DISEÑO DE VIDEOJUEGOS

PROYECTO EDITORIAL:
TECNOLOGÍAS DIGITALES

Asesor editorial:
Juan Pablo Ordóñez Ortega

DISEÑO DE VIDEOJUEGOS

Juan P. Ordóñez

EDITORIAL
SINTESIS

Consulte nuestra página web: **www.sintesis.com**
En ella encontrará el catálogo completo y comentado

© EDITORIAL SÍNTESIS, S. A.
Vallehermoso, 34. 28015 Madrid
Teléfono: 91 593 20 98
www.sintesis.com

ISBN: 978-84-9171-209-1
Depósito Legal: M-23.372-2018

Impreso en España. Printed in Spain

Índice

3

La experiencia de usuario

3.1. La experiencia de usuario y las mecánicas de juego

Como hemos visto antes, el diseño implica pensar en el comportamiento del usuario, en cómo las cosas se van a utilizar, para qué, cómo, dónde, cuánto, además de otras variables. Este conjunto de elementos es el que conocemos como *experiencia de usuario* o, en inglés, *user experience* (lo podemos encontrar representado por sus siglas UX). La experiencia de un usuario es una variable difícil de medir. Está compuesta por numerosos factores y elementos relacionados con la interacción del usuario dentro del sistema (nuestro juego), y en ella intervienen otros elementos tanto a nivel *hardware* como de *software,* sin olvidar aquello que ocurre en la mente del jugador cuando procesa toda la información que ponemos a su disposición.

A la hora de diseñar un juego, Hunicke, Zubek y Leblanc proponen el *framework MDA* (*mechanics-dynamics-aesthetics*), en el que se tienen en cuenta los siguientes tres elementos:

- *Mecánicas,* como componentes de interacción del juego, cada acción que el usuario puede realizar en él y el conjunto de algoritmos y estructuras de datos que las llevan a cabo, etc.
- *Dinámicas,* como el comportamiento en tiempo de ejecución de las mecánicas actuando en la información de entrada proporcionada por el jugador, e interactuando con otras mecánicas.
- *Estéticas,* como las respuestas emocionales evocadas en el jugador.

Este *framework* supone una buena aproximación para tratar de comprender la complejidad de los sistemas que componen la estructura básica de un videojuego a nivel de diseño de comportamientos y experiencia de usuario. Al margen de esta categorización, si profundizamos en el análisis de los elementos que intervienen en la experiencia de usuario, obtenemos lo siguiente:

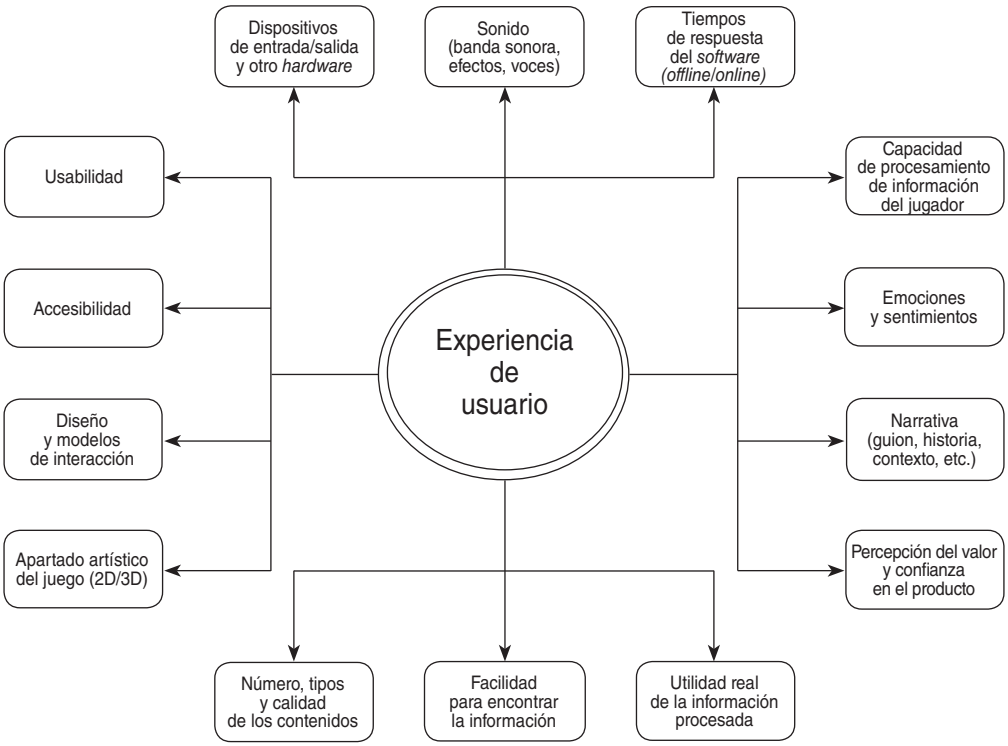

Figura 3.1. Algunos de los factores principales que componen la experiencia de un usuario dentro de un videojuego.

Una mecánica de juego no es más que uno de los múltiples elementos que componen los complejos sistemas que forman el videojuego, y que forman parte de la responsabilidad del diseñador de juego. Además, su desarrollo implica el trabajo de todos los departamentos que componen el equipo.

Para definir de forma visual qué es una mecánica, tenemos que hablar de otros dos elementos: reglas y dinámicas. Si bien es cierto que la parte más importante, al menos desde el punto de vista interno (del desarrollador), es la mecánica, las reglas de juego y las dinámicas están íntimamente ligadas a la primera, de forma que entre

las tres componen la verdadera pieza que afecta al comportamiento del jugador: la jugabilidad.

Como resumen, podríamos decir que:

- Una *regla de juego* es una sentencia atómica, que define con un alto grado de detalle una condición o estado dentro del juego, así como el comportamiento asociado o las consecuencias del mismo. Ejemplo: en un juego de conducción, si un coche no tiene gasolina, no puede arrancar ni moverse.
- Una *mecánica de juego* es un agente, un objeto, un elemento o un conjunto de varios de ellos, así como las relaciones entre estos, dentro del juego. Especifican qué hay, cómo se comporta y las posibles formas de interacción del jugador con el mundo de juego. Son como una "caja negra" para el jugador (Koster, 2011).
- Una *dinámica de juego* es el comportamiento emergente que surge de la jugabilidad definida por la mecánica o mecánicas de juego.

Podemos ver cómo están relacionadas estas tres piezas entre sí a través del siguiente esquema:

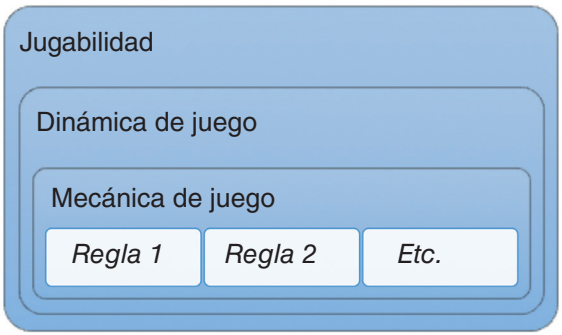

Figura 3.2. Composición de la jugabilidad y sus diferentes partes (dinámicas, mecánicas y reglas de juego) relacionadas entre sí.

La jugabilidad, pues, está compuesta por todos los mecanismos de interacción creados por el equipo de desarrollo para el usuario. Dichos mecanismos (una vez ordenados en el tiempo, y tras integrar la parte narrativa, visual y sonora, y el uso de los dispositivos de entrada y los de salida), unidos a la percepción y el procesamiento de la información que percibe el jugador, sí como la respuesta por parte de este al sistema y otros elementos, componen la experiencia de usuario. Cuando el jugador evalúa la calidad de la experiencia con nuestro producto, en realidad nos muestra el resultado

de su percepción e interacción bajo un complejo sistema formado por un gran número de elementos, y cuyo resultado puede ser positivo, negativo o neutro:

- *Positivo:* se genera *engagement,* también llamado *implicación del jugador* (Wikipedia, 2017) o adicción sana; el jugador obtiene una respuesta por parte del sistema lo suficientemente gratificante como para sentirse cómodo dentro del juego, mientras continúa explorando las distintas interacciones que tienen lugar dentro del juego. Esta adicción sana mantiene al jugador dentro de la historia, del sistema y del contexto. El mundo de juego, entonces, ya no forma parte de un entorno virtual, sino que el usuario lo percibe como si él mismo estuviera dentro. Se produce una sinergia entre el avatar, el entorno y el propio jugador, que resultará en una conexión incluso a nivel emocional en la que las reacciones del mundo de juego a las acciones del jugador, y viceversa, tendrán un efecto real en el cerebro de este.
- *Negativo:* el jugador genera rechazo hacia el juego, bien por una parte del mismo (no le gusta la historia; los sistemas de interacción no responden correctamente o, al menos, no como el usuario espera; el aspecto visual o sonoro no resultan de su agrado, etc.), o bien por un conjunto de varios factores simultáneamente. Este rechazo puede desembocar, si tiene lugar de forma continuada, en frustración. A dicha frustración debemos añadir también el factor de la dificultad; si esta es excesivamente alta, y durante demasiado tiempo, saturará al jugador. Y, si esto ocurre, también se generará un cierto rechazo, en este caso, por la pérdida de la sensación de control del sistema. Dicho de otro modo, el usuario siente que, haga lo que haga, no puede superar las diferentes situaciones a las que se le pide que se enfrente, dentro del juego.
- *Neutro:* el jugador siente cierta indiferencia al juego. Es posible que no hayamos logrado calarle emocionalmente a través de la historia, o que las mecánicas y la disposición en el espacio y el tiempo de nuestro mundo de juego le hayan resultado repetitivas, entre otros posibles factores. El caso es que, como consecuencia, es muy probable que ni siquiera lo llegue a terminar. En la práctica, muchos jugadores no llegan ni a completar la mitad del juego cuando este les genera indiferencia.

Hemos planteado tres posibles tipos de reacciones por parte del jugador al juego, pero la realidad es que raramente resultan tan puras. Lo habitual es que las respuestas de los jugadores al juego dependan de varios factores, que hacen que las reacciones sean híbridas e, incluso, que vayan cambiando a lo largo del juego. Este tipo de factores pueden ser:

- *Dificultad:* la cantidad de daño que provocan los enemigos sobre el avatar, la cantidad de daño que pueden asumir al recibir ataques por parte del avatar,

la velocidad de movimiento, la frecuencia de los ataques, etc. Estos valores, además de la precisión que pedimos al jugador para moverse por el escenario, y muchos otros más, componen la dificultad del juego. La percepción de la dificultad es relativa para el jugador; normalmente depende de muchos factores, tales como la experiencia del usuario en juegos del mismo tipo, la habilidad innata del jugador al interactuar dentro del sistema, las circunstancias que rodean al usuario en el momento en que juega (p. ej., si está cansado, tiene ruido alrededor, etc.) u otras circunstancias (todo lo que ocurre en el juego y alrededor del usuario influye en la percepción de la dificultad del juego). Por eso, muchos juegos permiten la selección de dificultad basada en varios niveles, acorde a la experiencia del jugador o al nivel de reto al que se quiera enfrentar.

- El *tiempo* que puede dedicar a jugar, la cantidad de horas invertida en el juego hasta el momento,
- Nivel de *estrés o exigencia:* está relacionado directamente con la dificultad. El nivel de exigencia que planteamos al jugador depende del nivel de estrés al que le sometemos. Sin embargo, la falta de exigencia al jugador, por ejemplo, cuando tiene que andar durante mucho tiempo por un mundo enorme sin tener que hacer nada, se suele materializar en un aburrimiento creciente que, de repetirse a menudo, puede conllevar el abandono del juego.
- Presencia y duración de las *situaciones y tiempos de reposo,* respectivamente. Define la presión que ejercemos sobre el jugador, así como los momentos en los que le dejamos descansar, y cómo ambas situaciones están equilibradas a lo largo del tiempo de juego.
- Diseño del proceso de *tutorización* del jugador, teniendo en cuenta el binomio aprende-ejecuta, que veremos más adelante.

Estos son solo algunos de los factores más habituales que influyen en la experiencia que el usuario tiene o siente mientras juega. Resulta muy complejo, y lejos del objetivo de este libro, analizar todos los factores que influyen en la percepción del jugador de un videojuego. Pero es importante que el lector sea consciente de su existencia, de modo que cada pequeña decisión que tome a la hora de estructurar y desarrollar la experiencia de juego esté lo más pensada posible, ya que cada una de ellas influyen en la percepción del jugador.

Desarrollar un videojuego que mantenga al jugador bajo los parámetros y valores correctos de ansiedad, adicción sana, estrés, aprendizaje, relajación, inquietud, etc., es una tarea compleja que requiere práctica. Por ello, además de diseñar el videojuego e implementarlo, es responsabilidad directa del diseñador equilibrar el juego constantemente, ajustando cualquier interacción posible, la información que hay en todo momento en el mundo de juego, la duración de todos los eventos, etc. Esta es una de

las principales razones por las que el diseñador de juego está presente durante todo el tiempo de desarrollo, trabajando no solo en la creación de nuevas mecánicas y sistemas, sino también corrigiendo y ajustando los que son implementados hasta el último día del proyecto.

Al depender la experiencia de juego en gran medida de cada jugador (factores culturales, sociales, personales, capacidad intelectual y de retención de información, experiencia con el mismo tipo de juego, con *hardware* similar, etc.), resulta imposible crear un videojuego cuya experiencia sea perfecta para todos los jugadores que lo utilicen. Además, al tratarse de sistemas complejos aquellos creados en los videojuegos actuales, no es posible abarcar y combinar todas las posibles opciones de interacción, tiempos de respuesta, formas de jugar, etc., incluso a la hora de realizar pruebas técnicas para encontrar fallos técnicos; si hoy en día es imposible publicar un juego con total certeza de que nunca dará un fallo a nivel técnico, mucho menos lo es asegurarlo a nivel de experiencia de usuario. Lo que sí podemos hacer como diseñadores de juego es diseñar, probar y equilibrar y, siempre que sea posible y las circunstancias de la empresa lo permitan, tratar de llevar a cabo sesiones de pruebas con *focus groups* o usuarios reales. Dichas pruebas nos ayudarán a medir con gente real, objetiva (no contaminada por formar parte del equipo de desarrollo, que conoce los detalles del juego así como el tipo de resultado deseado en dichas pruebas y, por tanto, que puede ofrecer un resultado real y útil), cuál es la forma de interactuar y de relacionarse con todas las variables del juego. Esto, que puede coincidir o no con nuestras estimaciones como diseñadores (la realidad es que nunca suele coincidir, al menos no totalmente), ayuda enormemente a dirigir nuestros esfuerzos para modificar la experiencia de juego y lo sistemas que la componen, acercándonos a nuestro objetivo de crear un videojuego que funcione correctamente, se entienda y guste al público al que está destinado.

3.2. *Core loop*

Con la proliferación de los videojuegos para *smartphone* y el abaratamiento de las herramientas de desarrollo de videojuegos, durante los últimos años, las tendencias en el desarrollo de modelos de juego se han ido enriqueciendo con nuevas experiencias y formas de jugar. Aunque esto no es exclusivo de los teléfonos inteligentes, dada la popularidad de esta plataforma, la penetración en el mercado de juegos rápidos y basados en partidas cortas ha ido proliferando rápidamente. Cuando la tendencia natural original del videojuego ha sido tradicionalmente tratar de mantener al usuario enganchado el mayor tiempo posible, en los últimos años se ha explotado un nuevo modelo: que el jugador permanezca jugando durante cortos periodos de tiempo (lo que se conoce comúnmente como partidas cortas), pero que juegue varias veces al día. Dicho de otro modo, se ha tratado de potenciar la recurrencia del jugador.

A la hora de diseñar la experiencia del jugador, no solo se tiene en cuenta el flujo de acciones secuenciales que tienen lugar durante el tiempo de uso; también se definen los periodos de cada partida, así como los tiempos de reposo o de espera, según el caso, entre ellas. Del mismo modo, se debe tener en cuenta y calcular el tipo de elementos que se consumen (p. ej., energía o combustible) y los que se obtienen (oro, gemas, etc.). El *core loop* (Kim, 2014) contempla todos estos elementos y define, de forma clara, cómo se relacionan con los tiempos de juego y espera del jugador.

3.3. Definiendo una mecánica de juego

A la hora de diseñar una mecánica de juego, son muchas las aproximaciones que se pueden llevar a cabo. Cada libro, autor, empresa, equipo de desarrollo, etc., tienen sus formas de hacerlo. Sin embargo, hay ciertos puntos importantes, sobre todo a la hora de comenzar a crear una mecánica nueva, que debemos tener en cuenta. Estos puntos sirven para establecer el foco principal y marcar el camino que debemos seguir para que la mecánica cumpla su función y forme parte del sistema y los métodos de interacción que componen el juego, haciéndolo de forma equilibrada y bajo nuestro control.

Antes hemos visto la diferencia entre regla de juego, mecánica de juego y dinámica de juego. Estos tres elementos son los principales núcleos de la experiencia de juego, aunque no los únicos. La combinación entre ellos, la correlación que se establece, genera una serie de respuestas en el cerebro del jugador, ya que gracias a dichos mecanismos el usuario es capaz de interactuar plenamente con nuestro juego (o, al menos, en la medida en que se ha diseñado el juego). Esta interacción, sin embargo, no está totalmente libre de efectos no deseados; utilizar una tecnología tan compleja como la que permite crear un videojuego (como cualquier otra que requiere de programación), puede (y suele) provocar todo tipo de comportamientos no deseados. Estos comportamientos ocurren cuando se da una situación no prevista, o se acumulan determinados datos de una forma concreta, provocando un resultado no previsto a la hora de diseñar el juego. En videojuegos y *software* en general, esto recibe el nombre de *bug*. Pero no siempre se trata de problemas relacionados con aspectos técnicos. A veces, una falta de definición en las mecánicas de juego o en cualquier otro factor en la creación del mundo digital puede provocar un efecto no deseado. En la generación aleatoria o procedural de escenarios o personajes, por ejemplo, suelen aparecer a menudo este tipo de efectos. Para controlarlos, durante la fase de diseño, no solamente debemos detallar cómo queremos que se comporten todos los componentes de nuestro proyecto, sino también establecer una serie de reglas que controlen los casos no deseados. Lo veremos a continuación con un ejemplo: supongamos que estamos diseñando un juego de tipo RPG *(role playing game)* parecido a *Diablo 3* en el que los enemigos se crean de forma totalmente automatizada. Cada enemigo se compone internamente de

cuatro variables: *daño de ataque, resistencia al daño del jugador* (la resistencia actúa como un escudo, bloqueando parte del daño recibido por el jugador)*, puntos de vida y velocidad de movimiento.* Supondremos también que el sistema cuenta con 10 puntos para repartir entre las distintas variables anteriormente mencionadas. Tendremos en cuenta las siguientes reglas:

- Si el personaje no tiene puntos de vida (el valor de dicha variable es igual a cero), se considera eliminado, y se desactivará, haciéndolo desaparecer del juego.
- Si el personaje no tiene puntos de daño (el valor de dicha variable es igual a cero), no podrá quitar puntos de vida al personaje controlado por el jugador.
- Si el personaje no tiene puntos de resistencia (el valor de dicha variable es igual a cero), absorberá todo el daño causado por el arma que lleve equipada el personaje controlado por el jugador.
- Si el personaje no tiene puntos de movimiento (el valor de dicha variable es igual a cero), no se podrá mover.
- Siempre que se pueda mover, el enemigo perseguirá al jugador y le golpeará.

En este punto, podría ocurrir que, durante la generación automatizada de un enemigo, el sistema repartiera todos los puntos a una única variable, quedando las otras tres a cero. Veamos qué tipo de consecuencias podría tener en cada uno de los casos:

– Asignación de todos los puntos a la variable "daño de ataque":

  - Daño = 10.
  - Resistencia = 0.
  - Puntos de vida = 0.
  - Velocidad de movimiento = 0.

  Probablemente el personaje resultaría demasiado poderoso, causando un daño desmesurado al golpear al jugador. Además, el personaje no tendría ningún punto de defensa, siendo fácil de eliminar; pero como no tendría puntos de vida, probablemente el sistema lo contaría como eliminado desde el mismo momento en que este personaje es generado. Incluso, si nos fijamos, el personaje no se podría mover, ya que la velocidad de movimiento sería igual a cero.

– Asignación de todos los puntos a la variable "resistencia al daño del jugador":

  - Daño = 0.
  - Resistencia = 10.

- Puntos de vida = 0.
- Velocidad de movimiento = 0.

Como resultado de la asignación de todos los puntos a la resistencia, el jugador se enfrentaría a un enemigo tremendamente difícil de eliminar pero, al no poder moverse (velocidad de movimiento igual a cero), no poder hacer daño (puntos de vida igual a cero) y no tener puntos de vida, resultaría completamente inútil.

– Asignación de todos los puntos a la variable "puntos de vida":

  - Daño = 0.
  - Resistencia = 0.
  - Puntos de vida = 10.
  - Velocidad de movimiento = 0.

  En este caso, el efecto sería similar a tener un mueble que no ataca, no resiste ni un solo golpe del jugador, y tampoco se mueve.

– Asignación de todos los puntos a la variable "velocidad de movimiento":

  - Daño = 0.
  - Resistencia = 0.
  - Puntos de vida = 0.
  - Velocidad de movimiento = 10.

  Este caso puede resultar incluso simpático, puesto que a efectos prácticos el enemigo no podría causar daño, sería considerablemente débil al asumir cualquier impacto plenamente, sin resistencia a los ataques del jugador; no tendría puntos de vida, pero se desplazaría siguiendo al avatar, como si fuera una mascota.

Como diseñadores de juego, ¿cómo podemos evitar este tipo de situaciones? Tal como hemos visto antes, la respuesta más obvia pasa por establecer un conjunto de reglas que controlen los posibles casos no deseados. En el ejemplo visto antes, la respuesta es tan sencilla como definir reglas como la siguiente:

- Todos los enemigos deben tener al menos 1 punto de vida. Así, eliminamos la posibilidad de que aparezcan y el propio sistema los desactive, puesto que el efecto es el mismo que si el jugador hubiera atacado y reducido todos sus puntos de vida a cero, tras sucesivos ataques.

Si nos fijamos bien, con esta regla también eliminamos la posibilidad de que todos los puntos sean asignados a una única variable, siendo este un caso que también tenemos que controlar en el ejemplo actual. No obstante, podría darse el caso de que, cumpliendo el requisito de que no todos los puntos se agreguen a una única variable, existan otras que se queden sin puntos. El lector se puede preguntar entonces si otros casos similares podrían ser viables, como, por ejemplo, que una o dos variables se quedaran sin puntos. Si analizamos lo que hemos visto hasta ahora, mantenemos bajo control el caso de que una variable contenga todos los puntos; esto ya no puede ocurrir, por lo que una o varias variables más contendrán puntos también. Ahora bien, podría darse el caso de que todos los puntos se repartieran entre dos de las variables, dejando las otras dos sin puntos. Del mismo modo, podríamos tener el caso de que solamente una variable se quedara sin puntos. ¿Cómo afectaría esto a la generación del enemigo? La respuesta es sencilla *a priori:* depende de qué variables queden sin puntos y el comportamiento que queramos que tenga cada tipo de enemigo generado.

Teniendo en cuenta las implicaciones en los comportamientos de los enemigos en función de la cantidad de puntos en sus variables, analizamos cómo podríamos utilizarlo en nuestro sencillo videojuego. Para ello, uno de los enfoques más prácticos que podemos aplicar es hacernos preguntas como las descritas a continuación:

- *¿Podemos tener un enemigo que no tenga puntos de resistencia?* La respuesta más rápida es que sí. De hecho, si el sistema de juego no lo necesita (por los modificadores que se aplican al daño recibido, objetos de tipo armadura si los hubiera, etc.), siempre es mejor trabajar con los sistemas más sencillos posibles. Si podemos aumentar el número de puntos de vida del enemigo o bajar los puntos de daño del avatar en lugar de introducir una nueva variable, el sistema será mucho más fácil de mantener bajo control. Incluso podremos hacer simulaciones de nuestro sistema si fuera necesario, lo que es mucho más fácil de enfocar que si contamos con un sistema mucho más complejo. Un enemigo que no tenga puntos de resistencia sencillamente absorberá todos los puntos resultantes del ataque del avatar.
- *¿Es viable en nuestro ejemplo tener un personaje sin puntos de movimiento?* Dependiendo del tipo de personaje que queramos definir, puede ser perfectamente viable contar con un personaje cuyo valor en los puntos de movimiento sea igual a cero. Sencillamente, se trataría de un enemigo estático, que no se desplazaría por el escenario ni perseguiría al jugador. Solamente le atacaría si el avatar pasase lo suficientemente cerca de él.
- *¿Podemos tener un personaje sin puntos de ataque?* Técnicamente, si se diera este caso, este tipo de enemigo no supondría un peligro real para nuestro avatar, salvo que definiéramos una rutina por la cual estorbase al jugador y

este pudiera caer por algún precipicio o algún caso similar. Pero en nuestro ejemplo de comportamiento sencillo, y exceptuando el caso en el que a nivel narrativo el personaje tuviera algún valor como enemigo (aportando una percepción de peligro en el jugador, aunque este no fuera real), no resultaría práctico tener un enemigo con cero puntos en el ataque.

- *¿Podemos tener un personaje sin puntos de vida?* Como ya hemos explicado antes, esta sería la opción menos viable de todas. Sencillamente, el sistema eliminaría el enemigo nada más ser generado, puesto que cumpliría la condición de desaparecer cuando sus puntos de vida llegaran a cero.

Como hemos visto, hay casos en los que podemos asumir un valor de cero puntos en ciertas variables y otros no. Por ello, lo correcto sería definir una nueva regla en la generación de enemigos, con el objetivo de controlar los dos casos más delicados que acabamos de ver:

- Un enemigo debe tener, al menos, 1 punto de vida.
- Un enemigo debe tener, al menos, 1 punto de ataque.

Teniendo en cuenta que en las cuatro variables del enemigo al menos dos de los puntos estarían asignados a las variables que acabamos de ver, quedarían ocho puntos por repartir entre las demás. Haciendo uso de una nomenclatura más apropiada, tendremos:

vida.Enemigo >= 1 y < 8

De este modo, con una sencilla línea definimos los valores máximo y mínimo posibles de la variable "puntos de vida" del enemigo, y acotamos los parámetros de generación aleatoria de dichos puntos. Para las demás variables que hemos definido procederíamos igual, teniendo en cuenta que el total de la suma de las variables debe ser siempre como máximo igual a 10 puntos.

Hemos visto cómo plantear el enfoque a la hora de definir las variables que componen internamente a un personaje de tipo enemigo sencillo. Aplicando este enfoque, podremos ir aumentando la complejidad del personaje, aunque es recomendable mantener el sistema lo más sencillo posible.

3.4. Picos de intensidad y situaciones o zonas de reposo

Durante todo el tiempo que dura el juego, incluso en cada pequeña partida, la experiencia que recibe el jugador está plagada de situaciones de alta exigencia y de zonas o