

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

28 De Abril, 2022

INTRODUCCIÓN

Este documento cuenta y presenta de una forma estructurada un proyecto de un videojuego muy divertido, interesante cuyo nombre es TETRIS, es el resultado de los esfuerzos de los jóvenes ingenieros **Dan Nisset S. Jacques** y **Marc-Edlyn Tertulien**, quien planificaron y desarrollaron el TETRIS con el objetivo de brindar la felicidad a los apasionados de los videojuegos, donde ellos podrán jugar TETRIS sin pagar y de forma sencilla e interesante.

A todas personas que tienen este documento, les deseamos una buena lectura, este documento es el juego en sí, la representación y presentación perfecta del videojuego.

CAPÍTULO I: VIDEOJUEGO Y HERRAMIENTAS DE DESARROLLO

1.1 Descripción

Este es un juego arcade creado en la URSS en 1985. A través de figuras geométricas descenden una tras otra (rectángulo, cuadrado, triángulo, etc.), tienes que apilarlas para formar líneas continuas que irán desapareciendo y aumentando tu puntuación.

Hay diferentes versiones de tetris. Cada versión de tetris es diferente pero mantiene la base del juego. Nuestro TETRIS es simple y efectivo.

Tetris tuvo un gran éxito en los años 80 y sigue siendo muy popular hoy en día.

Funciona con reglas bastante simples: piezas de diferentes formas llamadas "tetromino", compuestas por cuatro pequeños cuadrados, aparecen en la parte superior de la interfaz y descenden gradualmente. El jugador debe apilarlas lo mejor posible para que encajen entre sí y, cuando formen una línea, desaparezcan para dejar espacio a otras piezas.

1.2 Motivación

El juego Tetris como siendo un juego muy viejo, a nosotros los integrantes de proyectos, ya esta muy familiarizado y tenemos muchas historias, muchos recuerdo frente a este bonito juego. Desde pequeno jugamos a Tetris, un juego bien interesante, bien simple con todos sus components bien estructurados.

Hemos tomado la decision de elegir este juego por ser mas comodo a nosotros, despues de pensar a otros juegos similares (por ser Old Games) como Snake, Rush, Super Mario, nuestros interes in Tetris son mayores, tuvimos la motivacion de recrear los que disfrutemos desde mucho tiempo que es este juego de Tetris, de forma que lo vivimos como jugadores y tambien desarrolladores.

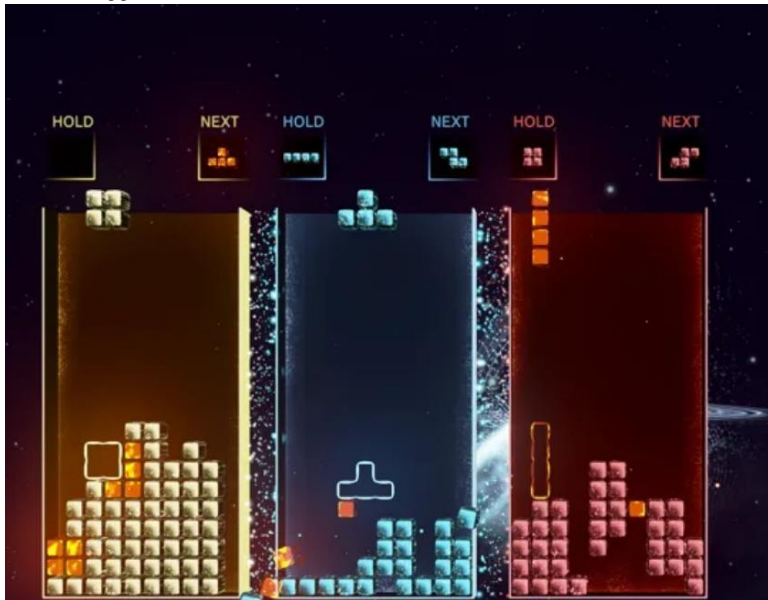
1.2.1 Originalidad de la idea

Como lo hemos mencionado en la parte de motivacion, vivir el tetris como jugador y desarrollador a la vez es algo bien chevere, entonces en adiccion tenemos idea como de dar vida otra vez al tetris en nuestra manera, traendo unas modificaciones, mejoramiento al juego de forma que podemos implementar una senzacion diferente del juego normal y tambien diferencia de diseno u modalidad.

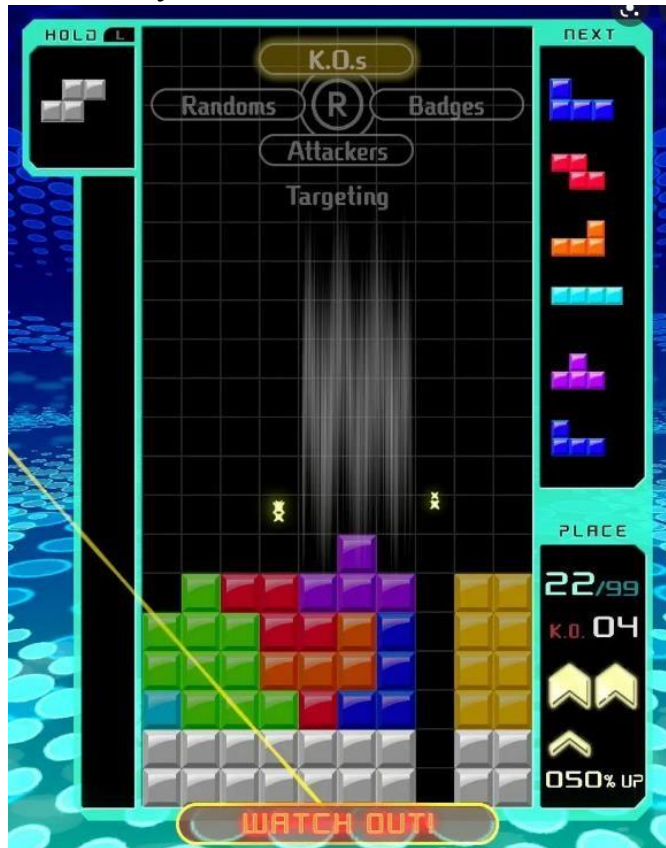
1.2.2 Estado del Arte

Mantenemos el estilo general del juego, principalmente en su parte de animacion que ya se queda al igual desde el tiempo viejo, muy dinamicas, de forma 2D con colores brillantes etc. realmente los mismos graficos a traves de los components del juego. Podemos sacar como ejemplo:

Tetris Effect Connected



Block Party Tetris



1.3 Objetivo general

- ✚ El objetivo del juego es evitar que los bloques se acumulen en la parte superior de la pantalla durante el mayor tiempo posible.

- ✚ Completar el nivel

1.4 Objetivos específicos

- ✚ Obtener las bonificaciones
- ✚ Hacer mayor puntuación
- ✚ Apile creativamente los bloques
- ✚ Elimine tantas líneas como sea posible con cada pieza que coloque

1.5 Escenario

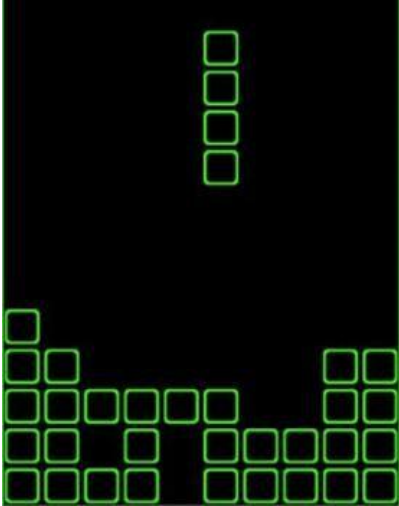
El juego esta atraves de un solo escenario, mediante lo cual encontramos los niveles, indicadores y todos los components del juego. Ubicado dentro de un espacio con diversas formas de bloques, velocidades sobre el movimiento de los bloques, algo que anima mucho el juego de forma de hacerlo mas interesante y tambien implementando dificultades al jugador a traves de estos cambios

(velocidad de los bloques, ya que la velocidad depende del nivel, y, las formas de los bloques)

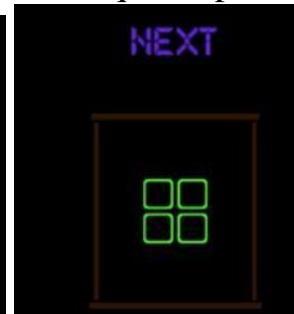
1.6 Contenidos

El juego se compone de:

Bloques: Son los materiales que el jugador usa para sacar su estrategia ubicando estos bloques en la mayor posicion possible a cada vez. Son de diversas formas...



Indicadores: Indica el punto actual del jugador en el escenario, su nivel actual y tambien indica al jugador sobre la forma del proximo bloque disponible.



Marcador: Presenta mensaje del estado del partido.
en este ejemplo es que el jugador ya pierde



1.7 Metodología

La metodología utilizada para el desarrollo del videojuego es la de SUM, ya que esta se ajustaba a las características de nuestro grupo de desarrollo y al tiempo en el que se desea realizar el proyecto. Las diferentes fases en las que se divide esta metodología de desarrollo son las siguientes:

Roles: Lo primero que se realiza en esta metodología son la definición de los Roles de cada miembro del equipo, distribuyéndonos las tareas de desarrollo, productor y probadores del juego, entre los miembros del equipo.

Ciclo de vida: Esta es la etapa en la que definimos toda la vida de desarrollo del videojuego, esta tiene diferentes fases en las cuales se va llevando a cabo el desarrollo del video juego, entre estas fases están:

Concepto: Definimos los aspectos del videojuego, con relación a los elementos que este contendrá, como por ejemplo las principales características, como debería ser el gameplay del juego, los niveles que tendría, como se desarrollaría el videojuego a lo largo del tiempo y también las herramientas que íbamos a utilizar, como lo fueron Unity 5 y VS Code.

Planificación: Aquí procedimos a definir el tiempo con el que contábamos y a realizar una división temporal entre las actividades que debíamos realizar, definiendo la importancia de cada una, para así elegir cuales tareas eran obligatorias de hacer y cuales más podía hacer el equipo en el tiempo que contábamos.

Elaboración: En esta etapa fue en la cual nos pusimos manos a la obra con el desarrollo de nuestro videojuego, con el objetivo de lograr una versión funcional del juego al finalizar esta etapa. Implementamos todas las ideas que fueron planeadas en etapas anteriores, respetando el espacio temporal de cada una y desarrollamos otras más de lo que se tenía pensado. Finalizando esta etapa también se realizó una evaluación en cuanto al estado en el que se encontraba nuestro videojuego.

Cierre: Esta fase se llevará a cabo como la entrega final del videojuego, se dará a cabo cuando los involucrados en el proyecto podamos llegar a un acuerdo entre todas las partes que consideren que el juego se encuentre en un buen estado para ser publicado y por supuesto antes del plazo temporal que se tiene.

Gestión de Riesgos: Esta fue una fase que se llevó a cabo junto con cada una de las demás fases, en especial en la de elaboración del videojuego, ya que esta contenía diferentes planes, para evitar que se complicara cualquiera de las diferentes fases, permitiéndonos llegar al final del videojuego en el tiempo establecido.

1.8 Arquitectura de la aplicación

La arquitectura utilizada para desarrollar este videojuego, se necesitaba que permitiera la reutilización de los objetos y que al mismo tiempo nos permitiera poder desarrollar el juego en un principio para la plataforma de Windows y Web, pero que en un futuro nos permita poder optimizar el juego para poder ejecutarlos en dispositivos móviles. Por lo que decidimos utilizar una Arquitectura por Capas, en la cual tenemos la capa principal conteniendo los manejadores del juego, del audio, de los niveles y de carga del juego, y luego tenemos capas inferiores en las que se encuentran alojados todos los aspectos UI del juego.

1.9 Herramientas de desarrollo

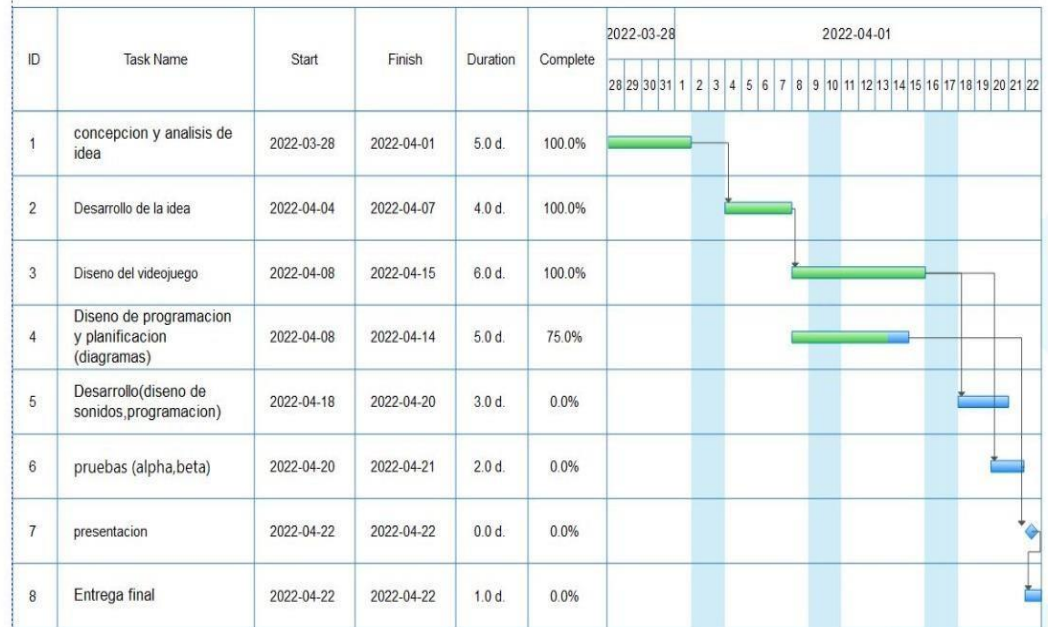
Las herramientas que utilizamos en el desarrollo de nuestro videojuego, están encabezadas por el motor de Unity 5, para desarrollar toda la parte con relación al juego. También utilizamos lo que viene siendo Visual Studio Code, como la herramienta de desarrollo de toda la parte del código necesario para el funcionamiento del videojuego.

CAPÍTULO II: DISEÑO E IMPLEMENTACIÓN

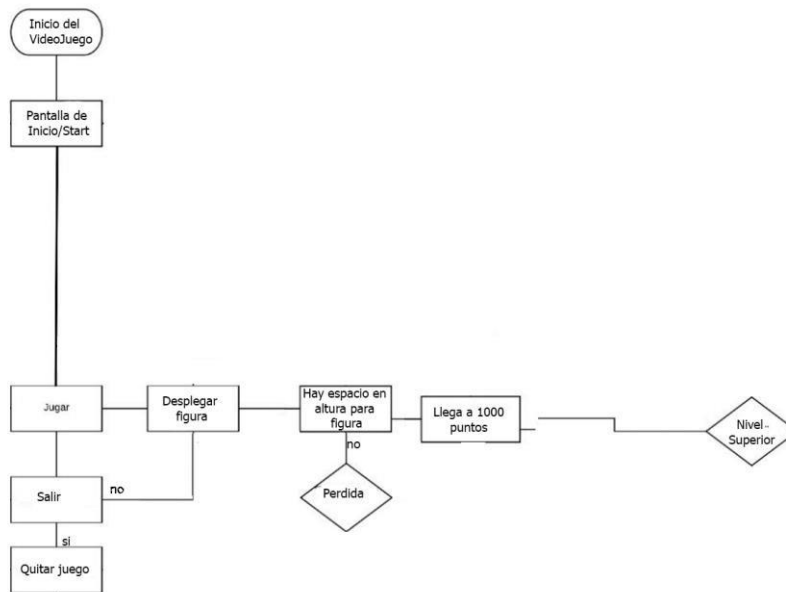
2.1 Planificación (Diagrama de Gantt)

Diagrama de gantt

proyecto videojuego



2.2 Diagramas y Casos de Uso



2.3 Plataforma

Windows, [Pc]

2.4 Género

Videojuego de emparejar fichas, falling block puzzle game, Videojuego de estrategia

2.5 Clasificación

Es un juego que no presentará un lenguaje no apropiado o algún tipo de violencia, por lo que este puede ser jugado por personas de todas las edades, lo que lo puede colocar en la clasificación para todas las edades o E

2.6 Tipo de Animación

Animación de tipo 2D

2.7 Equipo de Trabajo

Ingenieros de audio: Dan Nisset S. Jaques
Marc-Edlyn Tertulien

Diseñadores: Marc-Edlyn Tertulien
Dan Nisset S. Jaques

Ilustradores: Marc-Edlyn Tertulien
Dan Nisset S. Jaques

Programadores: Dan Nisset S. Jaques
Marc-Edlyn Tertulien

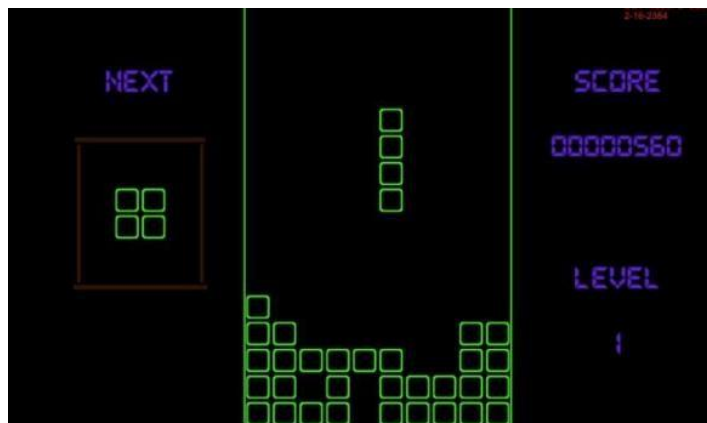
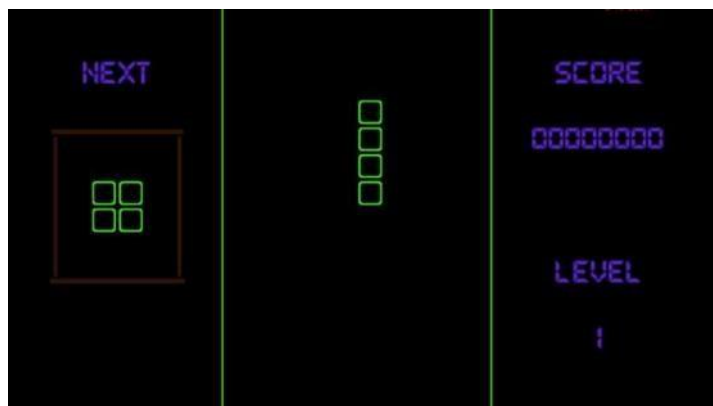
2.8 Historia

el videojuego “Tetris game” es un juego de lógica, es como otra versión del juego Tetris que fue lanzado en 1984 y fue disponible para casi cada consola de videojuegos y sistemas operativos de PC, en 2007, Tetris ocupó el segundo lugar en los 100 mejores videojuegos de todos los tiempos según IGN (Imagine Games Network) un sitio web estadounidense de videojuegos y medios de entretenimiento.

2.9 Guion

El videojuego no tiene un guion ya que no existen personajes, narrador o diálogos.

2.10 Storyboard





2.11 Personajes

Nuestro videojuego contiene objetos como las figuras geométricas que hemos entregado en el capítulo I, mientras que, no contiene personaje en verdad.

2.12 Niveles

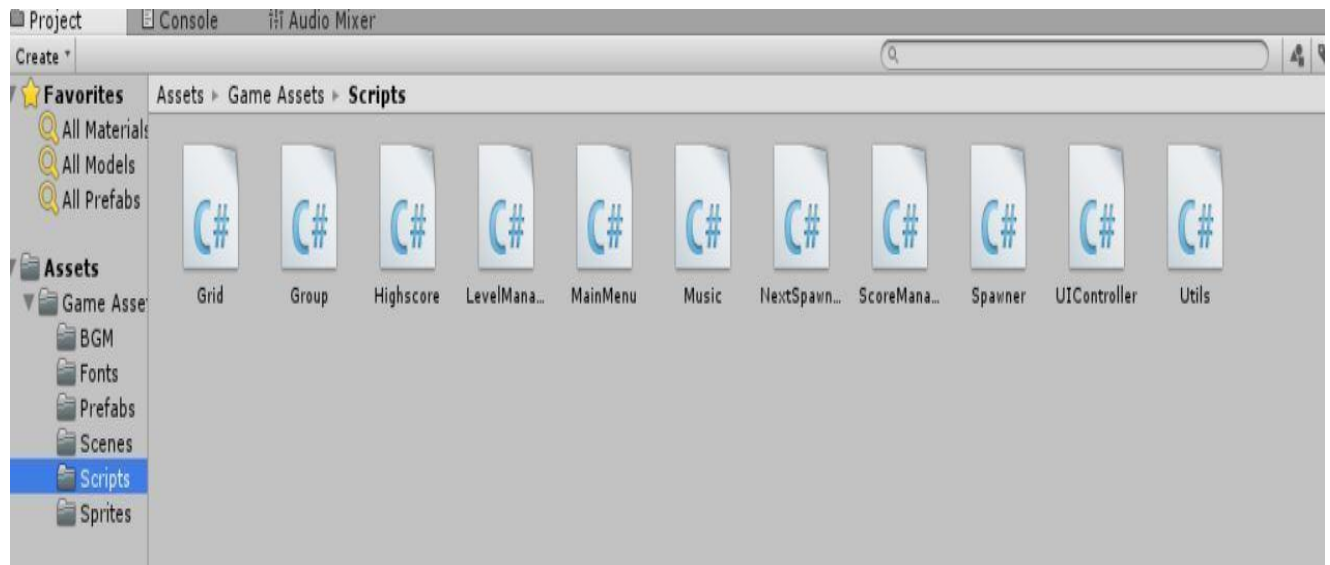
Los niveles se colocan dentro de un mismo escenario, de una forma dinámica donde a cada 1000 puntos del jugador, se avanza en Nivel Superior, Al entrar en un nivel superior, solo hay un cambio en la altura de las figuras que están en la pantalla.

2.13 Mecánica del Juego

Se usan figuras geométricas que se componen por cuatros bloques cuadrados unidos, puede decidir la rotación de la pieza y en qué lugar debe caer. Cuando una línea horizontal se completa, se desaparece y todas que están por encima descienden una posición y también el jugador gana puntos y a cada 1000 puntos se cambia de nivel del juego y la velocidad de las piezas aumentan.

CAPÍTULO III: DESARROLLO 3.1 Capturas de la Aplicación

Scripts



Grid.cs

```

1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4
5
6
7 public class Grid : MonoBehaviour {
8
9     |
10    public static int w = 10;
11    public static int h = 20;
12
13    public static Transform[,] grid = new Transform[w, h];
14
15
16    public static Vector2 roundVector2(Vector2 v) {
17        return new Vector2 (Mathf.Round (v.x), Mathf.Round (v.y));
18    }
19
20
21    public static bool insideBorder(Vector2 pos) {
22        return ((int)pos.x >= 0 &&
23                (int)pos.x < w &&
24                (int)pos.y >= 0);
25    }
26
27
28    public static void deleteRow(int y) {
29        for (int x = 0; x < w; x++) {
30            Destroy(grid[x, y].gameObject);
31            grid[x, y] = null;
32        }
33    }
34
35
36
37    public static void decreaseRow(int y) {
38        for (int x = 0; x < w; x++) {
39            if (grid[x, y] != null) {
40
41                grid[x, y - 1] = grid[x, y];
42                grid[x, y] = null;
43
44                grid[x, y-1].position += new Vector3(0, -1, 0);
45            }
46        }
47    }
48 }
49

```



```

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 }
80

```

Group.cs

```

1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4
5
6 public class Group : MonoBehaviour {
7
8
9     private float lastFall;
10
11
12     private float lastKeyDown;
13     private float timeKeyPressed;
14
15     public void AlignCenter() {
16         transform.position += transform.position - Utils.Center(gameObject);
17     }
18
19
20     bool isValidGridPos() {
21         foreach (Transform child in transform) {
22             Vector2 v = Grid.roundVector2(child.position);
23
24             // not inside Border?
25             if(!Grid.insideBorder(v)) {
26                 return false;
27             }
28
29             // Block in grid cell (and not par of same group)?
30             if (Grid.grid[(int)(v.x), (int)(v.y)] != null &&
31                 Grid.grid[(int)(v.x), (int)(v.y)].parent != transform) {
32                 return false;
33             }
34         }
35
36         return true;
37     }
38
39     void updateGrid() {
40         // Remove old children from grid
41         for (int y = 0; y < Grid.h; ++y) {
42             for (int x = 0; x < Grid.w; ++x) {
43                 if (Grid.grid[x,y] != null &&
44                     Grid.grid[x,y].parent == transform) {
45                     Grid.grid[x,y] = null;
46                 }
47             }
48         }
49
50         insertOnGrid();
51     }
52
53     void insertOnGrid() {
54         // add new children to grid
55         foreach (Transform child in transform) {
56             Vector2 v = Grid.roundVector2(child.position);
57             Grid.grid[(int)v.x,(int)v.y] = child;
58         }
59     }
60
61     void gameOver() {
62         Debug.Log("GAME OVER!");
63         while (!isValidGridPos()) {

```

```

62     Debug.Log("GAME OVER!");
63     while (!isValidGridPos()) {
64
65         transform.position += new Vector3(0, 1, 0);
66     }
67     updateGrid();
68     enabled = false;
69     UIController.gameOver();
70     Highscore.Set(ScoreManager.score);
71
72 }
73
74
75 void Start () {
76     lastFall = Time.time;
77     lastKeyDown = Time.time;
78     timeKeyPressed = Time.time;
79     if (isValidGridPos()) {
80         insertOnGrid();
81     } else {
82         Debug.Log("KILLED ON START");
83         gameOver();
84     }
85
86 }
87
88 void tryChangePos(Vector3 v) {
89
90     transform.position += v;
91
92
93     if (isValidGridPos()) {
94         updateGrid();
95     } else {
96         transform.position -= v;
97     }
98 }
99
100 void fallGroup() {
101     // modify
102     transform.position += new Vector3(0, -1, 0);
103
104     if (isValidGridPos()){
105         updateGrid();
106     } else {
107         transform.position += new Vector3(0, 1, 0);
108
109
110         Grid.deleteFullRows();
111
112         FindObjectOfType<Spawner>().spawnNext();
113
114         // Disable script
115         enabled = false;
116     }
117
118     lastFall = Time.time;
119
120 }
121
122
123
124

```

```

122     lastFall = Time.time;
123 }
124 }
125
126
127 bool getKey(KeyCode key) {
128     bool keyDown = Input.GetKeyDown(key);
129     bool pressed = Input.GetKey(key) && Time.time - lastKeyDown > 0.5f && Time.time - timeKeyPressed > 0.05f;
130
131     if (keyDown) {
132         lastKeyDown = Time.time;
133     }
134     if (pressed) {
135         timeKeyPressed = Time.time;
136     }
137
138     return keyDown || pressed;
139 }
140
141
142
143 void Update () {
144     if (UIController.isPaused) {
145         return; // don't do nothing
146     }
147     if (getKey(KeyCode.LeftArrow)) {
148         tryChangePos(new Vector3(-1, 0, 0));
149     } else if (getKey(KeyCode.RightArrow)) { // Move right
150         tryChangePos(new Vector3(1, 0, 0));
151     } else if (getKey(KeyCode.UpArrow) && gameObject.tag != "Cube") { // Rotate
152         transform.Rotate(0, 0, -90);
153
154         // see if valid
155         if (isValidGridPos()) {
156             updateGrid();
157         } else {
158             transform.Rotate(0, 0, 90);
159         }
160     } else if (getKey(KeyCode.DownArrow) || (Time.time - lastFall) >= (float)1 / Mathf.Sqrt(LevelManager.level)) {
161         fallGroup();
162     } else if (Input.GetKeyDown(KeyCode.Space)) {
163         while (enabled) { // fall until the bottom
164             fallGroup();
165         }
166     }
167 }
168
169 }
170 }
171 }
172

```

Highscore.cs

```
1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4 using UnityEngine.UI;
5
6 public class Highscore : MonoBehaviour {
7
8     public static int highscore = 0;
9
10    public static void Set(int score) {
11        if (score > highscore) {
12            highscore = score;
13        }
14    }
15
16    public static string Get() {
17        return System.String.Format("{0:D8}", highscore);
18    }
19 }
20
21 }
22
```

LevelManager.cs

```
1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4 using UnityEngine.UI;
5
6 public class LevelManager : MonoBehaviour {
7
8     public static int level;
9
10    Text levelText;
11
12    // Use this for initialization
13    void Awake () {
14        levelText = GetComponent<Text>();
15    }
16
17    void Start() {
18        level = 1;
19    }
20
21    // Update is called once per frame
22    void Update () {
23        if (ScoreManager.score >= level * 1000) {
24            level += 1;
25        }
26        levelText.text = level.ToString();
27    }
28 }
29
```

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

```
1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4
5 public class Music : MonoBehaviour {
6     public Object[] BGM;
7     public AudioSource audioSource;
8     public static AudioSource source;
9     private bool tryingChange = false;
10    public static int nextMusic;
11
12    void Awake() {
13        if (source == null) {
14            source = audioSource;
15            DontDestroyOnLoad(this);
16            randomInitialization();
17            source.Play();
18        }
19    }
20
21
22    void randomInitialization() {
23        nextMusic = Random.Range(0, BGM.Length);
24        source.clip = BGM[nextMusic] as AudioClip;
25    }
26
27
28    void selectNextMusic(){
29        source.clip = BGM[nextMusic] as AudioClip;
30        nextCircularPlaylist();
31    }
32
33
34    void nextCircularPlaylist() {
35        nextMusic = Utils.Mod(nextMusic + 1, BGM.Length);
36    }
37
38
39    void playNextMusic() {
40        selectNextMusic();
41        source.Play();
42    }
43
44
45    IEnumerator tryChange() {
46        tryingChange = true;
47        yield return new WaitForSeconds(1);
48        if (!source.isPlaying) {
49            playNextMusic();
50        }
51        tryingChange = false;
52    }
53
54    |
55    void Update () {
56        if (!tryingChange && !source.isPlaying) {
57            StartCoroutine(tryChange());
58        }
59    }
60 }
61
```

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
```

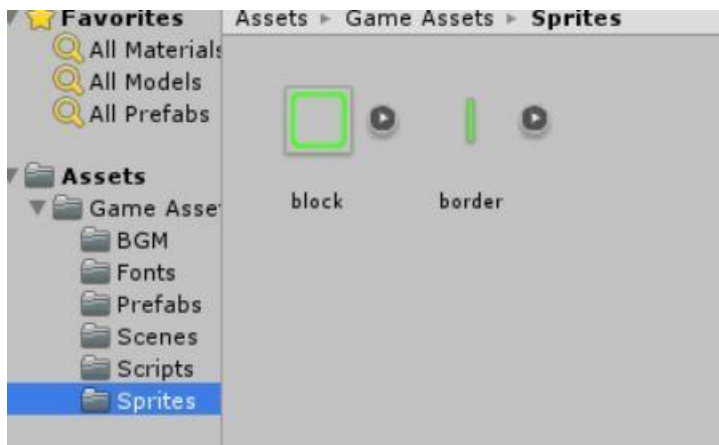
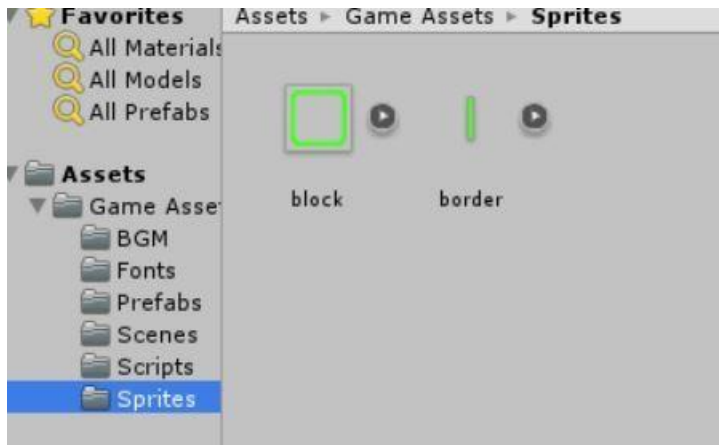
UiController.cs

```
1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4 using UnityEngine.UI;
5 using UnityEngine.SceneManagement;
6
7
8 public class UiController : MonoBehaviour {
9
10     public GameObject gameOverPanel;
11     public GameObject pausePanel;
12     public static GameObject gameOverPanelStatic;
13     public static float gameOverTime = 0;
14     public static bool isPaused = false;
15
16     void Awake() {
17         if (gameOverPanelStatic == null) {
18             gameOverPanelStatic = gameOverPanel;
19         }
20     }
21
22
23
24     void restart() {
25         Debug.Log("RESTART GAME!");
26         SceneManager.LoadScene(SceneManager.GetActiveScene().name);
27     }
28
29
30     public static void gameOver() {
31         gameOverPanelStatic.SetActive(true);
32         gameOverTime = Time.time;
33     }
34
35     void gotoMainMenu() {
36         SceneManager.LoadScene("MainMenu");
37     }
38
39     void togglePause() {
40         Time.timeScale = Time.timeScale > 0 ? 0f : 1f;
41         pausePanel.SetActive(!pausePanel.activeSelf);
42         isPaused = !isPaused;
43     }
44
45
46     void Update () {
47
48         // exit the game if presed ESC
49         if (Input.GetKeyDown(KeyCode.Escape)) {
50             Application.Quit();
51         } else if (Input.GetKeyDown(KeyCode.P)) {
52             togglePause();
53         }
54         else if (Input.GetKeyDown(KeyCode.R)) {
55             restart();
56         } else if (gameOverPanel.activeSelf && Input.anyKeyDown && Time.time - gameOverTime >= 0.5f) {
57             //gameOverPanel.SetActive(false);
58             gotoMainMenu();
59         }
60     }
61 }
62
```

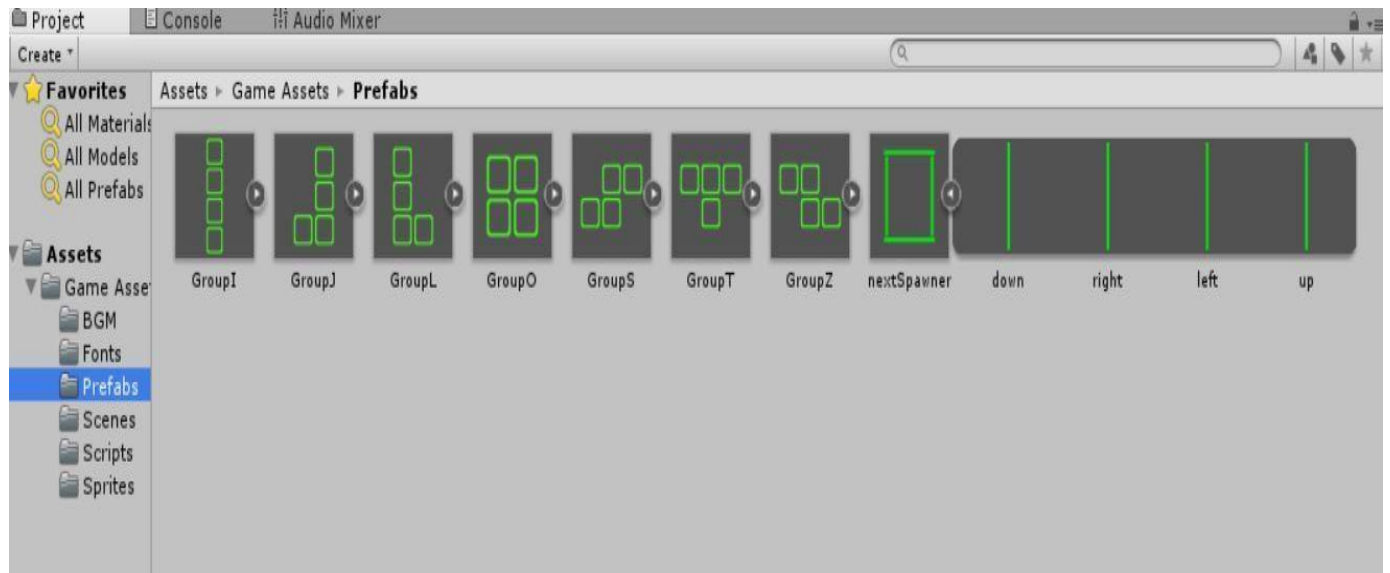

Utils.cs

```
1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4
5 public class Utils : MonoBehaviour {
6
7
8     public static Bounds getRenderBounds(GameObject obj){
9         var bounds = new Bounds(Vector3.zero, Vector3.zero);
10        var render = obj.GetComponent<Renderer>();
11        return render != null? render.bounds : bounds;
12    }
13
14
15    public static Bounds getBounds(GameObject obj){
16        Bounds bounds;
17        Renderer childRender;
18        bounds = getRenderBounds(obj);
19        if((int)bounds.extents.x == 0){
20            bounds = new Bounds(obj.transform.position, Vector3.zero);
21            foreach (Transform child in obj.transform) {
22                childRender = child.GetComponent<Renderer>();
23                if (childRender) {
24                    bounds.Encapsulate(childRender.bounds);
25                }else{
26                    bounds.Encapsulate(getBounds(child.gameObject));
27                }
28            }
29        }
30        return bounds;
31    }
32
33    // get the center of a gameobject
34    public static Vector3 Center (GameObject obj) {
35        return Utils.getBounds(obj).center;
36    }
37
38 }
39
40 public static int Mod (int n, int m){
41     return ((n % m) + m) % m;
42 }
43 }
```

Sprites



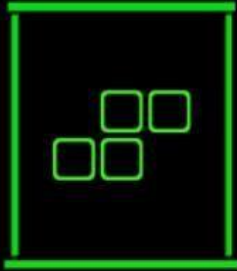
Prefabs



Imágenes



NEXT

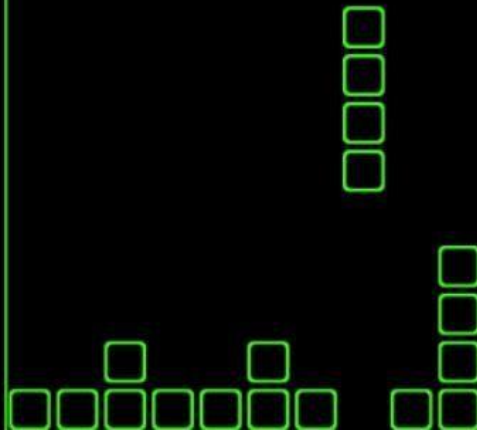


SCORE

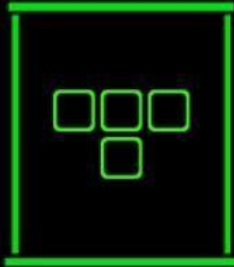
00000200

LEVEL

1



NEXT

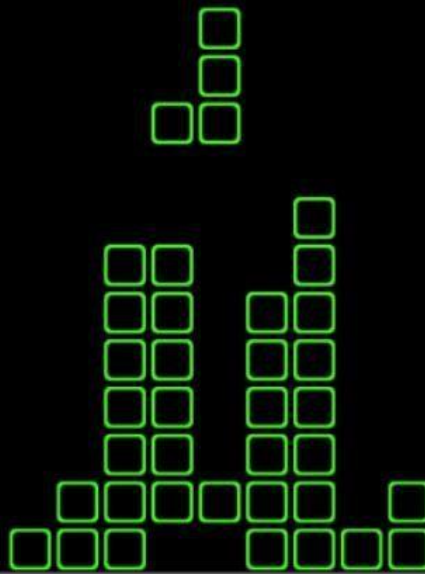


SCORE

0000 1000

LEVEL

2





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.

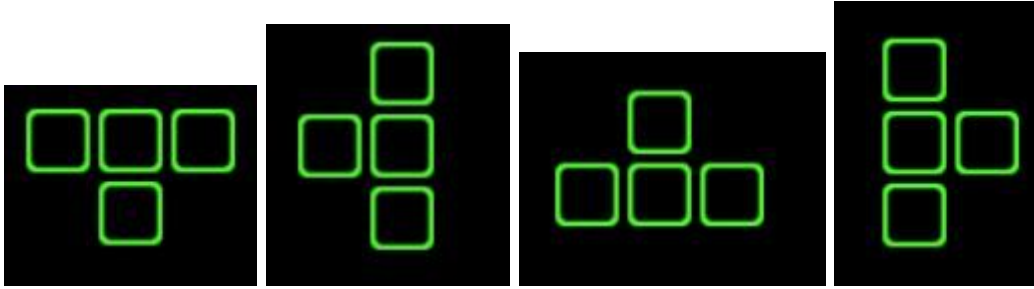
3.4 Usabilidad

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 probar 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 explicado
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 explicado
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 tenía 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 inestable. Se añadió un nivel más y algunos powerups

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 sugerencia de varios jugadores que han probado la primera versión.

CAPÍTULO IV: PUBLICACIÓN






4.1 Requisitos de la instalación

Mínimo

 Sistema Operativo	Windows 7, 8 u 8.1, 10, 11	32/64 bit
 Procesador	1 GB Ram	
 Disco Duro	0.6 GB	
 Tarjeta de video	AMD Radeon HD 5670 u NVIDIA GeForce GTS 450	
 Tarjeta de sonido	Tarjeta de sonido compatible con DirectX con los controladores más recientes	



Recomendado

 Sistema Operativo	Windows 7, 8 u 8.1, 10, 11	64 bit
 Procesador	2 GB Ram	
 Disco Duro	1 GB	
 Tarjeta de video	AMD Radeon HD 5680 u NVIDIA GeForce GTS 460	
 Tarjeta de sonido	Tarjeta de sonido compatible con DirectX con los controladores más recientes	

4.2 Instrucciones de Uso

En este videojuego, se maneja figuras las cuales vienen desde la parte arriba de la pantalla para caer o llegar en un espacio de abajo más o menos menor de donde vienen (eso depende también de la profundidad de las figuras ya llegadas), podemos cambiarlas de posición (+Derecha o +Izquierda), también la estructura de la figura, algo estratégico que existe para facilitar los jugadores y también motivarles en el juego para sacar su mejor en lo que tiene que ver con analizar e implementar su idea estratégica.

Instrucción:

Tecas de dirección:

Derecha	→	Mover la figura en una posición más a la derecha
Tecla Izquierda	←	Mover la figura en una posición más a la izquierda
Tecla de Arriba	↑	Cambiar la estructura/forma de la figura
Tecla de Abajo	↓	Hace que la figura baja más rápida

4.3 Bugs

El juego no tiene bugs de forma profunda, si el jugador tiene lo mejor en requisitos, no vaya tener bugs mediante problema funcional que a veces tienen que ver con los hardware...

La velocidad de la caída de las figuras puede variar a un 0.0034% y también cuando llega la figura, puede moverlo hasta después de un 0.015 segundo.

4.4 Proyección a Futuro

Actualmente, el videojuego está bien por lo que cumplimos y que los jugadores van a disfrutar durante este año del 2022 hasta el mes de Septiembre, después de este dicho mes, tendremos que empezar con los trabajos con el fin de bridar el videojuego primero en una disponibilidad para varias plataformas, de forma que los jugadores no solo van a poder jugarlo en Windows, si no que en Mac, en Nitendo y sobre todo en los Smartphone, que sea Android u iPhone. De esa forma, haremos una distribución nueva del videojuego mediante más sitio web y también a través de los Appstore como el de Google (**Google Playstore**) de los iPhone (**App Store**), en sitios populares como **Softonic**, **gameOceans** entre otros hasta 10 sitios web.

No vaya tener muchas diferencias de interfaces, tampoco de control porque nuestro videojuego realmente es algo global, entonces tendremos que, mediante los trabajos, modifican las resoluciones de pantalla, algunos sonidos y la lucidez también.

4.5 Presupuesto

Tarea	Tiempo/dias	Cuesta (\$)	Testing/dias
<i>Documentación</i>	120	14 400	30
<i>Sonido</i>	60	1 200	15
<i>SFX</i>	30	700	15
<i>Programación</i>	90	9 720	20
<i>Dibujo</i>	50	4 000	20

<i>Diseño</i>	60	4 800	25
<i>Equipaje</i>	30	7 000	10
<i>Marketing</i>	40	3 000	20
<i>Arte #1</i>	30	5 000	40
<i>Arte #2</i>	20	2 500	20
<i>Gastos fijos</i>	65	6 500	-
<i>Total</i>	120 (el máximo)	58 820	30 (el máximo)

4.6 Análisis de Mercado

Antes de todo, El informe de **NewZoo** indica que en el mundo, los gamers gastaron un 1.4% más que en 2020, y que las mayores compras se dieron en mobile gaming. Entre el 2020 y el 2021 el sector gaming se volvió en uno de los más relevantes y con menores afectaciones a causa de COVID-19, pues al estar los usuarios en casa se incrementó el número de gamers en el mundo, además del gasto que se hizo dentro de aplicaciones o títulos. De acuerdo con un reporte de la firma de análisis NewZoo, en el 2021 los ingresos que tuvo este sector fueron de 180,300 millones de dólares, un 1.4% más de lo que tuvo en 2020.

“Comenzamos el año pensando que sería imposible que el mercado igualara a 2020, ya que los retrasos en los lanzamientos de AAA y la escasez de suministro continuo para nuevo hardware de juegos dieron a las personas pocas oportunidades de gastar”, dijo Tom Wijman, jefe de Informes y Pronósticos de NewZoo. “Sin embargo, nueve meses después de 2021, nos dimos cuenta de que el mercado volvería a crecer. En todo caso, esto muestra que el mercado de los juegos definitivamente ya no es el negocio estacional impulsado por los éxitos que alguna vez fue”.

El año pasado ha sido particularmente... beneficioso para el mercado de los videojuegos. Impulsado por el lanzamiento de nuevas consolas con un éxito deslumbrante (como la PS5 de Sony, que ya se agotó para pedidos anticipados antes de Navidad), el auge de los juegos en la nube y muchos productos de software nuevos, el sector logró un desempeño histórico. La facturación batió su récord en 2020 en 5.300 millones de euros con un crecimiento del +11,3% respecto a los datos de 2019

Estos son los principales países más adictos a los videojuegos en 2020:

- 🚩 China (el 90% de los adultos admite jugarlo regularmente, casi el 70% de ellos en su teléfono inteligente)
- 🚩 Arabia Saudita (muchos jugadores a pesar de que la ley islámica restringe ciertos juegos)
- 🚩 Tailandia (ahora se permiten competencias de videojuegos allí y los casos de adicción están explotando con un promedio de 35 horas dedicadas a los videojuegos por persona por semana)
- 🚩 Estados Unidos

- ✚ Inglaterra
- ✚ Vietnam (con un dominio de los juegos de teléfonos inteligentes, el 80% de los adultos se entregan a ellos)
- ✚ Alemania
- ✚ Francia
- ✚ España
- ✚ Colombia

4.7 Viabilidad

Este videojuego se puede ser ejecutado vía computadora con sistema operativo de Windows 7, 8/8.1, 10,11, y sería gratis hasta la próxima versión y también lo podremos jugar mediante Mac y también dispositivo móvil.

CONCLUSIONES

Un videojuego muy interesante y seductivo, así podemos definir u calificar el resultado de nuestro proyecto que empecemos haces unos meses donde llegamos al fin de esta etapa. Como lo hemos definido antes, los videojuegos nos llevan en una atmosfera donde podemos disfrutar la pasión de jugar mediante nuestros dispositivos en un mundo imaginario.

A nosotros (Marc-Edlyn Tertulien y Dan Nisset S. Jacques) concluimos que disfrutemos el trabajo, los momentos de planificación y de labor de forma que ahora los jugadores pueden disfrutar este producto que realicemos.

REFERENCIAS BIBLIOGRÁFICAS

LINK GitHub:

<https://github.com/Danxjac/proyectoFINALvideojuego.git>
[marc-edlyn509/FinalProjectVideojuego \(github.com\)](https://github.com/marc-edlyn509/FinalProjectVideojuego)

Link publicación videojuego itch.io:

<https://danxjac.itch.io/tetriscgames>
[Tetris Proj by marcedlyn509 \(itch.io\)](https://danxjac.itch.io/tetriscgames)

[Imgur: The magic of the Internet](#)

[El mercado mundial de videojuegos - Mapas de El Orden Mundial - EOM](#)

[3 formas de encontrar fallas en un juego de video \(wikihow.com\)](#)

[La barba rolera | Blog de juegos de rol: Mecánicas desde cero: Cómo buscar y encontrar bugs](#)

