Universidad Nacional Autónoma de México

Facultad de Ingeniería

Estructuras de Datos y Algoritmos I

Proyecto final

¡¡¡Cuidado!!!

Ramírez Pérez Daniela Itzel

M.I. Marco Antonio Martínez Quintana

Semestre 2021-2

7/Agosto/2021

Proyecto Final

❖ RESUMEN

En el siguiente documento se expondrán las diferentes técnicas utilizadas para la creación del proyecto usando algunos de los conceptos aprendidos durante el semestre 2021-2 de la materia de Estructuras de Datos y Algoritmos.

El juego que se creó para el proyecto final tiene como objetivo dar un ejemplo de cómo funciona algunos de los conceptos de la estructura de Datos y en este caso se implementó la siguiente función: cuando se tienen datos almacenados se puede quitar el dato más reciente para insertar otro con cuando se obtiene una mejor puntuación o la comúnmente llamada "*High Score*" encontrada en muchos videojuegos y enseñárselo al jugador.

En mi documento también está presente todo el código que se referenció y utilizó para que mi juego guardara la puntuación máxima y el comportamiento de los objetos en el juego para alcanzar dicha puntuación.

También se hace una breve mención de los gastos que supuestamente ocurrirían durante el desarrollo del jugo, al utilizar softwares y materiales de uso libre del internet el proyecto no tuvo ningún gasto, pero si se puede considera para subirse a cualquier sitio que utilice el mismo método de muchas aplicaciones móviles de sólo poner anuncios dentro de la página en que hipotéticamente se subiría el juego para el disfrute de otros o también podría ser entre partidas del juego para tener una pequeña ganancia.

Por otra parte, también se desea señalar que por la simplicidad que tiene el juego teóricamente cualquiera puede personalizar y mejorarlo usando sus propias herramientas y de este modo hacer que su propio juego sea algo único en su categoría e incluso se puede profundizar el sistema de puntos que actualmente tiene este proyecto.

Cabe a destacar que la creación del juego no hubiera sido posible sin el uso del software gratis llamado "*Unity*", este programa es perfecto para cualquier principiante o experto con la intención de crear juegos 2D o 3D y fue gracias a ello que este proyecto fue posible con las diferentes herramientas y librerías que tiene para nuestras creaciones.

Así pues, hubo algunas dificultades al implantar ciertas propiedades del juego, con uno de los elementos más complicados e importante de implementar siendo el "*High score*", como se mencionó antes su implementación y uso fue muy importante al ser el objetivo principal del proyecto, pero se tuvo que consultar varias fuentes para darse una idea de cómo funcionaba el almacenamiento de información y como hacer que el jugador pudiera ver como sus acciones causaban los cambios en la información.

❖ INTRODUCCION

Mi proyecto fue diseñado con la idea de programar un juego del tipo “endless runner” que tiene como característica más notable, por como indica el nombre, ser un juego infinito donde el jugador debe evadir obstáculos por un tiempo ilimitado para conseguir la mejor puntuación que pueda.

Así pues, con un juego así se deben establecer parámetros a seguir para que se almacene correctamente los datos en los que el juego depende para que un jugador pueda obtener y conocer cual es su mejor puntuación de acuerdo con sus habilidades dentro del juego.

Un “endless runner” no necesariamente aumenta la dificultad para que un jugador siga avanzando, sino que la simplicidad del juego es lo que le permite que un jugador se empieza a cansar y al no estar atento a los obstáculos que aparecen aleatoriamente es lo que causa que pierda su partida.

Con la simplicidad de este juego cualquiera puede mejorar los parámetros de las que se hablaron previamente e incluso personalizarlos cuanto quiera, por ejemplo, en el “endless runner” el juego tiene como único objetivo que el jugador evite todos los objetos posibles hasta donde pueda llegar pero también se puede modificar para que tal vez la velocidad aumente cuando se llega a cierto punto o incluso subir la velocidad del jugador; como nos damos cuenta este tipo de juego depende por completo de cuanto tiempo se logró recorrer antes de que se tocara el obstáculo causando un “Game Over” inmediato y su modificación es relativamente fácil.

Por otro lado, el juego que se creó es similar a aplicaciones móviles ya que guardara el máximo puntaje de todos los jugadores en la computadora de cada jugador. En este caso se quiso probar como el juego al tener una nueva puntuación máxima o “High Score” automáticamente recordara la puntuación cada vez que se abra el juego y la actualizara cuando el jugador llega a superar su puntuación previa, probando así que el juego es capaz de almacenarla.

Cuando se almacenamos este tipo de información es asegurada nos permitimos asegurarle al jugador que no se pierde de nada de su progreso, con la única manera de borrarlo por completo es eliminar el juego por completo de su computadora y reinstalarlo.

Para concluir, poder manejar este tipo de información es muy importante ya que nos permite ver no sólo la forma en que videojuegos guardan cierta información, sino también darnos una idea de como otros programas utilizan sus propias librerías y almacenamiento para guardar la información necesaria y descartar la demás usada en sistemas de “Save” más complejos que usan los juegos modernos para guardar objetos, experiencia, numero de objetivos cumplidos, etc.

❖ DESARROLLO DEL PROYECTO

Descripción general del proyecto

Con este juego se tiene como objetivo mostrar a un jugador de manera creativa como sus acciones directas causan que el juego almacene un tipo específico de información para as así mostrar el progreso del jugador.

Algoritmo

PROBLEMA Obtener el mejor puntaje posible al evitar los obstáculos
ENTRADA Jugador
SALIDA Puntuación Máxima

1. El juego inicia
2. El jugador usa las flechas arriba y debajo de su teclado para evitar los obstáculos
3. El jugador avanza lo mas que pueda por el juego
4. El jugador choca con un obstáculo
5. El juego termina
6. Se inicia de nuevo el juego
7. En la parte de arriba se indica su puntuación máxima
8. El jugador repite el proceso hasta obtener la puntuación que quiera

- **Spawn Obstacle**

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

public class SpawnObstacles : MonoBehaviour
{
    public GameObject obstacle;
    public float maxX;
    public float minX;
    public float maxY;
    public float minY;
    public float timeBetweenSpawn;
    private float spawnTime;



    void Update()
    {
        //Indicamos el tiempo modificable en que aparecen los obstáculos
        if (Time.time > spawnTime)
        {
            Spawn();
            spawnTime = Time.time + timeBetweenSpawn;

        }

    }


    void Spawn()
    {
        // La distancia en que los obstáculos aparecen uno entre otro será   aleatorio dentro de los
parametros indicados
        float randomX = Random.Range(minX, maxX);
        float randomY = Random.Range(minY, maxY);

        Instantiate(obstacle, transform.position + new Vector3(randomX, randomY, 0),
transform.rotation);
    }

}
```

- **Score Manager**

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
public class ScoreManager : MonoBehaviour
{

    public Text scoreText;
    public Text highscoreText;


    int highscore = 0;

    private float score;

    void Start()

    {
      //Indicamos que es lo que quiero que el juego escriba al principio asi como poner el highscore en 0
        highscore = PlayerPrefs.GetInt("highscore", 0);
        scoreText.text = score.ToString() + " POINTS";
        highscoreText.text = "HIGHSCORE: " + highscore.ToString();
    }

    void Update()
    {
        //Indico como se le agrega un punto al jugador la mas distancuia que corre
        if (GameObject.FindGameObjectWithTag("Player") != null)
        {
            score += 1 * Time.deltaTime;
            scoreText.text =((int)score).ToString() + " POINTS";
        }
     //Indico como se cambia el highscore si es menor que el score que el jugador obtuvo
        if (highscore < score)
            PlayerPrefs.SetInt("highscore", ((int)score));
    }
}
```

- **Player**

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

public class Player : MonoBehaviour
{
    public float playerSpeed;
    private Rigidbody2D rb;
    private Vector2 playerDirection;


    void Start()
    {
          //Indico cual fisica quiero que siga el jugador
        rb = GetComponent<Rigidbody2D>();
    }


    void Update()
    {
        // Indico como quiero que el jugador solo se mueva en el eje Y
        float directionY = Input.GetAxisRaw("Vertical");
        playerDirection = new Vector2(0, directionY).normalized;
    }


    void FixedUpdate()

    { //Indico como la veloicdad del jugador la puedo cambiar a mi gusto
        rb.velocity = new Vector2(0, playerDirection.y * playerSpeed);
    }
}
```

- **Obstacle**

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

public class Obstacle : MonoBehaviour
{
    private GameObject player;

    //
    void Start()
    {
        player = GameObject.FindGameObjectWithTag("Player");
    }

    // Update is called once per frame
    private void OnTriggerEnter2D(Collider2D collision)
    {
        if (collision.tag == "Border")
        {
            //Indico como cualquier objeto que choque contra el borde invisible afuera del juego desaparece
            //para que asi no se haga lento el juego con cada obstaculo que aparece
            Destroy(this.gameObject);
        }
        else if (collision.tag == "Player")
        {
            //Indico como cualquier objeto que choque contra el jugador hara que el jugador sea destruido
            //y termine el juego

            Destroy(player.gameObject);
        }

    }

}
```

- **Menu**

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

public class Menu : MonoBehaviour
{

    public void PlayGame()
    {
        //Indico como al asignar un boton "Play Game" cargara la escena del juego
        SceneManager.LoadScene("Game");
    }

    public void mainMenu()
    {
        //Indico como al asignar un boton "MainMenu" cargara el menu principal
        SceneManager.LoadScene("MainMenu");
    }

    public void ExitButton() {
        //Indico como al asignar un boton "ExitButton" el juego cerrara
        Application.Quit();

    }
}
```

- **Looping Background**

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

public class LoopingBackground : MonoBehaviour
{
    public float backgroundSpeed;
    public Renderer backgroundRender;

    void Update()
    {
        //Indico como el arte en la parte de atras seguira apareciendo a una velocidad indicada
        backgroundRender.material.mainTextureOffset += new Vector2(backgroundSpeed * Time.deltaTime, 0f);
    }
}
```

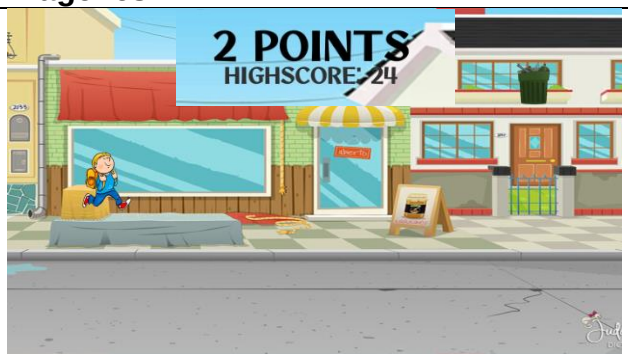
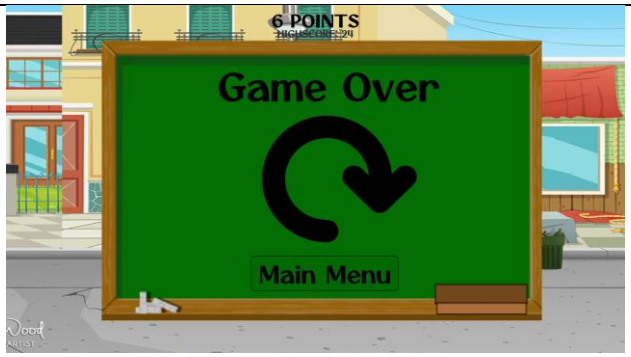
- **Game Over**

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

public class GameOver : MonoBehaviour
{
    public GameObject gameOverPanel;


    void Update()
    {
        if (GameObject.FindGameObjectWithTag("Player") == null)
        {
             //Indico como cuando el jugador colisiona con el obstcaulo un panel de "Game Over" aparece
            gameOverPanel.SetActive(true);
        }
    }

        public void Restart()
        {
            //Indico cuando un boton se le asigna "Restart" la escena del juego scarga otra vez
            SceneManager.LoadScene(SceneManager.GetActiveScene().name);
        }
    }
```

- **CameraMovement**

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

public class CameraMovement : MonoBehaviour
{


    public float cameraSpeed;


    void Update()
    {
        //Indica la velocidad modificable de la camara
        transform.position += new Vector3(cameraSpeed * Time.deltaTime, 0, 0);
    }


}
```

❖ **RESULTADOS DEL PROYECTO**

Imágenes	Descripción
 **Figura 1**	El juego inicia. Puntuación Maxima 24
 **Figura 2**	El juego da un “Game Over” y se da la opción de salir o reiniciar
 **Figura 3**	Se reinicia el juego

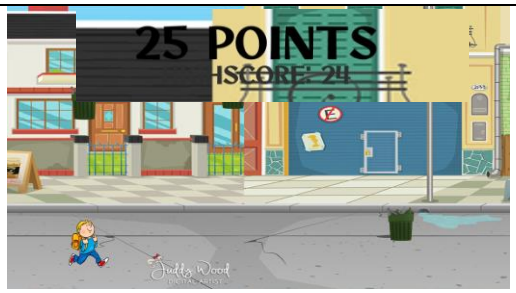
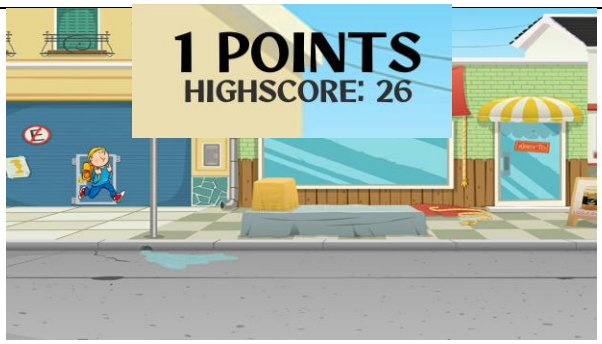
 **Figura 4.**	Es superada la puntuación máxima
 **Figura 5.**	Al reiniciar el nivel podemos observar cómo cambio la puntuación máxima de 24 a 26

Tabla 1. Software y Hardware utilizado

Unity 2020.3.14f1 (64-bit) Software de uso libre que permite la creación de juegos 2D y 3D	unity
Inno SetUp Compiler Compila los archivos de un programa para su fácil instalación en otra computadora	

Tabla 2. Costos propuestos

Software	“Assets”
Los costos se consideran cero por utilizar software de uso libre.	Utilice arte gratis del internet lo que se consideraría como cero gastos en el juego.
Lo más que ocurriría en torna mi propia ganancia sería subir mi juego a un sitio web con anuncios para ganar una cantidad mínima de 100 pesos	

Tabla 3. Diagrama de Gantt

Etapas	Julio 24- Agosto 1	Agosto 2 -Agosto 8
Desarrollo de Concepto		
Investigación de Código		
Creación de Materiales Gráficos		
Programación		
Refinamiento		

Repositorio GitHub

https://github.com/Dani4890UNAM/SegundoSemestre/blob/Proyecto-Final/%C2%A1%C2%A1%C2%A1Cuidado!!!-SetUp(x86).exe

Nota: Comprimí mi videojuego en una manera que se descarga lo necesario para correrlo en una computadora Windows.
No contiene programa maligno, aunque su computadora no quiera descargarlo por ser de origen desconocido, presionar en “Close and Run the program” para que no tenga dificultades.

Canal de YouTube

https://www.youtube.com/channel/UCnCkkWobpJGuWrNTf8BlmjQ/videos

❖ CONCLUSIONES

Con lo que se ha desarrollado durante este reporte se puede llegar a algunas conclusiones:

El desarrollo de programas con la capacidad de manejar el almacenamiento y cambio de todo tipo de información es muy importante hoy en día en la industria tecnológica por tener la gran ventaja de que el software guarde inmediatamente toda la información que ingresó el usuario y también por la capacidad que el software tiene de permitir cambiarla cuando se requiera. Por ejemplo, sitios como YouTube dependen enteramente de que sepan recopilar la información de sus usuarios para así proponerles otros medios de entretenimiento para mantener interés en sus sitios, así como guardar en su historial todos los videos que se vieron previamente. Con este ejemplo muchas industrias han permitido a los usuarios puedan decidir que información desean entregarle a un sitio para mejorar su experiencia en su sitio y así tener una ganancia gracias a las interacciones que reciben.

En mi opinión, con la conclusión de este proyecto, pude observar a primera mano como como puede operar la información que ingresa mi usuario y para darle a cambio entretenimiento, gracias a esto puedo entender como la estructura de datos me puede ayudar el manejo de esta información para usarla para el beneficio de quienes desean interaccionar con mi programa.

Para concluir, en comparación del semestre pasado, ahora tengo un mejor entendimiento de la organización de datos en un programa, así de como implementarlos en mi programa y con esta nueva información deseo aprender más como puedo crear mejores bases de datos para ayudar a sus aplicaciones en diferentes áreas.

❖ REFERENCIAS

- James Doyle [gamesplusjames]. (2019, 21 noviembre). Storing High Scores in Unity 2019 [Vídeo]. YouTube. https://www.youtube.com/watch?v=0zrZZN-QaDk

- DarrellL [Kapp – Game Dev]. (2021, 20 enero). HIGHSCORE & SCORE COUNTER + Saving Highscores in Unity | Easy Beginner Tutorial 2021 [Vídeo]. YouTube. https://www.youtube.com/watch?v=EJIDPo-nW-Q

- BMo. (2020, 13 septiembre). 5 Minute ENDLESS RUNNER Game UNITY Tutorial [Vídeo]. YouTube. https://www.youtube.com/watch?v=4YQVrs46f6k

- Maciej «Bionicl» Maj [Coco Code]. (2021, 19 enero). Points counter, HIGH SCORE and display UI in your game - Score points Unity tutorial [Vídeo]. YouTube. https://www.youtube.com/watch?v=YUcvy9PHeXs

- Hooson. (2021, 22 marzo). *How To Make A 2D Endless Runner For Beginners - Easy Unity Tutorial* [Vídeo]. YouTube. https://www.youtube.com/watch?v=U3sT-T5bKX4