UNIVERSIDAD TECNOLÓGICA DE SANTIAGO UTESA

SISTEMA CORPORATIVO

Presentado por

Marc-Edlyn Tertulien

Dan Nisset S. Jaques

Matricula

2 17 04 54

2 16 23 84

Presentado a

IVAN MENDOZA

Materia

Programación de Videojuegos

SANTIAGO DE LOS CABALLEROS, R.D

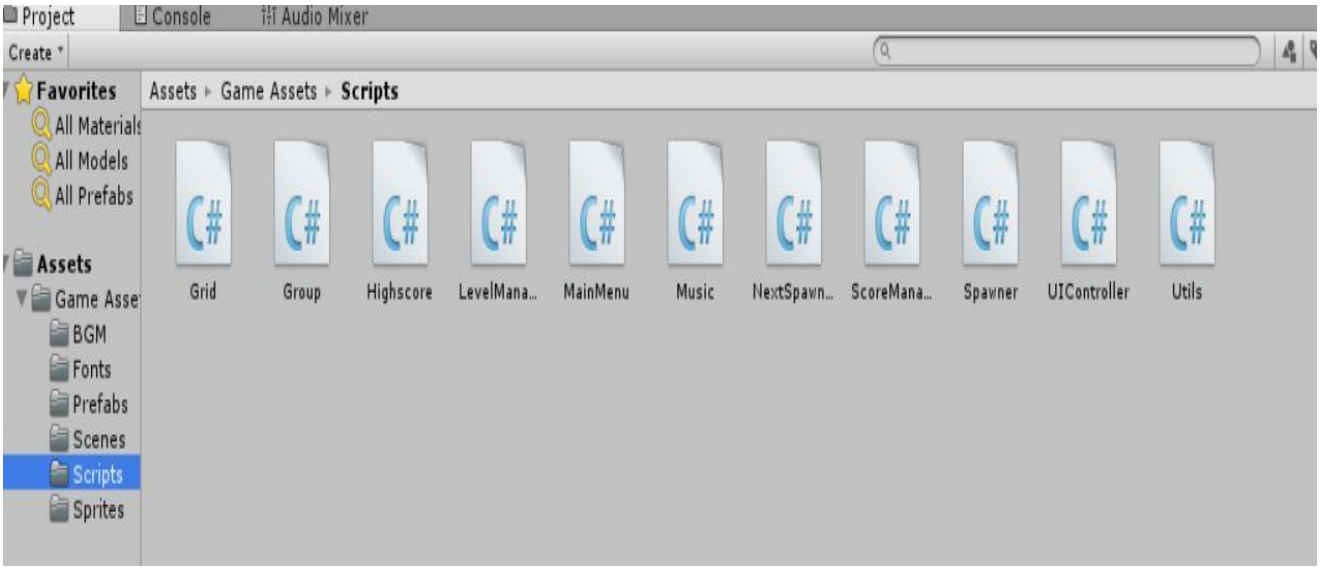
24 De Abril, 2022

CAPÍTULO III: DESARROLLO

3.1 Capturas de la Aplicación

(Hasta la página 22)

Scripts



Grid.cs

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



public class Grid : MonoBehaviour {

    
    public static int w = 10;
    public static int h = 20;

    public static Transform[,] grid = new Transform[w, h];


    public static Vector2 roundVector2(Vector2 v) {
        return new Vector2 (Mathf.Round (v.x), Mathf.Round (v.y));
    }


    public static bool insideBorder(Vector2 pos) {
        return ((int)pos.x >= 0 &&
                (int)pos.x < w &&
                (int)pos.y >= 0);
    }


    public static void deleteRow(int y) {
        for (int x = 0; x < w; x++) {
            Destroy(grid[x, y].gameObject);
            grid[x, y] = null;
        }
    }



    public static void decreaseRow(int y) {
        for (int x = 0; x < w; x++) {
            if (grid[x, y] != null) {

                grid[x, y - 1] = grid[x, y];
                grid[x,y] = null;


                grid[x, y-1].position += new Vector3(0, -1, 0);
            }
        }
    }

```

```
48      }
49
50
51      public static void decreaseRowAbove(int y) {
52          for (int i = y; i < h; i++) {
53              decreaseRow(i);
54          }
55      }
56
57
58      public static bool isRowFull(int y){
59          for (int x = 0; x < w; x++) {
60              if (grid[x, y] == null) {
61                  return false;
62              }
63          }
64          return true;
65
66      }
67
68      public static void deleteFullRows() {
69          for (int y = 0; y < h; y++) {
70              if (isRowFull(y)) {
71                  deleteRow(y);
72                  decreaseRowAbove(y + 1);
73
74                  ScoreManager.score += (h - y) * 10;
75                  --y;
76
77              }
78          }
79      }
```

Group.cs

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


public class Group : MonoBehaviour {


    private float lastFall;


    private float lastKeyDown;
    private float timeKeyPressed;

    public void AlignCenter() {
        transform.position += transform.position - Utils.Center(gameObject);
    }


    bool isValidGridPos() {
        foreach (Transform child in transform) {
            Vector2 v = Grid.roundVector2(child.position);

            // not inside Border?
            if(!Grid.insideBorder(v)) {
                return false;
            }

            // Block in grid cell (and not par of same group)?
            if (Grid.grid[(int)(v.x), (int)(v.y)] != null &&
                Grid.grid[(int)(v.x), (int)(v.y)].parent != transform) {
                return false;
            }
        }

        return true;
    }

    void updateGrid() {
        // Remove old children from grid
        for (int y = 0; y < Grid.h; ++y) {
            for (int x = 0; x < Grid.w; ++x) {
                if (Grid.grid[x,y] != null &&
                    Grid.grid[x,y].parent == transform) {
                    Grid.grid[x,y] = null;
                }
            }
        }

        insertOnGrid();
    }

    void insertOnGrid() {
        // add new children to grid
        foreach (Transform child in transform) {
            Vector2 v = Grid.roundVector2(child.position);
            Grid.grid[(int)v.x,(int)v.y] = child;
        }
    }

    void gameOver() {
        Debug.Log("GAME OVER!");
        while (!isValidGridPos()) {
```

```
        Debug.Log("GAME OVER!");
        while (!isValidGridPos()) {

            transform.position  += new Vector3(0, 1, 0);
        }
        updateGrid();
        enabled = false;
        UIController.gameOver();
        Highscore.Set(ScoreManager.score);

    }


    void Start () {
        lastFall = Time.time;
        lastKeyDown = Time.time;
        timeKeyPressed = Time.time;
        if (isValidGridPos()) {
            insertOnGrid();
        } else {
            Debug.Log("KILLED ON START");
            gameOver();
        }

    }

    void tryChangePos(Vector3 v) {

        transform.position += v;


        if (isValidGridPos()) {
            updateGrid();
        } else {
            transform.position -= v;
        }
    }

    void fallGroup() {
        // modify
        transform.position += new Vector3(0, -1, 0);

        if (isValidGridPos()){

            updateGrid();
        } else {

            transform.position += new Vector3(0, 1, 0);


            Grid.deleteFullRows();


            FindObjectOfType<Spawner>().spawnNext();


            // Disable script
            enabled = false;
        }

        lastFall = Time.time;

    }
```

```
        lastFall = Time.time;

    }


    bool getKey(KeyCode key) {
        bool keyDown = Input.GetKeyDown(key);
        bool pressed = Input.GetKey(key) && Time.time - lastKeyDown > 0.5f && Time.time - timeKeyPressed > 0.05f;

        if (keyDown) {
            lastKeyDown = Time.time;
        }
        if (pressed) {
            timeKeyPressed = Time.time;
        }

        return keyDown || pressed;
    }



    void Update () {
        if (UIController.isPaused) {
            return; // don't do nothing
        }
        if (getKey(KeyCode.LeftArrow)) {
            tryChangePos(new Vector3(-1, 0, 0));
        } else if (getKey(KeyCode.RightArrow)) {  // Move right
            tryChangePos(new Vector3(1, 0, 0));
        } else if (getKey(KeyCode.UpArrow) && gameObject.tag != "Cube") { // Rotate
            transform.Rotate(0, 0, -90);

            // see if valid
            if (isValidGridPos()) {

                updateGrid();
            } else {

                transform.Rotate(0, 0, 90);
            }
        } else if (getKey(KeyCode.DownArrow) || (Time.time - lastFall) >= (float)1 / Mathf.Sqrt(LevelManager.level)) {
            fallGroup();
        } else if (Input.GetKeyDown(KeyCode.Space)) {
            while (enabled) { // fall until the bottom
                fallGroup();
            }
        }

    }
}
```

Highscore.cs

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

public class Highscore : MonoBehaviour {

    public static int highscore = 0;

    public static void Set(int score) {
        if (score > highscore) {
            highscore = score;
        }

    }

    public static string Get() {
        return System.String.Format("{0:D8}", highscore);
    }

}

```

LevelManager.cs

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

public class LevelManager : MonoBehaviour {

    public static int level;

    Text levelText;

    // Use this for initialization
    void Awake () {
        levelText = GetComponent<Text>();
    }

    void Start() {
        level = 1;
    }

    // Update is called once per frame
    void Update () {
        if (ScoreManager.score >= level * 1000) {
            level += 1;
        }
        levelText.text = level.ToString();
    }
}

```

MainMenu.cs

```
1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4 using UnityEngine.SceneManagement;
5 using UnityEngine.UI;
6
7 public class MainMenu : MonoBehaviour {
8
9
10     public GameObject[] buttons;
11     public GameObject highScorePanel;
12     public Text highscoreText;
13     private int buttonSelected;
14     private int numberOfButtons;
15
16     void Awake() {
17         numberOfButtons = buttons.Length;
18         buttonSelected = 0;
19         SelectNewGame();
20         if (Highscore.highscore > 0) {
21             highscoreText.text = Highscore.Get();
22             highScorePanel.SetActive(true);
23         }
24     }
25
26     public void NewGame() {
27         SceneManager.LoadScene(1);
28     }
29
30     public void Exit() {
31         Application.Quit ();
32     }
33
34     void openSelected() {
35         if (buttonSelected == 0) {
36             NewGame();
37         } else if (buttonSelected == 1) {
38             Exit();
39         }
40     }
41
42     public void SelectNewGame() {
43         buttons[0].SetActive(true);
44         buttons[1].SetActive(false);
45         buttonSelected = 0;
46     }
47
48     public void SelectExitGame() {
49         buttons[1].SetActive(true);
```

```
43          buttons[0].SetActive(true);
44          buttons[1].SetActive(false);
45          buttonSelected = 0;
46      }
47
48      public void SelectExitGame() {
49          buttons[1].SetActive(true);
50          buttons[0].SetActive(false);
51          buttonSelected = 1;
52      }
53
54      void changePanel(int direction) {
55          buttons[buttonSelected].SetActive(false);
56
57          buttonSelected = Utils.Mod(buttonSelected + direction,  numberOfButtons);
58
59          buttons[buttonSelected].SetActive(true);
60      }
61
62      public void Update() {
63
64          if (Input.GetKeyDown(KeyCode.UpArrow)) {
65              changePanel(-1); // up
66          } else if (Input.GetKeyDown(KeyCode.DownArrow)) {
67              changePanel(1); // down
68          } else if (Input.GetKeyDown(KeyCode.Return)) {
69              openSelected(); // open selected button
70          } else if (Input.GetKeyDown(KeyCode.Escape)) {
71              Application.Quit(); // quit
72          }
73      }
74
75  }
76
```

Music.cs

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

public class Music : MonoBehaviour {
    public Object[] BGM;
    public AudioSource audioSource;
    public static AudioSource source;
    private bool tryingChange = false;
    public static int nextMusic;

    void Awake() {
        if (source == null) {
            source = audioSource;
            DontDestroyOnLoad(this);
            randomInitialization();
            source.Play();
        }
    }


    void randomInitialization() {
        nextMusic = Random.Range(0, BGM.Length);
        source.clip = BGM[nextMusic] as AudioClip;
    }


    void selectNextMusic(){
        source.clip = BGM[nextMusic] as AudioClip;
        nextCircularPlaylist();
    }


    void nextCircularPlaylist() {
        nextMusic = Utils.Mod(nextMusic + 1, BGM.Length);
    }


    void playNextMusic() {
        selectNextMusic();
        source.Play();
    }


    IEnumerator tryChange() {
        tryingChange = true;
        yield return new WaitForSeconds(1);
        if (!source.isPlaying) {
            playNextMusic();
        }
        tryingChange = false;
    }

    
    void Update () {
        if (!tryingChange && !source.isPlaying) {
            StartCoroutine(tryChange());
        }
    }
}

```

Nextspawner.cs

```
1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4
5 public class NextSpawner : MonoBehaviour {
6
7     private Spawner spawner;
8     private GameObject currentGroupObject;
9     private int currentGroupId;
10
11
12     void Awake () {
13         spawner = FindObjectOfType<Spawner>();
14     }
15
16
17     void createStoppedGroup () {
18         currentGroupObject = spawner.createGroup(transform.position);
19         currentGroupId = spawner.nextId;
20
21         var group = (Group) currentGroupObject.GetComponent(typeof(Group));
22
23         group.AlignCenter();
24         group.enabled = false;
25     }
26
27
28     void deleteCurrentGroup() {
29         Destroy(currentGroupObject);
30     }
31
32     void Start() {
33         createStoppedGroup();
34     }
35
36
37     void Update () {
38         if (currentGroupId != spawner.nextId) {
39             deleteCurrentGroup();
40             createStoppedGroup();
41         }
42     }
43 }
44
```

ScoreManager.cs

```
1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4 using UnityEngine.UI;
5
6 public class ScoreManager : MonoBehaviour {
7
8     public static int score;
9
10    Text scoreText;
11
12    |
13    void Awake () {
14        scoreText = GetComponent<Text>();
15    }
16
17    void Start() {
18        score = 0;
19    }
20
21
22    void Update () {
23        scoreText.text = System.String.Format("{0:D8}", score);
24    }
25 }
26
```

Spawner.cs

```
1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4
5 public class Spawner : MonoBehaviour {
6
7     public GameObject[] groups;
8     public int nextId;
9
10
11     void Start () {
12         nextId = Random.Range(0, groups.Length);
13         spawnNext ();
14     }
15
16
17     void Update () {
18
19     }
20
21     public GameObject createGroup(Vector3 v) {
22         GameObject group = Instantiate(groups[nextId], v, Quaternion.identity);
23
24         return group;
25     }
26
27
28     public void spawnNext() {
29
30         createGroup(transform.position);
31         nextId = Random.Range(0, groups.Length);
32     }
33 }
34
```

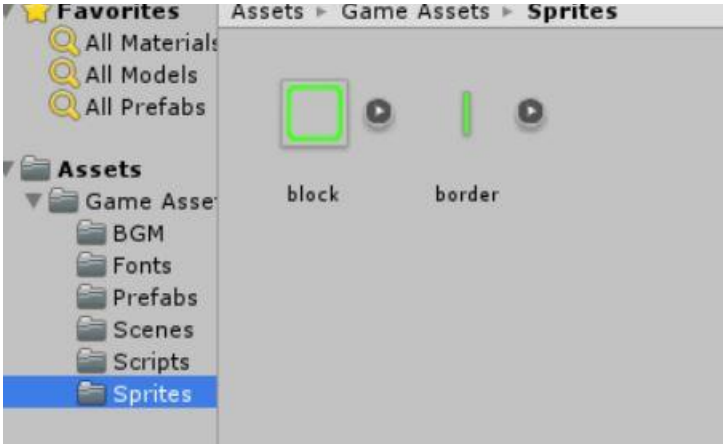
UiController.cs

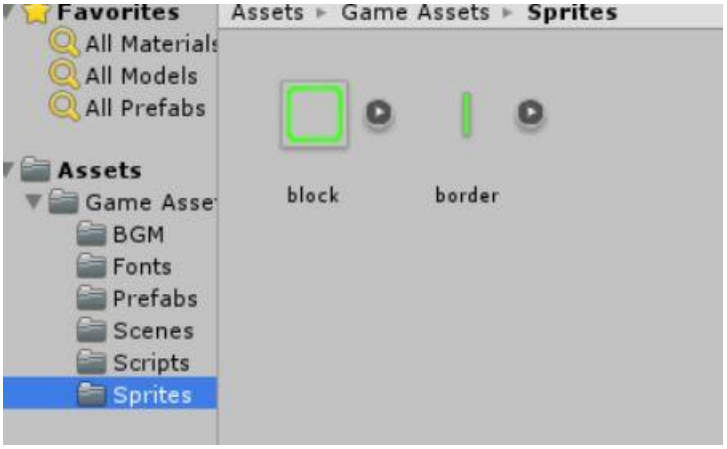
```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
using UnityEngine.SceneManagement;


public class UIController : MonoBehaviour {

    public GameObject gameOverPanel;
    public GameObject pausePanel;
    public static GameObject gameOverPanelStatic;
    public static float gameOverTime = 0;
    public static bool isPaused = false;


    void Awake() {
        if (gameOverPanelStatic == null) {
            gameOverPanelStatic = gameOverPanel;
        }
    }

    
    void restart() {
        Debug.Log("RESTART GAME!");
        SceneManager.LoadScene(SceneManager.GetActiveScene().name);
    }


    public static void gameOver() {
        gameOverPanelStatic.SetActive(true);
        gameOverTime = Time.time;
    }

    void gotoMainMenu() {
        SceneManager.LoadScene("MainMenu");
    }

    void togglePause() {
        Time.timeScale = Time.timeScale > 0 ? 0f : 1f;
        pausePanel.SetActive(!pausePanel.activeSelf);
        isPaused = !isPaused;
    }


    void Update () {

        // exit the game if presed ESC
        if (Input.GetKeyDown(KeyCode.Escape)) {
            Application.Quit();
        } else if (Input.GetKeyDown(KeyCode.P)) {
            togglePause();
        }
        else if (Input.GetKeyDown(KeyCode.R)) {
            restart();
        } else if (gameOverPanel.activeSelf && Input.anyKeyDown && Time.time - gameOverTime >= 0.5f) {
            //gameOverPanel.SetActive(false);
            gotoMainMenu();
        }
    }
}

```

Utils.cs

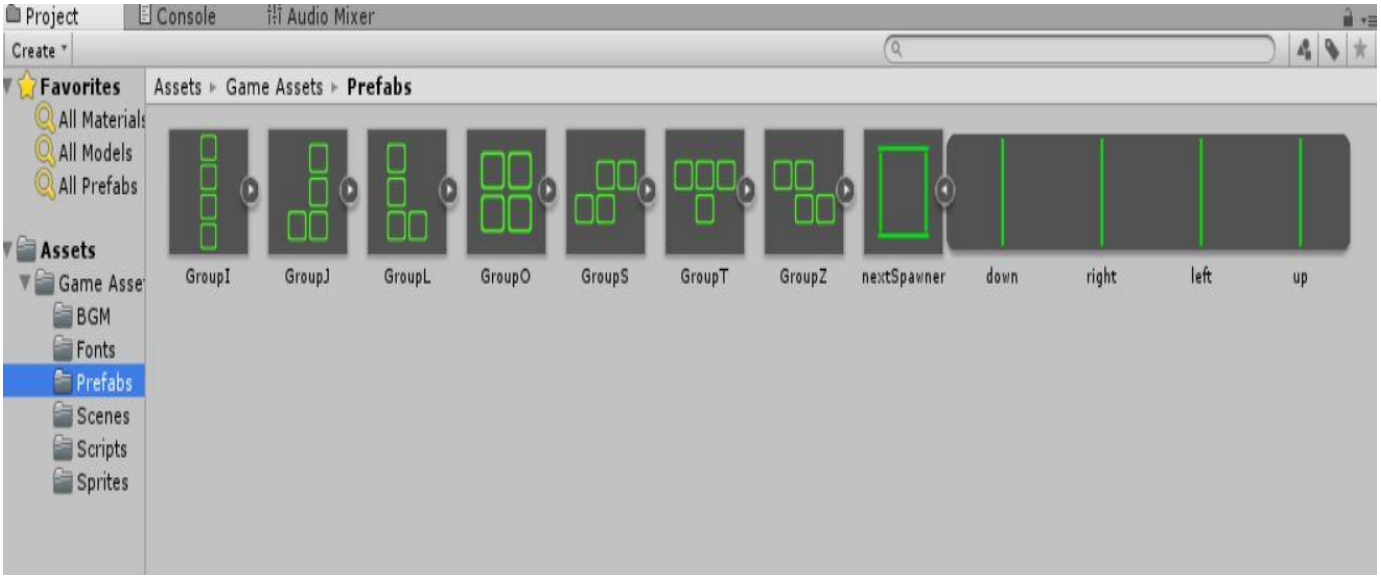
```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class Utils : MonoBehaviour {


    public static Bounds getRenderBounds(GameObject obj){
        var bounds = new  Bounds(Vector3.zero,Vector3.zero);
        var render = obj.GetComponent<Renderer>();
        return render != null? render.bounds : bounds;
    }


    public static Bounds getBounds(GameObject obj){
        Bounds bounds;
        Renderer childRender;
        bounds = getRenderBounds(obj);
        if((int)bounds.extents.x == 0){
            bounds = new Bounds(obj.transform.position,Vector3.zero);
            foreach (Transform child in obj.transform) {
                childRender = child.GetComponent<Renderer>();
                if (childRender) {
                    bounds.Encapsulate(childRender.bounds);
                }else{
                    bounds.Encapsulate(getBounds(child.gameObject));
                }
            }
        }
        return bounds;
    }

    // get the center of a gameobject
    public static Vector3 Center (GameObject obj) {
        return Utils.getBounds(obj).center;
    }


    public static int Mod (int n, int m){
        return ((n % m) + m) % m;
    }
}

```

Sprites


Favorites
All Materials
All Models
All Prefabs
Assets
Game Asse
BGM
Fonts
Prefabs
Scenes
Scripts
Sprites
Assets ▸ Game Assets ▸ Sprites
block
border

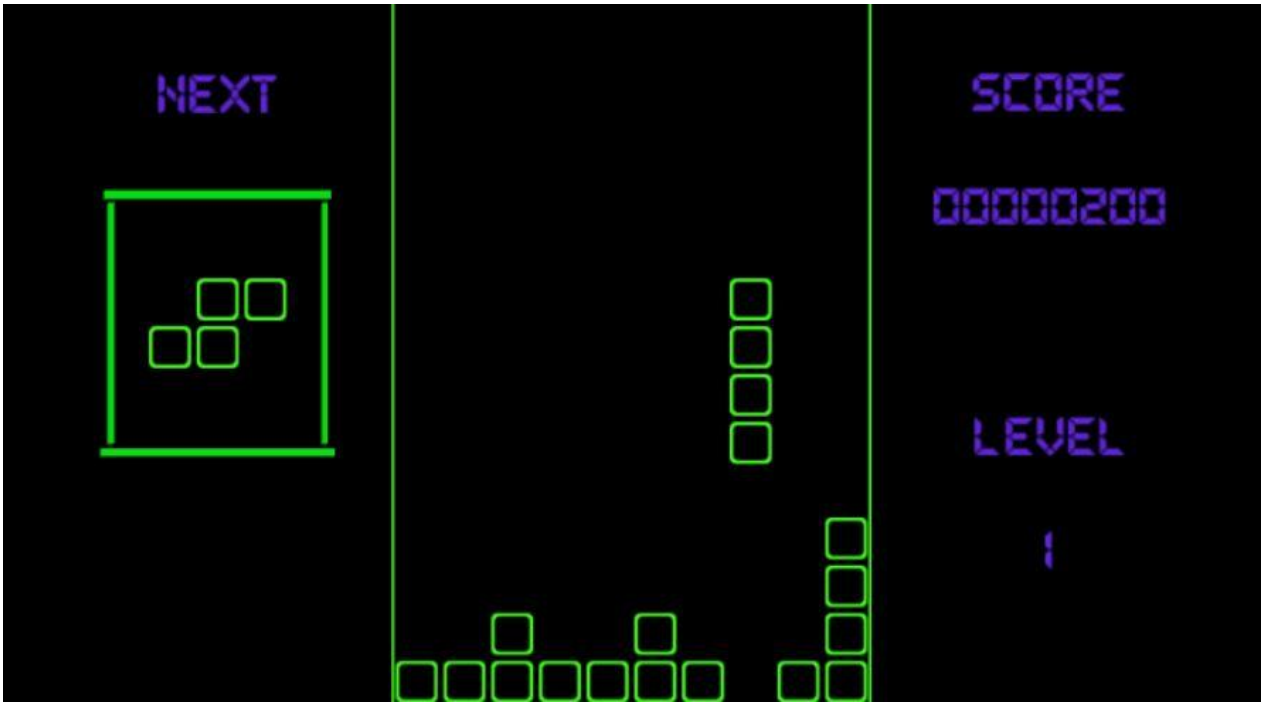

Favorites
All Materials
All Models
All Prefabs
Assets
Game Asse
BGM
Fonts
Prefabs
Scenes
Scripts
Sprites
Assets ▸ Game Assets ▸ Sprites
block
border

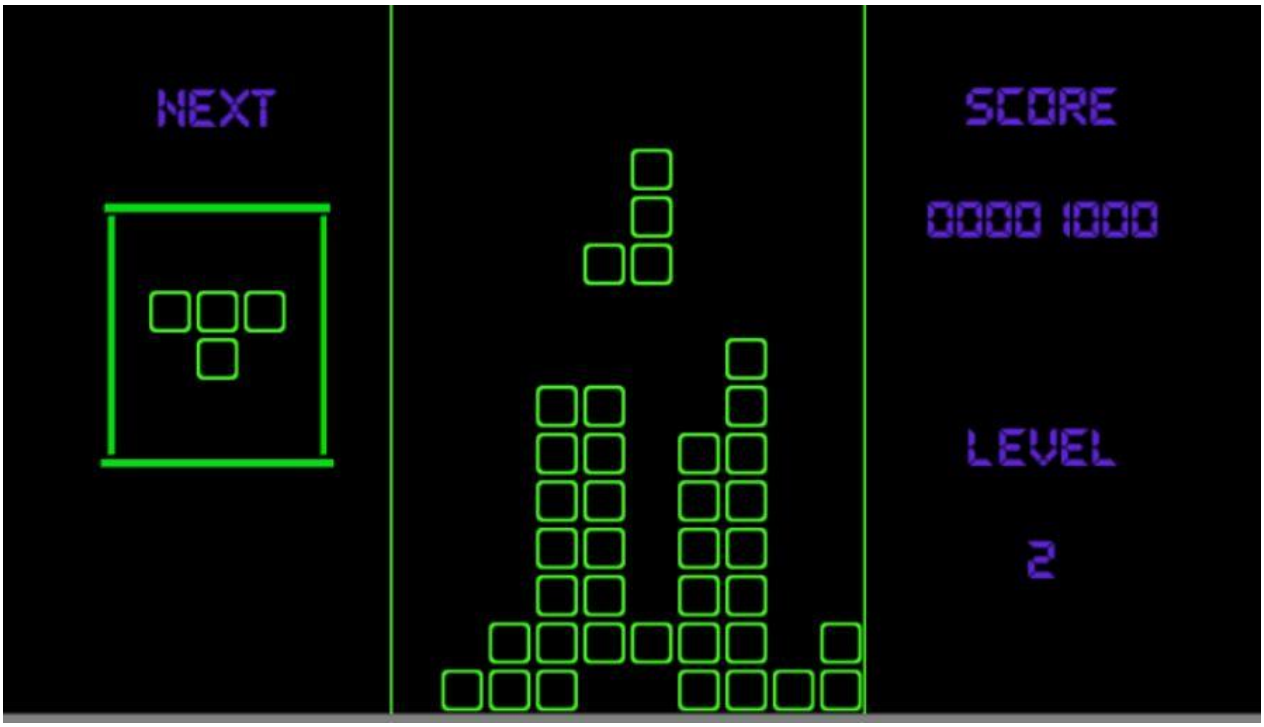
Prefabs



Imágenes


TETRIS GAME
NEW GAME
EXIT


NEXT
SCORE
00000200
LEVEL
1


NEXT
SCORE
0000 1000
LEVEL
2


NEXT
SCORE
GAME OVER
3


TETRIS GAME
› NEW GAME
EXIT
HIGHSCORE: 00002050

3.2 Prototipos

Los primeros prototipos a realizar serán de tipo Lo-Fi (prototipos de baja fidelidad). Luego de ir avanzando con el diseño, comenzaremos a realizar los prototipos de tipo Hi-Fi (prototipos de alta fidelidad).

Lo-Fi

- Primer prototipo con los primeros niveles y que se pueda mover la figura pero sin disposición de cambiar la estructura.
- Segundo prototipo con la figura capaz de cambiar su forma.
- Tercer prototipo con control del tiempo de la caída de las figuras.

Hi-Fi

- Primer prototipo con un nivel dinámico, un menú principal y el general
- Segundo prototipo con indicadores, control de los cambios de estructura, altura u profundidad ocupada y control de los sonidos del juego.

3.3 Perfiles de Usuarios

Los públicos objetivos del videojuego son:

- Para todo público realmente, al excepto de los niños menores de 4 años.
- Personas apasionadas a los juegos de tipo Árcade y Estrategia.
- Personas que por lo menos tengan acceso a computadoras para jugarlo.
- Persona que pueden manejar básicamente una PC, sea por recordar los botones y ya.

3.4 Usabilidad

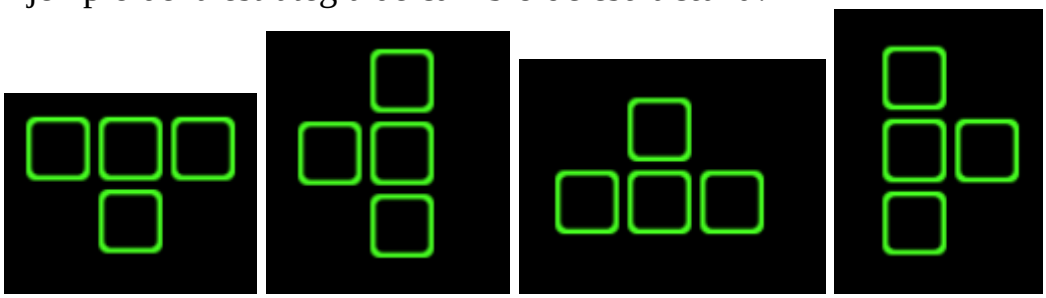
El juego en si posee una pantalla principal de inicio, y tras la escena por la cual está todo lo necesario para jugar y una pantalla final en donde se indica si el jugador perdió u otros Updates por su resultado.

La pantalla de inicio contiene un botón para iniciar el juego y es muy sencillo, de forma que el jugador puede empezar sin esperar un tiempo ni encontrar problema al momento que él desea iniciar el partidazo y también un botón de **Exit(Salir)** en caso de que el usuario quiere salir del juego (si tiene que ir para hacer sus tareas de preferencia y regresar ahorita a jugar, xD)

En la escena donde que se juega, el jugador tiene que usar como hardware las teclas de flechitas en su PC para controlar (mover, y, sobre todo cambiar la estructura de la figura presente, ósea hay figuras que tienen como 4 estructuras, vienen con una por defecto en la pantalla por lo tanto lo

podemos cambiar usando la tecla de **ARRIBA**) este punto de cambiar estructura también es algo estratégico que ayuda al jugador de posicionar u meter las figuras en forma más rentable y así que podría ganar más rápido o de forma controlada y también evitar de perder rápido porque a veces la estructura en la cual viene la figura es de una forma de desventaja entonces es al jugador de saber cuándo tiene que cambiar la estructura de la figura, y adema siempre es bueno de mover las figuras para ponerlas en las mejores posiciones, vienen por defecto al centro de la pantalla.

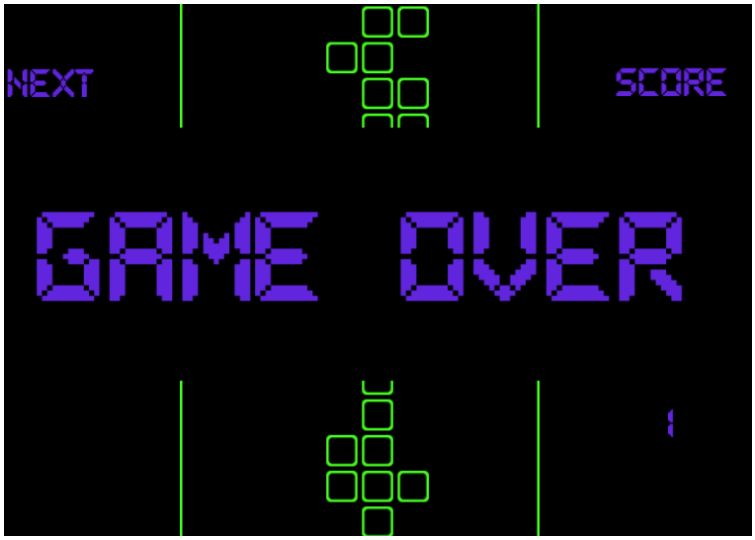
Ejemplo de la estrategia de **cambio de estructura**:



Observamos que en la primera estructura viene con la puntita hacia abajo, después de presionar el botón de **Arriba** la figura toma otra estructura y así sucesivamente.

Pegar figuras mientras deja un espacio vacío que contiene profundidad es algo estratégico que todo jugador debe saber, así crea su ganancia, así mismo vienen las victoria que aumentaran los puntos y avanzarse de Niveles. Aquellos usuarios que no saben crear espacio van a perder temprano porque las figuras van a ocupar todo la superficie de la pantalla de juego, verticalmente hablado y así su parte acabara rápido con un mensaje/indicación de **GAME OVER.**

Ejemplo:



3.5 Test

El desarrollo de este test es implementado de la siguiente manera:
Las personas encargadas de probas nuestros juegos se harán pasar por consumidores finales y puntuarán los diferentes aspectos del juego, imaginando que ese es el producto final. Teniendo en cuenta que debe ser una crítica constructiva y con total sinceridad.

Cada usuario deberá de responder dar su opinión sobre puntos claves del juego que nos permitirá tener una mejor visión de aquello en lo que se debe de mejorar para incrementar la aceptación del juego por los usuarios finales, entre estos puntos están:

- La jugabilidad
- El control del personaje
- Guía del usuario
- La información proporcionada por el juego
- Diseño visual
- Mecánicas del juego Para cada uno de estos puntos se deberá de dejar una puntuación y de ser posible algún comentario sobre el mismo, esto para mejorar la resolución de cualquier problema que presente el juego de manera más efectiva.

Notas entre 2 al 10

Version 1.0

Test 1

Individuo I

Rango de edad	20-25

Sexo	Femenino
Nivel de estudio	Univ./Ing. Mecánica
Pasa tiempo	Creación de materiales, Creatividad
Aficionado	Las Matemáticas

Resultado

Tareas	Puntuación	Comentario
Jugabilidad	8	Muy muy interesante
Dinamicidad	8	Cambio de estructura muy fácil
Guía del usuario	10	Bien expliacado
Mecánica del juego	6	Contiene lo necesario
Diseño Visual	8	"Mejores colores y figuras bien simples"

Individuo II

Rango de edad	15-20
Sexo	Femenino
Nivel de estudio	Univ./ Arte
Pasa tiempo	Dibujar
Aficionado	Diseños gráficos

Resultado

Tareas	Puntuación	Comentario
Jugabilidad	8	Interesante
Dinamicidad	10	Todo se controla con sencillez
Guía del usuario	8	-
Mecánica del juego	6	-
Diseño Visual	6	"Good"

Individuo III

Rango de edad	35-40
Sexo	Masculino
Nivel de estudio	Univ./Ing. Sistemas
Pasa tiempo	Programar y documentar
Aficionado	Las Matemáticas

Resultado

Tareas	Puntuación	Comentario
Jugabilidad	10	Muy muy interesante
Dinamicidad	8	Cambio de estructura muy fácil
Guía del usuario	8	Correcto
Mecánica del juego	5	Contiene realmente lo necesario
Diseño Visual	8	"colores atractivas y figuras bien diseñadas"

Como resultado frente a nuestra versión alfa, encontremos las siguientes medias.

Tareas	Media general
Jugabilidad	8.6
Dinamicidad	8.6
Guía del usuario	8.6
Mecánica del juego	7
Diseño Visual	7.3

Vimos que teníamos que mejorar la mecánica del juego y también el diseño visual sobre todo.

Versión 1.1

Notas entre 2 al 10

Test 2

Individuo I

Rango de edad	20-25
Sexo	Femenino
Nivel de estudio	Univ./Ing. Mecánica
Pasa tiempo	Creación de materiales, Creatividad
Aficionado	Las Matemáticas

Resultado

Tareas	Puntuación	Comentario
Jugabilidad	8	Muy muy interesante
Dinamicidad	8	Cambio de estructura muy fácil

Guía del usuario	10	Bien expliacado
Mecánica del juego	10	Muy elegante y pro
Diseño Visual	10	“Mejores colores y figuras bien simples”

Individuo II

Rango de edad	15-20
Sexo	Femenino
Nivel de estudio	Univ./ Arte
Pasa tiempo	Dibujar
Aficionado	Diseños gráficos

Resultado

Tareas	Puntuación	Comentario
Jugabilidad	8	Mas queiInteresante
Dinamicidad	10	Todo se controla con sencillez
Guía del usuario	10	-
Mecánica del juego	8	-
Diseño Visual	8	“Good”

Individuo III

Rango de edad	35-40
Sexo	Masculino
Nivel de estudio	Univ./Ing. Sistemas
Pasa tiempo	Programar y documentar
Aficionado	Las Matemáticas

Resultado

Tareas	Puntuación	Comentario
Jugabilidad	10	Muy muy interesante
Dinamicidad	8	Cambio de estructura muy fácil
Guía del usuario	10	Correcto
Mecánica del juego	9	Excelente
Diseño Visual	9	“colores atractivas y figuras bien diseñadas”

Como resultado frente a nuestra versión alfa, encontremos las siguientes medias.

Tareas	Media general
Jugabilidad	8.6
Dinamicidad	8.6
Guía del usuario	10
Mecánica del juego	9
Diseño Visual	9

3.6 Versiones de la Aplicación

Versión Alpha

Esta la primera versión funcional del videojuego. Solo tenia 1 nivel y fue para casi evaluar solamente el juego en nivel de desarrollador y el uso básico del mismo.

Versión Beta

Esta es una versión preliminar que aún es instable. Se añadió un nivel más y algunos power ups

Version 1.0

Versión en la cual vamos a publicarlo, contiene todo lo necesario y no tiene cantidad en nivel como la primera que tenía una sola y como un límite.

Versión 1.1

Versión lista para publicar próximamente, donde implementemos ya los cambios necesarios bajo sugestión de varios jugadores que han probado la primera versión.

Enlaces:
Danxjac/capitulo3GAMES · GitHub