Universidad de Valladolid
Escuela de Ingeniería Informática

TRABAJO FIN DE GRADO

Grado en Ingeniería Informática
Mención ingeniería del Software

CREACIÓN DE JUEGOS PARA EL APOYO DE LA ASIGNATURA "INFORMÁTICA" DEL GRADO DE INGENIERÍA DE DISEÑO INDUSTRIAL Y DESARROLLO DE PRODUCTO.

Alumno:
Oviedo García, Rubén

Tutores:
Pisabarro Marrón, Alma María
Vivaracho Pascual, Carlos Enrique

Resumen

Este Trabajo de fin de Grado forma parte de un proyecto de gamificación que tiene como objetivo facilitar el aprendizaje de conceptos básicos de programación mediante el uso de juegos serios, juegos cuyo objetivo no es exclusivamente lúdico. Este proyecto ya se ha implantado en el Grado de Ingeniería Informática con siete videojuegos que ilustran conceptos de programación en Java, con resultados muy satisfactorios, por lo que se pretende extender esta iniciativa a los grados de Matemáticas e Ingeniería de Diseño Industrial y Desarrollo de Producto.

Dentro de este marco, este trabajo de fin de grado tiene como tarea adaptar tres de los juegos ya existentes al lenguaje C, que es el que se utiliza en las asignaturas de programación de los grados de Matemáticas e Ingeniería de Diseño Industrial y Desarrollo de Producto.

De los tres juegos revisados uno no necesitó cambios, otro necesitó modificaciones y el tercero, cuyo objetivo es ilustrar algunos conceptos básicos sobre cadenas de caracteres, se ha rediseñado e implementado de nuevo.

La motivación tras este proyecto es la de facilitar a los alumnos, que se enfrentan por primera vez a la programación, la comprensión de conceptos abstractos de la programación de una forma divertida para conseguir su participación mas activa en la asignatura.

Índice

Índice de figuras

Índice de tablas

1. Introducción

En los últimos cursos académicos, los alumnos de la asignatura de Fundamentos de Programación del *Grado de Ingeniería informática* han formado parte de un proyecto para facilitar la comprensión y el aprendizaje de determinados conceptos de programación por medio de juegos serios. Debido a los resultados favorables con los alumnos de este grado, se pretende extender este proyecto a los grados de *Ingeniería de Diseño Industrial y Desarrollo de Producto* y el *Grado de Matemáticas*.

Las asignaturas de *Informática* de los grados de Ingeniería de Diseño Industrial y Desarrollo de Producto y de Matemáticas a los que se pretende trasladar esta iniciativa imparten clases sobre el lenguaje de programación C mientras que en el grado de Ingeniería Informática se utiliza Java, por lo que es necesario realizar ajustes en los juegos ya existentes a la vez que se crean nuevos juegos, ya que existen diferencias entre ambos lenguajes.

La tarea que se aborda en este Trabajo de Fin de Grado es desarrollar un juego que instruya sobre el tipo de dato Cadenas en el lenguaje C y algunas de sus funciones más básicas. Además, se realizarán modificaciones a alguno de los juegos ya desarrollados para que, en vez de ilustrar conceptos de Java, lo hagan sobre C.

El juego modificado ha sido el denominado *Caída de datos*. Este juego tiene por objetivo hacer que el jugador sea capaz de diferenciar los distintos tipos de datos. El juego creado de cero es *Musicadenas*, que pretende mostrar la diferencia entre las funciones básicas sobre las cadenas de caracteres del lenguaje de programación C (Add, delete y copy) y también diferenciar entre variables de entrada y salida. A diferencia de *Caída de datos*, que esta creado con el programa de creación de videojuegos Unity, este juego se ha creado en otro programa de creación de videojuegos distinto, Godot, por lo que mas adelante se explicarán sus diferencias.

1.1. Motivación

A menudo el estudio por medio de las técnicas habituales puede ser una tarea tediosa o complicada, sobre todo cuando se comienza a aprender una materia nueva, ya que hay una gran cantidad de conceptos que asimilar y puede que algunos sean difíciles de comprender. Existen varios trabajos [1] que corroboran la eficiencia de los juegos como método de aprendizaje, por lo que incluirlos dentro de un ámbito académico podría ayudar a paliar los problemas mencionados anteriormente

Los juegos creados para este proyecto están diseñados de forma que sus mecánicas supongan un reto divertido para el jugador y que los conocimientos aprendidos jugando puedan ser trasladados al ámbito de la programación. También, a titulo personal se ha intentado crear este juego utilizando únicamente herramientas Open Source

1.2. Contexto

Durante las últimas 2 décadas la industria del videojuego ha crecido hasta el punto de generar tantos ingresos o más que el resto de industrias del entretenimiento como la música o el cine. Este

éxito se debe en parte al fácil acceso que presenta para cualquier persona que quiera crear un videojuego. Se pueden crear desde pequeños juegos sin ningún tipo de presupuesto hasta producciones triple A invirtiendo millones de dólares en el proceso. Esto conlleva la posibilidad de que exista un videojuego para cada nicho de personas existente.

En este caso, el nicho en el que este proyecto se va a centrar es en el académico, ya que debido a los resultados positivos de los estudios realizados sobre la gamificación, estos métodos se están empezando a aplicar a los videojuegos.

La **gamificación** consiste en el uso de mecánicas, elementos y técnicas de diseño de juegos en contextos que no son juegos, para involucrar a los usuarios y resolver problemas [1]. La gamificación no es exclusiva en el ámbito académico, se puede utilizar en marketing para captar o mantener clientes utilizando su producto o en el ámbito empresarial para fomentar la comunicación y el compañerismo entre los trabajadores. Un ejemplo de gamificación fuera del ámbito académico son los ”Me Gusta”de las redes sociales. Los ”Me Gustaçrean una clasificación y disparan los mismos efectos psicológicos que la puntuación o la consecución de recursos en los juegos.

Los resultados de los distintos estudios y pruebas sobre la gamificación dicen que aportan las siguientes ventajas:

- **Motivar al alumno:** Crear un reto lo suficientemente sencillo de superar motivan al alumno a continuar jugando y eso ayuda a que tenga una mayor predisposición a aprender conceptos nuevos. Lo divertido que sea dicho juego y el elemento social también ayudan a esto
- **Utiliza los conocimientos aprendidos:** Que el alumno sea capaz de ver como sus conocimientos aprendidos sobre la materia le son útiles de manera directa ayuda a asentar dichos conocimientos
- **Recibir retroalimentación instantáneamente:** El alumno es capaz de ver sus fallos o sus aciertos al momento, lo cual ayuda a detectar que conocimientos son erróneos o no se han comprendido bien.
- **Favorece la socialización:** Aunque el juego haya sido desarrollado con un carácter totalmente competitivo, las personas acabarán socializando para intentar aprender como conseguir mejores puntuaciones [2].

1.3. Objetivos

Con este proyecto se pretende modificar un videojuego ya existente dentro del proyecto de gamificación del *Grado de Ingeniería Informática* y la creación de otro para potenciar el aprendizaje de la relación de tipos de datos simples y el funcionamiento de las cadenas de caracteres en el lenguaje de programación C. Este objetivo general se divide en los siguientes objetivos especificos

- Modificar un juego relacionado con tipos de datos en Java, para adaptarlo al lenguaje de programación C

- Crear un nuevo juego que seguirá la definición de minijuego. Esto significa que el juego requerirá poco tiempo para ser completado y deberá poseer una interacción sencilla basada en acciones simples de ratón
- El juego además de instructivo, deberá ser divertido y suponer un reto.
- Los juegos deberán integrarse en la plataforma GamiSpace, plataforma que usa el proyecto para la gestión tanto de juegos como de jugadores.

2. Plan de Proyecto

2.1. Metodología utilizada

Para el desarrollo de este proyecto se ha escogido una metodología agile basada en sprints orientada a la elaboración de TGFs llamada ASAP (Agile Student Academic Projects) o UVAgile [3].

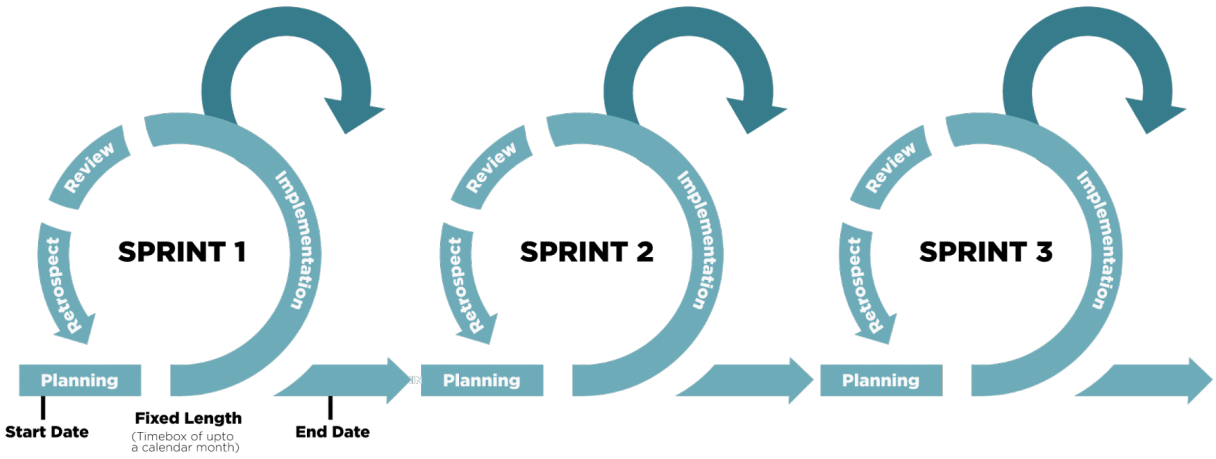


Figura 1: Ilustración de la metodología AGILE por sprints

Las metodologías ágiles se basan en la continua comunicación entre los distintos roles, creando sprints que engloban tareas de corta duración para ir construyendo el producto a base de pequeños incrementos.

La metodología UVAgile divide a los participantes en 4 roles distintos.

- **Estudiante:** Es el rol principal, ya que es el rol que va a realizar la mayoría de las tareas.
- **Tutor:** La o las personas que tutelan el TFG. Determina la forma en la que se realiza el seguimiento del proyecto y retroalimenta al estudiante en cada reunión de seguimiento.

- **Comunidad:** Personas externas que puedan apoyar en la realización del TFG (Profesores, alumnos, expertos, etc).
- **Tribunal:** La comisión evaluadora del TFG. La actividad de este rol es la de evaluar tanto el acto de la defensa del trabajo como el producto que se le presenta.

Los sprints deben tener una duración de no más de 1 mes y debe describir su propio objetivo. Esta división en sprints se realiza para reducir la incertidumbre, teniendo objetivos que realizar a corto plazo, construyendo el producto en pequeños incrementos. Cada sprint se divide en 4 fases 1.

- **Planificación:** Durante esta fase se discute con los tutores los objetivos que se tienen que cumplir en el sprint.
- **Implementación:** En esta parte del sprint el estudiante trabaja para cumplir los objetivos propuestos en la fase anterior dentro del limite de tiempo establecido. A lo largo de esta fase se realizan reuniones periódicas para comprobar el progreso del sprint y dar retroalimentación en caso necesario.
- **Revisión:** Se realiza al final del sprint, donde el estudiante presenta el trabajo realizado hasta el momento.
- **Retrospectiva:** Se realiza tras la comunicación de progresos y su finalidad es la de reflexionar sobre la calidad del progreso seguido.

2.2. Plan de riesgos

En esta sección se recogerán los riesgos que deben tenerse en cuenta a la hora de realizar el desarrollo del juego, clasificándolos por probabilidad e impacto y de ser posible presentar tanto medidas preventivas como planes de contingencia.

R-01	Enfermedad del desarrollador
Descripción	El desarrollador no es capaz de trabajar en el proyecto debido a una enfermedad incapacitante
Probabilidad	Media
Impacto	Medio
Medidas preventivas	Evitar poner en riesgo la salud
Plan de contingencia	Realizar la planificación del proyecto con márgenes de tiempo suficientes para poderse realizar dentro de ese tiempo aún en caso de imprevistos

Tabla 1: R-01: Enfermedad

R-02	Insuficiencia de tiempo del desarrollador
Descripción	El desarrollador no es capaz de trabajar en el proyecto debido a la incapacidad de conciliar el proyecto con la vida laboral
Probabilidad	Media
Impacto	Medio
Medidas preventivas	Ninguna
Plan de contingencia	Realizar la planificación del proyecto con márgenes de tiempo suficientes para poderse realizar dentro de ese tiempo aún en caso de imprevistos

Tabla 2: R-02: Falta de tiempo

R-03	Pérdida de información
Descripción	El desarrollador pierde parte o la totalidad del progreso del proyecto realizado debido a cualquier tipo de error humano o no
Probabilidad	Baja
Impacto	Alto
Medidas preventivas	Llevar un control de versiones al día tanto de forma local como remota o guardar copias de seguridad en un disco duro externo
Plan de contingencia	Realizar la planificación del proyecto con márgenes de tiempo suficientes para poderse realizar dentro de ese tiempo aún en caso de imprevistos

Tabla 3: R-03: Perdida de información

R-04	Fallo en el equipo de trabajo
Descripción	El desarrollador es incapaz de trabajar en el proyecto debido a una avería en su máquina de trabajo
Probabilidad	Baja
Impacto	Medio
Medidas preventivas	Llevar un control de versiones al día tanto de forma local como remota o guardar copias de seguridad en un disco duro externo
Plan de contingencia	Trabajar desde otro dispositivo

Tabla 4: R-04: Fallo en el equipo de trabajo

R-05	Falta de conocimiento sobre las tecnologías
Descripción	No se tiene suficiente experiencia en el uso de las herramientas
Probabilidad	Alta
Impacto	Bajo
Medidas preventivas	Formarse sobre las tecnologías que se van a utilizar antes de comenzar la realización del proyecto
Plan de contingencia	Consultar documentación sobre el tema, y en caso necesario retrasar la entrega

Tabla 5: R-05: Falta de Conocimiento sobre las tecnologías

R-06	Cambios en los requisitos del proyecto
Descripción	Se realizan cambios a los requisitos planteados al principio del proyecto
Probabilidad	Alta
Impacto	Bajo
Medidas preventivas	Realizar la planificación utilizando la metodología AGILE
Plan de contingencia	Replantear la planificación

Tabla 6: R-06: Cambios en los requisitos del proyecto

R-07	Retraso en el cumplimiento de la planificación
Descripción	Debido a retrasos en el desarrollo no se puede cumplir con la planificación.
Probabilidad	Alta
Impacto	Medio
Medidas preventivas	Realizar la planificación del proyecto con márgenes de tiempo suficientes para poderse realizar dentro de ese tiempo aún en caso de imprevistos
Plan de contingencia	Retrasar la fecha de entrega del proyecto

Tabla 7: R-07: Retraso del cumplimiento de la planificación

R-08	Falta de disponibilidad del cliente
Descripción	Es imposible comunicarse con el cliente debido a la falta de disponibilidad del mismo
Probabilidad	Media
Impacto	Bajo
Medidas preventivas	Utilizar metodologías AGILE para tener las reuniones semanales en una fecha ya establecida
Plan de contingencia	Realizar las tareas que no requieran retroalimentación inmediata del cliente

Tabla 8: R-08: Falta de disponibilidad del cliente

2.3. Presupuesto

El presupuesto real para este trabajo ha sido de 0 €, ya que como el único desarrollador ha sido el alumno y los materiales usados ya eran propiedad del desarrollador o eran totalmente gratuitos, no se han generado costes adicionales. Sin embargo, se simulará un análisis de costes del proyecto.

En primer lugar, se hará una estimación de las horas que se ha tardado en realizar cada etapa del proyecto.

Tarea	**Duración**
Modificación de los juegos anteriores	23 horas
Diseño interfaz	10 horas
Diseño sprites y fondos	35 horas
Desarrollo Scripts	72 horas
Desarrollo Interfaz	70 horas
Implementación con la página Web	21 horas
Documentación	72 horas
Total	**300 horas**

Tabla 9: Duración estimada del proyecto

Sabiendo que actualmente la media de salario de un programador en España está entre los 22000 y los 33000 [4] euros haremos el cálculo usando una cifra media de 28000 euros anuales o lo que es lo mismo, unos 13,48 € la hora. Esto supondría un precio total de 4044 €.

El hardware que se ha utilizado para la realización de este proyecto ha consistido en un ordenador de sobremesa con un coste aproximado de 1700 € con 2 pantallas valoradas ambas en 200 €. Para calcular la amortización de estos dispositivos se utilizará la **Tabla de coeficientes de amortización lineal de la Agencia Tributaria** [5] la cual informa que para Equipos electrónicos el coeficiente lineal es de un 20 % anual por lo que el coste del equipo sería de 54,02 € para el ordenador y 6.35 € por el uso de las pantallas.

Como se ha comentado anteriormente, el software ha sido totalmente gratuito, por lo que no supone un gasto de ningún tipo, el presupuesto para este proyecto sería de

Coste	**Precio (€)**
Sueldo desarrollador	4044
Hardware	55.37
Software	0
Total	**4099.37**

Tabla 10: Presupuesto estimado

2.4. Seguimiento del proyecto

En esta sección se mostrará de manera detallada los objetivos acordados para cada sprint y los posibles problemas surgidos durante la realización de los mismos. Todo el trabajo realizado se puede dividir en 5 grupos.

Los primeros 3 sprints del proyecto se centraron en:

- La formación del estudiante ya que este desconocía el funcionamiento de las herramientas que iba a utilizar y si era posible implementar el juego en la plataforma web en la que debia alojarse.
- La creación de la idea inicial del juego a desarrollar.
- Realización de la documentación preliminar necesaria para poder desarrollar el videojuego Musicadenas.

Una vez desarrollada la idea del videojuego que se iba a crear creando la documentación preliminar necesaria y comprobado que las herramientas que se pretendían utilizar podían ser implementadas en la plataforma web se procedió a revisar los juegos ya creados. Por tanto los siguientes sprints que se realizaron fueron los siguientes.

- Revisión de los videojuegos del proyecto asignados.
- Realización de los cambios necesarios.
- Exportar los juegos a la plataforma.

En el primer sprint se revisaron ambos juegos asignados y se vio que únicamente uno de los dos juegos necesitaba que se realizasen cambios. En el siguiente sprint se realizaron los cambios necesarios pero ya que eran mayormente cambios sencillos este sprint no duró mucho tiempo. Por ultimo se exportaron los juegos a la plataforma pero debido a diversos fallos tanto en la plataforma como en la configuración del proyecto a la hora de exportarlo este sprint no pudo ser completado en el tiempo previsto.

Una vez realizados y subidos los cambios de los juegos a la nueva plataforma se empezó a diseñar en mas detalle el nuevo videojuego. Los siguientes sprints fueron.

- Diseño de interfaz.
- Diseño de la arquitectura del proyecto.
- Diseño del modelo de dominio.
- Diseño de niveles

De estos sprints habría que destacar que hubo un ligero retraso en el diseño de la interfaz debido a que hubo que iterar varias veces sobre ello para que fuese perfectamente legible y entendible para el mayor número de gente posible, por lo que se tuvieron que realizar cambios varias veces.

A continuación se comenzó con el desarrollo del juego nuevo. Esta parte del proyecto fue la mas larga y en la que hubo más retrasos debido a la falta de conocimiento sobre las tecnologías utilizadas, sobre todo a la hora de conectar el juego con la plataforma. Los sprints de esta parte del proyecto fueron los siguientes.

- Diseño de sprites y fondos.
- Diseño de efectos de sonido y musica de fondo..
- Implementación de la interfaz juego.
- Implementación de la interfaz del menú principal.
- Implementación de los scripts que controlan el funcionamiento de los cantantes y las teclas del piano.
- Implementación de los scripts que controlan el funcionamiento de las batutas.
- Implementación de los scripts que controlan el funcionamiento de la grabadora.
- Implementación de los scripts de generación de cadenas, conteo de puntos y temporizador.
- Implementación de la pantalla de fin de juego.
- Implementación de selección de nivel y pantalla de tutorial.
- Conexión con la plataforma.

Por ultimo, tras desarrollar el juego, los últimos sprints se enfocaron en la realización de las pruebas unitarias y pruebas beta, comprobando que todo funcionase correctamente y enseñando el juego a algunos usuarios para conseguir mayor retroalimentación y en caso necesario cambiar algunas partes del juego. En ambos sprints se detectaron errores que fueron arreglados rapidamente. Los sprints fueron los siguientes.

- Realización de pruebas unitarias.
- Realización de pruebas beta.

3. Tecnologías Utilizadas

En esta sección se detallarán todas las herramientas utilizadas para el diseño y la creación del minijuego **Musicadenas**

3.1. Godot

Godot Engine es un motor de videojuegos. Un motor de videojuegos es una programa que permite la creación de videojuegos ya que contiene las herramientas necesarias para la creación del mismo como puede ser un simulador de físicas para calcular la gravedad que se ejerce sobre un personaje o las funciones necesarias para que dicho personaje se mueva por la pantalla.

El código fuente de Godot Engine fue lanzado al público en 2014 por la empresa *OKAM Studios* y ha ido recibiendo actualizaciones con el tiempo hasta llegar a la versión 4.2.2 en 2024.

A diferencia de los otros grandes motores del mercado como *Unity* o *Unreal Engine* Godot trabaja bajo una MIT License [6], lo que significa que es totalmente Open Source y aparte de que cualquier persona puede contribuir a mejorar el motor, sus ingresos dependen mayoritariamente de la comunidad para mantenerlo. El motor es totalmente gratuito para cualquier tipo de propósito, mientras que tanto Unity como Unreal Engine tienen un sistema de royalty's para proyectos publicados fuera del ámbito académico.

Este motor permite el desarrollo de videojuegos para varias plataformas, como Windows, Linux, para web o Android dentro del mismo proyecto (utilizando el mismo código se puede exportar el juego para todas las plataformas). Esto ahorra tiempo al equipo de desarrollo a la hora de crear un juego para más de una plataforma. Además, Godot tiene un motor 2D dedicado, lo que simplifica y mejora el desarrollo de juegos 2D como el que vamos a desarrollar aquí. Otros motores de videojuegos, como Unity, siempre trabajan en 3D, de manera que al desarrollar un videojuego en 2D lo que hace es proyectar la escena sobre el plano XY. Esto produce efectos indeseados, como la ocultación de objetos al estar solapados en el eje Z, complicando el desarrollo del juego. También posee un sistema de casillas para crear mapas de una forma más sencilla.

Aunque Godot posee su propio lenguaje de scripting del que hablaremos más adelante, de forma nativa también acepta programar en .NET/C# y usando plugins C y C++ lo cual es una ayuda para personas que vienen de otros motores como unity, ya que no tienen por qué aprender un lenguaje nuevo.

Por último comentar otras ventajas notables del motor.

- **Documentación bien organizada:** posee información bien escrita sobre todas las clases y funcionamientos de todas las características del motor, así como tutoriales para aprender a utilizarlo.
- **Librería de assets:** El motor tiene integrada una librería de assets donde la comunidad puede subir scripts o herramientas para que la gente pueda utilizarla, facilitando así el desarrollo.
- **Community Driven:** Para enlazar con el punto anterior al ser un proyecto open source los avances que tenga el motor dependen mucho de la comunidad que lo utiliza, por esa razón la gente que lo utiliza está más predispuesta a compartir su conocimiento o las herramientas que hayan creado para ayudar a otras personas que utilicen el motor.

3.2. GDScript

GDScript es el lenguaje de scripting nativo de Godot Engine y se trata de un lenguaje orientado a objetos y dinámico con una sintaxis basada en indentaciones parecido a Python. El hecho de que sea un lenguaje creado específicamente para este motor hace que se pueda optimizar lo máximo posible y tenga una sintaxis lo más simple posible.

Que GDScript sea un lenguaje dinámico tiene varias ventajas.

- El código es fácil de leer
- El código de ejecución es pequeño
- No es necesario compilar para hacer pruebas

Pero también tiene desventajas:

- Tiene menor rendimiento que los lenguajes de tipado estático.
- Algunos errores que se detectarían en el tiempo de compilación en lenguajes de tipado estático, solo aparecen cuando se ejecuta el código
- Menos flexibilidad para el autocompletado de código, ya que algunas variables solo se conocen en tiempo de ejecución

Pero algunas de estas desventajas se pueden minimizar porque, como se comentaba anteriormente, Godot permite utilizar varios lenguajes de programación dentro del mismo proyecto, por lo que para aquellas funciones o scripts que requieran una mayor optimización de los recursos de la maquina se pueden utilizar otros lenguajes estáticos como C++.

3.3. Pixilart

Pixilart es una página web que funciona como herramienta de dibujo estilo 'Pixel Art' ya que permite dibujar 2 y animar imágenes de poca resolución 3 y luego exportarlas reescalando la imagen.Ademas esta pagina web posee un apartado de red social donde las personas pueden compartir los dibujos que han hecho. Esta página/red social ha sido creada por Bryan Ware.

Esta herramienta ha sido utilizada para la creación de todos los personajes, animaciones, fondos e imágenes que componen la interfaz del videojuego Musicadenas

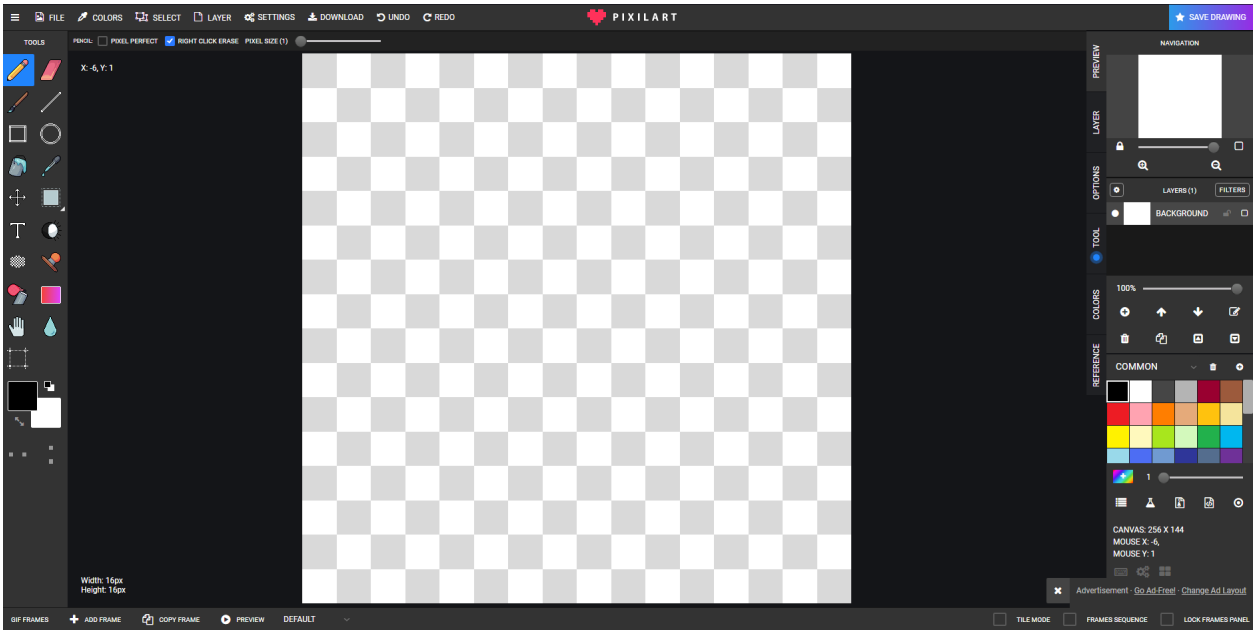


Figura 2: Menú de dibujo y animación de la pagina

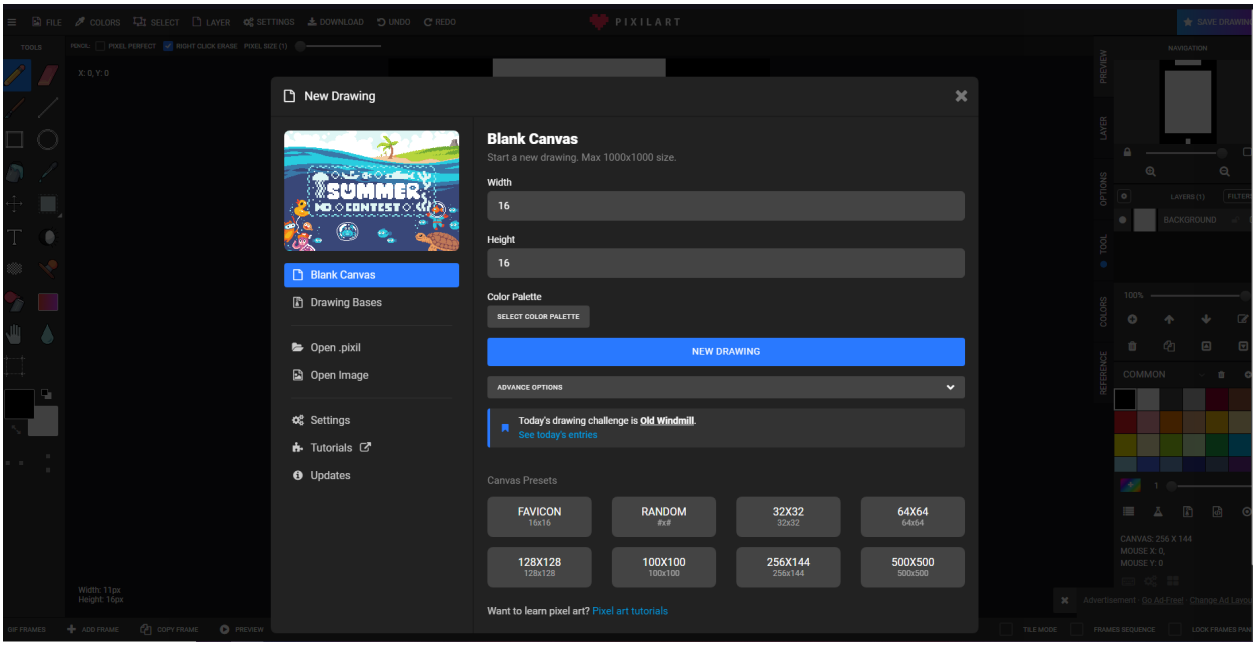


Figura 3: Menú selección tamaño Lienzo

3.4. BeepBox

BeepBox es un proyecto creado por John Nesky para poder bocetar música instrumental.Se ha elegido esta herramienta principalmente porque es capaz de crear musica estilo 'Chiptune' que es un estilo de musica que se intenta asemejar al de la musica generada por chips de sonido tipica de los videojuegos de la década de 1980. Es una herramienta muy intuitiva de utilizar si tienes unos

conceptos básicos de música ya que posee 4 pistas distintas de audio para, por ejemplo, tener en una pista notas mas graves simulando la percusión y en otra pista la melodía principal.

Esta herramienta se ha utilizado para crear la música del videojuego Musicadenas

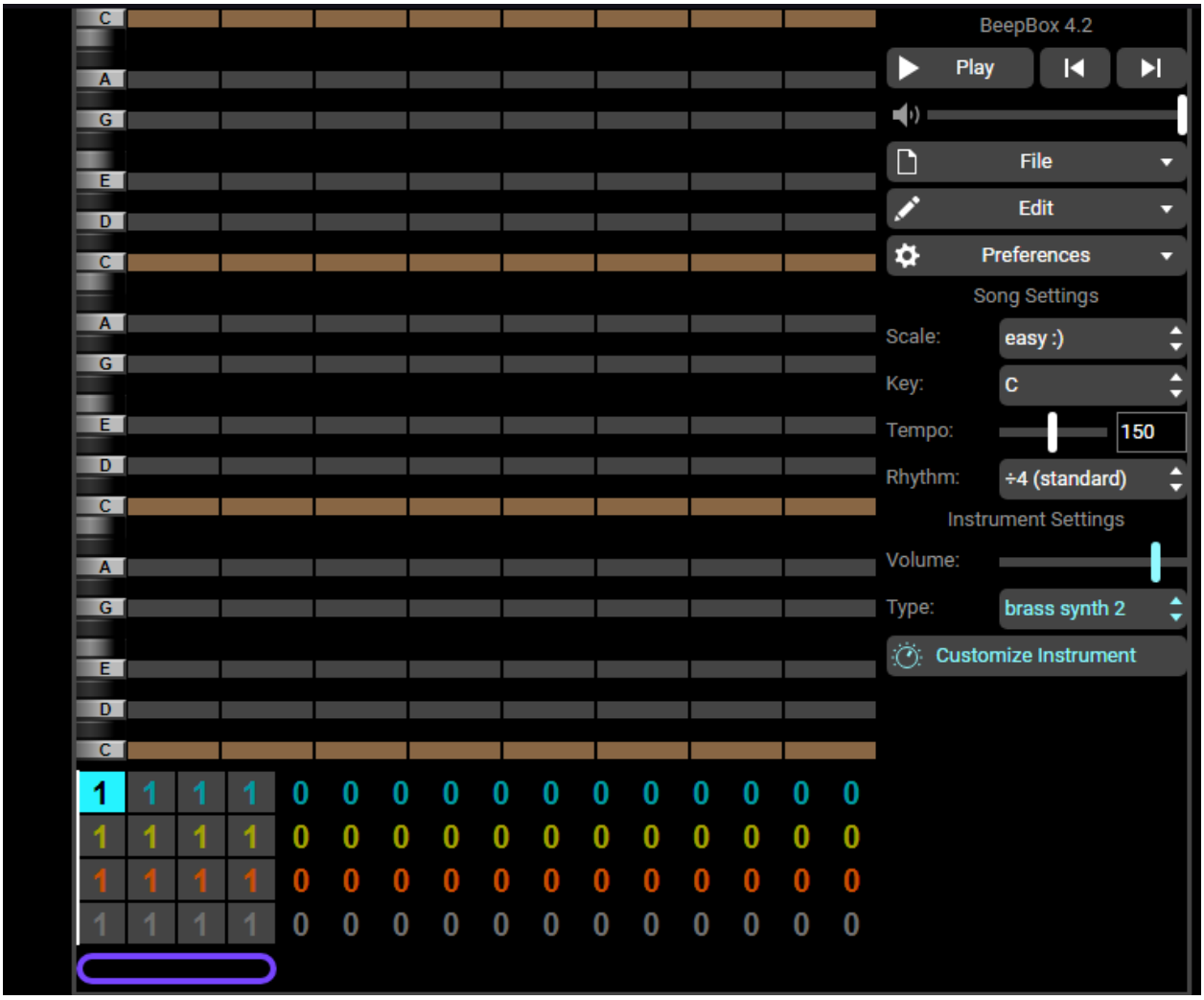


Figura 4: Interfaz de la herramienta BeepBox

4. Cambios realizados a los juegos de otros años

Ya que no se ha realizado un desarrollo completo de los juegos *Caida de Datos* y *Cafetería* y los cambios realizados han sido únicamente cambios en la interfaz, no realizaremos un documento de diseño completo. Únicamente nos centraremos en el análisis de requisitos y en detallar dichos cambios.

4.1. Análisis de Requisitos

Para la realización de esta sección del trabajo el cliente ha pedido que se realicen 2 tareas principales.

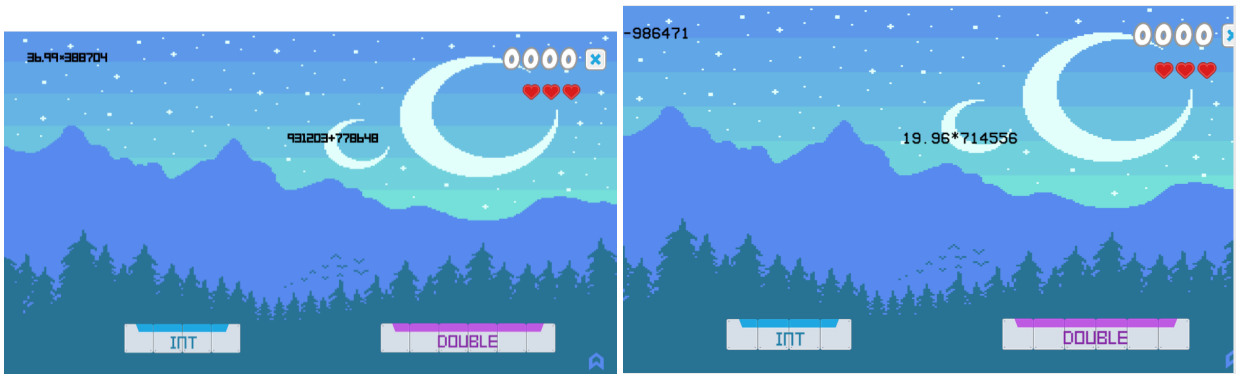
En primer lugar, se ha pedido que se revisen dos juegos ya creados para el proyecto de gamificación. La revisión consiste únicamente en comprobar si los conceptos que ilustra se pueden trasladar al lenguaje de programación C.

En segundo lugar, en caso de que alguno de estos juegos contenga algún elemento que no pertenezca al lenguaje de programación C se realizarán las modificaciones necesarias para eliminar o cambiar estos elementos.

4.2. Modificación de videojuegos: Caída de datos

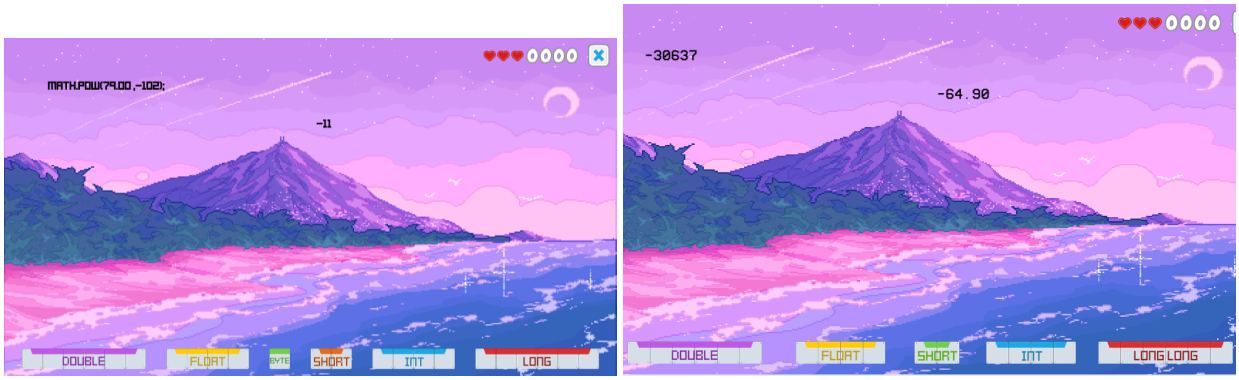
Caida de datos es un videojuego que intenta ayudar a los alumnos a diferenciar los distintos tipos de datos simples. El mecanismo del juego consiste en recoger, a través del movimiento en dos dimensiones, los tipos de datos en el recipiente adecuado. Los recipientes se encuentran en la parte baja de la pantalla, mientras que las expresiones van cayendo desde la parte superior de la pantalla. Se considera como recipiente adecuado el tipo de dato más pequeño de todos los disponibles en el que se pueda almacenar esa expresión [7].

Los cambios que se han tenido que realizar han sido mínimos, ya que muchos de los tipos de datos de Java son equiparables a los de C. Únicamente se han eliminado los tipos de datos *Float, Byte y String*, ya que estos son los únicos que no existen como tal en C y el tipo **Long** se ha renombrado a **Long Long**. En las Figuras 5, 6, 7, 8 y 9 se puede comprobar las diferencias entre la versión original del juego y la versión adaptada para C. Para mayor claridad las figuras (b) tienen un mayor tamaño para resaltarlas frente a las figuras (a) ya que las figuras (b) contienen las modificaciones realizadas.



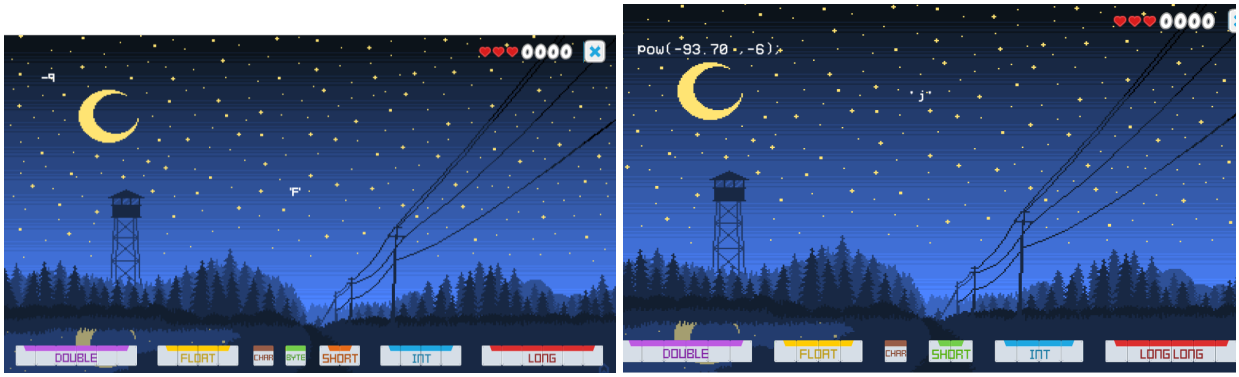
(a) Imagen de la versión original (b) Imagen de la versión de C

Figura 5: Diferencia de la interfaz entre el juego original y la versión de C



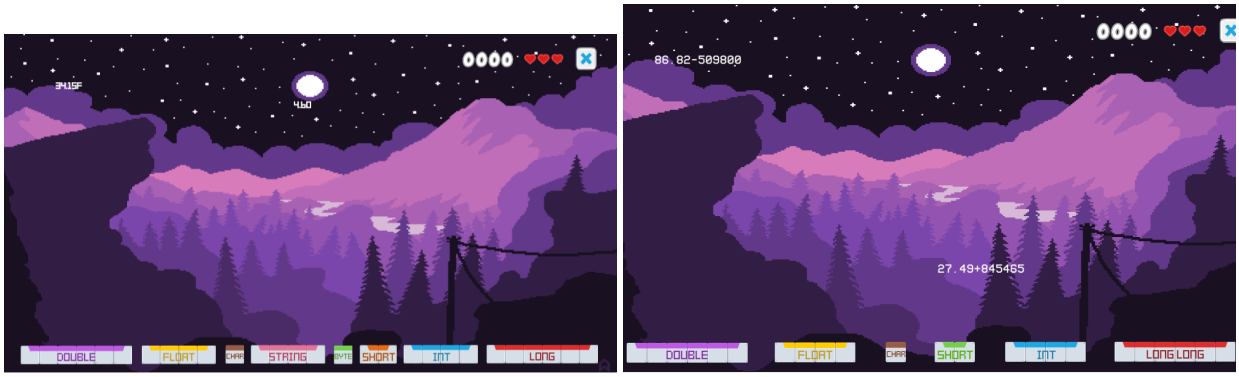
(a) Imagen de la versión original (b) Imagen de la versión de C

Figura 6: Diferencia de la interfaz entre el juego original y la versión de C



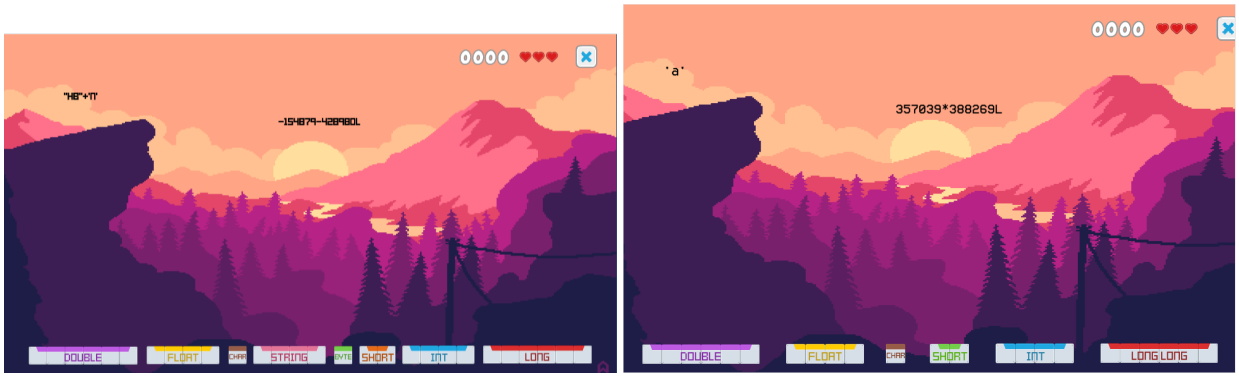
(a) Imagen de la versión original (b) Imagen de la versión de C

Figura 7: Diferencia de la interfaz entre el juego original y la versión de C



(a) Imagen de la versión original (b) Imagen de la versión de C

Figura 8: Diferencia de la interfaz entre el juego original y la versión de C



(a) Imagen de la versión original (b) Imagen de la versión de C

Figura 9: Diferencia de la interfaz entre el juego original y la versión de C

4.3. Revisión de Juegos: Cafetería

El videojuego **Cafetería** intenta mostrar las diferencias entre el paso de parámetros por valor y por referencia utilizando la analogía del contenido y continente de determinados recipientes. El mecanismo del juego consiste en simular una cafetería en la que se reciben pedidos (pizarra de la parte derecha) y hay que elaborarlos a partir de unos ingredientes y recipientes determinados (estante de la parte izquierda), utilizando una serie de métodos que realizan unas acciones determinadas (Figura 11). El objetivo del juego es obtener el pedido en el menor tiempo posible. No tiene un número de vidas determinado, por lo que se permite un número de fallos ilimitado [7].

Tras realizar la revisión de este videojuego se vio que no se tenia que realizar ningún cambio al código de este juego ya que los conocimientos que ilustraba eran perfectamente transferibles a C.

Figura 10: Menú Principal del videojuego Cafeteria

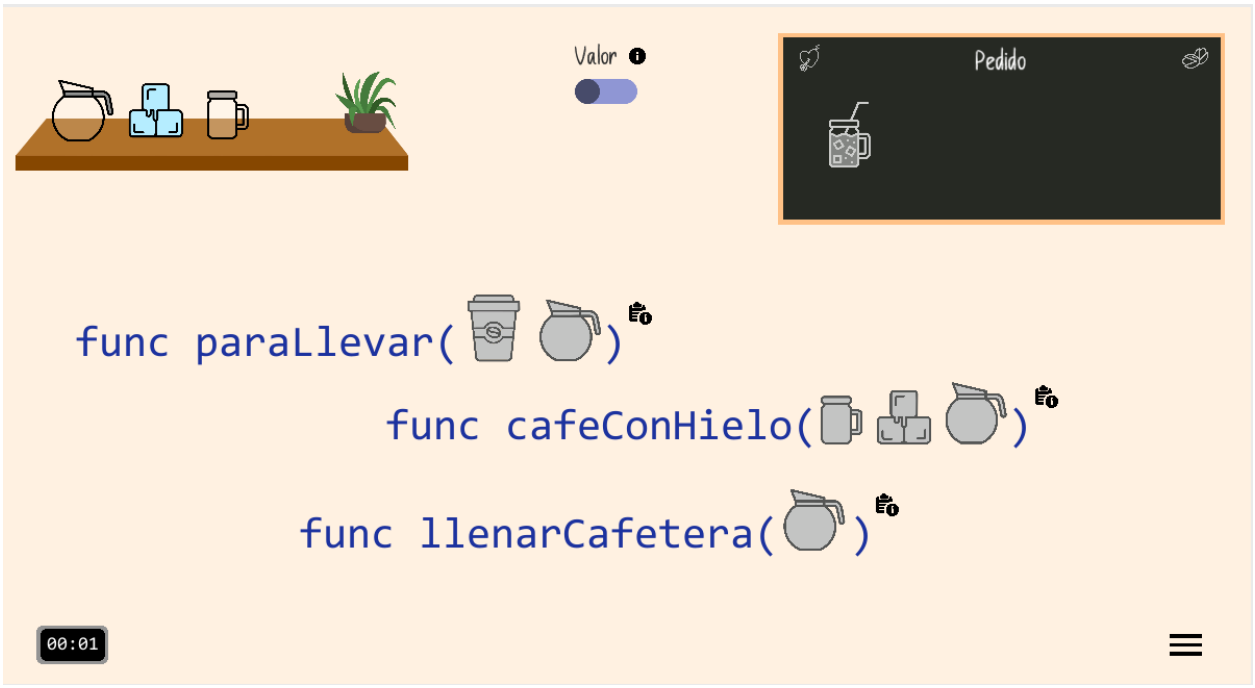


Figura 11: Captura del juego Cafeteria

5. Juego Nuevo: Musicadenas

5.1. Documento de Diseño de Juego (GDD)

5.1.1. Que es un GDD

Un Game Design Document(GDD) o Documento de Diseño de Juego es un documento estándar de la industria de los videojuegos que tiene como fin plasmar en texto el juego en desarrollo, desde su conceptualización general a detalles sobre mecánicas de juego, historia, personajes o diseño artístico [8].
La función principal del GDD es la de servir de guía entre los distintos miembros del equipo de desarrollo.

Este documento tiene un carácter informal, por lo que no tiene una estructura definida y se va actualizando conforme avanza el desarrollo. Debido a que es un documento que se actualiza a lo largo del desarrollo se suele utilizar para llevar un seguimiento de técnicas de desarrollo AGILE ya que todos los cambios y actualizaciones de funcionalidades y requisitos deben ser anotadas en el documento.

5.1.2. Introducción

Musicadenas pretende ser un videojuego didáctico sobre cadenas de caracteres en el lenguaje de programación C. Con este juego se quiere explicar el funcionamiento de algunas funciones básicas de las cadenas de caracteres en C.

Se ha elegido una temática de concierto porque los músicos deben seguir la partitura al pie de la letra y un director de orquesta es el que los dirige y asiste a seguir dicha partitura. Ya que se pretende trabajar con cadenas de caracteres aleatorios y no se van a utilizar notas musicales se quieren utilizar únicamente cantantes ya que se siente mas adecuado. Con esta temática también se quiere sacar una analogía en la que los cantantes son variables que cambian constantemente de valor y el director de orquesta es el programador que utiliza dichas variables.

El objetivo de este juego es el de crear una cadena de caracteres dada a partir de las partes de la cadena que se encuentra entre los cantantes y las teclas del teclado. Para ello se hará uso de las batutas que contienen las funciones de concatenar, borrar y copiar.

5.1.3. Audiencia y plataformas

Este juego forma parte del proyecto de gamificación para los alumnos del Grado de Ingeniería de Diseño Industrial y Desarrollo de Producto y el Grado de Matemáticas, así que la audiencia de ese juego es la de esos alumnos. La plataforma en la que se desarrollará dicho juego es el entorno web que se ha proporcionado, por lo que será accesible desde cualquier navegador

5.1.4. Mecánicas

En la figura 12 se puede ver la pantalla del juego. En la parte central inferior de la pantalla se puede ver una partitura con una cadena de caracteres la cual es la solución a la que se tiene que llegar utilizando las cadenas de caracteres que poseen tanto los bocadillos de los personajes de la parte central de la pantalla como las teclas de piano de la parte inferior derecha. Para llegar a esa solución se utilizarán las batutas situadas en la parte inferior de la pantalla para realizar la acción de concatenar, borrar y copiar cadenas respectivamente (hablaremos más adelante del funcionamiento de las batutas). Una vez se haya formado la cadena en uno de los personajes se deberá grabar en el tocadiscos de la parte inferior izquierda, diciendo si la cadena se ha formado correctamente o no.

Figura 12: Captura del menú de juego

Es importante comentar que únicamente los bocadillos de los personajes permiten modificar la cadena que contienen, ya que funcionan como cadenas de Entrada/Salida, mientras que las teclas del piano son solo de entrada.

En cuanto a las batutas, hay 3: una verde, una roja y una blanca. En primer lugar, se puede ver en la figura 13 el funcionamiento de la batuta verde. En el bocadillo en el que se pulse segundo es donde se añadirá al final de la cadena de texto los caracteres que forman la primera. El funcionamiento de esta batuta es el mismo que el de la función *strcat()* del lenguaje de programación C

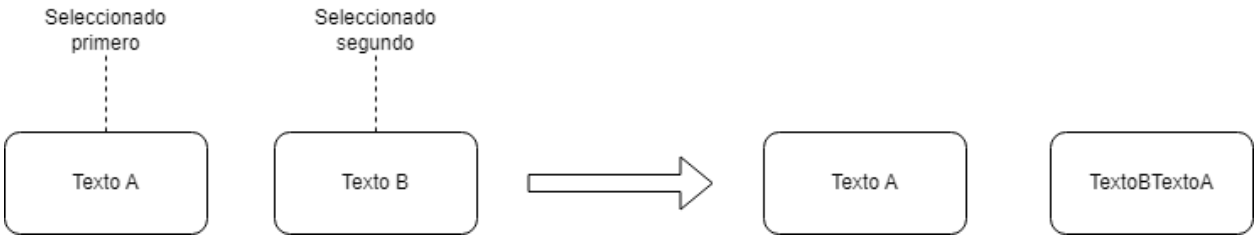


Figura 13: Funcionamiento de la batuta verde

En segundo lugar tenemos la batuta roja que tal y como se ve en la Figura 14 borra todo el texto que exista en la burbuja seleccionada. El funcionamiento de esta batuta es el mismo que el de asignar a una variable el valor *NULL* o el caracter núlo *\0*

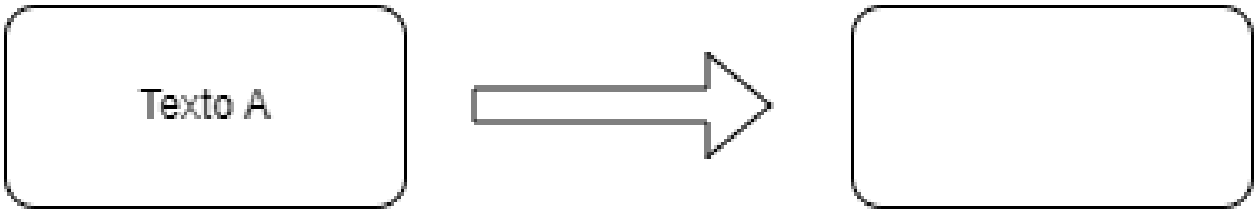


Figura 14: Funcionamiento de la batuta roja

Por ultimo, tenemos la batuta blanca su funcionamiento se representa en la Figura 15. En este caso se copiará el texto de la burbuja que seleccionemos primero en la burbuja que pulsemos en segundo lugar. El funcionamiento de esta batuta es el mismo que el de la función *strcpy()* del lenguaje de programación C

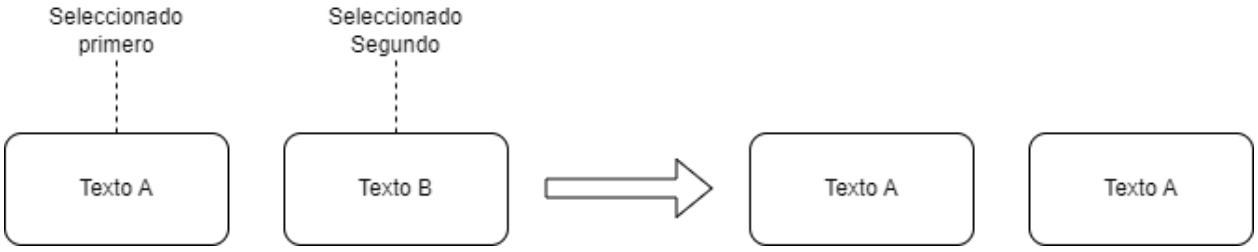


Figura 15: Funcionamiento de la batuta blanca

El jugador podrá realizar las siguientes acciones:

- Seleccionar una batuta de las disponibles para el nivel
- Ejecutar las funciones strcat(), strcpy() y var=NULL desde una tecla de piano o burbuja de texto de personaje a una burbuja de texto de personaje
- Comprobar si la solución es correcta presionando el botón de la grabadora y seleccionando la cadena que se quiere comprobar

El juego acaba una vez que se hayan resuelto 5 cadenas, independientemente de si el jugador ha acertado o no. Dependiendo del número de aciertos, al jugador se le otorgarán una cantidad determinada de puntos dependiendo del nivel (200, 300, 500 y 600 puntos respectivamente).Una vez se consigan los puntos necesarios en un nivel se desbloqueará el siguiente nivel. Los puntos que se tienen que conseguir en el primer nivel para desbloquear el segundo nivel son 500 puntos, para el tercer nivel se necesitan conseguir 750 punto en el segundo nivel y para desbloquear el cuarto se necesitan conseguir 1750 puntos en el tercer nivel.

5.1.5. Niveles

El juego consta de 4 niveles diferenciados y un tutorial donde se explican las mecánicas por medio de texto. Cada nivel supone un incremento de dificultad ya que se añaden nuevas mecánicas, o, en el caso de nivel 3 cambiar la mecánica de concatenar cadenas con la de copiar. Cada nivel contará con una cantidad lo suficientemente grande de cadenas a realizar para intentar evitar que se memoricen las cadenas y se consiga la máxima puntuación

5.1.5.1 Nivel 1

El nivel 1 el objetivo es aprender a utilizar la función de añadir (`add()`). Para ello solo se pueden utilizar la batuta verde (función strcat()) y la batuta roja (variable=NULL) como se puede ver en la Figura 16. Las partes de la cadena que tienen que formar la solución solo son colocadas dentro de las piezas del piano para simular la función añadir. En este nivel se otorgarán 200 puntos por cada acierto y estará desbloqueado desde el principio.

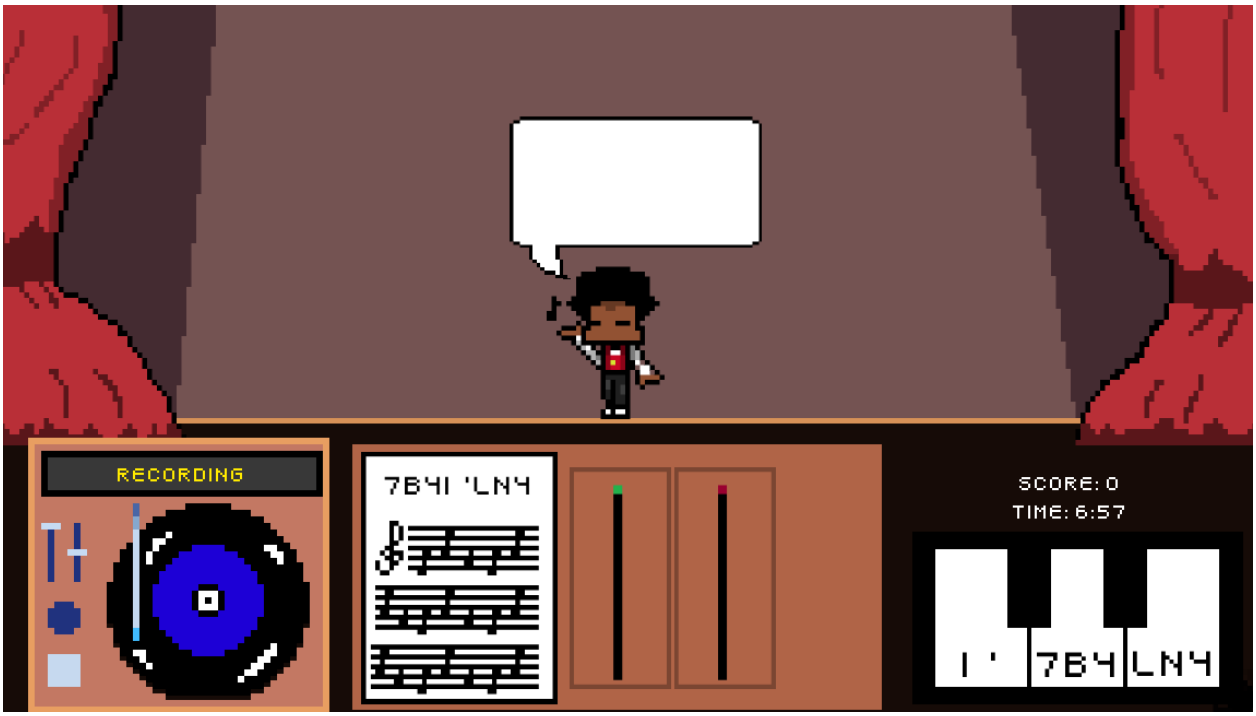


Figura 16: Imagen del Nivel 1

5.1.5.2 Nivel 2

Este nivel se desbloquea una vez que el jugador ha conseguido 500 puntos en el nivel anterior. A diferencia del nivel 1, aquí se añaden 2 cantantes nuevos y también aparecen cadenas de caracteres en los bocadillos pertenecientes a los cantantes como se puede ver en la Figura 17. Como ahora aparecen cadenas de caracteres en los bocadillos de texto de los cantantes se puede profundizar mejor en el uso de la batuta verde. En este nivel se otorgan 300 puntos por acierto.

Figura 17: Imagen del Nivel 2

5.1.5.3 Nivel 3

Este nivel se desbloquea cuando en el nivel anterior se han conseguido 750 puntos. En el nivel 3 se deshabilita la batuta verde y se habilita la batuta blanca (función strcpy()) como se puede ver en la Figura 18. El objetivo de este nivel es el de aprender a utilizar la función strcpy(). Este nivel otorga 500 puntos por cada acierto.

Figura 18: Imagen del Nivel 3

5.1.5.4 Nivel 4

Este nivel se desbloquea cuando en el nivel anterior se han conseguido 1750 puntos. En este nivel se pone a prueba todo lo aprendido anteriormente, por lo que aquí se dispone de las 3 batutas como se puede ver en la Figura 19. Este nivel otorga 600 puntos por cada acierto y .

Figura 19: Imagen del Nivel 4

5.2. Interfaz

La interfaz se puede dividir en 6 elementos: Los cantantes, la grabadora, la partitura, las batutas, el piano y las estadísticas. A continuación se explicará el funcionamiento de cada elemento.

Figura 20: Interfaz completa

En la parte superior de la pantalla se encuentran los **Cantantes**. En la Figura 21 se pueden ver que encima de los personajes hay unas burbujas de texto que contienen una cadena de caracteres. Estas burbujas de texto sirven como cadenas de caracteres de Entrada/Salida por lo que la cadena de caracteres que se tiene que crear se formará en una de estas burbujas.

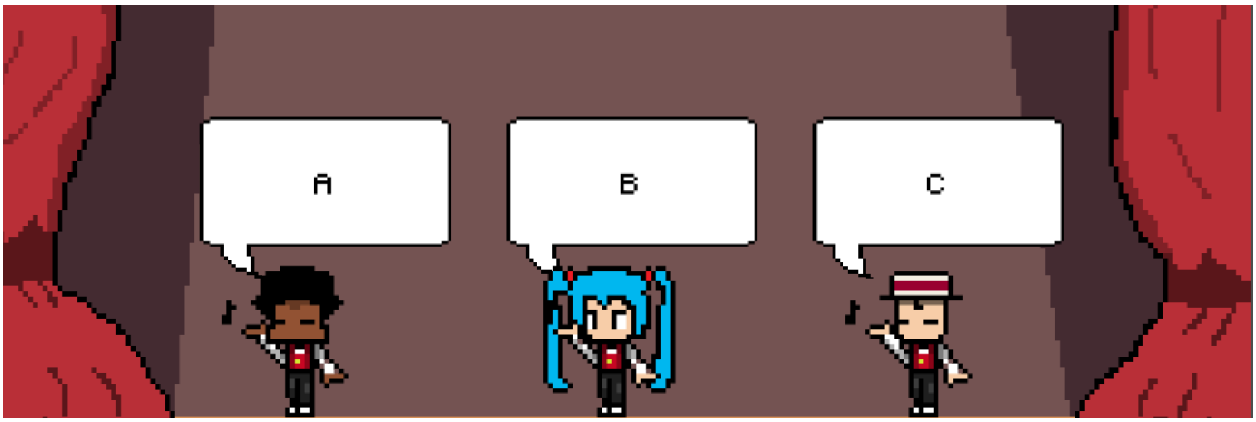


Figura 21: Cantantes en la pantalla

En la parte inferior derecha de la pantalla se puede ver el **Piano**. Este piano esta formado por 3 teclas, cada una con una cadena de caracteres distinta Figura 22. Su funcionamiento es similar al de los bocadillos de los cantantes con la única diferencia de que las teclas del piano únicamente son cadenas de entrada, es decir, no se puede cambiar el valor que contienen.

(a) Teclas del piano

(b) Teclas del piano pulsadas

Figura 22: El piano en sus 2 posibles estado

Justo encima del teclado se encuentran dos contadores: uno que muestra el tiempo restante para que se acabe el nivel y otro que muestra la puntuación obtenida tal y como se muestra en la Figura 23.

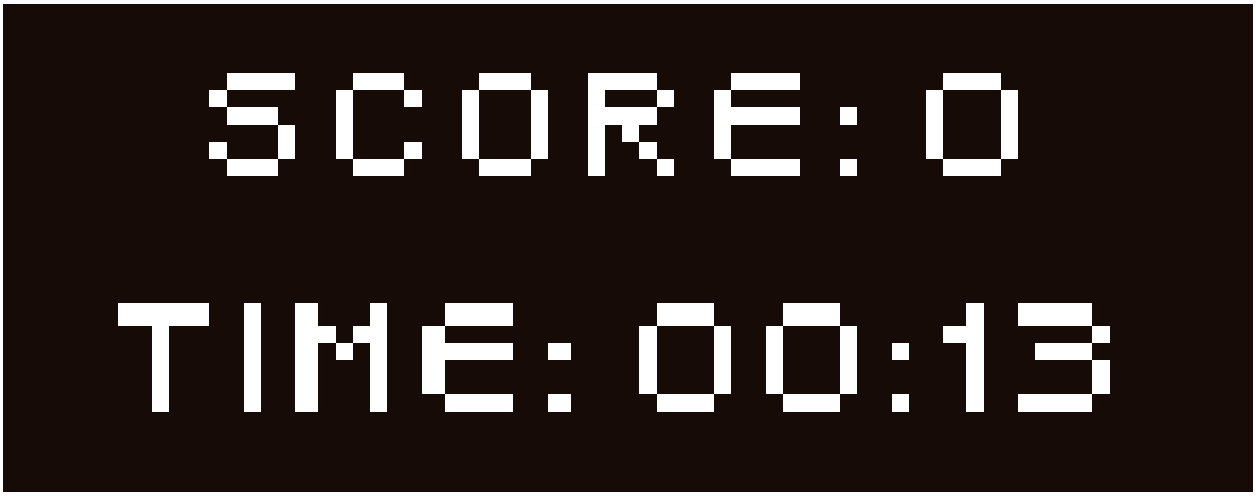


Figura 23: Estadísticas en la pantalla

A la izquierda del teclado y las estadísticas, en la parte central de la pantalla se encuentran las **Batutas** como se muestra en la Figura 24. Estas batutas sirven para ejecutar las funciones del lenguaje de programación C *strcat(), strcpy()* y *variable=NULL*. El funcionamiento de las batutas es el siguiente:

- Se selecciona una de las batutas.
- Se selecciona una tecla del piano o un cantante.
- En el caso de strcat() y strcpy() cuando se seleccione un cantante realizará la función asociada a la batuta seleccionada.

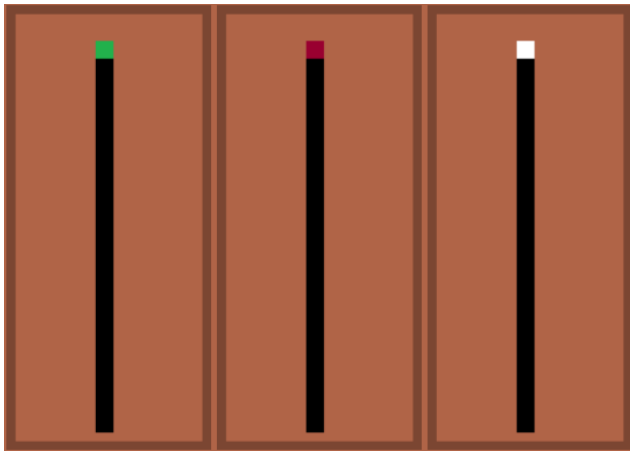


Figura 24: Batutas en la pantalla

Como ayuda visual para saber si una batuta esta seleccionada o no la imagen cambiará como en la Figura 25. La imagen también cambiará si el jugador sitúa el ratón encima de una batuta, mostrando además, un texto recordando que función realiza la batuta tal y como muestra la Figura 26

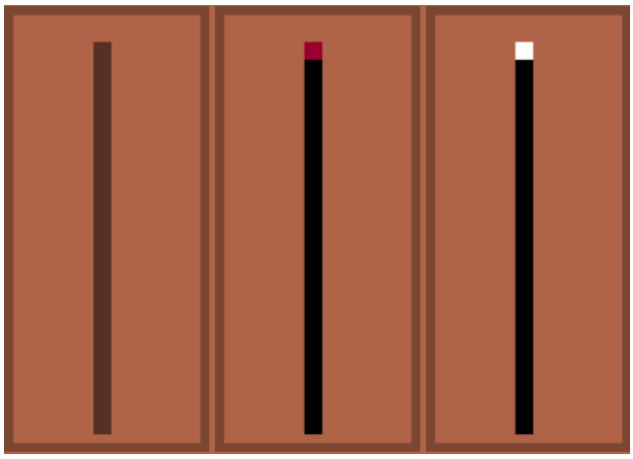
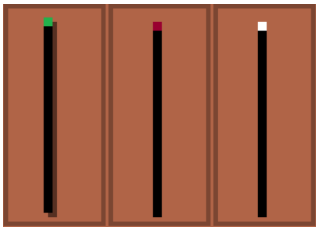
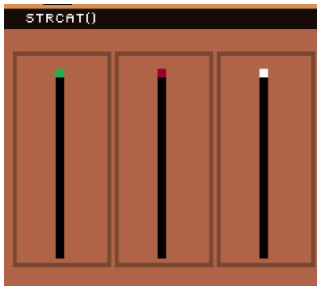


Figura 25: Batutas verde seleccionada



(a) Batuta cuando se sitúa el ratón encima



(b) Texto mostrado al situar el ratón sobre la batuta

Figura 26: Batutas cuando se sitúa el ratón encima de una de ellas

A la izquierda de las batutas podemos encontrar la partitura (Figura 27). Este elemento muestra la cadena de caracteres que se tiene que formar con las cadenas repartidas entre los cantantes y las teclas del teclado.

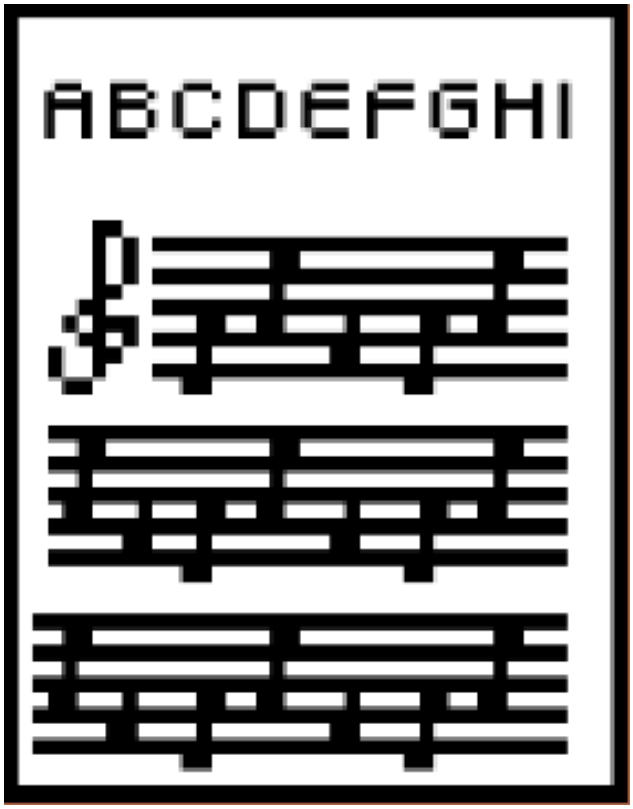
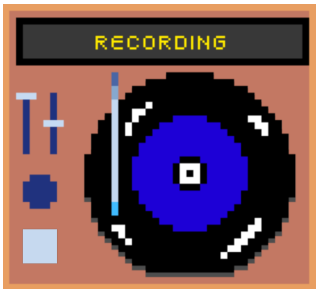
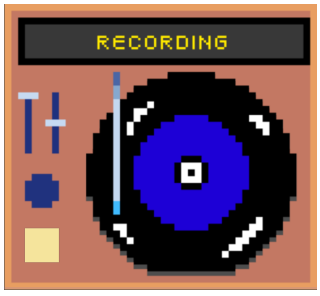


Figura 27: Partitura

Por ultimo en la parte inferior izquierda de la pantalla se encuentra la grabadora (Figura 28). Una vez se ha creado la cadena de caracteres que muestra la partitura el jugador debe pulsar en el botón de grabar y seleccionar la cadena que se quiere comprobar. En caso de que la cadena haya sido correcta el texto de GRABANDO pasará a mostrar CORRECTO o, en caso de que la cadena no coincida, el mensaje de ERROR (Figura 29)



(a) Grabadora



(b) Grabadora cuando se pulsa en el botón de grabar

Figura 28: Grabadora en su estado inicial y cuando se pulsa el botón de grabar

(a) La cadena introducida has sido correcta (b) La cadena introducida has sido errónea

Figura 29: Grabadora tras comprobar la cadena introducida

5.3. Multimedia

Los elementos utilizados para el desarrollo de este juego han sido los siguientes

- Para la creación de los **personajes, escenarios, iconos y menús** se han utilizado imágenes siguiendo un estilo ”Pixel art”. La totalidad de las imágenes utilizadas para este juego han sido dibujadas a mano utilizando la herramienta **Pixilart** [9].
- Para los **efectos de sonido** de los murmullos de la pantalla de inicio y los aplausos al acabar el nivel se han descargado de páginas de efectos de sonido de licencia gratuita [10]
- Para la **música de fondo** del nivel se ha creado a mano utilizando la herramienta **BeepBox** [11]

Dentro del juego podemos encontrar los sprites de los músicos. En el ámbito de los videojuegos un Sprite se refiere a un conjunto de imágenes que representan un personaje o un objeto. Cada vez que se inicia un nivel el diseño con el que aparece el personaje se elige aleatoriamente entre los 4 que existen. Esto se puede ver en las Figura 30

(a) Primer diseño de personaje

(b) Segundo diseño de personaje

(c) Tercer diseño de personaje

(d) Cuarto diseño de personaje

Figura 30: Diferentes diseños de personaje

Cada personaje tiene una animación para cuando esta cantando y la imagen de la figura 30 aparece cuando se pulsa en el bocadillo de texto que le corresponde para representar que ha sido seleccionado. Las animaciones se pueden ver en las figuras 31 32 33 y 34

Figura 31: Animación primer personaje

Figura 32: Animación segundo personaje

Figura 33: Animación tercer personaje

Figura 34: Animación cuarto personaje

Aparte de los personajes se ha creado un fondo animado con forma de escenario que abre y cierra el telón al iniciar y al acabar un nivel. En la Figura 35 se puede ver como quedaría el escenario con el telón cerrado que es como se ve la pantalla cuando se inicia el juego. La Figura 36 se utiliza mientras se esta jugando un nivel y la Figura 37 se utiliza como transición entre el menú principal y el nivel y para avisar de que el nivel ha terminado

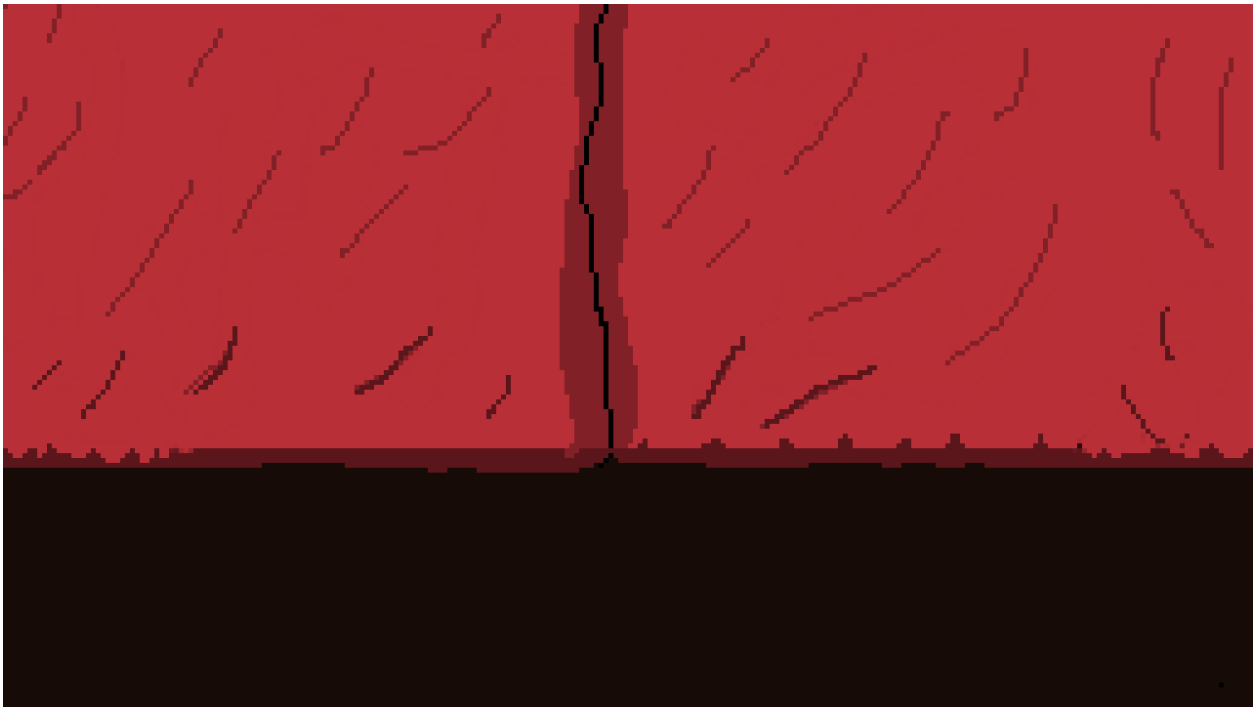
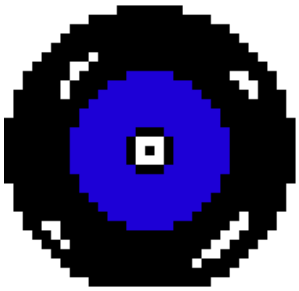


Figura 35: Telón Cerrado

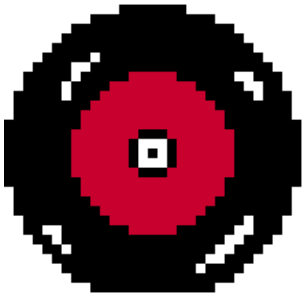
Figura 36: Telón Abierto

Figura 37: Animación del telón abriéndose

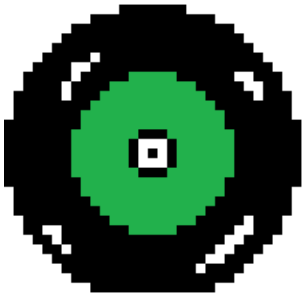
También es importante destacar los sprites de los vinilos que aparecen en la grabadora de la parte inferior izquierda de la pantalla. Al igual que los cantantes hay varios diseños de vinilo. En este caso cada vez que se comprueba si la cadena de caracteres es correcta el sprite del vinilo cambia. Este cambio sirve como ayuda visual para saber que se ha realizado la acción de grabar.



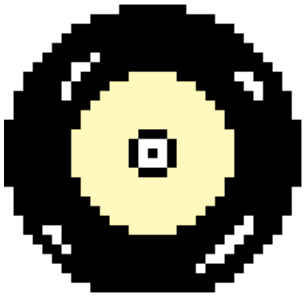
(a) Primer diseño de vinilo



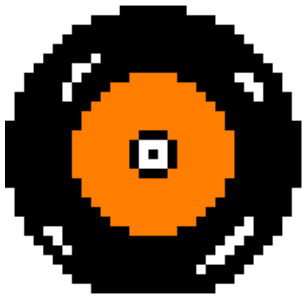
(b) Segundo diseño de vinilo



(c) Tercer diseño de vinilo



(d) Cuarto diseño de vinilo



(e) Quinto diseño de vinilo

Figura 38: Diferentes diseños de vinilo

6. Análisis

6.1. Elicitación de requisitos

6.1.1. Requisitos funcionales

En esta sección se detallarán todos los requisitos funcionales que debe cumplir el juego.

RF-01	Iniciar juego
Descripción	Al seleccionar el juego desde la plataforma este debe iniciarse y mostrar el menú principal, sin generar ningún error y pudiendo interactuar con dicho menú

Tabla 11: RF-01: Iniciar juego

RF-02	Salir del juego
Descripción	El sistema deberá permitir al usuario salir del juego desde el menú principal

Tabla 12: RF-02: Salir del juego

RF-03	Iniciar nivel
Descripción	Al seleccionar un nivel el sistema deberá cambiar a la pantalla correspondiente con el nivel seleccionado

Tabla 13: RF-03: Iniciar nivel

RF-04	Salir del nivel
Descripción	Dentro de un nivel el usuario deberá ser capaz de salir al menú principal cuando desee sin tener que reiniciar el juego

Tabla 14: RF-04: Salir del nivel

RF-05	Seleccionar batuta
Descripción	El usuario deberá ser capaz de seleccionar cualquier batuta que se muestre en la pantalla, cambiando la imagen de esta para mostrar que ha sido seleccionada

Tabla 15: RF-05:Seleccionar batuta

RF-06	Utilizar funciones de la batuta
Descripción	Dentro de un nivel el usuario deberá ser capaz de utilizar las funciones correspondientes a cada batuta al pulsar en los botones pertinentes

Tabla 16: RF-06: Utilizar funciones de la batuta

RF-07	Las teclas del piano solo servirán como salida
Descripción	Los botones con forma de tecla de teclado únicamente servirán como salida de datos y solo se cambiará su contenido cuando se haya resuelto una cadena

Tabla 17: RF-07: Salir del nivel

RF-08	Los bocadillos de texto servirán como Entrada/Salida
Descripción	La información que contienen los bocadillos debe de poder ser modificada por medio de las funciones que poseen las batutas

Tabla 18: RF-08: Los bocadillos de texto servirán como Entrada/Salida

RF-09	Guardar puntuación
Descripción	Al finalizar el nivel la puntuación obtenida debe ser guardada en la base de datos

Tabla 19: RF-09: Guardar puntuación

RF-10	Final de partida
Descripción	El juego finalizará cuando se hayan comprobado si 10 cadenas de caracteres se han formado correctamente, mostrando una pantalla con las estadísticas obtenidas en ese nivel

Tabla 20: RF-10: Final de partida

RF-11	Ganar puntos
Descripción	El usuario ganará puntos si la cadena comprobada coincide con la cadena objetivo que se muestra en pantalla

Tabla 21: RF-11: Ganar puntos

6.1.2. Requisitos no funcionales

RNF-01	Motor de desarrollo
Descripción	El sistema deberá ser desarrollado en el motor de juego Godot

Tabla 22: RNF-01: Motor de desarrollo

RNF-02	Lenguaje de programación
Descripción	El sistema deberá ser desarrollado en el lenguaje de programación GDScript

Tabla 23: RNF-02: Lenguaje de programación

RNF-03	Plataforma objetivo
Descripción	El sistema deberá ser desarrollado para su uso en Web

Tabla 24: RNF-03: Plataforma objetivo

RNF-04	Comunicación con la plataforma
Descripción	El sistema deberá comunicarse con la base de datos de la plataforma web para acceder y almacenar el progreso de cada jugador

Tabla 25: RNF-04: Comunicación con la plataforma

RNF-05	Legibilidad
Descripción	El texto del juego debe estar diseñado para poder ser leído fácilmente

Tabla 26: RNF-05: Legibilida

RNF-05	Tiempo de respuesta
Descripción	Todas las acciones realizadas en el juego deberán responder dentro de un intervalo de tiempo aceptable

Tabla 27: RNF-05: Tiempo de respuesta

6.2. Casos de uso

En los siguientes diagramas de casos de uso se pueden ver las distintas acciones que puede realizar el jugador en el menú principal **(Figura 39)** y dentro del nivel **(Figura 40)**. Este juego tiene únicamente un actor, que es el jugador

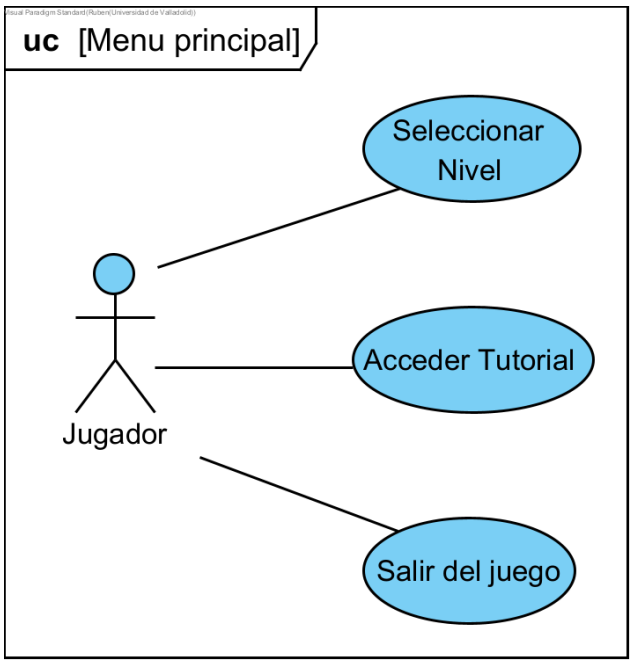


Figura 39: Diagrama de casos de uso Menú principal

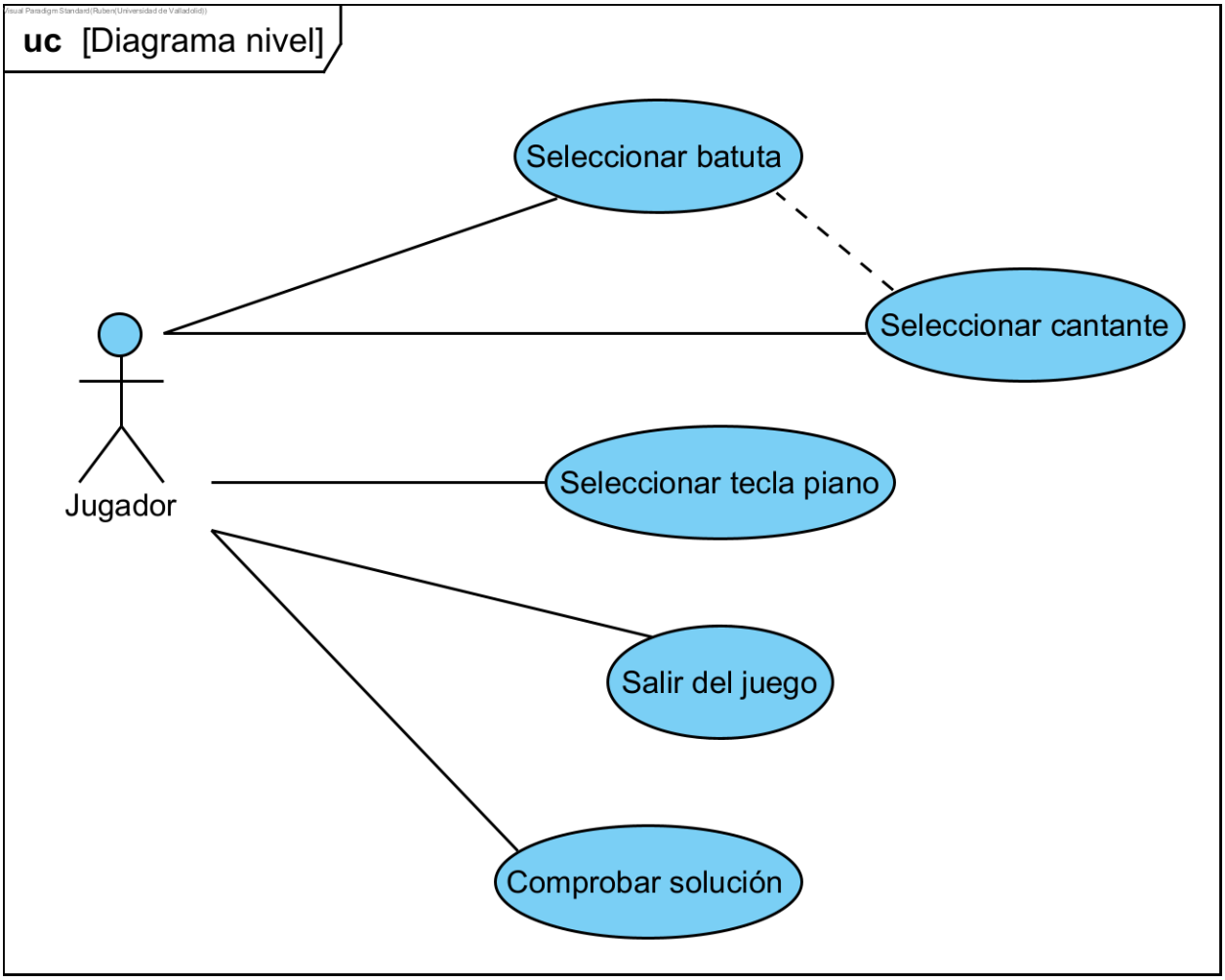


Figura 40: Diagrama de casos de uso Nivel

7. Diseño e Implementación

En esta sección se procederá a explicar como se ha diseñado el proyecto dentro de Godot. En primer lugar, vamos a detallar los elementos importantes dentro de Godot.

- **Nodo:** Son clases objeto predeterminadas con las que se trabaja dentro de Godot como puede ser un botón o un sprite. Cada nodo posee métodos y señales predeterminadas.
- **Señal:** Los nodos tienen acceso a señales las cuales se activan cuando se cumple alguna condición (Por ejemplo, en el caso de un botón cuando es pulsado mandaría una señal de que ha sido pulsado). Las señales se conectan a los Scripts del propio nodo o de otro nodo distinto.
- **Script:** En caso de que algún nodo requiera de algún método que no posea de forma nativa, se le puede asociar un script con el que se pueden crear o modificar métodos de la clase asociada.

- **Escena:** Se puede ver como un conjunto de nodos conectados a un nodo ”Principal”. Esto hace que se puedan reutilizar estas Escenas, por lo que es muy util. Una escena puede ser un personaje con varios sprites o un nivel entero formado por esos mismos personajes, una interfaz y música de fondo.

Todas las clases se han creado siguiendo las directivas de identificación del propio Godot, por lo que los métodos que se inicien a partir de una señal tendrán el siguiente formato `_on_`+ `nombreNodo_`+ `nombreSeñal`. Por ejemplo, si tenemos un botón llamado `button` y al presionarlo lanza la señal `pressed` , la clase a la que se conecta se llamará `_on_button_pressed()`.

7.1. Modelo de Dominio

En las figuras 41, 42, 43 y 44 se muestra el modelo de dominio del nivel. Como el diagrama completo es demasiado grande para verlo en una sola imagen se ha dividido en 2 partes, una con la Escena **Control** y la otra con la escena **Level**. También comentar que este modelo de dominio solo tiene en cuenta los métodos y variables añadidas por medio de un script y que contienen al menos una señal conectada a algún método.

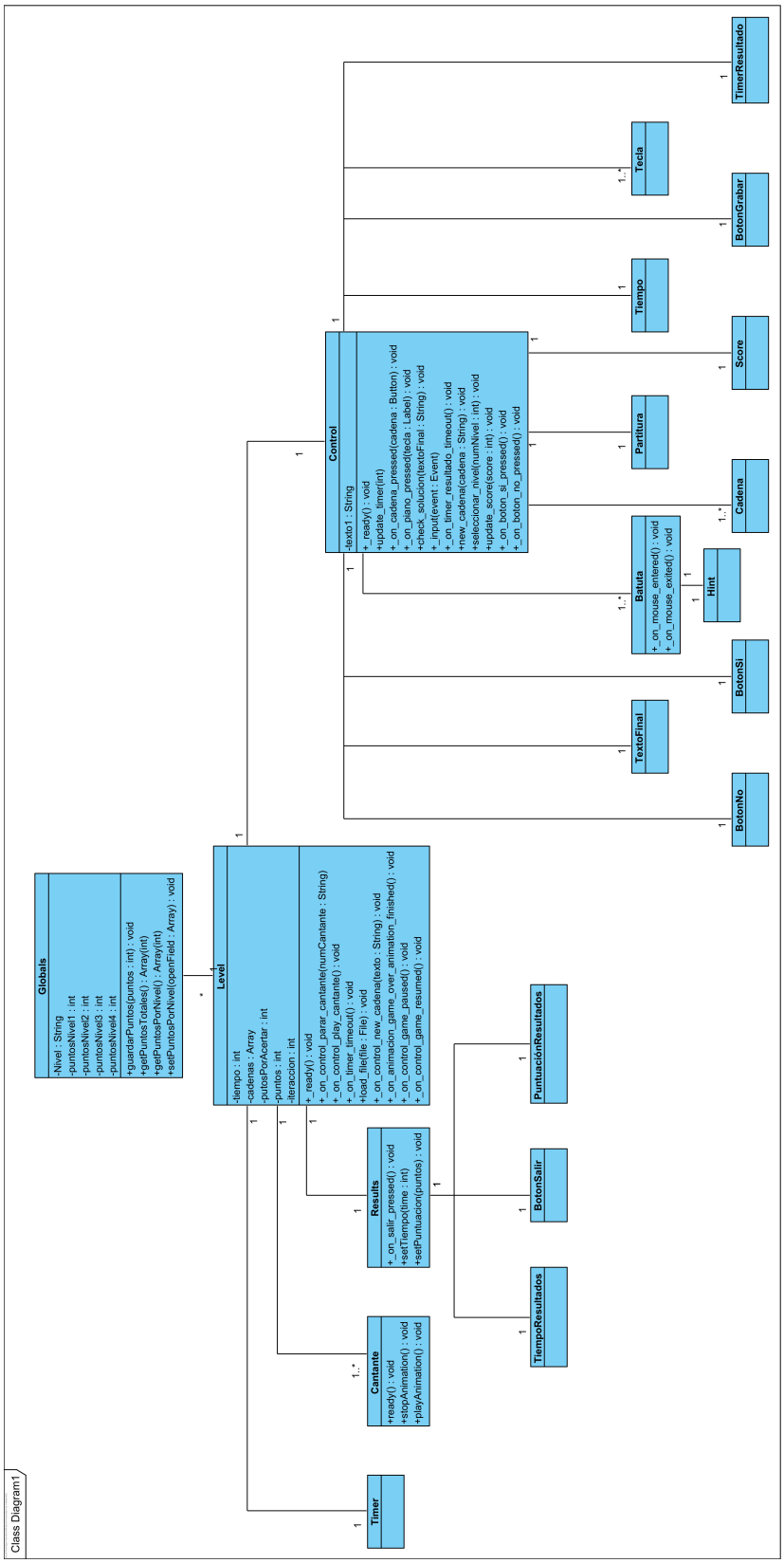


Figura 41: Diagrama de clases Completo

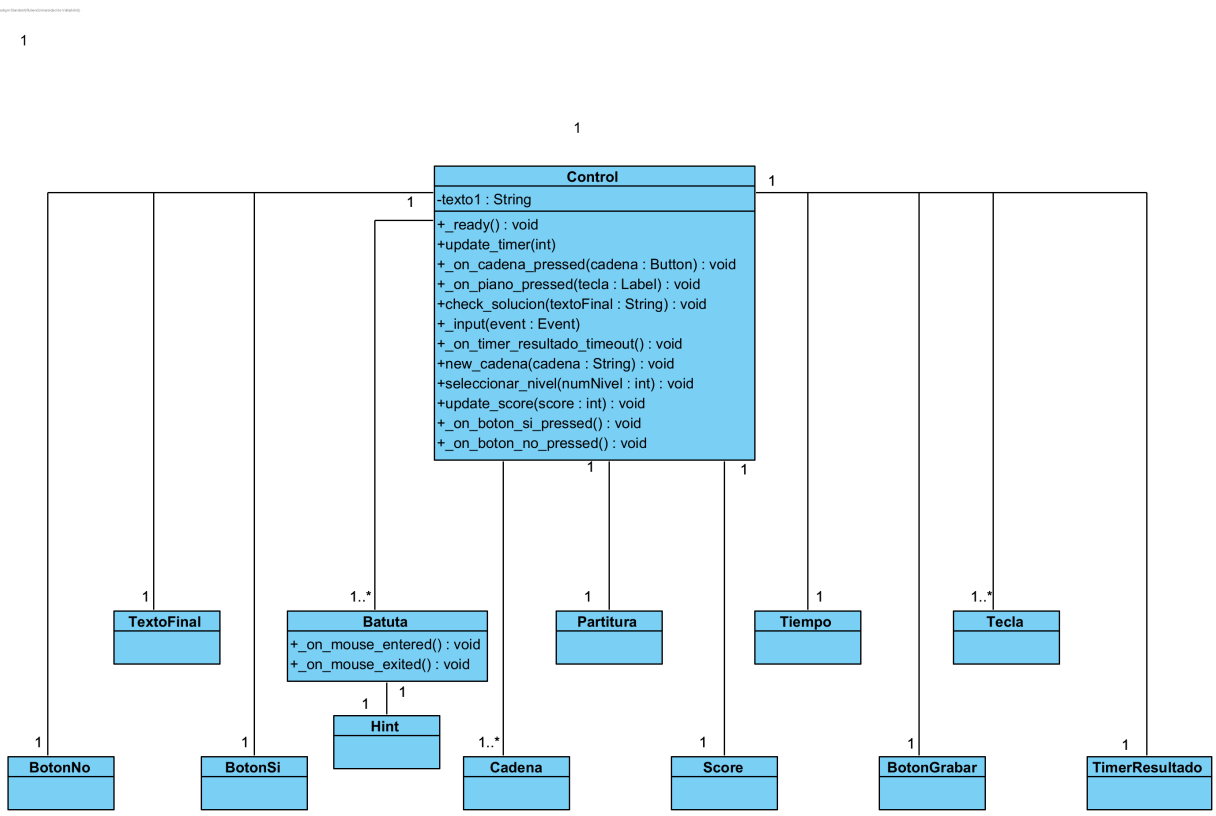


Figura 42: Diagrama de clases Control

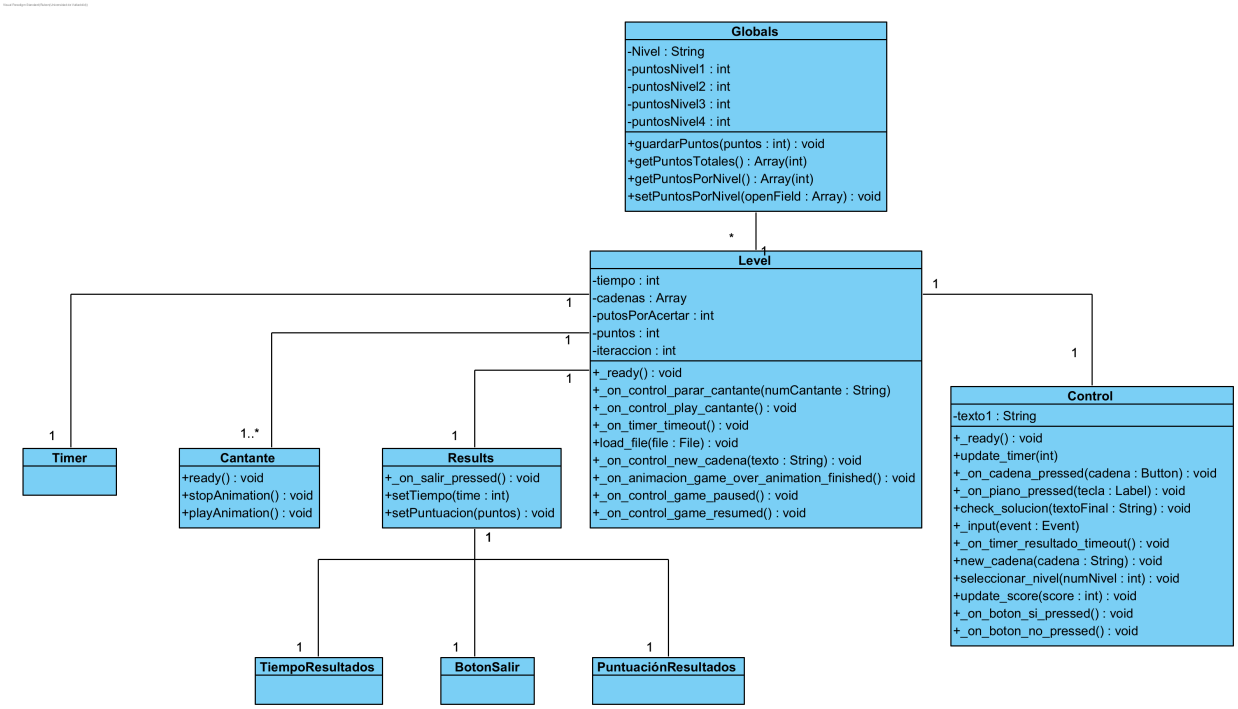


Figura 43: Diagrama de clases Nivel

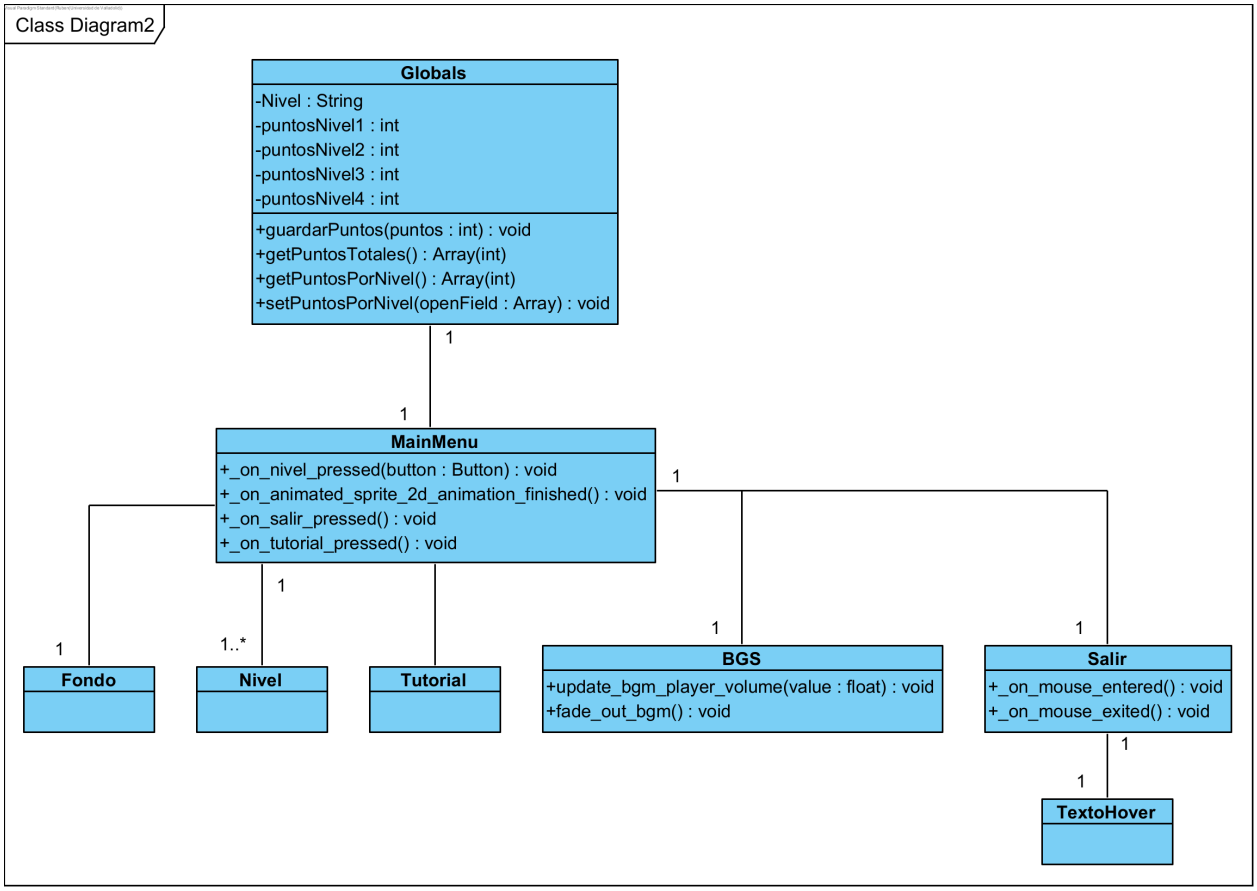


Figura 44: Diagrama de clases Menú Principal

En las figuras 43 y 44 se puede ver que hay una clase llamada Globals. Esto se debe a que es una clase *Singleton* cuyo objetivo principal es el de guardar las puntuaciones de forma local al acabar cada nivel y cada vez que se inicia el juego

7.2. Patrones de diseño utilizados

7.2.1. Patrón Mediador

El patrón mediador se utiliza para minimizar las dependencias entre objetos. Para ello, este patrón obliga a que todas las comunicaciones entre objetos pasen por esta clase mediador. Debo a la forma en la que Godot estructura las escenas, teniendo estas que partir de un nodo padre, se ha usado ese nodo como Mediador, haciendo que las señales de las clases se conecten con el nodo raíz de la escena.

Se ha decidido utilizar este método debido a la alta cantidad de nodos que deberían comunicarse entre sí enviándose datos para funcionar correctamente, tal y como se puede ver en la figura 45. Con este patrón se simplifica mucho esta relación entre nodos, ya que el mediador puede acceder a todos los datos que necesite fácilmente. Un ejemplo de la relación de clases sin utilizar este patrón se puede ver en la Figura 45

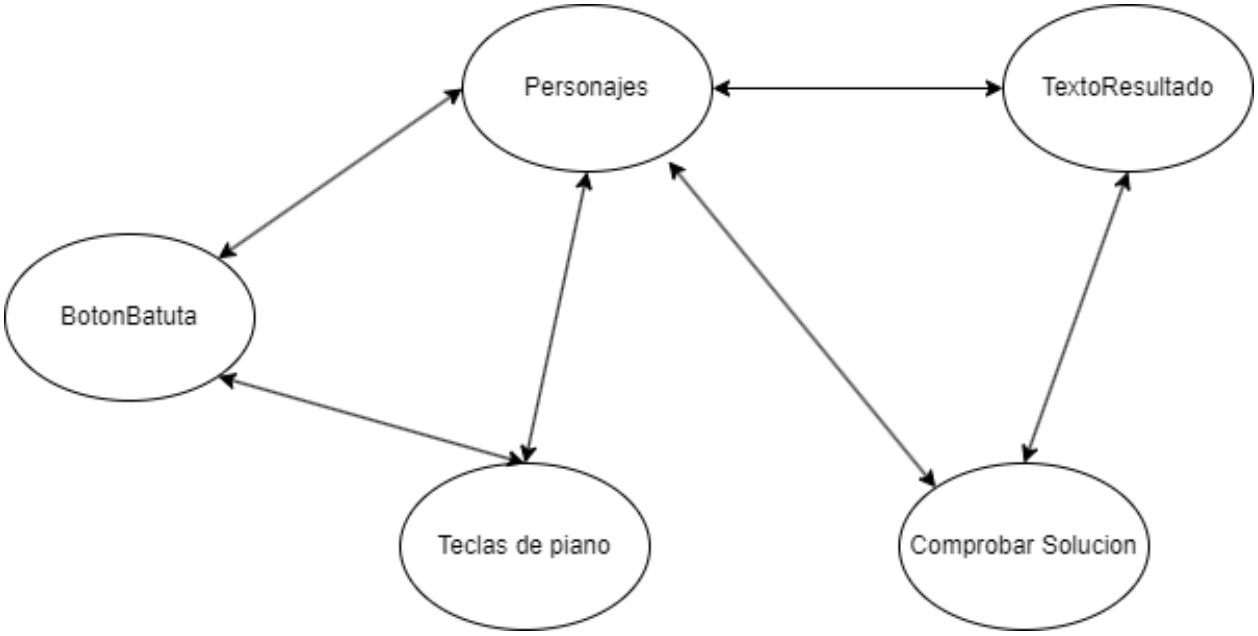


Figura 45: Posible relación de clases sin utilizar el patrón Mediador

7.3. Implementación y carga de niveles

7.3.1. Creación de niveles

Para la creación de niveles existe una única escena con todos los elementos y desde el propio menú principal se manda un mensaje con el número del nivel y al entrar dentro de dicha escena se ocultan todos los elementos que no van a ser utilizados en dicho nivel. Tal y como se puede apreciar en las Figuras 46a y 46b, el cambio en la interfaz es mínimo, por lo que aunque tener todos los elementos cargados en memoria todo el tiempo pueda suponer un incremento en el consumo de memoria de la maquina, en este caso la diferencia es despreciable para la mayoría de ordenadores modernos. Mantener todos los recursos cargados en memoria hace que se reduzca el tiempo de ejecución y como el juego esta planteado para un entorno web se ha optado por intentar reducir el tiempo de ejecución lo mas posible. Utilizando el mensaje mencionado anteriormente, también se selecciona cuantos puntos da cada respuesta acertada y qué fichero debe abrir para recoger las lista de cadenas que se utilizará en ese nivel.

(a) Imagen del Nivel 1 (b) Imagen del nivel 4

Figura 46: Diferencia de la interfaz entre el nivel 1 y el 4

Los ficheros que utiliza son ficheros .txt que siguen un patrón estandarizado para facilitar tanto la forma de añadir nuevas cadenas cuando se quiera, como luego su implementación en el código. Cada combinación de cadena que se tiene que resolver y sus elementos para resolverla se encuentra en una única línea separando dichos elementos por comas, estando primero la cadena completa, los siguientes 3 elementos son los que se colocarían en los cantantes, y los 3 últimos en las teclas del piano. Tomando como referencia la figura 47 la cadena que tendríamos que usar para que se mostrase como en dicha imagen sería `1,2,3,4,5,6,7`.

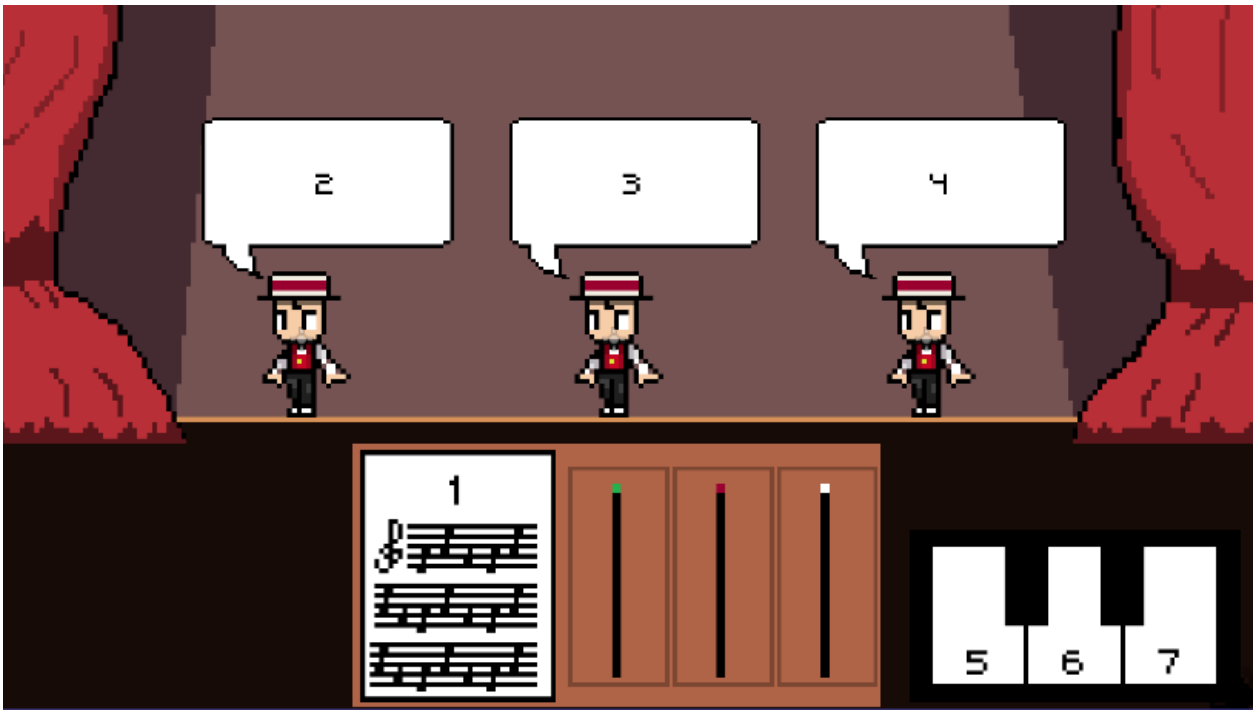


Figura 47: Orden en el que se colocan las cadenas

7.4. Integración con la plataforma

El juego Musicadenas va integrado dentro de una plataforma que es la que se encarga de iniciar, parar y guardar las puntuaciones que cada usuario obtiene al completar cada nivel y para esto es necesario que el juego sea capaz de realizar llamadas externas a la plataforma.

Por suerte Godot tiene integradas bibliotecas que permiten realizar llamadas externas fácilmente. Aprovechándonos de la API creada para otros juegos que se utilizan en la plataforma se utilizó la función de Godot `JavaScriptBridge` [12] que permite ejecutar código JavaScript dentro del propio proyecto de Godot, pudiendo llamar asi a las funciones `setOpenField` para guardar el campo abierto, `save_update` para guardar la puntuación total y `exit` para salir del juego. Las funciones setOpenField y updateScore son llamadas al acabar un nivel. Como se puede ver en el siguiente bloque de código se llama a la función eval que ejecutará la cadena de texto introducida en este caso setOpenField con los valores obtenidos en cada nivel dentro de un mismo String y en el caso de save_update se guarda un array con la puntuación total y un 0 o un 1 si se ha conseguido la estrella o no.

```
    var puntuacionPorNivel="%s,%s,%s,%s" %
        Globals.getPuntosPorNivel()
    JavaScriptBridge.eval('setOpenField("%s")'%
        puntuacionPorNivel)
JavaScriptBridge.eval('save_update([%d,%d])' %
        Globals.getPuntosTotales())
```

En el caso de la opción salir el funcionamiento sería el mismo que el comentado anteriormente. Pero en este caso la función eval se encuentra dentro del Script del menú principal al pulsar el boton de salir.

Para recoger los datos de la base de datos al iniciar el juego se tuvo que utilizar una llamada HTTP debido a que la función `getOpenField` de la API JavaScript generaba un objeto de tipo Promise que Godot no era capaz de utilizar. Para ello se hizo uso del nodo HTTP Request el cual hacia una petición al enlace https://greidi.infor.uva.es/programajugando_C/GameSpaceApi/ que posee el valor del campo OpenField de la base de datos. En este caso la función `request_completed.connect` conecta una señal a la función que se quiera una vez que se ha completado la llamada a la API (En este caso _campoLibre).

```
    $HTTPRequest.request_completed.connect(_campoLibre)
    $HTTPRequest.request("https://greidi.infor.uva.es/
    programajugando_C/GameSpaceApi/")
```

La función _campoLibre comprueba el código de respuesta recibido por la petición y en caso de que haya sido correcto guarda la puntuación de cada nivel en el Singleton Globals. Una vez ha guardado los datos comprueba si la puntuación de cada nivel es suficiente para desbloquear el siguiente y en caso afirmativo activa el botón correspondiente a dicho nivel.

```
    func _campoLibre(result, response_code, headers, body):
if (response_code==200):
print(body.get_string_from_utf8())
var puntuacion=body.get_string_from_utf8().split(",")
Globals.setPuntosPorNivel(puntuacion)
if Globals.puntosNivel1>=500:
            $"Atril/MarginContainer/HBoxContainer/
            Partitura2/VBoxContainer/Nivel 2".disabled=false
if Globals.puntosNivel2>=750:
            $"Atril/MarginContainer/HBoxContainer
            /Partitura2/VBoxContainer/Nivel 3".disabled=false
if Globals.puntosNivel3>=1750:
            $"Atril/MarginContainer/HBoxContainer
            /Partitura2/VBoxContainer/Nivel 4".disabled=false
```

8. Pruebas

En esta sección se detallarán todas las pruebas realizadas para comprobar el correcto funcionamiento del juego creado. En este caso se han utilizado test unitarios, y pruebas de integración.

8.1. Pruebas Unitarias

PU-01	Acceder a un nivel
Descripción	Se selecciona uno de los niveles del juego desde el menú principal y este debe cargar la escena del nivel mostrando la interfaz correspondiente
Resultado	Correcto

Tabla 28: PU-01: Acceder a un nivel

PU-02	Visualizar tutorial
Descripción	Al seleccionar el tutorial se debe de poder completar y volver al menú principal sin que el juego de ningún error
Resultado	Correcto

Tabla 29: PU-02: Visualizar tutorial

PU-03	Seleccionar varias batutas
Descripción	Dentro de un nivel, al seleccionar una batuta teniendo otra ya seleccionada debe deseleccionar la batuta en uso cambiando su imagen en la interfaz y cambiándola por la nueva batuta
Resultado	Correcto

Tabla 30: PU-03: Seleccionar varias batutas

PU-04	Utilizar una batuta
Descripción	Dentro de un nivel, al seleccionar una batuta y pulsar primero en un botón con texto aleatorio, y luego en un personaje, la batuta realiza la acción correspondiente
Resultado	Error

Tabla 31: PU-04: Utilizar una batuta

PU-05	Seleccionar varias teclas de piano
Descripción	Dentro de un nivel, al seleccionar una tecla de piano teniendo otra ya seleccionada debe deseleccionar la tecla en uso cambiando su imagen en la interfaz y cambiándola por la nueva tecla
Resultado	Correcto

Tabla 32: PU-05: Seleccionar varias teclas de piano

PU-06	Utilizar una tecla de piano
Descripción	Dentro de un nivel, al utilizar una batuta pulsando primero en una tecla de piano debe realizar la acción de la batuta cogiendo como primer valor el de la última tecla pulsada
Resultado	Correcto

Tabla 33: PU-06: Utilizar una tecla de piano

PU-07	Seleccionar una tecla de piano después de haber pulsado a un personaje con texto
Descripción	Dentro de un nivel, al utilizar una batuta pulsando primero en un personaje y luego en una tecla de piano, el texto de dicha tecla de piano no debe cambiar
Resultado	Correcto

Tabla 34: PU-07: Seleccionar una tecla de piano después de haber pulsado a un personaje con texto

PU-08	Seleccionar 2 veces el mismo personaje con una batuta seleccionada
Descripción	Dentro de un nivel, al utilizar una batuta en el mismo personaje 2 veces se debe realizar la acción correspondiente a la batuta utilizada
Resultado	Correcto

Tabla 35: PU-08: Seleccionar 2 veces el mismo personaje con una batuta seleccionada

PU-09	Pulsar clic derecho
Descripción	Dentro de un nivel, al pulsar clic derecho teniendo alguna batuta seleccionada, alguna tecla de piano, o el botón de comprobar solución, debe deseleccionar todos estos botones
Resultado	Correcto

Tabla 36: PU-09: Pulsar clic derecho

PU-10	Pausar nivel
Descripción	Dentro de un nivel, al pulsar el botón *ESC* el contador de tiempo debe pausarse.
Resultado	Error

Tabla 37: PU-10: Pausar nivel

PU-11	Salir del nivel
Descripción	Dentro de un nivel, al pulsar el botón *ESC* y presionar el botón si el juego debe volver al menú principal
Resultado	Correcto

Tabla 38: PU-11: Salir del nivel

PU-12	Comprobar solución acertada
Descripción	Dentro de un nivel, al comprobar una solución debe salir el mensaje de correcto y añadir los puntos correspondientes al nivel en el que se encuentra
Resultado	Correcto

Tabla 39: PU-13: Comprobar solución acertada

PU-13	Comprobar solución errónea
Descripción	Dentro de un nivel, al comprobar una solución debe salir el mensaje de error sin añadir puntos al marcador
Resultado	Correcto

Tabla 40: PU-13: Comprobar solución erronea

PU-14	Finalizar nivel
Descripción	Deberá salir la pantalla de resultados cuando se hayan comprobado 10 cadenas de caracteres y tras pulsar el botón salir, se volverá al menú principal
Resultado	Correcto

Tabla 41: PU-14: Finalizar nivel

PU-15	Desbloquear niveles
Descripción	Al acabar un nivel, si la puntuación ha sido suficiente para desbloquear un nivel, este deberá poder accederse desde ese momento
Resultado	Correcto

Tabla 42: PU-15: Desbloquear niveles

PU-16	Salir juego
Descripción	Pulsar el botón de salir del juego debe devolver al jugador a la página de inicio de la plataforma
Resultado	Correcto

Tabla 43: PU-16: Salir juego

PU-17	Iniciar juego tras haber completado algún nivel
Descripción	Si se entra al juego después de previamente haber jugado y conseguido la puntuación necesaria para completar algún nivel, esos niveles estarán desbloqueados por defecto
Resultado	Correcto

Tabla 44: PU-17: Iniciar juego tras haber completado algún nivel

PU-18	Temporizador nivel
Descripción	Al iniciar un nivel el temporizador debe marcar el tiempo correspondiente al nivel seleccionado e ir contando regresivamente hasta 0
Resultado	Correcto

Tabla 45: PU-18: Temporizador

PU-19	Ruido menú principal
Descripción	Al cambiar de escena o salir del juego de manera normal, el ruido de fondo debe parar
Resultado	Correcto

Tabla 46: PU-19: Ruido menú principal

PU-20	Música nivel
Descripción	La música del nivel debe sonar desde que se carga la escena hasta que sale la pantalla de resultados, donde dicha música debe cambiar al sonido de aplausos
Resultado	Correcto

Tabla 47: PU-20: Música nivel

8.2. Pruebas Beta

Las pruebas beta se realizan una vez el programa ha sido terminado y ha pasado todas las pruebas unitarias. En estas pruebas se les da acceso a algunos usuarios finales para que puedan utilizar el producto y reporten los errores que encuentren o vean los aspectos con los que tienen más dificultad para, de ser posible, cambiarlos antes de lanzar la versión final del juego.

Durante la realización de estas pruebas se pudieron ver los siguientes problemas:

- El tutorial no era lo suficientemente claro, por lo que se realizaron cambios en las explicaciones.
- Inicialmente cada nivel tenia 10 cadenas a completar, pero se vio que ese número era muy grande, por lo que se cambió el número a 5 cadenas.
- Inicialmente el nivel 4 poseía todas las cadenas de los niveles anteriores, pero esto hacia el ultimo nivel muy fácil, por ello se crearon cadenas de caracteres nuevas y se eliminaron las anteriores de ese nivel.
- Se añadió un limite de tiempo por nivel, ya que en la primera versión no existia ningun limite de tiempo.

9. Conclusiones

Una vez concluido el desarrollo de este Trabajo de Fin de Grado se han cumplido todos objetivos propuestos al inicio.

Se ha creado un juego serio completo que gamifica algunos conceptos del lenguaje de programación C para alumnos de otros grados diferentes al de informática, intentando facilitar el aprendizaje de estas ideas. Modificar un juego de tipos de datos Revisar un juego
Además, este proyecto tenía también la intención de utilizar el mayor número posible de herramientas *Open Source* posibles para apoyar estos proyectos, ya que se suelen obviar por otras alternativas de software propietario.

Este proyecto ha servido también para experimentar y aprender nuevos conceptos e ideas a la vez que se ponían a prueba todo lo visto anteriormente en la carrera. Por ejemplo, al haberse utilizado una metodología AGILE se han podido ver de primera mano las ventajas y los inconvenientes de utilizar este sistema. Al tener reuniones periódicas en las que hay que comentar los avances que

se han hecho, condiciona ligeramente al programador a trabajar constantemente él en el proyecto, y, en caso de duda, poder comentarla e intentar resolverla con la mayor brevedad posible. También, como se tienen estas reuniones periódicas en la que siempre hay retroalimentación, es muy común cambiar algunos requisitos o pedir cambios constantemente, haciendo que algunas partes del desarrollo se alarguen más de lo previsto.

9.1. Trabajo futuro

En un futuro se podrían realizar las siguientes modificaciones.

- Los archivos TXT podrían ser archivos externos para facilitar el que se puedan añadir o quitar cadenas.
- Modificar el apartado multimedia añadiendo más canciones, personajes o efectos de sonido
- Modificar el feedback de la interfaz, añadiendo animaciones o sonidos para que se pueda comprender mejor que esta ocurriendo en pantalla.
- Exportar el juego a distintas plataformas, como por ejemplo a dispositivos móviles.

Referencias

[1] K. M. Kapp, *The Gamification of Learning and Instruction: Game-Based Methods and Strategies for Training and Education*. Pfeiffer & Co, 2012.

[2] P. Miller, *Embracing social learning in fighting games*. dirección: https://pattheflip.medium.com/embracing-social-learning-in-fighting-games-de4eab2cde18.

[3] M. A. Martínez-Prieto, J. Silvestre, A. Bregon et al., *Una metodología basada en prácticas ágiles para la realización de Trabajos Fin de Grado*.

[4] *Sueldos para el puesto de Programmer en España*. dirección: https://www.glassdoor.es/Salaries/programmer-salary-SRCH_KO0,10.htm?countryRedirect=true (visitado 23-09-2023).

[5] *Amortizaciones*. dirección: https://sede.agenciatributaria.gob.es/Sede/impuesto-sobre-sociedades/que-base-imponible-se-determina-sociedades/amortizaciones.html?faqId=42c3904421205710VgnVCM100000dc381e0aRCRD (visitado 05-03-2024).

[6] MIT, *The MIT License*. dirección: https://mit-license.org (visitado 14-10-2023).

[7] A. Pisabarro-Marron, C. Vivaracho-Pascual, S. Arias-Herguedas, A. Ortega-Arranz y L. I. Jiménez, *Videojuegos para el aprendizaje de programación: sus características y preferencias de los estudiantes*. Jornadas de Enseñanza Universitaria de la Informática, 2024.

[8] T. Fullerton, *Game Design Workshop: A Playcentric Approach to Creating Innovative Games*. A K Peters/CRC Press, 2024.

[9] B. Ware, *Pixlart*. dirección: https://www.pixilart.com.

[10] *Mixkit*. dirección: https://mixkit.co (visitado 20-02-2024).

[11] J. Nesky, *BeepBox*. dirección: https://www.beepbox.co (visitado 10-02-2024).

[12] *JavaScriptBridge*. dirección: https://docs.godotengine.org/en/stable/classes/class_javascriptbridge.html.

[13] Werbach y Hunter, *For the Win: How Game Thinking Can Revolutionize Your Business*. dirección: https://oa.upm.es/35517/1/fundamentos%20de%20la%20gamificacion_v1_1.pdf (visitado 15-09-2023).