Medición de un paracaídas
Lic.Contenidos Digitales Interactivos

Jaime Saul Meneses Cruz

Rodrigo Renaud Rojas

Brian Enrique Lara Chaves

Jose Arturo Serrano Benitez

El proyecto a realizar será una medición de un paracaida en caída libre. El objetivo de este proyecto es poder mostrar a las personas cuál es la velocidad máxima que alcanza un paracaídas si lo dejamos caer desde una altura específica.

También dejaremos que el usuario tenga la libertad de experimentar con esta caída, ya que el proyecto será lanzado para PC

En ¿Qué consiste?

Este proyecto consiste en simular una caída libre de un paracaidista. En la cual en el celular de los usuarios se le mostrará la siguiente información:

- Velocidad máxima del paracaida.
- La distancia.
- Aceleración.

Los programas necesarios para el desarrollo del programa son:

- Unity3D.
- Modo: Para la elaboración de los modelos

Para que las personas puedan usar nuestro simulador se requieren:

- Un móvil
- Un cardboard.

Al terminar el desarrollo del proyecto, queremos que nuestros usuarios entiendan cómo son las fuerzas que interactúan al momento de lanzarse desde una altura tan alta. y utilizar el paracaídas.

Para poder llegar a nuestro resultado, se estará utilizando las fórmulas de la aceleración

- $a = dv / dt = (v_f – v_i)/(t_f – t_i)$
- La constante de la gravedad de 9.8. Ya que la simulación será en el planeta tierra.

Funciones del programa.

using System.Collections;

```
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;

public class DisplayAttributes : MonoBehaviour
{

        //Distance
        public GameObject rayStartPos;
        public LayerMask groundLayer;
        private Vector3 groundPos;
        private float distanceFromGround;
        public Text distanceText;

        //velocity
        private float velocity;
        private Vector3 finalPos;
        private Vector3 startPos;
        private float finalTime;
        private float startTime;
        private float time;
        public Text velocityText;


        //Acceleration
        private float acceleration;
        private float startAccel;
        private float finalAccel;
        private float finalATime;
        private float startATime;
        public Text accelerationText;

        // Use this for initialization
        void Start()
        {
        InvokeRepeating("GetVelocityValues", 0.01f, 0.2f);//sirve para llamar la funcion
dentro de 0.01 segundos y despues se ejecuta cada 0.1 segundos
        InvokeRepeating("GetAcceleretionValues", 0.01f, 0.5f);
        }

        // Update is called once per frame
        void Update()
        {
        GetDistance();
        GetVelocity();
        GetAcceleration();
```

```
        }
        public void GetGroundPos()
        {
        Ray ray = new Ray(rayStartPos.transform.position, new Vector3(0, -1f, 0));
        RaycastHit hitinfo;
        if (Physics.Raycast(ray, out hitinfo, groundLayer))
        {
        groundPos = hitinfo.point;
        }
        }
        public void GetDistance()
        {
        GetGroundPos();
        distanceFromGround = Vector3.Distance(rayStartPos.transform.position,
groundPos);
        Debug.Log(distanceFromGround);
        RoundDistance();
        ShowDistance();
        }
        public void RoundDistance()
        {
        distanceFromGround = distanceFromGround * 10;
        distanceFromGround = Mathf.Round(distanceFromGround);
        distanceFromGround = distanceFromGround / 10;
        }
        public void ShowDistance()
        {
        if (distanceFromGround >= 1)
        distanceText.text = "Distancia del suelo: " + distanceFromGround + "m";
        else
        distanceText.text = "Distancia del suelo: 0m";
        }

        public void GetVelocityValues()
        {
        finalPos = startPos;
        startPos = rayStartPos.transform.position;
        finalTime = startTime;
        startTime = time;
        }
        public void GetVelocity()
        {
        time += Time.deltaTime;
        velocity = (Vector3.Distance(finalPos,startPos)) / (startTime - finalTime);
        RoundVelocity();
        ShowVelocity();
```

```
        }
        public void RoundVelocity()
        {
        velocity = velocity * 10;
        velocity = Mathf.Round(velocity);
        velocity = velocity / 10;
        velocity = velocity * 3.6f;
        }
        public void ShowVelocity()
        {
        velocityText.text = "Velocidad: " + velocity + "km/h";
        }

        public void GetAcceleretionValues()
        {
        finalAccel = startAccel;
        startAccel = velocity;
        finalATime = startATime;
        startATime = time;
        }
        public void GetAcceleration()
        {
        acceleration = (finalAccel - startAccel) / (finalATime- startATime);
        RoundAcceleration();
        ShowAcceleration();
        }
        public void RoundAcceleration()
        {
        acceleration = acceleration * 10;
        acceleration = Mathf.Round(acceleration);
        acceleration = acceleration / 10;
        acceleration = acceleration * 3.6f;
        }
        public void ShowAcceleration()
        {
        accelerationText.text = "Aceleracion: " + acceleration + "M/s";
        }
}
```