
	<p align="center">UNIVERSIDAD NACIONAL DE SAN AGUSTÍN FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y SERVICIOS ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMAS</p>	
<b>Formato:</b> Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		
<b>Aprobación:</b> 2022/03/01	<b>Código:</b> GUIA-PRLD-001	<b>Página:</b> 1

## INFORME DE LABORATORIO

### INFORMACIÓN GENERAL

<b>ASIGNATURA</b>	Computación Gráfica, Visión Computacional y Multimedia				
<b>TÍTULO DE LA PRÁCTICA</b>	Motores de Física y Colisiones				
<b>NÚMERO DE LA PRÁCTICA</b>	1	<b>AÑO LECTIVO</b>	2025A	<b>NRO SEMESTRE</b>	IX
<b>FECHA DE PRESENTACIÓN</b>	5/5/2025	<b>HORA DE PRESENTACIÓN</b>	10:00		
<b>INTEGRANTE(S):</b> <ul style="list-style-type: none"> <li>• Ttito Campos Rutbel Carlos</li> </ul>				<b>NOTA</b>	
<b>DOCENTE(S):</b> <ul style="list-style-type: none"> <li>• Ing. Diego Alonso Iquira Becerra</li> </ul>					

### 1. INTRODUCCIÓN



Esta es una breve introducción para esta guía de laboratorio

### 2. OBJETIVOS

- Conoce, comprende y analiza aplicaciones multimedia
- Analiza una aplicación multimedia investigando las tecnologías que la conforman y como se realizó su desarrollo

### 3. SOLUCIÓN DE EJERCICIOS/PROBLEMAS

Utilicen las habilidades que han aprendido en los cursos anteriores para crear una aplicación enfocada en el lanzamiento de proyectiles utilizando el motor de física. Se puede crear un juego de baloncesto donde se busque lanzar una pelota en un aro de baloncesto o un juego donde lancen proyectiles a enemigos y los enemigos puedan atacar al usuario con proyectiles.

	<p>UNIVERSIDAD NACIONAL DE SAN AGUSTÍN FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y SERVICIOS ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMAS</p>	
<p><b>Formato:</b> Guía de Práctica de Laboratorio / Talleres / Centros de Simulación</p>		
<p><b>Aprobación:</b> 2022/03/01</p>	<p><b>Código:</b> GUIA-PRLD-001</p>	<p><b>Página:</b> 2</p>

- Tendrán que utilizar el motor de física para realizar el lanzamiento, donde se puedan modificar los valores físicos y estos afecten el desplazamiento del objeto.
- La aplicación debe permitir gestionar valores como la puntuación, basada en las canastas obtenidas, o la vida de los enemigos o del jugador.
- Deberán establecer cómo funcionarán las colisiones del proyectil.
- La aplicación debe permitir gestionar valores como la puntuación.
- Se busca que investiguen qué otros elementos pueden agregar a la aplicación para mejorar la experiencia del usuario, como efectos visuales y sonoros o la implementación de diferentes tipos de proyectiles.

### 3.1. Entorno y personajes

#### 3.1.1. Escenario

El escenario es una cancha de baloncesto con un jugador en medio. Algunos de sus elementos son:

- **Cancha (Floor):** Superficie que representa la zona de juego donde se mueve el jugador y se lanza la pelota.
- **Aro de Baloncesto (Hoop):** Incluye el tablero y un aro de baloncesto en el centro del escenario.
- **Fondo (Background):** Imagen de un estadio lleno de espectadores.
- **Elementos Adicionales:** Objetos como Canvas, Límites relacionados con la interfaz de usuario y lógica de colisiones.

#### 3.1.2. Jugador

El objeto Player representa al personaje controlable. Cuenta con:

- Capsule Collider para colisiones
- Script Basketball Controller para gestionar movimiento y acciones



	<p>UNIVERSIDAD NACIONAL DE SAN AGUSTÍN FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y SERVICIOS ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMAS</p>	
<p><b>Formato:</b> Guía de Práctica de Laboratorio / Talleres / Centros de Simulación</p>		
<p><b>Aprobación:</b> 2022/03/01</p>	<p><b>Código:</b> GUIA-PRLD-001</p>	<p><b>Página:</b> 3</p>



Figura 1: Escenario del juego



Figura 2: Personaje controlable

### 3.1.3. Balón

El objeto Ball representa una pelota de baloncesto con:

- Box Collider y Rigidbody
- Material físico (BouncyBall) para comportamiento realista

## 3.2. Interacción del usuario

### 3.2.1. Movimiento del jugador

El jugador se desplaza en el plano horizontal usando teclas WASD o flechas:



	<p>UNIVERSIDAD NACIONAL DE SAN AGUSTÍN FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y SERVICIOS ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMAS</p>	
<p><b>Formato:</b> Guía de Práctica de Laboratorio / Talleres / Centros de Simulación</p>		
<p><b>Aprobación:</b> 2022/03/01</p>	<p><b>Código:</b> GUIA-PRLD-001</p>	<p><b>Página:</b> 4</p>



Figura 3: Objeto Ball

```
// Movimiento del jugador
Vector3 direction = new(Input.GetAxisRaw("Horizontal"), 0, Input.GetAxisRaw("Vertical"));
transform.position += MoveSpeed * Time.deltaTime * direction;
if (direction != Vector3.zero)
    transform.LookAt(transform.position + direction);
```





Figura 4: Movimiento del personaje

### 3.2.2. Control del balón

Cuando el jugador tiene el balón (IsBallInHands == true):

```
if (IsBallInHands) {
    if (Input.GetKey(KeyCode.Space)){}
    else {
        // Animación de dribbling
        Ball.position = PosDribble.position + Vector3.up * Mathf.Abs(Mathf.Sin(Time.time * 5f));
        Arms.localEulerAngles = Vector3.right * 0;
    }
}
```

	<p>UNIVERSIDAD NACIONAL DE SAN AGUSTÍN FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y SERVICIOS ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMAS</p>	
<p><b>Formato:</b> Guía de Práctica de Laboratorio / Talleres / Centros de Simulación</p>		
<p><b>Aprobación:</b> 2022/03/01</p>	<p><b>Código:</b> GUIA-PRLD-001</p>	<p><b>Página:</b> 5</p>

}



Figura 5: Movimiento del balón

### 3.2.3. Cargar lanzamiento

Al mantener presionado espacio:

- El balón se mueve a la posición sobre la cabeza
- Se incrementa la fuerza del lanzamiento
- Se dibuja trayectoria parabólica con LineRenderer

```
if (Input.GetKey(KeyCode.Space)) {
    currentLaunchDistance += chargeSpeed * Time.deltaTime;
    currentLaunchDistance = Mathf.Clamp(currentLaunchDistance, baseLaunchDistance, maxLaunchDistance);
    Ball.position = PosOverHead.position;
    Arms.localEulerAngles = Vector3.right * 180;
    DrawTrajectory(PosOverHead.position, launchDirection, currentLaunchDistance);
    TrajectoryLine.enabled = true;
}
```

### 3.2.4. Lanzamiento del balón

Al soltar espacio:



	<p>UNIVERSIDAD NACIONAL DE SAN AGUSTÍN FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y SERVICIOS ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMAS</p>	
<p><b>Formato:</b> Guía de Práctica de Laboratorio / Talleres / Centros de Simulación</p>		
<p><b>Aprobación:</b> 2022/03/01</p>	<p><b>Código:</b> GUIA-PRLD-001</p>	<p><b>Página:</b> 6</p>



Figura 6: Carga del lanzamiento del balón

```



if (Input.GetKeyUp(KeyCode.Space)) {
    launchDistance = currentLaunchDistance;
    currentLaunchDistance = baseLaunchDistance;
    Lanzar();
    IsBallInHands = false;
    TrajectoryLine.enabled = false;
}

void Lanzar() {
    Vector3 launchDirection = transform.forward;
    Vector3 A = PosOverHead.position;
    Ball.position = A;
    Vector3 B = A + launchDirection * launchDistance;
    Rigidbody ballRB = Ball.GetComponent<Rigidbody>();
    ballRB.useGravity = true;
    ballRB.velocity = CalcularVelocidadInicial(A, B);
}

```



Figura 7: Lanzamiento del balón

	<p>UNIVERSIDAD NACIONAL DE SAN AGUSTÍN FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y SERVICIOS ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMAS</p>	
<p><b>Formato:</b> Guía de Práctica de Laboratorio / Talleres / Centros de Simulación</p>		
<p><b>Aprobación:</b> 2022/03/01</p>	<p><b>Código:</b> GUIA-PRLD-001</p>	<p><b>Página:</b> 7</p>

### 3.2.5. Cálculo de velocidad inicial

Fórmulas para trayectoria parabólica:

```
Vector3 CalcularVelocidadInicial(Vector3 origen , Vector3 destino) {
    Vector3 desplazamiento = destino - origen;
    float g = Mathf.Abs(gravity);
    float velocidadY = Mathf.Sqrt(2 * g * maxHeight);
    float t1 = velocidadY / g;
    float h = desplazamiento.y + maxHeight;
    if (h < 0) h = 0;
    float t2 = Mathf.Sqrt(2 * h / g);
    float tTotal = t1 + t2;
    float velocidadX = desplazamiento.x / tTotal;
    float velocidadZ = desplazamiento.z / tTotal;
    return new Vector3(velocidadX , velocidadY , velocidadZ);
}
```

### 3.3. Colisiones

Configuración de colisiones:

- **Jugador:** `Is Trigger`.<sup>a</sup>activado para detectar colisiones sin bloqueo físico
- **Balón:** `Collision Detection`.<sup>en</sup> modo Continuous para evitar traspasos



Figura 8: Propiedad `Is Trigger` del jugador

Manejo de colisiones:

```
private void OnTriggerEnter(Collider other) {
    if (!IsBallInHands) {
        IsBallInHands = true;
    }
}
```



	UNIVERSIDAD NACIONAL DE SAN AGUSTÍN FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y SERVICIOS ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMAS	
<b>Formato:</b> Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		
<b>Aprobación:</b> 2022/03/01	<b>Código:</b> GUIA-PRLD-001	<b>Página:</b> 8



Figura 9: Propiedad "Collision Detection" del balón

### 3.4. Puntuación

Sistema de puntuación:

- Dos colliders Trigger en el aro (superior e inferior)
- Collider inferior marca entradas inválidas
- Collider superior suma puntos si no es inválida



Figura 10: Collider superior e inferior de aro

Lógica de puntuación:

```

public bool CanScore() {
    return !enteredFromBelow && !scored;
}
public void RegisterScore() {
    scored = true;
    ScoreManager.Instance.AddPoint();
}

```





	<p>UNIVERSIDAD NACIONAL DE SAN AGUSTÍN FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y SERVICIOS ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMAS</p>	
<p><b>Formato:</b> Guía de Práctica de Laboratorio / Talleres / Centros de Simulación</p>		
<p><b>Aprobación:</b> 2022/03/01</p>	<p><b>Código:</b> GUIA-PRLD-001</p>	<p><b>Página:</b> 9</p>



Figura 11: Ejemplo de puntaje válido



Figura 12: Ejemplo de puntaje inválido

### 3.5. Efectos visuales

Dibujo de trayectoria:

```
void DrawTrajectory(Vector3 start, Vector3 initialVelocity) {
    TrajectoryLine.positionCount = trajectoryResolution;
    float g = Mathf.Abs(gravity);
    float totalTime = 2 * initialVelocity.y / g;

    for (int i = 0; i < trajectoryResolution; i++) {
        float t = i / (float)(trajectoryResolution - 1) * totalTime;
        Vector3 position = start + new Vector3(
            initialVelocity.x * t,
            initialVelocity.y * t - 0.5f * g * t * t,
            initialVelocity.z * t
        );
        TrajectoryLine.SetPosition(i, position);
    }
}
```

### 4. CONCLUSIONES

- adsassd [1].



	<p>UNIVERSIDAD NACIONAL DE SAN AGUSTÍN FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y SERVICIOS ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMAS</p>	
<p><b>Formato:</b> Guía de Práctica de Laboratorio / Talleres / Centros de Simulación</p>		
<p><b>Aprobación:</b> 2022/03/01</p>	<p><b>Código:</b> GUIA-PRLD-001</p>	<p><b>Página:</b> 10</p>



Figura 13: Ejemplo de trayectorias dibujadas

## REFERENCIAS Y BIBLIOGRAFÍA

### Referencias

- [1] GeeksforGeeks, *How to use Matplotlib*, 2024. [En línea]. Disponible en: [https://www.w3schools.com/python/matplotlib\\_plotting.asp](https://www.w3schools.com/python/matplotlib_plotting.asp).