TECNOLOGIA SERVIDORES Y BASES DE DATOS

Juego Unity con interacciones con una base de datos SQLite

CHRISTIAN ROMÁN GARCÍA

1. Descripción del Problema

He realizado con una librería de la unity asset store gratuita un juego 2D simple de plataformas, incluye al Player y a los enemigos además de un sistema de monedas que nos ayudara con la base de datos mas tarde. En el juego se interacciona con la base de datos usando la g y la r, que se indica al iniciar el nivel (tarda un momento). Al pulsar la g guardara la posición en la que estaba nuestro carácter, su vida y su cantidad de monedas. Cuando se presiona la R se devuelve la ultima fila de la tabla y restaura tanto la vida, las monedas y la posición con un transform.

La base de datos creada consta de una tabla datos con 4 columnas TEXT, la vida las monedas y el eje x e y del personaje (Se guarda en valores separados redondeados porque si no interfiere con la consulta al usar comas para distinguir los decimales).

Al ser una única tabla no se necesita un diagrama de entidad-relación.

2. Genera la base de datos

Base de datos del juego en formato csv:

```
Vida,Monedas,PosicionX,PosicionY
"5
",10,0,0
5,0,0,0
5,0,0,0
5,0,0,0
5,0,0,0
5,0,0,0
5,0,0,0
5,0,0,0
5,0,0,0
5,0,0,0
5,0,-58,-2
5,0,-38,-2
5,0,-38,-2
```

5,0,-38,-2

5,0,-58,-2

5,0,-37,-2

3,2,-67,-6

3,11,-5,-7

2,11,13,-9

3,0,-53,-4

3,0,-53,-144

3,0,-53,-111

3,0,-53,-106

3,0,-53,-108

3,0,-53,-108

3,0,-53,-115

3,0,-53,-916

5,0,-58,-2

5,0,-58,-2

5,0,-60,-2

5,0,-58,-2

5,0,-58,-2

5,0,-57,2

5,0,-67,-6

5,0,-67,-6

5,11,-25,-9

5,0,-58,-2

5,0,-58,-2

5,0,-58,-2

5,0,-58,-2

5,0,-67,-6

5,0,-59,-2

5,0,-63,-2

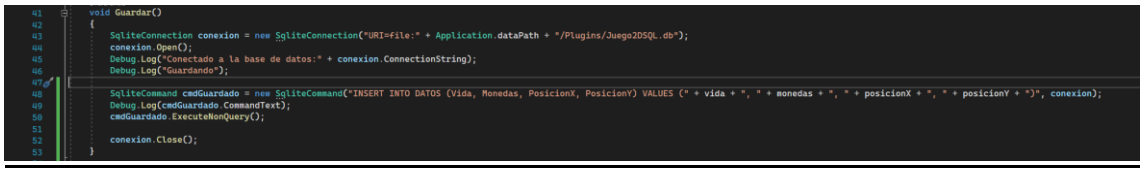
2. Desarrolla el código en Unity

Para el código reutilice el script de Prueba que nos mando crear para tener una sintaxis similar y ordenada, en mi script tenemos las funciones Guardar() y Restore() además de tener un Update() y un Start() que se ocupan de la lógica fuera de estas dos funciones. Además en el Start() del menú se realiza una consulta para obtener la ultima fila de la tabla DATOS y mostrarlo como el valor del ultimo guardado.

```
using UnityEngine;
using Mono.Data.SqliteClient;
using System.Data;
using System;

Script de Unity (1 referencia de recurso) | 0 referencias
public class Prueba : MonoBehaviour
{

    public string vida;
    public string vidaT;
    public string monedas;
    public string monedasT;
    public string posicionX;
    public string posicionXT;
    public string posicionY;
    public string posicionYT;
    NewPlayer player;
    Mensaje de Unity | 0 referencias
    void Start()
    {
        player = GameObject.Find("Player").GetComponent<NewPlayer>();

        IDbConnection conexion = new SqliteConnection("URI=file:" + Application.dataPath + "/Plugins/Juego2DSQL.db");
        conexion.Open();

        IDbCommand cmd = conexion.CreateCommand();
        cmd.CommandText = "SELECT * FROM DATOS";
        IDataReader datos = cmd.ExecuteReader();

        while (datos.Read())
        {
            string vida = datos.GetString(0);
            string monedas = datos.GetString(1);
            string posicionx = datos.GetString(2);
            string posiciony = datos.GetString(3);
            Debug.Log("Vida: " + vida + " Monedas: " + monedas + "PosicionX:" + posicionx + "PosicionY:" + posiciony);
        }
        conexion.Close();

        //También puedes utilizar esta otra nomenclatura si te resulta más cómodo...
    }
```

En esta parte del código del script Prueba vemos que estamos usando la librería Mono.Data.SqliteClient para realizar los métodos para conectarnos a la base de datos.

Después declaramos las variables que ayudaran con la lógica del programa. La función Start instancia de primeras al GameObject Player para obtener las variables de el componente NewPlayer, Luego establecemos la conexión con la base de datos dentro de la carpeta Plugins en Unity Assets y abrimos la conexión de la base de datos para leer las 4 columnas y escribir cada una de las filas en la consola de unity usando un debug, así comprobamos que está conectándose correctamente a la base de datos.

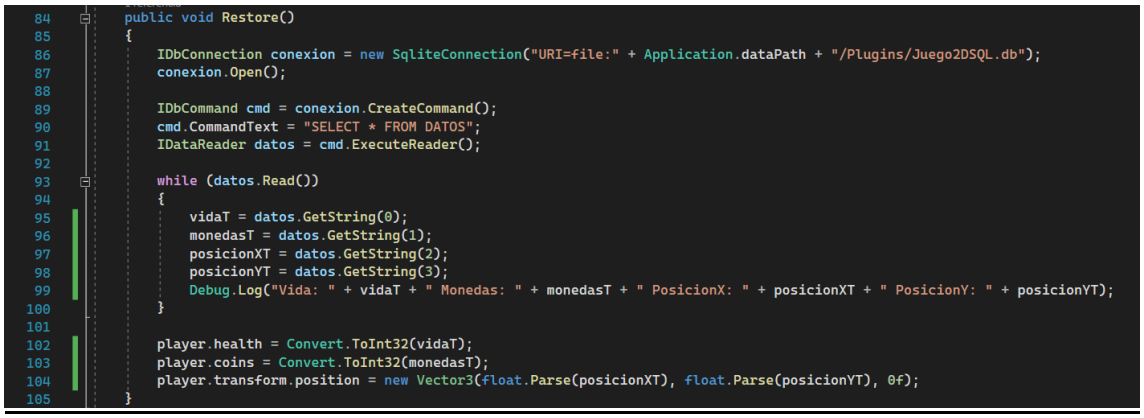


```
void Guardar()
{
    SqliteConnection conexion = new SqliteConnection("URI=file:" + Application.dataPath + "/Plugins/Juego2DSQL.db");
    conexion.Open();
    Debug.Log("Conectado a la base de datos:" + conexion.ConnectionString);
    Debug.Log("Guardando");

    SqliteCommand cmdGuardado = new SqliteCommand("INSERT INTO DATOS (Vida, Monedas, PosicionX, PosicionY) VALUES (" + vida + ", " + monedas + ", " + posicionX + ", " + posicionY + ")", conexion);
    Debug.Log(cmdGuardado.CommandText);
    cmdGuardado.ExecuteNonQuery();

    conexion.Close();
}
```

La función guardar realiza la conexión a la base de datos y ejecuta un INSERT INTO DATOS los valores mas actualizados de la vida las monedas y la posición del player.

```
private void Update()
{
    vida = Convert.ToString(player.health);
    monedas = Convert.ToString(player.coins);
    string postx = Convert.ToString(player.transform.position.x);
    string posty = Convert.ToString(player.transform.position.y);
    posicionX = Convert.ToString(Math.Round(float.Parse(postx)));
    posicionY = Convert.ToString(Math.Round(float.Parse(posty)));

    if (Input.GetKeyDown(KeyCode.G))
    {
        Guardar();
    }
    if (Input.GetKeyDown(KeyCode.R))
    {
        Restore();
    }
}
```

La función update es una función que se ejecuta en cada frame de juego, esta función comprueba la tecla pulsada para ejecutar la función de guardar o restaurar y además actualiza constantemente los valores a introducir en la base de datos con los del player.



```
public void Restore()
{
    IDbConnection conexion = new SqliteConnection("URI=file:" + Application.dataPath + "/Plugins/Juego2DSQL.db");
    conexion.Open();

    IDbCommand cmd = conexion.CreateCommand();
    cmd.CommandText = "SELECT * FROM DATOS";
    IDataReader datos = cmd.ExecuteReader();

    while (datos.Read())
    {
        vidaT = datos.GetString(0);
        monedasT = datos.GetString(1);
        posicionXT = datos.GetString(2);
        posicionYT = datos.GetString(3);
        Debug.Log("Vida: " + vidaT + " Monedas: " + monedasT + " PosicionX: " + posicionXT + " PosicionY: " + posicionYT);
    }

    player.health = Convert.ToInt32(vidaT);
    player.coins = Convert.ToInt32(monedasT);
    player.transform.position = new Vector3(float.Parse(posicionXT), float.Parse(posicionYT), 0f);
}
```

La función Restore realiza también un barrido por todas las filas de las 4 columnas para llegar a la última, una vez termine el while. Esto hace que el valor almacenado sea el ultimo que guardamos, por ello podemos aplicarse a las variables de Player para volver al punto anterior con la misma posición y estadística, por lo tanto, la vida perdida o cura se perderá y también las monedas. El script menú realiza lo mismo pero aplica los datos a 4 componentes text en el menú.

Aquí estaría el script Prueba:

```
using UnityEngine;
using Mono.Data.SqliteClient;
using System.Data;
using System;

public class Prueba : MonoBehaviour
{

    public string vida;
    public string vidaT;
    public string monedas;
    public string monedasT;
    public string posicionX;
    public string posicionXT;
    public string posicionY;
    public string posicionYT;
    NewPlayer player;
    void Start()
    {
        player = GameObject.Find("Player").GetComponent<NewPlayer>();

        IDbConnection conexion = new SqliteConnection("URI=file:" +
Application.dataPath + "/Plugins/Juego2DSQL.db");
        conexion.Open();

        IDbCommand cmd = conexion.CreateCommand();
        cmd.CommandText = "SELECT * FROM DATOS";
        IDataReader datos = cmd.ExecuteReader();

        while (datos.Read())
        {
            string vida = datos.GetString(0);
```

```
            string monedas = datos.GetString(1);
            string posicionx = datos.GetString(2);
            string posiciony = datos.GetString(3);
            Debug.Log("Vida: " + vida + " Monedas: " + monedas +
"PosicionX:" + posicionx + "PosicionY:" + posiciony);
        }
        conexion.Close();

        //También puedes utilizar esta otra nomenclatura si te resulta más
cómodo...
    }
    void Guardar()
    {
        SqliteConnection conexion = new SqliteConnection("URI=file:" +
Application.dataPath + "/Plugins/Juego2DSQL.db");
        conexion.Open();
        Debug.Log("Conectado a la base de datos:" +
conexion.ConnectionString);
        Debug.Log("Guardando");

        SqliteCommand cmdGuardado = new SqliteCommand("INSERT INTO DATOS
(Vida, Monedas, PosicionX, PosicionY) VALUES (" + vida + ", " + monedas + ",
" + posicionX + ", " + posicionY + ")", conexion);
        Debug.Log(cmdGuardado.CommandText);
        cmdGuardado.ExecuteNonQuery();

        conexion.Close();
    }

    //void TriggerAAAAA()
    //{
        //SqliteConnection conexion = new SqliteConnection("URI=file:" +
Application.dataPath + "/Plugins/Juego2DSQL.db");
        //conexion.Open();
        //string time = DateTime.Now.ToString();
        //Debug.Log("Guardado");
        //SqliteCommand MAMAAAAAAAAAA = new SqliteCommand("CREATE TRIGGER
save AFTER INSERT ON DATOS BEGIN INSERT INTO FECHA(Fecha) VALUES(" + time +
"); END; ");
        //MAMAAAAAAAAAA.ExecuteNonQuery();
        //Debug.Log("SUUUUUUUUUU");
        //conexion.Close();
    //}
    private void Update()
    {
        vida = Convert.ToString(player.health);
        monedas = Convert.ToString(player.coins);
        string postx = Convert.ToString(player.transform.position.x);
        string posty = Convert.ToString(player.transform.position.y);
        posicionX = Convert.ToString(Math.Round(float.Parse(postx)));
        posicionY = Convert.ToString(Math.Round(float.Parse(posty)));

        if (Input.GetKeyDown(KeyCode.G))
        {
            Guardar();
        }
        if (Input.GetKeyDown(KeyCode.R))
        {
            Restore();
        }
    }
    public void Restore()
    {
```

```
        IDbConnection conexion = new SqliteConnection("URI=file:" +
Application.dataPath + "/Plugins/Juego2DSQL.db");
        conexion.Open();

        IDbCommand cmd = conexion.CreateCommand();
        cmd.CommandText = "SELECT * FROM DATOS";
        IDataReader datos = cmd.ExecuteReader();

        while (datos.Read())
        {
            vidaT = datos.GetString(0);
            monedasT = datos.GetString(1);
            posicionXT = datos.GetString(2);
            posicionYT = datos.GetString(3);
            Debug.Log("Vida: " + vidaT + " Monedas: " + monedasT + "
PosicionX: " + posicionXT + " PosicionY: " + posicionYT);
        }

        player.health = Convert.ToInt32(vidaT);
        player.coins = Convert.ToInt32(monedasT);
        player.transform.position = new Vector3(float.Parse(posicionXT),
float.Parse(posicionYT), 0f);
    }
}
```