|  |
| --- |
| cifp Juan de colonia |
| Desarrollo de interfaces |
| Proyecto Final 1 Evaluación |
|  |
| **Alexis López Briongos Dam2t** |
| **29/10/2023** |

|  |
| --- |
|  |

*Índice*

[1.- Introducción 2](#_Toc150540451)

[2.- Investigación 2](#_Toc150540452)

[3.- Construcción del proyecto 4](#_Toc150540453)

[3.1 Diseño de interfaces 6](#_Toc150540454)

[3.2- Idea del juego 8](#_Toc150540455)

[3.3 Lógica del juego 9](#_Toc150540456)

[4. - Conclusiones 20](#_Toc150540457)

# 1.- Introducción

1. **Título del Proyecto**: Ordena tó
2. **Descripción del Proyecto**. Consiste en un juego didáctico en el cual su finalidad es que el usuario que lo juegue sepa identificar los componentes hardware de un ordenador y sepa colocarlos en sus zonas correspondientes para completar el juego.
3. **Objetivos**: El objetivo de este juego es poder fomentar las habilidades cognitivas y el aprendizaje de la informática como este ejemplo de componentes hardware (y aprobar).
4. **Público Objetivo**: El público objetivo está destinado a usuarios a niños entre 7 y 12 años.
5. **Justificación**: Este tipo de prácticas es beneficioso para abordar la necesidad educativa en cuanto la informática mediante dinámicas entretenidas como un videojuego.

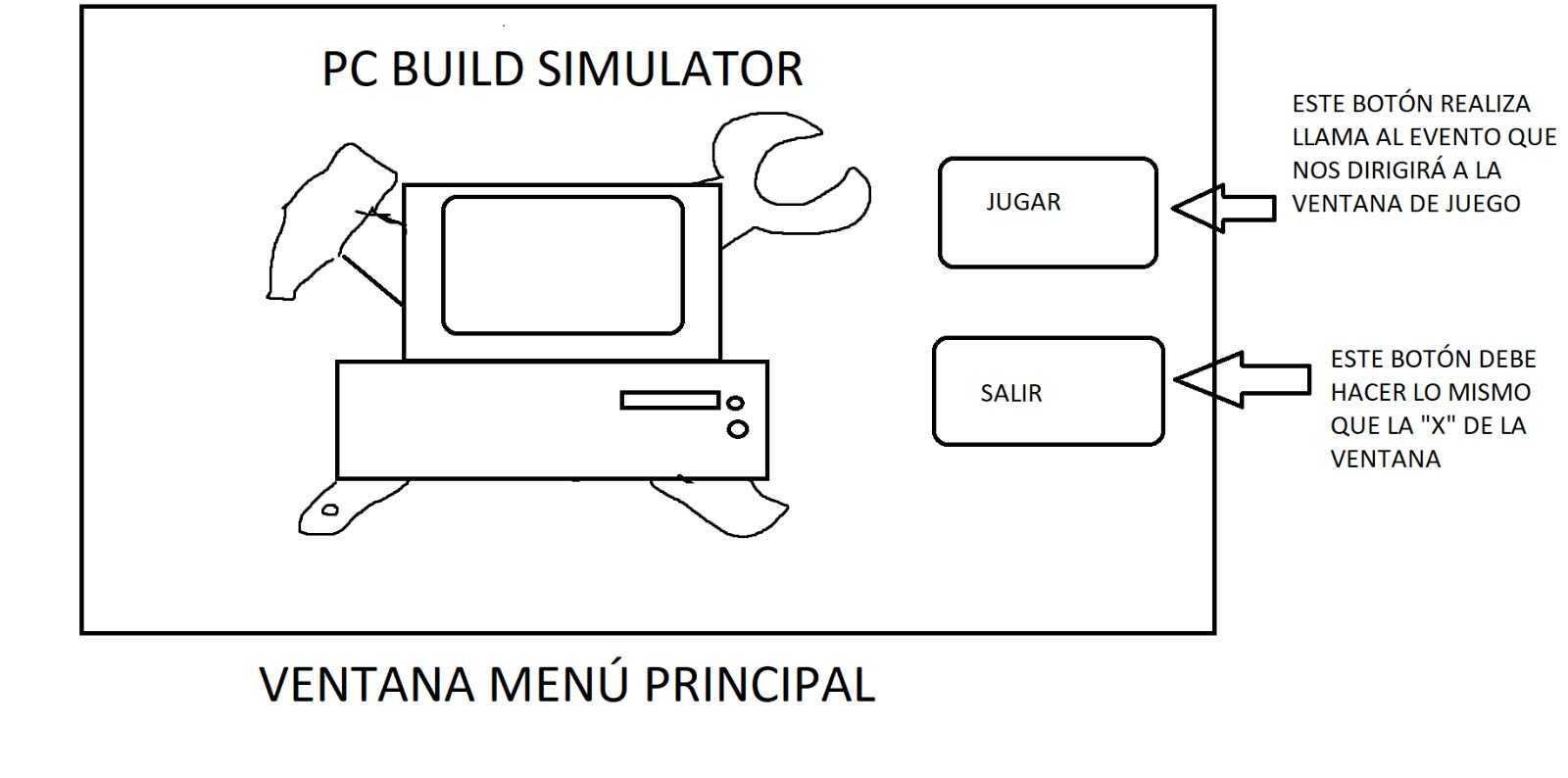
# 2.- Investigación

1. **Estudio de Mercado**: A continuación presentaremos algunos ejemplos de productos similares y tendencias del mercado.
   1. **Análisis de la Competencia**: Enumera y describe los juegos o productos similares que ya existen en el mercado. Destaca sus características, ventajas y desventajas. El juego más exitoso de este género que podemos encontrar en diferentes plataformas como Steam, Epic Games entre otros es el PC Building Simulator.
      * 1. **Características**: El juego está orientado a usuarios que estén familiarizados en el mundo de la informática y de los videojuegos. Tiene gráficos 3D y debes construir ordenadores satisfaciendo las necesidades de diversos clientes.
        2. **Ventajas**: Es un videojuego con una jugabilidad y apariencia atractiva. Una profundidad en la que debes ajustar presupuestos con los componentes y las peticiones de los clientes. Soporte de múltiples plataformas con sistemas de logros.
        3. **Desventajas**: El producto tiene de coste alrededor de 20€ (aunque se puede conseguir más barato en diferentes plataformas de compra de videojuegos). Necesitas unos previos conocimientos en cuanto a videojuegos y los componentes necesarios para adaptarse a los requisitos de los clientes.
   2. **Precios en el Mercado**: Precios de alrededor de 20€ en plataformas como Steam y Epic Games.
   3. **Demanda del Mercado**: Estos productos están desarrollados por empresas que buscan generar dinero y mi objetivo es proporcionar una herramienta didáctica en la que todo tipo de usuarios pueda aprender y divertirse aprendiendo.
   4. **Tendencias del Mercado**: Las nuevas dinámicas de aprendizaje están en auje como podrían ser los videojuegos didácticos y pudieran satisfacer una parte del mercado educativo para estos fines.
2. **Diferenciación de tu Proyecto**: A continuación expondremos todas las características, ventajas y desventajas de nuestro videojuego y como puede diferenciarse de la competencia.
   1. **Características**: Es un videojuego orientado a gente que no tiene la habilidad de jugar a videojuegos pero si quieren herramientas para poder aprender de una forma más entretenida y didáctica sobre el mundo de la informática.
   2. **Ventajas**: Es sencillo de jugar, interfaz intuitiva, con tener conocimientos básicos de controlar los periféricos de un ordenador, como el ratón y es totalmente gratuito.
   3. **Desventajas:** El apartado gráfico es muy simple y no tiene gran profundidad al estar orientado a niños o personas sin la suficiente habilidad de jugar a videojuegos o la informática en sí.

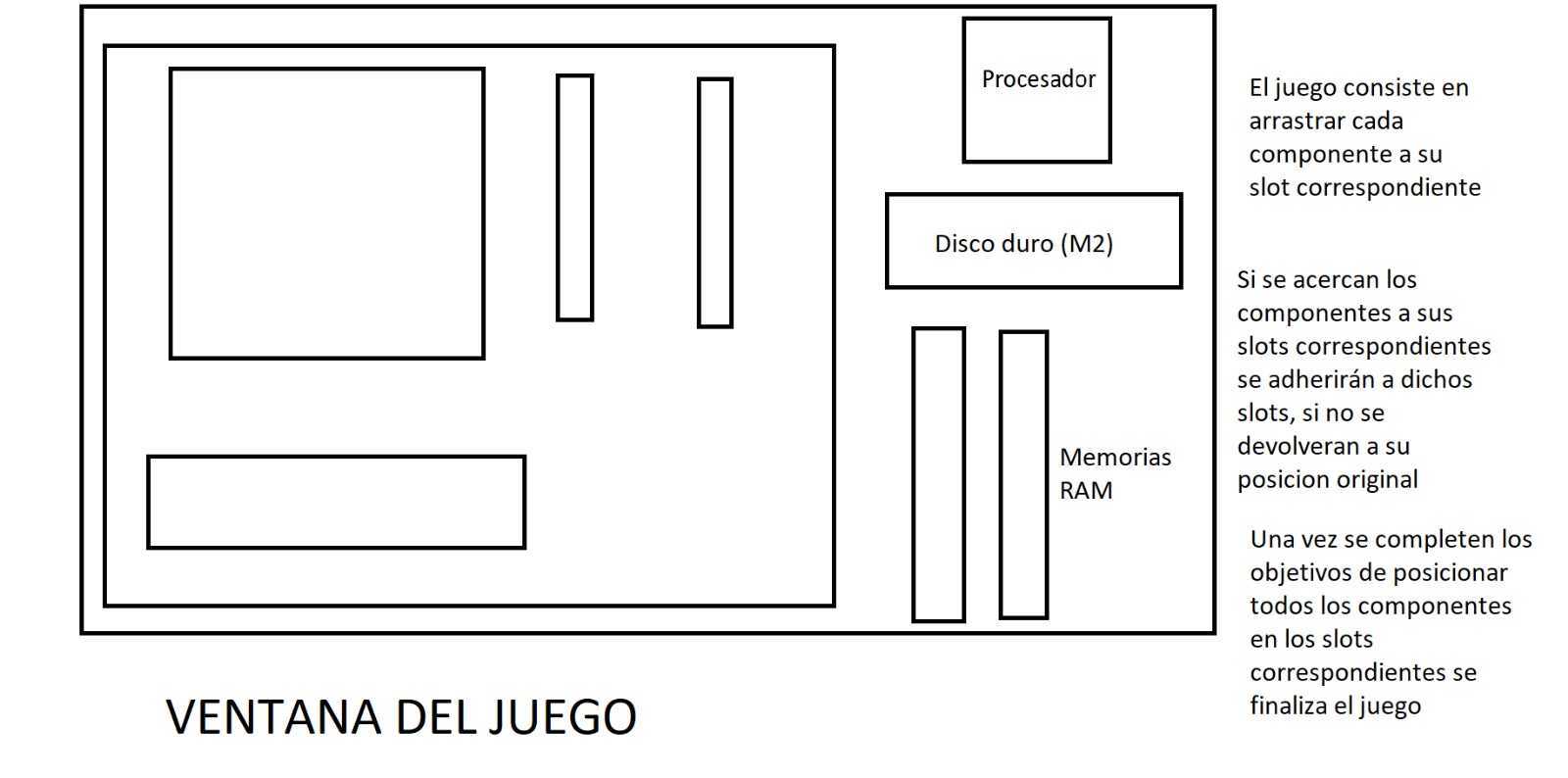
# 3.- Construcción del proyecto

1. **Boceto**: este es la primera actualización del proyecto en el que he elaborado un boceto básico (está sujeto a cambios) en el que muestro la Ventana principal y la Ventana de juego

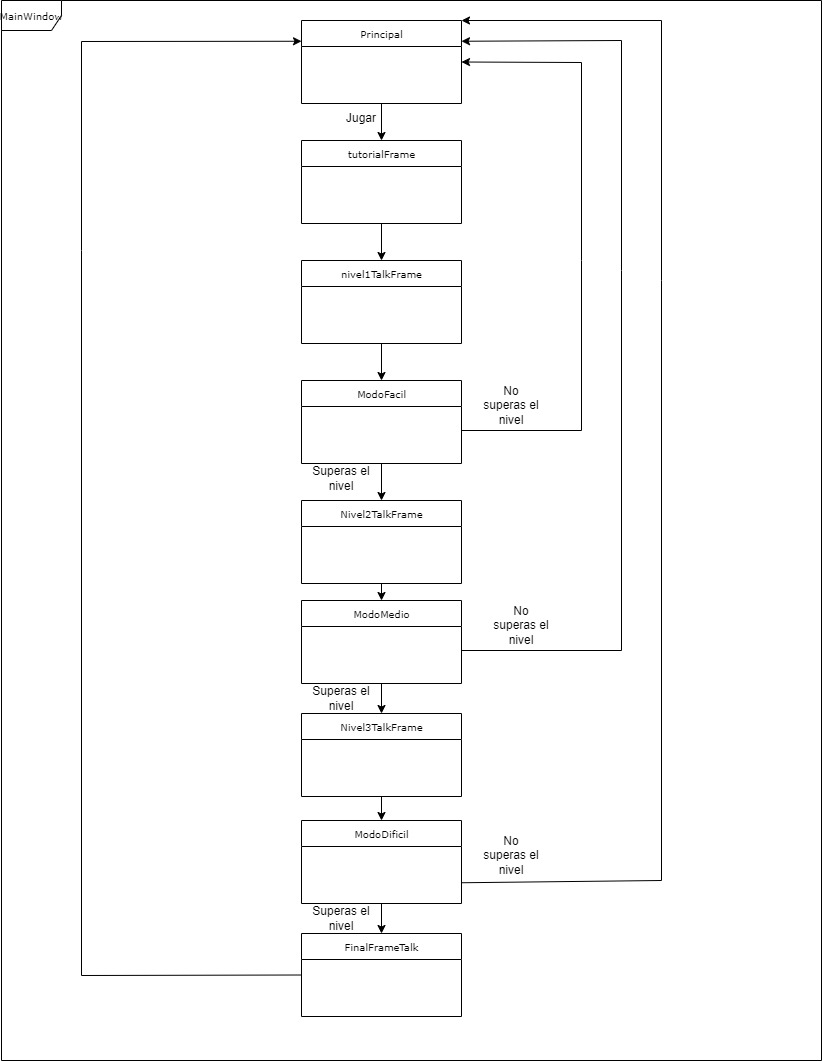
**Ventana Principal:**



**Ventana Juego:**

****

1. **Estructuración de la aplicación:**



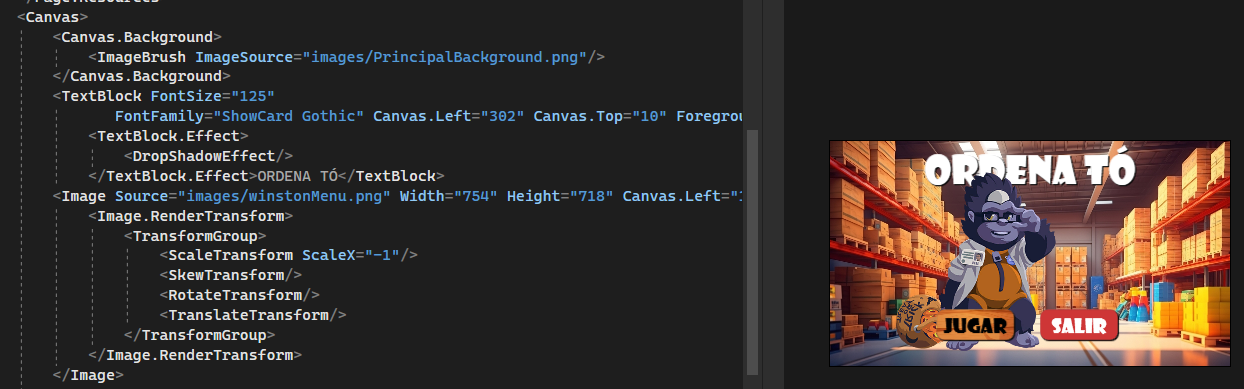
*Los contenedores grandes representan ventanas y los de su interior páginas xml. Las páginas denominadas ModoFacil, ModoMedio y ModoDificil representan los niveles 1 ,2 y 3. Están así denominadas por el anterior diseño del juego.*

* La aplicación se va a ejecutar sobre la misma ventana Mainwindow.
* En la ventana MainWindow es donde ocurrirá todo el bucle del videojuego. Estará formado por páginas para que haya una transición limpia entre niveles y estados del videojuego.

# 3.1 Diseño de interfaces

**CANVAS**

* Para poder rastrear posiciones, tamaños y dibujar en tiempo real elementos en la pantalla en este caso he utilizado el control **Canvas**.

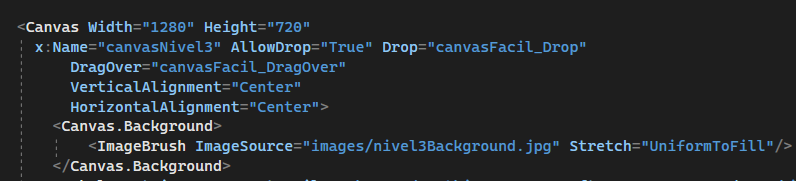


*Ejemplo del documento XAML del menú principal en el que podemos visualizar la implementación del control* ***Canvas****.*

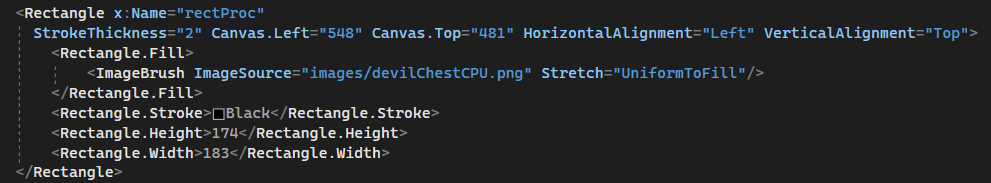
* Este control es clave para nuestro proyecto ya que contiene los métodos **Drop** y **Drag\_over**:
  + **Drag\_over**: este método permite ver el recorrido en tiempo real que hace el objeto cuando lo coges y lo desplazas dentro del Canvas.
  + **Drop**: este método sirve para controlar la posición en la que se queda el objeto cuando liberas el botón del ratón.
* Bien es cierto que estos métodos pueden estar en otros objetos también pero su rendimiento y su funcionamiento no es lo más recomendado para un videojuego, ya que si desplazas rápido el cursor y el cursor sale del objeto se deseleccionaría, teniendo que volver a pulsar sobre el objeto para arrastrar interrumpiendo por completo la mecánica y empeorando la experiencia del usuario.

**IMAGE**

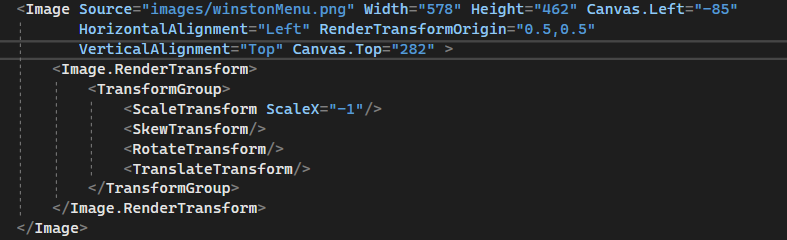
* Son objetos que utilizamos para representar a los personajes, escenarios, componentes, cajas, etc.
* Podemos implementarlos bien sea como una imagen con su ruta relativa o absoluta o dentro de un rectángulo para poder controlar sus colisiones de una forma más cómoda y sencilla.

****

*Ejemplo de implementación de una imagen en el control Canvas para establecerla como fondo del escenario.*



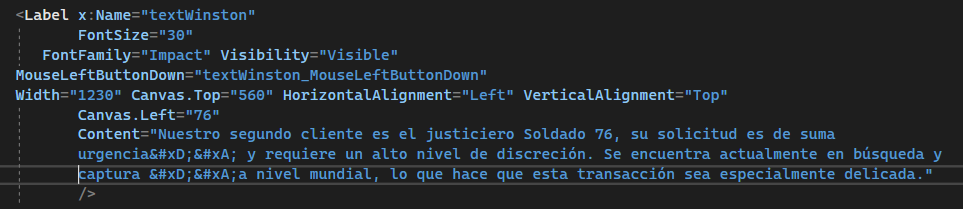
*Ejemplo de implementación de una imagen dentro de un rectángulo.*

**

*Ejemplo de implementación directamente en el objeto Image.*

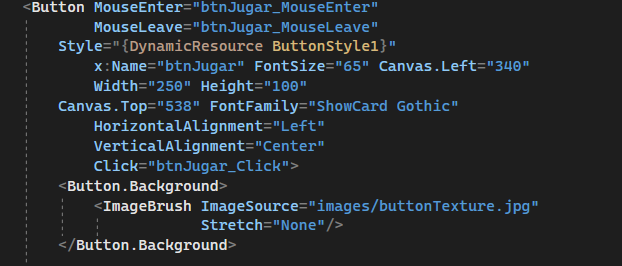
**LABEL**

* En estos objetos mostraremos información como puntuación, vidas, conversaciones entre los personajes, etc.

*Ejemplo de implementación en la conversación entre Winston y Soldado76 del Nivel2TalkFrame*

**BUTTON**

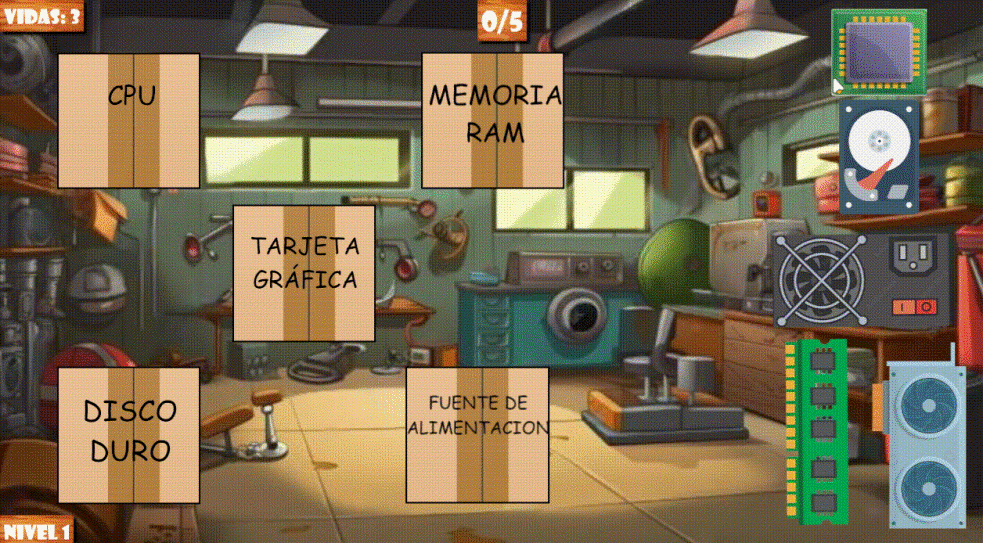
* Este objeto lo utilizaremos en casos como en el menú principal para empezar a jugar o salir de la aplicación.
* Son objetos destinados a ser pulsados y generar eventos a través de ellos.

*****Ejemplo de implementación de un Button el cual obtiene un estilo personalizado definido en el mismo documento XAML para tener un aspecto redondeado.*

# 3.2- Idea del juego

**IDEA:**

* La idea del juego planteada es una microhistoria en la que entramos en una empresa de venta, empaquetado, transporte entre otras cosas de componentes hardware.
* La mecánica principal es desplazar los componentes hardware a sus correspondientes cajas

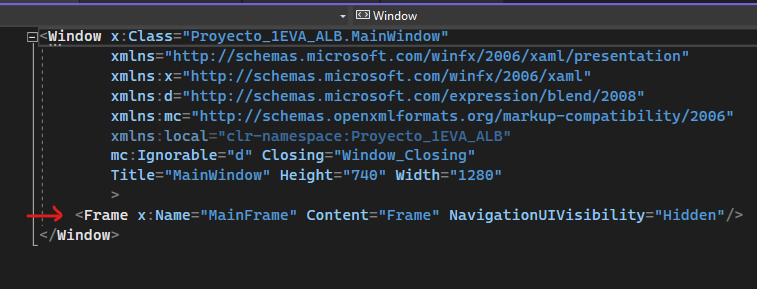


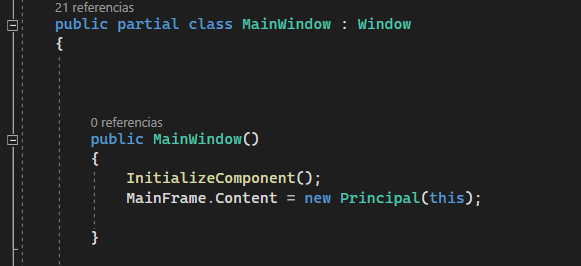
# 3.3 Lógica del juego

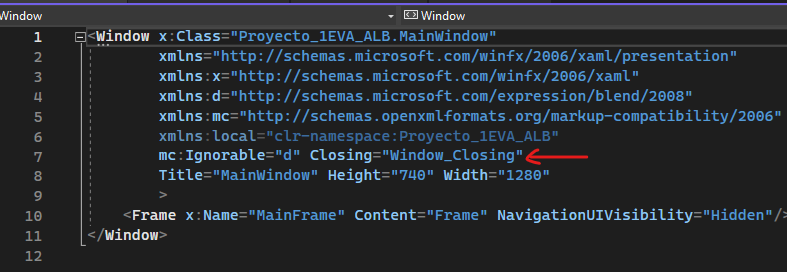
* El juego es una aventura interactiva lineal con la que tendremos que hablar con los clientes aunque la mecánica de jugabilidad principal será el Drag & Drop introduciendo componentes en sus cajas correspondientes antes de que su opacidad se reduzca a 0.
* La aplicación se ejecuta todo sobre la misma ventana “Mainwindow” y se ira avanzando en los niveles y en las interacciones con los personajes del juego mediante páginas.

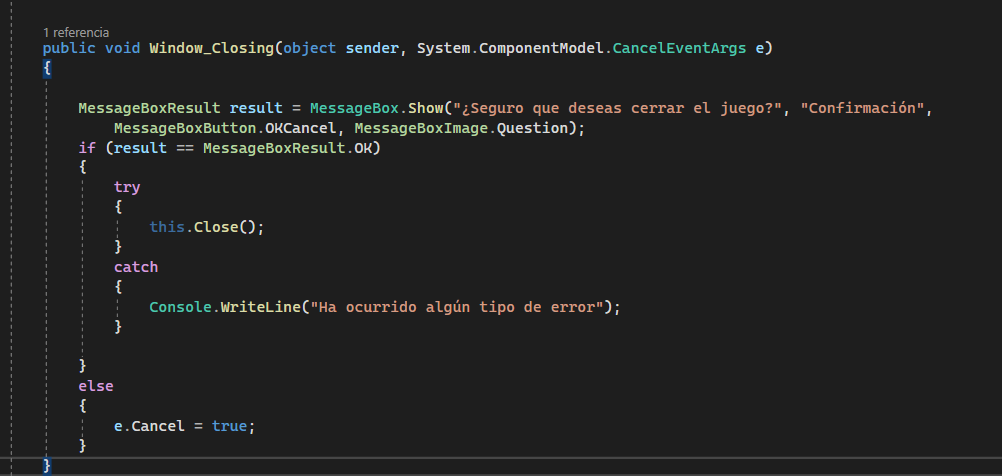
**MAINWINDOW**

* En el documento XAML de Mainwindow definimos el Frame para poder movernos entre los diferentes niveles o estados del juego mediante páginas. Esto mejora la experiencia del usuario ya que la transición es más fluida y orgánica.

*Ejemplo de la declaración del control Frame en el documento XAML de Mainwindow*

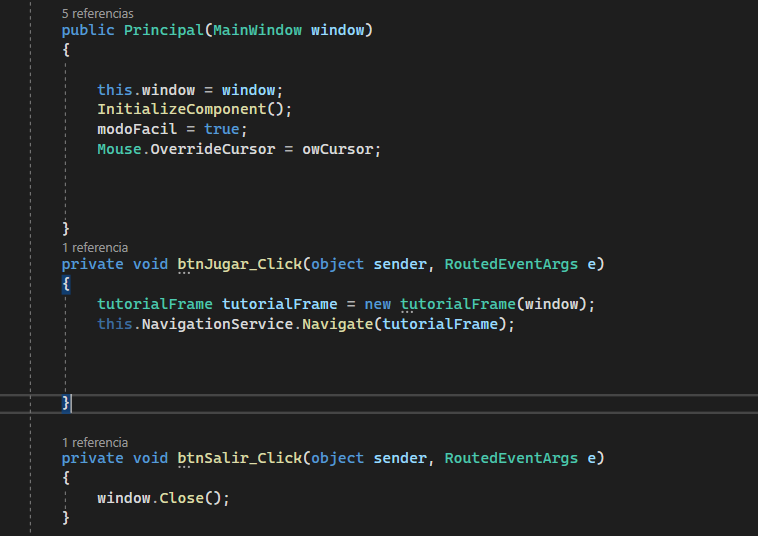
*Asignamos como página principal la clase “Principal” pasándole la configuración de la ventana como parámetro*

*Declarando el atributo Closing generamos un método para personalizar el evento de cerrado de la ventana*

*Modificamos el evento para realizar un dialogo de confirmación antes de cerrarla.*

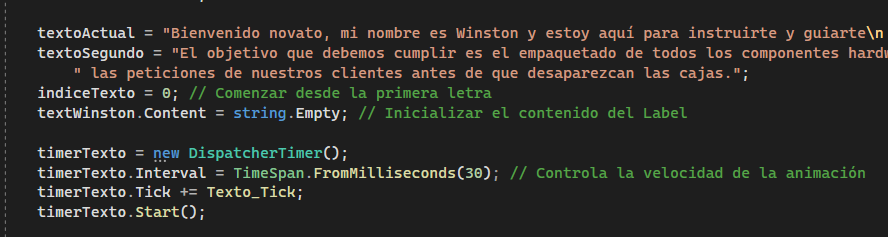
**PRINCIPAL**

* En esta página está destinada para darle un punto de entrada al usuario dándole la opción de empezar a jugar o salir de la aplicación.

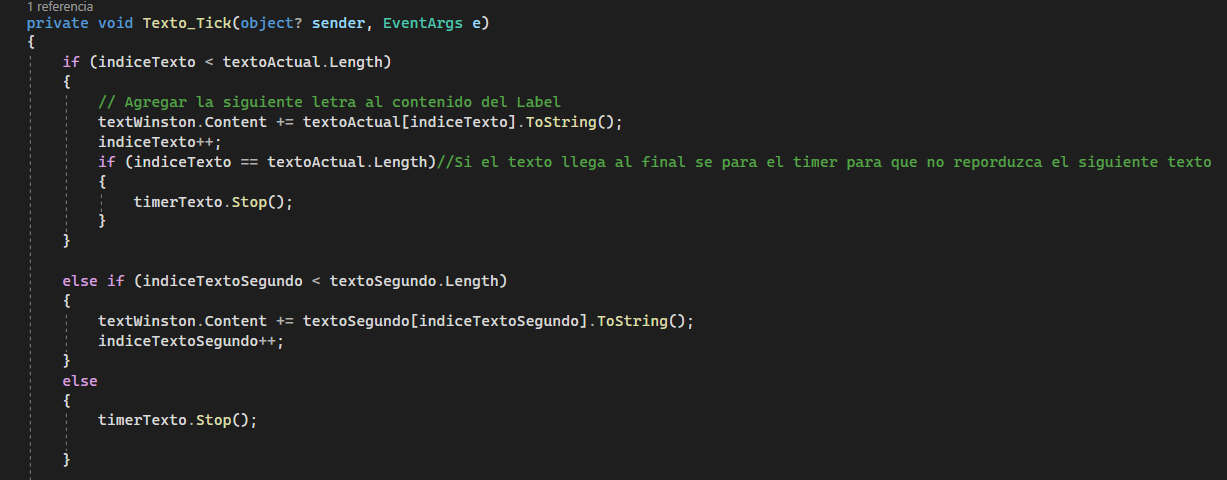
*Ejemplo en el que podemos ver los eventos de jugar y salir llamando al método Close() de la ventana pasada como parámetro.*

**TUTORIALFRAME, nivel1TalkFrame, Nivel2TalkFrame, Nivel3TalkFrame, FinalTalkFrame**

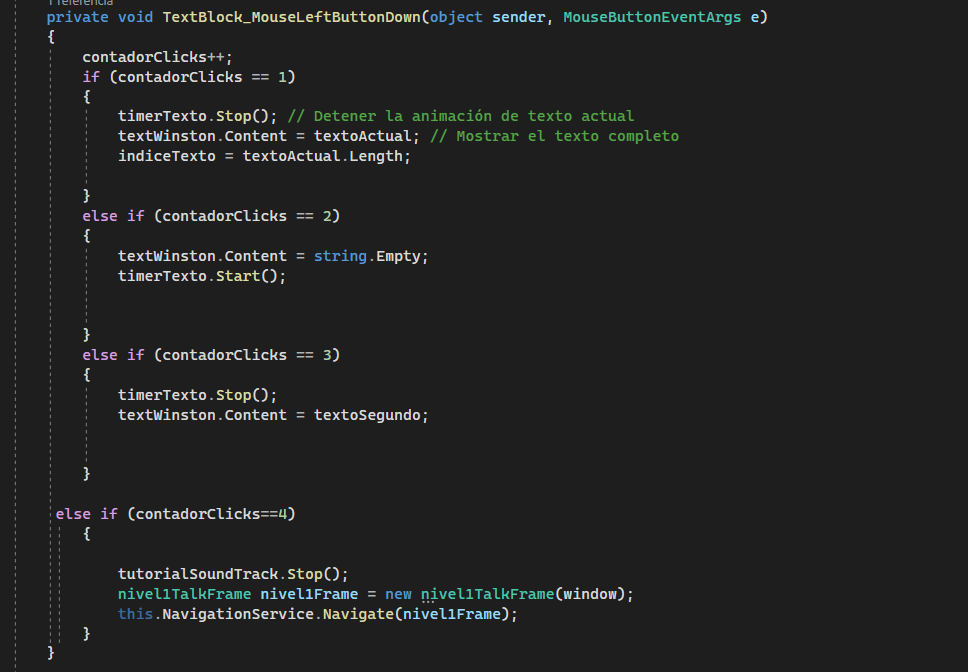
* En esta página básicamente el usuario será introducido en el juego presentándole al personaje Winston y se le mostrará un gif de la mecánica principal del juego (introducir los componentes en cajas antes de que desaparezcan).
* El texto se irá actualizando carácter a carácter cada 30 milisegundos generando así una animación que da más inmersión y realismo a las conversaciones con los personajes.
* Este código lo reciclaremos en futuras páginas en las que conoceremos a más personajes clientes antes de iniciar los niveles.
* El usuario deberá pulsar con el clic izquierdo sobre el texto de conversación bien sea para mostrar el texto de forma instantánea y/o pasar de dialogo.

****

* **textoActual,textoSegundo**: inicializamos los strings que queremos mostrar en el bocadillo de chat.
* **indiceTexto:** Inicializamos una variable en 0 para poder empezar desde el carácter 0 e ir iterando el índice para ir mostrando carácter por carácter la variable String.
* **textWinston.Content**: vacíamos el objeto label donde mostramos el texto.
* **timerTexto = new DIspatcherTImer();** inicializamos el DispatcherTime que nos ayudará a ejecutar métodos y funcionalidades de forma asíncrona.
* **timerTexto.Tick+= Texto\_Tick:** sobreescribimos el método Tick con un método en el que haremos que se muestre y se itere el índice del texto por cada tiempo de intervalo establecido.
* **timerTexto.Start():** inicializamos el DispatcherTime()

****

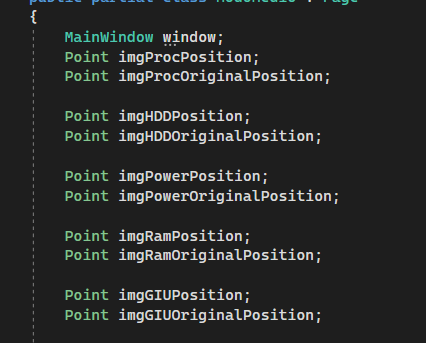
* Cada 30 milisegundos se ejecutará este método e irá escribiendo carácter por carácter desde el índice definido hasta la longitud del string iterando el índice por cada vez que entre en el método (30 milis).
* Cuando llegue el índice a la longitud del string se para el DispatcherTime() timer.
* El usuario lo reanudará mediante clics para que pase al segundo texto hasta que no haya más condiciones que tratar y se pare del todo.

****

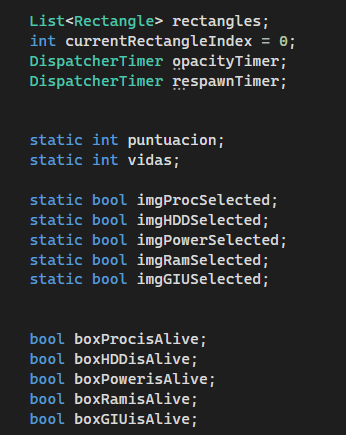
* Creamos un contador para controlar los clics del usuario, y con el método definido en el objeto cada vez que lo pulsemos sumaremos dicho contador y realizaremos operaciones como mostrar el texto instantáneamente, reanudar el timer de mostrar el texto carácter a carácter y por ultimo pasar a la siguiente página.

**MODOFACIL,MODOMEDIO,MODODIFICIL**

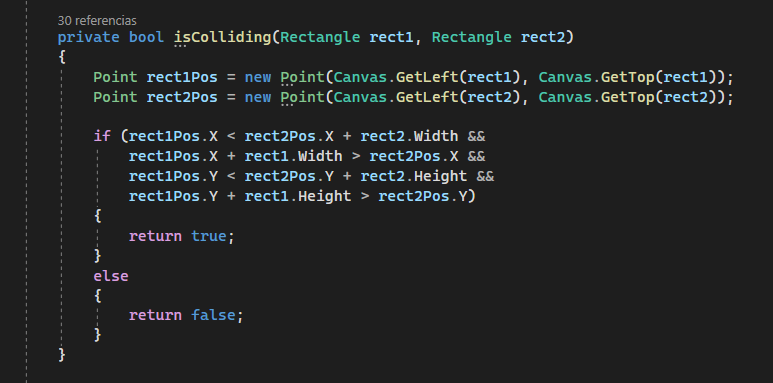
* En estas páginas vamos a establecer las mecánicas principales del juego

****

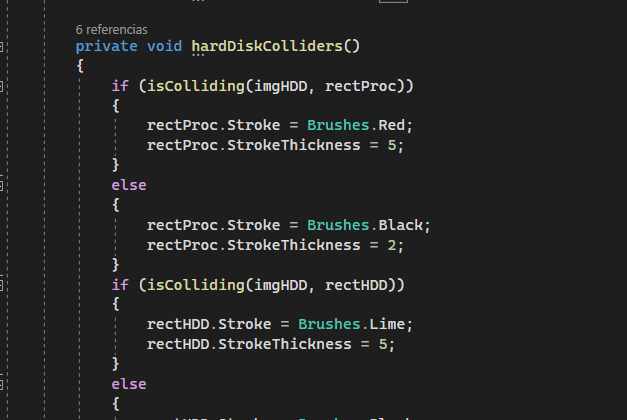
* Creamos variables de tipo Point para asignar la posición original de los componentes y la posición que modificaremos del objeto.

****

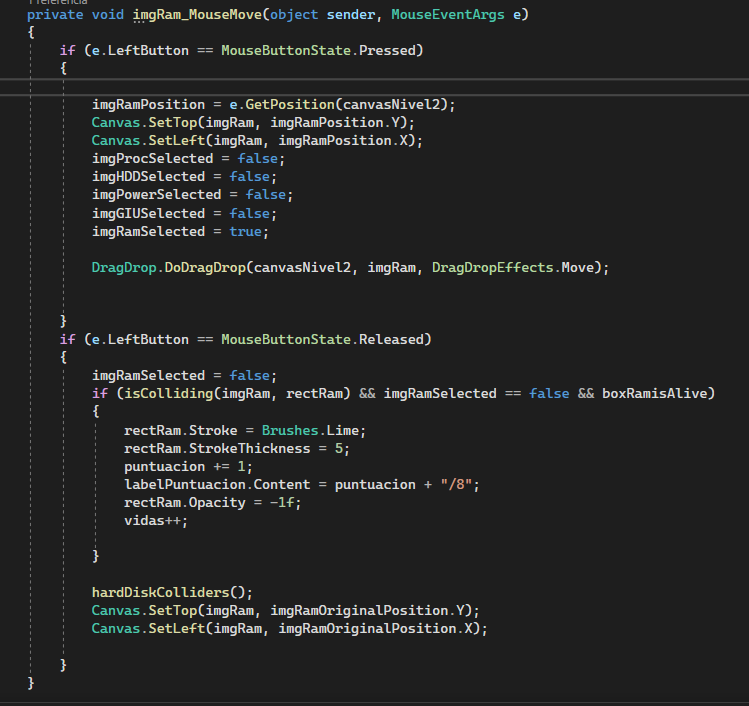
* Inicializamos lista para añadir los rectángulos que encapsularan las cajas, los DispatcherTimer, y los booleanos que controlaran cuando una caja esta activa para puntuar y cuando un componente esta seleccionado.

****

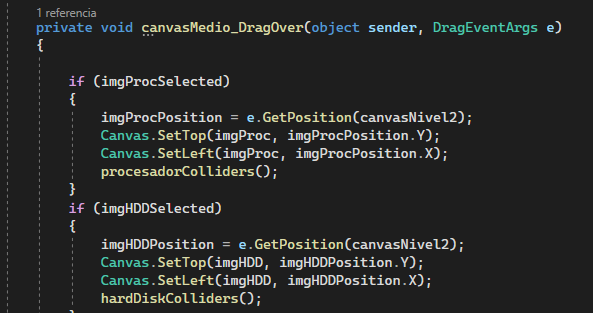
* Método en el que definiremos el sistema de colisiones pasándole 2 objetos de tipo rect.

****

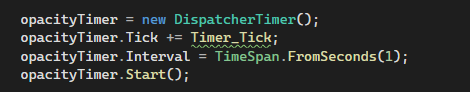
* En estos métodos controlamos las colisiones de todos los componentes con las cajas para crear la animación de colorear los bordes de las cajas no correspondientes con el componente.

****

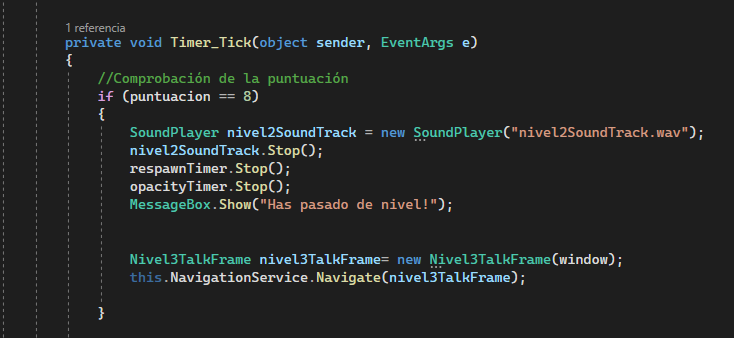
* En este método (1 por componente) establecemos la condición de si el botón medio está presionado sobre un componente automáticamente se deseleccionan todos los demás componentes.
* Después aplicamos el efecto de DragDropEffects.Move.
* Cuando el botón izquierdo se libera pasa a ser el booleano de selección a falso y si está colisionando con el objeto correcto se suma la puntuación se actualiza el Label de puntuación y se resta la opacidad del objeto.
* Fuera de la condición de colisión se reestablece el objeto a su posición original.

****

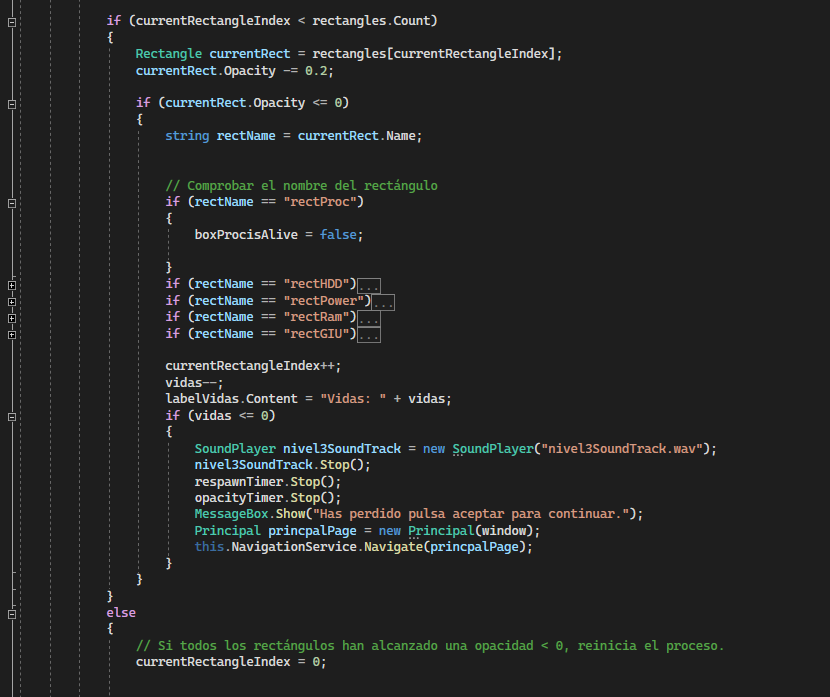
* En este método controlaremos la trayectoria del objeto seleccionado y mostraremos la animación de colisión del componente respecto a las cajas.
* Misma metodología para el método Drop.

****

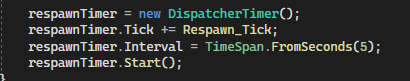
* Este DispatcherTimer controlará la opacidad de las cajas y otras variables del juego.

****

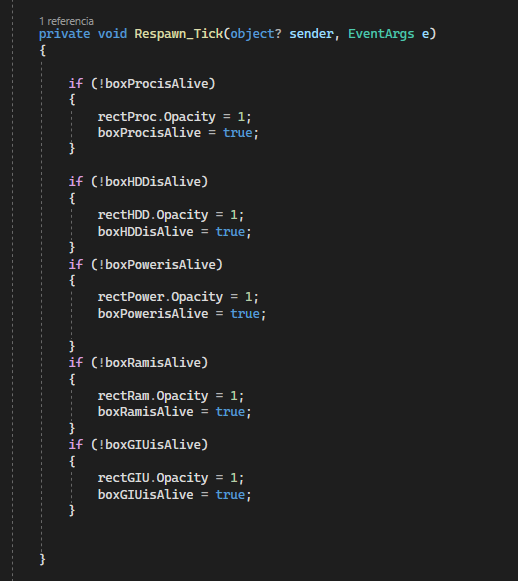
* Evaluamos la puntuación para saber si hemos superado el nivel y finalizamos los timers, la música de fondo y pasamos a la siguiente página.

****

* Evaluamos si el índice de los rectángulos es igual a la cantidad de los rectángulos de la lista (longitud) y si se cumple restamos 0.2 la opacidad de la caja hasta llegar a 0.
* Si la caja llega a 0 se obtiene el nombre para evaluarlo con los nombres de las cajas y cambiarlas el valor booleano de “”isAlive para controlar que cajas están completamente desaparecidas.
* Se le suma el índice si se cumple para pasar al siguiente rectángulo, se resta vidas y se actualiza el label para mostrar las vidas
* Si las vidas llegan a 0 se finalizan la música de fondo, los timers y se muestra un dialogo con un mensaje de confirmación.
* Si pulsa sobre el mensaje de confirmación nos devolverá a la página principal y si el índice llega a la longitud de la lista de rectángulos se reinicia a 0 el contador para establecer un bucle infinito de reducción de opacidad de las cajas.

****

* Este DispatcherTimer está destinado a hacer reaparecer las cajas para poder completar los niveles en los que necesitaremos más puntuación de 5.

****

* En el método del respawnTimer evaluaremos las variables booleanas de las cajas de si están vivas o están completamente desaparecidas para hacerlas reaparecer sobre el escenario.

# 4. - Conclusiones

¿Con qué dificultades nos hemos encontrado desde el proyecto inicial?

* Poco conocimiento del desarrollo de juegos y aplicaciones en WPF y C#.
* Añadir recursos en general.

¿Hemos tenido aplicar algún cambio de la idea inicial? ¿Por qué?

* Sí, ya que el inicio del proyecto fue ideado en desarrollar en diferentes ventanas y no páginas y tuve que reestructurar todo el proyecto y el código para adaptarlo.

¿Hemos cumplido los objetivos especificados inicialmente?

* Yo diría que los he superado, ya que el juego ahora consta de muchos más elementos y profundidad.

¿Hay alguna propuesta de mejora que se podría plantear para el futuro?

* Plantear una idea sólida de estructuración desde el principio para no tener que cambiar y estructurar la aplicación mediante páginas desde el inicio.
* Quizás realizar código más modular para poder depurar e identificar posibles errores.