
Aventura
Sobrecargas

Contenido

DESARROLADORES

Laura Ferrer Haba, Trabajo en equipo con compañeros (nombres omitidos por privacidad)

DESCRIPCCIÓN DEL JUEGO

El juego está realizado en la versión de Unity 2019.3.5f1.

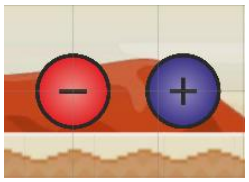
Descripción global del juego:

El juego trata de una carga que debe llegar al final de un recorrido, para ello, deberá ir cambiando de plataforma mientras vaya avanzando mediante el cambio del signo de la carga. El mundo donde se encuentra la carga es un desierto al más puro estilo viejo oeste con una banda sonora creada por TheBossSKNZ.

Elementos:

El proyecto consta de siete elementos:

1. ***La carga:*** será un Sprite cargado negativa o positivamente.

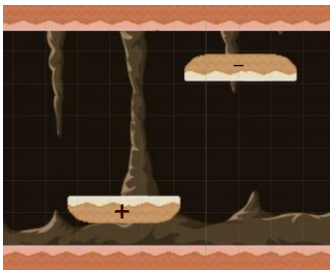


2. ***Dos Cámara:*** Una que seguirá al jugador y otra que se activará cuando el jugador pierda.
3. ***Paredes:*** Son dos paredes que delimitan el campo de juego, al llegar a la pared del final el jugador habrá ganado.

4. ***Suelos Superiores:*** Delimitan el campo de juego, están cagados negativamente.

5. ***Suelos inferiores:*** Delimitan el campo de juego, están cagados positivamente.

6. ***Plataforma móvil:*** Se trata de una plataforma móvil por la cual el jugador tendrá que pasar.



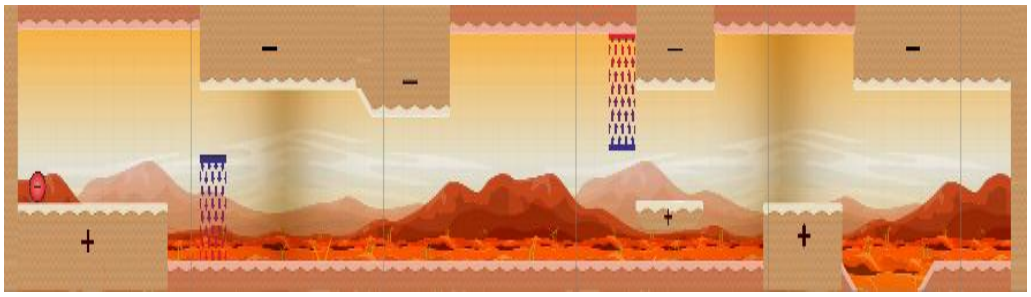
7. ***Condensadores:*** Los condensadores son obstáculos puestos por todo el recorrido y serán los que el jugador tendrá que ir superando, estará cargados negativa o positivamente.

Jugabilidad:

La carga podrá ser positiva o negativa y se podrá hacer la transformación utilizando los clics izquierda y derecho del ratón. La carga por sí sola no puede moverse hacia arriba o hacia abajo, solo podrá ir para la derecha o para la izquierda pulsando las teclas 'A' y 'D' respectivamente. Entonces para poder ir hacia arriba o abajo se tendrá que jugar con las fuerzas eléctricas, es decir, si estás como una carga negativa, serás atraído por el suelo cargado positivamente.

Mapa:

En nuestro trabajo el suelo positivo está colocado en la parte superior y el suelo negativo se sitúa en la parte inferior. El mapa se compone de diferentes plataformas. Está los bloques gordos, que se pueden tocar y rodar sobre ellos; y los bloques finos que tienen una tonalidad roja, esos no se deben tocar ya que sería el "GAME OVER" y se tendría que volver a empezar. También hay una plataforma que hace un recorrido recto. Y por último está el final del mapa, en el muro que se encuentra allí al tocarlo se gana el juego.



Finales:

Este juego tiene tres finales. Dos buenos, uno de ellos es seguir el recorrido directo al muro final y el otro es llegar a otro muro que se encuentra siguiendo un camino escondido. Y el final malo que realmente no es un final, ya que se repite hasta que se pase el juego.

Física:

Los fenómenos que influyen en nuestro videojuego son las leyes de la electrostática que hemos estudiado en clase. La ley que hemos usado es la ley del campo eléctrico, la cual se define como la fuerza eléctrica que experimenta una cagar en ducho punto, dividida entre la carga, y la ley del potencial eléctrico, la cual se define como el trabajo que realiza una fuerza para mover una partícula desde el punto A al punto B.

Glosario Scripts:

- *Script ParticleManager:* Aplica la fuerza que se necesita para mover la carga.

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

public class ParticleManager : MonoBehaviour
{
    private List<ChargedParticle> ChargedParticles;
    private List<MovingChargedParticle> MovingChargedParticles;
    void Start()
    {
        ChargedParticles = new
List<ChargedParticle>(FindObjectsOfType<ChargedParticle>());
        MovingChargedParticles = new
List<MovingChargedParticle>(FindObjectsOfType<MovingChargedParticle>());
    }
    void Update()
    {
        foreach (MovingChargedParticle mcp in MovingChargedParticles)
        {
            ApplyForce(mcp);
        }
    }

    private void ApplyForce (MovingChargedParticle mcp)
    {
        Vector2 NewForce = Vector2.zero;

        foreach(ChargedParticle cp in ChargedParticles)
        {
            if (mcp == cp)
                continue;

            float distance = Vector2.Distance(mcp.transform.position,
cp.gameObject.transform.position);
            float force = 100 * mcp.charged * cp.charged / Mathf.Pow(distance,
2);

            Vector2 direction = mcp.transform.position - cp.transform.position;
            direction = direction.normalized;

            NewForce = force * direction * Time.deltaTime;
            mcp.rb.AddForce(NewForce);
        }
    }
       }
```

❖ *Script MovingChargedParticle:* Hace que la carga se mueva con una velocidad.

```
using System.Collections;
using System.Collections.Generic;
using System.Collections.Specialized;
using System.Diagnostics;
using System.Runtime.InteropServices;
using System.Security.Cryptography;
using UnityEngine;
using UnityEngine.UI;

public class MovingChargedParticle : ChargedParticle
{
    public float mass = 1;
    public Rigidbody2D rb;
    [Range(0f, 1f)] public float speed = 1f;

    public Sprite cargaPositiva;    // Sprites
    public Sprite cargaNegativa;

    public KeyCode ratonDer;        // Cambios de Carga
    public KeyCode ratonIzq;

    public Text marcador;
    private int count;
    public GameObject CamaraGameOver;

    void Start()
    {
        rb = gameObject.GetComponent<Rigidbody2D>();
        rb.mass = mass;
        count = 0;
    }

    void Update()
    {
        if (Input.GetKey(ratonDer))
        {
            if (gameObject.GetComponent<SpriteRenderer>().sprite ==
cargaPositiva)
            {
                charged = charged * -1;
                this.gameObject.GetComponent<SpriteRenderer>().sprite =
cargaNegativa;
                this.gameObject.transform.localScale = new Vector3(1, 1, 1);
            }
        }
        if (Input.GetKey(ratonIzq))
        {
            if (gameObject.GetComponent<SpriteRenderer>().sprite ==
cargaNegativa)
            {
                charged = charged * -1;
                this.gameObject.GetComponent<SpriteRenderer>().sprite =
cargaPositiva;
                this.gameObject.transform.localScale = new Vector3(1, 1, 1);
            }
        }
        if (transform.localPosition.y < -1 && transform.localPosition.x < 32 ||
transform.localPosition.y > 9 || transform.localPosition.y < -4)
        {
            camaraMuerte();
            Invoke("Inicio", 3);
        }
    }
```

```
    void FixedUpdate()
    {
        float moveHorizontal = Input.GetAxis("Horizontal");
        float moveVertical = Input.GetAxis("Vertical");

        Vector3 movement = new Vector3(moveHorizontal, 0.0f, moveVertical);

        rb.AddForce(movement * speed);
    }

    void OnTriggerEnter2D(Collider2D algo)
    {
        if (algo.gameObject.CompareTag("Pick Up"))
        {
            algo.gameObject.SetActive(false);
            count = count + 1;
        }
    }

    private void OnCollisionEnter2D(Collision2D algo)
    {
        if (algo.collider.tag == "Pared")
        {
            transform.localPosition = new Vector3(70, 3, 0);
        }
        if (algo.collider.tag == "Muerte")
        {
            camaraMuerte();
            Invoke("Inicio", 3);
        }
        if (algo.collider.tag == "Teleport")
        {
            transform.localPosition = new Vector3(112, 5, 0);
        }
    }

    void Inicio ()
    {
        transform.localPosition = new Vector3(-8, 3, 0);
        CamaraGameOver.SetActive(false);
    }

    void camaraMuerte ()
    {
        CamaraGameOver.SetActive(true);
        transform.localPosition = new Vector3(90, 3, 0);
    }
   }
```

❖ *Script ChargedParticle:* Carga los elementos del juego positiva o negativamente.

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

public class ChargedParticle : MonoBehaviour
{
    public float charged;
}
```

❖ *Script DrawSceneLine:* Hace que la plataforma móvil se mueva de un lado a otro.

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

public class DrawSceneLine : MonoBehaviour {

	public Transform from;
	public Transform to;

	void OnDrawGizmosSelected(){
		if (from != null && to != null){
			Gizmos.color = Color.cyan;
			Gizmos.DrawLine(from.position, to.position);
			Gizmos.DrawSphere(from.position, 0.15f);
			Gizmos.DrawSphere(to.position, 0.15f);
		}
	}
}
```

❖ *Script Capacitor:* Hace que la carga se atraída por los elementos que están cargados.

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

 public class Capacitor : MonoBehaviour
 {
	public float fieldValue = 1f;

	void OnTriggerStay (Collider other)
	{
		Debug.Log ("Collision!");

	  if (other.attachedRigidbody)
	  {
	      other.attachedRigidbody.AddForce(Vector2.right * fieldValue *
other.gameObject.GetComponent<ChargedParticle>().charged * Time.deltaTime);
	  }
	}
   }
```

❖ *Script CamaraController:* Hace que la cámara siga la carga.

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

public class CamaraController : MonoBehaviour {

    public GameObject player;
    private Vector3 offset;
	// Use this for initialization
	void Start () {
	  offset = transform.position - player.transform.position;
	}

	// Update is called once per frame
	void LateUpdate () {
	  transform.position = player.transform.position + offset;
	}
}
```

- *Script PlataformaMovil:* Hace que la plataforma se mueva de un punto elegido a otro elegido.

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

public class PlataformaMovil : MonoBehaviour {

	public Transform target;
	public float speed;

	private Vector3 start, end;

	// Use this for initialization
	void Start () {
		if(target != null) {
			target.parent = null;
			start = transform.position;
			end = target.position;
		}
	}

	// Update is called once per frame
	void Update () {

	}

	void FixedUpdate(){
		if (target != null) {
			float fixedSpeed = speed * Time.deltaTime;
			transform.position = Vector3.MoveTowards(transform.position,
target.position, fixedSpeed);
		}

		if (transform.position == target.position){
			target.position = (target.position == start) ? end : start;
		}
	}

}
```