Herramienta inteligente para la asistencia al desarrollo de interfaces en videojuegos
Smart tool for assisting the development of UI in videogames

Trabajo de Fin de Grado
Curso 2021–2022

Autor
Sandra Mondragón Lázaro
Pablo Villapún Martín
Nicolás Fernández Descalzo

Director
Guillermo Jiménez Díaz

Grado en Desarrollo de Videojuegos
Facultad de Informática
Universidad Complutense de Madrid

Herramienta inteligente para la asistencia al desarrollo de interfaces en videojuegos
Smart tool for assisting the development of UI in videogames

Trabajo de Fin de Grado en Desarrollo de Videojuegos
Departamento de Ingeniería de Software e Inteligencia Artificial
Facultad de Informática

Autor
Sandra Mondragón Lázaro
Pablo Villapún Martín
Nicolás Fernández Descalzo

Director
Guillermo Jiménez Díaz

Grado en Desarrollo de Videojuegos
Facultad de Informática
Universidad Complutense de Madrid

29 de mayo de 2022

Agradecimientos

Sandra Mondragón Lázaro

A mi familia, por soportarme en los peores momentos y animarme siempre. Por apoyarme en los exámenes y entenderme como persona.

A David, Javi, Felipe, Marco, Antonio y por supuesto a Nico y a Pablo, por estos grandiosos y duros 4 años juntos en los que hemos vivido buenas y malas experiencias que han hecho que crezcamos juntos. Por todas esas tardes juntos resolviendo dudas y esas tardes en Moncloa mirando las nubes pasar. Porque, sin vosotros, esa dura época provocada por un virus hubiera sido mucho más dura de lo que fue. Y es que podemos llamarnos Familia Videojugadora, pero antes que eso, para mi sois, sin duda, mi segunda familia.

A Pablo, por separado y especial, por apoyarme en todo lo posible, sacarme sonrisas en los peores momentos, estudiar conmigo y soportar mis peores momentos de estrés o ansiedad.

A Guillermo, por los años de clase que hemos pasado contigo en los que hemos aprendido muchísimo, y por tu apoyo durante todo el transcurro de este proyecto. A Pedro Pablo, por la cantidad de conocimientos que hemos aprendido contigo en tan poco tiempo. Por tus explicaciones a plena voz que hacían que quisieras seguir aprendiendo.

Nicolás Fernández Descalzo

Tengo que agradecer, en primer lugar, a la familia y amigos, que me han acompañado durante toda la carrera y me han ayudado en lo que han podido.

En segundo lugar, obviamente a Pablo y Sandra, compañeros del proyecto y amigos que han pasado esto conmigo y por supuesto a Guillermo, que nos ha guiado a lo largo del proyecto y nos ha ayudado a que sea posible.

Por último, al profesorado del grado, que aunque con mejores y peores sensaciones me han enseñado lo que se hasta ahora sobre el desarrollo de videojuegos y que me ayudará de ahora en adelante.

Pablo Villapún Martín

A mis colegas de grado, por estos 4 años que hemos compartido en la universidad y fuera de ella.

A mis compañeros de TFG, tanto Nico como Sandra por su esfuerzo y dedicación en todos los proyectos y trabajos que hemos realizado a lo largo del grado, y a nuestro tutor y profesor Guille.

Y por último a todos los desarrolladores de videojuegos, con cuyas obras he disfrutado y me han hecho tomar este camino.

Resumen

Herramienta inteligente para la asistencia al desarrollo de interfaces en videojuegos

Las interfaces en videojuegos son un aspecto crucial a la hora de diseñar y ofrecer la experiencia deseada al jugador. La gran mayoría de juegos contienen un mínimo de elementos en sus interfaces de usuario, y cuanto más grande es el juego, más elementos contendrá potencialmente su interfaz. Es obvio que el diseño y creación de dicha interfaz de usuario no debe de ser arbitraria y por tanto hay que invertir una suma de tiempo importante para que realmente cumpla con su función. Sin embargo, debido a que a veces se tienen que poner muchos elementos en pantalla y existen muchas combinaciones y conceptos a tener en cuenta, el proceso se hace tedioso y se invierte demasiado tiempo en él.

En este trabajo, nuestro objetivo es proporcionar a todos estos diseñadores de interfaces una herramienta que reduzca considerablemente el tiempo de desarrollo de la misma. Esta herramienta ofrecerá una descripción de interfaz tomando los datos recogidos de un cuestionario que se realiza al diseñador sobre las características que desea, y basándose interfaces ya existentes. Este modelo se puede aplicar a cualquier género de videojuegos para generar una interfaz adecuada, aunque en este proyecto nos hemos centrado en los videojuegos de plataformas 2D y hemos realizado una implementación que permite visualizar el resultado en un motor de videojuegos.

Palabras clave

Videojuego, Interfaz de usuario, Case Based Reasoning

Abstract

Smart tool for assisting the development of UI in videogames

User interfaces in videogames are a crucial aspect when designing the desired experience to the player. The great mayority of games have a minimum number of elements in their interfaces, and the bigger the game is, the more elements will contain them. It is clear that the design and creation of it must not be arbitrary, thus a huge ammount of time is required to get the desired result. However, sometimes a high number of elements must be placed in the screen and also many concepts should be kept in mind, and because of this the process becomes tedious and people dedicates much time to it.

In this project, our goal is to provide all the interface designers a tool which considerably reduces the development time spent on it. This tool will build an interface gathering data from a survey made to the user, and basing the result in interfaces that already exists. This model can be applied to any videogame genre in order to generate a suitable interface, although in this proyect we have focused on the platform 2D genre and made an implementation that allows to visualize the result in a videogame engine.

Keywords

Videogame, User interface, Case Based Reasoning

Índice

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

"Hay totalidades, el comportamiento de los cuales no está determinada por la suma de sus elementos individuales, donde los procesos parciales son a su vez determinadas por la naturaleza intrínseca de la totalidad. Es la esperanza de la teoría de la Gestalt determinar la naturaleza de tales totalidades."
— M.Wertheimer

Las interfaces de usuario son el medio por el cual la máquina o dispositivo le indican al usuario el estado de la aplicación y ayudan al mismo a navegar por la interfaz. Actualmente existen muchas páginas web o aplicaciones que ayudan a realizar interfaces de usuario como Figma [1], Bubble [2] o Framer [3].

Durante las últimas décadas hemos visto cómo los videojuegos han ido mejorando a lo largo del tiempo, con estructuras más complejas y un aumento gradual del número de mecánicas y dinámicas que el usuario debe de aprender en el propio sistema del juego. Muchas de estas mecánicas y dinámicas requieren que la información retenida en la memoria del usuario sea mayor. Sin embargo, sabemos que el usuario medio no es capaz de retener tanta información mientras que precisa de la concentración suficiente para no perder en el juego en cuestión. Para lograr una mejor experiencia de juego, se muestra al usuario una interfaz o HUD (del inglés, Head Up Display), en la cual destacará la información importante del juego. Dependiendo del género se tiende a mostrar una información u otra con el fin de no sobrecargar la interfaz. Esto es importante, debido a que en caso de sobrecargar la interfaz, también sobrecargaríamos al usuario por la cantidad de información mostrada obteniendo así el efecto contrario al que queríamos. Muchos de los elementos (vida, escudos, puntuación, etcétera) vienen definidos por el género del juego y se ha acostumbrado al usuario (al jugador, en este caso) a que existan en la interfaz. En caso de no estar definidos, se rompería el estado de consistencia entre un juego y otro, y cada vez que un jugador quisiera aprender el nuevo entorno de un juego, sería más costoso dado que la interfaz sería completamente nueva. Por esto, las interfaces de usuario requieren de un gran desafío a nivel de diseño.

Podríamos decir entonces que uno de los principales desafíos a la hora de realizar una buena interfaz de usuario depende de la cantidad de información que queramos ofrecer al usuario. Otro de los principales desafíos a contar es cómo funciona el cerebro humano. Nuestro cerebro sigue una serie de principios psicológicos que se aplican a todos los ámbitos

[1] Figma: `https://www.figma.com/ui-design-tool/`
[2] Buble: `https://bubble.io/welcome`
[3] Framer: `https://www.framer.com/`

de la vida cotidiana. Hasta el diseño de la interfaz de los fogones de la cocina, que puede estar bien hecho (mapeando la posición de los fogones con la posición los interruptores) o ser un desastre. Esto, por supuesto, también se aplica a interfaces de usuario digitales (sean de aplicaciones o de videojuegos).

Sumado a todo esto, las interfaces en videojuegos pueden resultar incluso más costosas de entender debido a que el diseño del arte de la interfaz suele ser dado por el concepto artístico del juego en sí. Esto hace que una interfaz de un juego difiera, de manera sutil, de otro. Ya hemos hablado de que diferenciar interfaces puede provocar al usuario confusión. Por esto, es necesario tener en cuenta los principios psicológicos anteriormente mencionados para que esto no ocurra.

La realización de una interfaz de usuario adecuada es un proceso complejo de realizar, puesto que requiere muchos conocimientos previos y muchas pruebas con usuarios. Poniéndonos en la piel de un diseñador, el proceso de creación de una interfaz de un videojuego de un determinado género suele venir dado mediante la precondición de tener experiencia previa y conocer interfaces del género de juego en cuestión, con el fin de partir de una base y no romper la consistencia externa entre uno y otro. Si rompemos esta consistencia, podríamos confundir al usuario. El problema recae en que esta información (la cual queremos añadir a nuestra interfaz), no siempre es la misma que en otros juegos. Por ello copiar la interfaz de usuario de otro juego eliminando los elementos que no tengamos no siempre es la mejor solución. Es por ello que es importante partir de una base o de una interfaz prehecha o un pequeño prototipo el cual el diseñador pueda modificar a su gusto sin perder la consistencia de la que hablábamos anteriormente. Sin embargo, en la actualidad, aunque existen diversos programas o herramientas que proporcionan asistencia para la creación de interfaces de usuario para aplicaciones, no existe ninguna ayuda o apoyo al diseñador de interfaces para videojuegos.

1.1. Objetivos

Siguiendo esta línea de pensamiento, en este proyecto nos encargamos de realizar una herramienta inteligente que ayuda en el diseño de interfaces de usuarios adecuadas para cada género. Esta herramienta genera una definición de interfaz prototipo para que el diseñador pueda darle algunos detalles más. Para ello, se partiría de un conocimiento previo de interfaces que han tenido éxito, así como de los principios psicológicos de los que se ha hablado con anterioridad.

Para alcanzar el objetivo principal nos hemos planteado los siguientes subobjetivos:

- Investigar en el diseño de las interfaces de usuario, en los elementos que las componen y en las leyes que rigen la organización de estas interfaces.

- Diseñar e implementar una herramienta basada en un sistema de razonamiento basado en casos para ayudar en el diseño de la interfaz basándonos en interfaces previas y en reglas de diseño con el fin de que nos de la solución al problema que nos plantea el usuario mediante la información cualitativa proporcionada por el usuario.

- Ejemplificar el uso de la herramienta para un caso de estudio en concreto.

- Evaluar y validar la herramienta desarrollada.

1.2. Plan de Trabajo

Para el desarrollo de este trabajo se emplea el uso de la metodología ágil Scrum. Las reuniones o sprints se han realizado semanalmente con el fin de comprobar que trabajo se había realizado y que problemas había dado. No se hizo uso de ninguna herramienta adicional de seguimiento dado que el equipo constaba de una buena comunicación y apoyándonos en Discord contemplamos la tareas que se habían finalizado. Al ser un sprint semanal el primero de ellos consistía en evaluar como iba el trabajo y el siguiente a este se realizaba con el tutor.

Como hemos dicho, con el fin de ver el seguimiento del proyecto se realizará una reunión cada dos semanas con el tutor y los miembros del proyecto con el fin de establecer nuevas metas. En estas reuniones se informará del desarrollo del mismo y de las posibles dudas técnicas que hayan aparecido durante las dos semanas de trabajo.

Respecto al plan que se seguirá para realizar el proyecto, primeramente, realizaremos una investigación acerca de Case Based Reasoning y acerca de las Leyes de la Gestalt con sus respectivos resúmenes. Después nos centraremos en investigar interfaces de usuario de dos géneros en concreto: plataformas 2D y First Person Shooter, recopilando información útil de cada una de ellas.

Tras esto, se realizara un prototipo de sistema de CBR. Una vez se comprueben que el ciclo de entrada de datos, CBR y salida de datos funcione correctamente, se pasará a diseñar y desarrollar un prototipo de la herramienta relacionada con los objetivos reales del proyecto.

Posteriormente pasaremos a formar el caso de uso. Este caso de uso consistirá en usar la herramienta para desarrollar una interfaz funcional en un entorno de desarrollo de videojuegos, donde podamos visualizar el resultado final de nuestro proyecto. Tras esto se añadirán casos al sistema CBR basados en las interfaces que se han investigado en un primer punto.

Por último con el fin de comprobar nuestra herramienta funciona correctamente, realizaremos una evaluación de la misma.

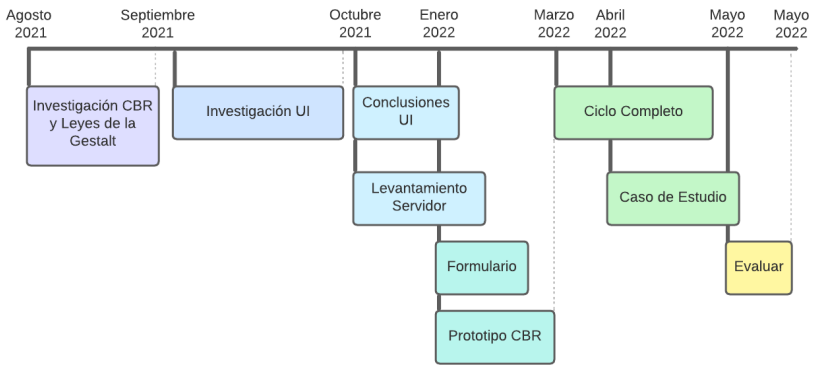

Figura 1.1: Plan de Trabajo

1.3. Estructura de la Memoria

Esta memoria se divide en diferentes apartados los cuales se describirán a continuación:

- En el capítulo 2, *Estado de la cuestión*, se indica todo el apartado de investigación

sobre los tipos de interfaces de usuarios de videojuegos y los tipos que existen, la información y análisis de las leyes de Gestalt, una descripción del Case Based Reasoning junto con su funcionamiento, etapas y aplicaciones, y por último la síntesis de las conclusiones sacadas tras investigar los elementos comunes de muchas interfaces de videojuegos para dos géneros concretos (Plataformas 2D y First Person Shooters).

- El capítulo 3, *Diseño de la herramienta*, se dedica al diseño de la herramienta: aquí se encuentran todas las decisiones que han sido tomadas de cara al funcionamiento e implementación de nuestra herramienta.
- En el capítulo 4, *Implementación*, encontramos explicaciones sobre la implementación de la herramienta, donde se encuentran los detalles mas técnicos del desarrollo de la misma.
- El capítulo 5, *Caso de estudio*, trata sobre el diseño e implementación de la descripción cualitativa que obtenemos del usuario y sobre la herramienta generada en Unity para poder visualizar la definición de la interfaz.
- En el capítulo 6, *Evaluación*, se explica como se ha realizado la evaluación de la herramienta y los resultados obtenidos de la evaluación.
- En los capítulos 7, *Conclusiones y Trabajo Futuro* y 8 *Conclusions and Future Work*, se cuentan las posibles mejoras o añadidos que se pueden realizar al proyecto actual.
- En el capítulo 9, *Contribuciones*, se explica cada una de las contribuciones de los miembros del equipo al proyecto.
- En el capítulo 10, *Anexo*, se encuentran las tablas realizadas para recoger información sobre las interfaces de usuario del género de plataformas 2D y FPS.

Capítulo 2

Estado de la Cuestión

En este capítulo comenzaremos a hablar sobre todo el trabajo de investigación que hemos realizado para formar la base del proyecto. Por una parte, recogeremos toda la información que hemos recopilado sobre interfaces en el mundo de los videojuegos, tanto los tipos que existen y sus diferencias, como los conjuntos de leyes y principios de Gestalt, aplicadas generalmente en el preámbulo de interfaces de usuario. Por otra parte, también nos centramos en sintetizar toda la información posible sobre los sistemas de razonamiento basado en casos (Case Based Reasoning o CBR), que sería la base de nuestro proyecto. Por último, detallaremos un estudio sobre los elementos comunes de distintas interfaces que hemos recopilado y analizado tanto para el género de plataformas 2D como para First Person Shooters (FPS).

2.1. UI en videojuegos

Las interfaces de usuario son un medio de comunicación entre la máquina, equipo, computadora o dispositivo con el usuario. Cuando hablamos de interfaces de usuario en videojuegos hablamos de uno de los medios de comunicación entre el usuario y el videojuego. Su objetivo es brindar la información necesaria para que el usuario interactúe con fluidez durante el juego, además de hacerle entender el estado del juego en todo momento. Guiará al usuario de manera directa o intuitiva, para que pueda accionar y recorrer la narrativa correctamente. Cuando se diseña una interfaz se tienen en cuenta varios puntos como el entorno, el contenido y el diseño visual o definición del arte. Según la clasificación que realizaron en su estudio Erik Fagerholt y Magnus Lorentzon (Fagerholt y Lorentzon (2009)), existen cuatro tipo de interfaces en videojuegos.

- **Diegéticas**: Interfaz que pertenece al mundo del juego, es decir, que el personaje que controla el jugador puede verla u oírla y en algunos casos interactuar con ella. Se encuentran en el mundo de manera inmersa.

- **No diegéticas**: Interfaz que no se ubica dentro del mundo del juego, sólo es visible y accesible para el jugador en su monitor. Son las más comunes.

- **Metas**: Se utiliza para representar hechos que pueden ocurrir en el mundo del juego, pero que no necesariamente se localizan espacialmente dentro de él. Un ejemplo son las manchas de tierra en la pantalla para representar que un coche entra en una zona de barro.

- **Espaciales**: Elementos que se encuentran dentro del mundo del juego, pero sólo son visibles para el jugador real para orientarlo o ayudarlo, ya que dentro de la narrativa del juego no es necesario que el personaje de juego conozca estos elementos.

2.2. Leyes de la Gestalt

Para diseñar de forma adecuada cualquier interfaz, deben seguirse ciertos criterios y evitar realizar prácticas que se han demostrado no ser ideales para cumplir con el objetivo de dicha interfaz. Las Leyes de la Gestalt (Hodent (2017, 2019)) son una serie de reglas que explican el origen de las percepciones a partir de determinados estímulos. Estas son usadas para aplicar bases y estudios de psicología a distintos campos, en nuestro caso al diseño de interfaces. Entre ellas, podemos destacar:

- **Principio de Semejanza**

  Cuando varios elementos comparten cierta funcionalidad, deberán de tener un diseño visual similar con el fin de que el usuario pueda intuir dicha semejanza. En caso contrario, en el que dos elementos divergen en su funcionalidad, es muy recomendable que ocurra también en su aspecto visual dentro de la interfaz para que el jugador no pueda asociar algún tipo de relación directa entre ellos.

  Este principio puede observarse en el videojuego Minecraft, donde el jugador puede almacenar bloques de construcción en su inventario que usará para construir todo tipo de estructuras, o de forma contraria podrá destruir los bloques del mundo gracias a ciertas herramientas que también guarda en su inventario. Para indicar al jugador si un objeto de su inventario sirve para construir o para destruir, los bloques de construcción siempre son representados en forma de cubo (aunque cada uno con sus colores específicos dependiendo del tipo de bloque) y las herramientas para destruir nunca toman esta forma, sino que adoptan una forma más plana y nunca cuadrada (Figura 2.1).

Figura 2.1: Ejemplo del principio de semejanza

- **Principio de Continuidad**

  El ojo va a seguir siempre el camino visual más coherente y sencillo. Es atraído por la continuidad de una línea o una curva. Este principio puede ser usado para posicionar elementos parecidos en líneas, ya sean los coleccionables o las diferentes habilidades. (Figura 2.2)

- **Principio de Figura y Fondo**

  Este principio sostiene que la mente humana necesita establecer un primer plano (el cual es normalmente en el que deposita su atención) y un segundo plano que se corresponde con el fondo. Sin un fondo claro, el primer plano no puede diferenciarse ni sostenerse, por lo que la mente no puede centrarse en él con claridad.

Figura 2.2: Ejemplo del principio de continuidad

Una mala aplicación de este principio puede observarse en el videojuego de Gris (Figura 2.3), donde algunos elementos que pertenecen al fondo del nivel (como por ejemplo algunas columnas) se dibujan con colores muy parecidos a los que tienen los elementos del primer plano, creando confusión en algunos momentos del juego ya que un elemento que está situado en el fondo se interpreta como elemento de primer plano.

Figura 2.3: Ejemplo de mal uso del principio de figura y fondo

- **Principio de Igualdad**

  De la misma manera en la que se debe evitar la ambigüedad entre figura y fondo en las imágenes que usemos, debemos evitar la confusión en una imagen que pueda parecer más de una cosa.

  Un ejemplo que da Celia Hodent en su libro Gamer's Brain (Hodent (2017)) es que en el diseño de un juego FPS, en una imagen que representaba un radar, ésta era reconocida por muchas personas como un trozo de pizza. Una vez desarrollada esta idea se hacía complicado ver el radar, así que fue modificada. (Figura 2.4)

- **Principio de simetría**

  Este principio indica que elementos situados simétricamente o delimitados por unos límites simétricos son concebidos como un grupo. Este principio puede ser aplicado

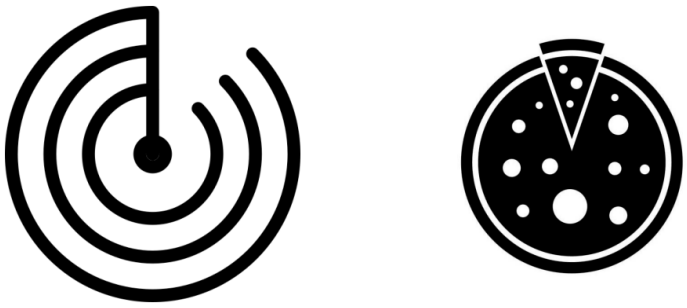

Figura 2.4: Ejemplo del principio de igualdad

en videojuegos para agrupar elementos relacionados entre sí.

- **Principio de proximidad**

  Cuando varios elementos comparten una característica en común podemos agruparlas de forma próxima con el fin de que el usuario interprete que tienen un uso o funcionalidad parecida.

  Este principio puede ser muy útil a la hora de organizar elementos de la interfaz sin la necesidad de líneas para delimitar el espacio o flechas que sugieran una orientación, ya que su propia disposición en pantalla junto con el resto de elementos próximos pueden sugerir su agrupamiento al jugador.

  Un ejemplo de mal uso del principio de proximidad podría ser el árbol de habilidades de FarCry4 (Figura 2.5), donde por proximidad, podría entenderse que el orden de mejora es por columnas, ya que cada elemento está más cerca del que tiene debajo que del que está a su lado. Sin embargo, el orden de mejora es por filas, de arriba a abajo y de izquierda a derecha. Debajo se propone una mejora posible que podría acabar con la duda (Figura 2.6).

Figura 2.5: Ejemplo del principio de proximidad

- **Principio de Dirección común**

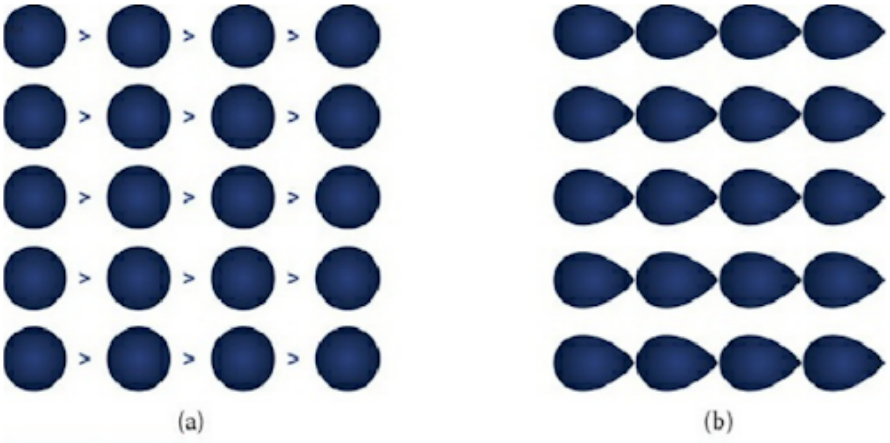

Figura 2.6: Propuesta de mejora de la Figura 2.5 haciendo buen uso de los principios de proximidad y continuidad

Parecido al de continuidad los objetos que forman un patrón en la misma dirección son percibidos como parte de un grupo. Tiene el mismo uso que el de continuidad. (Figura 2.7)

Figura 2.7: Ejemplo del principio de Dirección Común

2.3. Investigación sobre elementos comunes en interfaces de videojuegos

Para entender y clasificar los elementos que comparten la mayoría de interfaces de videojuegos, hemos hecho una revisión de los elementos más comunes de interfaces de varios juegos, dividiéndolos en dos grupos de acuerdo al género del juego. Para ello, nos hemos basado principalmente en la información proporcionada por la página Game UI Database[1]. Llamamos elemento de la interfaz (a partir de ahora elemento) a aquellos iconos que se encuentran en la interfaz y dan información al jugador sobre el el estado del juego. Un ejemplo de un elemento sería la vida o la puntuación, las cuales pueden ser representadas de diferentes formas dependiendo del género del juego.

Los dos géneros sobre los que hemos decidido realizar la investigación son Plataformas 2D y First Person Shooter. Dicha división viene dada porque las interfaces de los juegos que se incluyen en cada grupo comparten muchas propiedades y son más fáciles de clasificar

[1]Game UI Database: `https://www.gameuidatabase.com/`

dividiéndolos en estos dos grupos. Tras escoger y analizar las interfaces de varios de ellos, volcamos la información en tablas donde podíamos consultar para cada juego si contiene cierto elemento en su interfaz y, en caso de tenerlo, las propiedades de dicho elemento en ese juego. Estas propiedades son las siguientes:

- La posición del elemento, que indica su localización en el espacio de la interfaz. Permite distinguir entre elementos posicionados a los lados de la pantalla, si están centrados o no y su posición vertical. Las posiciones posibles son las mismas que se explicarán más adelante (Figura 3.2).

- La representación de cada elemento, siendo esta la forma en la que se representa en pantalla, desde un número hasta un icono o imagen para representar el elemento en cuestión.

- La importancia del elemento, tomando como importancia el valor que le da el diseñador en pantalla al elemento en concreto, siendo más grande visualmente o menos dependiendo del valor gradual de la importancia.

- Si el elemento representado en pantalla, en caso de que sea numérico, adquiere un valor discreto o continuo. Esto es muy importante a la hora de definir el aspecto que tendrá dicho elemento en la interfaz porque hay representaciones más afines a valores continuos y otras a valores discretos.

Un ejemplo sería la representación de los puntos de vida de un jugador: puede representarse como un conjunto de corazones dibujados en pantalla, donde cada uno representa un punto de vida, y cuando pierdes todos significa que el jugador pierde la partida, mientras que su valor continuo se correspondería con una barra, en la que el jugador no conoce el valor exacto de puntos de vidas que le quedan, sino que su conocimiento se limita a saber que cuando la barra se haya vaciado, significará que ha perdido la partida.

2.3.1. Plataformas 2D

Para el estudio de las interfaces de videojuegos de Plataformas 2D, seleccionamos un total de 22 juegos de dicho género para los cuales rellenamos las tablas de información con los siguientes elementos: vida, puntuación, escudos/maná, coleccionables, arma/habilidad, tiempo, progreso y minimapa. Una vez concluido este estudio fueron extraídos los siguientes atributos comunes en sus interfaces:

- **Tiempo**: Este elemento puede adoptar dos tipos de uso:

  - Cronómetro: Tiempo que ha pasado desde el inicio del nivel que el jugador está jugando.
  - Cuenta atrás: Una cuenta decreciente indicando el tiempo que le queda al jugador.

  Puede ser representado de las siguientes formas: barra, número o círculo.

  Además puede tomar dos tipos de valore: discreto (cuando el valor interno del tiempo disminuye o aumenta con valores enteros) o continuo (cuando el valor interno del tiempo disminuye o aumenta con valores decimales)

- **Vida**: Puntos de vida restantes del jugador.

  Sus representaciones varían entre: barra, número, círculo, iconos o sectores.

  Además puede tomar dos tipos de valores: Si se trata de representación continua, significa que el valor de la vida se comprende entre un rango finito de valores y puede variar entre ellos adoptando cualquier valor dentro de él, mientras que el valor discreto indica que la vida se representa como un conjunto de unidades que se expresan de forma individual en la interfaz

- **Escudos**: Puntos de salud extra complementarios a la vida.

  Sus representaciones varían entre: barra, número, círculo, iconos o sectores.

  Además puede tomar dos tipos de valores: discreto o continuo igual la vida

- **Puntuación**: Marcador de puntos conseguidos por el jugador.

  Puede ser representado de las siguientes formas: Imagen o texto junto a un número o un número.

- **Coleccionable**: Número de objetos o elementos recogidos a lo largo del nivel o partida.

  Puede ser representado de las siguientes formas: imagen o texto junto a un número o número

- **Progreso del personaje**: Representación del nivel o progreso del personaje del juego.

  Puede ser representado de las siguientes formas: sectores, círculo, barra o número.

- **Personaje**: Representación del personaje en la interfaz del juego.

  Su representación toma la forma de una imagen del personaje en cuestión.

- **Progreso del nivel**: Indicación de la parte ya superada del nivel o partida.

  Puede ser representado de las siguientes formas: barra o porcentaje(número).

- **Armas o habilidades**: Indicador de las armas o habilidades de las que dispone el jugador. Según el número de usos pueden ser:

  - Limitadas: aquellas que tienen un indicador de munición, por lo que después de agotar dicho indicador no podrán volver a ser usadas hasta que vuelva a disponer de munición
  - Infinitas: indica que la habilidad puede usarse de manera ilimitada.

  Puede ser representado de las siguientes formas: imagen o texto junto con un número.

2.3.2. First Person Shooter

Al igual que con el anterior género, se estudiaron un total de 15 videojuegos del género First Person Shooter, y una vez formadas las tablas se extrajeron las conclusiones reflejadas a continuación:

- **Munición**: Es la cantidad de balas que tiene el arma del jugador.

  Puede ser representado de las siguientes formas: barra (discreta o continua), un número que indica directamente el número de balas o iconos repetidos (uno por bala).

- **Tiempo de partida**: Indica el tiempo que le queda al jugador antes de que la partida termine.

  Puede ser representado de las siguientes formas: número, circunferencia o barra.

- **Minimapa**: Represetar el entorno que rodea al jugador en una vista desde arriba.

  Puede ser representado de las siguientes formas: cuadrado o círculo.

- **Brújula**: Indica los grados de rotación del jugador respecto a los puntos cardinales.

  Puede ser representado de las siguientes formas: línea o círculo(como una brújula común).

- **Vida**: Cantidad de puntos de salud del jugador.

  Puede ser representado de las siguientes formas: barra, circunferencia, número o porcentaje o icono.

- **Pociones**: Elementos que se usan para restaurar salud u otro tipo de atributos del jugador.

  Casi siempre su representación es con un icono y un número.

- **Escudos**: Indican la cantidad de vida extra o refuerzos complementarios a la vida del jugador.

  Puede ser representado de las siguientes formas: barra, circunferencia o número o porcentaje.

- **Feedback de bajas**: Registro de eliminaciones que se han producido en los momentos más recientes de la partida.

  Su representación suele ser un texto con un icono indicando la causa de la muerte.

- **Equipamiento**: indica la cantidad de equipo táctico del jugador (granadas, granada aturdidora...).

  Puede ser representado de las siguientes formas: icono o número.

- **Armas**: Informa de qué armas tiene a su disposición el jugador.

  Puede ser representado de las siguientes formas: icono o texto.

- **Recompensa**: Premio para el jugador por haber completado algún tipo de objetivo o misión.

  Puede ser representado de las siguientes formas: icono, barra o círculo.

- **Habilidades**: Efecto o poder especial que puede usar el jugador.

  Puede ser representado de las siguientes formas: icono o círculo.

2.3.3. Conclusiones

Tras el estudio de las tablas (Ver Anexo 10) con la información recogida de los distintos ejemplos de interfaces en videojuegos, hemos extraído una serie de conclusiones sobre los elementos que aparecen en estas.

Si se enumera para cada uno de los elementos distintos, el número de veces que aparece en cada uno de los 22 juegos de plataformas 2D estudiados (Tabla 2.1), se puede ver el siguiente nivel de prioridad:

$$Vida > Puntuación/Dinero > Escudos/Mana > Coleccionables > Arma/Habilidad > Tiempo > Progreso > Minimapa$$

Elemento	Número de juegos
Tiempo	7
Vida	14
Escudos/Maná	9
Puntuación/Dinero	14
Coleccionables	9
Minimapa	1
Personaje	4
Arma/Habilidad	8
Progreso del nivel	3

Tabla 2.1: Número de apariciones de cada elemento en los juegos de plataformas 2D

Al ver que casi ningún juego contiene un minimapa en su interfaz se decide eliminarla del caso de estudio.

Además de esto, se obtuvo unas conclusiones de cada elemento con la posición en la que solían estar, representación y algunos otros apuntes dependiendo de la importancia que tuvieran dentro de la interfaz. Se concluye que:

- **Tiempo** Su posición varía por la parte de arriba de la interfaz. Su representación no depende tanto de si los valores son continuos o discretos sino del resto de atributos. Pueden ser Iconos+números si en la representación de otros atributos (vida, puntuación, etc) también se hace uso de esta representación. En caso de no usarse, se utilizan números.

- **Vida** El posicionamiento suele ser arriba izquierda cuando su importancia es alta y abajo centro si es poco importante. Su representación varía entre: Si el valor es continuo, se representa como una barra o como un círculo. Si el valor es discreto, se representa como icono+número o corazones.

- **Escudos/maná** Se representan como barra (que puede ser circular). La posición suele ser arriba a la izquierda o arriba en el centro. Dependiendo de cuanto ocupe la vida y la posición de esta.

- **Puntuación/dinero** Su posición varía por la parte de arriba de la interfaz. Su representación suele ser Icono+Número.

- **Coleccionables** Su posición varía por la parte de arriba de la interfaz. Su representación suele ser Icono+Número.

- **Progreso Nivel Personaje** Su posición depende del resto de atributos con los que se pueda agrupar. Su representación suele ser un círculo.

- **Información del Personaje** Su posición suele estar arriba a la izquierda de la pantalla y se suele representar con un icono de dicho personaje.

- **Arma/Habilidad** Su posición suele estar en la parte de arriba en la parte contraria a la posición de la vida. Su representación es de icono. Si el número de usos es limitado mostrar Icono+Número.
- **Progreso de Nivel** Su posición suele estar en la parte inferior central cuando tiene importancia y en la parte contraria cuando su importancia es baja. Respecto a su representación suele ser en forma lineal, pero carecemos de muchos casos de estudio para sacar una clara conclusión sobre la representación.

Revisando también el número de apariciones de cada elemento para los 15 juegos First Person Shooter (Tabla 2.2) estudiados podemos definir la siguiente prioridad:

$$Munición > Vida > Equipo > Minimapa > Escudo > Objetivo > Arma > Brújula$$

$$> Premio > Habilidades > Pociones > FeedBack > Tiempo$$

Elemento	Número de juegos
Munición	15
Minimapa	9
Tiempo	2
Brújula	5
Vida	12
Pociones	3
Escudo	7
FeedBack muertes	3
Equipo	12
Arma	5
Premio	4
Habilidades	3
Objetivo	7

Tabla 2.2: Número de apariciones de cada elemento en los juegos FPS

De nuevo, extrajimos conclusiones de cada elemento respecto la posición, representación y la importancia que tuvieran dentro de la interfaz sacando las siguientes conclusiones para cada uno de los elementos:

- **Munición** En casi todos los casos se sitúa en la parte inferior de la pantalla, mayormente a la derecha. Siempre con valores discretos y representado en su mayoría con números, aunque a veces se presenta con iconos de balas restantes.
- **Minimapa** Su posicionamiento varía entre la parte superior de la pantalla y la inferior, pero aparece en la parte izquierda en la gran mayoría de casos. Su representación es o bien dentro de un círculo o de un cuadrado.
- **Tiempo** Su posición es siempre en la parte superior de la pantalla. La representación es con un número
- **Brújula** Su posicionamiento es generalmente en la zona superior del medio de la pantalla, excepto algunos casos en los que aparece acompañando al minimapa. Casi siempre representada mediante una línea.

- **Vida** Casi siempre aparece en la parte inferior izquierda, aunque también puede aparecer en la parte inferior derecha o incluso en la superior central. Cuando el valor es continuo se representa en forma de barra, y cuando es discreto en forma de Barra+número.

- **Pociones** Se posiciona abajo a la izquierda (normalmente cerca de la vida). Siempre representadas con un icono acompañado de un número.

- **Escudos** Siempre en la parte izquierda y tanto en la parte superior como inferior (normalmente acompaña a la vida). Representados mediante una barra y a veces acompañado de un número.

- **Equipamiento** Siempre en la zona inferior, generalmente en la parte derecha. Representados como un icono (alineados tanto vertical como horizontalmente si hay más de un objeto en el equipamiento) y a veces lo acompaña un número. En algunos juegos aparece en forma de texto. Su representación es de icono. Si el número de usos es limitado mostrar Icono+Número.

- **Armas** La posición suele ser en la parte inferior derecha de la pantalla. Su representación es con un icono.

- **Premio-Racha de puntos** Generalmente está situado en la parte de arriba, aunque también aparece en la parte inferior en algún juego. En cuanto a la representación, no se ha podido concluir qué tipo es el predominante ya que hay bastante variedad en los casos registrados.

- **Habilidades** Siempre en la zona inferior, y varía en la derecha e izquierda. Se representa tanto con forma de icono o dentro de un círculo.

- **Objetivos** Suele aparecer en espacios del HUD donde no haya más grupos de elementos, y siempre en forma de texto (a veces acompañado de un icono).

2.4. Razonamiento Basado en Casos

El problema que se aborda en este proyecto, la generación de una interfaz, es característico dado que no existe ningún algoritmo o procedimiento exacto para generar interfaces y además tiene una gran importancia la experiencia previa a la hora de diseñarlas. Por eso, un enfoque adecuado para abordarlo es el Razonamiento basado en casos.

El Razonamiento Basado en Casos (Case-Based Reasoning) (Richter y Weber (2013); Dwi H. Widyantoro (2014)) es una forma de resolución de problemas basada en ejemplos anteriores y conocidos. El caso (*case*) referencia a una experiencia o a un problema ya resuelto y se pueden exponer de diferentes maneras. Los casos se componen de dos partes: el enunciado del problema y su solución. Una base de casos (*case-base*) será, por tanto una colección de estos.

Por tanto, un sistema CBR es, en esencia, una técnica por la cual, a partir de un conocimiento previo (base de casos de la que se parte) y dado un problema, podemos buscar la solución más similar al mismo y adaptarla en caso necesario. Una vez se encuentre la solución, podrá añadirse junto con su descripción a la base de casos con el fin de preservarla para que en futuras consultas se tenga en cuenta.

2.4.1. Etapas del ciclo CBR

El CBR consta de las siguientes etapas (Figura 2.8). Podemos distinguir:

- Primero se comienza con una descripción parcial del problema de entrada, dado un usuario.

- **Recuperación** Se extraen de la base de casos los casos más similares a la consulta realizada al sistema CBR. Para ello, se hará uso de funciones de similitud, las cuales detectarán qué casos son más similares al de la consulta.

- **Reutilización** Una vez se han obtenido los casos más similares, existen dos formas de volver a utilizar dichos casos:

  - Copiar directamente su solución en caso de que su similitud sea igual o se perciba dentro de un rango impuesto por el sistema.
  - Adaptarlos en caso contrario. Para un uso sistemático de la adaptación se necesita un formalismo riguroso. Para este propósito, definimos los pasos de adaptación por reglas como acciones en general. Primero deberán de pasar una precondición para ver si pueden ser adaptados o no. Tras esto, se realizará una acción: añadir o crear algo, eliminar algo o modificar algo: Este "algo" puede ser el valor de una variable, parte de la descripción de un objeto (subclase), etc.

    Para identificar cuándo un caso devuelto debe ser adaptado o no, se pueden marcar unos umbrales en el valor de similitud devuelto para determinar si es necesario pasar por una fase de adaptación del caso.

- **Revisión** La evaluación de la solución se realiza aplicando la propia solución en el mundo real, es decir, probando qué sucedería si aplicáramos la solución obtenida en el contexto que se nos había producido el problema. Es evidente que la solución no tiene por qué probarse directamente sobre el mundo real, sino que este mundo puede simularse y recoger los datos que se obtienen de esta simulación. Una vez hecha esta evaluación hay que reparar los errores que pueda tener la solución.

- **Recuerdo** Una vez se tiene una solución revisada que satisface el problema, se debe tomar la decisión de si añadirla o no a la base de casos existente con el fin de que nuestro sistema aprenda del nuevo caso. Este proceso no siempre es necesario dado que pueden existir casos tan similares que no vale la pena guardarlos en memoria, o incluso que la solución encontrada no ha solucionado el problema, por lo que añadirlo a la base de casos no es recomendado en ciertas situaciones.

2.4.2. Descripción de un caso

En sistemas CBR podemos definir un caso de diferentes formas dependiendo de la complejidad del mismo, o de la cantidad de información referida al caso que quiera almacenarse. Así pues, un caso podría expresarse simplemente como una tupla <problema, solución> en sistemas poco complejos, mientras que en otros podría consistir en un conjunto de elementos como por ejemplo <problema, solución, justificación>. También podríamos definirlos como valores asignados a los atributos que describen los casos. En la capa de la implementación esto puede ser un registro (si se usan bases de datos) o un objeto (si se usa programación orientada a objetos).

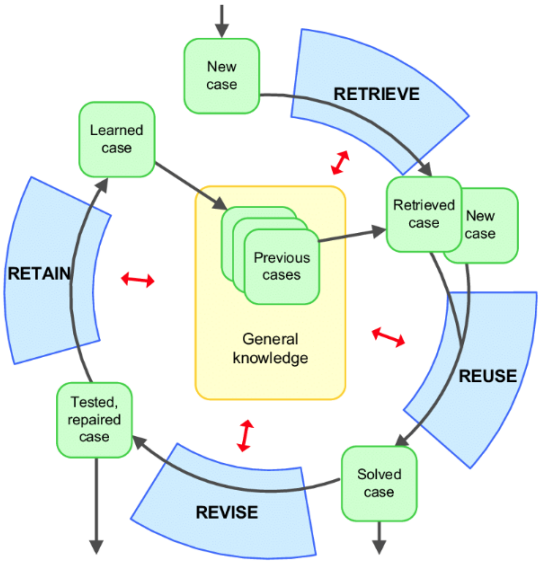

Figura 2.8: Ciclo Case Base Reasoning

La definición de problemas en un sistema CBR es crucial, ya que el razonamiento basado en casos tiene como objetivo dar una solución a un problema dado. Para poder resolver el problema en cuestión, se debe de conocer el contexto del mismo para que su descripción sea precisa. Por ejemplo, imaginemos que queremos dar una solución a la siguiente pregunta: ¿Cuál es el precio de este cuadro? La respuesta podría ser o bien *muy caro*, o *200 euros*. La diferencia entre estas dos respuestas es el contexto en el que se formula la pregunta, el cual debe de reflejarse en la formulación del problema.

En cuanto a las soluciones, pueden adoptar diferentes representaciones dependiendo de la complejidad del problema: desde soluciones que consisten en una frase explicativa, hasta soluciones complejas en las que se incluyen imágenes, comentarios, consejos, o incluso estrategias a seguir con las que se llegaron a la solución.

2.5. Conclusión

En este capítulo, hemos revisado varios conceptos teóricos que se aplicarán a lo largo del proyecto. En primer lugar, el conocimiento sobre las interfaces de los videojuegos y las conclusiones que se utilizarán más adelante a la hora de construir una interfaz lo mejor posible. A esto también ayudará el conocimiento de las Leyes de la Gestalt que también han sido revisadas en este capítulo. También saber que es un sistema CBR y como funciona nos será de gran utilidad dado que como ya se ha descrito anteriormente, es un enfoque válido que usaremos para abordar el problema de la generación la interfaz. En concreto, ayudarán a que la interacción del usuario sea lo más cómoda y fluida posible. Con todos estos conocimientos previos, podemos abordar el diseño de la herramienta.

Capítulo 3

Diseño de la Herramienta

En este capítulo se especificará cómo hemos planteado el diseño de la herramienta que ayuda en el diseño de interfaces de usuarios adecuadas para cada género. La herramienta consta de diferentes fases con distintas tareas: en la primera de ellas, el diseñador proporcionará información sobre qué elementos y sus atributos componen la interfaz que desea generar; la segunda de ellas es pasar estos datos recogidos a un razonador con el fin de obtener una definición de una interfaz satisfactoria.

A continuación se ven las especificaciones del diseño de los elementos que componen el razonador que va a ser usado para implementar nuestra herramienta. Como nuestro razonador es un sistema CBR, revisaremos desde la descripción de un caso y su solución, hasta las dos partes más importantes del mismo: las funciones de similitud y la fase de adaptación. Finalmente se explicará cómo la herramienta conservará los casos que considere necesarios para tenerlos en cuenta para futuras iteraciones.

3.1. Funcionalidad de la herramienta

Nuestra aplicación ha sido diseñada para conseguir una herramienta generalista, lo cual significa que dado una entrada a nuestro sistema (la cual consiste en la descripción cualitativa del juego en cuestión) devuelva una definición de la interfaz en un formato que pueda ser interpretado por cualquier otra herramienta. Es decir, podemos definir diferentes casos de uso tanto para la descripción cualitativa (entrada) mientras que esta respete los elementos y atributos que se necesitan para el razonador. Además también podemos definir diferentes casos para la herramienta que convierta la definición de la interfaz (la solución de nuestra herramienta) en una imagen visible para el usuario. La definición de la interfaz deberá adaptarse lo mejor posible a las características que haya proporcionado el usuario y mantener la coherencia con esta información. Además, deberá respetar y tener en cuenta en la mayor medida posible las Leyes de la Gestalt y otros principios de diseño que aporten información que haga óptima la interfaz.

En nuestra herramienta, la consulta que se usa para el razonador es una interpretación de la información proporcionada por el usuario que contiene la descripción de los aspectos más importantes de su juego que pretenda que se vean representados en la interfaz, por ejemplo, la existencia de un atributo de vida en el personaje principal o la existencia de varias habilidades disponibles. Por otro lado, la solución a dicha consulta realizada será la definición de la interfaz.

Durante el proceso de la herramienta será necesario tener en cuenta dos pasos de gran

importancia:

- Una vez el usuario nos ha proporcionado la información necesaria, la aplicación se encargará de enviar los datos recogidos al razonador para que obtenga una solución acorde a la petición del diseñador.
- Segundo, el usuario recibirá como respuesta una definición de la interfaz solución con toda la información detallada. Posteriormente, esta definición podrá ser interpretada por una herramienta que muestre visualmente la interfaz utilizándola y finalizando así el ciclo total de nuestra herramienta.

3.2. Diseño del razonador

Como hemos mencionado anteriormente, una parte de nuestra herramienta se encarga de obtener soluciones dadas unas peticiones mediante un razonador. Hemos decidido usar un sistema CBR como razonador dado que la generación de interfaces de usuario se basa en experiencias previas. Además de esto, no existe ningún algoritmo o procedimiento exacto para generar la mejor interfaz de usuario posible dado una descripción cualitativa, pudiéndonos poner en la piel de un experto y siendo capaces de usar casos que han funcionado y adaptarlos con los valores indicados por el usuario.

Ya hemos visto anteriormente en la sección 2.4 que un sistema Case Based Reasoning es una forma de resolución de problemas. La descripción cualitativa proporcionada por el diseñador como entrada, la utilizaremos para realizar una *query* o una petición sobre nuestro sistema, realizando así una búsqueda del caso más parecido.

Cuando el sistema obtenga el caso más parecido lo adaptará al caso de la consulta devolviendo así una nueva definición de interfaz que el usuario pueda visualizar mediante otra herramienta.

Cabe mencionar que ninguna de la información que el usuario haya elegido es modificada por el sistema CBR, ya que la inclusión de algún elemento extra o la eliminación de alguno de los solicitados en el resultado final, haría que la ayuda al usuario que pretende proporcionar nuestra herramienta sea mucho menor incluso dificultando el diseño.

Cuando se realiza un diseño de un sistema CBR, debemos de tomar una serie de decisiones teniendo en cuenta las características del problema a resolver y estas serán las que se describen en los siguientes apartados.

3.2.1. Descripción de un caso

Comencemos entonces describiendo cada caso de nuestro sistema de CBR, el cual estará compuesto tanto de una descripción como de una solución de este.

En un sistema CBR la descripción de un caso es la explicación de la situación o el problema en cuestión. Esta descripción consta de objetivos y restricciones para la consecución de los objetivos. Además de esto proporciona las características de la situación en la que se encuentra el sistema CBR. La descripción del caso de nuestro sistema contiene un campo para cada uno de los elementos posibles en la interfaz. Cada uno de estos elementos, contendrá una serie de atributos (Figura 3.1). Todos compartirán un atributo común que es la importancia del elemento. Además de la importancia, a continuación se detallan el resto de atributos para cada uno de los elementos de la interfaz:

- **Vida**: Contiene un atributo para saber si su representación interna en el juego es discreta o continua.

- **Escudos**: Contiene un atributo para saber si su representación interna en el juego puede ser discreta o continua.
- **Tiempo**: Contiene un atributo para saber si su uso dentro del juego es de cronómetro o de cuenta atrás.
- **Habilidades**: Contiene un atributo que guarda el número de habilidades que existen en el juego (aquellas que se quieren mostrar en la interfaz) y otro atributo para saber si estas son de uso limitado o infinito.
- **Armas**: Contiene un atributo que guarda el número de armas que existen en el juego (aquellas que se quieren mostrar en la interfaz) y otro atributo para saber si estas son de uso limitado o infinito.
- **Progreso de nivel**: Contiene un atributo para saber si su representación interna en el juego es discreta o continua y un segundo atributo para saber si se quiere mostrar el progreso superado o el restante.
- **Progreso del jugador**: Contiene un atributo para saber si su representación interna en el juego es discreta o continua.

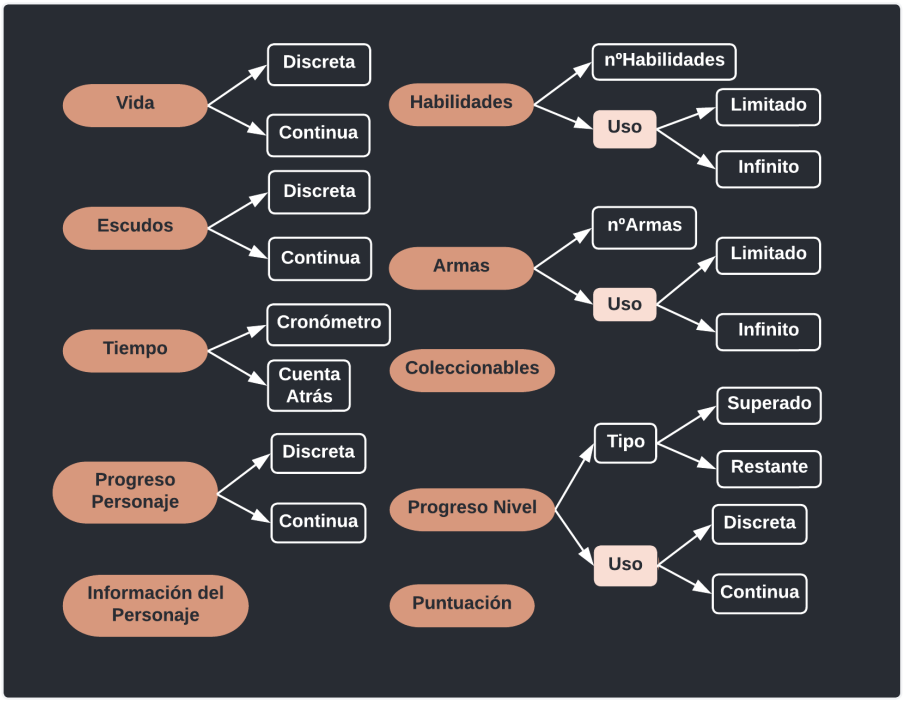

Figura 3.1: Elementos de la Descripción de un caso

3.2.2. Solución de un caso

El caso consta de una solución que tendrá para cada uno de los elementos posibles de la interfaz los valores del tamaño y posición en la pantalla, un identificador del elemento y una cadena de texto con el nombre de la imagen que representará al elemento en cuestión (Figura 3.3). Por un lado, el identificador del elemento servirá para distinguir unos de otros (por ejemplo vida y escudos). Por otro lado, la posición y el tamaño se guardarán de forma

simplificada. Para la posición, se guardará la posición horizontal, que puede ser izquierda centro o derecha, y la posición vertical que puede ser arriba medio o abajo (Figura 3.2). El tamaño será otro atributo de cada elemento y podrá variar entre: muy pequeño, pequeño, mediano, grande y muy grande.

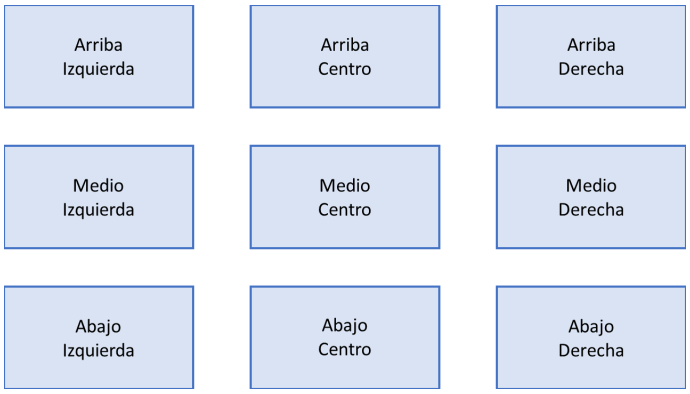

Figura 3.2: Posibles posiciones para un elemento en la solución de un caso

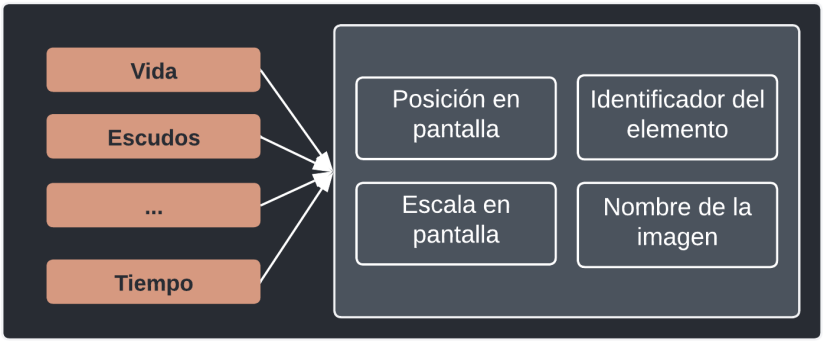

Figura 3.3: Elementos de la Solución de un caso

3.2.3. Consulta del sistema CBR

La consulta o query contendrá una descripción pero no dispone de una solución. Esta descripción consta de la misma estructura que tienen los casos.

3.2.4. Funciones de Similitud

El siguiente paso para diseñar un sistema de CBR es definir como se calculará la similitud entre dos casos, o en su defecto, entre una petición y un caso. Esto implica a los atributos que componen la descripción previamente nombrada. La similitud entre dos casos se define mediante una similitud global (descrita en el apartado 3.2.4.1) la cual es un conjunto de similitudes parciales (descrita en el apartado 3.2.4.2).

3.2.4.1. Similitud Global

Para la realización de la similitud global, se hará la combinación de las similitudes parciales de los elementos usando una media ponderada. Usaremos c para referirnos al atributo correspondiente del caso (case) y q para el de la consulta (query).

$$sim(c,q) = \alpha \cdot simVida(c,q) + \beta \cdot simEscudos(c,q) + \gamma \cdot simPuntuacion(c,q) +$$

$$\delta \cdot simTiempo(c,q) + \gamma \cdot simHabilidades(c,q) + \delta \cdot simArmas(c,q) +$$

$$\epsilon \cdot simProgresoNivel(c,q) + \zeta \cdot simProgresoPersonaje(c,q) +$$

$$\eta \cdot simInfoJugador(c,q) \ + \omega \cdot simColeccionable(c,q)$$

Para calcular la similitud global se hace uso de similitudes parciales(simVida, simTiempo,...) multiplicadas por un peso diferente para cada una de las similitudes(alpha, beta,...) El resultado de la similitud global dará un número comprendido entre 0 y 1 indicando la similitud entre la petición y cada uno de los casos del sistema de manera que se pueda encontrar el caso más similar a la petición. En nuestro caso, solo recuperamos el caso más similar al de la petición.

3.2.4.2. Similitud Parcial

Llamamos similitud parcial a aquella similitud que calcula cuán símil es un elemento de la consulta con el mismo elemento del caso con el que se está comprobando. Estas son similitudes calculadas para cada uno de los elementos de la interfaz y se calculan sobre los atributos de cada elemento. Vamos a utilizar tres medidas de similitud auxiliares:

- Medida de similitud por intervalo: se usará la inversa de la resta normalizada entre el valor de la consulta y el valor del caso

  $$sim(c,q) = 1 - (|q-c|/maxDistance),$$

  $$maxDistance = max - min,$$

  donde min y max son los valores mínimo y máximo del intervalo, respectivamente

- Medida de similitud por igualdad: se usará la igualdad entre el valor de la consulta y el valor del caso con el que se compara.

  $$sim(c,q) = \begin{cases} 1 & si \quad c = q \\ 0 & si \quad c \neq q \end{cases}$$

- Medida de similitud por intervalo con máximo variable: la inversa de la resta de los valores normalizados. Esta normalización se realiza con un máximo arbitrario.

  $$sim(c,q) = 1 - (|q-c|)/maximoArbitrario,$$

  El máximo arbitrario se obtiene cogiendo el máximo entre los dos casos (consulta y caso a ver).

La similitud parcial para la importancia se calcula utilizando la similitud por intervalo. Los valores enteros como el número de armas o habilidades utilizan la similitud por intervalo con máximo variable. El resto de atributos utiliza la similitud por igualdad. Para combinar las similitudes parciales de cada uno de los atributos de cada elemento utilizamos una media ponderada:

- Vida, Progreso del jugador, Escudos:

$$simVida(c,q) = \alpha \cdot sim_{intmax}(c[importancia], q[importancia]) +$$

$$\beta \cdot sim_{intmax}(c[tipo], q[tipo])$$

$$simEscudos(c,q) = \alpha \cdot sim_{intmax}(c[importancia], q[importancia]) +$$

$$\beta \cdot sim_{intmax}(c[tipo], q[tipo])$$

$$simProgresoPersonaje(c,q) = \alpha \cdot sim_{intmax}(c[importancia], q[importancia]) +$$

$$\beta \cdot sim_{intmax}(c[tipo], q[tipo])$$

- Información del jugador, Coleccionables, Puntuación:

$$simInfoJugador(c,q) = sim_{intmax}(c[importancia], q[importancia])$$

$$simColeccionable(c,q) = sim_{intmax}(c[importancia], q[importancia])$$

$$simPuntuacion(c,q) = sim_{intmax}(c[importancia], q[importancia])$$

- Habilidades, armas:

$$simHabilidades(c,q) = \alpha \cdot sim_{intmax}(c[importancia], q[importancia]) +$$

$$\beta \cdot sim_{intmax}(c[uso], q[uso]) + \gamma \cdot sim_{intmax}(c[numeroHabilidades], q[numeroHabilidades])(c,q)$$

$$simArmas(c,q) = \alpha \cdot sim_{intmax}(c[importancia], q[importancia]) +$$

$$\beta \cdot sim_{intmax}(c[uso], q[uso]) + \gamma \cdot sim_{intmax}(c[numeroArmas], q[numeroArmas])(c,q)$$

- Tiempo:

$$simTiempo(c,q) = \alpha \cdot sim_{intmax}(c[importancia], q[importancia]) + \beta \cdot sim_{intmax}(c[uso], q[uso])$$

- Progreso del nivel:

$$simProgresoNivel(c,q) = \alpha \cdot sim_{intmax}(c[importancia], q[importancia]) +$$

$$\beta \cdot sim_{intmax}(c[rango], q[rango]) + \gamma \cdot sim_{intmax}(c[tipo], q[tipo])$$

Podemos ver un ejemplo del su funcionamiento (Figura 3.4).

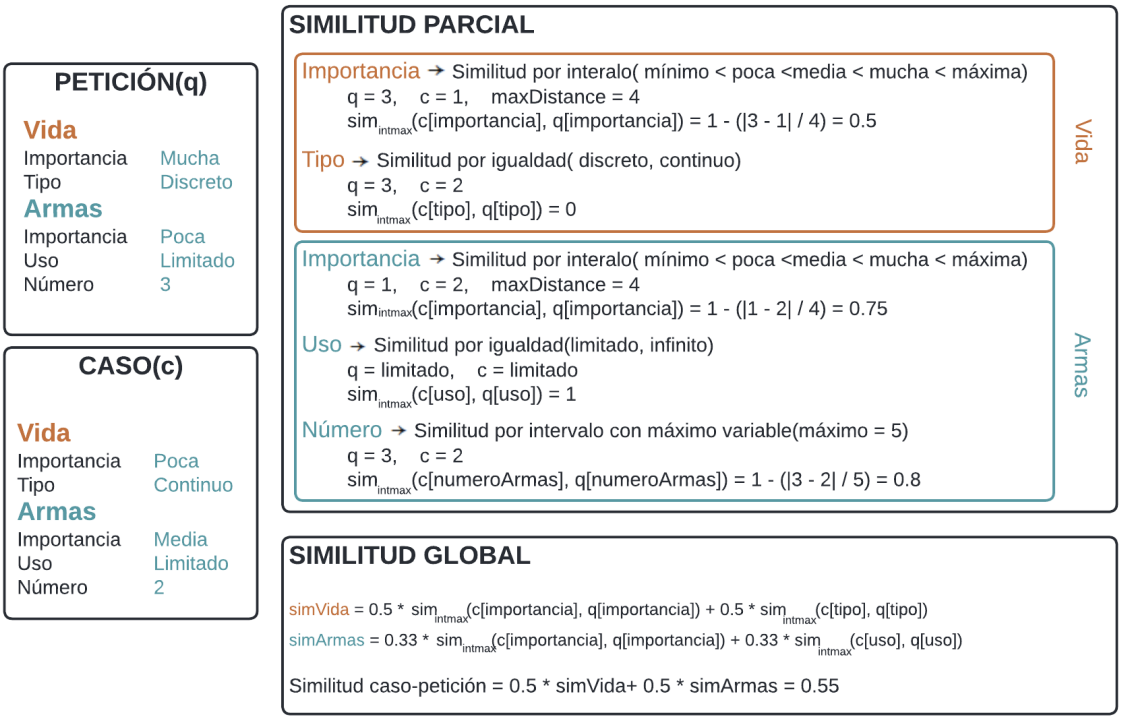

Figura 3.4: Ejemplo cálculo de similitud

3.2.5. Adaptación

Una vez se ha encontrado el caso más similar, este puede no tener los mismos elementos o ciertas características dentro de los propios elementos que tenían los de la consulta. Por este motivo, se necesita una fase de adaptación.

En una primera parte, se realizará una adaptación individual, es decir, por cada uno de los elementos. En esta fase se comprobará que la descripción del elemento de la consulta y del elemento que ha devuelto la solución sean iguales. En caso de no serlo, se deberá de modificar la solución añadiendo esta nueva información. Por ejemplo, si el usuario quiere el elemento vida con un valor continuo, pero nuestra solución ha generado uno con un valor discreto habrá que modificar esta información de tal forma que acabe tomando el valor continuo. Dentro de esta primera fase de adaptación se modifican:

- Si la solución no contiene el elemento de la consulta, este se añade a la solución con valores por defecto para que la fase de adaptación pueda seguir modificándolo.
- Si un atributo de un elemento puede tener más de un valor (siguiendo el ejemplo de la vida, sería continuo o discreto) y este valor no es el mismo en la consulta que en la solución se asigna en la solución el valor de la consulta.
- Si el valor de la escala para el elemento en la solución no corresponde con la importancia del elemento de la consulta se modificará la escala.

Una vez que cada uno de los elementos mantienen una cierta lógica con los de la consulta, queda hacer la segunda fase de la adaptación. En esta fase se tomarán en cuenta los principios de Semejanza y Proximidad, explicados en el apartado 2.2. Para ello primero se ha diseñado una tabla de similitud de adaptación (Figura 3.5) que contendrá los valores de similitud entre cada uno de los elementos existentes. Estas similitudes se utilizarán de la

siguiente manera: si dos elementos son muy similares y se encuentran en posiciones distintas en pantalla, se deberán de agrupar; por el contrario, si dos elementos son muy desiguales y se encuentran en posiciones iguales en pantalla, se deberán de separar el uno del otro. Los valores de la tabla provienen de las relaciones entre utilidad y comportamiento que tienen los diferentes elementos en los juegos del género Plataformas 2D.

	VIDA	PUNTOS	ESCUDOS	COLECCIONABLES	ARMAS	HABILIDADES	TIEMPO	PROGRESO NIVEL	INFO PERSONAJE	PROGRESO PERSONAJE
VIDA	1	0.3	0.8	0.3	0	0	0	0	0.75	0.75
PUNTOS	0.3	1	0.3	0.8	0.3	0.3	0.75	0.6	0.2	0.2
ESCUDOS	0.8	0.3	1	0.3	0	0	0	0	0	0
COLECCIONABLES	0.3	0.8	0.3	1	0	0	0.6	0.3	0	0
ARMAS	0	0.3	0	0	1	0.8	0	0	0	0
HABILIDADES	0	0.3	0	0	0.8	1	0	0	0	0
TIEMPO	0	0.75	0	0.6	0	0	1	0.8	0.4	0.4
PROGRESO NIVEL	0	0.6	0	0.3	0	0	0.8	1	0.65	0.65
INFO PERSONAJE	0.75	0.2	0	0	0	0	0.4	0.65	1	0.9
PROGRESO PERSONAJE	0.75	0.2	0	0	0	0	0.4	0.65	0.9	1

Figura 3.5: Tabla de Similitud para Adaptación

3.2.6. Recuerdo

Como política de recuerdo guardaremos todos los casos que se hayan generado como solución a una consulta a no ser que ya haya un caso muy similar ya guardado y que se hayan hecho pocos cambios en la fase de adaptación. Definimos como caso similar aquel que su función de similitud devuelva >0.8.

En sistemas CBR, es muy importante que la base de casos sea adecuada. Esto implica que no sea demasiado grande, ya que podría alargar mucho el tiempo de procesamiento, además de el espacio que ocupa en memoria. Tampoco debe ser muy pequeña, ya que eso supondría que para la mayoría de peticiones, no se encontraría un caso lo suficientemente similar que permitiese que la solución final sea interesante. Otra característica importante es que los casos guardados sean representativos, es decir, con la mayor diversidad posible entre ellos para poder abarcar una mayor variedad de peticiones con mejores resultados.

Con esta política de recuerdo, pretendemos mantener una base de casos que pueda ir creciendo y aprendiendo de las soluciones que genere para las nuevas peticiones pero que al mismo tiempo preserve la diversidad y no acumule una cantidad demasiado alta de casos.

3.2.7. Salida del sistema CBR

Una vez se ha encontrado el caso más parecido y se ha adaptado, se tiene que redistribuir cada uno de los elementos con esta nueva información obtenida, debido a que, por ahora, no tenemos ninguna definición de interfaz que el diseñador pueda utilizar.

La forma en la que hemos decidido distribuir cada uno de los elementos que pueden conformar una interfaz es la siguiente: por cada elemento se asocia su posición (la cual puede haber sido modificada o no en la fase de adaptación) a un **objeto combinado** con la misma posición. Es decir si tenemos el elemento vida con una posición ARRIBA-IZQUIERDA irá en el objeto combinado ARRIBA-IZQUIERDA. De la misma forma ocurre

con el resto de elementos y sus posiciones. Concretamente, se genera un objeto combinado por cada una de las zonas posibles para un elemento en pantalla (Figura 3.2). Podemos definir un objeto combinado como un objeto que puede contener más de un elemento dentro suyo. Los elementos que contiene el objeto combinado suelen estar relacionados de alguna forma debido a que la fase de adaptación habrá modificado su posición de acuerdo a la similitud entre los elementos. Además de la lista de objetos que forman ese combinado, estos tienen información sobre su posición.

Por ejemplo, podríamos tener el objeto combinado ARRIBA-IZQUIERDA con dos elementos: vida y escudos. Este objeto sería único en pantalla y solo contendría dos elementos. Si existiera otro elemento en la misma posición (ARRIBA-IZQUIERDA) entonces debería de estar contenido en el objeto combinado. Por otra parte, dado que hay más de un objeto combinado en pantalla, un elemento no puede estar en dos objetos combinados a la vez, debido a que eso supondría que el elemento en cuestión estaría repetido en pantalla en dos posiciones distintas (Figura 3.6).

Los objetos combinados nos ayudan a la hora de posicionar los elementos dentro de la pantalla incluyéndolos como un solo objeto unificado, como una caja negra que contiene información y de la cual nos podemos abstraer.

Una vez se hayan realizado todos los objetos combinados necesarios para formar nuestra interfaz, la salida del CBR devolverá la definición de interfaz que contiene todos los objetos combinados con sus respectivos elementos dentro. Cabe mencionar que los objetos combinados pueden estar vacíos o constar de un solo elemento dentro.

Además de esto, la salida devolverá también: el id del caso seleccionado para ser la solución con el fin de realizar una comprobación del caso devuelto de la base de casos; un número decimal con el valor de la similitud que ha haya devuelto la función de similitud global con el fin de saber con cuanta exactitud se parece el caso.

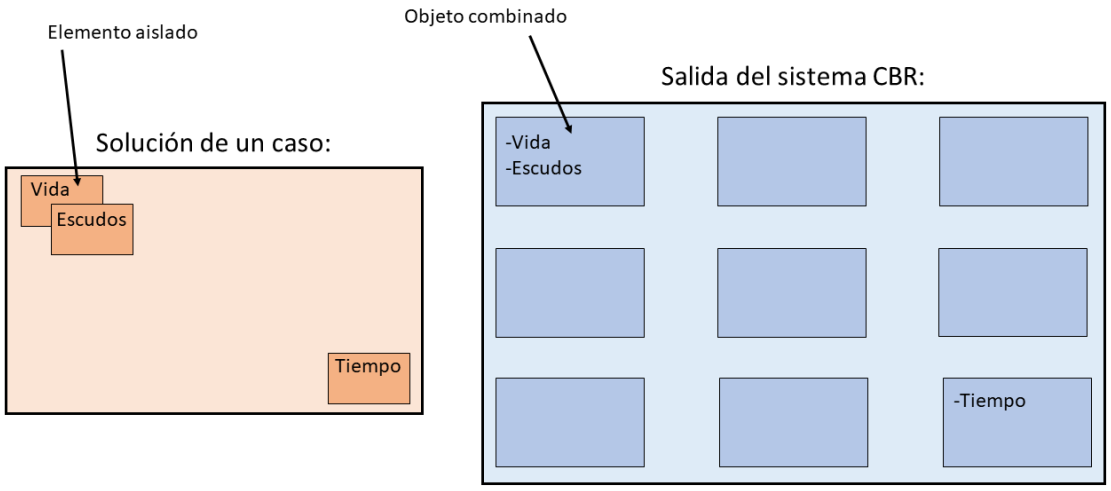

Figura 3.6: Ejemplo de generación de la salida del sistema CBR

3.2.8. Creación de casos para el sistema CBR

Como bien se ha explicado con anterioridad, un sistema CBR parte de una base de casos. En nuestro sistema, partiremos de casos creados a mano con la información que se recopiló en el apartado 2.3.1. Hemos realizado un total de 49 casos, de los cuales 13 han sido sin fijarnos en ninguna interfaz, y los 36 restantes provienen de juegos. Para los casos de los juegos nos hemos apoyado en la página web de Game UI Database [1] que proporciona imágenes de las distintas interfaces que juegos. Estos casos guardarán la información de entrada que nosotros veamos conveniente (expuesta en las tablas de investigación que realizamos para cada uno de los juegos) y la salida de la interfaz del juego en cuestión, siendo un intento de copia de la interfaz del juego adaptado las variables de nuestro sistema.

Por otra parte, además de los casos de cada una de las interfaces que analizamos, tenemos los casos especiales. Algunos de ellos los cubre el sistema de CBR mediante la adaptación, sin embargo existen otros que no. Estos son, por ejemplo, aquellos casos en los que sólo exista un elemento o en los que existan todos.

[1] Game UI Database https://www.gameuidatabase.com/

Capítulo 4

Implementación

En este capítulo se verá como se han llevado a cabo las ideas de diseño implementándolas en una plataforma y con una organización y estructura concreta. Se verá esto para cada uno de los componentes que conforman la herramienta, pasando desde la petición en el servidor hasta el formato de la definición de interfaz hecha por el razonador.

4.1. Arquitectura de la herramienta

La aplicación que hemos realizado está implementada a modo de servicio web de manera que al iniciar el servidor, que muestra nuestra página web, se inicializa el sistema de CBR y se configura para que esté listo a la hora de recibir una petición con la descripción cualitativa. Una vez el usuario decida realizar una petición con la información que ha rellenado, esta información se recoge y se envía al servidor mediante una petición (Figura 4.1).

Una vez la petición llega al servidor, hemos de reconstruir la información con el fin de poder realizar una petición al sistema CBR. Este buscará el caso más similar, realizará las correspondientes modificaciones y devolverá una definición de la interfaz. Esta definición vendrá dada en formato JSON, que será pasada por el servidor enviándola de nuevo al formulario.

4.2. Servicio REST

Para la implementación del servicio REST hemos utilizado Spring Boot[1], un framework que nos ayuda a implementar un servidor donde pueda correr el sistema CBR. En nuestro caso, hemos desplegado un servidor para conectar con los servicios REST. REST es una interfaz que sirve para conectar diferentes sistemas basados en el protocolo HTTP y nos sirve para enviar y recibir datos en formatos muy específicos como JSON o XML. REST se apoya en los métodos de petición de HTTP siendo POST, GET, PUT, PATCH y DELETE.

Una vez inicializado el servidor, dentro de este ha de inicializarse el sistema de CBR. Para ello, utilizamos la propiedad `initMethod` ya definida por SpringBoot para inicializar nuestro sistema. Dado que nuestro sistema CBR es un singleton, se inicializa una única vez y puede ser accedido desde distintas clases.

En cuanto a las peticiones, nuestro servidor Rest (Figura 4.2) acepta peticiones de tipo

[1]SpringBoot: https://spring.io/guides/tutorials/rest/

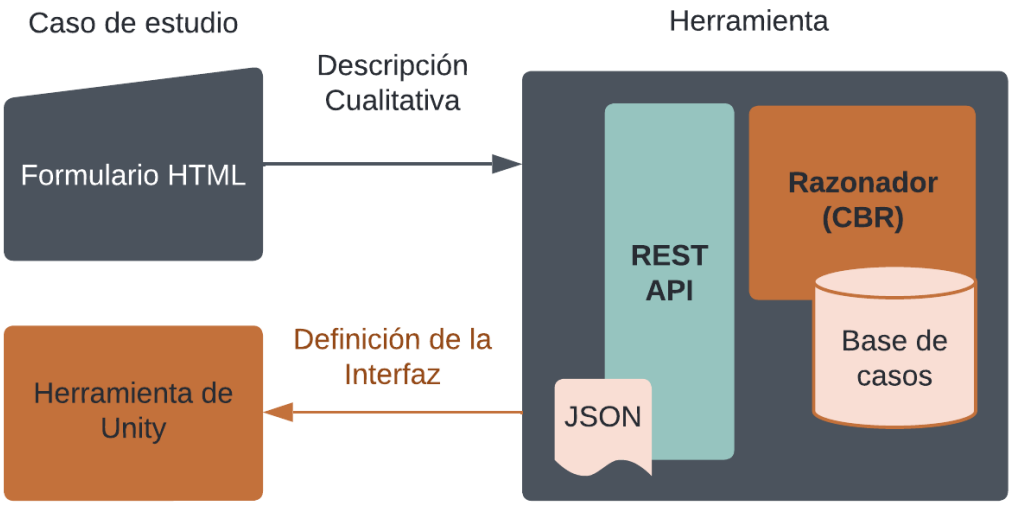

Figura 4.1: Esquema de la arquitectura de la aplicación

post a la dirección */platform* recibiendo como parámetro un objeto de tipo `PlatformQuery`. Esta clase contiene atributos que guardan cada uno de los elementos que se pueden seleccionar en el formulario. A su vez, cada atributo (elemento) contiene la información necesaria para si mismo, siendo la descrita en el apartado 3.2.1. Esta clase se usa como clase intermedia entre el formulario y la petición del sistema de CBR. Así pues, en la petición post también se realiza un proceso de mapeo de la clase `PlatformQuery` a la clase `CaseDescription` (la cual guarda la descripción de un caso) de la que hablaremos más adelante.

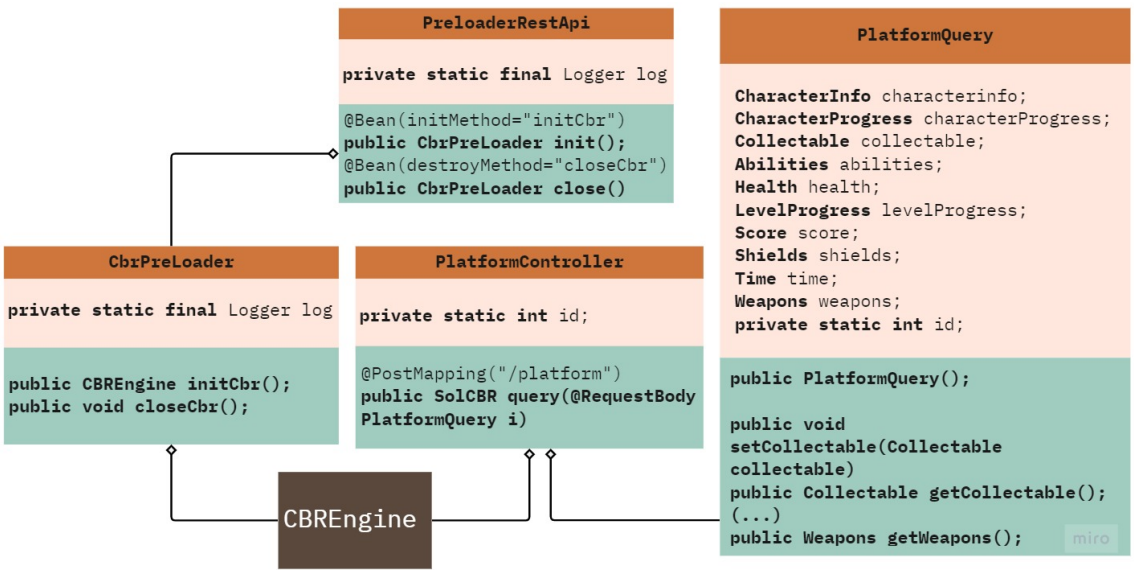

Figura 4.2: Diagrama UML simplificado de clases del servidor

4.3. Sistema CBR

El sistema CBR lo hemos implementado usando jCOLIBRI, un framework para el desarrollo de sistemas de Case-Based Reasoning. Hemos seleccionado este framework por toda la funcionalidad ya implementada que nos ofrece a la hora de crear un sistema CBR.

En nuestro sistema de CBR (Figura 4.3) partimos de la clase `CBREngine`, la cual implementa la clase `StandardCBRApplication` y es la responsable de configurar el sistema de CBR, realizar los ciclos (dada una petición) y devolver la salida entre otras cosas. Esta clase, además, es la encargada de realizar el ciclo completo de un sistema CBR (recuperación, reutilización, revisión y recuerdo). Además, también guarda la ruta donde se encuentra la base de casos.

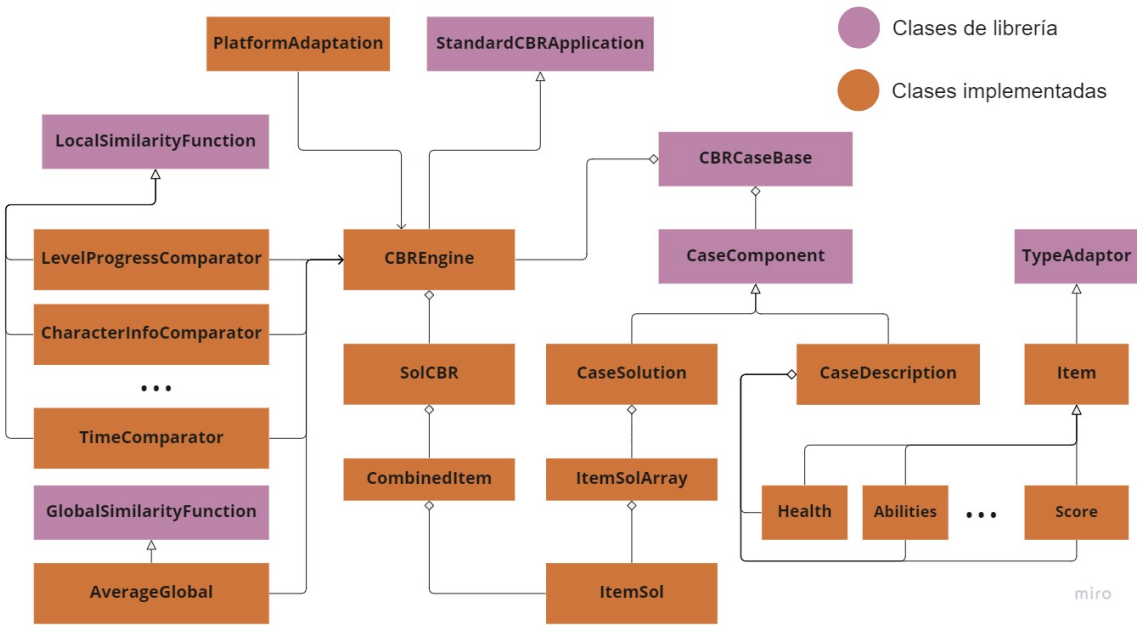

Figura 4.3: Diagrama de clases UML del sistema CBR

Todas estas clases forman la estructura general de nuestro sistema, aunque aún quedan por definir clases con funcionalidad más concreta: los casos, los cuales guardan la información referente a las descripciones y soluciones de nuestros casos.

4.3.1. Componentes de un Caso

Como bien hemos mencionado anteriormente, un CBR parte de una base de casos y para la creación de un caso se ha de necesitar de una descripción y de una solución. En jCOLIBRI los casos contienen tres `CaseComponent`: uno para la solución, otro para la descripción y otro para el resultado (este último no lo usaremos).

Hemos implementado la descripción en `CaseDescription` y la solución del caso en `CaseSolution`, ambas heredando de `CaseComponent` (Figura 4.4), con la información ya explicada en el punto 3.2.1 y 3.2.2 respectivamente. Por otra parte, cuando se lee de este archivo los casos se rellenan (sus `CaseComponent`) usando introspección.

- En `CaseDescription`, cada atributo de la clase hereda de `Item`, el cual guarda la importancia del elemento en si (siendo un tipo enumerado). La clase Item contiene

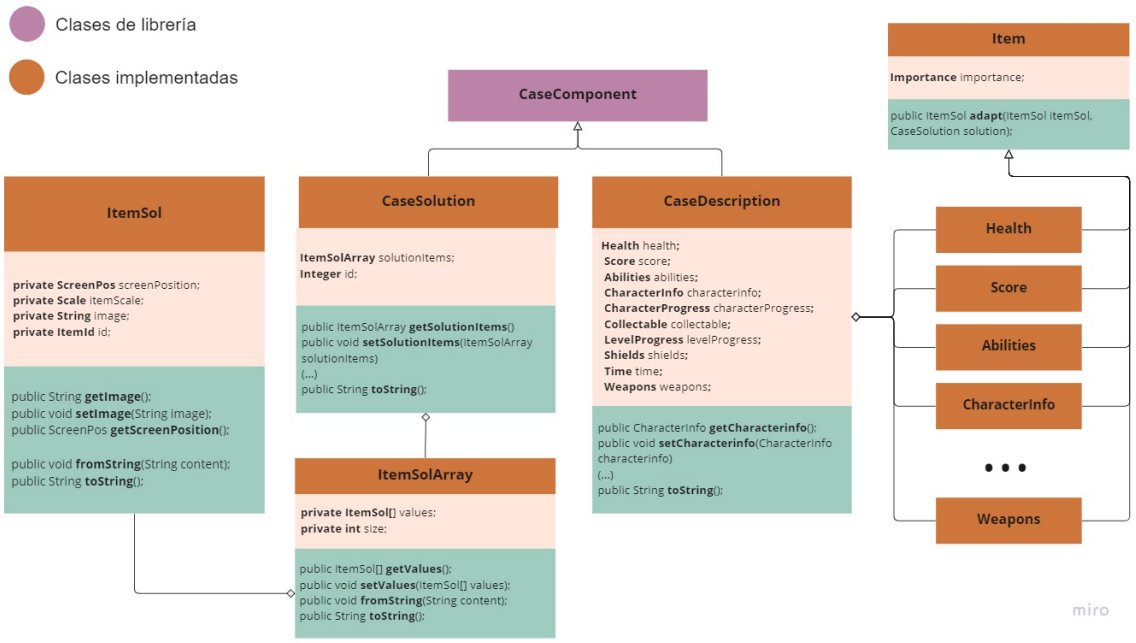

Figura 4.4: Diagrama Herencia CaseComponent

un método de adaptación que han de modificar todos los elementos del sistema CBR para seguir la primera fase de adaptación explicada en 3.2.5.

- En `CaseSolution`, se guarda un vector de `ItemSol`. `ItemSol` es una clase que contiene la posición y escala en la pantalla, el nombre de un fichero de imagen que lo representará y un identificador que identifica el tipo de elemento que es. Cabe destacar, que, con el fin de realizar un array y que este funcionara en el sistema de CBR, se tuvo que realizar una clase auxiliar `ItemSolArray` que implementa `TypeAdaptor` con los mismos métodos de serialización y deserialización.

4.3.2. Similitud y Adaptación

Por otra parte, y como se ha explicado en el apartado 3.2.4, hemos implementado las funciones de similitud parciales de cada uno de los elementos, utilizando una clase por cada uno. Estas clases heredan de `LocalSimilarityFunction` y reescriben el método `compute`. Este método es el encargado de buscar la similitud entre dos objetos. En este caso, cada objeto será un elemento y, por tanto, se buscará la similitud entre el elemento asociado a la query/consulta y el elemento asociado al caso actual de la base de casos (con el cual se está comparando). Por otra parte, para la función de similitud global (entre el caso de consulta y el de la base de casos) se utiliza la clase `AverageGlobal`. Esta clase calcula la similitud de un caso de forma global realizado una media ponderada.

Para la parte de adaptación seguiremos una implementación basada en el diseño que hemos realizado. Para la primera fase de adaptación hemos hecho un método `adapt` en la clase padre `Item` (de la cual heredan todos los atributos de `CaseDescription`) con el fin de que sea reescrito en cada uno de los elementos. Este método adapta los atributos de la solución a los de la consulta en caso de ser necesarios.

Además de esto, se ha implementado la clase `PlatformAdaptation` encargada de realizar toda la fase de adaptación del sistema CBR. Dicha fase se compone de dos partes: la primera adaptará la solución de acuerdo a la consulta realizada, llamando para ello al

método `adapt` de cada elemento y modificará las posiciones de los elementos de acuerdo con los conceptos explicados en 3.2.5. La segunda parte consiste en crear y devolver una instancia de la clase SolCBR dado el objeto de tipo `CaseSolution` ya adaptado en el paso anterior. La clase `SolCBR` guarda la información de los objetos combinados mencionados en el apartado 3.2.7.

4.3.3. Base de Casos

En este fichero vienen definidos los casos por descripción y solución separados por diferentes caracteres dependiendo de la serialización que se haya realizado. La lectura de los casos la realiza jCOLIBRI mediante un archivo de configuración, en el cual se ha definido los atributos que tienen tanto la descripción como la solución. De esta forma, y definiendo como se debe serializar y deserializar cada uno de los componentes que forman el caso, formado a su vez por la descripción y la solución, podemos cargar y guardar casos.

Para intentar afectar en la menor medida posible al rendimiento del sistema, la base de casos se carga a memoria durante el levantamiento del sistema CBR y mientras se mantenga la ejecución sigue cargada. Es solo durante el cierre del sistema, que se hace persistir los casos escribiéndolos de nuevo en el fichero.

4.4. Definición de la interfaz solución

Nuestra herramienta debe de devolver la descripción de una interfaz que se ajuste al juego descrito por el jugador. Para ello, dicha descripción consistirá en un archivo en formato JSON, el cual contendrá la información de cada conjunto de elementos, cuyas características fueron descritas en la sección 3.2.7.

El fichero contendrá una colección de nueve objetos. Cada objeto (*item*) representa una de las nueve posibles posiciones de pantalla y contiene, a su vez, una colección de objetos, donde cada uno guarda la información de un elemento. Estos objetos guardan información sobre: la posición de la pantalla a la que corresponde el objeto o elemento (`screenPosition`), la escala que tiene el elemento dentro de la pantalla (`itemScale`), el fichero de imagen que va a utilizar ese objeto/elemento (`image`) y finalmente un id asociado al elemento en sí (`id`). Además de esto, el objeto contenedor de los elementos contiene también un campo para la posición en pantalla (`screenPosition`). Trás la definición de todos los objetos combinados se encuentra el atributo `id` que guarda el número del caso asociado y `sim` que guarda el valor resultado de la función de similitud. (Figura 4.5)

Como puede verse en la imagen de ejemplo 4.5, el primer combinado descrito en el Json se corresponde con la posición superior izquierda de la pantalla (TOP LEFT), con un tamaño mediano (MEDIUM) y con dos elementos contenidos (HEALTH Y SHIELDS), seguido de otro conjunto que en este caso está vacío, conteniendo por tanto su posición en pantalla (TOP CENTER), su escala (MEDIUM) y una colección vacía de elementos.

Figura 4.5: Explicación del fichero de definición de interfaz

Capítulo 5

Caso de Estudio

En este capítulo veremos el caso de uso que hemos realizado para nuestra herramienta. Como ya se ha nombrado anteriormente, es generalista, es decir que tanto la entrada de datos (formulario) como la descripción de la interfaz solución que devuelve la herramienta no están ligadas a una herramienta o motor concretos. Por lo tanto, los apartados descritos en este capítulo se han utilizado para llevar a la práctica nuestro objetivo y poder visualizarla en funcionamiento en un caso real.

Para obtener la descripción cualitativa, se ha implementado un formulario en HTML, y para visualizar el resultado devuelto por nuestra herramienta (la definición de la interfaz), hemos decidido implementar una segunda herramienta en forma de plug-in para Unity[1] para usarlo como prueba de concepto y a la vez para poner en práctica nuestro proyecto sobre uno de los motores de videojuegos más extendidos en la industria.

En este capítulo, describiremos la prueba de concepto, siguiendo con la parte de su diseño y finalmente la implementación.

5.1. Recogida de datos

Para la parte de la recogida de datos del usuario (la definición de la interfaz), lo más cómodo es usar un formulario. La información que pretendemos recoger es primeramente que elementos de entre todos los posibles desea el usuario que aparezcan en su interfaz. Los elementos que puede elegir el usuario son aquellos que analizamos en la sección 2.3.3.

En cada uno de los elementos que puede elegir el usuario, encontraremos una importancia que le da a un elemento en concreto. Esta información se usará para determinar la posición y el tamaño del elemento dentro del espacio de la interfaz. Esta información vendrá representada por una escala de Likert, con el fin de que el usuario escoja un valor. Por otro lado, cada uno de los elementos podrán contener más detalles extra sobre su representación o funcionamiento que nos ayuda a garantizar una solución más acertada y similar al diseño que tendrá la interfaz una vez implementada en el juego.

A la hora de la implementación hemos utilizado un formulario HTML, con su correspondiente estilo definido en una hoja de estilo en cascada (CSS). Para dar formato a la página web, hemos hecho uso de la librería Boostrap[2].

Al final del cuestionario, hay un botón que al ser pulsado ejecuta una función que procesa la información del formulario en una estructura de tipo Json. Finalmente, utilizamos

[1] Unity: `https://unity.com/es`

[2] Boostrap: `https://getbootstrap.com/`

la librería Axios, mencionada en el capítulo de 4 para enviar esa estructura como petición post al servidor.

Podemos ver un ejemplo del formulario (Figura 5.1).

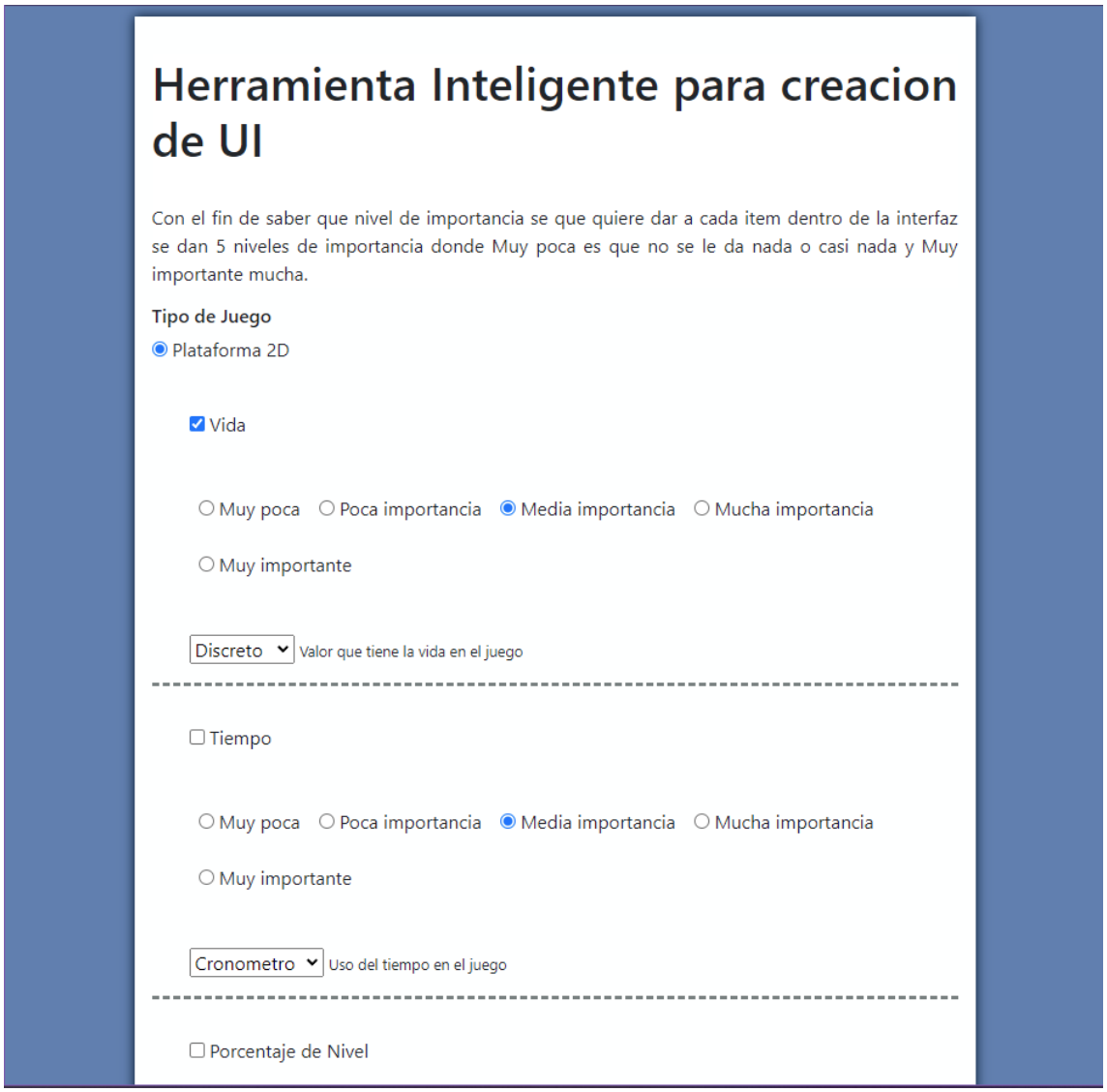

Figura 5.1: Aspecto final del formulario HTML

5.2. Plug-in en Unity

Para visualizar el resultado obtenido por nuestra herramienta, hemos implementado un plug-in en el motor Unity cuyo objetivo es interpretar el archivo devuelto y construir una interfaz en el editor para que el usuario pueda ver el resultado de forma gráfica.

Uno de los aspectos más importantes de este plug-in es el proceso sea sencillo y rápido para el usuario, es decir, que el número de pasos a realizar para poder utilizarlo sea el menor posible. Esto se ha tenido en cuenta a la hora de diseñar esta parte del proyecto, como se describirá en el próximo apartado.

Antes de pasar a explicar el diseño y la implementación del plug-in, es importante conocer qué es Unity y por qué hemos decidido usarlo para terminar el ciclo de nuestro

trabajo. Unity es un motor de videojuegos multiplataforma creado por Unity Technologies, disponible como plataforma de desarrollo para Microsoft Windows, Linux, Mac OS. Contiene soporte de compilación para multitud de plataformas (Windows, PlayStation, Xbox y Android entre otras). Además, es la plataforma usada por más de 1.3 millones de desarrolladores en todo el mundo, siendo el motor de muchos juegos de renombre como Hearthstone o Subnáutica.

Por tanto, la decisión de implementar el plug-in en esta plataforma tiene sentido sabiendo que es usada por muchos usuarios tanto jugadores como desarrolladores. Por otra parte, el propio motor nos ofrece varias herramientas que facilitan mucho el desarrollo de la propia herramienta. Utilizaremos los componentes para generar una interfaza partir del archivo de definición de la interfaz creado desde nuestra herramienta. Uno de los componentes cruciales que usaremos es el `Canvas`, el cual representa el espacio de la interfaz gráfica dentro de una escena de Unity, y además nos aporta elementos muy útiles como los pivotes, que sirven para establecer posiciones de objetos relativas a un espacio de la pantalla concreto, controlando así los problemas que trae la multitud de resoluciones que existen y facilitando los cálculos que realiza nuestra herramienta para posicionar todos los elementos en pantalla.

5.2.1. Diseño del plug-in

Nos encontramos en el punto en el que el usuario de nuestra aplicación tiene un fichero en formato JSON que guarda la información necesaria para construir la interfaz propuesta por nuestra herramienta, pero aún no puede visualizarla. Por tanto, enfocaremos el diseño del plug-in a resolver ésta última tarea y sobre todo facilitar el proceso al usuario.

La forma en la que el usuario va a interactuar con nuestro plug-in en el editor de Unity se realizará a través del inspector de un componente. Dicho componente tendrá varios campos públicos (visibles en el inspector) (Figura 5.2):

- Un campo `file`, que usaremos para especificar el fichero generado por nuestra herramienta.
- Un campo numérico que indica la separación que se generará entre cada elemento colocado, siendo parametrizable por el usuario.
- Varios campos de imágenes. Éstas serán la imágenes que se usarán para cada elemento que se coloque en la interfaz, por lo que si el usuario quiere que se use un sprite en concreto y no el que nosotros le hemos asignado por defecto sólo tendrá que asignarlo en el campo correspondiente al elemento.
- En la parte final de los atributos, el componente tendrá un botón que una vez pulsado llamará a la función encargada de interpretar y construir la interfaz deseada dentro de un canvas en la escena de Unity, siempre y cuando el fichero JSON sea correcto y se encuentre en el campo asignado del componente.

Antes de describir el proceso que determina la posición de cada elemento en pantalla, conviene aclarar varios aspectos a tener en cuenta:

- Nos referimos por conjunto de elementos a todos aquellos elementos que se sitúan en el mismo espacio de la pantalla. Estos espacios de pantalla han sido definidos y explicados en el capítulo 3.2.7.

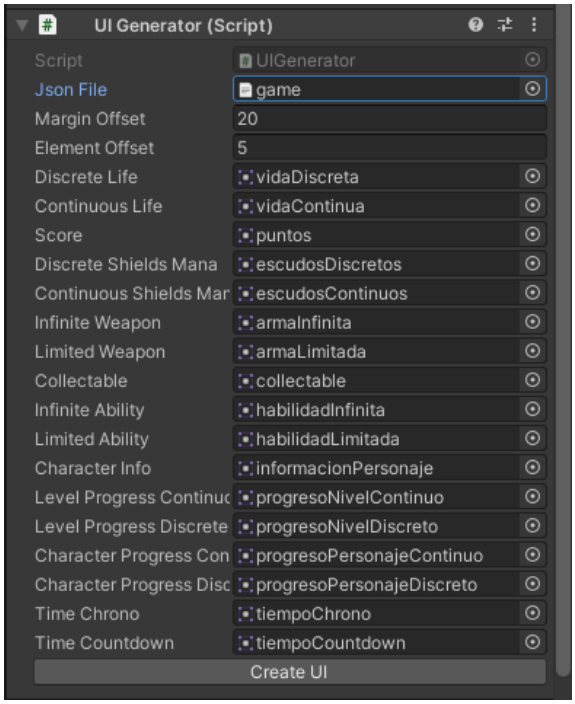

Figura 5.2: Ejemplo del componente

- El método que usará la herramienta para colocar cada conjunto de elementos en su posición correcta de la pantalla consiste en posicionar todos los elementos de dicho conjunto con una posición relativa a un pivote que servirá como base para formar una primera instancia de ese conjunto de elementos.

  Esto es realmente importante, pues gracias a ese pivote, más adelante podremos desplazarlo a la posición de la pantalla donde queramos situar a todo el conjunto, y dado que las distancias se hicieron en relación a dicho pivote, todos los elementos que lo contienen mantendrán las mismas distancias que cuando fueron colocados inicialmente. Esto simplifica mucho el proceso de posicionamiento de conjuntos en pantalla como se detallará más adelante.

Una vez aclarados estos conceptos y terminado la fase de interpretación del archivo, comienza un proceso para determinar la posición y escala de todos los elementos, siendo dicho proceso la siguiente sucesión de pasos para cada conjunto de elementos entrante:

1. Primero hay que diferenciar entre posiciones en pantalla de los conjuntos de elementos: horizontales (centro de pantalla tanto a la derecha como izquierda), verticales (centro de pantalla tanto arriba como abajo) y esquinas (resto de posiciones de los bordes de pantalla). Esta diferenciación se realiza para determinar en qué posición se va a colocar el pivote de referencia para los elementos. De igual forma se usará para determinar si se debe forzar la alineación horizontal en caso de encontrarnos en posiciones laterales, o vertical si el conjunto pertenece a una posición vertical. En el caso de las esquinas la alineación de cada elemento puede ser cualquiera de las dos.

2. Se busca el elemento más importante del grupo, determinado por su atributo de escala (el mayor será el más importante), seleccionándolo para ocupar el espacio más representativo de la sección de la pantalla destinada a su grupo. Por ejemplo: si el

elemento más importante seleccionado se encuentra en el conjunto situado en la parte superior izquierda de la interfaz, el elemento seleccionado ocupará la posición más próxima al pivote, dado que dicho pivote se encuentra en la esquina.

3. Una vez obtenido el elemento más importante, el resto se colocarán siguiendo la alineación escogida en el primer paso. En el caso de las esquinas se comprueba si el ancho de la imagen del elemento a colocar es mayor que su alto, y se compara con el ancho y alto total del conjunto de elementos que ya han sido colocados. Esta relación determina si el nuevo elemento se alineará en vertical u horizontal.

Como ha sido mencionado anteriormente, una vez colocados todos los elementos en relación al pivote de referencia, se trasladará dicho pivote (arrastrando a todos los elementos con él) a la posición en pantalla destino. Para que todos los elementos desplazados queden colocados de forma simétrica respecto al centro de la pantalla, se deberán de invertir las posiciones si el traslado del conjunto supone pasar su pivote al lado contrario de la pantalla respecto al punto central de la pantalla. Un ejemplo en el que los elementos se desplazan de una a otra esquina inferior de la pantalla siguiendo la simetría descrita anteriormente se puede observar en la figura 5.3.

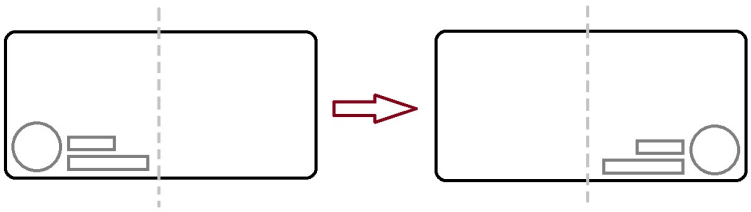

Figura 5.3: Ejemplo de la traslación de un conjunto de elementos

Una vez realizado el proceso descrito para un conjunto de elementos entero, se determina el espacio en pantalla que ocupará dicho conjunto (y por tanto el espacio individual de cada elemento). Si el conjunto contenía un número grande de elementos (considerando que no va a haber grupos que contengan más de 4 elementos a lo sumo), se le asignará más espacio en pantalla que a otro conjunto que contenga una menor cantidad, dejando así que los conjuntos de elementos más numerosos puedan verse con la misma facilidad que los que tienen menos.

Cabe mencionar que el máximo espacio que puede ocupar un conjunto de elementos en pantalla es de un tercio del ancho total en horizontal y un tercio del alto total en vertical, ya que existen tres posibles ubicaciones tanto a lo ancho como a lo alto donde nuestros conjuntos pueden colocarse.

5.2.2. Implementación del plug-in

Como ha sido mencionado anteriormente, una de las prioridades para este plug-in en Unity es que el proceso sea cómodo y no requiera demasiado tiempo. En consecuencia, hemos optado por usar los inspectores de componentes que nos proporciona Unity para facilitar el uso de la herramienta al usuario. De esta manera, el usuario lo único que debe hacer para poder utilizar la herramienta es arrastrar el fichero con la definición de interfaz que se haya descargado tras rellenar el formulario a su correspondiente entrada en el inspectory pulsar un botón para generar la interfaz. Los inspectores de componentes (Custom Editors) son scripts que reemplazan la apariencia en el inspector de un componente por

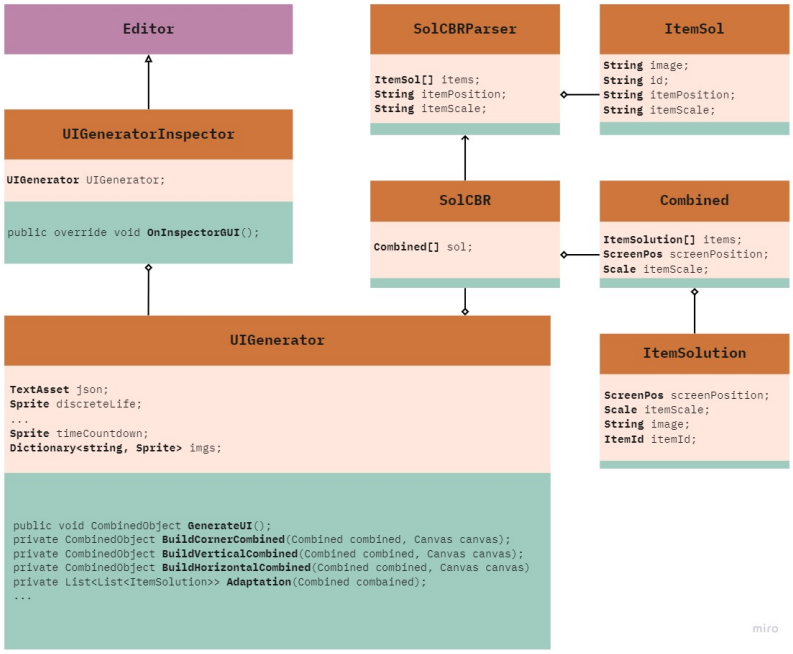

Figura 5.4: Diagrama de clases UML en Unity

otra personalizada por el programador. Dentro de las posibilidades que ofrecen los inspectores de componentes se encuentra la opción de crear un botón que llame a una función determinada, lo cual nos servirá para implementar lo mencionado en la sección de diseño.

Nuestro primer script llamado `UIGeneratorInspector` implementará el inspector del componente `UIGenerator`, el cual es el encargado de generar la UI en la escena. En el inspector se ha añadido el botón que llama a dicha funcionalidad. También es necesario el campo donde el usuario indicará que archivo quiere interpretar, aunque este campo se encontrará en el script `UIgenerator`. El diagrama de clases puede observarse en la figura 5.4.

Para realizar el parsing del archivo de texto a una clase que implementaremos a mano, `SolCBRParser`, usaremos la clase `JsonUtility` proporcionada por Unity que permite rellenar una clase definida por el programador con los campos de un JSON. Dado que este proceso se realiza mediante introspección, los atributos de la clase que será rellenada deben de tener el mismo nombre que los campos del JSON. Además, como la utilidad de Unity que hace la deserialización reconoce los campos que antes de ser serializados eran un enum como cadenas de texto, utilizamos una segunda clase, `SolCBR`, que tiene la misma información que la primera pero trabaja de nuevo con enums faciliando el trabajo.

5.3. Ejemplo de uso

Con el fin de comprobar que el proceso realizado es idóneo y utilizando el caso de uso anteriormente explicado hemos realizado una prueba sobre los casos de los que partimos. Para este ejemplo hemos partido del juego de Monster Boy and the Cursed Kingdom (Figura 5.5) en la cual podemos ver:

- En la posición ARRIBA-IZQUIERDA un elemento vida discreto y un elemento información del personaje.
- En la posición ABAJO-IZQUIERDA dos elementos de habilidades limitadas.

- En la posición ARRIBA-DERECHA un elemento de puntuación.

Figura 5.5: Interfaz del juego Monster Boy and the Cursed Kingdom

Para iniciar el proceso de comprobación, la primera fase será pasar por el formulario HTML (Figura 5.6) y rellenar los datos de acuerdo a lo que vemos en la interfaz del juego. Para ello, nosotros hemos introducido los mismos valores (la importancia y valor de los atributos del elemento) que guardan la base de casos.

Figura 5.6: Formulario realizado como ejemplo

Una vez enviemos la petición, nuestra herramienta devolverá la definición de la interfaz en formato JSON (Figura 5.7). En este archivo podemos observar diferentes partes:

- En el objeto combinado que tiene `screenPosition` con valor *TOP-LEFT* (parte izquierda de la imagen) encontramos dos elementos: una vida o *HEALTH* (discreta y con escala media) y una información del personaje o *CHARACTER-INFO* (con una escala grande)
- En el objeto combinado que tiene `screenPosition` con valor *TOP-RIGHT* (parte izquierda de la imagen) encontramos un elemento: una puntuación o *SCORE* con tamaño grande.

- En el objeto combinado que tiene `screenPosition` con valor *BOTTOM-LEFT* (parte derecha de la imagen) encontramos dos elementos: ambos de ellos son habilidades o *ABILITIES* las cuales son limitadas y cuya escala es grande.
- Encontramos (parte derecha de la imagen al final) dos valores: el `id`, que nos indica el caso que ha sido elegido de la base de casos. Este es el quinto caso, el cual está asociada a la imagen de referencia del juego anteriormente mencionado. Por otra parte tenemos el atributo `sim`, que guarda el valor de la similitud entre el caso devuelto y el de la petición. Podemos observar que el valor es 1, afirmando así que el caso obtenido por la herramienta es el idóneo (puesto que es el que hemos introducido por petición).


```
{
  "sol": [
    {
      "items": [
        {
          "screenPosition": "TOP_LEFT",
          "itemScale": "MEDIUM",
          "image": "vidaDiscreta",
          "id": "HEALTH"
        },
        {
          "screenPosition": "TOP_LEFT",
          "itemScale": "BIG",
          "image": "informacionPersonaje",
          "id": "CHARACTER_INFO"
        }
      ],
      "screenPosition": "TOP_LEFT"
    },
    {
      "items": [],
      "screenPosition": "TOP_CENTER"
    },
    {
      "items": [
        {
          "screenPosition": "TOP_RIGHT",
          "itemScale": "BIG",
          "image": "puntos",
          "id": "SCORE"
        }
      ],
      "screenPosition": "TOP_RIGHT"
    },
    {
      "items": [],
      "screenPosition": "MIDDLE_LEFT"
    },
    {
      "items": [],
      "screenPosition": "MIDDLE_CENTER"
    },
    {
      "items": [],
      "screenPosition": "MIDDLE_RIGHT"
    },
    {
      "items": [
        {
          "screenPosition": "BOTTOM_LEFT",
          "itemScale": "BIG",
          "image": "habilidadLimitada",
          "id": "ABILITIES"
        },
        {
          "screenPosition": "BOTTOM_LEFT",
          "itemScale": "BIG",
          "image": "habilidadLimitada",
          "id": "ABILITIES"
        }
      ],
      "screenPosition": "BOTTOM_LEFT"
    },
    {
      "items": [],
      "screenPosition": "BOTTOM_CENTER"
    },
    {
      "items": [],
      "screenPosition": "BOTTOM_RIGHT"
    }
  ],
  "id": 5,
  "sim": 1
}
```


Figura 5.7: JSON obtenido por la herramienta

Este JSON lo usamos en la herramienta del caso de uso creada en Unity y obtenemos la imagen que encontramos debajo (Figura 5.8).

5.4. Conclusiones

Tras el ejemplo de uso descrito en el anterior apartado, podemos observar que nuestra herramienta ha generado una interfaz en la que todos los elementos de la interfaz del juego original (Monster Boy and the Cursed Kingdom) han sido reflejados en la solución.

Figura 5.8: Interfaz creada en Unity

También se puede apreciar que el elemento de la vida (situado en la esquina superior izquierda de la pantalla) se sitúa debajo del icono de personaje, mientras que en el juego original tiene una alineación horizontal respecto al icono del personaje. Esto ocurre porque la herramienta agrupa los conjuntos de elementos de la forma definida en el apartado 5.2.1. Sin embargo, esto no influye en el resultado final, en el que los grupos de elementos están situados en los espacios de pantalla correspondientes y respetando la importancia que el usuario ha dado a cada elemento.

Capítulo 6

Evaluación

En este capítulo se cuenta la evaluación que se ha realizado sobre nuestra herramienta. Nuestro objetivo es comprobar la precisión de nuestro sistema CBR para conocer como han sido los resultados y saber si cumple correctamente su función. Es la parte principal de nuestra herramienta y del proyecto en general y por eso hemos decidido valorar diferentes aspectos en su comportamiento.

También nos interesaría evaluar el rendimiento del proceso de evaluación para hacernos una idea de cómo evoluciona el tiempo de ejecución necesario conforme con el número de casos para evaluar el sistema cuando el número de casos del mismo aumenta.

6.1. Evaluación del sistema CBR

Conocer el funcionamiento de nuestro sistema CBR es de gran importancia dado que forma el grueso del proyecto y es este el encargado de que la solución dada sea realmente buena y por lo tanto útil. Para ello hemos diseñado una serie de pruebas basadas en objetivos que nos permitan ver la calidad de su funcionamiento.

6.1.1. Objetivos y métricas

Durante el proceso de evaluación nos centraremos en dos objetivos con sus correspondientes preguntas de investigación:

- Analizar si la herramienta es rápida (a nivel de eficiencia)
  - ¿Se obtienen tiempos de poca duración durante una ejecución?
    Métrica: Tiempo medio por ejecución de varias peticiones en función del número de casos.
  - ¿Se obtienen tiempos de poca duración durante la búsqueda de un caso?
    Métrica: Tiempo medio por ciclo de una varias peticiones en función del número de casos.
- Analizar si la precisión de la herramienta es idónea.

  Métrica: Similitud media por ejecución de varias peticiones en función de número de casos.

6.1.2. Metodologías para métricas

Para realizar la métricas, haremos uso de la técnica *LeaveOneOut*, la cual consiste en seleccionar uno de los casos existentes en la base de casos, extraerlo de la misma y realizar una consulta al sistema utilizando como descripción de la consulta los datos del caso extraído.

- Para las métricas de medición de tiempos, en jCOLIBRI existen funciones procedentes de la clase `Evaluator` que realizan este tipo de mediciones en ejecución. Esta clase nos permitirá medir el tiempo de ejecución tanto de un ciclo(proceso de extracción de un caso y consulta al sistema utilizándolo como descripción) como de la ejecución total (es decir, el conjunto de todos los ciclos).

  Dado que nos interesa ver la evolución de dichos tiempos en función del tamaño de la base de casos, realizaremos diferentes ejecuciones con distintos tamaños de la base de casos en las que obtendremos los datos deseados.

- Por otra parte, para la métrica que calcula la similitud media, realizaremos la técnica de *LeaveOneOut* de manera manual y una vez el sistema nos devuelva la solución, comprobaremos la similitud entre el caso que ha seleccionado el sistema para crear la solución con el caso de la consulta. Este valor indica la similitud entre el caso extraído inicialmente y el que más se parece a él dentro de la base de casos, dando a entender que los valores bajos indican una distancia considerable entre los dos casos y valores altos indicarán por el contrario que ambos casos son muy parecidos entre si.

  Como nuestra intención es comprobar la precisión global del sistema, realizaremos la técnica descrita para todos los casos de nuestra base de casos. Una vez terminado el proceso, se calculará la media de todos los resultados obtenidos en cada ciclo, consiguiendo finalmente el valor que nos servirá para determinar el nivel de precisión del sistema.

6.2. Resultados y Conclusiones

Respecto a la precisión hemos podido comprobar (Figura 6.1) que conforme se aumenta la base de casos aumenta la media de la similitud, llegando a 0.65 con 63 casos. Podemos afirmar que la evolución gradual del número casos va de la mano con la similitud y si un usuario realiza una consulta con una base de casos mayor o igual a la realizada tendrá un buen resultado. Cabe recordar que los casos que superan el 0.8 de similitud no son añadidos a la base de casos y es por esto que llegar a un 0.65 de media es una buena señal de que funciona correctamente. Podemos concluir por tanto que nuestra herramienta consta de una precisión idónea.

Por otra parte, los resultados obtenidos del proceso de medición de tiempos se pueden observar en la figura 6.2. Podemos observar que el tiempo de ejecución es relativamente bajo, pero va aumentando gradualmente conforme la base de casos aumenta, siendo de 918 milisegundos cuando tenemos 63 casos. Respecto al tiempo por ciclo, podemos afirmar que es un tiempo constante (varía entre 10 y 20) y no depende del número de ciclos. El hecho de que el tiempo por ciclo sea relativamente constante es una buena noticia, ya que la eficiencia del ciclo del programa no depende del número de casos. Sin embargo se debería de mejorar el tiempo de ejecución completo(es decir, de varios ciclos) dado que si este que aumenta gradualmente y con bases de casos más elevadas puede llegar a tardar.

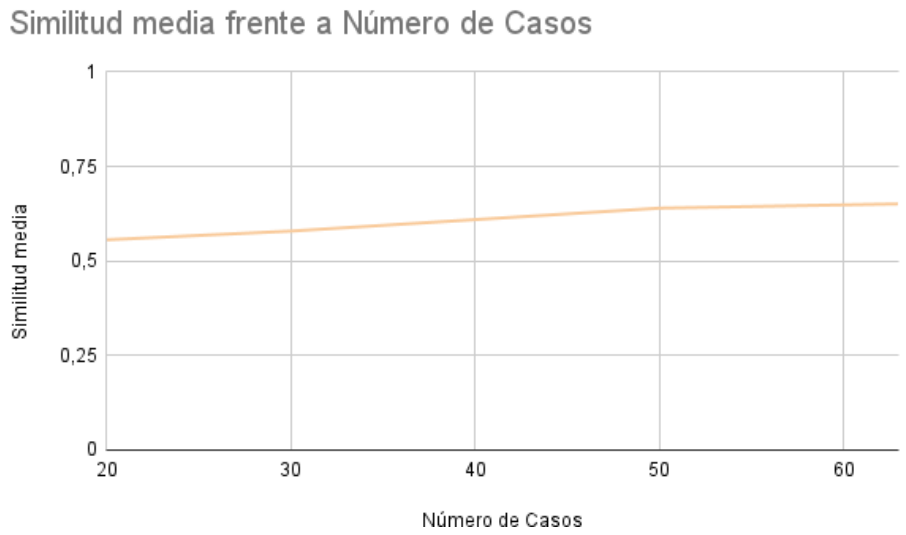

Figura 6.1: Similitudes por número de casos en la base de casos

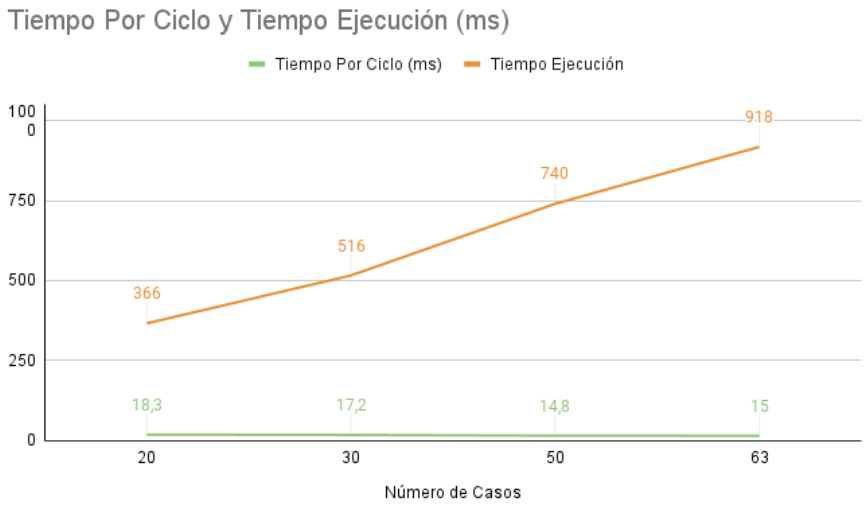

Figura 6.2: Tiempos de ejecución por número de casos en la base de casos

Capítulo 7

Conclusiones y Trabajo Futuro

El objetivo principal de este proyecto ha sido el de diseñar e implementar una herramienta generalista capaz de crear una interfaz para un género de videojuego determinado de manera inteligente. En nuestro caso, para el género de plataformas 2D. La herramienta funciona basándose en un sistema CBR.

Para ello, hemos realizado una investigación sobre diversas interfaces en distintos juegos del género con el fin de obtener información sobre los elementos comunes de las interfaces del género. Aunque ya teníamos experiencia jugando a juegos de estos géneros, el haber hecho un estudio mucho más exhaustivo de la interfaces de estos nos ayudó principalmente a detectar patrones de diseño comunes a la mayoría de juegos de los que seguramente no nos habríamos dado cuenta si no hubiésemos hecho esta investigación dedicando el tiempo requerido. Aunque podríamos haber aumentado el número de juegos objeto de estudio, consideramos que las conclusiones extraídas son lo suficientemente fuertes como para considerarlas fiables. Muchos de estos juegos hacen sus interfaces similares con el objetivo de que el jugador pueda aplicar los conocimientos adquiridos en títulos que haya jugado anteriormente, de manera que se consiga reducir al máximo la fricción entre el jugador y el juego mediante la interfaz. Finalmente, las conclusiones extraídas, que en su mayoría tienen que ver con las representaciones y posiciones (incluyendo agrupaciones) de los elementos nos fueron de gran ayuda a la hora de decidir como calcular la similitud y sobre todo en la fase de adaptación del ciclo CBR.

La fase de adaptación en nuestro proyecto es de especial importancia. Por una parte, por las características del problema que se plantea, en el que debemos respetar las decisiones tomadas por el diseñador de manera que la solución de la herramienta siempre dé los elementos y sus atributos que el diseñador haya elegido. Además, el hecho de que unas interfaces contengan algunos elementos mientras que otras no y haya que respetar los elementos solicitados por el usuario, obliga a que algunos de los elementos desparezcan o haya que incluir nuevos a la solución del caso más similar aumentando el peso de la adaptación. Por otra parte, la fase de adaptación es la oportunidad perfecta para comprobar y asegurar que se cumplen algunas de las Leyes de la Gestalt como por ejemplo la de proximidad entre elementos relacionados. Todas estas características han obligado a hacer una adaptación local, elemento por elemento, y una global.

Respecto al cálculo de la similitud entre dos casos, es otra de las partes con mayor peso de la herramienta ya que es lo que hará que se elija entre un caso u otro como base previa a la adaptación. Las funciones de similitud parcial las hemos realizado en su mayoría utilizando similitud por intervalo, mientras que la similitud global es una media ponderada de las similitudes parciales. Esto se debe a que la mayoría de la información solicitada al

usuario y la que compone la descripción de un caso es de carácter cualitativo. Es más, alguna como podría ser la importancia de un elemento, la cual se relaciona en gran medida con su tamaño en la interfaz, podría haber sido un valor numérico, pero consideramos que era mejor reducir los posible valores y hacerlo algo cualitativo principalmente porque facilita la tarea del diseñador.

La herramienta ha sido implementada como un servicio web al que se realizan peticiones. Para el caso de uso hemos decidido hacerlo mediante un formulario HTML, el cual es lanzado desde un servidor REST. Este servidor es el encargado de inicializar a su vez el sistema CBR. Haber utilizado Spring ha hecho que la implementación del servidor sea mucho más sencilla. El servidor procesa la petición y se la envía al CBR con el fin de obtener una solución. Tras completar el ciclo de CBR y obtener la solución deseada, el servidor devolverá al usuario la información en un archivo con un formato genérico y no dependiente de la plataforma en la que se va a presentar. Este era otro de los principales objetivos de nuestro proyecto, que la información devuelta por el sistema CBR fuese conceptual, en el sentido de que aunque es un fichero en formato JSON, es una definición de interfaz completa. La recogida de datos del usuario mediante un formulario HTML y la interpretación de esta en unity solo forman parte del caso de estudio que pone en práctica la herramienta.

Para poder visualizar la solución ofrecida por la herramienta, hemos realizado un plug-in en el motor de videojuegos Unity, el cual es capaz de interpretar el fichero genérico devuelto por la herramienta y construir la interfaz descrita por él dentro de una escena de Unity, permitiendo al usuario ver el resultado final. Aunque se podría haber dotado de mayor funcionalidad a la interfaz que se muestra en el motor, consideramos que esto no tenía demasiado sentido ya que el objetivo es generar un prototipo que pueda ayudar al usuario y no proporcionar la interfaz final. Decisiones como permitir que el usuario pueda modificar las imágenes que se usan para generar el prototipo acercan este al resultado final.

7.1. Trabajo Futuro

Este proyecto puede ser extendido en varias áreas, así para mejorarlo o darle un acabado más cerrado.

Una primera ampliación del proyecto sería hacer que nuestra herramienta pudiera generar interfaces del género First Person Shooter. Uno de nuestros objetivos iniciales fue hacer una herramienta tanto para plataformas·2D como para First Person Shooter. Este segundo género, aunque se hizo la parte del análisis, no se implementó en la herramienta. Para realizarlo, se deberían de implementar los métodos de similitud y adaptación de acuerdo a los datos y conclusiones que obtuvimos en nuestro estudio de interfaces de dicho género. Además, sería necesario crear las clases necesarias para la descripción del caso y la solución del mismo.

Otras ampliaciones que se podrían hacer vienen respecto al formulario HTML. Este consta un estilo muy básico y simple y podría mejorarse a uno más atractivo para que el usuario final lo entienda con mayor facilidad. Uno de los añadidos sería realizar la página web de forma dinámica (ya que actualmente se presenta como una web estática proporcionada por el servidor) con el fin de que la modificación de alguno de los apartados en el sistema CBR modifique la creación del formulario.

Por otra parte y con el fin de que el acabado de la herramienta quede más profesional o más cerrada a un mismo caso de uso (dado que ahora debemos de conectarnos en un formulario HTML, descargar el archivo y usarlo en Unity) se propone realizar la visualización de la interfaz final desde el propio navegador, con el fin de cerrar el ciclo en un

explorador de internet, de tal forma que el propio documento HTML que nos muestra el formulario, nos mostrara también como queda la interfaz sin necesidad de que el usuario tenga que descargar ningún archivo si no desea utilizar ninguna herramienta externa para su visualización.

Otra alternativa a la anterior mejora sería ofrecer la oportunidad al usuario de poder utilizar y visualizar el resultado de la herramienta desde el propio editor de Unity, de forma que rellene el formulario y envíe la información desde el propio editor y una vez recibe su respuesta, la visualización de la solución se represente en la escena de Unity.

Como última propuesta de mejora y ampliación del proyecto, se podría llevar a cabo una evaluación con usuarios que permita averiguar si el resultado final de la aplicación es realmente útil para la funcionalidad que se le quiere dar, además de poder extraer información sobre posibles mejoras en la interacción con el usuario.

Chapter 8

Conclusions and Future Work

The main goal of this proyect has been the design and implementation of a generalist tool which is able to create an interface for a videogame of the platforms 2D genre. The tool is based on a Case-Based Reasoning system.

For that purpose we have made a research about different videogame interfaces of the genre in order to gather information about the common elements of their interfaces. Even though we have some experience playing this kind of games, the fact that we made an exhaustive study of game interfaces helped us to detect common features in design patterns that we probably could not notice if we did not made this research. Despite we could have increased the ammount of researched games, we consider that the conclusions are enought strong to rely on them. Many of this games make their interfaces in a similar way, so the player can apply his knowledge from previous games of the same genre that he played, and reduce the disengagement between the player and the game through the interface. Finally, the drawn conclusions from the research (most of them were about the element representations and screen position) were specially useful at the time to decide how to calculate the similarity between cases and the adaptation phase of our CBR system.

The adaptation phase of our proyect is of particular relevance. In the one hand, we have to respect the decisions taken by the designer, which means that an interface that does not match with some of the required game elements or have a wrong representation must not be given as a solution. Moreover, the fact that some interfaces has some elements that are not contained in others and the need of adding all the elements requested by the user involves making some elements disapear from the solution, and adding elements in other cases. On the other hand, it is the perfect opportunity to check that the Gestalt principles are fulfilled, for instance the proximity principle. All of this features have led to either a local adaptation, element by element, and a global adaptation.

Regarding the calculation of similarity between two cases, it is one of the other important parts of the proyect due to the fact that it is what will make the decision of choosing one case or another. The partial similarity functions have been mostly done using interval similarity, meanwhile the global similarity is a weighted average that uses the partial similarities. This is because most of the information requested to the user has a qualitative content. Furthermore, some atributes as the element importance could have been a numeric value, but we considered that it was better to have qualitative atributes rather than quantitative values to ease the process to the designer.

The tool has been implemented as a web service to which queries are send through a HTML formulary, which is given by a REST server. This server is also responsible of

initialize the CBR system. The server process the query and send it to the CBR cycle, waiting for a solution. After completing the CBR cycle and get the desired solution, the server will return to the user the information through a generic format file. This was another objective of our proyect, to get a conceptual information file from the tool. The data gathering from the user through a HTML formulary and the interpretation of the data in Unity is part of the case of study that puts the tool in practice.

With the purpose of visualize the solution provided by the tool, we have developed a plug-in in the Unity engine which is able to interpret the generic file returned by the tool and build the interface described on it, letting the user see the final result.

8.1. Future work

This proyect can be extended in many areas, either to improve it or give it a better finishing touches.

A first possible ampliation of our proyect is making our tool able to generate interfaces for the First Person Shooter genre. One of our original goals was making a tool for either platform 2D and First Person Shooter games. This second genre, despite that the analysis of it was made, was never implemented in the tool. To do this, the similarity and adaptation methods must be implemented according to the gathered data and conclusions that we got in our research made from the genre. Furthermore, it would be necessary to create the needed classes for the case descriptions and solutions.

Regarding the HTML formulary, there are other ampliations that can be made. This formulary has a basic and simple style, so it can be upgraded to get a formulary with a better look to the user. In addition, the web can be changed to be launched dynamically (currently is a static web) with the purpose of letting the CBR system change parts of the formulary.

Besides, with the goal of giving the tool a more profesional and close finish (due to the fact that currently we have to complete the formulary, download the file and import it to a Unity project), we suggest to make the visualization of the final interface through the web browser itself, with the aim of closing the CBR cycle in the browser. The result of this would be a web that gives us the formulary, and once it is send, the final result would be displayed in the same web, without the need of download any file if the user does not want to use any external tool to visualize the result.

Another alternative to the last upgrade would be to give the user the opportunity to use and visualize the result of our tool through the Unity editor, so the user would be able to fill the formulary and see the final interface returned by the tool in the unity scene.

As a last proposal, a user evaluation could be done to find out if the final result of the tool is actually useful for the desired functionality, also to extract information about possible upgrades with the user interaction.

Capítulo 9

Contribuciones

9.1. Nicolás Fenández Descalzo

El trabajo comenzó en el verano (Junio de 2021) con la lectura de un libro recomendado por el tutor para comenzar a familiarizarnos con el Razonamiento Basado en Casos (CBR), tanto con su funcionamiento como utilidades. Con el arranque del curso comenzamos escribiendo las conclusiones que habíamos sacado de la lectura del verano para tener un resumen con los conceptos más importantes. Además, también realicé la lectura de un libro también recomendado por el tutor (Hodent (2019)). Concretamente, de la parte que trata sobre las leyes de la Gestalt, con ejemplos y conclusiones para posteriormente traducirlo a un resumen que sintetizase la importancia de estas en el diseño de interfaces en videojuegos.

A continuación, comenzamos con la investigación sobre interfaces de videojuegos en sí. Como ya habíamos tomado la decisión previamente de centrarnos en videojuegos de plataformas 2D y FPS (First Person Shooter) previamente, comenzamos elaborando en colaboración una lista con los elementos que aparecían en juegos de ambos géneros y que serían objetivos de estudio. Para extraer la mayoría de juegos de ejemplo y sus interfaces nos basamos en la página de Game UI Database[1]. Estas listas se acabaron traduciendo en unas tablas que contenían información sobre todos los elementos listados para cada juego que tomamos como ejemplo. En estas tablas, yo trabajé en mayor medida en la de juegos de plataformas 2D mientras que mis compañeros trabajaban en la de juegos FPS aunque posteriormente me ayudaron añadiendo nuevos casos para equilibrar el trabajo. Una vez determinados que elementos de los listados aparecían en cada juego y que representación se usaba para ellos, llevé a cabo iteraciones sobre estas mismas tablas con mi compañera Sandra para determinar la importancia de cada elemento que aparecía, teniendo en cuenta su tamaño en pantalla y la posición en la que se mostraba. De la lista de elementos con los que trabajaríamos en un futuro, algunos fueros eliminados ya que aparecían en una cantidad muy pequeña de juegos y no merecía la pena tenerlos en cuenta. Una vez las tablas estaban completas, me junté de nuevo con mis compañeros con el objetivo de sacar conclusiones, en una primera instancia habladas y a continuación escritas junto con Pablo. En este punto, decidimos que sería conveniente primero centrarnos en que el sistemas estuviese completo para uno de los dos géneros escogidos y nos centramos en la implementación para juegos de plataformas 2D.

Paralelamente, comenzamos a pensar como sería la entrada y salida del sistema de CBR

[1]Game UI Database: `https://www.gameuidatabase.com/`

y una vez supimos que la entrada sería a través de una petición a un servidor REST nos pusimos con ello. Comenzamos valorando las distintas opciones para montar un servidor REST y finalmente nos decidimos por usar SpringBoot. Descargamos la plantilla y aunque tuvimos algún problema con ella por las versiones y dependencias montamos un primer prototipo sobre el que realizar algunas pruebas usando PostMan [2]. Con el funcionamiento básico implementado, colaboré con Sandra para crear las clases que prepararían al servidor para recibir la información del formulario y pasarla al sistema CBR.

Con el formulario y el servidor en funcionamiento, nos pasamos a la parte del CBR. Comenzamos, los tres juntos, por acoplar la librería jCOLIBRI al servidor y así hacer que esta se levantase y cerrase junto con el segundo. A continuación y de nuevo juntos, nos pusimos con el diseño del sistema de CBR especificando por escrito como íbamos a implementar cada una de las partes para que funcionase lo mejor posible aunque dejamos un poco abierta la parte de adaptación que sería revisitada más adelante. Con el diseño hecho, hice junto con Pablo una primera versión que aunque solo utilizaba uno de los elementos que habíamos pero que permitía un ciclo completo petición-respuesta. El diseño del sistema CBR tuvo que ser modificado en parte, principalmente, la función de similitud. Junto con Sandra, llevamos a cabo el nuevo diseño de estas funciones y su implementación. Durante esta fase, aunque no pertenecen como tal al sistema CBR, también se definió el formato del objeto json que devolvería el servidor como respuesta a la petición.

Para la parte de unity que haría visible la información del json devuelto por el servidor, comencé con Pablo haciendo un documento de diseño. En este se explicaba como sería el proceso para transformar los elementos del json en imágenes correctamente colocadas y escaladas y los módulos que compondrían esta parte. Mientras tanto, Sandra trabajaba en la fase de adaptación del CBR. También comencé junto con Pablo con la implementación a lo que se nos uniría Sandra más adelante ya que algunas de las decisiones de diseño tuvieron que ser replanteadas.

Sobre la evaluación, trabajé junto con mis compañeros en implementarla en nuestro proyecto. Mientras que Sandra trabajaba en el código propio de la evaluación, trabajé en la corrección de algunos últimos errores y en asegurarme de que la base de casos de la que dispondríamos para la evaluación estaba suficientemente completa y era adecuada para realizar pruebas concluyentes.

En cuanto a la memoria, hemos ido trabajando paralelamente a el resto de desarrollo procurando que cada vez que alcanzábamos un hito del proceso, todo fuese quedando escrito. En cuanto a los apartados que la componen, he trabajado en mayor medida en el estado de la cuestión, diseño e implementación. En la primera, en las leyes de la Gestalt y en la investigación sobre elementos comunes en interfaces de videojuegos. En la segunda en el diseño del sistema CBR y en la tercera en la recogida de datos del usuario y el servidor REST.

[2]Postman: `https://www.postman.com/`

9.2. Sandra Mondragón Lázaro

El trabajo que he llevado a cabo se remonta hasta antes del inicio del curso. En una primera instancia, mis compañeros y yo recopilamos información sobre Case Base Reasoning y realizamos un resumen del mismo, con el fin de ver las distintas implementaciones de un sistema. Por otra parte, también realicé investigaciones sobre las Leyes de la Gestalt con el fin de conocer cómo funciona el cerebro y la memoria humana frente a las interfaces de usuario. Otro de los temas importante que tuve que investigar fue sobre los tipos de interfaces en videojuegos, del cual realizamos un resumen con el fin de añadirlo en un futuro a nuestra memoria.

Con el fin de conocer qué elementos íbamos a utilizar para nuestro proyecto, elaboré junto con mis compañeros un listado de los mismos. Este listado serviría para la investigación que llevamos más adelante sobre los géneros de videojuegos plataformas 2D y First Person Shooter. Junto con Pablo elaboramos tablas de FPS, y tras terminar junto con Nicolás Fernández elaboramos juntos tablas de Plataformas. Estas tablas contenían el juego que se estaba analizando con el elemento que se asociaba. Con cada elemento observamos su importancia respecto a la inferfaz (determinada por la escala del objeto y apuntes que nosotros veíamos), la posición en la que se encontraba el elemento, su representación, y algunos datos de interés como si el valor era discreto o continuo. Para realizar estas tablas nos apoyamos de Game UI Database y en distintas imágenes o gameplays que encontramos de los juegos que quisimos analizar.

Una vez esta parte de la investigación quedó concluida, realizamos el diseño de como sería la salida y la entrada de nuestro sistema CBR. Además, comenzamos a realizar conclusiones de forma hablada de las tablas que habíamos realizado. En estas conclusiones nos dimos cuenta de que algunos elementos no se encontraban casi nunca o nunca en las interfaces de usuario para el género de Plataformas 2D, por lo que decidimos eliminarlos. También decidimos enfocarnos en Plataformas 2D para tener un ciclo cerrado y tener como trabajo futuro FPS con la información recopilada en las tablas.

Después, mientras mis compañeros pasaban a escrito las conclusiones, me dediqué a investigar e implementar la parte del formulario html con el fin de tener una primera toma de contacto con el usuario. Este formulario fue una primer prototipo del mismo, el cual simplemente guardaba la información sin pasársela a ningún tipo de servidor.

Busqué junto con Pablo Villapún Martín y Nicolás Fernández Descalzo sobre los servidores de REST. Nos decantamos a usar Spring Boot y realizamos un prototipo, el cual construimos a través de una plantilla, con el fin de ver el funcionamiento del mismo. Para ver como funcionaba usamos PostMan y realizamos algunas peticiones POST simples al servidor a la dirección /platform (que sería la que usaríamos en un futuro). Cabe mencionar que este servidor es el que despliega el sistema de CBR. Junto con Nicolás realicé las clases POJO necesarias para la entrada de la petición al servidor. Estas guardaban los datos que se habían puesto en una primera instancia en el prototipo del formulario, pero un futuro se añadirían más.

Realicé junto con mis compañeros el diseño para el sistema CBR, desde la descripción de un caso o la solución de un caso hasta la solución del propio sistema CBR (siendo esta un archivo .json que definimos). Una vez el diseño estuvo claro, añadimos la librería de jCOLIBRI al proyecto y realizamos juntos un pequeño prototipo del sistema CBR para comprobar que el ciclo completo (servidor-cbr) funcionaba correctamente. Para ello tuvimos que configurar desde el servidor el cierre y apertura de nuestro sistema cbr con ayuda de los Bean que nos ofrece REST.

Una vez comprobamos que esta parte funcionaba comencé a extender toda la parte del

formulario con el fin que se pudiera comunicar con el servidor. Para ello, añadí todos los elementos al formulario y en javaScript los procese a una estructura json con ayuda de un FormData. Para comunicar la parte del formulario con el servidor, utilicé la librería de axios, realizando una petición POST con la información de la estructura realizada. Además, en la parte del servidor procesé la petición del formulario para así convertirla en una petición para el sistema CBR.

Junto con Nicolás, realicé el diseño de las funciones de similitud del sistema y su implementación en código. Estas funciones sirven para la búsqueda del caso más similar dentro del sistema de CBR.

Después, y mientras mis compañeros realizaban el proceso del diseño del caso de estudio que realizaríamos en Unity, me dispuse a realizar la fase de adaptación de nuestro sistema CBR. Para ello, realicé tablas de similitud con el fin de que la salida del sistema contuviera objetos combinados siguiendo las Leyes de la Gestalt.

Una vez terminada la fase de adaptación, ayudé a mis compañeros con la implementación de la herramienta de Unity con el fin de que el diseño del mismo estuviera más apoyado en las Leyes de la Gestalt. Realicé algunos sprites con el fin de usarlos en la herramienta como muestra de elemento en la interfaz.

Como última parte realicé, mientras mis compañeros corregían distintos errores de última instancia, un proceso de evaluación sobre nuestra herramienta, en la cual calculé tiempos de ejecución de la misma para saber cuán eficiente era además de realizar una evaluación para saber la precisión de nuestro sistema.

Paralelamente a todo este trabajo, hemos ido escribiendo la memoria conforme realizábamos determinados objetivos. En una primera instancia hicimos uso de Google Drive y fuimos poco a poco pasando la información a un proyecto de Latex, rellenando los nuevos capítulos. He trabajado de forma solitaria en el capítulo de Introducción, en el apartado de Implementación de Case Based Reasoning y añadiendo en distintos sub-apartados como 3.2.1.5. Adaptación o 2.3.4. Etapas del Case-Based Reasoning.

9.3. Pablo Villapún Martín

Antes de comenzar el desarrollo del proyecto, nuestro tutor nos recomendó la lectura del libro Case-Based Reasoning - A Textbook (Richter y Weber (2013)), el cual leí durante el verano para interiorizar los aspectos más importantes del Case-Based Reasoning (CBR) y realicé un resumen del mismo para reunir información acerca del proceso y aplicaciones del Case-Based Reasoning, en concreto me centré en resumir los aspectos más importantes del ciclo Case-Based Reasoning junto con algunas de las técnicas que se aplican en el proceso para buscar dentro del espacio de soluciones disponible y determinar con la máxima exactitud la similitud entre casos. También incorporé algunas imágenes que sintetizan bastante bien partes clave del proceso Case-Based Reasoning.

Antes de recopilar información de interfaces de videojuegos, investigué sobre los tipos de interfaces existentes (diegéticas, no diegéticas, meta y espaciales) para catalogar de mejor forma la información que buscaríamos en las siguientes semanas.

En esta primera fase del trabajo, todos los miembros del grupo nos dedicamos a buscar y recolectar información sobre diferentes interfaces en la base de datos de videojuegos Game UI Database, donde reuní información tanto de interfaces del género de plataformas 2D como de First Person Shooter. Nos centramos en determinar la importancia, posición y representación, escribiendo todos estos datos en unas tablas comunes donde, una vez reunimos suficientes datos, debatimos y obtuvimos características similares de cada grupo de interfaces junto a las conclusiones en las que nos basaríamos más adelante para diseñar el sistema CBR. Sin embargo no todos los datos que recopilamos sobre las interfaces eran válidos, ya que algunas presentaban elementos en ellas que consideramos irrelevantes (como por ejemplo el elemento del minimapa en las interfaces del género Plataformas 2D, ya que aparecía en tan pocas ocasiones que decidimos eliminarlo de la lista de elementos). Este proceso de filtrado lo realicé junto a mis dos compañeros.

Una vez comenzado el trabajo junto a mis compañeros, investigué acerca de los servidores REST y sobre la herramienta de Spring Boot, la cual permite desplegar el servidor sobre el que hemos implementado el sistema CBR. Junto con Sandra Mondragón y Nicolás Fernández realicé una versión prototipo del servidor REST usando la herramienta postman.

Durante el proceso de diseño del sistema CBR colaboré junto con ambos compañeros para definir cómo sería el formato en el que se representan los datos de entrada y salida del servidor, al igual que con las clases internas del Case-Based Reasoning y diseño de las funciones de similitud.

Una vez comenzamos a implementar el código de la aplicación, realicé junto a Sandra Mondragón las clases destinadas a almacenar los datos de los juegos de plataformas 2D. Durante todo el proceso de desarrollo del ciclo de CBR colaboré en distintas partes, aunque cabe mencionar que durante casi todo el tiempo en el que escribimos el código de la aplicación, lo hicimos conjuntamente los tres miembros del grupo.

Ya en la fase final de la implementación de la herramienta, mientras que mi compañera Sandra Mondragón se dedicaba a hacer la página web donde se realizaría el cuestionario al usuario, mi compañero Nicolás Fernández y yo nos dedicamos a terminar la implementación de la estructura y ciclo de Case-Based Reasoning en el proyecto de Java y a probar con unos casos de prueba para asegurarnos que todo funcionaba perfectamente y los datos de salida eran coherentes y estaban escritos en el formato adecuado. Para realizar dichas pruebas creamos un fichero Json de ejemplo para introducirlo como input en nuestro sistema CBR, de tal forma que mediante depuración nos cercioramos que todas las operaciones dentro del sistema se realizaban correctamente y ofrecía la salida deseada.

Tras completar la parte del servidor, escribí parte de la memoria en referente a las tablas que habíamos obtenido en la primera parte del proyecto y conclusiones obtenidas de ellas, sintetizando las características comunes de todos ellos y la relación de su importancia con el tipo de juego en cuestión.

Trabajé junto con Nicolás Fernández en el diseño del Plug-in en Unity, describiendo cómo se colocarían los diferentes elementos en la pantalla, asignando y ajustando su escala tras haber interpretado el archivo Json devuelto por el servidor. Después de dejar el diseño claro, comenzamos a implementar la herramienta, comenzando por las clases encargadas de interpretar los datos del archivo Json entrante. También implementé el sistema para agrupar los elementos de una sección concreta de la pantalla en subgrupos dependiendo de su similitud entre sí. De igual forma realizamos el inspector de componente encargado de generar la interfaz en la escena a través de un botón que se encargaba de limpiar los elementos del canvas y reconstruir una nueva interfaz dado el archivo que el usuario hubiera indicado a la herramienta. Me encargué de implementar la traslación de los conjuntos de elementos de sus posiciones relativas a las definitivas en pantallas utilizando los pivotes proporcionados por Unity e invirtiendo los ejes de cada elemento colocado cuando resultara necesario por la traslación. Toda esta parte la escribí en la memoria en el apartado Caso de Estudio una vez terminada, diferenciando la parte de diseño de la herramienta de la implementación de la misma, y ofreciendo una explicación sobre las razones por las que decidimos implementar dicha herramienta en Unity.

En la parte final del proyecto, añadí manualmente a la base de casos de nuestro sistema CBR un conjunto de casos con sus correspondientes imágenes. Además, y junto a mis compañeros de proyecto, realizamos el proceso de evaluación de nuestro sistema en el que comprobamos que la base de casos contenía suficientes casos distintos entre sí para que fuera preciso al ofrecer una solución. En esta parte también corregí algún error en el código de la herramienta.

Bibliografía

Y así, del mucho leer y del poco dormir, se le secó el celebro de manera que vino a perder el juicio.

Miguel de Cervantes Saavedra

Dwi H. Widyantoro, B. H., U. Ungkawa. *Case-based Reasoning Approach for Form Interface Design*. Institute of Electrical and Electronics Engineers, Bandung, Indonesia, 2014. ISBN 978-1-4799-7996-7.

Fagerholt, E. y Lorentzon, M. *Beyond the HUD User Interfaces for Increased Player Immersion in FPS Games*. CHALMERS UNIVERSITY OF TECHNOLOGY, Göteborg, Sweden, 2009.

Hodent, C. *How neuroscience and UX can impact video game design*. 2017. ISBN 978-1498775502.

Hodent, C. *Cognitive Psychology Applied to User Experience in Video Games*. Springer, 2019.

Richter, M. M. y Weber, R. O. *Case-Based Reasoning - A Textbook*. Springer, 2013. ISBN 978-3-642-40166-4.

Capítulo 10

Anexo

En este apéndice se incluyen las tablas sobre las que se recogió información de las interfaces in-game de los videojuegos objeto de estudio, tanto juegos de Plataformas 2D como First Person Shooters.

10.1. Plataformas 2D

ARMA / HABILIDAD					
Juego	Posición	Representación	Importacia	Discreto/Continuo	Diegético
Ori and the Blind Forest	-	-	-	-	-
Blasphemous	-	-	-	-	-
Cuphead	Abajo Izquierda	Iconos	Poca Importancia	Discreto	-
Guacamelee 2	-	-	-	-	-
Monster Boy and the Cursed Kingdom	Abajo Izquierda	Iconos + barra	Importante	Discreto	-
Mad Rat Dead	-	-	-	-	-
New Super Mario Bros	-	-	-	-	-
New Super Mario Bros 2	-	-	-	-	-
Super Meat Boy	-	-	-	-	-
Rayman Adventures	-	-	-	-	-
Gonner 2	Arriba Izquierda	Icono	Muy Importante	Discreto	-
Spelunky 2	-	-	-	-	-
The End is Nigh	-	-	-	-	-
Tembo the Badass Elephant	-	-	-	-	-
Velocity 2X	-	-	-	-	-
Alex Kid in Miracle World DX	Arriba derecha	Icono	Muy Importante	-	-
Apotheon	Abajo Derecha	Iconos	Imporante	Discreto	-
Hollow Knight	-	-	-	-	-
Wonderboy: The Dragons Trap	Arriba derecha	Icono + numero	Importante	Discreto	-
SpongeBob: Patty Pursuit	-	-	-	-	-
10 Second Ninja X	Arriba medio	Icono	Importante	Discreto	-
Rogue Legacy	Arriba derecha	Icono	Importante	Discreto	-

Figura 10.1: Estudio de armas y habilidades en interfaces de videojuegos plataformas 2D

10.2. First Person Shooters

COLECCIONABLES					
Juego	Posición	Representación	Importacia	Discreto/Continuo	Diegético
Ori and the Blind Forest	-	-	-	-	-
Blasphemous	Arriba Izquierda	Icono	Poca Importancia	Discreto	-
Cuphead	-	-	-	-	-
Guacamelee 2	-	-	-	-	-
Monster Boy and the Cursed Kingdom	-	-	-	-	-
Mad Rat Dead	-	-	-	-	-
New Super Mario Bros	Arriba Izquierda	Iconos	Importante	Discreto	-
New Super Mario Bros 2	-	-	-	-	-
Super Meat Boy	-	-	-	-	-
Rayman Adventures	Arriba Cenro	Barra	Muy importante	Discreto	-
Gonner 2	Arriba Izquierda	Icono + número	Poca Importancia	Discreto	-
Spellunky 2	Arriba Izquierda	Icono + número	Poca Importancia	Discreto	-
The End is Nigh	Arriba Izquierda	Icono + numero	Poco importante	Discreto	-
Tembo the Badass Elephant	-	-	-	-	-
Velocity 2X	Abajo derecha	Icono + numero(fraccion)	Poco importante	Discreto	-
Alex Kid in Miracle World DX	-	-	-	-	-
Apotheon	-	-	-	-	-
Hollow Knight	-	-	-	-	-
Wonderboy: The Dragons Trap	-	-	-	-	-
SpongeBob: Patty Pursuit	Arriba derecha	Barra	Importante	Discreto	-
10 Second Ninja X	Arriba derecha	Icono + numero	Medio	Discreto	-
Rogue Legacy	-	-	-	-	-

Figura 10.2: Estudio de coleccionables en interfaces de videojuegos plataformas 2D

ESCUDOS / MANÁ					
Juego	Posición	Representación	Importacia	Discreto/Continuo	Diegético
Ori and the Blind Forest	Abajo Centro	Barra	Importante	Discreto	-
Blasphemous	Arriba Izquierda	Barra	Importante	Continuo	-
Cuphead	-	-	-	-	-
Guacamelee 2	Arriba Izquierda	Barra	Medio	Discreto	-
Monster Boy and the Cursed Kingdom	-	-	-	-	-
Mad Rat Dead	-	-	-	-	-
New Super Mario Bros	-	-	-	-	-
New Super Mario Bros 2	-	-	-	-	-
Super Meat Boy	-	-	-	-	-
Rayman Adventures	-	-	-	-	-
Gonner 2	Arriba Izquierda	Barra	Importante	Discreto	-
Spellunky 2	-	-	-	-	-
The End is Nigh	-	-	-	-	-
Tembo the Badass Elephant	Arriba Izquierda	Barra	Importante	Continuo	-
Velocity 2X	Abajo izquierda	Barra	Importante	Continuo	-
Alex Kid in Miracle World DX	-	-	-	-	-
Apotheon	Abajo Centro	Barra	Poco Importante	Discreto	-
Hollow Knight	Arriba Izquierda	Círculo	Muy importante	Continuo	-
Wonderboy: The Dragons Trap	-	-	-	-	-
SpongeBob: Patty Pursuit	-	-	-	-	-
10 Second Ninja X	-	-	-	-	-
Rogue Legacy	Arriba Izquierda	Barra + numero	Importante	Discreto	-

Figura 10.3: Estudio de escudos y maná en interfaces de videojuegos plataformas 2D

MINIMAPA					
Juego	Posición	Representación	Importacia	Discreto/Continuo	Diegético
Ori and the Blind Forest	-	-	-	-	-
Blasphemous	-	-	-	-	-
Cuphead	-	-	-	-	-
Guacamelee 2	-	-	-	-	-
Monster Boy and the Cursed Kingdom	-	-	-	-	-
Mad Rat Dead	-	-	-	-	-
New Super Mario Bros	-	-	-	-	-
New Super Mario Bros 2	-	-	-	-	-
Super Meat Boy	-	-	-	-	-
Rayman Adventures	-	-	-	-	-
Gonner 2	-	-	-	-	-
Spellunky 2	-	-	-	-	-
The End is Nigh	-	-	-	-	-
Tembo the Badass Elephant	-	-	-	-	-
Velocity 2X	-	-	-	-	-
Alex Kid in Miracle World	-	-	-	-	-
Apotheon	-	-	-	-	-
Hollow Knight	-	-	-	-	-
Wonderboy: The Dragons Trap	-	-	-	-	-
SpongeBob: Patty Pursuit	-	-	-	-	-
10 Second Ninja X	-	-	-	-	-
Rogue Legacy	Arriba derecha	Rectangular	Muy importante	-	-

Figura 10.4: Estudio de minimapa en interfaces de videojuegos plataformas 2D

PERSONAJE					
Juego	Posición	Representación	Importacia	Discreto/Continuo	Diegético
Ori and the Blind Forest	Abajo Centro	Circulo(progreso del personaje)	Importante	Continuo	-
Blasphemous	Arriba Izquierda	Circulo(progreso del personaje)	Medio	Continuo	-
Cuphead	-	-	-	-	-
Guacamelee 2	Arriba Izquierda	Icono Personaje	Medio	-	-
Monster Boy and the Cursed Kingdom	Arriba Izquierda	Icono Personaje	Importante	-	-
Mad Rat Dead	-	-	-	-	-
New Super Mario Bros	-	-	-	-	-
New Super Mario Bros 2	-	-	-	-	-
Super Meat Boy	-	-	-	-	-
Rayman Adventures	-	-	-	-	-
Gonner 2	-	-	-	-	-
Spelunky 2	-	-	-	-	-
The End is Nigh	-	-	-	-	-
Tembo the Badass Elephant	-	-	-	-	-
Velocity 2X	-	-	-	-	-
Alex Kid in Miracle World DX	-	-	-	-	-
Apotheon	-	-	-	-	-
Hollow Knight	-	-	-	-	-
Wonderboy: The Dragons Trap	-	-	-	-	-
SpongeBob: Patty Pursuit	-	-	-	-	-
10 Second Ninja X	-	-	-	-	-
Rogue Legacy	-	-	-	-	-

Figura 10.5: Estudio de información del peronaje en interfaces de videojuegos plataformas 2D

PROGRESO					
Juego	Posición	Representación	Importacia	Discreto/Continuo	Diegético
Ori and the Blind Forest	-	-	-	-	-
Blasphemous	-	-	-	-	-
Cuphead	-	-	-	-	-
Guacamelee 2	-	-	-	-	-
Monster Boy and the Cursed Kingdom	-	-	-	-	-
Mad Rat Dead	Abajo Izquierda	Barra	Media	Discreto	-
New Super Mario Bros	-	-	-	-	-
New Super Mario Bros 2	-	-	-	-	-
Super Meat Boy	-	-	-	-	-
Rayman Adventures	-	-	-	-	-
Gonner 2	Abajo Izquierda	Iconos (slots)	Importante	Discreto	-
Spelunky 2	Arriba derecha	Número	Muy poca	Discreto	-
The End is Nigh	-	-	-	-	-
Tembo the Badass Elephant	-	-	-	-	-
Velocity 2X	-	-	-	-	-
Alex Kid in Miracle World DX	-	-	-	-	-
Apotheon	-	-	-	-	-
Hollow Knight	-	-	-	-	-
Wonderboy: The Dragons Trap	-	-	-	-	-
SpongeBob: Patty Pursuit	-	-	-	-	-
10 Second Ninja X	-	-	-	-	-
Rogue Legacy	-	-	-	-	-

Figura 10.6: Estudio de progreso de nivel en interfaces de videojuegos plataformas 2D

PUNTUACIÓN / DINERO					
Juego	Posición	Representación	Importacia	Discreto/Continuo	Diegético
Ori and the Blind Forest	-	-	-	-	-
Blasphemous	Arriba Dcha	Numero	Importante	Discreto	-
Cuphead	-	-	-	-	-
Guacamelee 2	-	-	-	-	-
Monster Boy and the Cursed Kingdom	Arriba Dcha	Icono + numero	Importante	Discreto	-
Mad Rat Dead	Arriba Izquierda	Numero	Importante	Discreto	-
New Super Mario Bros	Arriba Izquierda	Icono + numero	Importante	Discreto	-
New Super Mario Bros 2	Arriba Izquierda	Icono + numero	Muy importante	Discreto	-
Super Meat Boy	-	-	-	-	-
Rayman Adventures	Arriba Centro	Numero	Muy importante	Discreto	-
Gonner 2	-	-	-	-	-
Spellunky 2	Arriba Derecha	Icono + numero	Poco importante	Discreto	-
The End is Nigh	-	-	-	-	-
Tembo the Badass Elephant	Arriba Izquierda	Icono + numero	Poco importante	Discreto	-
Velocity 2X	Abajo derecha	Icono + numero(fraccion)	Poco importante	Discreto	-
Alex Kid in Miracle World DX	Arriba Izquierda	Icono + número	Importante	Discreto	-
Apotheon	-	-	-	-	-
Hollow Knight	Arriba Izquierda	Icono + número	Poco importante	Discreto	-
Wonderboy: The Dragons Trap	Arriba Derecha	Icono + numero	Importante	Discreto	-
SpongeBob: Patty Pursuit		Barra + numero	Importante	Discreto	-
10 Second Ninja X	-	-	-	-	-
Rogue Legacy	Arriba izquierda	Icono + numero	Poco importante	Discreto	-

Figura 10.7: Estudio de puntuación en interfaces de videojuegos plataformas 2D

TIEMPO					
Juego	Posición	Representación	Importacia	Discreto/Continuo	Diegético
Ori and the Blind Forest	-	-	-	-	-
Blasphemous	-	-	-	-	-
Cuphead	-	-	-	-	-
Guacamelee 2	-	-	-	-	-
Monster Boy and the Cursed Kingdom	-	-	-	-	-
Mad Rat Dead	Arriba Izquierda	Número (cuenta atrás)	Importante	Discreto	-
New Super Mario Bros	Arriba Derecha	Numero(cuenta Atrás)	Media	Discreto	-
New Super Mario Bros 2	Arriba derecha	Número (cuenta atrás)	Poca	Discreto	-
Super Meat Boy	Arriba izda	Número (creciente)	Poca	Continuo	-
Rayman Adventures	-	-	-	-	-
Gonner 2	-	-	-	-	-
Spelunky 2	Arriba derecha	Icono + número (creciente)	Muy poca	Discreto	-
The End is Nigh	-	-	-	-	-
Tembo the Badass Elephant	-	-	-	-	-
Velocity 2X	Arriba izquierda	Icono + número (cuenta atras)	Media	Discreto	-
Alex Kid in Miracle World DX	-	-	-	-	-
Apotheon	-	-	-	-	-
Hollow Knight	-	-	-	-	-
Wonderboy: The Dragons Trap	-	-	-	-	-
SpongeBob: Patty Pursuit	-	-	-	-	-
10 Second Ninja X	Arriba medio	Icono + número (cuenta atrás)	Muy importante	Continuo	-
Rogue Legacy	-	-	-	-	-

Figura 10.8: Estudio de tiempo en interfaces de videojuegos plataformas 2D

VIDA					
Juego	Posición	Representación	Importacia	Discreto/Continuo	Diegético
Ori and the Blind Forest	Abajo Centro	Barra	Importante	Discreto	-
Blasphemous	Arriba Izquierda	Barra	Importante	Continuo	-
Cuphead	Abajo Izquierda	Número	Poca Importancia	Discreto	-
Guacamelee 2	Arriba Izquierda	Barra	Medio	Continuo	-
Monster Boy and the Cursed Kingdom	Arriba Izquierda	Corazones	Media	Discreto	-
Mad Rat Dead	-	-	-	-	-
New Super Mario Bros	Arriba Izquierda	Icono + numero	Media	Discrreto	-
New Super Mario Bros 2	-	-	-	-	-
Super Meat Boy	-	-	-	-	-
Rayman Adventures	-	-	-	-	-
Gonner 2	Arriba Izquierda	Corazones	Media	Discreto	-
Spellunky 2	Arriba Izquierda	Icono + numero	Importante	Discreto	-
The End is Nigh	-	-	-	-	-
Tembo the Badass Elephant	Arriba Izquierda	Barra	Importante	Continuo	-
Velocity 2X	Arriba Izquierda	Círculo (icono)	Importante	Continuo	-
Alex Kid in Miracle World DX	-	-	-	-	-
Apotheon	Abajo Centro	Barra	Poco Importante	Discreto	-
Hollow Knight	Arriba Izquierda	Corazones	Importante	Discreto	-
Wonderboy: The Dragons Trap	Arriba Izquierda	Corazones	Importante	Discreto	-
SpongeBob: Patty Pursuit	-	-	-	-	-
10 Second Ninja X	-	-	-	-	-
Rogue Legacy	Arriba Izquierda	Barra + Numero	Muy importante	Discreto	-

Figura 10.9: Estudio de vida en interfaces de videojuegos plataformas 2D

	Armas				
Juego	**Posición**	**Representación**	**Importancia**	**Discreto/Continuo**	**Diegético**
Apex legends	Abajo Derecha	Icono	Medio	-	-
Alien Rage	-	-	-	-	-
CoD bo3 (MP) (DOM)	-	-	-	-	-
Battefield 1(SP)	-	-	-	-	-
Bioshock					
Doom	Abajo Derecho	Icono	Importante	-	-
Far cry 4	-	-	-	-	-
Borderlands	-	-	-	-	-
Deep Rock Galatics	-	-	-	-	-
Destiny	Abajo Izquierda	Icono	Medio	-	-
Deus Ex. Humn	-	-	-	-	-
CSGO	Abajo Derecha	Icono	Muy Importante	Discreto	-
Far Cry 3	-	-	-	-	-
Metroid Prime	Abajo Derecha	Icono	Importante	-	Dentro del casco del personaje
Dead Space 2	-	-	-	-	-

Figura 10.10: Estudio de las armas en interfaces de videojuegos FPS

	Brújula				
Juego	**Posición**	**Representación**	**Importancia**	**Discreto/Continuo**	**Diegético**
Apex legends	Arriba Centro	Línea	Poca importancia	-	-
Alien Rage	-	-	-	-	-
CoD bo3 (MP) (DOM)	Arriba Izquierda(con mapa)	Linea	Muy poca	-	-
Battefield 1(SP)	-	-	-	-	-
Bioshock	-	-	-	-	-
Doom	Arriba Centro	Linea	Muy poca	-	-
Far cry 4	-	-	-	-	-
Borderlands	-	-	-	-	-
Deep Rock Galatics	Arriba Centro	Linea	Poca importancia	-	-
Destiny	-	-	-	-	-
Deus Ex. Humn	-	-	-	-	-
CSGO	-	-	-	-	-
Far Cry 3	-	-	-	-	-
Metroid Prime	Arriba Izquierda	Circulo	Importante	-	Dentro del casco del personaje
Dead Space 2	-	-	-	-	-

Figura 10.11: Estudio de la brújula en interfaces de videojuegos FPS

	Equipamiento				
Juego	**Posición**	**Representación**	**Importancia**	**Discreto/Continuo**	**Diegético**
Apex legends	Abajo Derecha	Icono H	Muy poca	-	-
Alien Rage	Abajo Derecha	Icono V	Media	-	-
CoD bo3 (MP) (DOM)	Abajo Derecha	Icono	Poca Importancia	-	-
Battefield 1(SP)	Abajo Derecha	Icono	Muy poca	-	-
Bioshock	Abajo izquierda	Texto	Poca Importancia	-	
Doom	Abajo Derecha	Icono	Poca Importancia	-	-
Far cry 4	Abajo Derecha	Icono + Número	Poca Importancia	Discreto	-
Borderlands	Abajo Derecha	Icono V	Poca Importancia	-	-
Deep Rock Galatics	Abajo Derecha	Icono H	Poca importancia	-	-
Destiny 2	Abajo izquierda	Icono H	Importante	-	-
Deus Ex. Humn	Abajo Derecha	Icono H	Poca importancia	-	-
CSGO	Abajo Derecha	Icono	Importante	Discreto	-
Far Cry 3	-	-	-	-	-
Metroid Prime	-	-	-	-	-
Dead Space 2	-	-	-	-	-

Figura 10.12: Estudio del equipamiento en interfaces de videojuegos FPS

	Escudos				
Juego	**Posición**	**Representación**	**Importancia**	**Discreto/Continuo**	**Diegético**
Apex legends	Abajo izquierda	Barra	Media	Discreto	-
Alien Rage	-	-	-	-	-
CoD bo3 (MP) (DOM)	-	-	-	-	-
Battefield 1(SP)	-	-	-	-	-
Bioshock	Arriba izquierda	Barra	Importante	Continuo	-
Doom	Abajo izquierda	Barra	Medio	Discreto	-
Far cry 4	-	-	-	-	-
Borderlands	Abajo Izquierda	Barra	Media	Continuo	-
Deep Rock Galatics	Abajo Izquierda	Barra	Media	Continuo	-
Destiny	-	-	-	-	-
Deus Ex. Humn	Arriba izquierda	Barra	Muy poca	Discreto	-
CSGO	Abajo Izquierda	Número + Barra	Media	Discreto	-
Far Cry 3	-	-	-	-	-
Metroid Prime	-	-	-	-	-
Dead Space 2	-	-	-	-	-

Figura 10.13: Estudio de los escudos en interfaces de videojuegos FPS

	Feedback Muertes (Online)				
Juego	**Posición**	**Representación**	**Importancia**	**Discreto/Continuo**	**Diegético**
Apex legends	Abajo Izquierda	Texto	Media	-	-
Alien Rage	-	-	-	-	-
CoD bo3 (MP) (DOM)	Abajo Izquierda	Texto	Media	-	-
Battefield 1(SP)	-	-	-	-	-
Bioshock	-	-	-	-	-
Doom	-	-	-	-	-
Far cry 4	-	-	-	-	-
Borderlands	-	-	-	-	-
Deep Rock Galatics	-	-	-	-	-
Destiny	-	-	-	-	-
Deus Ex. Humn	-	-	-	-	-
CSGO	Arriba Derecha	Texto + Icono	Muy poca	Discreto	-
Far Cry 3	-	-	-	-	-
Metroid Prime	-	-	-	-	-
Dead Space 2	-	-	-	-	-

Figura 10.14: Estudio del feedback de muertes en interfaces de videojuegos FPS

	Habilidades				
Juego	**Posición**	**Representación**	**Importancia**	**Discreto/Continuo**	**Diegético**
Apex legends	Abajo Izquierda	Icono	Poca importancia	-	-
Alien Rage	-	-	-	-	-
CoD bo3 (MP) (DOM)	Abajo Derecha	Circunferencia	Importante	-	-
Battefield 1(SP)	-	-	-	-	-
Bioshock					
Doom					
Far cry 4	-	-	-	-	-
Borderlands	-	-	-	-	-
Deep Rock Galatics					
Destiny	Abajo Izquierda	Icono	Medio	-	-
Deus Ex. Humn	-	-	-	-	-
CSGO	-	-	-	-	-
Far Cry 3	-	-	-	-	-
Metroid Prime	-	-	-	-	-
Dead Space 2	-	-	-	-	-

Figura 10.15: Estudio de las habilidades en interfaces de videojuegos FPS

	Minimapa				
Juego	**Posición**	**Representación**	**Importancia**	**Discreto/Continuo**	**Diegético**
Apex legends	Arriba Izquierda	Cuadrado	Importante	-	-
Alien Rage	-	-	-	-	-
CoD bo3 (MP) (DOM)	Arriba Izquierda	Cuadrado	Muy Importante	-	-
Battefield 1(SP)	Abajo Izquierda	Circulo	Importante	-	-
Bioshock	-	-	-	-	-
Doom	-	-	-	-	-
Far cry 4	Abajo Izquierda	Cuadrado	Muy importante	-	-
Borderlands	-	-	-	-	-
Deep Rock Galatics	-	-	-	-	-
Destiny	Arriba Izquierda	Circulo	Medio	-	-
Deus Ex. Humn	Abajo Izquierda	Cuadrado	Poca importancia	-	-
CSGO	Arriba Izquierda	Circulo	Muy importante	-	-
Far Cry 3	Abajo Izquierda	Circulo	Importante	-	-
Metroid Prime	Arriba Derecha	Circulo(mapa en 3D)	Importante	-	Dentro del casco del personaje
Dead Space 2	-	-	-	-	-

Figura 10.16: Estudio del minimapa en interfaces de videojuegos FPS

	Munición				
Juego	**Posición**	**Representación**	**Importancia**	**Discreto/Continuo**	**Diegético**
Apex legends	Abajo Derecha	Número	Importante	Discreto	Balas en el Arma
Alien Rage	Abajo Derecha	Número	Muy importante	Discreto	-
CoD bo3 (MP) (DOM)	Abajo Derecha	Número	Medio	Discreto	-
Battefield 1(SP)	Abajo Derecha	Numero	Media	Discreto	-
Bioshock	Abajo izquierda	Número	Importante	Discreto	-
Doom	Abajo Derecha	Número	Poca Importancia	Discreto	-
Far cry 4	Abajo Derecha	Icono balas	Importante	Discreto	-
Borderlands	Abajo Derecha	Número	Importante	Discreto	-
Deep Rock Galatics	Abajo Derecha	Icono balas	Medio	Discreto	-
Destiny 2	Abajo izquierda	Numero	Importante	Discreto	-
Deus Ex. Humn	Abajo Derecha	Numero	Poca Importancia	Discreto	-
CSGO	Abajo derecha	Número	Medio	Discreto	-
Far Cry 3	Abajo derecha	Icono balas	Medio	Discreto	-
Metroid Prime	Medio Derecha	Barra	Medio	Discreto	Dentro del casco del personaje
Dead Space 2	-	Numero	Media	Discreto	Balas en el arma

Figura 10.17: Estudio de la munición en interfaces de videojuegos FPS

	Objetivo				
Juego	**Posición**	**Representación**	**Importancia**	**Discreto/Continuo**	**Diegético**
Apex legends	-	-	-	-	-
Alien Rage	Arriba Derecha	Texto	Media	-	-
CoD bo3 (MP) (DOM)	-	-	-	-	-
Battefield 1(SP)	Centro Derecha	Texto	Media	-	-
Bioshock	-	-	-	-	-
Doom	Centro	Texto	Media	-	-
Far cry 4	Arriba Izquierda	Icono + Texto	Media	Discreto	-
Borderlands	-	-	-	-	-
Deep Rock Galatics	-				
Destiny	Centro Derecha	Texto	Media	-	-
Deus Ex. Humn					
CSGO	-	-	-	-	-
Far Cry 3	Arriba Izquierda	Texto	Media	-	-
Metroid Prime	-	-	-	-	-
Dead Space 2	-	Texto	Media	-	Aparece al lado del personaje dentro del juego

Figura 10.18: Estudio del objetivo en interfaces de videojuegos FPS

	Poción				
Juego	**Posición**	**Representación**	**Importancia**	**Discreto/Continuo**	**Diegético**
Apex legends	Abajo Izquierda	Icono + Numero	Muy poca	Discreto	-
Alien Rage	-	-	-	-	-
CoD bo3 (MP) (DOM)	-	-	-	-	-
Battefield 1(SP)	-	-	-	-	-
Bioshock	-	-	-	-	-
Doom	-	-	-	-	-
Far cry 4	Abajo izquierda	Icono + Numero	Importante	Discreto	-
Borderlands	--	-	-	-	-
Deep Rock Galatics	-	-	-	-	-
Destiny 2	-	-	-	-	-
Deus Ex. Humn	-	-	-	-	-
CSGO	-	-	-	-	-
Far Cry 3	Abajo izquierda	Icono + Numero	Muy poca	Discreto	-
Metroid Prime	-	-	-	-	-
Dead Space 2	-	-	-	-	-

Figura 10.19: Estudio de las pociones en interfaces de videojuegos FPS

	Premio-Racha de puntos				
Juego	**Posición**	**Representación**	**Importancia**	**Discreto/Continuo**	**Diegético**
Apex legends	-	-	-	-	-
Alien Rage	Arriba Izquierda	Numero	Media	Discreto	-
CoD bo3 (MP) (DOM)	Abajo derecha	Icono	Media	-	-
Battefield 1(SP)	-	-	-	-	-
Bioshock					
Doom	Arriba Derecha	Circulo	Muy poca	-	-
Far cry 4	-	-	-	-	-
Borderlands	-	-	-	-	--
Deep Rock Galatics	Arriba Derecha	Numero	Muy poca	Discreto	-
Destiny	-	-	-	-	-
Deus Ex. Humn	-	-	-	-	-
CSGO	-	-	-	-	-
Far Cry 3	-	-	-	-	-
Metroid Prime	-	-	-	-	-
Dead Space 2	-	-	-	-	-

Figura 10.20: Estudio de los premios en interfaces de videojuegos FPS

	Tiempo				
Juego	**Posición**	**Representación**	**Importancia**	**Discreto/Continuo**	**Diegético**
Apex legends	Arriba Izquierda	Número	Muy poca	Discreto	-
Alien Rage	-	-	-	-	-
CoD bo3 (MP) (DOM)	-	-	-	-	-
Battefield 1(SP)	-	-	-	-	-
Bioshock	-	-	-	-	-
Doom	-	-	-	-	-
Far cry 4	-	-	-	-	-
Borderlands	-	-	-	-	-
Deep Rock Galatics	-	-	-	-	-
Destiny 2	-	-	-	-	-
Deus Ex. Humn	-	-	-	-	-
CSGO	Arriba Centro	Número	Poco importante	Discreto	-
Far Cry 3	-	-	-	-	-
Metroid Prime	-	-	-	-	-
Dead Space 2	-	-	-	-	-

Figura 10.21: Estudio del tiempo en interfaces de videojuegos FPS

	Vida				
Juego	**Posición**	**Representación**	**Importancia**	**Discreto/Continuo**	**Diegético**
Apex legends	Abajo Izquierda	Barra	Importante	Continuo	-
Alien Rage	-	-	-	-	-
CoD bo3 (MP) (DOM)	-	-	-	-	-
Battefield 1(SP)	Abajo Derecha	Barra	Muy poca	Continuo	-
Bioshock	Arriba izquierda	Barra	Importante	Continuo	-
Doom	Abajo Izquierda	Barra	Medio	Discreto	-
Far cry 4	Abajo Izquierda	Barra	Medio	Continuo	-
Borderlands	Abajo Izquierda	Barra	Media	Continuo	-
Deep Rock Galatics	Abajo Izquierda	Barra	Importante	Continuo	-
Destiny 2	Abajo Izquierda	Barra	Muy poca	Continuo	-
Deus Ex. Humn	Arriba izquierda	Barra + numero	Muy poca	Discreto	-
CSGO	Abajo Izquierda	Número + Barra	Media	Discreto	-
Far Cry 3	-	-	-	-	-
Metroid Prime	Arriba Medio	Barra	Poca importancia	Discreto	Dentro del casco del personaje
Dead Space 2	-	Barra	Importante	Discreto	La barra que tiene en la espalda el personaje indica la vida que tiene.

Figura 10.22: Estudio de la vida en interfaces de videojuegos FPS