIENTE VPN

Dentro del presente documento se dan las pautas básicas para la configuración de un cliente VPN, empleado para llevar a cabo el acceso seguro a los servicios institucionales administrados por los funcionarios de la Universidad del Cauca. Este anexo está basado enteramente en el "manual de instalación y configuración de servicios", orientado hacia la creación de un cliente VPN, con fecha de creación del 20 de marzo de 2013 y realizado por los ingenieros Fabián Andrés Mera (creador del documento) y Betty Fernández (actualización del documento – diciembre 4 de 2015). El documento original se encuentra disponible en la siguiente dirección: ftp://ftp.unicauca.edu.co/Documentos_Publicos/.Fortigate/.

1. CONCEPTOS ESENCIALES DEL SERVICIO

VPN: Una red privada virtual, RPV, o VPN de las siglas en inglés de Virtual Private Network, es una tecnología de red que permite una extensión segura de la red local sobre una red pública o no controlada como Internet. Permite que la computadora en la red envíe y reciba datos sobre redes compartidas o públicas como si fuera una red privada con toda la funcionalidad, seguridad y políticas de gestión de una red privada.1 Esto se realiza estableciendo una conexión virtual punto a punto mediante el uso de conexiones dedicadas, encriptación o la combinación de ambos métodos.

2. REQUERIMIENTOS HARDWARE Y SOFTWARE

Sistema Operativo Windows XP o superiores.

3. INSTALACIÓN Y CONFIGURACIÓN DEL SERVICIO

A. Descargar el archivo FortiClientOnlineInstaller.exe desde el siguiente enlace: ftp://ftp.unicauca.edu.co/Documentos_Publicos/.Fortigate/, como se muestra en la figura 1.

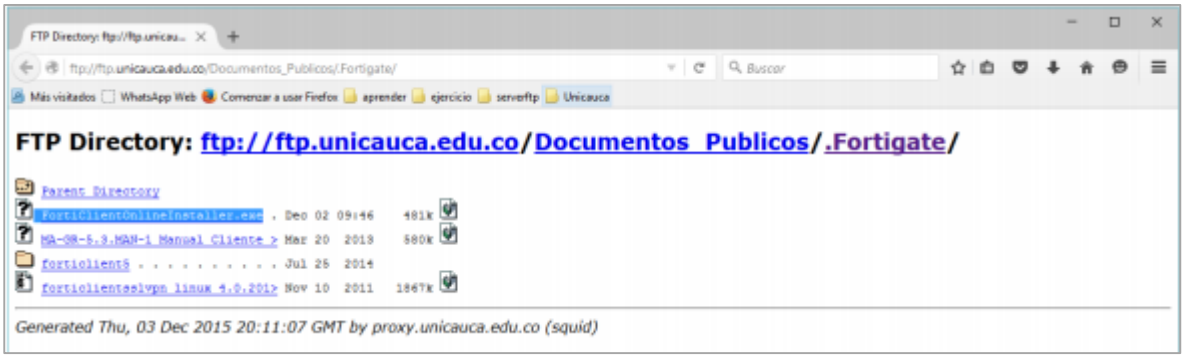


Figura 1. Descarga del archivo FortiClientOnlineInstaller.exe.

B. Llevar a cabo la instalación del programa con las condiciones por defecto que solicita el cliente, como se muestra en las figuras 2a, 2b, 2c, 2d, 2e, 2f y 2g.

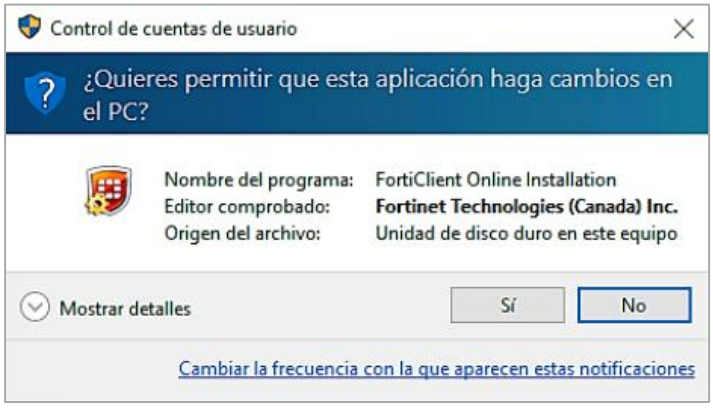


Figura 2a. Pasos de instalación (1).

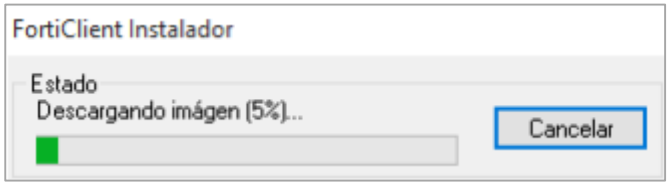


Figura 2b. Pasos de instalación (2).

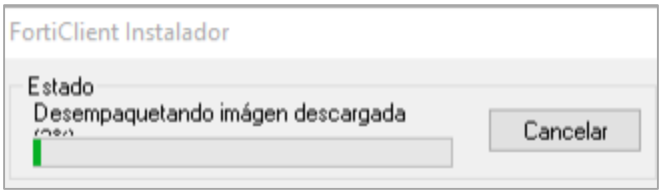


Figura 2c. Pasos de instalación (3).

Figura 2d. Pasos de instalación (4).

Nota: Seleccionar la opción “VPN Only” y continuar con la instalación.

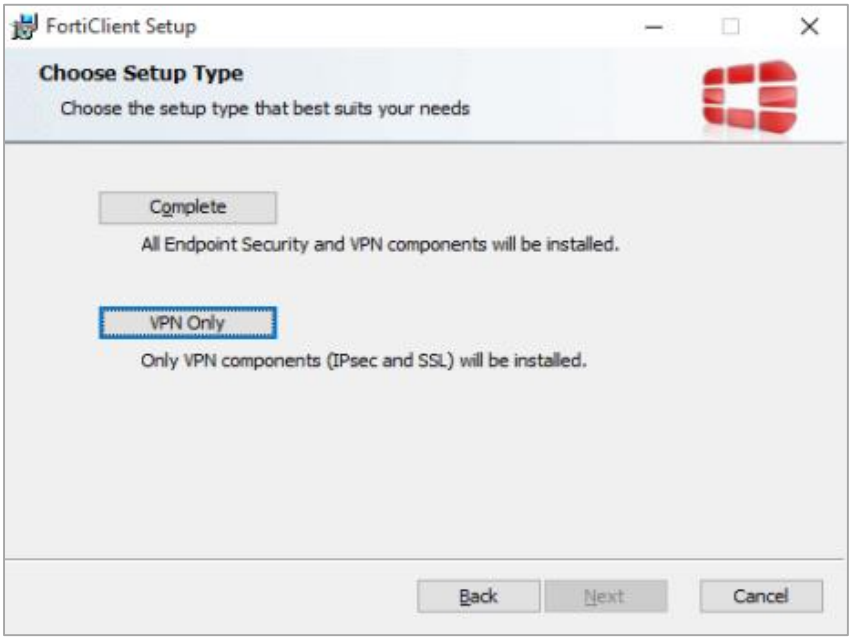


Figura 2e. Pasos de instalación (5).

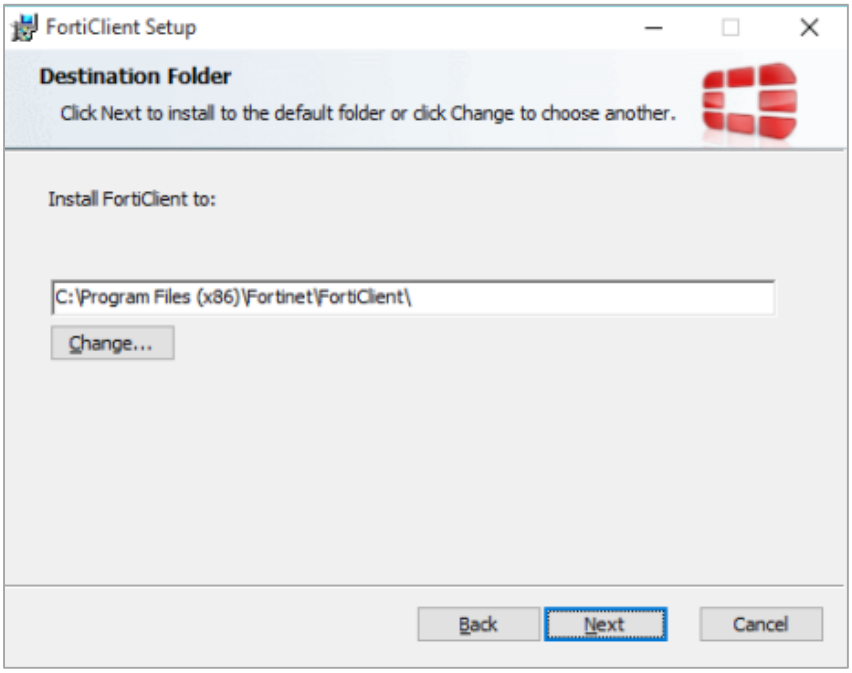


Figura 2f. Pasos de instalación (6).

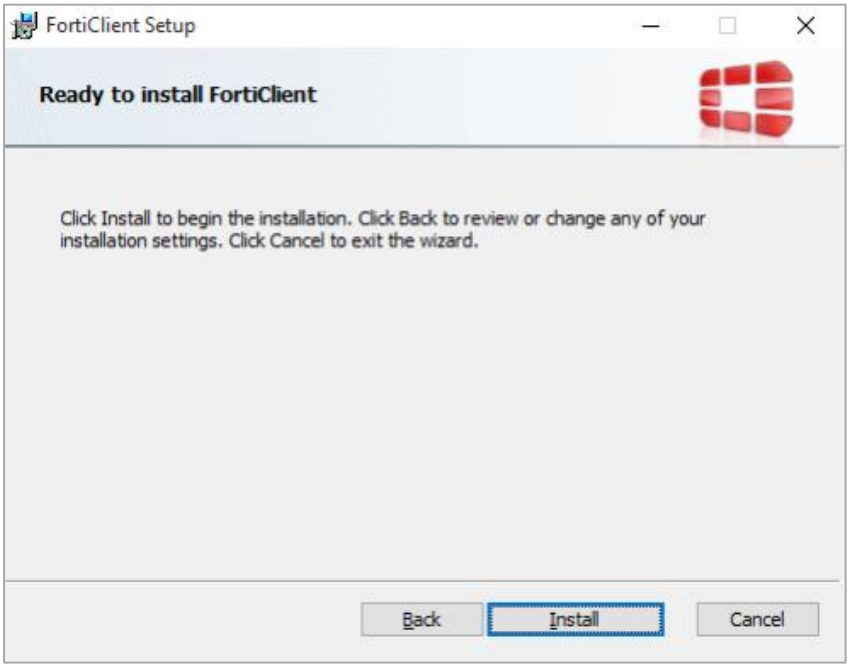


Figura 2g. Pasos de instalación (7).

C. Iniciar el cliente de VPN, para ello Ingrese al acceso directo ubicado en el escritorio:

Figura 3. Acceso directo a FortiClient.

- También puede iniciar el cliente dirigiéndose al siguiente directorio: Inicio → todas las aplicaciones y busque el cliente, que aparece como se muestra en la figura 4:

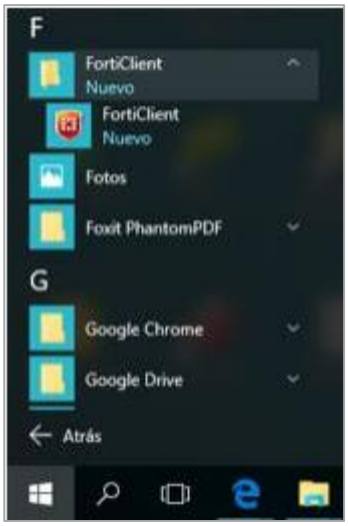


Figura 4. Acceso directo a FortiClient en el menú de inicio de Windows.

- Al ejecutar el acceso, se abre la siguiente interfaz, ingresar a la opción “Configurar VPN” (ver figura 5):

Figura 5. Configuración del cliente VPN.

- Seleccionar la opción VPN SSL, y llenar los campos respectivos, como se muestra en la figura 6:

  Nombre de Conexión = Nombre que usted desee, según la utilidad.

  Descripción = Descripción de cual va ser la funcionalidad de la VPN.

  Gateway Remoto = accesoremoto.unicauca.edu.co.

  Puerto = 10450

- Seleccionar la opción “Guardar login” y llenar el campo nombre de usuario.

  Nombre de Usuario = Es el usuario asignado por parte del área de Servicios Internet.

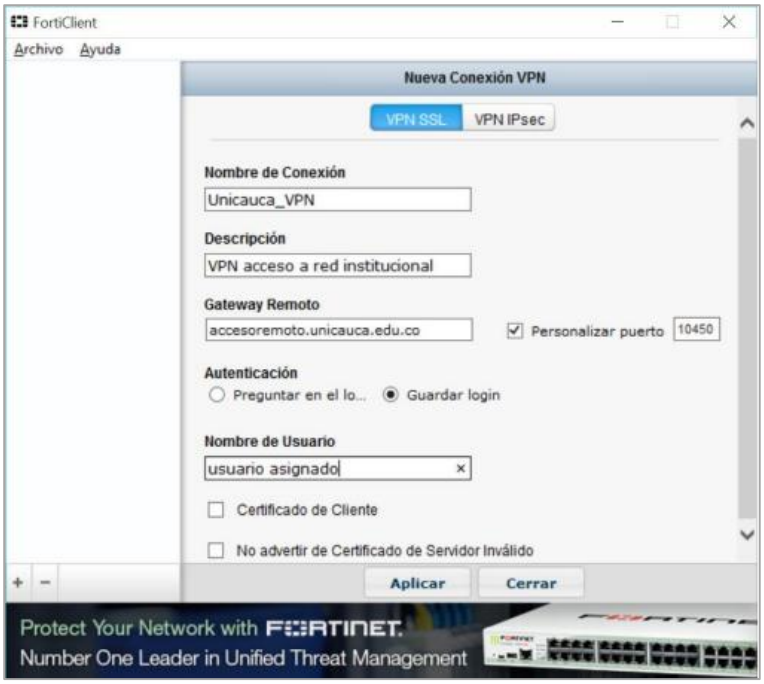


Figura 6. Campos requeridos para la instalación.

- Posteriormente seleccionar la opción aplicar e ingrese la contraseña. **Contraseña** = Asignada por el área de Servicios Internet. Dar click en Conectar (ver figura 7):

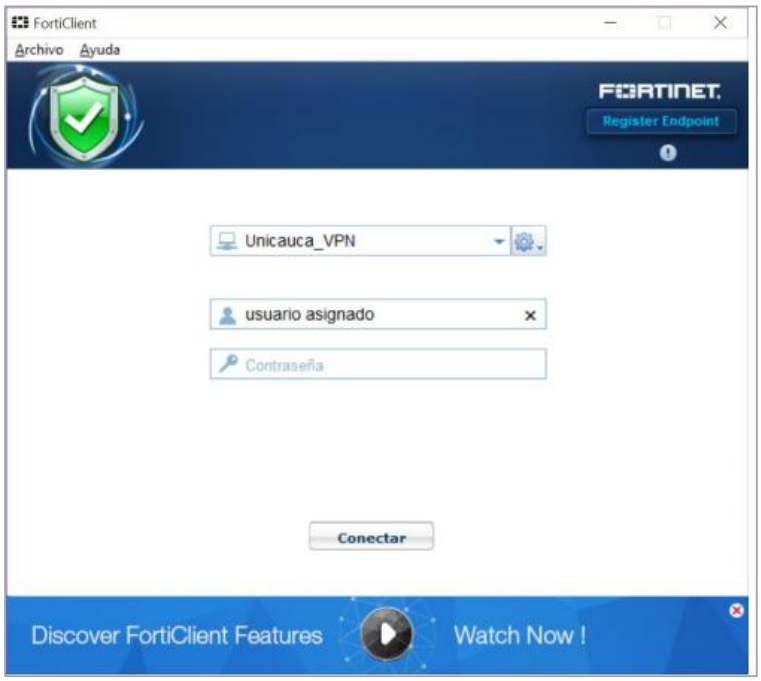


Figura 7. Usuario y contraseña de autentificación.

- Al conectarse debe aparecer la siguiente notificación:

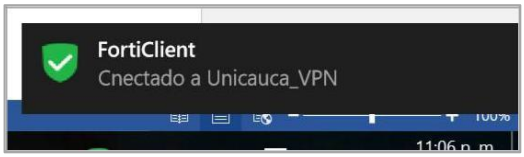


Figura 8. Conexión exitosa.

- Al establecer la conexión se puede usar los clientes de administración para ingresar al servicio solicitado.

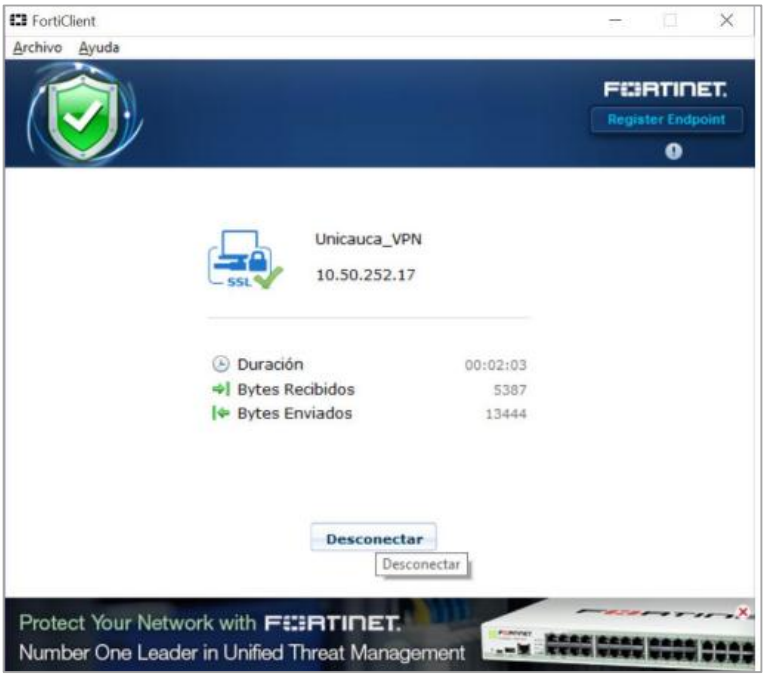


Figura 9. Conexión a la red privada.

- **Nota:** Al terminar de realizar las actividades por FAVOR desconectar del cliente VPN dando click en Desconectar.

D. ASPECTOS DE SEGURIDAD.

- La contraseña es personal, tome las medidas necesarias para mantenerla de forma segura.

- En caso de que la contraseña sea comprometida solicite al área de servicios de internet de la Universidad del Cauca el cambio de la misma. Cualquier anomalía o comportamiento extraño notifíquelo a la cuenta: serviciosinternet@unicauca.edu.co.

- El servicio de VPN es únicamente para el acceso a los sistemas administrados, por favor evite usarla para situaciones que estén por fuera del contexto institucional.

Anexo E

MANUAL DEL PROGRAMADOR PARA PRÁCTICAS DE FÍSICA MECÁNICA

Dentro del presente documento se muestra la estructura de codificación de cada una de las practicas virtuales contenidas en la aplicación web, haciendo énfasis en el archivo JavaScript, el cual es el de mayor importancia ya que en él se lleva a cabo la parte funcional de todo el sistema. En cuanto a los archivos HTML y CSS3, al desplegar la “herramienta para desarrolladores” dentro del navegador utilizado se tiene acceso a la estructura y diseño de cada uno de ellos, lo que facilita la comprension del codigo de los dos archivos mencionados.

I. ESTRUCTURA GENERAL DE JAVASCRIPT

A continuación se muestra la estructura general del código de cada uno de los archivos .js, en ella se observa que inicialmente todo va a estar contenido en la funcion **load (),** la cual se ejecuta apenas cada vez que la página se carga:

```
function load(){
  var game = new Phaser.Game (ancho, alto, Phaser.CANVAS, 'juego');
        var LogicaJuego = function () {}
        LogicaJuego.prototype = {
        Init: function (){},
        Preload: function (){},
        Créate: function (){},
        Update: function (){},
        }
        game.state.add('Game', LogicaJuego, true)
}
```

Posteriormente se crea la variable *game* en donde se hace uso del método **Game** de phaser y se especifican parámetros como el ancho y el alto de la caja que va a contener el escenario, el modo de ejecución y manejo de gráficas definido para

este caso como Phaser.CANVAS. Posteriormente se indica el nombre del identificador de la caja a la cual se quiere mandar el escenario (creado en el archivo HTML).

Dado que Phaser es un framework para videojuegos basado en estados, se requiere la creación de uno que contenga toda la logica de simulación, para ello se crea la variable *LogicaJuego*, la cual, según la estructura anterior, contiene las funciones predefinidas de Phaser, además de otras que fueron necesarias agregar.

Las funciones predefinidas de phaser tienen las siguientes caracteristicas:

- **Init:** Permite definir parámetros iniciales requeridos para el desarrollo.

- **Preload:** Función en donde se cargan los recursos que serán empleados, como por ejemplo las imágenes y los sprites.

- **Create:** Dentro de esta función se hace el llamado de otras funciones necesarias, como por ejemplo la función encargada de crear el escenario y las asociadas a los botones creados.

- **Update:** Es una función que tienen la propiedad de actualizarse constantemente, por lo tanto en ella se llevan a cabo procesos importantes como por ejemplo mover los sprites y activar las animaciones.

II. METODOS Y PROPIEDADES DE CANVAS

Para realizar los fondos de los escenarios de simulación, las figuras que dependen de los datos de entrada y la inclusión de texto dentro del escenario se hace uso de los métodos y propiedades de Canvas. A continuación se hace una breve descripción de cada uno de ellos, de modo que al ser empleados dentro de este documento no haya la necesidad de explicarlos nuevamente y solo baste con mencionarlos:

- **beginPath():** Es un método que indica que se va a iniciar una nueva figura, por lo tanto debe ser llamado antes de iniciar el trazado.

- **createLinearGradient(x_1,y_1, x_2, y_2):** Este método crea un objeto que será usado para implementar un gradiente lineal. Los atributos x_1, x_2, x_3 y x_4 equivalen a los puntos donde será agregado el gradiente y a partir de ellos se especifica inclinación deseada.

- **addColorStop(posicion, color):** Con este método se especifican los colores que se van a agregar al gradiente. El atributo posición es un valor entre 0 y 1 que indica donde comienza el degradado para el color especificado.

- **strokeStyle:** Es una propiedad con la que se define el color para el contorno de la figura.

- **fillStyle:** Con esta propiedad se declara el color para rellenar el interior de la figura.

- **fill():** Este método es empleado para rellenar la figura creada con el color especificado en la propiedad fillStyle.

- **stroke():** Este metodo realiza el trazado del contorno de la figura con el color especificado en la propiedad strokeStyle.

- **moveTo(x, y):** A través de este método el lápiz de dibujo es movido a una posición especifica para comenzar o continuar el trazado.

- **lineTo(x, y):** Por medio de este método se traza una linea recta desde la posición actual del lápiz hasta la posición indicada por los atributos x, y del mismo.

- **Rect(x, y, ancho, alto):** Este método genera un rectángulo. Los atributos x, y indican la posición desde donde será agregado.

- **arc(x, y, radio, angulo inicio, angulo final, dirección):** Este método genera un arco cuyo centro se especifican en los atributos x, y con un radio y desde un ángulo de inicio hasta uno de final, especificados por medio de los atributos correspondientes.

A continuación se muestra el proceso de codificación realizado dentro de cada una de las seis prácticas incluidas dentro de la aplicación web.

1. CODIFICACIÓN DE LA PRÁCTICA DE VECTORES

1.1. Actividad 1 – Práctica de vectores

Antes de seguir la estructura de desarrollo de Phaser, dentro del código JavaScript de la actividad 1 se definen las variables correspondientes al ancho y alto del escenario de simulación, como se muestra en la figura 1:

```
function load () {

var resolucion_ancho=1366
var resolucion_alto=768
var porcentaje_ancho_pantalla=0.6
var porcentaje_alto_pantalla=0.869
var ancho = resolucion_ancho* porcentaje_ancho_pantalla
var alto=resolucion_alto*porcentaje_alto_pantalla
```

Figura 1. Definición del ancho y alto del escenario de simulación.

Posteriorente se crea la variable que relaciona pixeles con metros, definida para este caso como *diezU*, a través de ella se determina que 10 metros equivalen a 51.4 pixeles. Este valor se fija de acuerdo a los requerimientos del sistema físico que se quiere representar, de esta manera todo el código debe quedar en función de esta variable, de modo que si se decide tomar otra relación el diseño del escenario no vaya a sufrir alteraciones. La variable *diezU* se define como se muestra en la figura 2:

```
var diezU = (ancho - 100) / 14 /*diezU equivale a 10 metros. El 100 representa 50 px de espacio hacia la izquierda y
50 hacia la derecha. 14 significa que se requieren 14 valores de 10 metros cada uno ya que en el eje x los valores
van desde -70 hasta 70*/
```

Figura 2. Creación de la variable para relacionar pixeles con metros.

También se define una variable de control inicializada en cero con la que se determina cuantas veces es presionado el botón cargar datos, con el fin de ir ingresando los vectores en una determinada posición de la tabla creada (ver figura 3).

```
var m = 0 //variable para controlar cuando el boton cargar datos es presionado
```

Figura 3. Variable de control para el ingres de los vectores.

Posterior a la declaración de las variables anteriores se procede a adaptar la lógica con base a la estructura de código de Phaser mostrada inicialmente. Para ello se inicia con la línea de código de la figura 4:

```
game = new Phaser.Game(ancho, alto, Phaser.CANVAS, 'juego');
```

Figura 4. Inicio del desarrollo con phaser.

También se crea la variable que va a contener el estado donde se agregan las funciones utilizadas, llamada para este caso *LogicaJuego*, dentro de la cual a su vez se define la variable *enMovimiento* con un valor inicial de *false*, esto con el fin de indicar cuando comienzan y finalizan los movimientos y las animaciones dentro de la simulación (ver figura 5).

```
var LogicaJuego = function () { // Creacion de la variable que contiene el estado donde se agrega la logica
    enMovimiento = false
}

LogicaJuego.prototype = { // JSON javascript object notation
```

Figura 5. Creacion de la variable que va a contener el estado donde se agrega la logica.

A partir de este punto se procede a la configuración de las funciones predefinidas de Phaser y se agregan otras que serán requeridas dentro de la lógica de simulación.

1.1.1. Función Init ()

Se inicia definiendo la función **Init ()**, como se muestra en el código de la figura 6, donde se crean las variables *centro* y *posición* del personaje empleadas para ubicarlo en una posición inicial correspondiente al centro del plano cartesiano y para conocer la ubicación de este cada vez que se agrega un vector, respectivamente (ver figura 6):

```
init: function () {

    centro = {
        x: game.width / 2,
        y: game.height / 2
    }

    posicionPersonaje = {
        x: centro.x,
        y: centro.y
    }
},
```

Figura 6. Definición de la función init.

1.1.2. Funcion Preload ()

En esta función se cargan las imágenes y el sprite correspondiente al personaje. En el caso de las imágenes se debe especificar el directorio donde se encuentran alojadas y para el caso del sprite se requiere indicar el tamaño de cada uno de los cuadros de imagen que lo conforman, como se muestra en la figura 7.

```
preload: function () {  //se cargan los recursos

    game.load.spritesheet('personaje','Actividad1/imagenes/personaje.png', 32, 48)
    game.load.spritesheet('puntosc','Actividad1/imagenes/rosaw.png')
    game.load.spritesheet('trian','Actividad1/imagenes/roja.png')
    game.load.spritesheet('trianr','Actividad1/imagenes/negra.png')

},
```

Figura 7. Definición de la función preload.

1.1.3. Función Create ()

Debido a que en esta función se realizan los procesos de crear el fondo, añadir imágenes y definir las animaciones, lo que se hace inicialmente es agregar el objeto bitmapData por medio de la variable *bmd*, ya que este es el objeto para gráficos que brinda Phaser, con el cual se da la opción de acceder a los métodos de Canvas para el diseño de figuras y despliegue de texto entre otros.

Con base en lo anterior se procede a crear el fondo azul sobre el que estará situado el plano cartesiano, haciendo uso de los métodos de Canvas **beginPath ()**, **createLinearGradient** (x_1, y_1, x_2, y_2), **fillRect** $(x, y,$ **ancho, alto)**, **addColorStop ()**, y la propiedad fillStyle. También se agrega el título de la práctica por medio del método **text** de bitmapData. Este proceso se puede ver en el código de la figura 8:

```
create: function () {

    bmd = this.add.bitmapData(game.width, game.height)   //se crea el bitmapdata para acceder a los metodos de canvas
    bmd.addToWorld()
    ctx=bmd.context;

    //fondo

    ctx.beginPath();
    var grd = ctx.createLinearGradient(0,0,0,game.height);
    grd.addColorStop(0.5, '#22B197');
    grd.addColorStop(1, '#24F3CD');
    ctx.fillStyle= grd;
    ctx.fillRect(0,0, game.width, game.height);

    bmd.text("VECTORES", centro.x-diezU*0.8,diezU/2, diezU/2.5+'px Aref Ruqaa', '#25231F')
```

Figura 8. Creacion del fondo del escenario de simulación.

Con el fondo creado se agregan las imágenes del escenario y el sprite que contiene los cuadros de movimiento del personaje, para esto dentro de los atributos se especifica la posición inicial (x, y) donde serán colocados estos elementos y el nombre dado en la función Preload. Para el caso del sprite se definen las animaciones haciendo uso del método **animations**. Como se ilustra en las tres últimas líneas de código de la figura 9 se agregaron tres animaciones denominadas derecha, centro e izquierda, en las cuales además fueron

especificados los cuadros de imágenes del sprite correspondientes a cada animación, así como también la velocidad de transición entre cada imagen.

```
rosa = game.add.sprite(game.width -diezU*2.5, diezU/10, 'puntosc')
rosa.scale.setTo(game.width / 1000);

personaje = game.add.sprite(centro.x, centro.y, 'personaje')
personaje.anchor.set(0.5, 1)

personaje.animations.add('derecha', [5, 6, 7, 8], 6, true)
personaje.animations.add('centro', [4], 6, true)
personaje.animations.add('izquierda', [0, 1, 2, 3], 6, true)
```

Figura 9. Proceso de agregar imágenes y definir animaciones.

Como siguiente paso se procede a llamar las funciones encargadas de dibujar el plano cartesiano y colocar los respectivos números (figura 10):

```
lineasSinRotar = this.calcularPuntosEje()
this.dibujarEjes()
this.numeros()
```

Figura 10. Llamado de las funciones encargadas de dibujar el plano cartesiano.

Estas funciones se explican a continuacion:

a) Función calcularPuntosEje ()

Para realizar las líneas correspondientes al plano cartesiano se emplea la propiedad line(x_1, y_1, x_2, y_2, color) del objeto bitmapData. La propiedad recibe como atributos los puntos de inicio (x_1, y_1) y final (x_2, y_2) de las líneas (ejes) que se van a trazar, por lo que se hace necesario determinar estos dos puntos. Para llevar a cabo este procedimiento inicialmente se define la variable *margenY***,** con la cual se indica la margen del eje vertical del plano (eje y). Dicha margen se calcula con base en los valores que va a contener el eje y, que para este caso son diez. De manera similar la margen en x se establece a través de la variable *margenX*, con un valor de 50.

Posteriormente se define la variable *líneas*, de tipo array, en la cual se guardan los puntos mencionados para la creación de cada línea por medio del método **push**. En los comentarios se indica la línea de código que corresponde a una determinada línea del eje cartesiano. Como se observa en el código de la figura 11, todo se deja en función de la variable *diezU*, ya que la separación entre cada valor del plano cartesiano es de 10 metros, y *diezU* representa este valor en pixeles. La función retorna la variable líneas con todos los puntos especificados.

```
calcularPuntosEje: function () {
    var margenY = (game.height - 10 * diezU) / 2
    var margenX=50

    var lineas = []
    lineas.push([ [ margenX, centro.y], [ game.width-margenX, centro.y ] ]) // linea principal del eje X
    lineas.push([ [ centro.x, margenY ], [ centro.x,  game.height - margenY ] ]) // linea principal del eje Y


    for(var m=0; m<=7*diezU; m+=diezU){
        lineas.push([ [ margenX+m,centro.y-5], [margenX+m,centro.y + 5 ] ]) //lineas pequeñas del eje x negativo
    }

    for(var m=centro.x; m<14*diezU; m+=diezU){
        lineas.push([ [ diezU+m,centro.y - 5], [diezU+m,centro.y + 5 ] ]) //lineas pequeñas del eje x positivo
    }

    for(var j=0; j<=5*diezU; j+=diezU){
        lineas.push([ [ centro.x-5,centro.y - j], [centro.x + 5,centro.y - j ] ])//lineas pequeñas del eje y positivo
    }

    for(var j=diezU; j<=5*diezU; j+=diezU){
        lineas.push([ [ centro.x-5,centro.y + j], [centro.x + 5,centro.y + j ] ])//lineas pequeñas del eje y negativo
    }

    return lineas

},
```

Figura 11. Funcion calcularPuntosEje.

b) Función dibujarEjes ()

Por medio de la función **dibujarEjes ()** se llama a otra función denominada **dibujarPuntosEjes ()**, la cual recibe como atributos los puntos guardados en la variable *líneas* de la función **calcularPuntosEje ()** y el valor correspondiente al color con que es dibujada cada línea (eje), ya que estos son los atributos que requiere el método **line**. En la estructura de código de la figura 12 se observa que dentro de la función **dibujarPuntosEje ()** se crean las respectivas líneas del sistema coordenado (haciendo uso del método **line)** a través de un ciclo *for* que controla la longitud que se ha dado a la variable *líneas*:

```
dibujarEjes: function () {
    this.dibujarPuntosEjes(lineasSinRotar, '#111111')
},

    dibujarPuntosEjes: function (lineas, color) {
    // lineas = [ l1, l2, ...] : i
    // l1 = [ p1, p2] : 0 - 1
    // p1 = [ x, y] : 0 - 1
    for (var i = 0; i < lineas.length; i++) {
        bmd.line(lineas[i][0][0], lineas[i][0][1], lineas[i][1][0], lineas[i][1][1], color)
    }
},
```

Figura 12. Función dibujarEjes.

c) Función Números ()

En esta función se hallan las posiciones donde será ubicado el texto asociado al plano cartesiano como por ejemplo el sistema numérico usado y la información de los ejes, Para ello se emplea el método **text**.

Continuando con la descripción de la función **create (),** dentro de ella son llamadas también las funciones asociadas a los botones creados: agregar vector, vector resultante y borrar vectores, para ello se emplea el evento de JavaScript *onclick* con el que se determina si uno de estos botones ha sido presionado. La lógica de la simulación va a depender de estas funciones. Como ejemplo se muestra la forma en que se define la lógica correspondiente a los botones vector resultante y borrar vectores (figura 13), donde se observa que tales botones solo tienen efecto cuando la variable enMovimiento toma el valor de *false*, es decir, cuando no hay ninguna animación activa. En caso de que esto se cumpla se llaman a las funciones respectivas.

```
document.getElementById('resultante').onclick = function (ev) {
  if (!enMovimiento) {
      if(m==0){alert('DEBE INGRESAR VECTORES')}
      else {this.dibujarResultante()}
  }
}.bind(this)


// evento boton limpiar
document.getElementById('limpiar').onclick = function (ev) {
  if (!enMovimiento) {
      this.limpiar()
      m=0;
  }
}.bind(this)
```

Figura 13. Definicion de las funciones asociadas a los botones.

1.1.4. Funciones asociadas al botón “Agregar vector”

Como se mencionó anteriormente, en la función **Create ()** se definen las funciones asociadas a cada botón. Para este caso se realiza un proceso en el que de acuerdo a los datos de entrada de cada vector ingresado (magnitud y sentido), se hallan internamente las coordenadas de posición en pixeles para cada uno de ellos, de esta manera se disponen de doce casos posibles, correspondientes a las opciones que puede tomar el sentido de cada vector respecto a los puntos cardinales (norte, sur, este y oeste). En la figura 14 se muestra como ejemplo dos de estos casos, en donde al final se obtienen las variables dx y dy , las cuales son de gran importancia para realizar los movimientos del personaje dentro del plano cartesiano.

```
    else if (sentido=="Oeste del Norte"){

       var xp = magnitud*Math.cos(90*Math.PI/180+teta) * diezU / 10
       var yp = - magnitud*Math.sin(90*Math.PI/180+teta) * diezU / 10
    }

    else if (sentido=="Oeste del Sur"){

       var xp = magnitud*Math.cos(270*Math.PI/180-teta) * diezU / 10
       var yp = - magnitud*Math.sin(270*Math.PI/180-teta) * diezU / 10
    }

    sen=sentido
dx=xp;
dy=yp;
```

Figura 14. Obtención de las variables *dx* y *dy*.

Una vez definidos los doce casos de posible orientación de los vectores, siempre que un botón es presionado, el sistema se encarga de llamar a las funciones **moverPersonaje(*dx, dy*)** y **agregarVectorTabla (m)** como se observa en el codigo de la figura 15.

```
this.moverPersonaje (dx, dy)

this.agregarVectorTabla(m)
```

Figura 15. Llamado de las funciones asociadas al botón cargar datos en la funcion Create ().

A continuacion se describe los procesos llevados a cabo dentro de estas dos funciones:

a) Función moverPersonaje(dx, dy):

En esta función se hallan los puntos de recorrido del sprite y se activan las animaciones correspondientes. Esta función recibe las variables respectivas de la posición en pixeles (dx, dy) de cada vector ingresado a través de las cuales se determina la ubicación del personaje dentro del plano, ya que por medio de las variables $posicionPersonaje.x \mathrel{+}= dx$ y $posicionPersonaje.y \mathrel{+}= dy$ se van sumando las posiciones en x y y cada vez que un vector es ingresado.

Para encontrar los puntos de recorrido del sprite se emplea el método de la interpolación lineal, el cual permite hallar la trayectoria lineal que pasa por ciertos puntos dados. Para aplicarlo a este caso particular solo basta con especificar el punto inicial y el punto final del desplazamiento del personaje, correspondientes a la cola y la cabeza de cada vector. Por medio de la variable *pasos* se determina cuantos puntos se quiere que recorra el sprite, y a través de un ciclo *for* que va

desde 0 hasta 1 (porcentaje de interpolación), se van guardando los puntos que este método retorna en la variable *this.camino*.

Para indicar que se van a activar los movimientos y las animaciones la variable *enMovimiento* toma el valor lógico *true*, de esta manera se despliegan las secuencias de animación correspondientes por medio del método **animations.play ()** según sea el caso (ver figura 16).

```
moverPersonaje: function (dx, dy) {

xv=dx/(diezU/10)
yv=dy/(diezU/10)
magnitud=Math.round(Math.sqrt(Math.pow(xv,2)+Math.pow(yv,2)))
if (magnitud<15){pasos=magnitud*6}
else{pasos=magnitud*3 } //se emplea para definir los puntos que recorrera el sprite

    posicionPersonaje.x += dx
    posicionPersonaje.y += dy

    this.camino = []
    var x = 1 / pasos

for (var i = 0; i <= 1; i += x) {
  var px = this.math.linearInterpolation([posicionPersonaje.x - dx, posicionPersonaje.x], i)
  var py = this.math.linearInterpolation([posicionPersonaje.y - dy, posicionPersonaje.y], i)

  this.camino.push( { x: px, y: py })
    }
    enMovimiento = true
    if (dx > 0)
        personaje.animations.play('derecha')
    else
        personaje.animations.play('izquierda')

    this.paso = 0
},
```

Figura 16. Funcion moverPersonaje(dx, dy).

b) Función agregarVectorTabla(m)

En esta función se asigna la posición dentro de la tabla en la que será agregada la información correspondiente a cada vector ingresado, para esto se emplea la variable m, la cual es inicializada con un valor de cero, de modo que cada vez que el botón agregar vector es presionado se aumenta en una unidad el conteo de las posiciones dentro de la tabla hasta llegar a cinco, valor que corresponde al límite de vectores que se permiten ingresar.

1.1.5. Funciones asociadas al botón “Vector resultante”

a) Función dibujarResultante ()

En esta función se traza la resultante de los vectores ingresados y se muestra la información asociada a este vector como datos de salida. Para trazar el vector

resultante se emplea el método **line** y para obtener la información que será mostrada (magnitud, dirección y coordenadas en x y y) se llama a la función **resultante ()**. Los demás pasos no se especifican ya que corresponden a la lógica empleada para rotar el triángulo que representa la dirección del vector resultante (ver figura 17).

```
dibujarResultante: function () {

    bmd1.line(centro.x, centro.y, posicionPersonaje.x, posicionPersonaje.y, '#D50000', 2.2)

    var resultante=this.resultante()
```

Figura 17. Función dibujarResultante ().

b) Función resultante ()

Retorna la posición final del personaje dentro del plano a través de sus coordenadas en x y y. Por medio de esta función se determinan los datos de posición, magnitud y dirección del vector resultante, como se muestra en la figura 18.

```
resultante: function () {

    //Manda los datos de la posicion final
    return {
        x: Math.round((posicionPersonaje.x - centro.x) * 10 / diezU * 100) / 100,
        y: -Math.round((posicionPersonaje.y - centro.y) * 10 / diezU * 100) / 100
    }
},
```

Figura 18. Función resultante ().

1.1.6. Funciones asociadas al botón “Borrar vectores”

La función **limpiar ()** es la asociada a este botón, a través de la cual se limpian los bitmapData creados mediante **bmd.clear()** y se restaura el fondo a su condición inicial, trasladando el personaje al centro del plano cartesiano a la espera del ingreso de nuevos vectores. También se limpian los input de texto donde se muestran los datos de salida, de esta manera el escenario de simulación queda como cuando se abre la aplicación por primera vez (ver figura 19).

```
limpiar: function () {

    bmd.clear()
    bmd1.clear()

    document.getElementById("i1").innerHTML=""
    document.getElementById("j1").innerHTML=""
    document.getElementById("i2").innerHTML=""
    document.getElementById("j2").innerHTML=""
    document.getElementById("i3").innerHTML=""
    document.getElementById("j3").innerHTML=""
    document.getElementById("i4").innerHTML=""
    document.getElementById("j4").innerHTML=""
    document.getElementById("i5").innerHTML=""
    document.getElementById("j5").innerHTML=""

    document.getElementById("despl").value=""
    document.getElementById("mag").value=""
    document.getElementById("angulo").value=""

    //fondo
    ctx.beginPath();
    var grd = ctx.createLinearGradient(0,0,0,game.height);
    grd.addColorStop(0.5, '#22B197');
    grd.addColorStop(1, '#24F3CD');
    ctx.fillStyle= grd;
    ctx.fillRect(0,0, game.width, game.height);
```

Figura 19. Función limpiar ().

1.1.7. Función Update ()

Gracias a que esta función de Phaser se actualiza constantemente, dentro de ella se asignan los puntos encontrados en la función **moverPersonaje(dx, dy)** a la posición del personaje dentro del plano cartesiano. Esta función solo tiene efecto cuando la variable *enMovimiento* tiene un valor lógico igual a *true*. La forma de asignar estos puntos es creando las variables *px* y *py*, las cuales, mediante un ciclo *for*, van tomando el valor de cada uno de los puntos guardados en la variable *this.camino* tanto para x como para y. Estas variables se asignan a la posición del personaje de la siguiente manera: $personaje.x = px$ y $personaje.y = py$, en el momento en que el personaje recorre cada uno de los puntos el sistema detiene las animaciones mediante el método **animations.stop()** y la variable *enMovimiento* vuelve a tomar el valor lógico *false*. Este proceso se observa en el código de la figura 20:

```
update: function () {

    var teta= parseFloat(document.getElementById("teta").value*Math.PI/180)
    if (enMovimiento) {
        var px = this.camino[this.paso].x
        var py = this.camino[this.paso].y
        personaje.x = px
        personaje.y = py

        bmd1.rect(px, py, 1.8, 1.8, '#fff'); //Crea el vector

        if (this.paso < pasos - 1) {
            this.paso++
        } else {


            enMovimiento = false
            personaje.animations.stop()

            personaje.animations.play('centro')
            personaje.animations.stop()
```

Figura 20. Función update ().

1.2. Actividad 2 – Práctica de vectores

Para la actividad 2 de vectores se maneja la misma lógica de programación que en la actividad 1, con la diferencia de que en este caso varían los datos de entrada, es decir, en vez de ingresar la magnitud y dirección de cada vector el usuario debe proporcionar sus coordenadas rectangulares, de esta forma el proceso para determinar su dirección dentro de los cuadrantes del plano se simplifica ya que no es necesario suministrar su orientación respecto a los puntos cardinales (norte, sur, este y oeste). La actividad 2 cuenta con una variante que permite la rotación de los ejes coordenados a través de un botón adicional denominado “rotar”. En este punto se explican las funciones asociadas a dicho botón:

1.2.1. Funciones asociadas al boton ‘Rotar’

En la funcion **Create ()** se establece que si el botón rotar es presionado, la variable *enMovimiento* toma el valor lógico *false* y hay por lo menos un vector ingresado, se llaman a las funciones **dibujarEjesRotados(θ)** y **datos_cambiados()** para llevar a cabo la rotación de los ejes del plano cartesiano y el cambio en las coordenadas de los vectores ingresados respecto a este nuevo sistema, como se observa en la estructura de código de la figura 21:

```
document.getElementById('rotar').onclick = function (ev) {
  if (!enMovimiento) {
      if(m==0){alert('DEBE INGRESAR VECTORES')}
      else{
      this.dibujarEjesRotados((document.getElementById("rotarInput").value)*Math.PI/180)
      this.datos_cambiados()}
  }
}.bind(this)
```

Figura 21. Definición de las funciones asociadas al botón rotar.

a continuación se describe de manera breve los pasos llevado a cabo en estas funciones:

a) Función dibujarEjesRotados(θ)

En esta función se lleva a cabo la rotación del plano cartesiano original por medio de un par de funcionales adicionales denominadas **rotarLineas(lineas,angulo)** y **dibujarPuntosEjes(lineasrotadas, color)**. La primera de ellas es la encargada de recibir los puntos encontrados en la función **calcularPuntosEjes(),** explicada en el desarrollo de las actividad 1. Dependiendo del ángulo suministrado en los datos de entrada (θ°) se llevará a cabo la rotación de estos ejes de acuerdo a las ecuaciones (de rotación) correspondientes. Una vez que se ha realizado internamente este proceso, por medio de la función **dibujarPuntosEjes(lineasrotadas, color)** se traza el plano cartesiano rotado, como se muestra en la figura 22:

```
dibujarEjesRotados: function (angulo) {
    var lineasRotadas = this.rotarLineas(this.lineasSinRotar, angulo)
    this.dibujarPuntosEjes(lineasRotadas, '#546E7A')
},
```

Figura 22. Función dibujarEjesRotados (θ).

b) Función Datos_cambiados ()

Finalmente, en esta función se toman las posiciones (x, y) de cada vector ingresado y se encuentran las nuevas posiciones (x', y') respecto al plano cartesiano rotado. Los nuevos valores de coordenadas hallados son consignados en su respectiva posicion dentro de una tabla.

2. CODIFICACIÓN DE LA PRÁCTICA DE MOVIMIENTO EN 2 DIMENSIONES

De forma similar a la practica de vectores, el desarrollo dentro del archivo JavaScript comienza con la definición de las variables que se van a emplear. También se añaden elementos que serán usados dentro de las simulaciones, como por ejemplo el archivo de audio que representa el sonido del agua cuando el balón cae dentro del lago, como se observa en la segunda línea de código de la figura 23. Igualmente, luego de definir el ancho y el alto del escenario se crea la variable *diezU*, encargada de la relación pixeles-metros. Por medio de esta variable se establece que dentro del escenario 10 metros equivaldrán a 44.84 pixeles.

```
function load () {

var audio = new Audio('imagenes/agua.wav')

var resolucion_ancho=1366
var resolucion_alto=768
var porcentaje_ancho_pantalla=0.75
var porcentaje_alto_pantalla=0.84
var ancho = resolucion_ancho* porcentaje_ancho_pantalla
var alto=resolucion_alto*porcentaje_alto_pantalla

var diezU = ancho/22.85 //dieU equivale a 10 metros por lo tanto para este caso 10 metros equivalen a 44.84 px.
```

Figura 23. Definición de las dimensiones del escenario.

.

También se crean las variables que van a contener los datos de las gráficas de movimiento de cada experimento. Dado que dentro del sistema se disponen de cuatro tipos de curvas para ilustrar gráficamente los eventos ocurridos en cada simulación (posición en x vs tiempo, posición en y vs tiempo, aceleración total vs tiempo y velocidad vs tiempo), se requieren cuatro variables para guardar los datos correspondientes a dichas curvas. De acuerdo a la figura 24 estas variables han sido definidas como: *datos_x_vs_t, datos_y_vs_t, datos_v_vs_t* y *datos_a_vs_t*, cada una con un arreglo de cuatro objetos correspondientes a los cuatro tramos del sistema que serán graficados.

```
//-------------------------variables donde se guardan las graficas-------------------------------
var datos_x_vs_t = [
   {values: []},
   {values: []},
   {values: []},
   {values: []}
   ];

var datos_y_vs_t = [
   {values: []},
   {values: []},
   {values: []},
   {values: []}
   ];

var datos_v_vs_t = [
   {values: []},
   {values: []},
   {values: []},
   {values: []}
   ];

var datos_a_vs_t = [
   {values: []},

   {values: []},
   {values: []},
   {values: []}
   ];
//-----------------------------------------------------------------------------------------------
```

Figura 24. Creación de las variables asociadas a cada gráfica de movimiento.

En este punto comienza el desarrollo con Phaser, el cual fue llevado a cabo de forma similar a como se hizo dentro de la práctica de vectores, y se hace una breve descripcion de las funciones utilizadas en los procesos más importantes dentro de la simulación.

2.1. Función Init ()

Dentro de esta función se crean las variables fijas necesarias para el desarrollo de la lógica de la simulación, como por ejemplo *base_rampas* y *base_rampas_metros*, variables correspondientes a la longitud de la base de las rampas en pixeles y metros respectivamente. Estas variables son empleadas en la realización de los gráficos, en la definición de los movimientos sobre los planos inclinados (valor en pixeles) y para calcular los valores numéricos asociados a dichos movimientos (valor en metros). Similarmente se define la variable *dist_inicio_rampa1_x* que equivale a la distancia en pixeles desde donde se comienza a graficar la rampa 1. También se establece la distancia desde el final de la rampa 2 hasta el lago, tanto en pixeles como en metros (ver figura 25).

```
init: function(){

base_rampas_metros=40 //base de las rampas en metros
base_rampas=diezU*base_rampas_metros/10  //base de las rampas en pixeles
dist_inicio_rampa1_x=diezU // distancia de inicio de la rampa 1
dist_rampa2_a_inicio_lago_metro=60 //distancias desde el final de la rampa 2 hasta el comienzo del lago en metros
dis_rampa2_a_inicio_lago=diezU*dist_rampa2_a_inicio_lago_metro/10 //distancias desde el final de la rampa 2 hasta el co
dist_inicio_rampa1y2_y= game.height-diezU //punto en 'y' donde reposan las rampas en pixeles
},
```

Figura 25. Función Init ().

2.2. Función Preload ()

En esta función se cargan las imágenes que forman parte del escenario de simulación, entre ellas se encuentra la figura de una pelota que representa el cuerpo que se desplaza a lo largo del sistema físico, un globo aerostático desde donde se pone en movimiento la pelota, un lago ubicado a cierta distancia de la base de la rampa 2 y un árbol como elemento de decoración del escenario.Para cada imagen es necesario especificar un nombre con el que pueden ser referenciadas y utlizadas, además del directorio o ubicación dentro del ordenador donde están guardadas en formato .png (ver figura 26).

```
preload: function () {

    //game.load.spritesheet('mano', 'imagenes/mano4.png', 65, 49)
    game.load.spritesheet('balon', 'imagenes/balonsprite.png', 46, 46)
    game.load.spritesheet('globo', 'imagenes/globorojo1.png')
    game.load.spritesheet('globo1', 'imagenes/globorojo2.png')
    game.load.spritesheet('lago', 'imagenes/Lago2D-2.png', 364, 100)
    game.load.spritesheet('nube', 'imagenes/nubecita1a.png')
    game.load.spritesheet('nube1', 'imagenes/Nube2.png')
    game.load.spritesheet('nube2', 'imagenes/Nubecita2a.png')
    game.load.spritesheet('arbol', 'imagenes/arbol3.png')

    },
```

Figura 26. Definición de la función Preload ().

2.3. Función Create ()

A partir de la presente práctica (y para todas las que siguen) se lleva a cabo un proceso de codificación un poco diferente al realizado en la práctica de vectores, ya que tanto la creación del fondo como el diseño del escenario se realizan en funciones separadas. En cuanto a la función **Create ()**, dentro de ella se realizan los procesos de validación de los cambios de datos en el panel de entrada, llamado a las funciones asociadas a los botones y llamado a las funciones encargadas de crear el escenario y actualizar gráficas. A continuación se explica el primero de estos procesos (validación los cambios en los datos del panel de entrada).

Como se observa en la figura 27, se emplea la funcion **jQuery change()** para detectar los cambios de valor en los inputs de entrada del panel de datos iniciales, de esta forma dicha función se evalúa para todos estos valores, entre los que se tienen: los ángulos de elevación de las dos rampas, la altura desde donde se suelta la pelota, la velocidad inicial del balón cuando parte de la rampa 2 y el selector donde se elige la opción de simulación. Por lo tanto, cuando hay un cambio en estos datos iniciales la variable k toma un valor de cero.

```
//--------- Empleado para verificar cuando hay un cambio en los inputs de entrada---------------
   $(document).ready(function(){
                $("#teta2").change(function(){
                    k=0
                });
                $("#teta1").change(function(){
                    k=0
                });
                 $("#altura").change(function(){
                    k=0
                });

                 $("#selec").change(function(){
                    k=0
                });

                 $("#vo").change(function(){
                    k=0
                });
            });
//-----------------------------------------------------------------------------------------------
```

Figura 27. Validación de los cambios en los datos del panel de entrada.

La variable k se emplea para controlar el comportamiento de los botones y puede tomar los siguientes valores:

- Si hay una variación en los datos de entrada toma el valor de 0.
- Si el botón Cargar datos es presionado toma el valor de 1.
- Si el botón Simulación es presionado toma el valor de 2.
- Si el botón Resultados es presionado toma el valor de 3.

A continuación se muestran las funciones asociadas a cada botón (figura 28), los cuales tienen los siguientes identificadores: Cargar datos (enviar), Simulación (enviar1) y Resultados (enviar 2).

Cuando el botón Cargar datos es presionado se llaman las funciones **cargardatos()** y **limpiar_graficas()**, y la variable k toma el valor de 1. En caso contrario, si se presiona el botón Simulación y k es diferente de cero, se llama la función **mover_balon_caida()** y la variable k toma el valor de 2. Por último, si se oprime el botón Resultados k es igual a 2 (indica que ya se realizó el proceso de simulación) se llama a la funcion **resultados ()**.

En la figura 28 se observa que dentro de la función **Create ()** también es llamada la función **inicializar_graficas (),** con la que se despliegan los ejes para cada gráfica cuando se abre la aplicación.

```
document.getElementById('Enviar').onclick = function (ev) {
        if (!balonMovimiento) {
        bmd.clear()
        this.cargardatos()
        this.limpiar_graficas()
        k=1
    }
  }.bind(this)

  document.getElementById('Enviar1').onclick = function (ev) {
        if (!balonMovimiento) {

        if(k==0){alert('DEBE CARGAR DATOS')}
        else{this.mover_balon_caida()
            k=2}

        }
  }.bind(this)

  document.getElementById('Enviar2').onclick = function (ev) {
        if (!balonMovimiento) {
            if(k==2){this.resultados()
                k=3}
            else if(k==0){alert('DEBE CARGAR DATOS')}
            else if(k==1) {alert('DEBE SIMULAR PRIMERO')}

        }
  }.bind(this)

  this.inicializar_graficas()

},
```

Figura 28. Definición de las funciones asociadas a los botones.

2.4. Funciones asociadas al botón Cargar datos

A continuación se describen las funciones que son llamadas cada vez que el botón Cargar datos es presionado:

a) Función Cargardatos ()

En esta función se llevan a cabo los procesos para la creación del fondo, los títulos de la interfaz y las rampas inclinadas, todo por medio de los métodos de Canvas. Además se agregan las imágenes incluidas dentro del escenario de simulación y se definen las diferentes animaciones. De esta manera en la figura 29 se muestra la creación del fondo del escenario, partiendo del método **createLinearGradient ()**, dentro del cual se implementa un gradiente con cuatro colores para representar el cielo y el suelo de dicho escenario. También se coloca el título de la simulación mediante el método **text** de bitmapData.

```
//fondo

ctx.beginPath();
var grd = ctx.createLinearGradient(0,0,0,game.height);
grd.addColorStop(0.0, '#168598');
grd.addColorStop(0.7, '#E3ED89');
grd.addColorStop(0.7, '#51642A');
grd.addColorStop(0.9, '#C1E685');

ctx.fillStyle= grd;
ctx.fillRect(0,0, game.width, game.height);

//titulo

bmd.text("MOVIMIENTO EN 2 DIMENSIONES", diezU*7, diezU*1, diezU/2+'px Cursive', '#E1F5FE')
```

Figura 29. Creación del fondo del escenario.

De manera similar en el código de la figura 30 se muestra el proceso de creación de la rampa 1 (plano inclinado descendente), dentro del cual es creado inicialmente el gradiente de colores con el que se va a rellenar el interior de dicha rampa. Posteriormente se asignan los colores del relleno y el contorno del plano inclinado a las propiedades fillStyle y strokeStyle respectivamente. Luego se requiere indicar cada uno de los puntos a partir de los cuales se conforma la figura deseada. Para encontrar algunos de estos puntos se emplea la semejanza de triángulos.

```
//rampa1
ctx.beginPath();
var grd1 = ctx.createLinearGradient(dist_inicio_rampa1_x,game.height-altura_rampa1,base_rampas+dist_inicio_rampa1_x
grd1.addColorStop(0.1, '#9F9265');
grd1.addColorStop(0.9, '#8F8354');
ctx.fillStyle= grd1;
ctx.strokeStyle="white";
ctx.moveTo(dist_inicio_rampa1_x ,dist_inicio_rampa1y2_y);
ctx.lineTo(dist_inicio_rampa1_x,dist_inicio_rampa1y2_y-altura_rampa1);
ctx.lineTo(base_rampas+dist_inicio_rampa1_x,dist_inicio_rampa1y2_y);
ctx.fill();
ctx.stroke();

ctx.beginPath();
var grd2 = ctx.createLinearGradient(diezU,game.height-altura_rampa1-diezU*1.5,base_rampas+diezU+diezU,game.height-a
grd2.addColorStop(0, '#9F9265');
grd2.addColorStop(0.8, '#DDD0A2');
ctx.fillStyle= grd2;
ctx.moveTo(dist_inicio_rampa1_x ,dist_inicio_rampa1y2_y-altura_rampa1);
ctx.lineTo(dist_inicio_rampa1_x+diezU/2 ,game.height-(altura_rampa1*(base_rampas-diezU/2)/(base_rampas))-diezU*1.5)
ctx.lineTo(base_rampas+dist_inicio_rampa1_x, game.height- diezU*1.5 );
ctx.lineTo(base_rampas+dist_inicio_rampa1_x,dist_inicio_rampa1y2_y);
ctx.strokeStyle="white";
ctx.fill();
ctx.stroke();
bmd.text('θ1='+teta11+"°", diezU*3.6, game.height-diezU*1.1, diezU/3.7+'px Calibri ', '#F7BF71')
```

Figura 30. Creacion de la rampa 1.

Para la creación de la rampa 2 se sigue un procedimiento similar al realizado con la rampa 1. En este punto se procede a colocar la información adjunta sobre cada uno de los elementos que conforman el escenario, como es el caso de las dos rampas (longitud de la base), en el lago (diámetro del lago circular) y en los puntos que limitan los diferentes trayectos, como se observa en el código de la figura 31. En ella se muestra como ejemplo la manera en que se colocan los puntos que diferencian cada uno de los tramos que componen el sistema físico utilizando el método **text**, dentro del cual se requiere especificar para cada texto el sitio de ubicación dentro del escenario, el tamaño y el color.

```
bmd.text(base_rampas_metros+'m', dist_inicio_rampa1_x+base_rampas+diezU*1.6,game.height-diezU/1.5, diezU/2.8+'px Calibri',
bmd.text('A', dist_inicio_rampa1_x-diezU/3.5,game.height-altura_rampa1-diezU+diezU/5, diezU/2.8+'px Calibri', '#DF3E54')
bmd.text('B', dist_inicio_rampa1_x-diezU/9+base_rampas,game.height-diezU+diezU/3.2, diezU/2.8+'px Calibri', '#DF3E54')
bmd.text('C', dist_inicio_rampa1_x+diezU/24+base_rampas*2,game.height-altura_rampa2-diezU+diezU/3, diezU/2.8+'px Calibri',
bmd.text('O', dist_inicio_rampa1_x-diezU/2,game.height-diezU-altura_rampa1-(altura*diezU/10), diezU/2.8+'px Calibri', '#DF
```

Figura 31. Texto adjunto a los trayectos del sistema.

Posteriormente se agregan las imágenes del escenario, correspondientes a las nubes, el árbol, el lago, el globo y el balón. El punto inicial donde es agregado el balón depende de la opción de simulación escogida, ya que de acuerdo al código de la figura 32, si la opción de simulación escogida es el “tramo O-A-B” o el “tramo Total”, el balón será ubicado en el punto O , llevando a cabo un movimiento de caída libre. Por su parte, si la opción de simulación es el “tramo B-C-D”, la pelota inicia su movimiento desde el punto B (punto de partida de la rampa 2).

```
if((opcion=="Tramo O-A-B")||(opcion=="Tramo Total")){
balon = game.add.sprite(dist_inicio_rampa1_x, dist_inicio_rampa1y2_y-altura_rampa1-diezU/1.5-(altura*diezU/10), 'balon')
balon.scale.setTo(game.width / 1607.14);
//balon.anchor.set(0.5);
}
else{
balon = game.add.sprite(dist_inicio_rampa1_x+base_rampas, dist_inicio_rampa1y2_y-balon.height-(altura_rampa2*(balon.width/
balon.scale.setTo(game.width / 1607.14);
}
```

Figura 32. Forma de agregar el balón al escenario de simulación.

En la figura 33 se define una animación que simule a la pelota rodando:

```
balon.animations.add('uno', [1,2,3,4], 7, true)
```

Figura 33. Animación del balón rodando.

Finalmente cada vez que este botón es presionado se requiere que se limpien las tablas de resultados numéricos de simulaciones pasadas, para ello se utiliza la propiedad innerHTML, la cual permite la manipulación de este tipo de texto (figura 34).

```
document.getElementById("tab").innerHTML=""
document.getElementById("i1").innerHTML=""
document.getElementById("j1").innerHTML=""
document.getElementById("i2").innerHTML=""
document.getElementById("j2").innerHTML=""
document.getElementById("i3").innerHTML=""
document.getElementById("j3").innerHTML=""
document.getElementById("i4").innerHTML=""
document.getElementById("j4").innerHTML=""
document.getElementById("i5").innerHTML=""
document.getElementById("j5").innerHTML=""
document.getElementById("i6").innerHTML=""
document.getElementById("j6").innerHTML=""
document.getElementById("i7").innerHTML=""
document.getElementById("j7").innerHTML=""
document.getElementById("i8").innerHTML=""
document.getElementById("j8").innerHTML=""
document.getElementById("i9").innerHTML=""
document.getElementById("j9").innerHTML=""
document.getElementById("i10").innerHTML=""
document.getElementById("j10").innerHTML=""
document.getElementById("i11").innerHTML=""
document.getElementById("j11").innerHTML=""
document.getElementById("i12").innerHTML=""
document.getElementById("j12").innerHTML=""
```

Figura 34. Proceso para limpiar las tablas de resultados.

b) Función Limpiar gráficas ()

Esta función es la encargada de borrar la información que brindan las gráficas de simulaciones anteriores, de esta manera los ejes cartesianos quedan vacios, como cuando la aplicación es abierta por primera vez. Para ello hay que manipular las variables que contienen la información de cada gráfica, además de los valores de rango y dominio de las mismas, asignando a cada una un *array* vacío, como se muestra en el código de la figura 35.

```
limpiar_graficas: function(){
    datos_x_vs_t[0].values = [];
    datos_x_vs_t[1].values = [];
    datos_x_vs_t[2].values = [];
    datos_x_vs_t[3].values = [];
    grafica_x_vs_t.option('range', [])
    grafica_x_vs_t.option('domain', [])
    grafica_x_vs_t.update(datos_x_vs_t)

    datos_y_vs_t[0].values = [];
    datos_y_vs_t[1].values = [];
    datos_y_vs_t[2].values = [];
    datos_y_vs_t[3].values = [];
    grafica_y_vs_t.option('range', [])
    grafica_y_vs_t.option('domain', [])
    grafica_y_vs_t.update(datos_y_vs_t)

    datos_v_vs_t[0].values = [];
    datos_v_vs_t[1].values = [];
    datos_v_vs_t[2].values = [];
    datos_v_vs_t[3].values = [];
    grafica_v_vs_t.option('range', [])
    grafica_v_vs_t.option('domain', [])
    grafica_v_vs_t.update(datos_v_vs_t)

    datos_a_vs_t[0].values = [];
    datos_a_vs_t[1].values = [];
    datos_a_vs_t[2].values = [];
    datos_a_vs_t[3].values = [];
    grafica_a_vs_t.option('range', [])
    grafica_a_vs_t.option('domain', [])
    grafica_a_vs_t.update(datos_a_vs_t)
```

Figura 35. Función limpiar gráficas.

2.5. Función asociada al botón Simulación

La función asociada a este botón se denomina **mover_balon_caida ()** y en ella se realiza el proceso para determinar la trayectoria de movimiento del balón, dependiendo de la opción de simulación escogida. Para ello se realizan los cálculos numéricos correspondientes al tramo O-A-B, ya que estos son independientes de la opción de simulación, y se determinan los puntos de esta trayectoria lineal por medio del método matemático denominado interpolación lineal de Phaser, Inicialmente se dan cinco puntos (p1, p2, p3, p4, p5) y por medio

de este método se obtienen los otros puntos de dicha trayectoria vertical, como lo muestra la figura 36:

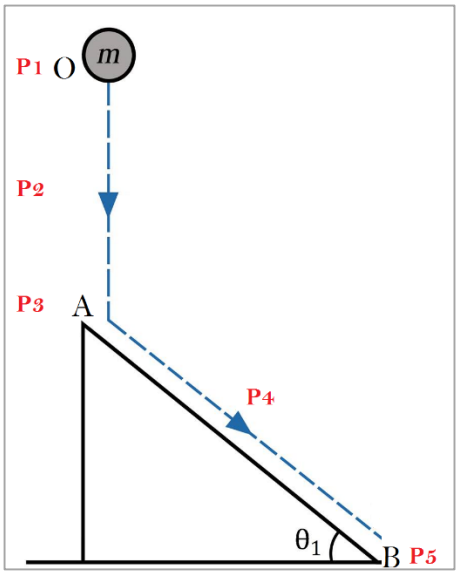


Figura 36. Puntos iniciales de la trayectoria.

Los puntos determinados son guardados en forma de arreglo dentro de la variable *this.camino* a través del método push (figura 37).

```
this.camino = []
this.camino1 = []
this.camino2 = []
this.camino3 = []

ta=Math.sqrt(2*altura/9.8)
va=Math.sqrt(2*9.8*altura)
d1=base_rampas_metros/Math.cos(teta1)
vb=Math.sqrt(Math.pow(va,2)+(2*9.8*Math.sin(teta1)*d1))
tb=(-va+Math.sqrt(Math.pow(va,2)+(2*9.8*Math.sin(teta1)*d1)))/(9.8*Math.sin(teta1))
//vb=va+9.8*Math.sin(teta1)*tb
if(opcion=="Tramo O-A-B"){
this.pasos = 100
var x = 1 / this.pasos
for (var i = 0; i <= 1; i += x) {
var px = this.math.linearInterpolation([dist_inicio_rampa1_x,dist_inicio_rampa1_x, dist_inicio_rampa1_x, dist_inici
var py = this.math.linearInterpolation([dist_inicio_rampa1y2_y-altura_rampa1-diezU/3.6-(altura*diezU/10), dist_inic
dist_inicio_rampa1y2_y-altura_rampa1-balon.height,
dist_inicio_rampa1y2_y-balon.height-(altura_rampa1*(diezU*2)/(base_rampas)),
dist_inicio_rampa1y2_y-balon.height-diezU/5], i);

this.camino.push( { x: px, y: py })


}
```

Figura 37. Definición de la trayectoria dentro del tramo O-A-B.

Posteriormente se realizan los cálculos matemáticos necesarios y se hallan los puntos de la trayectoria seguida por la pelota dentro el tramo B-C-D y el tramo total, los cuales tienen en común el movimiento parabólico realizado por el cuerpo antes de llegar al suelo. Primero se define una variable, llamada en este caso m, con la que se valida si la velocidad que lleva el balón es suficiente para realizar el

movimiento parabólico. Esta variable se halla a partir de la velocidad final sobre la rampa 2, la cual dentro de la ecuación pertinente se hace igual a cero, y se halla la distancia que alcanza a recorrer dentro de este plano inclinado, cambiando el sistema de referencia para dejarla en función de la base de dicha rampa. De esta manera, en la figura 38 se muestra el código correspondiente a este proceso, en donde m varía para el tramo total y para el tramo B-C-D, ya que la velocidad con la que parte el balón de la rampa 2 difiere para estos dos casos, además se tiene la condición de que si m es mayor que la base de la rampa 2, el balón realiza el movimiento parabólico, en caso contrario no alcanza a salir de la rampa 2.

```
if(opcion=="Tramo Total"){
m=(Math.pow(vb,2)*Math.cos(teta2))/(2*9.8*Math.sin(teta2))
}
else if(opcion=="Tramo B-C-D"){
m=(Math.pow(vo,2)*Math.cos(teta2))/(2*9.8*Math.sin(teta2))
}

if(m>base_rampas_metros){
```

Figura 38. Variable para verificar si el balón sale de la rampa 2.

Con base en lo anterior, si el balón sobrepasa la rampa 2, para los trayectos lineales se emplea nuevamente el método de interpolación lineal, pero para el caso del movimiento parabólico se emplean las ecuaciones de posición de dicho movimiento adaptadas al sistema planteado (figura 39).

```
for (var q = 0; q <= td; q+=y ) {

    var ox=  (vox*q)*diezU/10+(base_rampas*2+dist_inicio_rampa1_x)
    var oy=  dist_inicio_rampa1y2_y -(((-9.8*Math.pow(q,2))/2)+voy*q+hrampa2)*diezU/10-balon.height
    this.camino1.push({ x: ox, y: oy })

}
```

Figura 39. Forma de hallar los puntos que recorre el balón dentro del movimiento parabólico.

2.6. Funciones asociadas al botón Resultados

a) Función Resultados ()

Para esta función, dependiendo de la opción de simulación se obtienen los datos de salida que serán mostrados en la tabla respectiva y se llaman las funciones donde se realizan las gráficas. A continuación se muestra como ejemplo los resultados para el tramo O-A-B (figura 40) donde por medio de la propiedad innerHTML se le asigna a cada identificador de la tabla creada el correspondiente dato de salida y al final del código se llama a la función **tramo_OAB ()** donde se lleva a cabo el proceso de graficar.

```
if((opcion=="Tramo O-A-B")&& (ta)){
    document.getElementById("tab").innerHTML="TRAMO O-A-B"
    document.getElementById("i1").innerHTML="vo"
    document.getElementById("j1").innerHTML="0"
    document.getElementById("i2").innerHTML="vA"
    document.getElementById("j2").innerHTML= Math.round(va*100)/100
    document.getElementById("i3").innerHTML="vB"
    document.getElementById("j3").innerHTML=Math.round(vb*100)/100
    document.getElementById("i4").innerHTML="tA"
    document.getElementById("j4").innerHTML=Math.round(ta*100)/100
    document.getElementById("i5").innerHTML="tB"
    document.getElementById("j5").innerHTML=Math.round(tb*100)/100
    document.getElementById("i6").innerHTML="ho"
    document.getElementById("j6").innerHTML= altura
    document.getElementById("i7").innerHTML="a_cl"
    document.getElementById("j7").innerHTML=Math.round(9.8*100)/100
    document.getElementById("i8").innerHTML="a_R1"
    document.getElementById("j8").innerHTML=Math.round(9.8*Math.sin(teta1)*100)/100
    document.getElementById("i9").innerHTML="L_R1"
    document.getElementById("j9").innerHTML=Math.round(base_rampas_metros/Math.cos(teta1)*100)/100
    document.getElementById("i10").innerHTML="-"
    document.getElementById("j10").innerHTML="-"
    document.getElementById("i11").innerHTML="-"
    document.getElementById("j11").innerHTML="-"
    document.getElementById("i12").innerHTML="-"
    document.getElementById("j12").innerHTML="-"

    this.tramo_OAB()

    }
```

Figura 40. Función resultados () - Opción tramo O-A-B.

b) Función Tramo_OAB ()

En esta función se realizan las gráficas correspondientes al tramo O-A-B, y dado que se involucran dos trayectos, mediante los tiempos t_a (caída libre) y t_b (movimiento sobre plano inclinado descendente) se hallan, por medio de un ciclo *for*, los valores correspondientes a la posición en x, posición en y, velocidad y aceleración, para ser asignados a las variables creadas inicialmente, como se muestra en la figura 41.

```
for (var taa = 0; taa <= ta; taa += 0.01) {



    datos_x_vs_t[0].values.push({x: taa, y: 0});
    datos_y_vs_t[0].values.push({x: taa, y: (((-9.8*Math.pow(taa,2))/2)+htotal)});
    datos_v_vs_t[0].values.push({x: taa, y: 9.8*taa});
    datos_a_vs_t[0].values.push({x: taa, y: 9.8});


}

for (var tbb = ta; tbb <= tb+ta; tbb += 0.01) {


    x2=((va*(tbb-ta)) + (0.5*9.8*Math.sin(teta1)*Math.pow((tbb-ta),2)))*Math.cos(teta1)
    y2=hrampa1-((va*(tbb-ta)) + (0.5*9.8*Math.sin(teta1)*Math.pow((tbb-ta),2)))*Math.sin(teta1)
    v2=va+9.8*Math.sin(teta1)*(tbb-ta);
    a2=9.8*Math.sin(teta1)
    datos_x_vs_t[1].values.push({x: tbb, y: x2  });
    datos_y_vs_t[1].values.push({x: tbb, y: y2  });
    datos_v_vs_t[1].values.push({x: tbb, y: v2  });
    datos_a_vs_t[1].values.push({x: tbb, y: a2  });
}
```

Figura 41. Forma de asignar los valores a graficar dentro de las variables creadas inicialmente.

Cuando se tienen los datos a graficar se procede a especificar el rango y el dominio para cada curva de movimiento. Las variables anteriores son asignadas a cada gráfica mediante el método **Update** (ver figura 42).

```
grafica_x_vs_t.option('ticks', { top: 0, right: 0, bottom: 8, left: 8 })
grafica_x_vs_t.option('range', [0, base_rampas_metros])
grafica_x_vs_t.option('domain', [0, Math.ceil(ta+tb)])
grafica_x_vs_t.update(datos_x_vs_t)

grafica_y_vs_t.option('ticks', { top: 0, right: 0, bottom: 8, left: 8 })
grafica_y_vs_t.option('range', [0, htotal+5])
grafica_y_vs_t.option('domain', [0, Math.ceil(ta+tb)])
grafica_y_vs_t.update(datos_y_vs_t)

grafica_v_vs_t.option('ticks', { top: 0, right: 0, bottom: 8, left: 8 })
grafica_v_vs_t.option('range', [0, vb+5])
grafica_v_vs_t.option('domain', [0, Math.ceil(ta+tb)])
grafica_v_vs_t.update(datos_v_vs_t)

grafica_a_vs_t.option('ticks', { top: 0, right: 0, bottom: 8, left: 8 })
grafica_a_vs_t.option('range', [0, 10])
grafica_a_vs_t.option('domain', [0, Math.ceil(ta+tb)])
grafica_a_vs_t.update(datos_a_vs_t)
```

Figura 42. Definición del dominio y el rango de las gráficas – Asignación de las variables a graficar.

c) Función tramo_BCD ()

Se realiza el mismo proceso que en la función del tramo O-A-B pero con la información del trayecto B-C-D. Para encontrar los valores a graficar se trabaja con los tiempos t_c (movimiento en la rampa 2) y t_d (movimiento parabólico).

d) Función tramo_total ()

En esta función se realizan las gráficas para el tramo total, de esta manera a través de un ciclo *for*, evaluado para los cuatro tiempos t_a, t_b, t_c y t_d, se encuentran los valores pertinentes y se asignan a las gráficas respectivas.

3. CODIFICACIÓN DE LA PRÁCTICA DE MOVIMIENTO CIRCULAR

En la figura 43 se muestra el codigo inicial en JavaScript dentro del cual se agrega un audio que representa el ruido que se reproduce cuando el motociclistas cae de la rampa circular. Posteriormente se crean las variables que definen el ancho y alto del escenario de simulacion y se crea tambien la variable para relacionar pixeles con metros llamada diezU, con la cual se hace la equivalencia de que 10 metros corresponden a 150 pixeles.

```
function load () {

    var audio = new Audio('imagenes/caida.wav')

    var resolucion_ancho=1366
    var resolucion_alto=768
    var porcentaje_ancho_pantalla=0.8
    var porcentaje_alto_pantalla=0.869
    var ancho = resolucion_ancho* porcentaje_ancho_pantalla
    var alto=resolucion_alto*porcentaje_alto_pantalla


    var diezU = ancho/7.285 //10 metros equivalen a 150 pixeles
```

Figura 43. Definición de las dimensiones del escenario.

Tambien se agregan las variables que van a contener los valores a graficar, y dado que la simulación muestra ocho graficas de movimiento se crean ocho variables para ello, como se muestra en la figura 44.

```
//-----------------------------variables para graficar---------------------------------
    var datos_v1_vs_t = [
    {values: []},
    {values: []}
    ];

    var datos_a1_vs_t = [
    {values: []},
    {values: []}
    ];

    var datos_v_vs_teta = [
    {values: []}
    ];

    var datos_N_vs_teta = [
    {values: []}
    ];

    var datos_at_vs_teta = [
    {values: []}
    ];

    var datos_ar_vs_teta = [
    {values: []}
    ];

    var datos_a_vs_teta = [
    {values: []}
     ];
//-------------------------------------------------------------------------------
```

Figura 44. Creacion de variables para graficar.

De manera similar se sigue la estructura de phaser definida inicialmente en prácticas anteriores, por lo tanto se muestran a continuacion las funciones mas importantes que se han incluido dentro del desarrollo de la presente práctica:

3.1. Funcion init ()

Dentro de esta funcion se definen las variables mas importantes requeridas para crear el escenario, así como también para realizar los respectivos movimientos por los trayectos lineal y circular respectivamente, como es el caso de la distancia de inicio de la moto tanto en x como en y, el ancho del circulo y el punto final en dirección y de la plataforma lineal. Sin embargo hay otras variables que son creadas en las respectivas funciones ya que dependen de ciertos parámetros que solo se encuentran en estas (ver figura 45).

```
    init: function () {

    centro = {
        x: game.width / 2,
        y: game.height / 2
    }

    distanciaInicioMoto_x= diezU/15 //distancia desde donde se coloca la llanta trasera de la moto en pixeles
    distanciaInicioMoto_y= game.height-diezU/1.8
    anchorCirulo=diezU/18
    puntoFinPlataformaLineal_y=game.height-diezU/1.95
},
```

Figura 45. Funcion init.

3.2. Funcion Preload ()

En la funcion preload se realiza el procedimiento para cargar las imágenes complementarias del escenario (árbol, nubes, montaña y el sprite del motociclista, como se muestra en la figura 46

```
preload: function () {

    game.load.spritesheet('arbol', 'imagenes/arbol3.png', 769, 720)
    game.load.spritesheet('moto', 'imagenes/motosprite2.png', 284, 246)
    game.load.spritesheet('nubecita', 'imagenes/nubecita1.png')
    game.load.spritesheet('nubecita1', 'imagenes/nubecita3.png')
    game.load.spritesheet('montaña', 'imagenes/mountain3.png')

    },
```

Figura 46. Funcion preload.

3.3. Funcion create ()

En esta funcion se llevan a cabo los mismos procedimientos realizados en la practica de movimiento en 2 dimensiones, los cuales consisten en validar los

cambios en los datos de entrada y la creación de las funciones asociadas a los diferentes botones de la interfaz. Para más detalles se recomienda revisar la funcion **Create ()** de la practica anterior.

3.4. Funciones asociadas al botón Cargar datos

Dentro de esta función se realizan los diferentes procesos que involucra la creación del escenario y la acción de limpiar las gráficas de simulaciones pasadas, para ello se emplean las siguientes funciones:

a) Función cargar_datos

En la figura 47 se muestra el proceso de creación del fondo del escenario (cielo de color naranja y el suelo), para ello se emplea el método **createLinearGradient**, mencionado al inicio del presente anexo.

```
ctx1.beginPath();
var grd = ctx1.createLinearGradient(0,0,0,game.height);
/*grd.addColorStop(0.4, '#00ABEB');*/
grd.addColorStop(0.6, '#F7F08D');
grd.addColorStop(0.0, '#F5A53C');
grd.addColorStop(0.75, '#F7F08D');
grd.addColorStop(0.75, '#77A12D');
grd.addColorStop(1, '#D2F695');
ctx1.fillStyle= grd;
ctx1.fillRect(0,0, game.width, game.height);

//titulo
bmd1.text('MOVIMIENTO CIRCULAR', centro.x-diezU*0.8,diezU/4, diezU/6+'px Cursive', '#F7EA5A')
```

Figura 47. Creacion de fondo.

Posteriormente se agregan las imágenes complementarias del escenario, es decir, las figuras de las montañas, las nubes y el arbol situado al lado de la rampa circular (figura 48), a través de las respectivas lineas de código para cada una, donde se especifica la posición inicial en x y y donde son ubicadas, la escala de la imagen respecto al ancho del escenario y el punto de referencia de la imagen usada para su ubicación.

```
montaña1 = game.add.sprite(0, game.height-diezU, 'montaña')
montaña1.scale.setTo(game.width / 3700);
montaña1.anchor.set(0,1);

arbol = game.add.sprite(game.width-diezU*1.7, game.height-diezU/1.5, 'arbol')
arbol.scale.setTo(game.width / 3200);
arbol.anchor.set(0,1);

nubecita1 = game.add.sprite(diezU, diezU/1.6, 'nubecita1')
nubecita1.scale.setTo(game.width / 700);
nubecita1.anchor.set(0,1);
```

Figura 48. Imágenes que se agregan al fondo.

De igual manera se agrega el sprite del motociclista y se definen las animaciones que representan los movimientos de este a lo largo de los dos segmentos (rectilíneo y circular). La animación “uno”, correspondiente a los cuadros de imágenes 0, 1 y 2 ilustran el giro de las ruedas durante el desplazamiento, mientras que la animación “dos” se emplea para simular el proceso de caída del piloto desde un punto determinado de la plataforma circular, cuando su velocidad no es suficiente para dar un giro completo (ver figura 49).

```
//moto
moto = game.add.sprite(distanciaInicioMoto_x, distanciaInicioMoto_y, 'moto')
moto.scale.setTo(game.width / 2900);
moto.anchor.set(0,1);
moto.animations.add('uno', [0, 1, 2], 6, true)
moto.animations.add('dos', [3, 4], 6, true)
```

Figura 49. Definicion de animaciones para el motociclista.

A continuación se procede a la creación de la plataforma lineal y circular. Para el caso del tramo circular se hace uso del método de Canvas denominado **arc(x, y, ángulo inicio, ángulo final, dirección)**, donde se especifica la posición del centro de la circunferencia que se va a crear, el radio y los ángulos de inicio y final del arco implementado (ver figura 50).

```
ctx.beginPath();
ctx.lineWidth=18
var grd1 = ctx.createLinearGradient(longitud,game.height-diezU/2.2-radio*2,longitud,game.height-diezU/3);
grd1.addColorStop(0.5, '#7F6F51');
grd1.addColorStop(0.6, '#99855F');
ctx.fillStyle=grd1;
ctx.arc(longitud+moto.width+distanciaInicioMoto_x,distanciaInicioMoto_y-radio,radio+anchorCirulo,0,Math.PI*2,true);
ctx.strokeStyle=grd1
ctx.stroke();
```

Figura 50. Creacion de la plataforma circular.

Como paso final dentro de esta función se limpian los inputs donde se despliegan los datos de salida de simulaciones previas (ver figura 51).

```
document.getElementById('va').value=""
document.getElementById('va2').value=""
document.getElementById('teta').value=""
```

Figura 51. Limpiar input de salida.

b) Función limpiar_graficas ()

De manera similar que en la práctica de movimiento bidimensional esta función es empleada para borrar las curvas de simulaciones anteriores y dejar los ejes coordenados vacios para las gráficas correspondientes a nuevos ejercicios de

simulación. Esta función se implementa en las demás prácticas, por lo tanto sigue una estructura similar (ver figura 52).

```
limpiar_tablas: function(){
    datos_v1_vs_t[1].values = [];
    grafica_v1_vs_t.option('range', [])
    grafica_v1_vs_t.option('domain', [])
    grafica_v1_vs_t.update(datos_v1_vs_t)

    datos_a1_vs_t[1].values = [];
    grafica_a1_vs_t.option('range', [])
    grafica_a1_vs_t.option('domain', [])
    grafica_a1_vs_t.update(datos_a1_vs_t)

    datos_v_vs_teta[0].values = [];
    grafica_v_vs_teta.option('range', [])
    grafica_v_vs_teta.option('domain', [])
    grafica_v_vs_teta.update(datos_v_vs_teta)

    datos_N_vs_teta[0].values = [];
    grafica_N_vs_teta.option('range', [])
    grafica_N_vs_teta.option('domain', [])
    grafica_N_vs_teta.update(datos_N_vs_teta)

    datos_at_vs_teta[0].values = [];
    grafica_at_vs_teta.option('range', [])
    grafica_at_vs_teta.option('domain', [])
    grafica_at_vs_teta.update(datos_at_vs_teta)

    datos_ar_vs_teta[0].values = [];
    grafica_ar_vs_teta.option('range', [])
    grafica_ar_vs_teta.option('domain', [])
    grafica_ar_vs_teta.update(datos_ar_vs_teta)

    datos_a_vs_teta[0].values = [];
    grafica_a_vs_teta.option('range', [])
    grafica_a_vs_teta.option('domain', [])
    grafica_a_vs_teta.update(datos_a_vs_teta)

    },
```

Figura 52. Función limpiar graficas.

3.5. Función asociada al boton Simulación

La función asociada a este botón se denomina **mover_moto ()** y en ella se realizan los procesos necesarios para definir la trayectoria que sigue el motociclista en los dos tramos que componen el sistema total. Por lo tanto, inicialmente se activa la animación denominada “uno”, la cual representa el movimiento de las ruedas de la moto, indicando que el piloto se encuentra realizando un desplazamiento, como se observa en la primera línea de código de la figura 53. También, dentro de esta estructura se llevan a cabo los cálculos numéricos correspondientes a la aceleración, velocidad y tiempo para el tramo lineal.

```
moto.animations.play('uno')

if(miu>0){
acele=-miu*9.8
ta=(-vo+Math.sqrt(Math.pow(vo,2)+2*acele*l))/acele

velocidad=vo+acele*ta
}
else{

    ta=l/vo
    velocidad=vo
    acele=0
}
```

Figura 53. Ecuaciones para el tramo lineal.

Posteriormente se lleva a cabo el proceso necesario para calcular el ángulo que recorre el piloto dentro del tramo circular. La lógica empleada para ello, mostrada en la figura 54, inicia asignando el valor de cero a la variable *teta1* y definiendo las variables f y e como aquellas que van a contener a los dos términos de la inecuación que representa la condición para determinar el ángulo que recorre el piloto dentro de este segmento (ecuación 73 del anexo A).

Por medio del ciclo *while* se indica al sistema que mientras se cumpla la condición especificada, la variable *teta1* se vaya incrementando 0.1 radianes, de modo que cada vez que cambie, se evalua la variable e. En el momento en que la condición ya no se cumpla, la variable $teta$ toma el valor de $teta1$ y así queda definido el valor del angulo alcanzado. Si se cumple que e siempre es menor que f significa que el piloto logra dar un giro completo y el ángulo recorrido es igual a 360°.

```
teta1=0
e=(Math.sin(teta1-Math.PI/2)+miu*Math.cos(teta1-Math.PI/2))*teta1
f=Math.pow(velocidad,2)/(2*9.8*r)

while(e<=f && teta1<2*Math.PI) {
        teta1=teta1+0.1*Math.PI/180
        e=(Math.sin(teta1-Math.PI/2)+miu*Math.cos(teta1-Math.PI/2))*teta1
        }

    teta=(Math.round((teta1-0.1*Math.PI/180)*10000)/10000)

    tetap=teta1
```

Figura 54. Proceso para encontrar el ángulo recorrido por el piloto en el tramo circular.

Posteriormente, dependiendo de si el piloto alcanza o no a recorrer toda la plataforma circular se determinan los puntos de las trayectorias por donde se desplaza el motociclista. En el codigo de la figura 55 se muestra como ejemplo el

caso en que el motociclista no alcanza a recorrer toda la circunferencia, de tal manera que inicialmente se encuentran los puntos del tramo recto haciendo uso del método de interpolación lineal y a continacion para definir el movimiento por el tramo circular se definen las variables $px5$ y $py5$, las cuales vienen dadas por las ecuaciones de posición del movimiento circular ($x = rcos\theta$ y $y = rsin\theta$) adaptadas respecto al radio desde donde se crea la plataforma circular, de tal manera que por medio de un ciclo for que va desde $-\pi/2$ hasta el ángulo recorrido encontrado se va guardando cada posicion en la variable *camino5* de tipo array.

```
else{

    var x = 1 / pasos

    for (var j = 0; j <= 1; j += x) {
     var px = this.math.linearInterpolation([distanciaInicioMoto_x,distanciaInicioMoto_x+longitud], j);

     var py = this.math.linearInterpolation([distanciaInicioMoto_y,distanciaInicioMoto_y], j);

      camino.push( { x: px, y: py })
    }


    teta2=-Math.PI/2+teta-moto.width/radio

    for (var teta5=-Math.PI/2 ; teta5<=teta2 ; teta5 += Math.PI/180) {
     var px5 = radio*Math.cos(teta5)+longitud+moto.width+distanciaInicioMoto_x

     var py5 = -radio*Math.sin(teta5)+distanciaInicioMoto_y-radio

      camino5.push( { x: px5, y: py5 })
    }

    enMovimiento = true
    paso=0;
    paso5=0;
```

Figura 55. Definición de los puntos del recorrido del piloto dentro del tramo lineal y circular.

Cuando el motociclista no consigue llevar a cabo un giro completo se tienen dos opciones posibles: que caiga de la plataforma o se devuelva a través de ella. Como ejemplo se ilustra el caso en que el piloto recorre un ángulo menor o igual a 105° y por lo tanto se devuelve (figura 56). Para encontrar la trayectoria realizada se emplea un ciclo for que comienza con el valor del angulo recorrido y decrece en un radián hasta llegar al punto de partida de la plataforma circular. Dentro de este ciclo se evalua la posicion en x y y del piloto a través de las variables $px6$ y $py6$ respectivamente, asi mismo cada punto se guarda dentro de la variable $amino6$.

```
for (var teta6=teta2 ; teta6>-Math.PI/2  ; teta6 -= Math.PI/180) {
 var px6 = radio*Math.cos(teta6)+longitud+moto.width+distanciaInicioMoto_x

 var py6 =  -radio*Math.sin(teta6)+distanciaInicioMoto_y-radio

  camino6.push( { x: px6, y: py6 })
}
```

Figura 56. Proceso para encontrar los puntos de retroceso del motociclista.

3.5. Funciones asociadas al boton Resultados

Finalmente se lleva a cabo a cabo el despliegue de resultados gráficos y numéricos a través de la función **resultados ()**. Dentro de esta función se grafican las curvas correspondientes a velocidad (v) vs teta (θ), fuerza normal (N) vs teta (θ), aceleración tangencial (a_t) vs teta (θ), aceleración radial (a_r) vs teta (θ) y aceleración total (a) vs teta (θ) para el movimiento dentro del tramo circular, como se observa en la figura 57, donde a cada variable creada se le asigna la función respectiva para ser graficada tanto en x como en y.

```
datos_v_vs_teta[0].values.push({x: tet*180/Math.PI, y: Math.sqrt(((Math.pow(velocidad,2))-(2*tet*r*9.8*Math.sin(-Math.PI/2
datos_N_vs_teta[0].values.push({x: tet*180/Math.PI, y: m*(Math.pow(vel,2)/r)+9.8*Math.cos(tet-Math.PI/2)});
datos_at_vs_teta[0].values.push({x:tet*180/Math.PI, y: -9.8*Math.sin(-Math.PI/2+tet)-(miu*Math.pow(vel,2))/r-miu*9.8*Math.
datos_ar_vs_teta[0].values.push({x:tet*180/Math.PI, y: (N1/m)-9.8*Math.cos(-Math.PI/2+tet)});
datos_a_vs_teta[0].values.push({x:tet*180/Math.PI, y: Math.sqrt(Math.pow(at1,2)+Math.pow(ar1,2))});
```

Figura 57. Asignacion de los valores a graficar (tramo circular).

De manera similar en la figura 58 se muestra la codificación realizada para graficar las curvas respectivas de velocidad (v) vs tiempo (t) y aceleración total (a) vs tiempo (t) correspondientes al tramo lineal.

```
for(taa=0; taa<=ta; taa+=0.1) {

        datos_v1_vs_t[1].values.push({x: taa, y: acele*taa+vo});
        datos_a1_vs_t[1].values.push({x: taa, y: acele});

}
```

Figura 58. Asignacion de los valores a graficar (tramo lineal).

4. CODIFICACIÓN DE LA PRÁCTICA DE LEYES DEL MOVIMIENTO

Para esta práctica se tienen inicialmente las variables que involucran el ancho y alto del escenario seguido por la variable *Unmetro* con la que se hace la relación pixeles con metros, llegando a que un metro equivale a 36 pixeles (ver figura 59).

```
function load () {

    var resolucion_ancho=1366
    var resolucion_alto=768
    var porcentaje_ancho_pantalla=0.72
    var porcentaje_alto_pantalla=0.86
    var ancho = resolucion_ancho* porcentaje_ancho_pantalla
    var alto=resolucion_alto*porcentaje_alto_pantalla

    var Unmetro = ancho/27.32  //un metro equivale a 36 pixeles
```

Figura 59. Definición del ancho y alto del escenario.

Del mismo modo a continuación se crean las variables donde se guardarán los valores a graficar, por lo que es necesario crear una por cada gráfica, con sus respectivos arreglos de objetos como se observa en la figura 60, donde se tienen cuatro variables y cada una contiene un arreglo de 3 objetos en los cuales se grafican los tres tramos del sistema: tramo A-A', tramo A'-B y tramo B-C

```
var datos_x_vs_t = [
{values: []},
{values: []},
{values: []}
];

var datos_y_vs_t = [
{values: []},
{values: []},
{values: []}
];

var datos_v_vs_t = [
{values: []},
{values: []},
{values: []}
];

var datos_a_vs_t = [
{values: []},
{values: []},
{values: []}
];
```

Figura 60. Variables empleadas para graficar.

De manera similar que para las anteriores prácticas se sigue la estructura de Phaser por lo que a continuación se realiza una breve descripción de las funciones necesarias para dicho desarrollo.

4.1. Función init ()

Dentro de esta función se definen las variables *centro*, *base_rampa_metros* y *base_rampa_pixeles*. Las dos últimas variables mencionadas son utilizadas para realizar los calculos numéricos, para la creación del escenario y para encontrar la trayectoria lineal sobre la rampa. Posteriormente se defienieron también las variables *distancia_comienzo_rampa_x*, *punto_y_reposa_rampa* y una última que se ha denominado *punto_en_y_posicioninicial_resorte* (ver figura 61).

```
init: function () {

centro = {
    x: game.width / 2,
    y: game.height / 2
}
base_rampa_metros=7//valor base de la rampa en metros empleada para calculos numericos
base_rampa_pixeles=Unmetro*base_rampa_metros //valor de la base de la rampa en pixeles empleada para graficar
distancia_comienzo_rampa_x= Unmetro*2 //Diancia en x desde donde comienza la rampa en pixeles
punto_en_y_reposa_rampa=game.height-Unmetro*3.9 // punto en 'y' sobre el cual reposa la rampa
punto_en_y_posicioninicial_resorte=game.height-Unmetro*4.1 //punto inicial en y del resorte

},
```

Figura 61. Funcion init ().

4.2. Función Preload ()

En esta función se cargan las imágenes complementarias del escenario de simulación, como por ejemplo la mesa donde se sitúa la rampa ascendente, el resorte, el bloque o masa que será puesta en movimiento, la diana o punto que indica la ubicación exacta del blanco, la puerta y la ventana del laboratorio, por lo que a cada una se le asigna un nombre y se especifica el directorio donde se encuentran alojadas. Para los sprites correspondientes al resorte y al bloque se debe indicar el tamaño de cada cuadro de imagen que lo conforma (ver figura 62).

```
    preload: function () {

    game.load.spritesheet('mesa', 'imagenes/mesalab1.png')
    game.load.spritesheet('resorte', 'imagenes/resortesprite6.png', 200, 79)
    game.load.spritesheet('bloque', 'imagenes/bloquesprite.png', 200, 40)
    game.load.spritesheet('diana', 'imagenes/diana3.png')
    game.load.spritesheet('pared', 'imagenes/labfuerzas.png')
    game.load.spritesheet('puerta', 'imagenes/puerta1.png')
    game.load.spritesheet('ventana', 'imagenes/ventana1.png')

},
```

Figura 62. Función Preload ().

4.3. Función Create ()

Debido a que en la función **Create ()** se realizan los mismos procesos que en la mayoría de las prácticas, esta vez no se hará referencia a ella. Por lo tanto, para la solución de cualquier duda remitirse a la práctica de movimiento en 2 dimensiones.

4.4. Funciones asociadas al botón Cargar datos

a) Función cargar_datos ()

De modo similar que en las anteriores prácticas, se procede a crear el fondo del escenario, el cual está conformado por una pared y el suelo como parte de una sala de laboratorio, para esto se crea un gradiente con cuatro colores mediante el método **createLinearGradient()**, donde los dos primeros colores son para la pared y los dos restantes para el suelo (figura 63).

```
ctx1.beginPath();
var grd = ctx1.createLinearGradient(0,0,0,game.height);
/*grd.addColorStop(0.4, '#00ABEB');63A49A*/

grd.addColorStop(0.1, '#26A69A');
grd.addColorStop(0.6, '#E0F2F1');
grd.addColorStop(0.6, '#DDDF91');
grd.addColorStop(1, '#B9BC5F');

ctx1.fillStyle= grd;
ctx1.fillRect(0,0, game.width, game.height);
```

Figura 63. Creación del fondo del escenario.

Posteriormente se agregan los demás componentes al escenario, como la puerta, la ventana, la mesa donde se sitúa la rampa y la diana que indica el sitio de ubicación del blanco a donde debe ir a parar el bloque, para luego seguir con la creación de la rampa, la cual depende del ángulo de inclinación y por lo tanto debe crearse con los métodos de Canvas. El diseño de la rampa inicia con la creación del triángulo inferior, para lo cual se emplea el primer bloque de código de la figura 64, en él se especifican los puntos de los vértices del triángulo. Posteriormente se crea la parte superior de la rampa de manera similar a como se implementa el triángulo inferior, es decir, indicando punto por punto por el trazado de las líneas de dibujo. Para esto es necesario emplear la semejanza de triángulos con el fin de encontrar dichos puntos, los cuales que dependen del ángulo de inclinación de esta rampa.

```
ctx.beginPath()
var grd1 = ctx.createLinearGradient(Unmetro*3,game.height-Unmetro*3-Unmetro*5.5*Math.tan(teta),Unmetro*3+Unmetro*5.5
grd1.addColorStop(0, '#8F8354');
grd1.addColorStop(1, '#9F9265');
ctx.moveTo(distancia_comienzo_rampa_x, punto_en_y_reposa_rampa)
ctx.lineTo(distancia_comienzo_rampa_x+base_rampa_pixeles, punto_en_y_reposa_rampa )
ctx.lineTo(distancia_comienzo_rampa_x+base_rampa_pixeles, punto_en_y_reposa_rampa-base_rampa_pixeles*Math.tan(teta))
ctx.closePath()
ctx.fillStyle=grd1
ctx.strokeStyle="#F6ED6C"
ctx.stroke()
ctx.fill()

ctx.beginPath() //parte superior de la rampa sobre donde reposa el resorte, es necesario emplear semejanza de triang
var grd2 = ctx.createLinearGradient(Unmetro*3,game.height-Unmetro*3-Unmetro/1.5-Unmetro*5.5*Math.tan(teta),Unmetro*3-
grd2.addColorStop(0, '#DDD0A2');
grd2.addColorStop(0.9, '#9F9265');
ctx.moveTo(distancia_comienzo_rampa_x, punto_en_y_reposa_rampa)
ctx.lineTo(Unmetro*1.8, punto_en_y_reposa_rampa-Unmetro/1.5)
ctx.lineTo(distancia_comienzo_rampa_x+base_rampa_pixeles*0.88, punto_en_y_reposa_rampa-Unmetro/1.5-base_rampa_pixele
ctx.lineTo(distancia_comienzo_rampa_x+base_rampa_pixeles, punto_en_y_reposa_rampa-base_rampa_pixeles*Math.tan(teta))
ctx.fillStyle=grd2
ctx.closePath()
ctx.stroke()
ctx.fill()
```

Figura 64. Creación de la rampa ascendente.

Luego de esto se agrega el sprite del resorte en la posición indicada por medio de las variables *distancia_comienzo_rampa_x* y *punto_en_y_posicioninicial_resorte*, definidas en la función **Init ()**. A continuación el resorte debe ser rotado un ángulo correspondiente a la inclinación de la rampa mediante la propiedad rotation, con signo negativo ya que se rota en un sentido contrario a las manecillas del reloj. Posteriormente se definen las siguientes animaciones: “uno”, correspondiente al cuadro de imagen número diez, el cual representa el resorte totalmente comprimido y es empleado cuando se abre la aplicación inicialmente, antes de simular. “dos”, que abarca las imágenes cero, uno, dos, tres, cuatro, cinco, seis, siete, ocho y nueve, utilizadas para simular el proceso de descomprensión del resorte y liberación del bloque. “tres”, que comprende el cuadro de imagen número once, con el que se representa el resorte solo (sin la masa) y es utilizado cuando el bloque realiza su movimiento independiente y dicho resorte queda estático en la posición inicial. Este proceso se observa en el código de la figura 65.

```
resorte = game.add.sprite(distancia_comienzo_rampa_x, punto_en_y_posicioninicial_resorte, 'resorte')
resorte.scale.setTo(game.width / 2200);
resorte.anchor.set(0,1);
resorte.rotation=-teta
resorte.animations.add('uno', [10], 1, true)
resorte.animations.add('dos', [9,8,7,6,5,4,3,2,1,0], 8, true)
resorte.animations.add('tres', [11], 1, true)
resorte.animations.play('uno')
```

Figura 65. Definición de las animaciones.

Como paso final dentro de esta función se realiza el proceso para limpiar las tablas haciendo uso del método **innerHTML**, el cual permite modificar el texto de

los elementos creados, que en este caso son las cajas para conformar las tablas (ver figura 66).

```
document.getElementById("j1").innerHTML=""
document.getElementById("j2").innerHTML=""
document.getElementById("j3").innerHTML=""
document.getElementById("j4").innerHTML=""
document.getElementById("j5").innerHTML=""
document.getElementById("j6").innerHTML=""
document.getElementById("j7").innerHTML=""
document.getElementById("j8").innerHTML=""
document.getElementById("j9").innerHTML=""
document.getElementById("j10").innerHTML=""
```

Figura 66. Proceso para limpiar tablas.

b) Función limpiar_graficas ()

Es otra de las funciones asociadas al botón cargar datos, con la cual se accede a cada una de las variables creadas con el fin de realizar las gráficas, asignándoles una cadena de datos vacía (ver figura 67).

```
    limpiar_graficas: function(){

    datos_x_vs_t[0].values = [];
    datos_x_vs_t[1].values = [];
    datos_x_vs_t[2].values = [];
    grafica_x_vs_t.option('range', [])
    grafica_x_vs_t.option('domain', [])
    grafica_x_vs_t.update(datos_x_vs_t)

    datos_y_vs_t[0].values = [];
    datos_y_vs_t[1].values = [];
    datos_y_vs_t[2].values = [];
    grafica_y_vs_t.option('range', [])
    grafica_y_vs_t.option('domain', [])
    grafica_y_vs_t.update(datos_y_vs_t)

    datos_v_vs_t[0].values = [];
    datos_v_vs_t[1].values = [];
    datos_v_vs_t[2].values = [];
    grafica_v_vs_t.option('range', [])
    grafica_v_vs_t.option('domain', [])
    grafica_v_vs_t.update(datos_v_vs_t)

    datos_a_vs_t[0].values = [];
    datos_a_vs_t[1].values = [];
    datos_a_vs_t[2].values = [];
    grafica_a_vs_t.option('range', [])
    grafica_a_vs_t.option('domain', [])
    grafica_a_vs_t.update(datos_v_vs_t)

},
```

Figura 67. Función limpiar_graficas ().

4.5. Funciones asociadas al boton Simulación

Para este boton se tiene asociada la funcion simulacion en la cual se encuentran los puntos que seran asignados a los sprites para realizar los respectivos movimientos por la trayectoria establecida.

a) Función simulación ()

Aquí se crean las variables donde se guardan los puntos para cada animación. Por ejemplo la variable *this.camino1* es utilizada para realizar la primera animación, correspondiente a la secuencia de descomprensión del resorte un instante antes de separarse el bloque. Por su parte en la variable *this.camino2* se guardan los puntos de la trayectoria que realiza el bloque sin el resorte hasta llegar al final de la rampa. En la variable *this.camino3* se guardan los puntos del movimiento parabólico y finalmente *this.camino4* es utilizada para poder realizar la animación de cuando el bloque ha caído justo en el blanco (ver figura 68).

```
this.camino1 = []
this.camino2 = []
this.camino3 = []
this.camino4 = []
```

Figura 68. Variables donde se guardan los trayectos a recorrer.

Una vez que han sido definidas estas variables se implementan las respectivas ecuaciones de movimiento para los diferentes tramos del sistema, las cuales son empleadas para desplegar los resultados numéricos en la tabla de la interfaz, para graficar las diferentes curvas y para determinar si el bloque entra o no en el blanco. Este proceso se muestra en la figura 69.

```
//formulas
m=0.4
l=base_rampa_metros/Math.cos(teta)
k=4000
a_res= (k*x)/m-9.8*(Math.sin(teta)+miu*Math.cos(teta))  //aceleracion resorte
vfr=Math.sqrt(2*a_res*x)
a_ramp= -9.8*(Math.sin(teta)+miu*Math.cos(teta))

pru=2*a_ramp*(l-x)+Math.pow(vfr,2)
if(pru<=0){
   alert("La longitud de comprension debe ser mayor al valor ingresado")
   return
}

vf=Math.sqrt(2*a_ramp*(l-x)+Math.pow(vfr,2))
ta_r=vfr/a_res
tr_b=Math.abs((vf-vfr)/a_ramp)
ta_b=ta_r+tr_b

hrampa=base_rampa_metros*Math.tan(teta)
z=(9.8*hrampa)/Math.pow(vf,2)
td=(vf/9.8)*(Math.sin(teta)+Math.sqrt(Math.pow(Math.sin(teta),2)+(2*z)))
vox=vf*Math.cos(teta)
voy=vf*Math.sin(teta)
R=vox*td
hmax=(Math.pow(vf,2)*Math.pow(Math.sin(teta),2)/(2*9.8))+hrampa
vx=vox
vy=voy-9.8*td
v=Math.sqrt(Math.pow(vox,2)+Math.pow(voy,2))
vd=Math.sqrt(Math.pow(vx,2)+Math.pow(vy,2))
```

Figura 69. Definicion de las formulas empleadas.

A continuación se definen los puntos para la simulación del movimiento del bloque en los diferentes tramos. Por lo tanto, para llevar a cabo la primera animación, correspondiente a la descomprensión del resorte, se guardan los puntos que va generando el método de la interpolación lineal a través de ciclo *for*. Como para este caso se requiere que el sprite no se mueva, los puntos generados corresponden a la posición inicial del resorte, de esta manera, cada punto es asignado a una misma posición y se activa la animación “dos”, como se observa en la figura 70.

```
y=1/44



for (var j = 0; j <= 1; j += y) {

    var px1 = this.math.linearInterpolation([distancia_comienzo_rampa_x, distancia_comienzo_rampa_x], j);
    var py1 = this.math.linearInterpolation([punto_en_y_posicioninicial_resorte, punto_en_y_posicioninicial_resorte], j);

  this.camino1.push( { x: px1, y: py1 })
}
```

Figura 70. Definición del primer tramo de simulación.

En la figura 71 se muestra el proceso llevado a cabo para realizar la primera animación, correspondiente a la descomprensión del resorte en la función **Update ()**. Para ello se crean las variables $px1$ y $py1$, a las que se les asigna la variable *this.camino1*, la cual contiene los puntos por donde se mueve el resorte. Estas variables son asignadas a la posicion en x y y del resorte, y dependiendo del número de puntos encontrados se tiene un condicional mediante el cual dichas variables ($px1$ y $py1$) van tomando cada uno de estos puntos y los va asignado a la posición del sprite. Una vez que se hayan asignado todos los puntos establecidos a la posición del resorte, y cuando la secuencia indica que el cuerpo está a punto de separarse, se hace necesario agregar otro sprite con el bloque solo, el cual recorrerá el trayecto restante. Luego, a partir de esta imagen se simula el proceso de entrada del bloque en el blanco.

```
update: function(){

    if (enMovimiento) {


        var px1 = this.camino1[this.paso1].x
        var py1 = this.camino1[this.paso1].y

        resorte.x = px1
        resorte.y = py1

if (this.paso1 < 43) {
        this.paso1++
        resorte.animations.play('dos')
        if(this.paso1==42){
        /*Para colocar el bloque en la posicion inicial se tomo como base el triangulo que forma
        el resorte suponiendo que la hipotenusa corresponde al ancho del resorte*/

        bloque = game.add.sprite(distancia_comienzo_rampa_x+Math.cos(teta)*resorte.width*0.78,
        punto_en_y_posicioninicial_resorte-Math.sin(teta)*resorte.width*0.81, 'bloque')
        bloque.scale.setTo(game.width / 1100);
        bloque.anchor.set(0,1);
        bloque.rotation=-teta
        bloque.animations.add('z', [0,1,2,4], 5, true)

        //document.getElementById("m").value=bloque.width
    }
        }
```

Figura 71. Proceso en la función Update () donde se agrega el bloque sin el resorte.

Dentro del proceso anterior también se determinan los puntos de las dos trayectorias restantes, es decir, aquella en que el bloque se separa del resorte hasta llegar al final de la rampa y desde este punto hasta llegar al suelo, siguiendo un movimiento parabólico. Como ejemplo se ilustra la forma de encontrar los puntos para la trayectoria parabólica del bloque con base en las ecuaciones de posición en función del tiempo para este movimiento. Para ello se emplea un ciclo *for* mediante el cual se evalúan tales ecuaciones desde un instante igual a cero hasta el tiempo que se demora el bloque en realizar todo el desplazamiento parabólico. Los puntos encontrados se guardan en la variable *this.camino3* y se asignan a la posición del bloque dentro de la función **Update ()** (ver figura 72).

```
p=td/160

for (var q = 0; q <= td; q+=p ) {

var ox=  (vox*q)*Unmetro+(distancia_comienzo_rampa_x+base_rampa_pixeles)
var oy=  punto_en_y_reposa_rampa -(((-9.8*Math.pow(q,2))/2)+voy*q+hrampa)*Unmetro
this.camino3.push({ x: ox, y: oy })

    }
```

Figura 72. Puntos definidos dentro de la trayectoria circular.

4.6. Funciones asociadas al botón Gráficas

a) Función gráficas ()

De modo similar que en las anteriores prácticas, para realizar el proceso de graficar las funciones asociadas al movimiento del bloque se accede a las variables creadas al inicio y mediante un ciclo *for* se asignan las ecuaciones respectivas para dibujar las curvas dentro del plano cartesiano. En el código mostrado en la figura 73 se tienen tres ciclos *for*, uno para cada tramo del sistema, con el fin de graficar el movimiento del bloque dentro de los tres tramos que dividen el sistema: el primero equivale al tramo A-A', el segundo al tramo A'-B y el ultimo al tramo B-C. En los tres ciclos *for* se emplean los tiempos calculados para llevar a cabo las diferentes gráficas. Al eje x se le asigna los valores de tiempo y al eje y los respectivos valores de posición en x, posición en y, velocidad del bloque y aceleración total.

```
for (var tr = 0; tr <= ta_r; tr += 0.0001) {

    datos_x_vs_t[0].values.push({x: tr, y: 0.5*a_res*Math.pow(tr,2)*Math.cos(teta)});
    datos_y_vs_t[0].values.push({x: tr, y: 0.5*a_res*Math.pow(tr,2)*Math.sin(teta)});
    datos_v_vs_t[0].values.push({x: tr, y: a_res*tr});
    datos_a_vs_t[0].values.push({x: tr, y: a_res});


}

for (var tb = ta_r; tb <= (ta_r+tr_b); tb += 0.001) {

    datos_x_vs_t[1].values.push({x: tb, y: (x+vfr*(tb-ta_r)+(0.5*a_ramp*Math.pow((tb-ta_r),2)))*Math.cos(teta)});
    datos_y_vs_t[1].values.push({x: tb, y: (x+vfr*(tb-ta_r)+(0.5*a_ramp*Math.pow((tb-ta_r),2)))*Math.sin(teta)});
    datos_v_vs_t[1].values.push({x: tb, y: vfr+a_ramp*(tb-ta_r)});
    datos_a_vs_t[1].values.push({x: tb, y: a_ramp});
}


for (var tv = (ta_r+tr_b); tv <= (ta_r+tr_b+td); tv += 0.001) {

    datos_x_vs_t[2].values.push({x: tv, y: vox*(tv-ta_r-tr_b)+7});
    datos_y_vs_t[2].values.push({x: tv, y: voy*(tv-ta_r-tr_b)-0.5*9.8*Math.pow((tv-ta_r-tr_b),2)+hrampa });
    datos_v_vs_t[2].values.push({x: tv, y: Math.sqrt(Math.pow(vox,2)+Math.pow(voy-9.8*(tv-ta_r-tr_b),2))});
    datos_a_vs_t[2].values.push({x: tv, y: -9.8});
}
```

Figura 73. Asignación de valores a graficar a cada una de las variables creadas inicialmente.

5. CODIFICACIÓN DE LA PRÁCTICA DE TRABAJO Y ENERGÍA

El proceso de codificación dentro de la práctica de trabajo y energía se muestra a continuación. Posterior a la definición de las variables correspondientes para el dimensionamiento del escenario de simulación (ancho y alto) se define otra variable, denominada *Unmetro*, con la que se hace la relación pixeles a metros para todo el sistema. En este caso un metro equivale a 75 pixeles, valor que se toma por conveniencia dependiendo del diseño del escenario que se va a implementar (ver figura 74).

```
var resolucion_ancho=1366
var resolucion_alto=768
var porcentaje_ancho_pantalla=0.75
var porcentaje_alto_pantalla=0.85
var ancho = resolucion_ancho* porcentaje_ancho_pantalla
var alto=resolucion_alto*porcentaje_alto_pantalla

var Unmetro = ancho/13.66 //un metro equivale a 75 pixeles
```

Figura 74. Definición de las dimensiones del escenario.

También se crean las variables donde se guardan los valores a graficar. Dado que son tres las gráficas desplegadas dentro de la práctica, se tienen tres variables para almacenar tales valores: la variable *datos_x_vs_y* que se emplea para graficar la trayectoria seguida por el obrero, la variable *datos_x1_vs_y1* donde se guardan los valores para la gráfica correspondiente a energía vs desplazamiento y finalmente la variable *datos_w_vs_x*, en la que se grafica el trabajo que realiza el obrero en cada punto del trayecto, como se ilustra en la gráfica 75.

```
var datos_x_vs_y = [
{values: []},
{values: []},
{values: []},
{values: []},
{values: []}
];

var datos_x1_vs_y1 = [
{values: []},
{values: []},
{values: []},
{values: []},
{values: []}
];

var datos_w_vs_x = [
{values: []},
{values: []},
{values: []},
{values: []},
{values: []}
];
```

Figura 75. Variables creadas para guardar los valores a graficar.

En este punto se describen las funciones utilizadas en el desarrollo de la práctica de trabajo y energía:

5.1. Función Init ()

Como en los anteriores casos, dentro de esta función se definen las variables empleadas para la creación del escenario y en la definición de la trayectoria del obrero. Se crean las variables *base_rampas_metros* y *base_rampas*, las cuales corresponden a los valores respectivos en metros y pixeles de la base de las rampas por donde el bloque es empujado (camino A) o por donde el bloque se desliza (camino C). También se crea la variable *punto_final_x_caminoA_recto* que indica el punto en dirección x hasta donde llega el tramo recto de los caminos A y C. Similarmente se definen las variables de la posición inicial del estante y el obrero dentro del escenario, entre otras (ver figura 76).

```
init: function () {

centro = {
    x: game.width / 2,
    y: game.height / 2
}
base_rampas_metros=3 //base de las rampas en metros
base_rampas=base_rampas_metros*Unmetro //base rampas en pixeles
punto_final_x_caminoA_recto=centro.x+centro.x*0.05 //distancia en x hasta donde llega el tramo recto camino A
ancho_carreteras_caminoRecto=game.height*0.051 // altura del rectangulo con la que se hacen las carreteras rectas
punto_inicio_y_caminoA=game.height*0.5 //punto inicial en 'y' desde donde se comienza a trazar el camino A
punto_final_y_caminoA=game.height*0.55 ////punto en 'y' donde se termina el camino A
punto_inicio_y_caminoB=game.height*0.572 //punto inicial en 'y' desde donde se comienza a trazar el camino B
punto_y_texto_caminoA=game.height*0.545 //punto en y donse se coloca el texto del camino A
punto_y_texto_caminoB=game.height*0.618 //punto en y donse se coloca el texto del camino B
punto_inicial_x_estanteYsprite=Unmetro/1.2 //punto en 'x' donde se coloca el sprite y el estante
punto_inicial_y_estanteYsprite_caminoA=game.height*0.535 //punto en 'y' donde se coloca el sprite y el estante caminoA
punto_inicial_y_estanteYsprite_caminoB=game.height*0.61 //punto en 'y' donde se coloca el sprite y el estante caminoB
punto_inicial_y_estanteYsprite_caminoC=game.height*0.68 //punto en 'y' donde se coloca el sprite y el estante caminoC
punto_inicio_y_caminoC=game.height*0.672 //punto inicial en 'y' desde donde se comienza a trazar el camino C
radio_x_elipse=(punto_final_x_caminoA_recto)/2 //radio de la elipse en x, camino C
radio_y_elipse=game.height*0.02 //radio de la elipse en y, camino C
},
```

Figura 76. Función Init ().

5.2. Función Preload ()

En esta función se cargan las imágenes correspondientes a los tres sprites que representan la secuencia de movimiento del obrero y algunas imágenes adicionales al escenario, como por ejemplo los arbustos al lado del camino A, los tres camiones de destino de las cajas, los estantes que contienen a las cajas de igual tamaño y las nubes que adornan el fondo de dicho escenario, como se muestra en la figura 77.

```
preload: function () {

game.load.spritesheet('personaje', 'imagenes/spritecajaazul2.png', 90.7, 114)
game.load.spritesheet('personaje1', 'imagenes/spritecajagris.png', 90.7, 114)
game.load.spritesheet('personaje2', 'imagenes/spritecajaroja2.png', 90.7, 114)
game.load.spritesheet('arb', 'imagenes/arbusto.png')
game.load.spritesheet('carro1', 'imagenes/camion1.png')
game.load.spritesheet('carro2', 'imagenes/camion2.png')
game.load.spritesheet('carro3', 'imagenes/camion3.png')
game.load.spritesheet('estante1', 'imagenes/estante1.png')
game.load.spritesheet('estante2', 'imagenes/estante2.png')
game.load.spritesheet('estante3', 'imagenes/estante3.png')
game.load.spritesheet('nube2', 'imagenes/Nubecita2.png')

},
```

Figura 77. Función Preload ().

5.3. Función Create ()

Como se mencionó en la práctica de movimiento bidimensional, en esta función se verifican los cambios en los datos de entrada. Se requiere además la función **jQuery change ()** para verificar que los datos nuevos que sean ingresados sean cargados. También se crean las funciones asociadas a cada botón y se llaman las funciones encargadas de inicializar las gráficas y de crear el escenario.

5.4. Funciones asociadas al botón Cargar datos

a) Función escenario ()

En esta función se crea el fondo del escenario, se agregan las diferentes imágenes y los sprites con las secuencias de animación, se definen las animaciones y se limpian las tablas. De esta manera se comienza con la creación del fondo correspondiente al cielo azul y al suelo verde respectivamente, para lo que se emplea el método **createLinearGradient()** con cinco colores para su conformación: los dos primeros para el cielo y los restantes para el suelo (ver figura 78).

```
ctx.beginPath();
var grd = ctx.createLinearGradient(0,0,0,game.height);
grd.addColorStop(0.1, '#00BFA5');
grd.addColorStop(0.47, '#F6FBBB');
grd.addColorStop(0.47, '#A1CF58');
grd.addColorStop(0.6, '#7D954D');
grd.addColorStop(1, '#9B9C53');

ctx.fillStyle= grd;
ctx.fillRect(0,0, game.width, game.height);
```

Figura 78. Creación del fondo del escenario.

Posteriormente se crea el tramo recto del camino A y del camino B, como ejemplo se muestra el primer caso, donde a partir del método **fillRect(x,y,ancho,alto)** se genera un rectángulo que inicia en la posición (0, punto_inicio_y_caminoA), con un ancho y alto determinado por las variables que se muestran en la última línea de código de la figura 79.

```
ctx.beginPath();
ctx.fillStyle="#738072";
ctx.fillRect(0,punto_inicio_y_caminoA, punto_final_x_caminoA_recto, ancho_carreteras_caminoRecto);
```

Figura 79. Creación del tramo recto de los caminos A y B.

Los otros tramos se crean haciendo uso de los métodos de Canvas **moveTo()**, **lineTo, fill()**, **stroke()** y para escribir los textos se emplea el método **text** del objeto bitmapData. Para la mayoría de estos métodos es necesario especificar una posición en x y y a conveniencia según el diseño que se le quiera dar. Por su parte, para la creación de la elevación o meseta del camino C se hace uso del método **ellipse()**, con el cual se genera una elipse con centro en el punto especificado y un radio en x y y determinado por las variables creadas en la función **Init ()**, como se observa en la línea de código número cinco de la figura 80.

```
ctx.beginPath()//elipse
var grd4 = ctx.createLinearGradient(0,game.height*0.651,centro.x+centro.x*0.15,game.height*0.651);
grd4.addColorStop(0.4, '#B27642');
grd4.addColorStop(0.9, '#46412E');
ctx.ellipse((punto_final_x_caminoA_recto)/2, punto_inicio_y_caminoC,radio_x_elipse,radio_y_elipse,0,2*Math.PI, fals
ctx.fillStyle=grd4;
ctx.fill()
ctx.strokeStyle="#605C2A";
ctx.stroke();
bmd.text("Camino C",Unmetro/10,game.height*0.7, Unmetro/6+'px Calibri', '#F3DDB2')
```

Figura 80. Creación de la elipse para el camino C.

Una vez que han sido creadas todas las figuras con Canvas se procede a agregar las imágenes que complementan la estética del escenario, entre ellas los estantes (sin la caja seleccionada) en la etiqueta select. Para esto, dado que se tienen nueve casos, correspondientes a los tres caminos y a las tres cajas, se requieren nueve condicionales a partir de los cuales, dependiendo de la opción, son colocados los estantes en determinadas posiciones. Como ejemplo se muestran los tres casos para el camino A (ver figura 81), en los que el estante se ubica en la misma posición pero variando la imagen que se agrega, ya que para el primer caso se agrega la imagen con nombre estante1 y para los dos casos restantes la imagen con nombre estante2 y estante3 respectivamente.

```
if(opcion=="Camino A - Caja Roja"){
estante1 = game.add.sprite(punto_inicial_x_estanteYsprite, punto_inicial_y_estanteYsprite_caminoA, 'estante1')
estante1.scale.setTo(game.width / 1000);
estante1.anchor.set(0,1);}
else if(opcion=="Camino A - Caja Azul"){
estante2 = game.add.sprite(punto_inicial_x_estanteYsprite, punto_inicial_y_estanteYsprite_caminoA, 'estante2')
estante2.scale.setTo(game.width / 1000);
estante2.anchor.set(0,1);}
else if(opcion=="Camino A - Caja Negra"){
estante3 = game.add.sprite(punto_inicial_x_estanteYsprite, punto_inicial_y_estanteYsprite_caminoA, 'estante3')
estante3.scale.setTo(game.width / 1000);
estante3.anchor.set(0,1);}
```

Figura 81. Condicionales para ubicar los estantes según la opcion escogida.

De manera similar que en el caso anterior, se requieren nueve condicionales para agregar el sprite correspondiente al obrero, ya que para cada color de caja se tiene un sprite diferente debido a que en el punto inicial, cuando el obrero selecciona una determinada caja, estas se encuentran en una posición diferente. De esta manera, para cada sprite el orden de la secuencia de animación difiere, como se observa en el código de la figura 82, donde se muestran como ejemplo las opciones para el camino A, en las cuales el sprite del obrero para las tres cajas es ubicado en la misma posición pero con diferencia en la imagen utilizada y las secuencias de animación, ya que a pesar de que las animaciones han sido creadas con el mismo nombre, los cuadros de las imágenes que componen la secuencia son diferentes.

```
if(opcion=="Camino A - Caja Azul"){

personaje = game.add.sprite(punto_inicial_x_estanteYsprite, punto_inicial_y_estanteYsprite_caminoA, 'personaje')
personaje.scale.setTo(game.width / 1000);
personaje.anchor.set(0,1);
personaje.animations.add('dos', [5, 6, 7, 8, 9], 4, true)
personaje.animations.add('uno', [0, 1, 2, 3, 4], 1.7, true)
personaje.animations.add('tres', [11, 12, 13], 1.9, true)
personaje.animations.add('cuatro', [14, 15, 16, 17], 3, true)
personaje.animations.add('cinco', [18, 19, 20, 21, 22], 1.9, true)
personaje.animations.add('seis', [23,24], 5, true)

}
else if(opcion=="Camino A - Caja Negra"){

personaje = game.add.sprite(punto_inicial_x_estanteYsprite, punto_inicial_y_estanteYsprite_caminoA, 'personaje1')
personaje.scale.setTo(game.width / 1000);
personaje.anchor.set(0,1);
personaje.animations.add('dos', [6, 7, 8, 9, 10], 4, true)
personaje.animations.add('uno', [0, 1, 2, 3, 4, 5], 1.7, true)
personaje.animations.add('tres', [12, 13, 14], 1.9, true)
personaje.animations.add('cuatro', [15, 16, 17, 18], 3, true)
personaje.animations.add('cinco', [19, 20, 21, 22, 23], 1.9, true)
personaje.animations.add('seis', [24,25], 5, true)

}

else if(opcion=="Camino A - Caja Roja"){

personaje = game.add.sprite(punto_inicial_x_estanteYsprite, punto_inicial_y_estanteYsprite_caminoA, 'personaje2')
personaje.scale.setTo(game.width / 1000);
personaje.anchor.set(0,1);
personaje.animations.add('dos', [4, 5, 6, 7, 8], 4, true)
personaje.animations.add('uno', [0, 1, 2, 3], 1.3, true)
personaje.animations.add('tres', [10, 11, 12], 1.9, true)
personaje.animations.add('cuatro', [13, 14, 15, 16], 3, true)
personaje.animations.add('cinco', [17, 18, 19, 20, 21], 1.9, true)
personaje.animations.add('seis', [22,23], 5, true)

}
```

Figura 82. Condicionales para agregar el obrero en determinada posición.

Se definieron así seis animaciones correspondientes a las siguientes acciones: la acción de coger la caja del estante (‘uno’), la acción de caminar sobre el tramo recto (‘dos’), la acción de dejar la caja en el suelo (‘tres’), la acción de arrastrar la caja sobre la rampa inclinada del camino A (‘cuatro’), la acción de levantar la caja (‘cinco’) y la acción de dejar la caja respectiva en los determinados camiones (seis). Para el camino C se define una animación adicional llamada ‘a’, la cual corresponde al bloque solo, empleado para deslizarse por el tramo descendente de este camino. Finalmente, como en las anteriores prácticas, se realiza el proceso de limpieza de las tablas accediendo a través del método **innerHTML** a cada uno de los identificadores que la conforman (figura 83).

```
document.getElementById("i1").innerHTML=""
document.getElementById("i2").innerHTML=""
document.getElementById("i3").innerHTML=""
document.getElementById("i4").innerHTML=""
document.getElementById("i5").innerHTML=""
document.getElementById("j1").innerHTML=""
document.getElementById("j2").innerHTML=""
document.getElementById("j3").innerHTML=""
document.getElementById("j4").innerHTML=""
document.getElementById("j5").innerHTML=""
document.getElementById("r1").innerHTML=""
document.getElementById("r2").innerHTML=""
document.getElementById("r3").innerHTML=""
document.getElementById("r4").innerHTML=""
document.getElementById("r5").innerHTML=""
```

Figura 83. Proceso para limpiar las tablas de resultados.

b) Función limpiar_gráficas ()

Esta función se encarga de borrar las curvas desplegadas en simulaciones anteriores, dejando los ejes coordenados disponibles para una nueva simulación, para ello se accede a cada una de las variables creadas y se les asigna un array vacío, como se muestra en la figura 84.

```
    limpiar_graficas: function(){

    datos_x_vs_y[0].values = [];
    datos_x_vs_y[1].values = [];
    datos_x_vs_y[2].values = [];
    datos_x_vs_y[3].values = [];
    datos_x_vs_y[4].values = [];
    grafica_x_vs_y.option('range', [])
    grafica_x_vs_y.option('domain', [])
    grafica_x_vs_y.update(datos_x_vs_y)

    datos_w_vs_x[0].values = [];
    datos_w_vs_x[1].values = [];
    datos_w_vs_x[2].values = [];
    datos_w_vs_x[3].values = [];
    datos_w_vs_x[4].values = [];
    grafica_w_vs_x.option('range', [])
    grafica_w_vs_x.option('domain', [])
    grafica_w_vs_x.update(datos_w_vs_x)

    datos_x1_vs_y1[0].values = [];
    datos_x1_vs_y1[1].values = [];
    datos_x1_vs_y1[2].values = [];
    datos_x1_vs_y1[3].values = [];
    datos_x1_vs_y1[4].values = [];
    //datos_x1_vs_y1[5].values = [];
    grafica_x1_vs_y1.option('range', [])
    grafica_x1_vs_y1.option('domain', [])
    grafica_x1_vs_y1.update(datos_x1_vs_y1)


},
```

Figura 84. Proceso para limpiar las gráficas.

5.5. Funciones asociadas al botón Simulación

a) Función simulación ()

Dentro de esta función se comienza por definir las variables en las que se van a guardar los puntos por donde se mueve el obrero en los diferentes caminos, ya que dependiendo del camino se requiere un número mayor o menor de acciones para realizar los movimientos. Para el desplazamiento dentro del camino C se requieren completar siete acciones, por lo tanto se crean siete variables para ello, como se observa en la figura 85.

```
camino = []
camino1 = []
camino2 = []
camino3 = []
camino4 = []
camino5 = []
camino6 = []
```

Figura 85. Definición de las acciones dentro del camino C.

Posteriormente se requieren tres condicionales para los tres caminos, dentro de los cuales se encuentran los puntos para cada trayectoria. Por ejemplo para el caso del camino B se llevan a cabo dos acciones por parte del obrero que

consisten en levantar la caja escogida y caminar con ella a los largo del trayecto, por ello se tienen dos ciclos *for*, uno para cada proceso mencionado.

Por ejemplo para el primer caso, dado que se necesita que el obrero permanezca en el mismo punto mientras levanta la caja, dentro del método de la interpolación lineal se le asignan dos puntos iguales (en x y y) correspondientes a su posición inicial. En el segundo ciclo for, por medio de las variables $px1$ y $py1$ se aplica el método de la interpolación lineal, donde se asignan dos puntos correspondientes a la posición inicial y final del tramo, con los que se definen los puntos que se encuentran dentro de esta trayectoria lineal, justo antes de llegar al camión 2 (ver figura 86).

```
else if((opcion=="Camino B - Caja Roja")||(opcion=="Camino B - Caja Azul")||(opcion=="Camino B - Caja Negra")){


    y=1/160

       for (var j = 0; j <= 1; j += y) {
     var px = this.math.linearInterpolation([punto_inicial_x_estanteYsprite,punto_inicial_x_estanteYsprite], j);

     var py = this.math.linearInterpolation([punto_inicial_y_estanteYsprite_caminoB,punto_inicial_y_estanteYsprite_caminoB], j);

      camino.push( { x: px, y: py })
    }

    x=1/270

       for (var j = 0; j <= 1; j += x) {
     var px1 = this.math.linearInterpolation([punto_inicial_x_estanteYsprite,punto_final_x_caminoA_recto+base_rampas+distancia_camina_f

     var py1 = this.math.linearInterpolation([punto_inicial_y_estanteYsprite_caminoB,punto_inicial_y_estanteYsprite_caminoB], j);

      camino1.push( { x: px1, y: py1 })
    }

    document.getElementById("dato").innerHTML=""

}
```

Figura 86. Definición de la trayectoria para el camino B.

5.6. Funciones asociadas al botón Resultados

a) Función datos ()

En la función **datos ()** se llevan a cabo los cálculos numéricos asociados a cada camino recorrido por el obrero. Como ejemplo se muestra el caso para el camino A (ver figura 87), donde se inicia encontrando el trabajo realizado por el obrero al levantar la caja, el cual difiere dependiendo del color escogido, por lo que se requieren tres condicionales para evaluar esta selección. Posteriormente se encuentra la longitud del tramo ascendente y se halla el trabajo total realizado sobre

la caja en el punto A del sistema. También para este caso se define el ángulo y la fuerza que se aplica al bloque cuando es empujado a través del tramo

ascendente. De igual manera se calcula el trabajo para los demás puntos del camino.

```
if((opcion=="Camino A - Caja Roja") || (opcion=="Camino A - Caja Azul") || (opcion=="Camino A - Caja Negra")  ){


    if(opcion=="Camino A - Caja Roja"){
        wperA1=0
        wpesA1=0
        y=0
    }
    else if(opcion=="Camino A - Caja Azul"){
        wperA1=masa*9.8*0.4
        wpesA1=-masa*9.8*0.4
        y=0.4
    }
    else if(opcion=="Camino A - Caja Negra"){
        wperA1=masa*9.8*0.8
        wpesA1=-masa*9.8*0.8
        y=0.8
    }

    L1=base_rampas_metros/Math.cos(teta)

    wperA2=-masa*9.8*0.8
    wpesA2=masa*9.8*0.8

    B=angulo_fuerza_del_bloque_caminoA*Math.PI/180
    Fuerza_bloque_caminoA=1.1*(masa*9.8*(Math.sin(teta)+miu1*Math.cos(teta)))/(Math.cos(B)-miu1*Math.sin(B))
    //document.getElementById("cc").value=Fuerza_bloque_caminoA

    wperA2_A3=Fuerza_bloque_caminoA*L1*Math.cos(B)
    wpesA2_A3=-masa*9.8*Math.sin(teta)*L1

    wfrA=Math.round((-miu1*(masa*9.8*Math.cos(teta)+Fuerza_bloque_caminoA*Math.sin(B))*L1)*100)/100

    wperA3=masa*9.8*0.8
    wpesA3=-masa*9.8*0.8
```

Figura 87. Cálculos numéricos asociados al camino A.

b) Función Resultados ()

Los procesos dentro de la función **resultados ()** comienzan con el llamado a la **función datos ()** para poder acceder a los valores numéricos encontrados a través de ella. Posteriormente se asignan los valores a las tablas para cada camino mediante el método **innerHTML**, como se observa en la figura 88, y posteriormente se llama a la función respectiva encargada de realizar las gráficas. Para el caso del camino A se llama la función **gráfica_tramoA()**.

```
    if((opcion=="Camino A - Caja Roja") || (opcion=="Camino A - Caja Azul") || (opcion=="Camino A - Caja Negra")  ){



    document.getElementById("m1").innerHTML="LONGITUD DEL TRAMO A2-A3 (m)"
    document.getElementById("n1").innerHTML=Math.round(L1*100)/100
    document.getElementById("m2").innerHTML="VELOCIDAD FINAL TRAMO A2-A3 (m/s)"
    document.getElementById("n2").innerHTML=Math.round(vfA3*100)/100
    document.getElementById("m3").innerHTML="ACELERACION TRAMO A2-A3 (m/s^2)"
    document.getElementById("n3").innerHTML=Math.round(aA*100)/100
    document.getElementById("m4").innerHTML="TRABAJO NETO DEL OBRERO (J)"
    document.getElementById("n4").innerHTML=wperA_neto
    document.getElementById("m5").innerHTML="TRABAJO NETO DEL PESO (J)"
    document.getElementById("n5").innerHTML=wpesA_neto
    document.getElementById("m6").innerHTML="TRABAJO NETO FUERZA DE ROZAMIENTO (J)"
    document.getElementById("n6").innerHTML=Math.round(wfrA*100)/100
    document.getElementById("m7").innerHTML="TRABAJO NETO (J)"
    document.getElementById("n7").innerHTML=wnetoA

    document.getElementById("ti").innerHTML="RESULTADOS NUMERICOS Y GRAFICAS CAMINO A"


    this.grafica_tramoA()

}
```

Figura 88. Resultados numéricos asociados al camino A.

6. CODIFICACIÓN DE LA PRÁCTICA DE COLISIONES

Como se muestra en la figura 89, el desarrollo de la práctica de colisiones parte con la definición de las dimensiones del escenario. También se define la variable Uncm, la cual depende del valor especificado para el ancho de dicho escenario, para llevar a cabo la relación centímetros con pixeles, con lo que se establece que un centímetro equivale a 3.10 pixeles.

```
var resolucion_ancho=1366
var resolucion_alto=768
var porcentaje_ancho_pantalla=0.7
var porcentaje_alto_pantalla=0.85
var ancho = resolucion_ancho* porcentaje_ancho_pantalla
var alto=resolucion_alto*porcentaje_alto_pantalla

var Uncm = ancho/308  //un centimetro equivale a 3.10 pixeles
```

Figura 89. Definición de las dimensiones del escenario de simulación.

Posteriormente se implementa la estructura de desarrollo de Phaser, empezando con la **función Init ()** como se muestra a continuación:

6.1. Función Init ()

En esta función se definen las variables más importantes requeridas para el diseño de la aplicación, como por ejemplo los puntos del recuadro por donde se puede mover la bola 2 (bola amarilla), los puntos de los vértices de las buchacas, los puntos de la posición inicial de la 1 (bola blanca), la distancia entre el taco y la bola 1 entre otras, como se observa en la figura 90.

```
init: function () {

centro = {
    x:game.width / 2,
    y:game.height / 2
}
mover=true
cont=0
k=0
//Los siguientes puntos se encontraron con base a la mesa de billar utlizando la poscion del mouse
//puntos del recuadro por donde se puede mover la bola amarilla
punto_inicial_y_recuadroBlanco=196
punto_final_y_recuadroBlanco=454
punto_inicial_x_recuadroBlanco=686
punto_final_x_recuadroBlanco=809

//puntos de los vertices de las buchacas
punto_x_buchacaSuperiorIzquierda=240
punto_x_buchacas1y2=863
punto_y_buchacaSuperior_izquierdaYderecha=184
punto_y_buchacaInferior_izquierdaYderecha=468


//variables para la ubicacion del taco, y la bola blanca
dis_entre_tacoYbolaBlanca=Uncm*30
dis_comienza_taco=Uncm*5
largo_taco_aprox=Uncm*123
punto_inicial_x_bolaBlanca=dis_comienza_taco+largo_taco_aprox+dis_entre_tacoYbolaBlanca
punto_inicial_y_bolaBlanca=centro.y-Uncm*11.9
punto_inicial_y_taco=centro.y-Uncm*11.9


},
```

Figura 90. Definición de la función Init ().

Por su parte, en la figura 91 se muestra el juego de billar implementado con los respectivos vértices de las buchacas dados en pixeles. Estos puntos son de gran importancia ya que a partir de ellos se determina si la bola amarilla es ingresada en alguna de estas dos buchacas. También son la base para encontrar los puntos de la mesa en los que las dos bolas impactan con las bandas, de modo que se pueda definir la trayectoria que siguen después de la colisión.

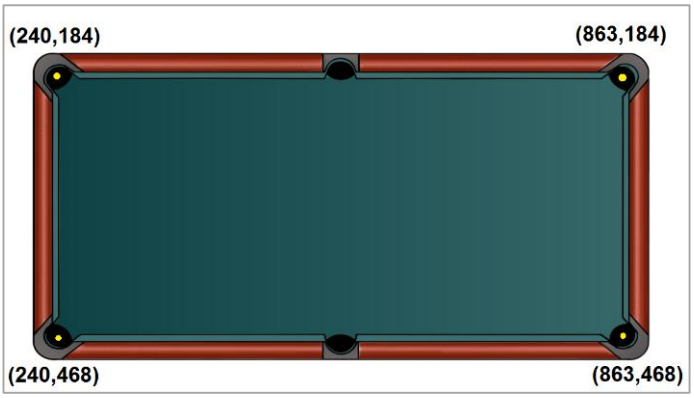


Figura 91. Vértices de la mesa en pixeles.

6.2. Función Preload ()

Dentro de esta función se cargan las imágenes correspondientes a la mesa de billar, el piso del escenario sobre el que está puesta la mesa, el taco de billar, la bola 1 (blanca) y la bola 2 (amarilla). Como se observa en las líneas cuatro y seis del código de la figura 92 es necesario definir las dimensiones (ancho y alto) de cada cuadro de imagen de los sprites que representan a ambas bolas de billar para implementar las animaciones de su movimiento dentro de la mesa.

```
    preload: function () {

    game.load.spritesheet('piso', 'imagenes/pisomadera.png')
    game.load.spritesheet('mesa', 'imagenes/mesadebillar1.png')
    game.load.spritesheet('taco', 'imagenes/taco1.png')
    game.load.spritesheet('blanca', 'imagenes/spritebolablanca1.png',32,32)
    game.load.spritesheet('roja', 'imagenes/bolaroja1.png')
    game.load.spritesheet('amarilla', 'imagenes/spritebolamarilla.png',32,32)

},
```

Figura 92. Función Preload ().

6.3. Función Create ()

Esta función se encarga de llamar a la función **cargar_datos ()**, usada en la creación del escenario. Posteriormente se definen las funciones asociadas a los botones, en las cuales, por medio de la variable k se controla que cada nuevo proceso de simulación se realice siempre y cuando sea presionado el botón

reiniciar. De esta manera, dependiendo de los condicionales, si se presiona el botón reiniciar se llama la función **cargar_datos ()**, mientras que si se presiona el botón simulación la función que se llama es **simulación ()** (ver figura 93).

```
create: function(){

this.cargar_datos()

/*this.inicializar_graficas()*/

    document.getElementById('Enviar').onclick = function (ev) {
    if (!enMovimiento) {
        this.cargar_datos()
        k=0
        }
    }.bind(this)

     document.getElementById('Enviar1').onclick = function (ev) {
    if ((!enMovimiento) && (!mover)){

        if(k==0){this.simulacion()
                    k=1}
        else {
            alert('DEBE REINICIAR PARA VOLVER A SIMULAR')}
}

else if(mover=true){alert('DEBE FIJAR EL ANGULO DE TIRO')}
    }.bind(this)


},
```

Figura 93. Definición de la función Create ().

6.4. Funciones asociadas al botón Reiniciar

a) Función cargar_datos ()

En esta función se inicia agregando los elementos que conforman el escenario de simulación, como por ejemplo el piso y la mesa de billar. De esta manera, como se observa en la figura 94, el piso se agrega en la posición (0,0), correspondiente al extremo superior izquierdo del lienzo, y la mesa de billar se agrega a conveniencia dejando 20 centímetros a la derecha, ya que se ubica en una posición en x de *game.width-Uncm*20* y se toma como punto de referencia para mover la imagen en x y y, como se especifica mediante el método **anchor.set(1, 0.5)**.

```
piso = game.add.sprite(0, 0, 'piso')

mesadebillar = game.add.sprite(game.width-Uncm*20, centro.y, 'mesa')
mesadebillar.scale.setTo(game.width / 1400);
mesadebillar.anchor.set(1,0.5);
```

Figura 94. Procedimiento para agregar el piso y la mesa de billar al escenario.

Posteriormente se agregan las imágenes correspondientes al taco de billar, la bola amarilla y la bola blanca. Para el caso de los sprites de las bolas blanca y amarilla

se define una animación que es utilizada para simular su movimiento dentro de la mesa, como se observa en la figura 95. Además, con el método **anchor.set (0.5,0.5)** se establece como punto de referencia de movimiento el centro de las bolas.

```
taco = game.add.sprite(largo_taco_aprox+dis_comienza_taco, punto_inicial_y_taco, 'taco')
taco.scale.setTo(game.width / 1000);
taco.anchor.set(1,0.5);

amarilla = game.add.sprite(xam*Uncm+punto_x_buchacaSuperiorIzquierda,-yam*Uncm+punto_y_buchacaSuperior_izquierdaYderecha, 'amarilla
amarilla.scale.setTo(game.width / 1400);
amarilla.anchor.set(0.5,0.5);
amarilla.animations.add('uno', [0,1,2,3,4], 12, true)

blanca = game.add.sprite(punto_inicial_x_bolaBlanca, punto_inicial_y_bolaBlanca, 'blanca')
blanca.scale.setTo(game.width / 1400);
blanca.anchor.set(0.5,0.5);
blanca.animations.add('uno', [0,1,2,4], 12, true)
```

Figura 95. Procedimiento para agregar el taco y las dos bolas de billar.

También se crean cuatro bitmapData para dibujar las respectivas líneas, figuras y texto dentro de la interfaz. Por ejemplo, con la variable *bmd1* y su contexto se escribe la información que es estática, como el título de la práctica, la información de las coordenadas de las buchacas y los datos que se muestran debajo de la mesa de billar. Por su parte con la variable *bmd2* se realiza el dibujo del arco sobre el cual se mueve el taco de billar y también el rectángulo que representa el área de ubicación de la bola amarilla. Con la variable *bmd3* y su contexto se traza la línea que va desde la punta del taco hasta la bola amarilla, con la cual se indica la dirección del taco a medida que va cambiando, como producto de la posición del mouse. Finalmente con la variable *bmd4* y su contexto se realizan las líneas encargadas de indicar la dirección de la bola blanca y amarilla después de que ocurra la colisión. Todos estos procesos se muestran en la figura 96.

```
bmd1 = this.add.bitmapData(game.width,game.height)
bmd1.addToWorld()

bmd1.clear()
ctx1=bmd1.context;

bmd2 = this.add.bitmapData(game.width,game.height)
bmd2.addToWorld()

bmd2.clear()
ctx2=bmd2.context;

bmd3 = this.add.bitmapData(game.width,game.height)
bmd3.addToWorld()

bmd3.clear()
ctx3=bmd3.context;

bmd4 = this.add.bitmapData(game.width,game.height)
bmd4.addToWorld()

bmd4.clear()
ctx4=bmd4.context;
```

Figura 96. Creación de los cuatro bitmapData para graficar.

Finalmente, dentro de esta función se limpian las tablas de simulaciones anteriores y también se muestra la información correspondiente a la posición de la bola blanca y los puntos de los vértices de la buchaca 1 y buchaca 2, por medio de la estructura de código que se muestra en la figura 97. En el segundo bloque de código se emplea el método **round()** para redondear los valores numéricos obtenidos. Como se desea mostrar números con dos decimales, lo que se quiere redondear se multiplica por 100 y después de redondeado se divide por 100.

```
document.getElementById("j1").innerHTML=""
document.getElementById("j2").innerHTML=""
document.getElementById("j3").innerHTML=""
document.getElementById("j4").innerHTML=""
//document.getElementById("j5").innerHTML=""
document.getElementById("j6").innerHTML=""
//document.getElementById("j7").innerHTML=""

document.getElementById("x2").value=Math.round(((blanca.x-punto_x_buchacaSuperiorIzquierda)/Uncm)*100)/100
document.getElementById("y2").value=-Math.round(((blanca.y-punto_y_buchacaSuperior_izquierdaYderecha)/Uncm)*100)/100
document.getElementById("x4").value=Math.round(((punto_x_buchacas1y2-punto_x_buchacaSuperiorIzquierda)/Uncm)*100)/100
document.getElementById("y4").value=0
document.getElementById("x6").value=Math.round(((punto_x_buchacas1y2-punto_x_buchacaSuperiorIzquierda)/Uncm)*100)/100
document.getElementById("y6").value=-Math.round(((punto_y_buchacaInferior_izquierdaYderecha-punto_y_buchacaSuperior_izquierdaYdere

},
```

Figura 97. Limpieza de la tabla e Información de las buchacas.

Funciones asociadas al boton Simulación

a) Funcion fórmulas ()

En esta función se realizan los cálculos numéricos necesarios para encontrar las respectivas velocidades de las bolas de billar en los diferentes puntos de la mesa.

b) Funcion simulacion

Esta función inicia con la creación de las variables requeridas para llevar a cabo los movimientos de los elementos del billar, de esta manera, como lo muestra la figura 98, se definen las variables para guardar los puntos de las trayectorias del taco y de las bolas de billar. Para llevar a cabo estos procesos se requieren cinco variables, las cuales son empleadas de la siguiente manera:

- ***Camino:*** Se guardan los puntos de la trayectoria que sigue el taco de billar hasta golpear la bola 1 (bola blanca).

- ***Camino1:*** En esta variable se guardan los puntos de la trayectoria seguida por la bola blanca desde el punto inicial hasta el punto de colisión, cuando es golpeada por el taco.

- ***Camino2:*** Se emplea para guardar los puntos de la trayectoria seguida por la bola blanca después de la colisión hasta llegar a impactar una de las bandas o entrar en la buchaca.

- ***Camino3:*** Se guardan los puntos de la trayectoria realizada por la bola amarilla después de la colisión hasta impactar alguna de las bandas o entrar en las buchacas.

- **Camino4 y camino5:** Finalmente en estas variables se guardan los puntos de las trayectorias de rebote en las bandas seguidas por la bola blanca y la bola amarilla respectivamente.

```
camino = []
camino1 = []
camino2 = []
camino3 = []
camino4 = []
camino5 = []

enMovimiento = true
paso=0;
paso1=0;
paso2=0;
paso3=0;
paso4=0;
paso5=0;
```

Figura 98. Creacion de las variables para simular.

Posterior a la creación de las variables mencionadas inicialmente, se establece la variable *enMovimiento* con un valor lógico de *true* para indicar al sistema que se pueden activar las animaciones y los movimientos definidos, los cuales son controlados en la función **Update ()**. También se crean las variables *paso*, *paso1*, *paso2*, *paso3*, *paso4* y *paso5*, cada una asociadas a las variables *camino*, las cuales son empleadas para acceder a los puntos guardados en estas variables. Por medio de un ciclo for en la función **Update ()** las variables paso se van aumentando en una unidad, recorriendo cada uno de los valores de las variables *camino* de tipo array.

Por otro lado se requiere que cada vez que sea presionado el botón simulación se borren los trazos de dibujo correspondientes al arco que forma el taco de billar, el rectángulo de ubicación de la bola amarilla, las líneas que indican la dirección de tiro del taco y la dirección de las bolas blanca y amarilla después del choque respectivamente. Para ello se limpian los bitmapData con los que se crearon los trazos anteriores (ver figura 99).

```
bmd4.clear()
bmd3.clear()
bmd2.clear()
```

Figura 99. Procedimiento para limpiar los bitmapData.

Posteriormente se llama a la función donde se encuentran las fórmulas de movimiento y se asignan los valores respectivos a cada uno de los identificadores de las cajas de la tabla de resultados numéricos por medio del método **innerHTML**, como se observa en el código de la figura 100.

```
this.formulas()


    document.getElementById("j1").innerHTML=Math.round(vi_b_t*100)/100//teta_amarilla
    document.getElementById("j2").innerHTML=Math.round(v1i*100)/100
    document.getElementById("j3").innerHTML=Math.round(v1f*100)/100
    document.getElementById("j4").innerHTML=Math.round(v2f*100)/100

    if(cof2>0){
    document.getElementById("j6").innerHTML=Math.round(v2f1*100)/100
    }
    else{
    document.getElementById("j6").innerHTML=0
    }
```

Figura 100. Llamado a la función fórmulas () y asignación de valores a la tabla de resultados.

La parte más importante de esta función consiste en encontrar los puntos de las trayectorias seguidas por las bolas de billar, para esto se emplea el método de la interpolación lineal, donde se debe especificar el punto de inicio y el punto de llegada para cada trayecto. Estos puntos fueron encontrados en la función **Update ()**, donde se realiza la mayor parte de la lógica de simulación debido a su capacidad de actualizarse constantemente, propiedad que permite obtener la posición del mouse cada vez que es desplazado. A continuación se detalla un poco más esta función:

6.5. Función Update ()

En esta función se definen las variables m y n, a través de las cuales se obtiene la posición del mouse dentro de la mesa de billar, como se observa en el código de la figura 101, donde el valor obtenido se redondea con dos decimales mediante el método matemático de JavaScript denominado **round()**. Dado que son dos los decimales utilizados en la representación de los números, de acuerdo al método se multiplica y se divide por cien.

```
m=Math.round(game.input.mousePointer.x*100)/100
n=Math.round(game.input.mousePointer.y*100)/100
```

Figura 101. Creación de las variables con las que se obtiene la posición del mouse.

Posteriormente se crea la funcion asociada al evento ondbclick, el cual detecta cuando se hace doble click sobre algun elemento. Esta función se muestra a continuación:

6.6.1. Creación de la función que permite cambiar la posición de la bola amarilla.

En la figura 102 se muestra el código donde se define la función asociada al evento ondbclick, con el cual, si se hace doble click sobre el área demarcada para la ubicación de la bola amarilla, esta toma la posición del mouse mediante las variables *amarilla.x=m* y *amarilla.y=n*.

```
document.getElementById("juego").ondblclick= function (ev) {

if((n>=punto_inicial_y_recuadroBlanco) && (n<=punto_final_y_recuadroBlanco) && (m>=punto_inicial_x_recuadroBlanco) &&(m<=punto_fin

if((!enMovimiento)) {
amarilla.x=m
amarilla.y=n
document.getElementById("x").value=Math.round(((m-punto_x_buchacaSuperiorIzquierda)/Uncm)*100)/100
document.getElementById("y").value=-Math.round(((n-punto_y_buchacaSuperior_izquierdaYderecha)/Uncm)*100)/100
}
else{return}
}
}
```

Figura 102. Definición de la función asociada al evento ondbclick.

En la figura x97 se muestra el codigo donde se define la funcion asociada al evento ondbclick en el cual si se hace doble click sobre el area demarcada por el rectangulo blanco, la posicion de la bola blanca toma la posicion del mouse de tal manera que mediante amarilla.x=m y amarilla.y=n se le asigna dicha posicion.

6.6.2. Proceso para encontrar el ángulo de tiro y la dirección de las bolas de billar.

El proceso para determinar el ángulo de tiro del taco de billar, el cual depende de la posición del mouse, requiere que se cumpla el condicional del código de la figura 103. Para que esto se cumpla, el mouse debe estar en movimiento, la variable *mover* debe tener un valor lógico de *true* y la variable *enMovimiento* debe tener el valor de false.

```
if((document.onmousemove) && (mover) && (!enMovimiento)){
```

Figura 103. Condicional para empezar el proceso de definición del ángulo de tiro.

Con la variable *mover* se controla el proceso para fijar el ángulo de tiro, ya que en un inicio cuando se abre la aplicación, esta variable tiene el valor lógico *true*. Para fijar el valor del ángulo, como es necesario que se haga click sobre el área de la mesa de billar, la variable *mover* toma el valor de *false*, indicando que el ángulo de tiro deja de depender de la posición del mouse. Este proceso se controla mediante la función asociada al evento onclick, como se observa en el código de la figura 104, donde se muestra un fragmento de esta función.

Se tiene inicialmente la variable *cont*, la cual se va incrementando en una unidad cada vez que se hace clik sobre el elemento con identificador juego, que para este caso corresponde al escenario. De esta manera, cuando se hace el primer click, la variable *cont* toma el valor de uno y por lo tanto la variable *mover* adquiere el valor lógico de *false*. Si se vuelve a hacer click sobre el área señalada, la variable *cont* ahora toma el valor de dos, indicando que se puede volver a mover el ángulo de tiro. Debido a esto, la variable *mover* vuelve a tomar el valor lógico true y el contador se inicializa nuevamente en cero.

```
document.getElementById("juego").onclick= function (ev) {
    cont=cont+1

    if(cont==1){
    mover=false


    if(cont==2){
        cont=0
        mover=true }
```

Figura 104. Parte de la función asociada al evento onclick.

Por otro lado para encontrar el ángulo de tiro, la lógica empleada consiste en determinar cuál es el área por donde se puede mover el cursor del mouse para que la bola blanca colisione con la bola amarilla, como se observa en la figura 105. Si la posición en y se encuentra en el área que demarcan los corchetes de esta figura, la bola blanca colisiona con la amarilla y el área que abarca la bola de impacto (blanca) será proyectada sobre la bola objetivo (amarilla).

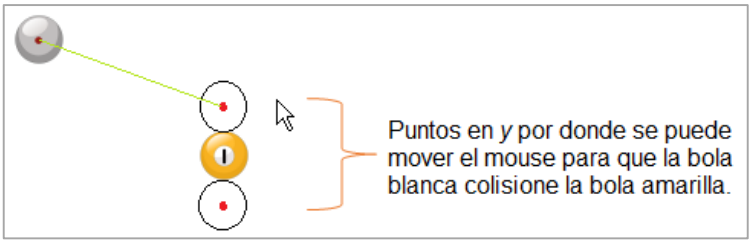


Figura 105. Puntos en y por donde se puede mover el mouse para que la bola blanca colisione la bola amarilla.

Cuando se presenta el caso de la figura 106, donde el cursor del mouse no se encuentra en el área demarcada de la figura 105 pero sí se proyecta una línea desde el centro de la bola blanca hasta el cursor y esta sigue dicha trayectoria, también habrá colisión con la bola amarilla. Dado que toda la lógica se halla con base en los puntos del área demarcada en la figura 105, se requiere recalcular el punto en y de la posición del cursor, que para este caso sería la variable n.

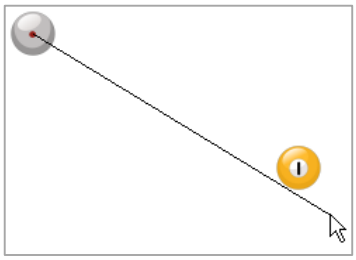


Figura 106. Casos donde hay que recalcular el valor en y de la posición del mouse.

Posteriormente se tienen tres casos adicionales, los cuales son los siguientes:

a. El puntero del mouse se ubica en una posición en la cual la bola blanca no colisiona con la bola amarilla.

b. El puntero del mouse se encuentra dentro de las posiciones en y que ocupa la bola amarilla.

c. El puntero se ubica por encima o por debajo de la bola amarilla sin embargo con dicho ángulo de tiro que se forma, la bola blanca alcanza a colisionar a la bola amarilla.

Para el caso (a) la bola blanca solamente se desplaza hasta la posición indicada por el puntero del mouse, sin embargo para los casos (b) y (c), se tiene un desarrollo matemático bastante extenso ya que se tiene que evaluar si la bola amarilla está por encima o por debajo de la bola blanca. También se debe mirar si la bola blanca colisiona por arriba o por abajo a la bola amarrilla, ya que según los anteriores casos, el punto de colisión y la dirección de las bolas después del choque, así como los posibles rebotes en las bandas cambian.

Para encontrar las trayectorias después de la colisión se hallan los ángulos de dirección para las dos bolas con base en los cálculos matemáticos que encuentran en el anexo A, posteriormente, al tener el valor de esta dirección se hace uso de la ecuación de la recta que pasa por dos puntos para encontrar la trayectoria que siguen las bolas después de la colisión y cuando rebotan.

Anexo F

ENCUESTA DE MANEJO DE APLICACIÓN

La encuesta de manejo de la aplicación software para cada práctica fue diseñada con base en una serie de preguntas con respuestas únicas y excluyentes (SI o NO), y su proceso de implementación y difusión se llevó a cabo a través de la plataforma Drive de Google, por medio de sus formularios gratuitos. El esquema general de las preguntas contenidas en cada encuesta se realizó con base en el siguiente formato (ver tabla 1):

N°	Tipo de pregunta	Opción	
1	¿Al leer la guía para el desarrollo de la práctica le resultó fácil el ingreso a la aplicación?	Si	No
2	¿En la guía, los procedimientos para realizar la práctica se enuncian de forma clara?	Si	No
3	¿La prueba de conocimiento al final de cada actividad está relacionada con los conceptos desarrollados dentro de la práctica?	Si	No
4	¿La información teórica y las fórmulas dadas en la guía fueron de utilidad para resolver las respectivas pruebas de conocimiento?	Si	No
5	¿El manejo de los diferentes componentes de la aplicación fue intuitivo?	Si	No
6	¿La herramienta software presentó algún problema durante la simulación?	Si	No
7	¿Comprendió los resultados numéricos generados por la herramienta?	Si	No
8	¿Las gráficas generadas en el simulador le brindaron	Si	NO

	información clara?		
9	¿La interacción con la aplicación le resultó agradable?	Si	No
10	¿Considera útil esta aplicación para complementar los conceptos teóricos vistos en clase?	Si	No
11	¿Considera usted que se debe realizar alguna mejora o modificación a la herramienta?	Si	No
12	Observaciones:		

Tabla 1. Esquema general de la encuesta de manejo de la aplicación web.

De acuerdo a los datos obtenidos a través de los formularios de Google Drive los resultados numéricos arrojados por la encuesta son los siguientes (ver tabla 2):

Numero de encuestados: 48 estudiantes.				
Pregunta	**SI**	**%**	**NO**	**%**
1	47	97,91	1	2,09
2	44	91,66	4	8,34
3	48	100	0	0
4	45	93,75	3	6.25
5	33	68,75	15	31.25
6	32	66,66	16	33,34
7	47	97,91	1	2.9
8	45	93,75	3	6,25
9	45	93,75	3	6,25
10	43	89,58	5	10,42
11	30	62,5	18	37,5

Tabla 2. Resultados numéricos de las encuesta de manejo de la aplicación.

Nota: El respectivo análisis de estos resultados se lleva a cabo dentro de la sección 3.4.2 de la monografía.

Anexo G

EJECUTABLE DE LA APLICACIÓN

Para la entrega del archivo ejecutable se procedió inicialmente a empaquetar la aplicación por medio del framework para JavaScript **Electron,** el cual permite desarrollar aplicaciones de escritorio haciendo uso de tecnologías web. Sin embargo la aplicación que se obtuvo no poseía todas las características de la versión web, ya que, en primer lugar no permitía la opción de abrir los documentos en pdf y tampoco era posible actualizar la página, sumado a esto algunas de las prácticas no las desplegaba de manera correcta.

De este modo las funcionalidades faltantes de la versión empaquetada debían ser agregadas por medio de otros procedimientos, muchos de ellos bastante engorrosos. Otra de las desventajas presentadas tenía que ver con el tamaño de la carpeta que contiene la aplicación empaquetada, la cual presentaba un peso muy grande (aproximadamente 462 MB) lo que dificultaba su distribución a través de internet. Por esta y otras razones se descartó la entrega de este archivo y se buscó una opción más práctica. En la figura 1 se muestra como quedo empaquetada la aplicación con Electron, en ella se verifica la realización de este proceso, además en la figura 2 se observan las propiedades de la carpeta en la cual se guarda la página empaquetada donde se verifica que en efecto tiene un tamaño de 462 MB:

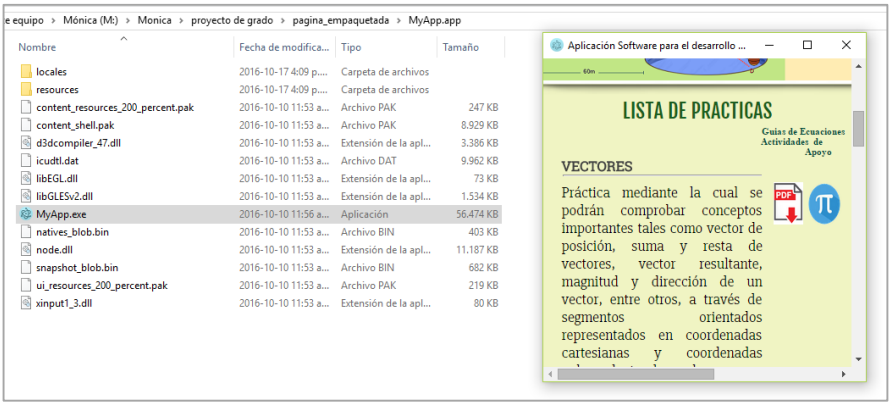


Figura 1. Aplicación empaquetada con Electron.

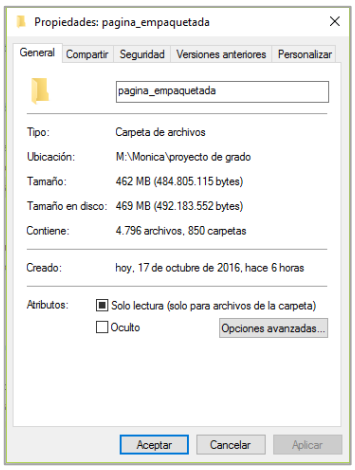


Figura 2.Propiedades de la carpeta que contiene la aplicación empaquetada.

Por esta razón se optó por entregar dos archivos que contienen la aplicación web, uno donde se tiene acceso al código y otro donde se aplicó la técnica de ofuscar el código .js, de modo que no sea posible acceder a las líneas tan fácilmente.

La técnica de ofuscar consiste en hacer inentendible el código para cualquier usuario que acceda a él, pero sin afectar su lógica y funcionalidad. También se emplea esta técnica con el fin de reducir el tamaño del código fuente ya que todo lo comprime a una línea. Para realizar este proceso se empleó la siguiente página:

http://www.javascriptobfuscator.com/Javascript-Obfuscator.aspx

Donde solo basta con copiar el código que se quiere ofuscar y pegarlo en el lugar indicado dentro de dicha página. El procedimiento genera un código poco entendible que deberá ser pegado en el lugar de origen.

De esta manera la carpeta que contiene los archivos necesarios para acceder al código se denomina '**página**', y una vez abierta solo basta con abrir el archivo aplicación.html para acceder a la aplicación, como se muestra en la figura 3.

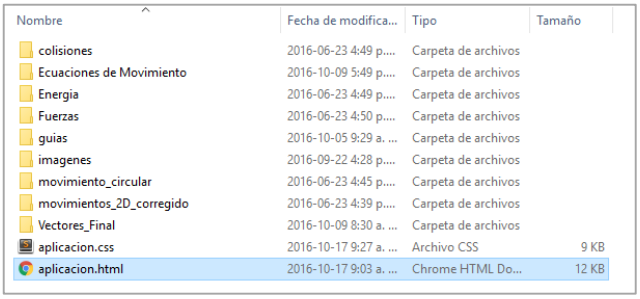


Figura 3. Acceso a la carpeta página donde se tiene la aplicación.

En la figura 4 se observan las propiedades de la carpeta “página" donde se ve que tiene un tamaño de 55.7 MB. De esta manera se corrobora que el tamaño de esta carpeta es mucho menor que la correspondiente al archivo empaquetado, lo que permite una mejor distribución por diferentes medios.

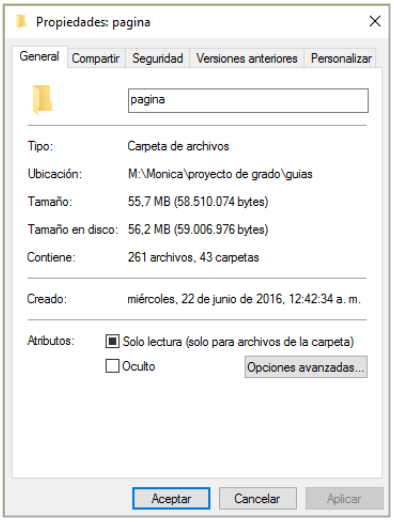


Figura 4. Propiedades de la carpeta página.

Similarmente en la carpeta con nombre “página ofuscada” se tiene la aplicación con los archivos .js ofuscados. Estos archivos fueron subidos al servidor gratuito para evitar que se pueda acceder al código con facilidad. En conclusión para esta aplicación no es conveniente tener un ejecutable ya que demanda más procesos y tamaño en disco, por lo tanto es mejor poder acceder a los archivos .html ya que solo se necesita tener instalado un navegador.