GRADO SUPERIOR EN DESARROLLO DE APLICACIONES MULTIPLATAFORMAS (DAM)

PROYECTO FINAL (Videojuego Unity)

CRISTIAN SABALL MAURO

CURSO: 2019/2020
FECHA: 12/06/2020

ÍNDICE

I. OBJETIVOS DEL PROYECTO **4**
II. MÓDULOS FORMATIVOS APLICADOS EN EL TRABAJO **6**
1. PROGRAMACIÓN 6
2. ENTORNOS DE DESARROLLO 6
3. LENGUAJE DE MARCAS 6
4. DESARROLLO DE INTERFACES 7
5. INGLÉS 7
6. PROGRAMACIÓN MULTIMEDIA Y DISPOSITIVOS MÓVILES 8
III. EL JUEGO **9**
1. FASES DEL PROYECTO **9**
1.1 Motor de juego **9**
1. 2 Unity **9**
1.3 Cuphead **10**
2. ANÁLISIS DEL SISTEMA 11
2.1 El videojuego **11**
2.2 Logo **12**
2.3 Plataformas **12**
2.4 Interfaz de usuario **13**
2.5 Ambientación **13**
2.6 Movimiento **13**
2.7 Elementos aleatorios **13**
2.8 Vidas **13**
2.9 Niveles **13**
2.10 Menús **14**
2.11 Resolución **14**
2.12 Funcionamiento básico **14**
2.14 Clasificación de edad **14**
2.15 Manual de usuario **14**
2.16 Página web **14**
2.17 Restricciones de hardware **15**
2.18 Restricciones de Software **15**
2.19 Entorno de desarrollo **15**
a) Hardware **15**
b) Software **15**

3.1 Entrada y salida **16**
3.2 Diagrama **16**
3.3 Componentes del juego **18**
b) Audio **18**
c) Físicas **18**
3.4 Componentes lógicos **19**
a) *Barra* **19**
b) *Bloques* **19**
c) *Bola* **20**
d) *Cámara* **20**
e) *Marcador* **20**
4. PROGRAMACIÓN 20
4.1 Barra **20**
4.2 Bola **21**
4.3 Cubo **22**
4.4 Controlador de juego **23**
4.5 Controlador de UI **25**
4.6 Controlador de sonidos **26**
4.7 Controlador de niveles de cubos **27**
4.8 Pared inferior **30**
4.9 Objetos **30**
4.10 Controlador de Idioma **34**
5. PRUEBAS 34
5.1 Formulario **34**
5.2 Respuestas del formulario **36**
5.3 Conclusiones del formulario **37**
IV. LA WEB **38**
1. ANÁLISIS **38**
2. OBJETIVOS **38**
3. DISEÑO **38**
3.1 Bocetos 40
4. PROGRAMACIÓN **45**
4.1 Fuente 45
4.2 Idiomas 45
4.3 Diseño de cabecera 46
4.4 Sección diferentes plataformas 47

4.5 Sección opiniones 47
4.6 Requisitos 48
4.7 Jugar en web 49
4.8 Footer 49
4.9 Flecha arriba 50
4.10 CSS 50
V. CONCLUSIONES FINALES **59**
VI. BIBLIOGRAFÍA **60**

I. OBJETIVOS DEL PROYECTO

En una época en la que las herramientas sobre el desarrollo de videojuegos están al alcance de cualquiera y de manera mucho más fácil e intuitiva para el desarrollador, he decidido juntar dos grandes de mis pasiones para el proyecto que son la programación con los conocimientos adquiridos durante los dos años y los videojuegos, el cual llevo disfrutándolos toda mi vida.

Actualmente está en auge llevar los videojuegos antiguos (comúnmente llamados retro) y adaptarlos a la actualidad.

Motivo por el que he escogido para este proyecto un videojuego arcade conocido por la mayoría de la gente, ***Arkanoid***, videojuego basado en ***Breakout*** originalmente lanzado en 1976 que con el paso del tiempo ha recibido incontables revisiones y versiones basados en el mismo sistema, una barra en la parte inferior de la pantalla se desplaza a la izquierda y derecha para golpear una bola e ir destruyendo los bloques de la parte superior.

Breakout fue desarrollado por Atari, tuvo un gran éxito comercial como la mayoría de los videojuegos arcade, debido a que puede llegar a todos los públicos. **Breakout** dejó un legado y el más popular y que perdura hasta hoy es **Arkanoid.**

La serie **Arkanoid**, lanzado originalmente en 1986 desarrollado por *Taito* y propiedad de *Square Enix* fue tan bien recibido por el público que un año después se lanzaría otro más, llamado **Akanoid: Revenge of Doh** y desde entonces hasta la actualidad, no han dejado de lanzarse juegos desarrollados por la misma compañía.

Entre ellos estarían Arkanoid: Doh It Again (1997), Arkanoid Returns (1997), Arkanoid DS (2007), Arkanoid Live y Arkanoid Plus!, Arkanoid vs Space Invaders (2015).

Cabe destacar, que además de la icónica marca **Arkanoid**, también se han disfrutado de incontables clónicos llamados de otro modo, versiones para todas las consolas y también para móvil y navegadores.

Es por ello, que basándome en mi experiencia con los videojuegos he escogido **Arkanoid** como punto de referencia para crear el mío propio, pudiendo llegar a diferentes edades y personalidades sin distinción, haciendo posible que con la tecnología de hoy en día sea disfrutable tanto en Android, como Windows o desde el propio navegador web, sin necesidad de instalación previa.

II. MÓDULOS FORMATIVOS APLICADOS EN EL TRABAJO

1. PROGRAMACIÓN

En base a la metodología de la programación, sobre cómo aplicar la lógica en lenguaje programable para que el ordenador lo entienda y aprendiendo fundamentalmente Java he usado otros lenguajes siendo C# para el caso de Unity y JavaScript para web junto a PHP también para web.

2. ENTORNOS DE DESARROLLO

He usado GitHub para salvaguardar mi proyecto continuamente, teniendo siempre presente una copia de seguridad disponible.

3. LENGUAJE DE MARCAS

El proyecto contiene una página web para publicitar y jugar al videojuego, se ha realizado con HTML y CSS aplicando los lenguajes de programación PHP y JavaScript.

4. DESARROLLO DE INTERFACES

Se ha aplicado los conocimientos aprendidos sobre JavaScript y el manual de usuario con la ayuda de HelpNdoc, que tiene como uso que el usuario pueda obtener soporte del videojuego y los conocimientos mínimos, así como los requisitos.

También un instalador para que los usuarios puedan instalar Arkanoid desde Windows con una interfaz sencilla y familiar.

5. INGLÉS

Tanto el videojuego como la página web contienen como alternativa al español, el inglés para que sea más accesible a todo el mundo, sobre todo la página web que contiene la mayoría de los textos importantes.

6. PROGRAMACIÓN MULTIMEDIA Y DISPOSITIVOS MÓVILES

La mayor concentración del proyecto está en este módulo con Unity como motor del videojuego.

III. EL JUEGO

1. FASES DEL PROYECTO

1.1 Motor de juego

Este término hace referencia a una serie de rutinas de programación, el cual permite resolver una tarea específica que permiten la creación y funcionamiento de un videojuego.

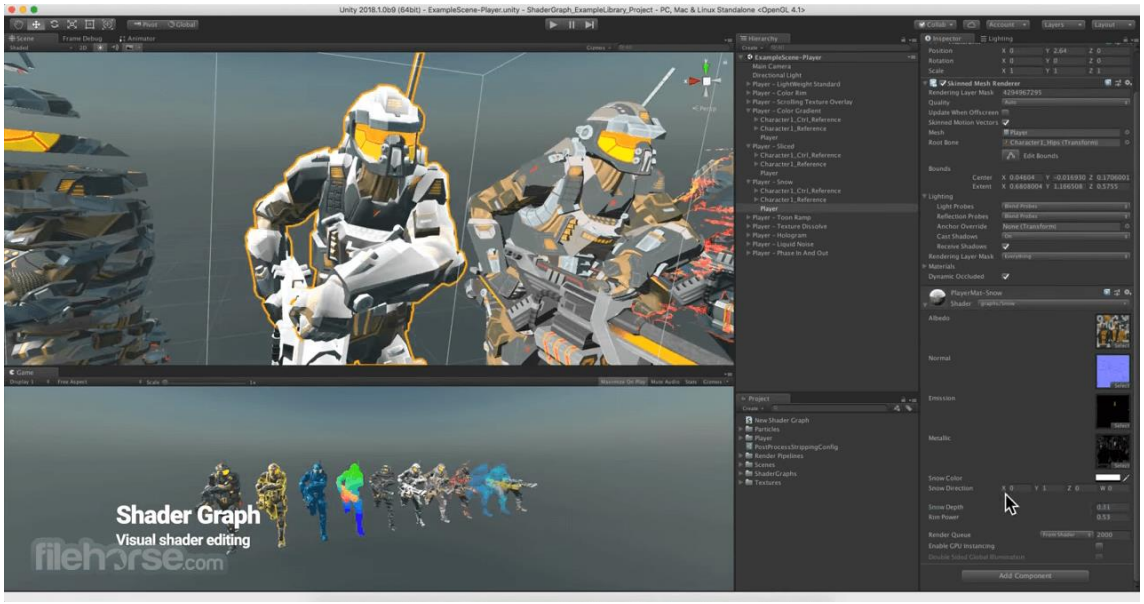
Se incluye en este ámbito un motor gráfico para renderizar gráficos 2D y 3D, un motor físico que simula las leyes de la física con detección de colisiones, motor de sonidos, gestión de memoria, soporte para lenguaje de secuencia de comandos.

Los personajes, enemigos, terrenos, colisiones, conductas, etc., son elementos que forman parte del motor de nuestro videojuego, que, juntándolos, forman el videojuego y el funcionamiento del mismo.

Para este videojuego, he elegido Unity, sotfware con el que he estado trabajando durante el curso para el desarrollo de un juego, ya que es el motor que usaré he decidido adentrarme un poco más en él.

1. 2 Unity

Unity inicialmente comenzó siendo una herramienta el desarrollo de videojuegos exclusiva para Mac, aún que únicamente se podría desarrollar en equipos Mac, los proyectos realizados en el mismo podrían ser exportados para diferentes plataformas, como en la actualidad. Aunque no era el mejor motor gráfico entonces, tenía el factor diferencial, un entorno más o menos intuitivo y fácil de manejar.



Debido al éxito Unity fue actualizándose en diferentes versiones, ofreciendo mejoras en rendimiento y basándose hasta ahora en un mismo corazón, OpenGL.

Al tratarse de una herramienta gratuita, le ha llevado al crecimiento y aceptación por casi toda la industria ya que permite trabajar desde el principio partiendo de un presupuesto nulo o bajo, acercándolo a cualquier usuario que esté interesado en el desarrollo de videojuegos.

Unity permite implementaciones de red, que aun que no son muy extensas quiere decir que si queremos podemos incluirle estadísticas en red, multiplayer, etc.

La comunidad que contiene Unity es bastante extensa, existe bastante material para documentarse y muchísima gente compartiendo proyectos sin ánimo de lucro, permitiendo adentrarse partiendo casi de cero para cualquier usuario que quiera aprender, eso sí, hay que destacar que la mayoría de la documentación, como es lógico está en inglés.

Para que entendamos la magnitud de lo que significa Unity, pondremos como ejemplo uno de los grandes videojuegos que se han hecho con él, con un éxito comercial incuestionable.

1.3 Cuphead

Desarrollado por StudioMDHR perteneciente a Microsoft, fue desarrollado por tan solo 3 personas, fue lanzado en 2017, se caracteriza por su diseño gráfico, inspirado en los antiguos dibujos de Walt Disney, los dos protagonistas deben derrotar diferentes enemigos para completar una deuda pendiente con el Diablo.

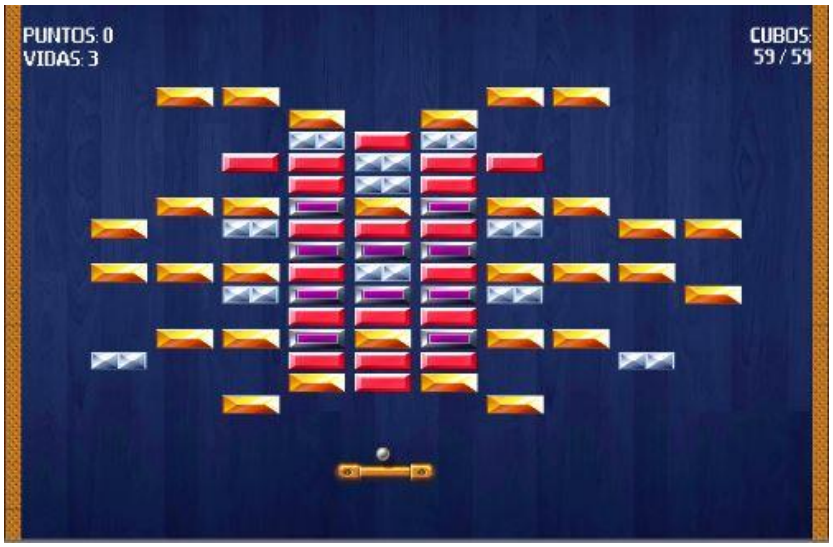
Actualmente Cuphead, ha vendido más de 5 millones de copias a nivel mundial, se trata de un hito si tenemos en cuenta las dimensiones del proyecto, alabado por la crítica y los jugadores, llegando incluso al cine con una serie de animación disponible en diferentes plataformas.

2. ANÁLISIS DEL SISTEMA

En este punto se representan los requisitos y características que se detectan para el desarrollo de Arkanoid. El objetivo final de este apartado es señalar las funciones que debe tener el videojuego.

2.1 El videojuego

Arkanoid es un juego basado en el clásico del mismo nombre llevado a las plataformas actuales, realizado en 2D controlamos una barra con el objetivo de golpear la bola hasta destruir todos los bloques, completando los diferentes niveles y sumando puntos al marcador, el juego termina cuando se pierden todas las vidas y se completan todos los niveles.



2.2 Logo

Se usará el clásico logo de Arkanoid que es bien conocido por todos los públicos.

2.3 Plataformas

El objetivo principal es llegar a los máximos usuarios posibles con las herramientas que disponemos en Unity y los tiempos establecidos, es por ello que estará disponible en Android, Windows y navegador web.

2.4 Interfaz de usuario

El jugador tendrá como elementos de comunicación de entrada, donde el usuario podrá mover la barra y los elementos de salida, donde la pantalla representa los niveles, bloques, etc.., también el altavoz para reproducir sonidos.

2.5 Ambientación

Arkanoid se basará en un ambiente clásico de sus antecesores, abstracto con diferentes tonalidades de colores y sensación espacial.

2.6 Movimiento

Solo se ejercita una función, el movimiento lateral de la barra, impidiendo que la bola pase al nivel inferior para no perder vidas, se tendrá en cuenta la inercia con la que se mueve para la dirección que debe tomar la bola, dependiendo del sistema será con el ratón o con el teclado.

2.7 Elementos aleatorios

Al golpear los cubos, existe un cierto porcentaje de aleatoriedad que hacen al juego más emocionante, haciendo caer diferentes objetos que interactúan con el escenario y la barra.

- Multibolas: Se suman a la pantalla más bolas iguales que la original.
- Extensión: La barra se extiende por los lados haciéndose más grande.
- Contracción: La barra sufre perdida de tamaño por los laterales.
- Pérdida de vida: El nivel de vidas se reduce en uno, lo que puede terminar con la partida.

2.8 Vidas

Se dispone de 3 vidas, podrá verse disminuida por los elementos aleatorios que hacen perder una vida, una vez se pierden todas las vidas, el juego finaliza.

2.9 Niveles

Se compone de 8 niveles, pero con actualizaciones, se podrán modificar, suprimir o añadir niveles al juego.

2.10 Menús

Pantalla de menú inicial al comienzo del juego, pantalla de pausa en caso de que pulsemos el botón escape (atrás en dispositivos Android) y también si lo dejamos en segundo plano, por último, pantalla de derrota y victoria.

2.11 Resolución

Adaptado para jugar a Arkanoid en modo horizontal en Android, el juego activará al dispositivo Android forzosamente en ese modo una vez iniciado.

2.12 Funcionamiento básico

Al eliminar todos los cubos rompibles de la pantalla, se pasa al siguiente nivel, hasta completarlos todos o perder todas las vidas, al final, se muestra una pantalla con la puntuación obtenida y la máxima.

2.14 Clasificación de edad

Nos ceñimos a PEGI para escoger la edad mínima de juego, el juego no contiene escenas violentas ni sonidos violentos, por lo que hablamos de que el juego se considera PEGI 3.

PEGI 3

El contenido de los juegos con esta clasificación se considera apto para todos los grupos de edades. Se acepta cierto grado de violencia dentro de un contexto cómico (por lo general, formas de violencia típicas de dibujos animados como Bugs Bunny o Tom y Jerry). El niño no debería poder relacionar los personajes de la pantalla con personajes de la vida real, los personajes del juego deben formar parte exclusivamente del ámbito de la fantasía. El juego no debe contener sonidos ni imágenes que puedan asustar o amedrentar a los niños pequeños. No debe oírse lenguaje soez.

2.15 Manual de usuario

En el caso de Windows, se acompaña en la instalación un manual de usuario, por si el usuario tiene dudas al respecto.

2.16 Página web

Se realiza una página web para que la promoción sobre el videojuego sea mayor, pudiendo darse a conocer mucho más más allá de Android, además la web es donde podrá descargar el usuario la aplicación ya sea para PC o para Android.

Punto a destacar, la web contiene un apartado para poder disfrutar del videojuego sin necesidad de descargar la aplicación.

2.17 Restricciones de hardware

Las que se producen por que el hardware del equipo no cumple con los requisitos establecidos para un funcionamiento óptimo, al tratarse de una aplicación donde mayoritariamente será ejecutado en Android y que gran parte de los usuarios disponen de terminales de gama media/baja habrá que trabajar con las limitaciones de hardware que éstos tienen.

2.18 Restricciones de Software

Estas restricciones provienen del sistema operativo, dado que la vida útil de un dispositivo Android es de 15 meses y la vida media está establecida entre 18 y 24 meses será importante hacerlo compatible con dispositivos actuales y desde más atrás, para llegar al máximo público posible.

Arkanoid ha sido testeado con un Huawei P10 con Android 10, con un Samsung S6 con Android 8 y en ambos casos ha funcionado de igual manera, óptima.

2.19 Entorno de desarrollo

a) Hardware

Ordenador Asus A55V – I73630QM @ 2.40 GHZ – 8G RAM – 1TB HDD

b) Software

- Sistema operativo Windows 10.
- Unity 2019 con licencia gratuita.
- Entorno de programación Visual Studio 2019.
- Android 10.

3. DISEÑO DEL SISTEMA

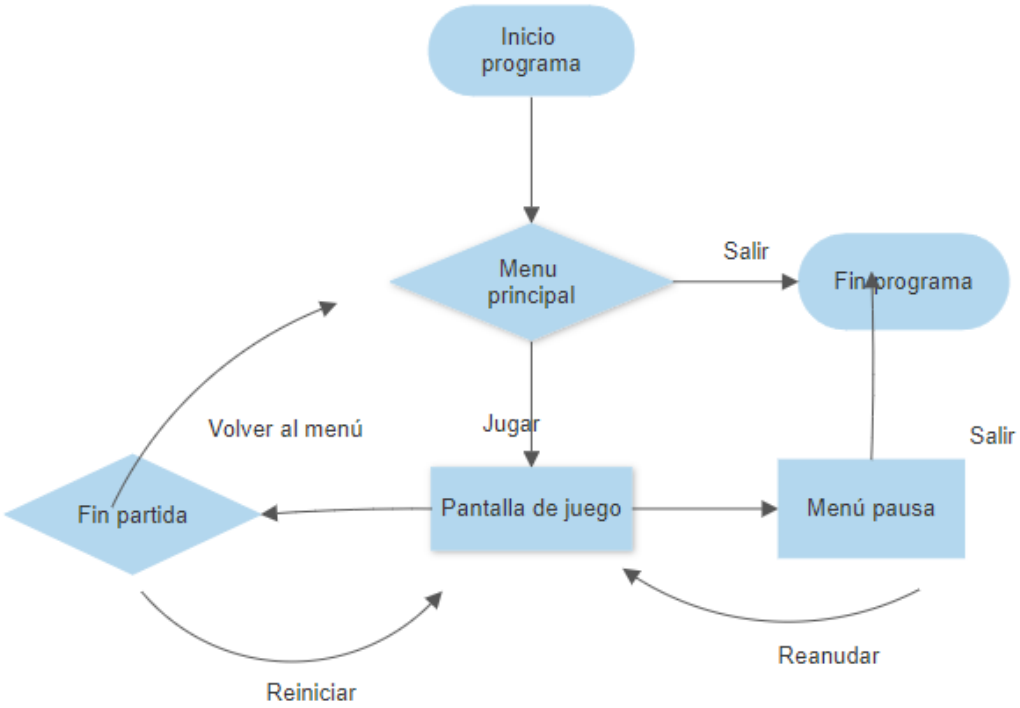
El objetivo es hacer lo más correcto este apartado para que los errores a subsanar en la fase de implementación sean mínimas y cumplir correctamente con los plazos estimados a la entrega del proyecto.

3.1 Entrada y salida

- **Entrada:** Componente que se encarga de la gestión que le llega del usuario al videojuego, encargándose de la interfaz mediante la pantalla táctil en este caso, desde arrancar el programa, seleccionar las opciones, idiomas o mover la barra del juego.
- **Salida**: Componente que se encarga de trasmitir la información al usuario, que será a través de la pantalla para mostrar la imagen de resultado de la aplicación y el altavoz, donde se expulsa el sonido que emite la aplicación, sonido de fondo, golpes a los bloques, etc…

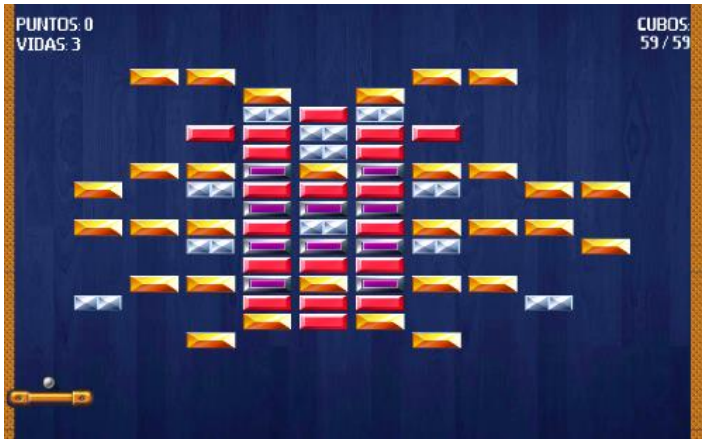
3.2 Diagrama

Diagrama de flujo del funcionamiento general de Arkanoid



- Inicio del programa: El usuario arranca la aplicación.
- Menú principal: Se muestran dos opciones, jugar que pasa a la pantalla de juego o salir, que acompaña a la finalización del programa.

- Pantalla de juego: El usuario puede pausar la partida o terminar la partida una vez pierda o gane.



- Menú de pausa: El usuario puede volver a la pantalla de juego o bien salir.

- Fin de partida: Se muestra la pantalla de victoria o derrota según termine.

3.3 Componentes del juego

a) Escenas

El juego se compone de dos escenas, una del menú principal y otra del juego en funcionamiento.

b) Audio

El componente de audio se compone de diferentes pistas, que reproducirán los sonidos según sean adecuados, estos serán, un único script se centra en todo el sonido.

- Melodía menú principal
- Melodía del juego en funcionamiento
- Sonido al golpear un bloque
- Sonido al destruir un bloque
- Sonido al perder vida
- Sonido de derrota
- Sonido de victoria

c) Físicas

Se utiliza el motor de físicas por defecto de Unity que gestiona las colisiones que existen entre la bola, las paredes, los bloques y la barra, así como la caída de objetos. La gravedad será por defecto (-9.81).

Las colisiones se controlan mediante los eventos del gestor de colisiones, utilizando las etiquetas, para diferenciar que componente está chocando y de que manera debe responder al choque con el RigyBody, por ejemplo, la detección de choque entre la bola y los bloques.

3.4 Componentes lógicos

El juego contiene varios componentes con lógica programada, para que actúen según convenga, estos serán.

a) *Barra*

La barra, será el único componente del gameplay que el jugador controla, este responde al movimiento izquierda y derecha, ya sea bien con el teclado, con el ratón o con los dedos siendo táctil.

b) *Bloques*

Existen cuatro tipos de bloques, pero todos responden a una misma lógica, tres de ellos al golpeo de la bola y el último se mantiene estático.

Los que necesitan tres golpes para ser totalmente destruidos, cambiarán de Sprite para cambiar el estado según los golpes restantes.

Los que necesitan dos golpes, al igual que los de tres, cambia el estado del Sprite para responder al número de golpes.

Bloque de único golpeo.

Bloque fijo, este se mantiene en la pantalla sin poder ser destruido hasta que se complete el nivel.

c) *Bola*

La bola responde a los impactos que recibe de los componentes del escenario.

d) *Cámara*

La cámara muestra el escenario al completo, no cumple ningún requisito exigente.

e) *Marcador*

Se establece la puntuación, vida disponible y los bloques existentes en el escenario / restantes.

4. PROGRAMACIÓN

Fase más importante del proyecto, donde se implementan todas las ideas hasta el momento para dar forma a Arkanoid, serán los scripts los que respondan a toda la lógica del juego.

4.1 Barra

El script que se encarga del movimiento y reacción de la barra, a elección del administrador, para elegir un tamaño adaptado a la necesidad del juego, pondremos como público el tamaño de la barra.

La función más importante responde al movimiento.

```
1 referencia
private void MovimientoBarra()
{
    float barraShift = (defaultBarraWidthEnPixels - ((defaultBarraWidthEnPixels / 2) * tamBarra.size.x)) / 2;
    float limiteIzquierda = defaultLimiteIzq - barraShift;
    float limiteDerecha = defaultLimiteDer + barraShift;
     float mousePositionPixels = Mathf.Clamp(Input.mousePosition.x, limiteIzquierda, limiteDerecha); //Límites a la izquierda y derecha
     float mousePositionWorldX = mainCamera.ScreenToWorldPoint(new Vector3(Input.mousePosition.x, 0, 0)).x;
    transform.position = new Vector3(mousePositionWorldX, barraInicialY, 0);
    transform.position += new Vector3(Input.GetAxis("Horizontal"), 0, 0) * 20 * Time.deltaTime;

}
```

A la izquierda y derecha pondremos unos límites que se establecen con Mathf.Clamp, para que el usuario no pueda sobrepasar estos límites.

Detectamos la posición del ratón y el movimiento para que la barra responda a ello.

También responde a las teclas izquierda y derecha del teclado.

4.2 Bola

```
2 referencias
private void BolaInicial()
{

    Vector3 posicionBarra = BarraControlador.Instance.gameObject.transform.position; //Sacamos la posición inicial de la barra
    Vector3 posicionInicial = new Vector3(posicionBarra.x, posicionBarra.y + .27f, 0); //Cargamos la posición inicial donde irá la bola sumándole .27f a l
    bolaInicial = Instantiate(bolaPrefab, posicionInicial, Quaternion.identity); //Ponemos la bola cargando el prefab
    bolaInicialRb = bolaInicial.GetComponent<Rigidbody2D>();

    this.bolas = new List<BolaController>
    {
        bolaInicial
    };

}
```

Momento de crear la bola y asignársela a la barra.

Al cargar el juego, la bola tendrá una posición inicial, esta se corresponde con el centro de la barra.

```
0 referencias
void Start()
{

    BolaInicial(); //Método que cargará una bola en una posición

}

// Update is called once per frame
0 referencias
void Update()
{
    if(!JuegoControlador.Instance.IsGameStarted)
    {
        //Si el juego no ha comenzado, ponemos la bola en la barra y hacemos que siga a la barra
        Vector3 posicionBarra = BarraControlador.Instance.gameObject.transform.position;
        Vector3 bolaPosicion = new Vector3(posicionBarra.x, posicionBarra.y + .27f, 0);
        bolaInicial.transform.position = bolaPosicion; //Asignamos por cada frame que la bola esté junto a la barra aun que se mueva.

        if(Input.GetKeyDown(KeyCode.UpArrow) || Input.GetMouseButtonDown(0)) //Si se presiona flecha arriba
        {
            bolaInicialRb.isKinematic = false; //Le quitamos el kinematic (las físicas ahora le afectan)
            bolaInicialRb.AddForce(new Vector2(0, velocidadInicialBola)); //Le damos fuerza a la bola
            JuegoControlador.Instance.IsGameStarted = true; //Indicamos al controlador que el juego ha comenzado

        }
    }
}
```

```
0 referencias
private void OnCollisionEnter2D(Collision2D collision)
{
    if (collision.gameObject.CompareTag("Bola")) //Si realiza contacto con la bola, entra en el método
    {
        if(this.golpes < 4)
        {


            this.golpes--;

            if (this.golpes <= 0)
            {
                CubosSprites.Instance.cubosRestantes.Remove(this);
                SonidosControlador.Instance.ReproducirSonidoExplosionCubo();
               OnCuboDestruccion?.Invoke(this);
                OnCuboDestruido();
                Destroy(this.gameObject); //Si se gastan todos los puntos, destruimos el objeto
            }
            else
            {
                SonidosControlador.Instance.ReproducirSonidoGolpeoCubo();
               if(tipoCubo == 3)
               {
                sprite.sprite = CubosSprites.Instance.sprites3golpes[this.golpes - 1]; //Se le asigna el sprite según la vida que le quede
               }
               if (tipoCubo == 2)
               {
                   sprite.sprite = CubosSprites.Instance.sprites2golpes[this.golpes - 1]; //Se le asigna el sprite según la vida que le quede
               }
            }
        }
    }
}
```

Inicialmente cargamos la primera bola, si el juego no ha empezado, la bola irá siguiendo a la barra, hasta que se pulse la barra espaciadora, botón izquierdo del ratón o pulsación táctil, le afectarán las físicas e impulsamos fuerza para que la bola salga despedida.

4.3 Cubo

En este punto, creamos la lógica correspondiente a los cubos que se mostrarán en pantalla y la reacción que deben tener.

Este método, indica que si detecta una colisión correspondiente a la bola deberá responder y dependiendo del estado, será de una manera u otra, dependiendo de la vida que le quede al cubo o si es un cubo fijo, reduciendo su vida en caso de que sea necesario o bien destruyéndolo, se encarga también de llamar al método que indica lo que debe de ocurrir cuando el cubo es destruido, también reproduce el sonido.

```
1 referencia
private void OnCuboDestruido()
{

    float probabilidad = UnityEngine.Random.Range(0, 100f);
    float deProbabilidad = UnityEngine.Random.Range(0, 100f);
    bool armaListo = false;
    if(probabilidad <= ObjetosControlador.Instance.probabilidadPositivos)
    {
        armaListo = true;
        Objeto newBuff = this.SpawnObjeto(true);
    }
    if (probabilidad <= ObjetosControlador.Instance.probabilidaNegativos && !armaListo)
    {
        Objeto newBuff = this.SpawnObjeto(false);
    }


}
```

En esta función, cuando el cubo es destruido ocurre las probabilidades de caer un objeto positivo y negativo.

4.4 Controlador de juego

Este Script, se encarga de las funciones generales del juego, que todo funcione

```
0 referencias
void Start()
{
    Screen.SetResolution(960, 640, false);
    vidas = vidasDisponibles;
    BolaController.OnMuerteBola += OnMuerteBola; //Se crea el evento y cuando surja algún cambio entra al método
    CuboControlador.OnCuboDestruccion += OnCuboDestruccion;


}
```

correctamente.

Lo primero que le indicamos será la resolución que va a tener el juego, también establecemos públicamente las vidas que puede tener el jugador.

Establecemos además el menú de pausa en caso de que se pulse escape o en su defecto volver atrás en Android

```
0 referencias
void Start()
{
    Screen.SetResolution(960, 640, false);
    vidas = vidasDisponibles;
    BolaController.OnMuerteBola += OnMuerteBola; //Se crea el evento y cuando surja algún cambio entra al método
    CuboControlador.OnCuboDestruccion += OnCuboDestruccion;


}
```

```
0 referencias
private void Update()
{
    if (Input.GetKeyDown(KeyCode.Escape))
    {
        Time.timeScale = 0f;
        panelPausa.SetActive(true);
    }
}
```

También señalamos lo que ocurrirá en caso de que el usuario pase a segundo plano el juego en Android, se mostrará el menú de pausa a la vez que se para el juego.

```
0 referencias
void OnApplicationPause(bool pauseStatus)
{
    if (pauseStatus == true)
    {
        panelPausa.SetActive(true);
        Time.timeScale = 0f;
    }
}
```

En caso de que un cubo sea destruido, comprueba si todos los cubos han sido eliminados, en ese caso se pasa al siguiente nivel, se resetean las bolas y se reproduce el sonido.

```
1 referencia
private void OnCuboDestruccion(CuboControlador obj)
{
    //UIControlador.puntos +=10;
    if(CubosSprites.Instance.cubosRestantes.Count <= 0)
    {
        SonidosControlador.Instance.ReproducirSonidoSiguienteNivel();
        BolasControlador.Instance.ResetearBolas();
        JuegoControlador.Instance.IsGameStarted = false;
        CubosSprites.Instance.CargarSiguienteNivel();
    }
}
```

Si se pierde una vida se resta del contador y en caso de que se pierdan todas, se activa el panel de muerte.

```
public void PerderVida()
{
    vidas--;
    OnVidaPerdida?.Invoke(vidas);
    if (vidas < 1)
    {


        UIControlador.Instance.ActivarpanelMuerte();

    }
}
```

Los siguientes métodos se encargan de cargar el panel de victoria, de los botones de reiniciar, salir y volver al menú. A la derecha, de guardar y cargar el resultado de puntuación.

```
public void MostrarPanelVictoria()
{
    UIControlador.Instance.ActivarpanelVictoria();
}

0 referencias
public void ReiniciarJuego()
{

    SceneManager.LoadScene(1);

}
0 referencias
public void ContinuarJuego()
{
    Time.timeScale = 1f;
    panelPausa.SetActive(false);
}
0 referencias
public void VolverMenu()
{

    SceneManager.LoadScene(0);

}
```

```
5 referencias
public int GetMaxScore()
{
    return PlayerPrefs.GetInt("MaximaPuntuacion", 0);
}
1 referencia
public void SetMaxScore(int puntuacion)
{
   PlayerPrefs.SetInt("MaximaPuntuacion", puntuacion);
}
}
```

4.5 Controlador de UI

Aquí irán los métodos que actualizan los canvas.

```
void Start()
{


    OnVidaPerdida(JuegoControlador.Instance.vidasDisponibles);

    puntos = 0;
    //recordpuntos.text = $"RECORD: {JuegoControlador.Instance.GetMaxScore()}";
    recordpuntos.text = $"{IdiomasJuego.Instance.textos[1]}: {JuegoControlador.Instance.GetMaxScore()}";
}
```

Al comenzar, establecemos las vidas al marcador, los puntos y el record máximo.

También comprobamos si el panel final del canvas se ha activado para actualizar todos los marcadores.

```
0 referencias
private void Update()
{

    if (panelFinal.active == true)
    {
        vidasTexto.GetComponent<Text>().enabled = false;
        puntosTexto.GetComponent<Text>().enabled = false;
        cubosTexto.GetComponent<Text>().enabled = false;
        recordpuntos.GetComponent<Text>().enabled = false;
        //puntuacionFinal.text = puntos.ToString();
        if (puntos > JuegoControlador.Instance.GetMaxScore())
        {
            JuegoControlador.Instance.SetMaxScore(puntos);


        }

    }

    //ActualizarCubosRestantesTexto();
    //OnVidaPerdida(JuegoControlador.Instance.vidas);
    OnNivelCargado();
}
```

Los siguientes métodos se encargan de controlar las puntuaciones dependiendo del estado del juego, véase cuando se pierde una vida, cuando se destruye un cubo, actualizar los marcadores, etc.

```
2 referencias
private void ActualizarCubosRestantesTexto()
{   //Indicamos los cubos que hay y los que quedan
    cubosTexto.text = $"{IdiomasJuego.Instance.textos[3]}:{Environment.NewLine}{CubosSprites.Instance.cubosRestantes.Count} / {CubosSprites.Insta
}
0 referencias
private void OnDisable()
{
    CuboControlador.OnCuboDestruccion -= OnCuboDestruccion;
    CubosSprites.OnNivelCargado -= OnNivelCargado;
    JuegoControlador.OnVidaPerdida -= OnVidaPerdida;
    //vidasTexto.GetComponent<Text>().enabled = false;
}

2 referencias
public void ActivarpanelMuerte()
{
    SonidosControlador.Instance.ReproducirSonidoGameOver();
    finalWinLossTexto.text = $"{IdiomasJuego.Instance.textos[4]}";
    finalPunTexto.text = $"{IdiomasJuego.Instance.textos[5]}:   {puntos}";
    finalPunRecordTexto.text = $"{IdiomasJuego.Instance.textos[1]}:   {JuegoControlador.Instance.GetMaxScore()}";
    panelFinal.SetActive(true);
    botonReiniciar.SetActive(true);
    botonMenuPrincipal.SetActive(false);
}
1 referencia
public void ActivarpanelVictoria()
{
    SonidosControlador.Instance.ReproducirSonidoVictoria();
    if(IdiomasJuego.Instance.idioma== 0)
    {
        finalWinLossTexto.text = "¡ CONGRATULATIONS !";
    }
    if (IdiomasJuego.Instance.idioma == 1)
    {
        finalWinLossTexto.text = "¡ ENHORABUENA !";
```

```
// vidasTexto.text = "VIDAS: " + vidasRestantes;
vidasTexto.text = $"{IdiomasJuego.Instance.textos[2]}: {vidasRestantes}";

}
```

4.6 Controlador de sonidos

Este script es bastante sencillo, simplemente se encarga tener preparadas las pistas para reproducir los sonidos cuando reciba llamadas de otras clases.

```
private AudioSource reproductor;
public AudioClip sonidoExplosionCubo;
public AudioClip sonidoBolaGolpe;
public AudioClip sonidoMuerteBola;
public AudioClip sonidoSiguienteNivel;
public AudioClip sonidoGolpeoCubo;
public AudioClip sonidoGameOver;
public AudioClip sonidoVictoria;
0 referencias
void Start()
{
    reproductor = GetComponent<AudioSource>();
}

1 referencia
public void ReproducirSonidoExplosionCubo()
{
    reproductor.clip = sonidoExplosionCubo;
    reproductor.Play();

}
1 referencia
public void ReproducirSonidoGolpeoBola()
{
    reproductor.clip = sonidoBolaGolpe;
    reproductor.Play();

}
0 referencias
public void ReproducirSonidoMuerteBola()
{
    reproductor.clip = sonidoMuerteBola;
    reproductor.Play();
```

4.7 Controlador de niveles de cubos

Es también uno de los scripts más importantes de la aplicación, ya que se encarga de gestionar todos los cubos que aparecen en pantalla, como deben aparecer y cuando.

```
private void Start()
{
    this.contenidorCubos = new GameObject("contenidorCubos");
    this.niveles = this.CargarNiveles();
    this.cubosRestantes = new List<CuboControlador>();
    this.cubosFijos = new List<CuboControlador>();
    this.GenerarCubos();

}

2 referencias
public void CargarNivel(int nivel)
{
    nivelActual = nivel;
    LimpiarCubosRestantes();
    LimpiarCubosFijos();
    GenerarCubos();
}

1 referencia
```

Nada más comenzar, lo primero que hacemos es generar un contenedor de cubos, donde irán los cubos rompibles que aparecen en pantalla, una lista de cubos restantes y otra de cubos fijos, invocamos el método que los genera.

Al cargar un nivel se limpiarán los cubos que hubiera, tanto fijos como rompibles y se generan los nuevos.

```
1 referencia
private List<int[,]> CargarNiveles()
{
    TextAsset text = Resources.Load("niveles") as TextAsset;

    string[] filas = text.text.Split(new String[] { Environment.NewLine }, StringSplitOptions.RemoveEmptyEntries);



    List<int[,]> niveles = new List<int[,]>();
    int[,] nivelActual = new int[maxFilas, maxColum];
    int filaActual = 0;

    for(int fila = 0; fila < filas.Length; fila++)
    {
        string linea = filas[fila];
        if (linea.IndexOf("--") == -1)
        {
            string[] cubos = linea.Split(new char[] { ',' }, StringSplitOptions.RemoveEmptyEntries);
            for (int col = 0; col < cubos.Length; col++)
            {
                nivelActual[filaActual, col] = int.Parse(cubos[col]);
            }
            filaActual++;
        }
        else
        {
            filaActual = 0;
            niveles.Add(nivelActual);
            nivelActual = new int[maxFilas, maxColum];
        }
    }
    return niveles;
```

La carga de niveles, previamente diseñados, se harán desde un archivo de texto (txt), el cual identifica los dígitos por línea y los carga al nivel actual. Los 0 serán que no hay cubo, el 1,2 y 3 serán los golpes que necesitan y el 4 indica que es un cubo fijo (no rompible). La función realiza un Split para separar por coma (,), cuando termina la línea salta a la siguiente hasta terminar el nivel.

Momento de generar los cubos del nivel cargador.

```
private void GenerarCubos()
{
    this.cubosRestantes = new List<CuboControlador>();
    this.cubosFijos = new List<CuboControlador>();
    int[,] nivelActualDatos = this.niveles[this.nivelActual];
    float spawnX = posicionInicialCuboX;
    float spawnY = posicionInicialCuboY;
    float zShift = 0;


    for (int fila = 0; fila < this.maxFilas; fila++)
    {
        for ( int col =0; col < maxColum; col++)
        {
            int tipoCubo = nivelActualDatos[fila, col];

            if (tipoCubo > 0 && tipoCubo <  4)
            {
                //Creamos el cubo
                    CuboControlador cuboNuevo = Instantiate(cuboPrefab, new Vector3(spawnX, spawnY, 0.0f - zShift),Quaternion.identity) as CuboControlador
                    SpriteRenderer spritea = cuboNuevo.GetComponent<SpriteRenderer>(); //Extraemos el componente sprite
                    spritea.sprite = sprites[tipoCubo-1]; //Según el tipo que sea, le asignamos la imagen correspondiente
                    cuboNuevo.golpes = tipoCubo; //El número de golpes que debe recibir
                    //spritea.color = this.ColoresBloque[tipoCubo-1]; //Le asignamos el color al sprite
                    this.cubosRestantes.Add(cuboNuevo);



                //
                //zShift += 0.0001f;
                zShift += 0.0001f;
```

Una vez cargado el nivel, identifica cada número y lo crea el cubo, carga el Sprite correspondiente al número de cubo y la vida que debe de tener, lo añade al contenedor de cubos, cada vez que genera uno, añade un espacio para colocar el siguiente.

```
if (tipoCubo == 4) //Creamos el cubo de tipo fijo
{
    CuboControlador cuboFijo = Instantiate(cuboFijoPrefab, new Vector3(spawnX, spawnY, 0.0f - zShift), Quaternion.identity) as CuboControlador
    // SpriteRenderer spritea.sprite = sprites[tipoCubo - 1]; //Según el tipo que sea, le asignamos la imagen correspondiente
    cuboFijo.golpes = 4; //El número de golpes que debe recibir
                         //spritea.color = this.ColoresBloque[tipoCubo - 1]; //Le asignamos el color al sprite
    this.cubosFijos.Add(cuboFijo);
}
spawnX += shifAmount+0.8f; //espacios que habrá entre cubos en el eje x
if(col +1 == this.maxColum)
{
    spawnX = posicionInicialCuboX;
}
```

Si es un cubo de tipo 4, genera uno fijo, en número de golpes indicamos 4 para que se lo pase al constructor del cubo, el cual ignora reacciones a este número. Los dos siguientes métodos limpian la lista de cubos.

```
1 referencia
private void LimpiarCubosRestantes()
{
    foreach(CuboControlador cubo in this.cubosRestantes.ToList())
    {
        Destroy(cubo.gameObject);
    }


}
```

```
private void LimpiarCubosFijos()
{

    foreach (CuboControlador cubo2 in this.cubosFijos.ToList())
    {
        Destroy(cubo2.gameObject);
    }

}
```

La carga del siguiente nivel detecta si existen más líneas, en caso contrario, llama a la UI para mostrar el panel de victoria.

```
public void CargarSiguienteNivel()
{
    nivelActual++;
    if(nivelActual >= niveles.Count)
    {
        JuegoControlador.Instance.MostrarPanelVictoria();
    }
    else
    {
        CargarNivel(nivelActual);
    }
}
```

4.8 Pared inferior

Se trata de que cuando la bola impacte con la parte inferior, se destruya y reproduzca el sonido de muerte.

```
0 referencias
public class MuertePared : MonoBehaviour
{
    0 referencias
    private void OnTriggerEnter2D(Collider2D collision)
    {
        if(collision.tag == "Bola")
        {
            GameObject sonidosControlador = sonidosControlador = GameObject.FindGameObjectWithTag("SonidosControlador");
            sonidosControlador.SendMessage("ReproducirSonidoMuerteBola");
            BolaController bola = collision.GetComponent<BolaController>(); //Capturamos el objeto bola que ha entrado
            BolasControlador.Instance.bolas.Remove(bola); //Lo eliminamos del array de bolas que están en juego
            bola.muerte();
        }
    }
}
```

4.9 Objetos

Los objetos responden siempre a la barra, pero no de la misma forma, es por ello que haremos un script general para objetos y después scripts que responden según el objeto que choca con la barra.

```
12 referencias
public abstract class Objeto : MonoBehaviour
{
    0 referencias
    private void OnTriggerEnter2D(Collider2D collision)
    {
        if (collision.tag == "Barra")
        {
            this.AplicarEfecto();
        }

        if (collision.tag == "Barra" || collision.tag == "ParedMuerte")
        {
            Destroy(this.gameObject);
        }
    }

    4 referencias
    protected abstract void AplicarEfecto();
}
```

En el caso que el objeto entre en contacto con la barra, se aplica el efecto correspondiente y tanto si es con la barra o con la pared inferior se elimina.

Cada efecto tiene su propia clase, extender o contraer, multibolas o perder vidas.

Extender o contraer, llama al método en la barra para realizar el efecto.

```
0 referencias
public class Extender : Objeto
{
    public float nuevoTam = 2.5f;
    4 referencias
    protected override void AplicarEfecto()
    {

        if(BarraControlador.Instance !=null && !BarraControlador.Instance.barraTransformada)
        {
            BarraControlador.Instance.AnimTam(nuevoTam);
        }

    }
}
```

Se comienza una corrutina para realizar la transformación de la barra, después vuelve a su tamaño original.

```
public void AnimTam(float nuevoTam)
{
    StartCoroutine(AnimarBarra(nuevoTam));
}

1 referencia
public IEnumerator AnimarBarra(float nuevoTam)
{
    barraTransformada = true;
    StartCoroutine(ResetBarraTiempo(this.duracionExtendido));

    if (nuevoTam > this.tamBarra.size.x)
    {
        float tamActual = tamBarra.size.x;
        while(tamActual < nuevoTam)
        {
            tamActual += Time.deltaTime * 2;
            tamBarra.size = new Vector2(tamActual, tamAlto);
            ColliderBarra.size = new Vector2(tamActual, tamAlto);
            yield return null;
        }
    }
    else
    {
        float tamActual = tamBarra.size.x;
        while (tamActual > nuevoTam)
        {
            tamActual -= Time.deltaTime * 2;
            tamBarra.size = new Vector2(tamActual, tamAlto);
            ColliderBarra.size = new Vector2(tamActual, tamAlto);
            yield return null;
        }
    }
    barraTransformada = false;
}
```

```
1 referencia
private IEnumerator ResetBarraTiempo(float segundos)
{
    yield return new WaitForSeconds(segundos);

    AnimTam(this.barratam);

}
}
```

```
0 referencias
public class PerderVida : Objeto
{

    4 referencias
    protected override void AplicarEfecto()
    {



        JuegoControlador.Instance.PerderVida();
```

Perder vida resta la vida un punto, pudiendo hacer que el jugador pierda todas sus vidas, invoca al método perder vida del controlador de juego.

```
1 referencia
public void PerderVida()
{
    vidas--;
    OnVidaPerdida?.Invoke(vidas);
    if (vidas < 1)
    {


        UIControlador.Instance.ActivarpanelMuerte();


    }
}
```

Multibolas hace aparecer en la pantalla 2 bolas más a la que hubiera, llama al método nueva bola del controlador de bolas, al lado el número de bolas, haciendo que salgan las bolas despedidas.

```
public class MultiBolasEfecto : Objeto
{
    4 referencias
    protected override void AplicarEfecto()
    {
        foreach (BolaController bola in BolasControlador.Instance.bolas.ToList())
        {
            BolasControlador.Instance.NuevaBola(bola.gameObject.transform.position, 2);
        }
    }
}
```

Aumentar velocidad realiza un incremento en la velocidad de golpeo de la bola, el cual se restablece pasados 10 segundos, para ello, aplicamos el efecto al entrar al método y comenzamos una rutina para recuperar la velocidad original.

```
public void AumentarVel(float vel)
{


    velocidadInicialBola = vel;
    StartCoroutine(ResetMasVel());
}
public IEnumerator ResetMasVel()
{
    yield return new WaitForSeconds(10);
    velocidadInicialBola = 350;
}
```

4.10 Controlador de Idioma

Este controlador gestiona el lenguaje seleccionado por el usuario, también se encarga de guardar la elección, para la próxima vez que arranque el juego, cargue el idioma seleccionado.

```
1 referencia
public void empezar()
{
    idioma = PlayerPrefs.GetInt("Idioma");
    textoIngles = new string[7];
    textoSpanish = new string[7];
    textos = new string[6];

    CargarDiccionarioIdiomaSpanish();
    CargarDiccionarioIdiomaIngles();
    if (idioma == 0)
    {
        CargarDiccionarioIdiomaIngles();
        IdiomaIngles();
    }
    else if (idioma == 1)
    {
        CargarDiccionarioIdiomaSpanish();
        IdiomaSpanish();
    }
}
2 referencias
public void CargarDiccionarioIdiomaSpanish()
{
    textoSpanish[0] = "PUNTOS";
    textoSpanish[1] = "P.MÁXIMA";
    textoSpanish[2] = "VIDAS";
    textoSpanish[3] = "CUBOS";
    textoSpanish[4] = "FIN DE LA PARTIDA";
    textoSpanish[5] = "PUNTUACIÓN OBTENIDA";
    textoSpanish[6] = "REINICIAR";
```

5. PRUEBAS

Momento en el que pasamos a testear el videojuego con diferentes usuarios, para que éstos reporten como les ha funcionado y comprobar que se cumplen todos los requisitos anteriormente mencionados.

Las pruebas consistirán en responder a un cuestionario con diferentes preguntas acerca de la experiencia que han tenido tras probar la aplicación, con esto podremos comprobar que Arkanoid no tenga fallos, sea fiable y conocer distintas opiniones, por si hubiera problemas, resolverlos.

5.1 Formulario

1 - ¿Ha tenido algún problema en la instalación de la aplicación?

- **Si**
- No -> En caso de marcar esta opción, indique cual ha sido el problema ……..

2 - ¿Puede arrancar correctamente Arkanoid?

- **Si**
- No -> En caso de marcar esta opción, indique cual ha sido el problema ……..

3 - ¿Desde el menú, puede salir correctamente del juego?

- **Si**
- No -> En caso de marcar esta opción, indique cual ha sido el problema ……..

4 - ¿Es capaz de iniciar la partida sin problema?

- **Si**
- No -> En caso de marcar esta opción, indique cual ha sido el problema ……..

5 - ¿Puede realizar una pausa volviendo “atrás”?

- **Si**
- No -> En caso de marcar esta opción, indique cual ha sido el problema ……..

6 - ¿El menú de pausa le permite volver a la partida correctamente?

- **Si**
- No -> En caso de marcar esta opción, indique cual ha sido el problema ……..

7 - Ha podido probar el movimiento de la barra ¿cree que es correcto?

- **Si**
- No -> En caso de marcar esta opción, indique cual ha sido el problema ……..

8 - Al perder una vida ¿Puede volver a jugar si lo desea?

- **Si**
- No -> En caso de marcar esta opción, indique cual ha sido el problema ……..

9 – Si cambia de aplicación ¿Al volver al juego se inició el menú de pausa?

- **Si**
- No -> En caso de marcar esta opción, indique cual ha sido el problema ……..

10 – Ha completado todos los niveles ¿Puede volver al menú principal?

- **Si**
- No -> En caso de marcar esta opción, indique cual ha sido el problema ……..

9 – Si cambia de aplicación ¿Al volver al juego se inició el menú de pausa?

- **Si**
- No -> En caso de marcar esta opción, indique cual ha sido el problema ……..

10– ¿Ha podido completar todos los niveles?

- **Si**
- No -> En caso de marcar esta opción, indique cual ha sido el problema ……..

11 – ¿Cree que el juego es demasiado complicado?

- **Si**
- No -> En caso de marcar esta opción, indique cual ha sido el problema ……..

12 – ¿Le parece divertido el juego?

- **Si**
- No -> En caso de marcar esta opción, indique cual ha sido el problema ……..

5.2 Respuestas del formulario

Se escogieron 5 sujetos, los jugadores 1,2 y 3 son habituales de los videojuegos, el 4 es ocasional y el 5 no suele jugar a videojuegos.

Pregunta	Sujeto 1	Sujeto 2	Sujeto 3	Sujeto 4	Sujeto 5
¿Ha tenido algún problema en la instalación de la aplicación?	No	No	No	No	No
¿Puede arrancar correctamente Arkanoid?	Si	Si	Si	Si	Si
¿Desde el menú, puede salir correctamente del juego?	Si	Si	Si	Si	Si
¿Es capaz de iniciar la partida sin problema?	Si	Si	Si	Si	Si
¿Puede realizar una pausa volviendo "atrás"?	Si	Si	Si	Si	Si
¿El menú de pausa le permite volver a la partida correctamente?	Si	Si	Si	Si	Si
Ha podido probar el movimiento de la barra ¿cree que es correcto?	Si	Si	Si	Si	No
Al perder una vida ¿Puede volver a jugar si lo desea?	Si	Si	Si	Si	Si
Si cambia de aplicación ¿Al volver al juego se	Si	Si	Si	Si	Si

inició el menú de pausa?					
¿Ha podido completar todos los niveles?	Si	No	Si	No	No
¿Cree que el juego es demasiado complicado?	No	No	No	No	Si
¿Le parece divertido el juego?	Si	Si	Si	Si	Si

5.3 Conclusiones del formulario

Tras analizar las respuestas y leer los comentarios, se puede apreciar que el juego ha funcionado correctamente en todos los casos, sin detectarse fallos.

Se observa que el juego tiene una dificultad media, ya que no todos los jugadores han podido completar el juego, sin embargo, a todos les ha resultado divertido, lo que quiere decir que el juego a pesar de proponer un reto, es de agrado para todos los públicos, cumpliendo así la primera necesidad de la aplicación, divertir.

Todos los usuarios a excepción del menos experimentado han sentido cómodo el movimiento de la barra, tras analizarlo con el sujeto, se observa que es la falta de experiencia.

IV. LA WEB

1. ANÁLISIS

Uno de los objetivos que se marcaban en este proyecto, es acompañar al videojuego de una página web, como anteriormente se mencionó, la web aparte de poder descargar el videojuego para las diferentes plataformas disponibles, permitirá jugarlo, gracias a la tecnología OpenGL que Unity aporta, además de aprender con algo más de profundidad los caminos de desarrollar una web.

2. OBJETIVOS

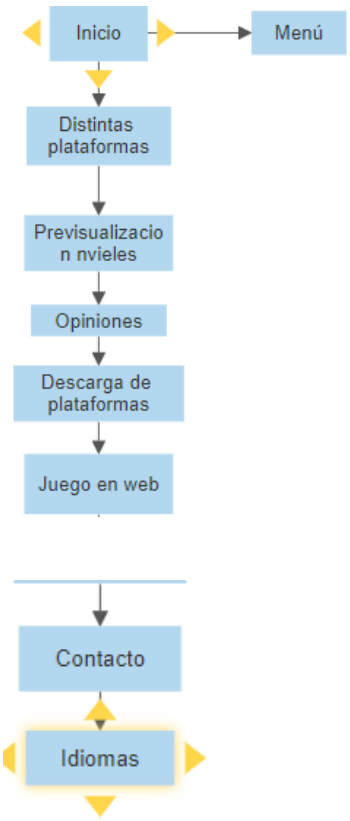
- Debe de tener una estética diferente, para que capte la atención de los usuarios, siendo una web atractiva.
- La web debe ser sencilla, sin demasiados textos y centrados en la intención, que el usuario pruebe el juego.
- La web será única, no contendrá secciones en diferentes links, si no apartados en la misma página principal.
- Contendrá un menú en la parte superior con los apartados más importantes
- Se ofrecerá la posibilidad de descargar el juego en las diferentes plataformas disponibles.
- Un apartado único para poder jugar al juego desarrollado anteriormente.
- Estará disponible tanto en inglés como en Castellano.
- Se deben de cumplir las 8 reglas de usabilidad que son orientada al usuario, rápida, funcional, fiable, optimizada, información clara, navegable y con texto sencillo.

3. DISEÑO

El lenguaje que se utilizará principalmente será HTML junto CSS, para cumplir con el requisito de idiomas se utilizará PHP.

Para poder testear la web correctamente como si estuviera alojada en un servidor se utilizará la herramienta xampp, que permite utilizar los recursos de Apache y podemos conectarnos desde localhost a ellos.

- La interfaz tendrá el siguiente diseño.

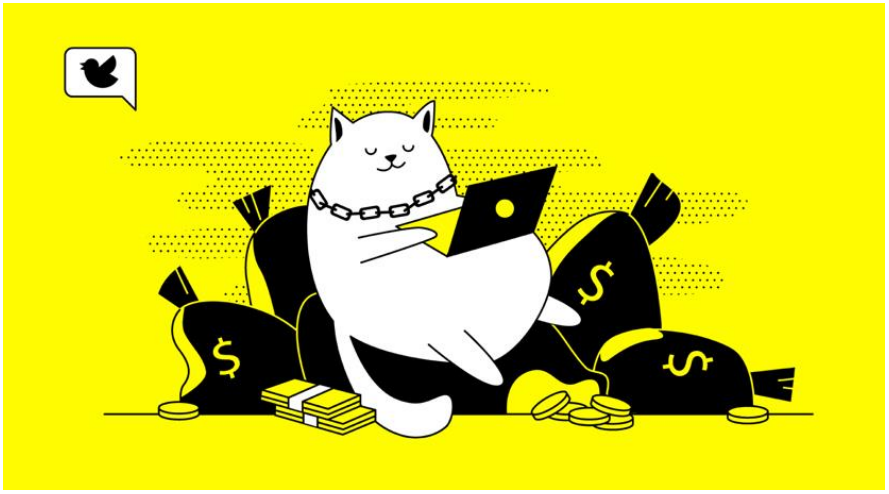


- La elección de colores se realiza bajo una exhaustiva búsqueda de colores que están de moda este 2020 para sitios web, entre ello hemos encontrado la siguiente noticia.

https://www.mercadonegro.pe/branding/diseno/8-combinaciones-de-colores-que-funcionaran-este-2020/

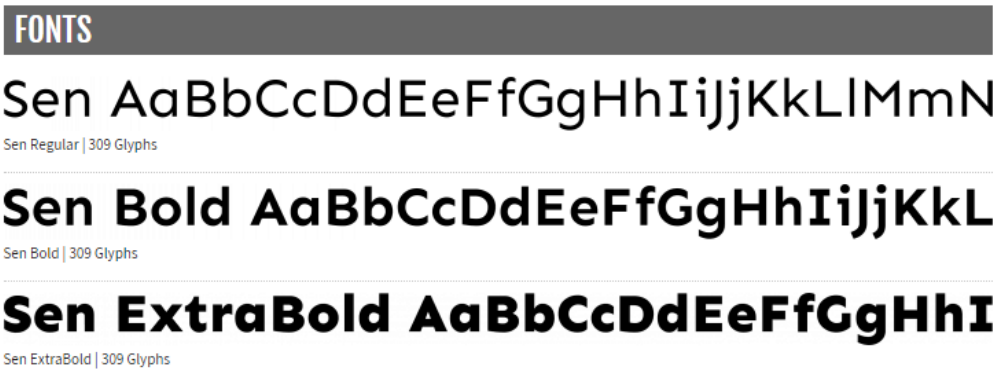
Negro y amarillo

Casi amarillo neón y negro es una combinación muy moderna en este momento que solo llama la atención. El éxito del combo de **color** se debe a la simplicidad con la que logra llamar la atención. El dúo clásico en blanco y negro se complementa con un amarillo rebelde, lo que le da a esta combinación un aspecto increíblemente nervioso.



Es por ello que negro y amarillo serán los colores que van a predominar en el sitio web.

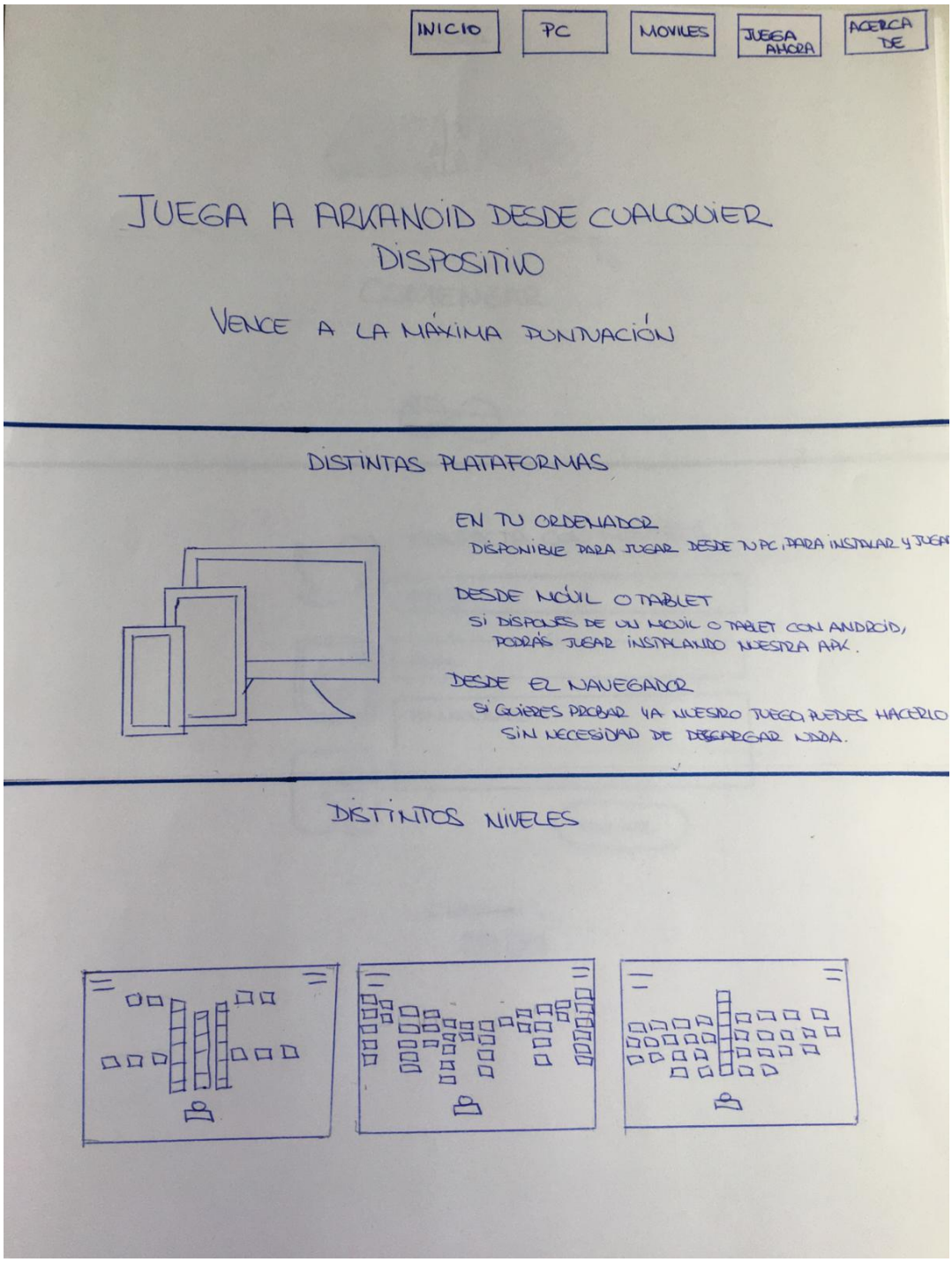
- El tipo de fuente, siguiendo el diseño minimalista de la web será “Sens”



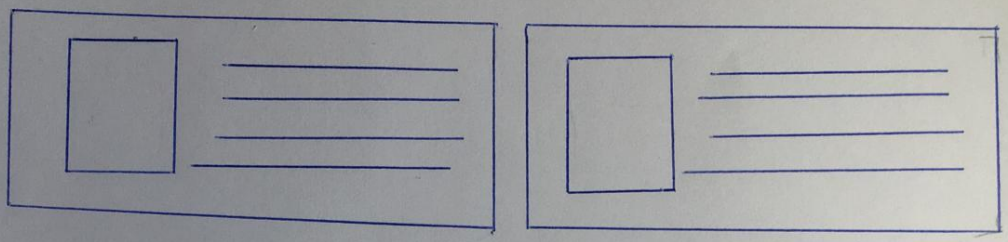
https://www.fontsquirrel.com/fonts/sen

3.1 Bocetos

Conceptualmente se realiza un diseño inicial para hacer una idea de cómo quedaría representado.


INICIO
PC
MOVILES
JUEGA AHORA
ACERCA DE
JUEGA A ARKANOID DESDE CUALQUIER DISPOSITIVO
VENCE A LA MÁXIMA PUNTUACIÓN
DISTINTAS PLATAFORMAS
EN TU ORDENADOR
DISPONIBLE PARA JUGAR DESDE TU PC, PARA INSTALAR Y JUGA
DESDE MÓVIL O TABLET
SI DISPONES DE UN MÓVIL O TABLET CON ANDROID, PODRÁS JUGAR INSTALANDO NUESTRA APK.
DESDE EL NAVEGADOR
SI QUIERES PROBAR YA NUESTRO JUEGO PUEDES HACERLO SIN NECESIDAD DE DESCARGAR NADA.
DISTINTOS NIVELES

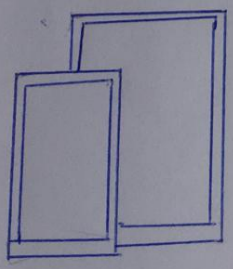
OPINIÓN DEL JUGÓN



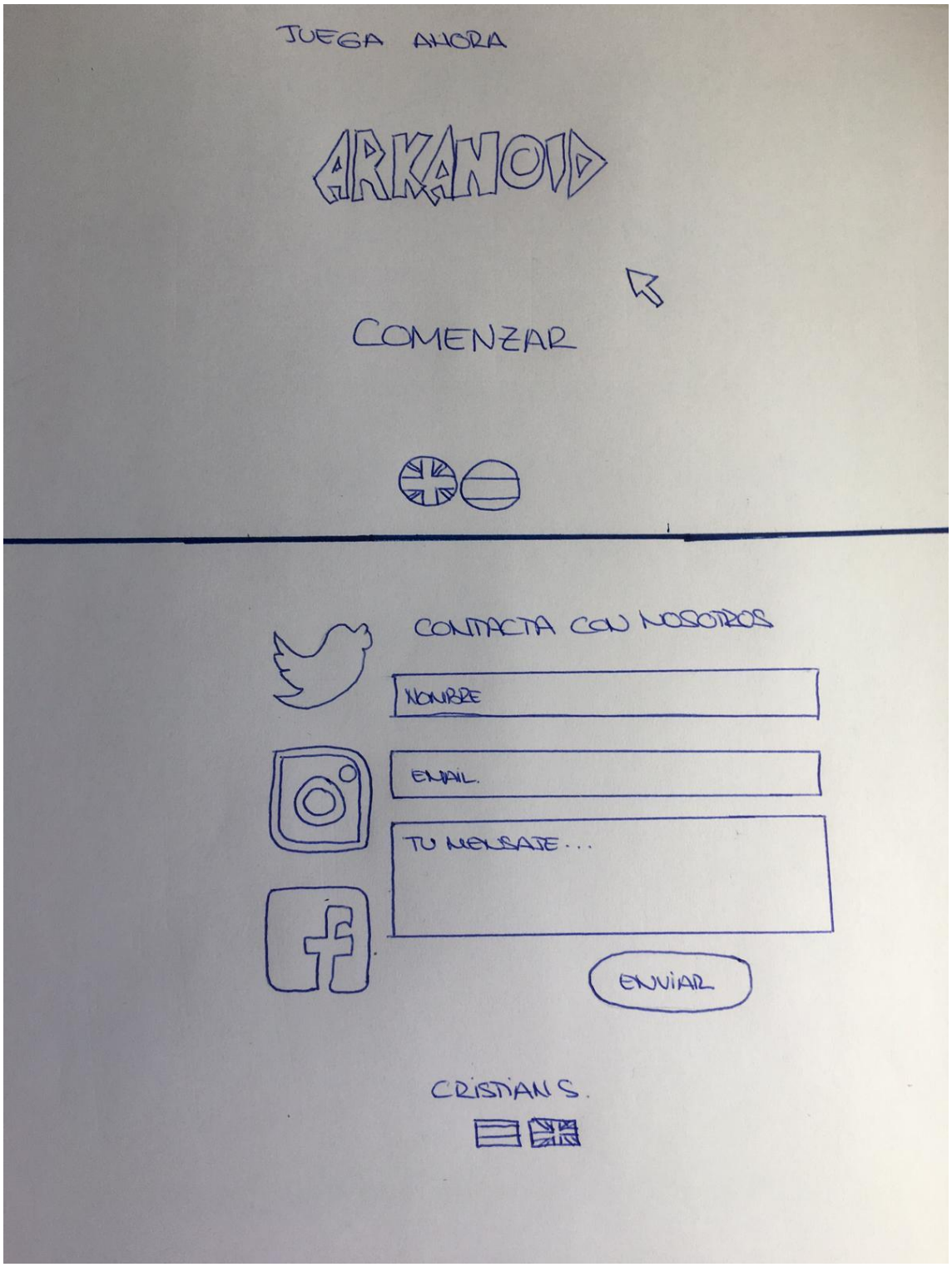
DESCARGAR PARA PC

SISTEMA OPERATIVO : WINDOWS 7, 8 O 10 (EN SU VERSIÓN DE 64-BIT)
PROCESADOR : INTEL CORE I5 @ 2.8GHZ O EQUIVALENTE EN AMD.
MEMORIA : 8GB DE MEMORIA RAM.
TARJETA GRÁFICA : NVIDIA GEFORCE GTX 660 (2GB DE VRAM)
20GB ESPACIO DISPONIBLE
DIRECTX : VERSIÓN 11.

DOWNLOAD FOR ANDROID



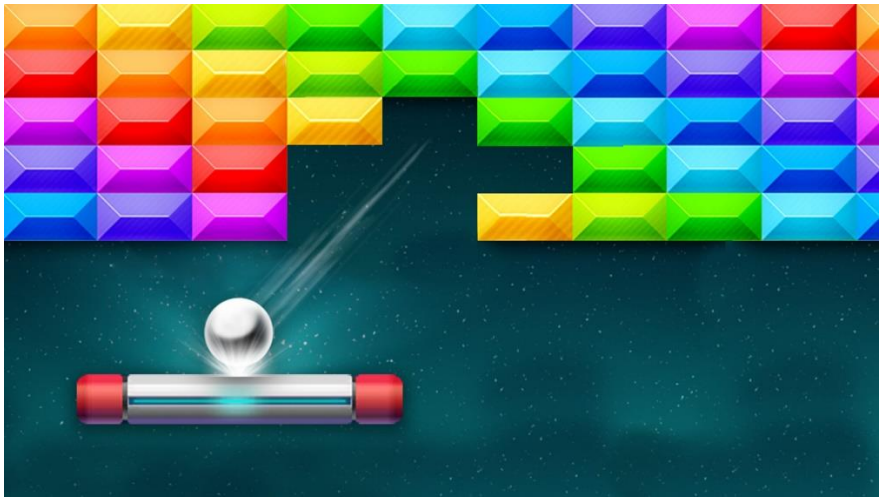
ANDROID 5.0 LOLLIPOP ONWARDS
GPU ADRENO 330, MALI-T628 MP20, MALI-G72 MP12 OR UP.
MEMORY: 3GB OF MEMORY RAM.



- En la parte superior derecha, un menú con secciones al inicio de la página, PC, Android y Jugar ahora
- Justo debajo irá un fondo representativo de Arkanoid con un par de textos.
- Explicación sobre las plataformas disponibles.
- Sección con los requisitos mínimos para PC y un link de descarga.
- Apartado con requisitos mínimos para Android y su descarga.

- Distintas opiniones de usuarios.
- Módulo para jugar al videojuego
- Contacto
- Selector de idiomas (Inglés / Español)

Para el fondo de la web será de color negro, pero en el encabezado se mostrará este fondo para que tenga un impacto inicial elegante y represente desde el primer momento a lo que va enfocado.



Al tener un fondo de color negro, la letra será de color blanca el contraste definitivo para una lectura cómoda y sin cansar a la visión.

- Se incorpora finalmente un botón para subir desde cualquier lugar de la web al inicio.

4. PROGRAMACIÓN

```
<head>
    <meta charset="UTF-8">
    <link href="https://fonts.googleapis.com/css?family=Sen&display=swap" rel="stylesheet">
    <title><?php echo $lang["titulo"] ?></title>
    <link rel="shortcut icon" href="img/logo.png">
    <link rel="stylesheet" href="css/estilos.css">
    <script src="http://code.jquery.com/jquery-latest.js"></script>
    <script src="TemplateData/UnityProgress.js"></script>
    <script src="Build/UnityLoader.js"></script>
    <script>
      var unityInstance = UnityLoader.instantiate("unityContainer", "Build/opengl.json", {onProgress: UnityProgress});
    </script>

</head>
```

4.1 Fuente

Para la fuente, utilizaremos la API de Google, por lo que debemos incluirla.

Añadimos una hoja de estilos (CSS) y para el botón de subir será necesario aplicar un script, que explicaremos más adelante. También el script necesario para ejecutar el juego en la web.

Antes del HTML será necesario agregar el código PHP para poder utilizarlo y así dar la posibilidad de usar los dos idiomas.

4.2 Idiomas

```
<?php
session_start();
if (isset($_GET["lang"]))
    {
    $lang = $_GET["lang"];
    if(!empty($lang))
        {
        $_SESSION["lang"] = $lang;
        }
    }


if (isset($_SESSION["lang"]))
    {
    $lang = $_SESSION["lang"];
    require "lang/".$lang.".php";
    }
else
    {
     require "lang/es.php";
    }
?>

<!DOCTYPE html>
<html lang="es">
```

Los idiomas se cargarán en documentos aparte, y según lo elegido se cargará en los textos, que detecta el idioma escogido y PHP se encarga de hacer la lectura y cargarlo.

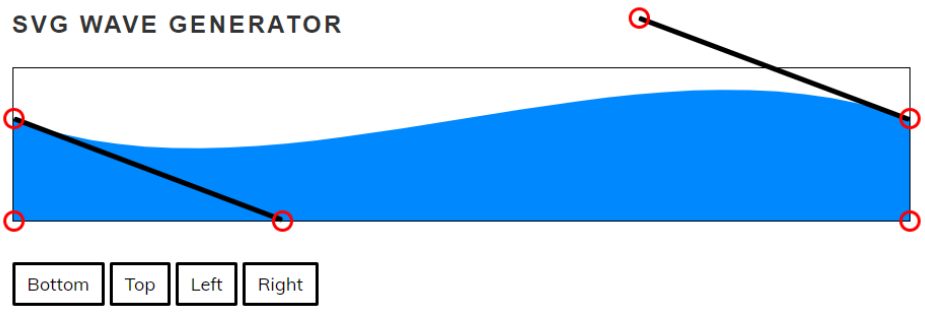
4.3 Diseño de cabecera

```
<body>
    <header>
        <nav>
            <a href="#"><?php echo $lang["inicio"] ?></a>
            <a href="#descargapc"><?php echo $lang["consigueloPC"] ?></a>
            <a href="#descargaandroid"><?php echo $lang["dispositivosmoviles"] ?></a>
            <a href="#juegaahora"><?php echo $lang["juegaahora"] ?></a>
            <a href="#contacta"><?php echo $lang["acercade"] ?></a>
        </nav>


        <section class="textoCabecera">
            <h1><?php echo $lang["juegaArkanoid"] ?></h1>
            <h2><?php echo $lang["venceMaxima"] ?></h2>
        </section>

        <span class="flechaArriba"><?php echo $lang["subir"] ?></span>
        <div class="triangulos"><svg xmlns="http://www.w3.org/2000/svg" viewBox="0 0 1440 320">
  <path fill="#2A2924" fill-opacity="1" d="M0,160L360,128L720,0L1080,64L1440,0L1440,320L1080,320L720,320L360,320L0,320Z"></path>
</svg></div>


    </header>
```

En esta parte escribimos el menú y la cabecera, después añadimos una “ola” que dará un diseño elegante para dividir la imagen de fondo con el fondo negro.

https://smooth.ie/blogs/news/svg-wavey-transitions-between-sections

Lo generamos con las propiedades que queramos.



La propia web nos genera el código a implementar.

```
<section class="contenedor distintasPlataformas">

    <h2 class="titulo"><?php echo $lang["distintasPlataformas"] ?></h2>
    <div class="contenedorDistintasPlataformas">

        <img src="img/equipos.svg" alt="" class="imagen3Plataformas">
        <div class="TextosDistintasPataformas">
            <ul>
            <li><h3><?php echo $lang["En tu ordenador"] ?></h3></li>
            <p><?php echo $lang["Disponible para jugar desde tu PC, para instalar y jugar."] ?></p>
            <li><h3><?php echo $lang["Desde móvil o tablet"] ?></h3></li>
            <p><?php echo $lang["Si dispones de un móvil o tablet con Android, podrás jugar instalando nuestra APK"] ?></
            p>
            <li><h3><?php echo $lang["Desde el navegador"] ?></h3></li>
            <p><?php echo $lang["Si quieres probar ya nuestro juego, puedes hacerlo sin necesidad de descargar nada"] ?>
            </p>
            </ul>
        </div>
    </div>
</section>
```

4.4 Sección diferentes plataformas

Esta sección se corresponde al anuncio de las diferentes plataformas, que lo acompaña una imagen.

Tres div para para mostrar diferentes capturas del juego.

```
<section id="seccioapturas">
    <div class="contenedor">
        <h2 class="titulo"><?php echo $lang["Distintos niveles"] ?></h2>
        <div class="galeriaCapturas">

            <div class="capturaIndividual">
                <img src="img/captura1.jpg" alt="">

            </div>
            <div class="capturaIndividual">
                <img src="img/captura2.jpg" alt="">

            </div>
            <div class="capturaIndividual">
                <img src="img/captura3.jpg" alt="">

            </div>
            <div class="capturaIndividual">
                <img src="img/captura4.jpg" alt="">

        </div>
    </div>
</section>
```

4.5 Sección opiniones

```
<section class="clientes contenedor">
    <h2 class="titulo"><?php echo $lang["Opinión del jugón"] ?></h2>
    <div id="CajasDetarjetas">
        <div class="ExteriorTarjeta">
            <img src="img/persona1.png" alt="">
            <div class="textoTarjeta">
                <h4><?php echo $lang["Carlos"] ?></h4>
                <p><?php echo $lang["Es un juego muy divertido y adictivo, te mantendrá durante horas enganchado"] ?></p>
            </div>
        </div>
        <div class="ExteriorTarjeta">
            <img src="img/persona2.png" alt="">
            <div class="textoTarjeta">
                <h4><?php echo $lang["Raúl"] ?></h4>
                <p><?php echo $lang["Es increiblemente bueno, sin duda, el mejor Arkanoid que he jugado"] ?></p>
            </div>
        </div>
    </div>
</section>
```

Las fichas de opiniones, acompañadas de una foto de personas aleatorias.

4.6 Requisitos

```
<section class="contenedor lasPlataformas">
    <h2 class="titulo" id="descargapc"><?php echo $lang["Descargar para PC"] ?></h2>
    <div class="plataforma">
        <img src="img/pc.svg" alt="" class="imagen-plataforma">
        <div class="textolistaPlataforma">
                <ul>
                    <li><?php echo $lang["Sistema Operativo: Windows 7, 8 o 10 (en su versión de 64-bit)"] ?></li>
                    <br>
                    <li><?php echo $lang["Procesador: Intel Core i5 @ 2.8 GHz o equivalente en AMD."] ?></li>
                    <br>
                    <li><?php echo $lang["Memoria: 8 GB de memoria RAM."] ?></li>
                    <br>
                    <li><?php echo $lang["Tarjeta gráfica: Nvidia GeForce GTX 660 (2GB de VRAM)"] ?></li>
                    <br>
                    <li><?php echo $lang["20GB de espacio disponible."] ?></li>
                    <br>
                    <li><?php echo $lang["DirectX: Versión 11."] ?></li>
                </ul>
        </div>
    </div>
    <div class="plataformaDescarga">
        <h1><?php echo $lang["Descargar"] ?></h1>
        <img src="img/descargar.svg" alt="" class="imagen-descargar">
    </div>
</section>
```

```
<section class="contenedor lasPlataformas">
    <h2 class="titulo" id="descargaandroid"><?php echo $lang["Descargar para Android"] ?></h2>
    <div class="plataforma">
        <img src="img/phonetablet.svg" alt="" class="imagen-plataforma">
        <div class="textolistaPlataforma">
                <ul>
                    <li><?php echo $lang["Android 5.0 Lollipop en adelante"] ?></li>
                    <br>
                    <li><?php echo $lang["GPU Adreno 530, Mali-G71 MP20, Mali-G72 MP12 o superior"] ?></li>
                    <br>
                    <li><?php echo $lang["Memoria: 3 GB de memoria RAM."] ?></li>

                </ul>
        </div>
    </div>
    <div class="plataformaDescarga">
        <h1><?php echo $lang["Descargar"] ?></h1>
        <img src="img/descargar.svg" alt="" class="imagen-descargar">
    </div>
</section>
```

Los requisitos y descarga de las diferentes plataformas, también acompañadas de una imagen.

```
  <section class="contenedor principal-jugarweb">
      <h2 class="titulo" id="juegaahora"><?php echo $lang["Juega ahora"] ?></h2>
      <div class="contenedor-juego">
      <div class="webgl-content">
<div id="unityContainer" style="width: 340px; height: 560px"></div>
</div>

      </div>
  </section>
```

4.7 Jugar en web

La sección para poder jugar directamente en web, el div lo genera automáticamente Unity, escogemos una resolución idónea para el juego.

```
      <section class="contenedor principal-jugarweb">
        <h2 class="titulo" id="juegaahora"><?php echo $lang["Juega ahora"] ?></h2>
          <div class="contenedor-juego">
          <div class="webgl-content">
    <div id="unityContainer" style="width: 960px; height: 640px"></div>
    <div class="footer">
      <div class="webgl-logo"></div>
      <div class="fullscreen" onclick="unityInstance.SetFullscreen(1)"></div>
      <div class="title">Arkanoid</div>
    </div>
  </div>

          </div>
        </section>
```

4.8 Footer

```
<footer>
    <div id="footer">
        <div class="socialMedia" id="contacta">

            <div class="iconoFooter">
                <a href="#"><img src="img/twitter.svg" alt=""></a>
            </div>
            <div class="iconoFooter">
            <a href="#"> <img src="img/instagram.svg" alt=""></a>
            </div>
            <div class="iconoFooter">
            <a href="#">  <img src="img/facebook.svg" alt=""></a>
            </div>
        </div>
            <div class="cajaContacto">
                <h4> <?php echo $lang["Contacta con nosotros"] ?></h4>
            <br>

                <form class="formularioContacto" action="index.html" method="post">
                 <input type="text" class="contactoCaja" placeholder="<?php echo $lang["Nombre"] ?>">

                <input type="email" class="contactoCaja" placeholder="<?php echo $lang["email"] ?>">

                <textarea type="email" class="contactoCaja" placeholder="<?php echo $lang["Tu mensaje..."] ?>"></textarea>

                <input type="submit" class="botonCaja" value="<?php echo $lang["Enviar"] ?>">
                </form>
        </div>

    </div>
```

El footer contiene el formulario de contacto y las redes sociales disponibles, también finalmente se incluye el selector de idiomas.

```
    <div class="nombreFooter">
        <h2>Cristian S.</h2>
    </div>
    <div class="idiomas">
        <a href="index.php?lang=es"><img src="img/spain.png"></img></a>
        <a href="index.php?lang=en"><img src="img/uk.png"></img></a>
    </div>
</footer>
```

4.9 Flecha arriba

El script se encarga sobre el botón de desplazamiento hacia arriba, para ello hemos cogido el Script del siguiente tutorial.

http://www.falconmasters.com/web-design/boton-ir-arriba-javascript/

```
<script>
 var x = document.getElementsByClassName("capturaIndividual");
</script>

<script>
    $(document).ready(function(){

    $('.flechaArriba').click(function(){
        $('body, html').animate({
            scrollTop: '0px'
        }, 300);
    });

    $(window).scroll(function(){
        if( $(this).scrollTop() > 0 ){
            $('.flechaArriba').slideDown(300);
        } else {
            $('.flechaArriba').slideUp(300);
        }
    });

});
</script>
```

4.10 CSS

```
* {
    margin: 0;
    padding: 0;
    box-sizing: border-box; /*El padding crezca hacia dentro*/
}

body {
    font-family: 'Sen', sans-serif; /*Fuente sacada de fonts google api*/
   background-color: #2A2924;

}

.flechaArriba {
    display:none;
    padding:10px;
    background:hsla(45, 98%, 37%, 0.459);
    font-size:20px;
    color:#fff;
    cursor:pointer;
    position: fixed;
    bottom:20px;
    right:20px;
}

.contenedor {
    padding: 60px 0; /*espacio entre contenedores*/
    width: 90%;
    max-width: 1000px;
    margin: auto; /*todo centrado*/
    overflow: hidden; /*Lo que se quede fuera del contenedor, oculto*/
}
```

Aplicamos al todo el documento el tipo de fuente que escogimos, Sen, como segunda opción es sans-serif y el color, un tono muy cercano al negro absoluto.

Al botón le damos un diseño cuadrado con unos tonos amarillos, dando como resultado.


```css
/********************* Cabecera *********************/

header {
    width: 100%; /*Ancho y alto y el fondo sacado de uigradients que será gradiente y pasado a hsla para tener transparencia*/
    height: 600px;
    background: #bc4e9c;
    /* fallback for old browsers */
    background: -webkit-linear-gradient(to right, hsla(45, 98%, 37%, 0.459), hsla(0, 0%, 0%, 0.664)), url(../img/portada.jpeg);
    /* Chrome 10-25, Safari 5.1-6 */
    background: linear-gradient(to right, hsla(45, 98%, 37%, 0.459), hsla(0, 0%, 0%, 0.664)), url(../img/portada.jpeg);
    /* W3C, IE 10+/ Edge, Firefox 16+, Chrome 26+, Opera 12+, Safari 7+ */

    background-attachment: fixed; /*Al hacer scroll, la imagen se mantiene, no se mueve con el contenedor*/
    position: relative;
}
```

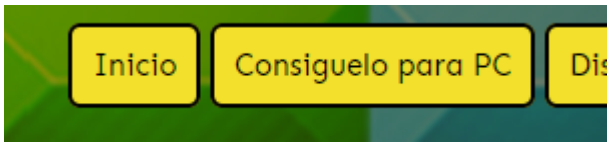
La cabecera, con una altura y anchura determinada, contiene un gradiente amarillo a negro sacado de https://uigradients.com/#DarkKnight , lo trasformamos a HSLA para poder añadirle una trasparencia y adjuntamos el fondo mencionado anteriormente.

La transformación de Hexadecimal a HSLA lo hacemos desde la siguiente web: http://w3.unpocodetodo.info/utiles/conversor.php

La parte más importante para darle juego al fondo es el background-attacment al decirle que esa fixed el fondo se queda fijo y no se mueve mientras lo hace la web, dando una senación de profundidad.

```css
nav{ /*El nav alineado a la derecha con un padding izq y abajo*/
    text-align: right;
    padding: 30px 50px 0 0;
}



nav > a{
    text-decoration: none;
    padding: 10px;
    font-weight: 500;
    font-size: 15px;
    color: black;
    background-color: #f2e02c;
    border-radius: 6px;
    border: 2px solid black;
}

nav > a:hover{ /*Al pasar el ratón encima le ponemos el subrayado y efecto sombreado de luz*/
    text-decoration: underline;
    color: rgba(255, 255, 255, 1) !important;
    box-shadow: 0 4px 16px rgba(242, 224, 44, 1);
    transition: all 0.2s ease;
}
```

El estilo del nav, que será el menú, aplicamos diferentes según sea la situación, en este caso le quitamos los filtros por defecto y le añadimos un diseño elegante, al pasar el ratón por encima (hover) cambiará añadiendo una sombra.




```css
header .textoCabecera{ /*Se comporte según el tamaño de la pantalla*/
    display: flex;
    flex-direction: column;
    height: 430px;
    width: 100%;
     /*Todo centrado*/
    align-items: center;
    justify-content: center;


}
/*Estilos al h1 y h2 que están dentro del header*/
.textoCabecera h1{
    font-size: 55px;
    color:white;
}

.textoCabecera h2{
    font-size: 35px;
    color:white;
}
```

El texto que contiene el header, con un h1 y un h2, lo centraremos y le daremos diferentes tamaños de texto.

Justo debajo irá la “ola” que diseñamos, serán unos picos.

```css
.triangulos{  /*Diseño a los tri
    position: absolute;
    overflow: hidden;
    width: 100%;
    height: 150px;
}
```

```css
.contenedorDistintasPlataformas{
    display: flex;
    justify-content: space-evenly;  /*Reparte el espacio del mismo tamaño pa
}

.imagen3Plataformas{
    width: 30%;
}

.TextosDistintasPataformas{
    width: 48%;
}

.TextosDistintasPataformas h3{
    margin-bottom: 15px; /*Separación entre el titulo y el texto*/
    color:white;
}


.TextosDistintasPataformas ul{
    padding: 0px 0px 30px 15px;
    list-style:none;

}

.TextosDistintasPataformas p{ /*El "cuerpo"/texto que viene después*/
    padding: 0px 0px 30px 15px;
    font-weight: 300;
    text-align: justify;
    color: white;
}
```

La primera sección que habla de las distintas plataformas, será una imagen a la izquierda seguido de tres textos a la derecha, con un título y una definición del mismo.


```
/*********************  Capturas ********************/


#seccionCapturas{
    background: #44423A;

}

.galeriaCapturas{
    display: flex;
    justify-content: space-evenly;
    flex-wrap: wrap; /*Separamos las imágenes*/
}

.capturaIndividual{
    width: 24%;
    margin-bottom: 10px;
    overflow: hidden;
    position: relative;
   /* cursor: pointer; /*Cambiamos el cursor a una manita*/
    box-shadow: 0 0 6px 0 rgba(0, 0, 0, .5); /*Sombreado y borders a cada contenedor de imagen*/
}

.capturaIndividual > img{
    width: 100%;
    height: 100%;
    object-fit: cover;
    display: block;
    overflow:hidden;
}

.capturaIndividual > img:hover {
    filter: grayscale(100%);
}
```

La sección de capturas con un display flex para mostrarlos ordenadamente en una misma línea y con una separación como dicta el space-evenly.

Cuando se pase el ratón por encima, como indica el hover, pasaremos a escala de grises.

Las tarjetas de opiniones, contienen un borde redondeado, serán amarillas y dentro contiene una imagen en pequeño a la izquierda, con borde normal.

```css
/********************* Tarjetas ********************/

#CajasDetarjetas{
    display: flex;
    justify-content: space-evenly;


}

.ExteriorTarjeta{
    background: #f2e02c;
    display: flex;
    width: 46%;
    height: 200px;
    align-items: center;
    justify-content: space-evenly;
    border-radius: 15px;
    box-shadow: 0 0 6px 0 rgba(0, 0, 0, 0.6);


}

.ExteriorTarjeta img{
    width: 100px;
    height: 100px;
    object-fit: cover;
    border: 1px solid black;
    border-radius: 0%;
    display: inline-block;

}
```

```css
/********************* Plataformas ********************/

.lasPlataformas{
    padding: 30px 0 20px 0;
    background-color: #44423A;

     /*Espacio que hay arriba y abajo en este contenedor*/
}
.plataforma{

    display: flex;
    justify-content: space-evenly;
    align-items: center;  /*Reparte el espacio del mismo tamaño para todo*/
}

.imagen-plataforma{
    width: 30%;

}


.textolistaPlataforma ul { /*El "cuerpo"/texto que viene después*/
    padding: 0px 0px 30px 15px;
    font-weight: 300;
    text-align: justify;
    list-style:none;
    color: white;
}
```

```css
.textolistaPlataforma ul { /*El "cuerpo"/texto que viene después*/
    padding: 0px 0px 30px 15px;
    font-weight: 300;
    text-align: justify;
    list-style:none;
    color: white;
}

.plataformaDescarga{
    display: flex;
    width: 100%;
    border-top: yellow 1px solid;
    justify-content: center; /*centramos*/
    align-items: center;/*centramos*/
     /*Se divide el contenido uno debajo de otro*/
     padding-top: 2%;
}
.plataformaDescarga h1{
    color: white;
    text-transform: uppercase;
}

.plataformaDescarga img{
    width: 10%;
    margin-left: 5%;
    overflow: hidden;

}
```

Las secciones de plataformas tanto de pc como de Android siguen un mismo estilo, una imagen a la izquierda, con un texto a la derecha en forma de lista, pero sin diseño. Justo debajo la parte de descarga. El contenedor contendrá una escala de gris más claro al fondo.


```css
/********************* Juega ahora *********************/

main .principal-jugarweb{
    padding: 60px 0 60px 0;
     /*Espacio que hay arriba y abajo en este contenedor*/
}
.contenedor-juego{
    display: flex;
    justify-content: space-evenly;
    align-items: center;  /*Reparte el espacio del mismo tamaño para todo*/
    color:white;
}
```

El contenedor para jugar es bastante sencillo, incluimos un padding para su separación de sección y lo centramos.

```css
footer{
    background: #44423A;
    padding: 30px 0 40px 0;
    margin: auto;

}

#footer {
    display: flex;
    width: 400px;
    /* justify-content: space-evenly; */
    margin: auto;
    padding-bottom: 20px;

}

.socialMedia {
    display: flex;
    flex-wrap: wrap;
    padding-right: 4%;
    width: 25%;
    align-items: center;
}

.iconoFooter{
    text-align: center;
    width:100%;

}

.iconoFooter img{
    width:100%;
}

.nombreFooter{
    border-top: 2px solid #ccc;

}
.nombreFooter h2{
    padding-top: 2%;
    text-align: center;
    font-size: 17px;
    color: white;
}
```

```css
.cajaContacto{

    width:80%;

}
.cajaContacto h4{
    padding: 10px;
    color:white;
}

.formularioContacto{
    margin:auto;
    overflow: hidden;
}

.contactoCaja{
    display: block;
    width:100%;
    box-sizing: border-box;
    margin: 16px 0;
    padding:10px 40px;
    border:0;
    background:#111;
    outline: none;
    color:white;
    transition: 0.5s;
}

textarea.contactoCaja{
    resize: none;
    height: 100px;
}

.botonCaja{
    float: right;
    border:0;
    background:#111;
    color:#fff;
    padding: 12px 50px;
    border-radius: 20px;
    cursor: pointer;
```

En el footer irá la caja de contacto y el selector de idiomas

Centramos el contenido y le damos forma a las imágenes, irán una debajo de otra y a la derecha irá el cuadro de formulario

La caja de contacto, tendrá un diseño al mismo estilo del resto de la web, simple y minimalista. Sin bordes con el fondo en negro y el texto en blanco. Al botón enviar le asignamos una diferente forma del ratón, para que se diferencia de que se trata de un botón.

Finalmente, los idiomas estarán en pequeño en la parte final del a web, por si el usuario quiere cambiar el idioma.

```css
.idiomas{

    padding:15px;
    text-align:center;

}


.idiomas a img{
    width: 27px;
    height:20px;
    background-color:red;

}
```

El diccionario de idiomas lo contienen dos archivos, uno por cada idioma. El funcionamiento es sencillo, se detecta la palabra clave y se carga según el idioma seleccionado.

```php
<?php

$lang = array (
    "titulo" => "Arkanoid",
    "inicio" => "Home",
    "consigueloPC" => "Get it for PC",
    "dispositivosmoviles" => "Get it for Andr
    "juegaahora" => "Play now",
    "acercade" => "About us",
    "juegaArkanoid" => "Play Arkanoid from an
    "venceMaxima" => "Beat the highest score",
    "subir" => "SUBIR",
```

```php
$lang = array (
    "titulo" => "Arkanoid",
    "inicio" => "Inicio",
    "consigueloPC" => "Consiguelo para PC",
    "dispositivosmoviles" => "Dispositivos móviles"
    "juegaahora" => "Juega ahora",
    "acercade" => "Acerca de",
    "juegaArkanoid" => "Juega a Arkanoid desde cual
    "venceMaxima" => "Vence a la máxima puntuación"
    "subir" => "SUBIR",
```

V. CONCLUSIONES FINALES

Una de las grandes metas de este proyecto, como indiqué inicialmente era poder trasformar mi pasión de videojuegos, conociendo lo que significar desarrollar uno en todas sus fases, finalmente he podido cumplirlo.

Cabe destacar que uno se da cuenta que te gusten los videojuegos no implica que hacer uno vaya a ser de agrado para todos, ya que implica muchísimo esfuerzo, aprendizaje y constancia, cosa que puede llevar a una idea muy equivocada inicialmente.

También me ha servido para desarrollarme en otros lenguajes que conocía menos como puede ser C# o incluso que desconocía al completo como PHP, me doy cuenta que aun que cada uno tiene sus particulares las bases son prácticamente las mismas, aplicar la lógica que es lo que más he aprendido durante todo este tiempo.

Desarrollar un videojuego nunca ha fue tan "fácil" lo que Unity nos proporciona está a un nivel que antes de su existencia ni imaginaría y aun que no es tarea ni mucho menos fácil puede llegar a conseguir que una sola persona pueda desarrollar un videojuego a la altura que quiera invirtiendo el tiempo que se necesitará en él.

Me queda claro, por tanto, que a partir de ahora mi próximo objetivo se centrará en realizar uno con mucha más ambición, que algún día pueda ser publicado en diferentes plataformas / videoconsolas.

También aprender nuevos lenguajes será uno de mis próximos objetivos, puesto que encuentro que estudiarlos me resulta muy motivador para encontrar nuevos retos.

VI. BIBLIOGRAFÍA

- **Recurso de idiomas**: https://www.youtube.com/watch?v=a9ZNjjptfjg
- **Triángulos para la web** : https://smooth.ie/blogs/news/svg-wavey-transitions-between-sections **Y** https://www.getwaves.io/
- **Colores**: https://uigradients.com/#DarkKnight
- **Srpites Arkanoid:** https://www.spriters-resource.com/mobile/arkanoidvsspaceinvaders/sheet/115280/?source=genre
- **Botón subir dentro de la web** : http://www.falconmasters.com/web-design/boton-ir-arriba-javascript/
- **Sonidos**: https://www.sounds-resource.com/nes/arkanoid/sound/3698/
- **Iconos:** https://www.flaticon.es/ y https://freeicons.io/
- **Recursos utilizados para la programación de Arkanoid:**
  https://www.youtube.com/watch?v=jNqkS10DXOg
  https://www.youtube.com/watch?v=v_aL7rOCCMU
  https://www.youtube.com/watch?v=fVeLWGsdFn0
- **Lectura de textos C#:**
  https://docs.unity3d.com/ScriptReference/Resources.Load.html
  http://decodigo.com/c-sharp-leer-archivo-de-texto